Controlling α-entropy with a Neural α-Feature Detector

Ryotaro Kamimura*
Information Science Laboratory,
Tokai University,
1117 Kitakaname Hiratsuka,
Kanagawa 259-12, Japan

A neural α-feature detector is proposed and used to extract a small number of main or essential features in input patterns. Features can be detected by controlling α-entropy for α-feature detectors. The α-entropy is defined by the difference between Rényi entropy and Shannon entropy. The α-entropy controller aims at maximizing information contained in a few important α-feature detectors, while information for all other feature detectors is minimized. Thus, the α-entropy controller can maximize and minimize information, depending on situations. The neural α-feature detector is applied to four problems: an artificial *F-H* detection problem, recognition of six alphabet characters, the feature detection of 26 alphabet characters, and the inference of consonant cluster formation. Experimental results confirm that by controlling α-entropy a small number of principal features can be detected, which can be interpreted intuitively. In addition, it is shown that generalization performance is improved by minimizing α-entropy.

1. Introduction

Many attempts have been made to describe neural learning from the information theoretic points of view [1–4]. In those information theoretic methods, *information*, appropriately defined, has been maximized or minimized. Information maximization has been used to extract and generate features or to interpret explicitly internal representations obtained by learning [3, 5]. By maximizing information, information on input patterns can be condensed in ways that are easy to interpret. On the other hand, information minimization has been used to speed up learning [6] and to improve generalization [7, 8]. By minimizing information, unnecessary information—especially information on noises—is reduced, leading to improved generalization.

Information maximization and minimization have until now been independently studied. However, we can definitely say that simple infor-

*Electronic mail address: ryo@cc.u-tokai.ac.jp.

mation maximization and minimization cannot explain our complex cognitive processes. For example, information is maximized for some parts and minimized for some other parts. Complex processes of information maximization and minimization cooperate in our brain. The neural α-feature detector is introduced in this paper to realize complex processes of information processing. The feature detector is designed to maximize and minimize information as the first approximation to our cognitive process. Information about important features is maximized, and information concerning unimportant features is decreased as much as possible.

To realize this information maximization and minimization, our α-feature detector is mainly composed of an α-entropy controller and α-feature detectors. The α-entropy is used to maximize or minimize information of the α-feature detectors. More exactly, the entropy can be controlled to maximize information only for a few important α-feature detectors. For all other α-feature detectors, information is minimized. The α-feature detectors with higher information are supposed to represent main or essential features in input patterns.

The paper is organized as follows. In section 2, we explain the concept of a neural α-feature detector. More concretely, the section is concerned with the explanation of basic mechanisms of the feature detector and basic components composed of the feature detector and how to realize the concept of the feature detector in an actual network architecture. In section 3, we formulate and explain Shannon entropy, Rényi entropy, and α-entropy. Especially, we explain how α-entropy is used to maximize and minimize information. In section 4, we formulate a neural α-feature detector by borrowing a concept of information and update rules are formulated to control α-entropy. In section 5, we discuss three experimental results: an artificial F and H detection problem, recognition of six alphabet characters, and recognition of 26 alphabet characters. In all experimental results, it is shown that main or essential features in input patterns, which are intuitively interpretable, can be detected. In section 6, the inference of consonant cluster formation is discussed. In this problem, in addition to the feature detection, we show that generalization performance is improved by minimizing α-entropy.

2. Concept of neural α-feature detector

A neural α-feature detector is proposed to detect a small number of important features in input patterns. As shown in Figure 1, a neural α-feature detector is basically composed of an input component, an α-entropy controller, a feature detector (α-feature detectors and a common feature detector), a feature generator (α-feature generators and common feature generators), and an output component.

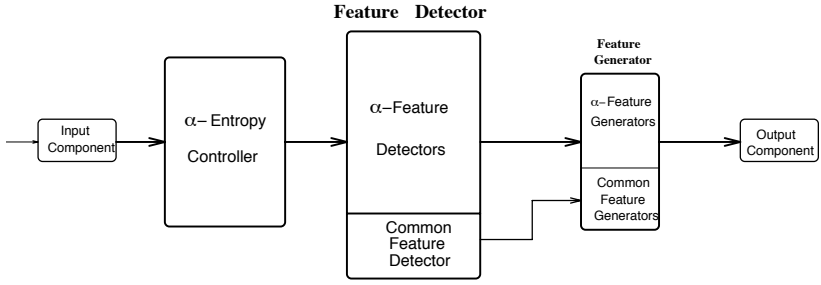


Figure 1. Components of a neural α-feature detector: an input component, an α-entropy controller, a feature detector (α-feature detectors and a common feature detector), a feature generator (α-feature generators and common feature generators), and an output component.

Components	Elements	Functions
Feature detector	α-feature detectors	detecting main and essential features
	common feature detector	detecting common or unnecessary features
α-Entropy controller	α-entropy controllers	controlling α-entropy to maximize and minimize information
Feature generator	α-feature generators	generating α-features
	common feature generators	generating common features

Table 1. Summary of major components, elements, and functions of a neural α-feature detector.

Table 1 summarizes the basic components, elements, and functions of the feature detector. The most important component is a feature detector with two elements: α-feature detectors and a common feature detector. The α-feature detectors aim at detecting some important features in input patterns. Feature detection by α-feature detectors is possible by using the α-entropy controller. The controller is used to maximize or minimize information in α-feature detectors. Figure 2 shows a basic mechanism of the α-entropy controller. By controlling the α-entropy, the first and the third (from the top) α-feature detector are forced to have maximum information. On the other hand, information in the other three α-feature detectors is minimized. The α-feature detectors with higher information can be considered to be important feature detectors. To understand what kind of features the feature detector actually extracts, we should generate features from the feature detector. The α-feature generators are used to generate features from α-feature detectors. A common feature detector is devised to detect features common

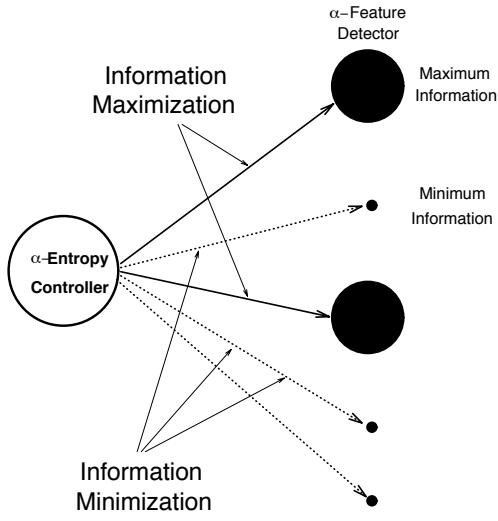


Figure 2. The function of an α-entropy controller. The controller is used to maximize information for the first and the third (from the top) α-feature detectors (hidden units). On the other hand, information for the other detectors is minimized.

to all input patterns. In some cases, the detector is used to detect unnecessary features in input patterns. Common feature generators are used to generate features from the common feature detector. In an output component, the performance of the α-feature detectors and the common feature detector are evaluated by reproducing exactly input patterns.

Then, the next problem is how the concept can be realized in a neural network (NN) architecture. For the neural α-feature detector, we have used an ordinary feed-forward network. Figure 3 shows a possibility of the neural α-feature detector in a feed-forward network. As shown in the figure, input-hidden connections are transformed into α-entropy controllers. Thus, α-entropy controllers are used not only for controlling α-entropy but also for decreasing errors between targets and outputs. However, the main objective of the controllers is to control α-entropy. Hidden units are replaced by α-feature detectors. Bias to output units are transformed into a common feature detector. Hidden-output connections and bias-output connections are α-feature generators and common feature generators respectively. Finally, an output component is composed of usual output units.

3. Shannon, Rényi, and α-entropy

In this section, we explain α-entropy in a general framework of information theory. Let A take a finite number of possible values $a_1, a_2, \ldots, a_R$

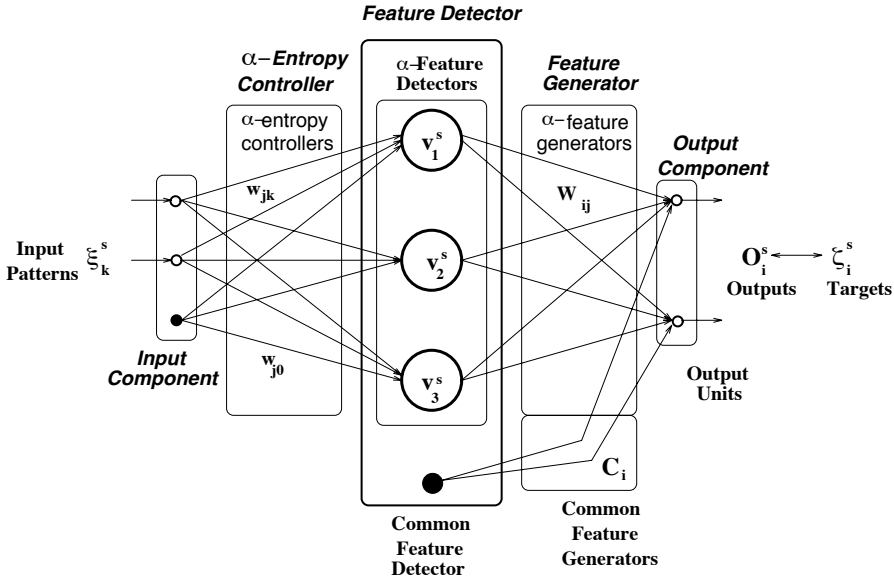


Figure 3. A neural α-feature detector realized by a feed-forward network.

with probabilities $p(a_1), p(a_2), \ldots, p(a_R)$ respectively. Then, the uncertainty $H(A)$ [9–11] of the random variable A is defined by

$$H(A) = -\sum_{r=1}^{R} p(a_r) \log p(a_r). \tag{1}$$

Consider conditional uncertainty after observing another random variable B, taking possible values $b_1, b_2, \ldots, b_S$ with probabilities $p(b_1)$, $p(b_2), \ldots, p(b_S)$ respectively. Conditional uncertainty $H(A \mid B)$ [9–11] can be defined as

$$H(A \mid B) = -\sum_{s=1}^{S} p(b_s) \sum_{r=1}^{R} p(a_r \mid b_s) \log p(a_r \mid b_s). \tag{2}$$

We can easily verify that conditional uncertainty is always less than or equal to initial uncertainty. Information for A obtained by observing B is usually defined as the decrease of this uncertainty [10, 12, 13]

$$\begin{aligned} I(A \mid B) &= H(A) - H(A \mid B) \\ &= -\sum_{r=1}^{R} p(a_r) \log p(a_r) \\ &\quad + \sum_{s=1}^{S} p(b_s) \sum_{r=1}^{R} p(a_r \mid b_s) \log p(a_r \mid b_s). \end{aligned} \tag{3}$$

When prior uncertainty is maximum, that is, a prior probability is equiprobable (1/R), then information is

$$I(A \mid B) = \log R + \sum_{s=1}^{S} p(b_s) \sum_{r=1}^{R} p(a_r \mid b_s) \log p(a_r \mid b_s) \tag{4}$$

where $\log R$ is the maximum uncertainty. Information theoretical methods, developed in NNs, have been concerned with this information maximization or minimization [3, 6–8, 14]. Our neural α-feature detector aims at maximizing and minimizing information, especially, in terms of equation (4).

For information maximization and minimization, we use Rényi entropy [14] of the random variable A, given B

$$H(A \mid B; \alpha) = \frac{1}{1-\alpha} \sum_{s=1}^{S} p(b_s) \log \sum_{r=1}^{R} p^{\alpha}(a_r \mid b_s) \tag{5}$$

where α is the parameter and $\alpha \geq 0$. Note that we may use a simplified expression $H(\alpha)$ instead of $H(A \mid B; \alpha)$. Concerning Shannon and Rényi entropy, the following properties can easily be verified:

$$\begin{aligned} &H(\alpha) \geq H \quad \text{for } 0 \leq \alpha < 1 \\ &H(\alpha) \leq H \quad \text{for } \alpha > 1. \end{aligned} \tag{6}$$

It is easy to prove that Rényi entropy becomes Shannon entropy as $\alpha \to 1$, that is,

$$\lim_{\alpha \to 1} H(\alpha) = H. \tag{7}$$

Then, suppose that the parameter α for Rényi entropy ranges between zero and one, namely, $0 \leq \alpha < 1$, then Rényi entropy is always greater than or equal to Shannon entropy. The difference between Shannon and Rényi entropy

$$D(\alpha) = H(\alpha) - H \tag{8}$$

is referred to as α-entropy. As the parameter α is smaller, uncertainty is larger. When the parameter α is zero, α-entropy is written as

$$D(0) = \log R + \sum_{s=1}^{S} p(b_s) \sum_{r=1}^{R} p(a_r \mid b_s) \log p(a_r \mid b_s) \tag{9}$$

where $\log R$ is the maximum entropy. This means that the α-entropy minimization corresponds to information minimization when the parameter α is zero.

When the parameter α is greater than or equal to zero, this α-entropy minimization can be used to maximize and minimize information. When

information defined in equation (4) is maximized, only one symbol occurs with a probability one, while the probability of the occurrence of all other symbols is zero. On the other hand, when information is minimized, the probability of the occurrence of all symbols is equiprobable. The minimum α-entropy is attained exactly when the information defined in equation (4) is maximized or minimized.

Let us further explain the actual meaning of α-entropy minimization by one of the simplest examples. Suppose a binary symmetric information channel composed of two symbols {0,1} [9]. Then, p represents the probability of receiving a symbol 0 for a transmitted symbol 1 and $\bar{p}$ denotes the probability of receiving a symbol 1 for transmitted symbol 1. In this case, α-entropy is written as

$$D(\alpha) = \sum_{s=1}^{S} p(b_s) \left\{ \frac{1}{1-\alpha} \log \sum_{r=1}^{R} p^{\alpha}(a_r \mid b_s) + \sum_{r=1}^{R} p(a_r \mid b_s) \log p(a_r \mid b_s) \right\}$$

$$= \frac{1}{1-\alpha} \log \left\{ p^{\alpha} + \bar{p}^{\alpha} \right\} + p \log p + \bar{p} \log \bar{p}. \tag{10}$$

When the parameter α is zero, the equation is transformed into

$$D(0) = \log 2 + p \log p + \bar{p} \log \bar{p}, \tag{11}$$

which is just the definition of information.

Figure 4(a) shows Rényi and Shannon entropy for different values of the parameter α. As shown in the figure, uncertainty (entropy) is minimum when Shannon entropy is used. Then, uncertainty is increased as the parameter α is decreased from 0.5 to 0.1, and finally the entropy attains the maximum value ($\log 2$) when the parameter α is zero. Figure 4(b) shows information ($\alpha = 0$) in equation (11) and α-entropy for different values of the parameter α. When α-entropy is minimized, a probability should be one, zero (maximum information), or 0.5 (minimum information). In terms of information, when α-entropy is minimized, information is maximized or minimized. Thus, we can say that α-entropy is used to maximize and minimize information, depending on situations.

4. Controlling α-entropy in neural networks

4.1 Minimizing α-entropy

The α-entropy controller is used to control α-entropy in α-feature detectors. Especially, we attempt to minimize α-entropy ($0 \le \alpha < 1$). Since the parameter α is restricted to range between one and zero, this minimization corresponds to information maximization and minimization.

To control α-entropy, we should compute an output from the jth α-feature detector denoted by v_j^s. Let ξ_k^s denote the kth element of the

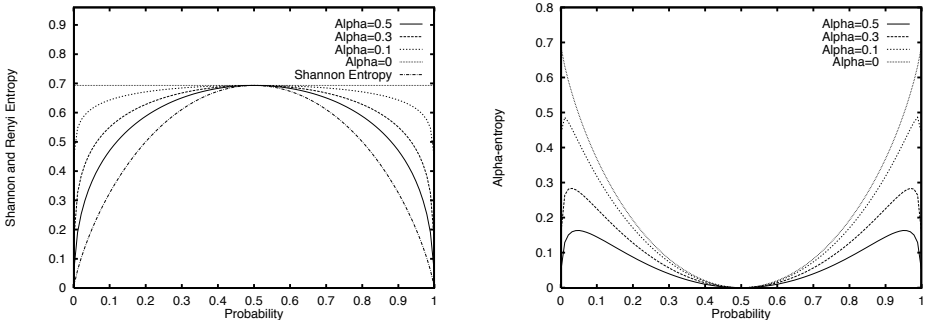


Figure 4. Shannon, Rényi, and α-entropy for four different values of the parameter α as a function of a probability: (a) Shannon and Rényi entropy, (b) α-entropy.

sth input pattern and w_{jk} represent an α-entropy controller from the kth input unit to the jth α-feature detector, then a net input to the jth α-feature detector u_j^s is computed by

$$u_j^s = \sum_{k=0}^{L} w_{jk}\xi_k^s \tag{12}$$

where L is the number of input units, and w_{j0} is the bias to the jth α-feature detector. Then, the jth α-feature detector produces an output

$$v_j^s = f(u_j^s) \tag{13}$$

where f is a sigmoid activation function defined by

$$f(u_j^s) = \frac{1}{1 + \exp(-u_j^s)}. \tag{14}$$

We deal with each α-feature detector separately. Suppose that the probability of input patterns is equiprobable:

$$p(b_s) \approx \frac{1}{S}. \tag{15}$$

The output from the jth α-feature detector can be considered to represent a probability that the jth α-feature detector fires. Then, in formulating α-entropy, the conditional probability $p(a_r|b_s)$ is approximated by the output from the α-feature detector v_j^s:

$$p(a_r|b_s) \approx v_j^s. \tag{16}$$

Then, α-entropy (D_j) for the jth α-feature detector, given the sth input

pattern, can be approximated by

$$D_j(\alpha) = \sum_{s=1}^{S} p(b_s)\left\{\frac{1}{1-\alpha}\log\sum_{r=1}^{R} p^{\alpha}(a_r \mid b_s) + \sum_{r=1}^{R} p(a_r \mid b_s)\log p(a_r \mid b_s)\right\}$$
$$\approx \frac{1}{S}\sum_{s=1}^{S}\left[\frac{1}{1-\alpha}\log\left\{(v_j^s)^{\alpha} + (\bar{v}_j^s)^{\alpha}\right\} + v_j^s \log v_j^s + \bar{v}_j^s \log \bar{v}_j^s\right] \tag{17}$$

where

$$\bar{v_j^s} = 1 - v_j^s. \tag{18}$$

Since all α-feature detectors are treated separately, the total α-entropy (D_{tot}) can be approximated by

$$D_{\text{tot}}(\alpha) \approx \sum_{j=1}^{M}\left[\frac{1}{S}\sum_{s=1}^{S}\left[\frac{1}{1-\alpha}\log\left\{(v_j^s)^{\alpha} + (\bar{v}_j^s)^{\alpha}\right\} + v_j^s \log v_j^s + \bar{v}_j^s \log \bar{v}_j^s\right]\right]. \tag{19}$$

Differentiating the α-entropy function with respect to α-entropy controllers w_{jk}, we have

$$-\beta S\frac{\partial D_{\text{tot}}(\alpha)}{\partial w_{jk}} = -\beta S\sum_{s=1}^{S}\left[u_j^s + \frac{\alpha\left\{(v_j^s)^{\alpha-1} - (\bar{v}_j^s)^{\alpha-1}\right\}}{(1-\alpha)\left\{(v_j^s)^{\alpha} + (\bar{v}_j^s)^{\alpha}\right\}}\right] v_j^s \bar{v}_j^s \xi_k^s \tag{20}$$

where β is the parameter. Thus, update rules for α-entropy controllers are

$$\Delta w_{jk} = -\beta\sum_{s=1}^{S}\left[u_j^s + \frac{\alpha\left\{(v_j^s)^{\alpha-1} - (\bar{v}_j^s)^{\alpha-1}\right\}}{(1-\alpha)\left\{(v_j^s)^{\alpha} + (\bar{v}_j^s)^{\alpha}\right\}}\right] v_j^s \bar{v}_j^s \xi_k^s + \eta\sum_{s}^{S}\left\{\sum_{i}^{N}(\zeta_i^s - O_i^s)W_{ij}\right\} v_j^s \bar{v}_j^s \xi_k^s \tag{21}$$

where η is the learning parameter. Equation (21) was obtained by differentiating α-entropy and the cross entropy defined in equation (27).

4.2 Common feature detector, generators, and output component

The common feature detector is a very special unit, which is always turned on. The detector should capture features common to all input

patterns. Let C_i be a common feature generator into the ith output unit, then an output with only this feature generator is

$$O_i^c = f(C_i). \tag{22}$$

The common feature detector should decrease the difference between targets and outputs only with common feature generators. The difference between targets and outputs can be represented by the cross entropy function [16–18]. Thus, an error function for the common feature detector is defined by

$$G^c = \sum_{i=1}^{N} \left[\frac{1}{S} \sum_{s=1}^{S} \left\{ \zeta_i^s \log \frac{\zeta_i^s}{O_i^c} + \bar{\zeta}_i^s \log \frac{\bar{\zeta}_i^s}{\bar{O}_i^c} \right\} \right] \tag{23}$$

where ζ_i^s is a target to the actual output from the ith output unit, given the sth input pattern, $\bar{\zeta}_i^s = 1 - \zeta_i^s$, $\bar{O}_i^c = 1 - \bar{O}_i^c$, and N is the number of output units. Updating rules are obtained by differentiating a cross entropy function with respect to common output connections,

$$\Delta C_i = -\mu S \frac{\partial G^0}{\partial C_i} = \mu \sum_{s=1}^{S} \left(\zeta_i^s - O_i^c \right) \tag{24}$$

where μ is the parameter.

4.3 Output component and α-feature generators

An output component is used to evaluate the performance of α-feature detectors and the common feature detector. In formulating the output component, we should compute a net input from α-feature detectors. The ith output unit receives

$$h_i^s = \sum_{j=1}^{M} W_{ij} v_j^s \tag{25}$$

where W_{ij} is an α-feature generator from the jth α-feature detector to the ith output unit, and M is the number of α-feature detectors. Then, the ith output unit produces the final output O_i^s:

$$O_i^s = f(h_i^s). \tag{26}$$

We use the cross entropy cost function

$$G = \sum_{i=1}^{N} \left[\frac{1}{S} \sum_{s=1}^{S} \left\{ \zeta_i^s \log \frac{\zeta_i^s}{O_i^s} + \bar{\zeta}_i^s \log \frac{\bar{\zeta}_i^s}{\bar{O}_i^s} \right\} \right] \tag{27}$$

where ζ_i^s is a target for the ith output unit O_i^s. For updating α-entropy generators W_{ij}, we must differentiate this cross entropy function with

respect to α-entropy generators W_{ij} and obtain the usual update rules:

$$\Delta W_{ij} = -\eta S \frac{\partial G}{\partial W_{ij}} = \eta \sum_{s=1}^{S} (\zeta_i^s - O_i^s) v_j^s \tag{28}$$

where η is the learning parameter.

5. Application to character recognition

5.1 *F-H* problem

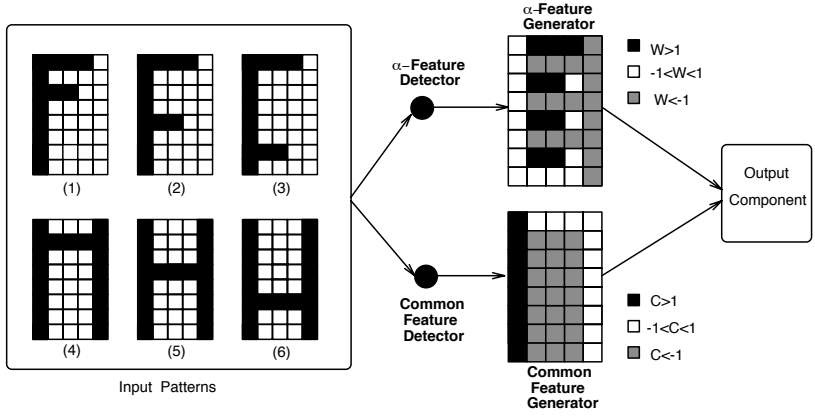
In this problem, we made six simple input patterns transformed from the two alphabet characters *F* and *H*. As shown in Figure 5, six patterns are classified into two groups: *F* and *H*. The objective of this experiment is to show that our α-feature detector explicitly extracts essential features and classifies six patterns into two groups. It is also shown that standard autoencoders always have difficulty in classifying these six simple patterns.

Input patterns are represented in 8×5 bits. Thus, the number of input units, α-feature detectors, and output units were 40, 10, and 40 respectively. For simplicity, the parameter η for the cross entropy function between targets and outputs and μ for the cross entropy with the common feature generators were kept to the small value 0.01 for all experiments. We do this to examine the effect of the parameter β more explicitly. Learning was considered to be finished when absolute errors between targets and outputs were all below 0.1 for all input patterns or the number of epochs was larger than 2000. We use the same parameter values and stopping criteria in the following sections.

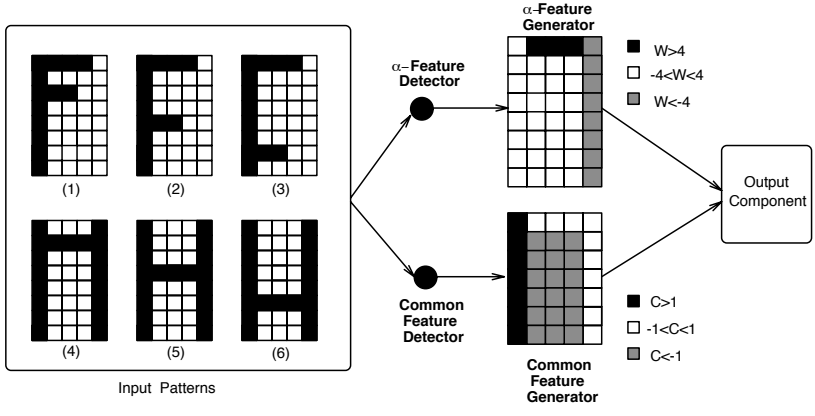
The α-entropy controller aims at increasing or decreasing information, depending on situations. Thus, we should examine information for each α-feature detector, when α-entropy is significantly decreased. Figure 6 shows the values of information for a standard method (a) and for three different values of the parameter α: (b), (c), and (d). Note that in the standard method the parameter β was set to zero, and the ordinary bias to output units are used. Information ($D_j(0)$) for the jth α-feature detector, given the sth input pattern, is computed by

$$D_j(0) = \log 2 + \frac{1}{S} \sum_{s=1}^{S} \left(v_j^s \log v_j^s + \bar{v}_j^s \log \bar{v}_j^s \right). \tag{29}$$

As shown in Figure 6(a), by using a standard backpropagation (BP) method ($\beta = 0$), the values of information fluctuate randomly, and it is very difficult to detect some important α-feature detectors. Then, the parameter α is set to 0.9 (Figure 6(b)). It can be immediately seen that one important α-feature detector (fourth α-feature detector), containing



(a) $|W_{ij}| > 1$



(b) $|W_{ij}| > 4$

Figure 5. An α-feature detector, α-feature generators, a common feature detector, and common feature generators for the *F-H* problem. The parameter α is 0.3. (a) The absolute strength of the generators is greater than one, (b) the absolute strength of the α-feature detectors is greater than four.

almost maximum information, can be detected. When the parameter α is decreased from 0.9 to 0.3 (Figure 6(c)), only the fourth α-feature detector is close to maximum information (log 2), while all other α-feature detectors tend to be close to zero information. Finally, when the parameter α is further decreased from 0.3 to 0.1, a typical state is obtained in which all α-feature detectors tend to have small information. In this case, the α-entropy controller is used only to minimize information.

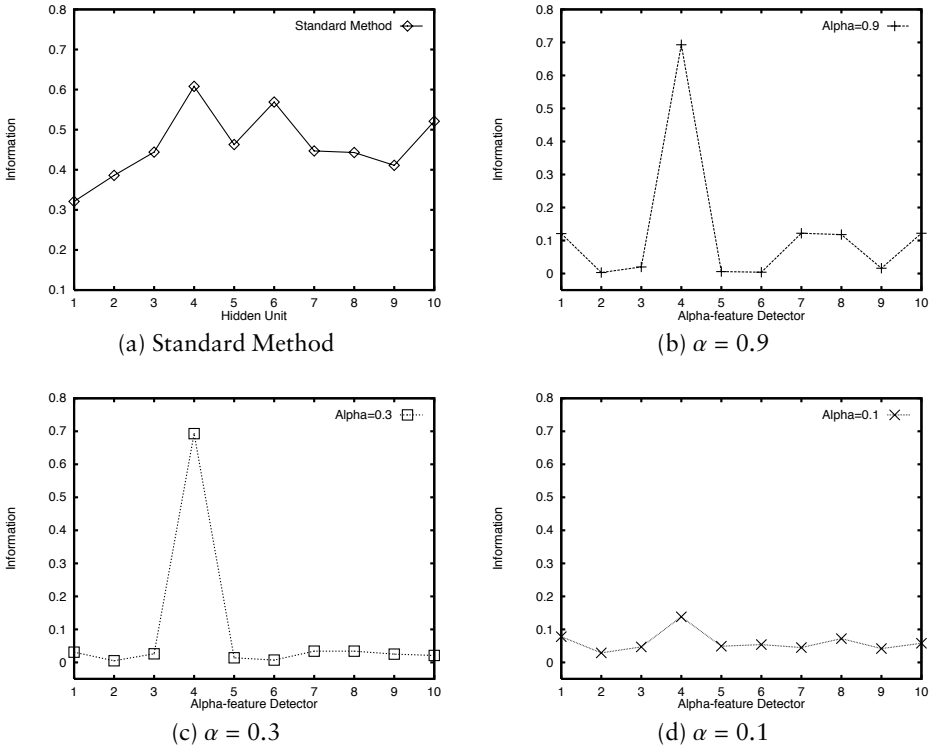
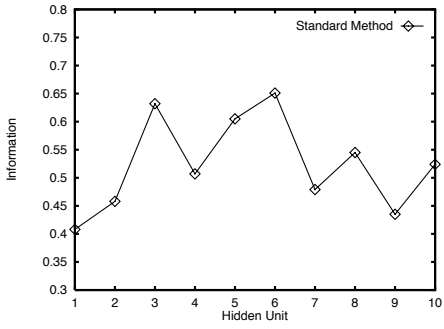
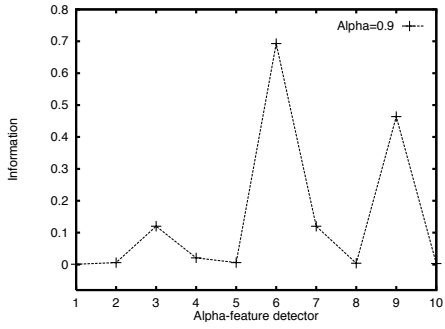


Figure 6. The values of information for each hidden unit and each α-feature detector for the six alphabet characters shown in Figure 5: (a) information for a standard BP method ($\beta = 0$) and (b) α-entropy with $\alpha = 0.9(\beta = 0.2)$.

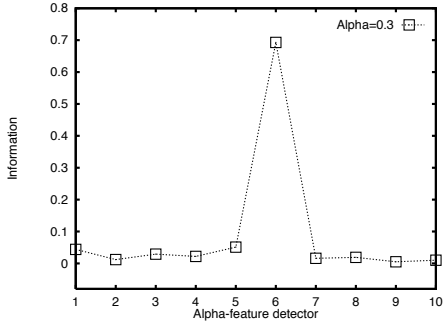
By examining α-feature generators from α-feature detectors, it can be explicitly seen that the α-feature detector with maximum information performs a principal role in feature detection. The α-feature detector can detect a feature which classifies six patterns into two groups and the common feature detector can extract a feature common to all input patterns explicitly. Figure 5 shows the fourth α-feature detector (detector with maximum information in Figure 6(c)), α-feature generators, common feature detector, and common feature generators, when α is 0.3 and α-entropy is sufficiently small. When we examine α-feature generators with absolute values greater than one (Figure 5(a)), it can be seen that all features except a common feature (the leftmost column), detected by the common feature generators, are represented in the feature generators. This naturally means that all the information is compressed into only one α-feature detector. However, when the absolute value is much larger (from one to four, as shown in Figure 5(b)), it can be clearly seen that the α-feature generator extracts a row at the top and a column



(a) Standard Method



(b) $\alpha = 0.9$



(c) $\alpha = 0.3$

Figure 7. Information for 10 hidden units and α-feature detectors for the six alphabet characters shown in Figure 8: (a) information values by a standard method ($\beta = 0$), (b) information by α-entropy with $\alpha = 0.9$ ($\beta = 0.27$), and (c) information with $\alpha = 0.3$, $\beta = 0.04$.

on the extreme right as an α-feature, which are essential factors to discriminate between *F* and *H*. As already mentioned, the common feature detector clearly detects the leftmost column as a common feature. In addition, it can also be seen that parts at the middle are represented in strongly negative connections, because these parts are not important for the distinction of the patterns. These results show that the α-feature detector extracts the main or essential features by which we can classify six input patterns into two groups.

5.2 Feature detection for six alphabet characters

This method is applied to six alphabet characters as shown in Figure 8. First, we examine information for each α-feature detector. Figure 7 shows values of information for all hidden units and all α-feature detectors. Figure 7(a) shows information for 10 hidden units by the standard BP method ($\beta = 0$). Values of information fluctuate randomly, and

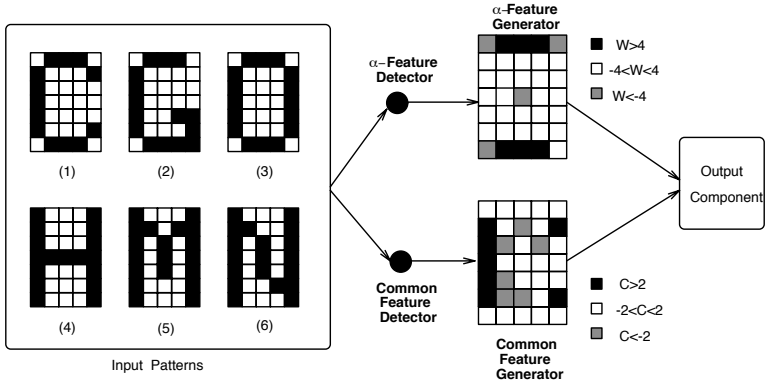


Figure 8. The sixth α-feature detector, α-feature generators, a common feature detector, and common feature generators when the parameter α is 0.3.

it is extremely difficult to extract some important α-feature detectors. Figure 7(b) shows information for 10 α-feature detectors when the parameter α is 0.9. As shown in the figure, two α-feature detectors tend to have relatively higher information. Finally, when the parameter α is decreased from 0.9 to 0.3, only one α-feature detector is close to maximum information, while all other α-feature detectors tend to have small information values, as shown in Figure 7(c).

Figure 8 shows the sixth α-feature detector with maximum information ($\alpha = 0.3$), α-feature generators, a common feature detector, and common feature generators. As shown in Figure 8, an α-feature detector detects rows at the top and at the bottom as important features. In addition, three parts at the corner and a part at the center are also detected as features. Thus, if the α-feature detector is strongly activated, three letters *C*, *G*, and *O* are generated by the feature detector. A common feature detector mainly detects a column on the extreme left as a feature. In addition, some parts at the middle of feature generators are detected as common features, because these parts are not used for input patterns. The feature detection by a neural α-feature detector corresponds to our intuition of six alphabet characters.

5.3 Feature detections for 26 alphabet characters

In this section, 26 alphabet characters shown in Figure 9 are given to our neural α-feature detector. It is shown that salient features, corresponding to our intuition, can be detected by our α-feature detector.

The number of input units, α-feature detectors, and output units were 35, 20, and 35 respectively. The α-entropy was decreased as much as possible. Figure 10 shows information ($D_j(0)$) for 20 hidden units or α-feature detectors. When a standard method ($\beta = 0$) is used, it

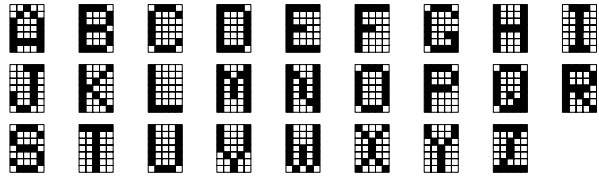
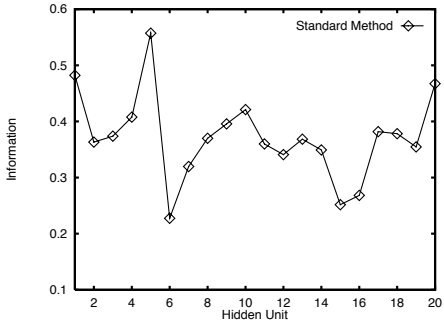
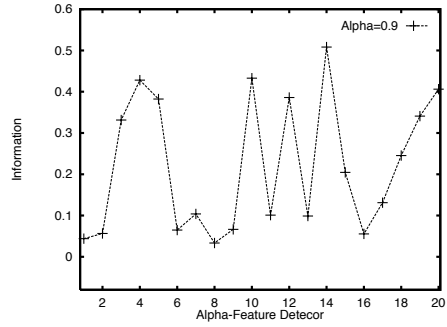


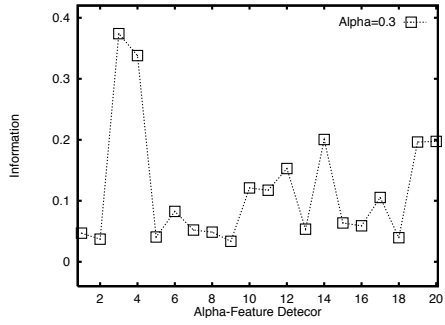
Figure 9. 26 alphabet characters given to a neural α-feature detector.



(a) Standard Method



(b) $\alpha = 0.9$

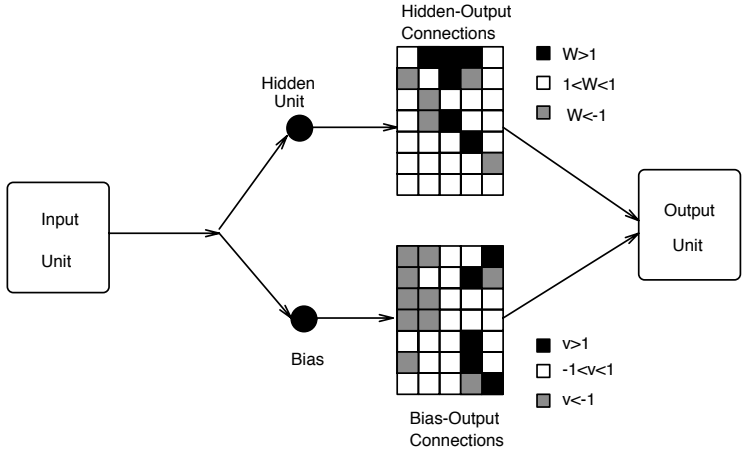


(c) $\alpha = 0.3$

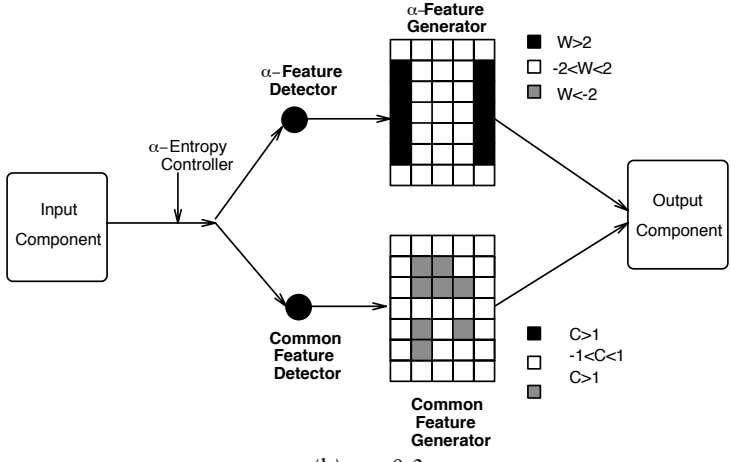
Figure 10. Information for 20 hidden units and α-feature detectors: (a) by a standard method, (b) by α-entropy with $\alpha = 0.9$, $\beta = 0.27$, and (c) with $\alpha = 0.3$, $\beta = 0.04$.

is impossible to detect some important α-feature detectors. When the parameter α is set to 0.9, some characteristics can be seen, but it is still impossible to detect important α-feature detectors. Finally, when the parameter α is decreased from 0.9 to 0.3, only two α-feature detectors with higher information can be seen.

We should also examine what kind of features α-feature detectors extract. Figure 11(b) shows the third α-feature detector with α-feature generators (the highest information detector) and a common feature



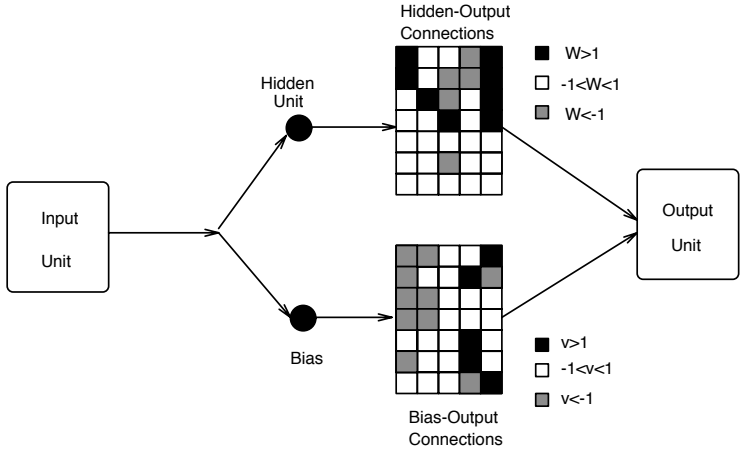
(a) Standard Method



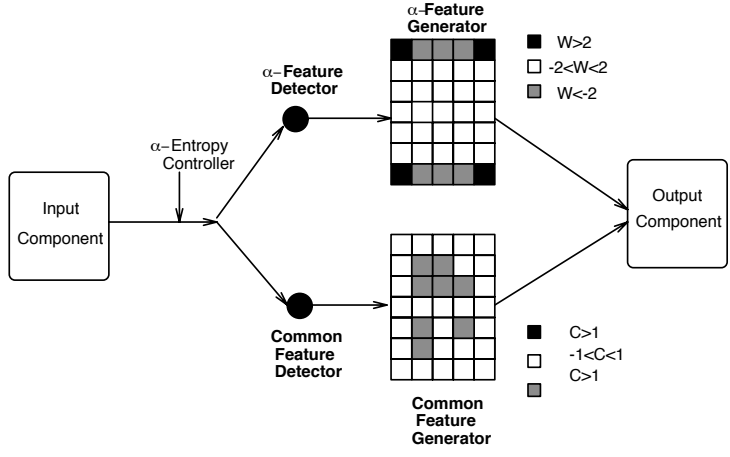
(b) $\alpha = 0.3$

Figure 11. (a) Hidden-output and bias-output connections into a hidden unit with the highest information; (b) the third α-feature detector, α-feature generators, common feature detector, and common feature generators. The parameters α and β were 0.3 and 0.04 respectively.

detector with common feature generators. As clearly shown in the figure, the α-feature detector extracts two columns at both sides as fundamental features. A common feature detector extracts parts not mainly used as features in the 26 alphabet characters. This means that the common feature detector aims at detecting unnecessary parts in input patterns. On the other hand, Figure 11(a) shows hidden-output connections and bias-output connections from the highest information



(a) Standard Method



(b) $\alpha = 0.3$

Figure 12. (a) Hidden-output and bias-output connections with the highest information. (b) The fourth α-feature detector, α-feature generators, common feature detector, and common feature generators. The parameters α and β are 0.3 and 0.04 respectively.

hidden unit by a standard method ($\beta = 0$). The strength of hidden-output and bias-output connections are rather small, and it is impossible to detect some features in the connections.

The second highest information detector is the fourth α-feature detector. As shown in Figure 12(b), the α-feature generator shows the same patterns as those obtained in the previous six-character recognition problem (Figure 8). In addition, we can see that the third and fourth

α-feature detectors are complementary to each other. Both α-feature detectors jointly detect the outmost parts in input patterns. On the other hand, it is also shown that a standard method always has ambiguous patterns in hidden-output and bias-output connections, as can be seen in Figure 12(a).

As already shown in Figure 10(c), the third and fourth α-feature detectors are main α-feature detectors. As information is lower, detected features become more obscure. However, we can still interpret features detected by smaller information detectors.

6. Application to consonant cluster well-formedness

In this section we demonstrate the performance of the neural α-feature detector by applying the method to the inference of the well-formedness of consonant clusters. We think that one of the ultimate objectives for NNs is to describe and explain human intellectual activities. Natural language is surely one of the most important aspects composed of intellectual activities. Thus, it is important to clarify the process of natural language understanding by NNs. However, at the present stage of development, it is impossible to cope with syntactic and semantic aspects of natural languages. We focus now upon phonetic or phonological aspects of natural languages.

The problem is the inference of the well-formedness of consonant clusters. Human beings can perfectly and very easily infer the well-formedness of given words or sentences. However, this simple fact cannot easily be explained by a traditional rules-specification approach. In addition to linguistic aspects governed by perfect rules, many kinds of exceptions are easily pointed out, as is the case with many other aspects of human behavior. Thus, the problem is well suited to NNs, which can cope with exceptional cases by learning [18]. Though the problem discussed in this section is simple, this should be considered to be the first step toward a basic understanding of human natural languages by NNs.

6.1 Inference by sonority

It is known that in natural languages consonants or vowels are not randomly connected with each other but combined in regular or systematic ways to produce actual words or sentences. One of the explanations for consonant cluster formation is the *sonority principle*. According to this principle each consonant has its own sonority value, for an example see Appendix A.1. As the sonority value of a consonant is higher, the consonant should be closer to a vowel. For example, the sonority of a phoneme /p/ is lower than the sonority of /r/. Thus, a consonant cluster such as /pr/ (present) is well-formed. Figure 13(a) shows that a network must be trained to produce *yes*, because a consonant cluster observes the

sonority principle. In the figure, each consonant is represented in five bits using a phonological representation from [18]. The actual representation is shown in Appendix A.2. On the other hand, /rp/ is ill-formed, because the sonority of /r/ is larger than that of /p/. Figure 13(b) shows an ill-formed case in which a network must be trained to produce *no* because of the violation of the sonority principle.

There are many exceptions to this sonority principle. We first take the most familiar case of /sp/ (speak). The sonority of /s/ is lower than that of /p/. Thus, /sp/ is ill-formed according to the sonority principle. However, a consonant cluster /sp/ at the beginning of words is used in the English language. Though many exceptions are known, we suppose for simplicity that the sonority principle is strictly observed in our experiments.

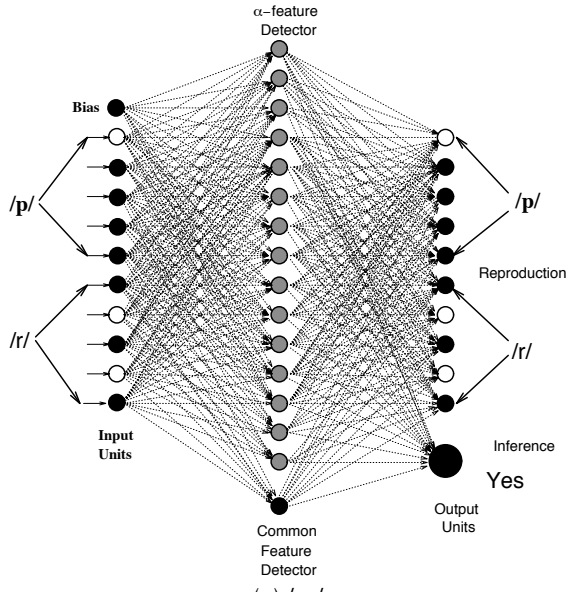
In addition to the inference of the well-formedness of consonant clusters by the sonority principle, networks were trained to exactly reproduce input patterns. As can be seen in Figure 13, output units are divided into two parts: the reproduction part and the inference part. The number of α-feature detectors is 15 in the experiments, which is very redundant. Thus, it is not difficult for networks to reproduce input patterns. Our objective is to examine whether networks can discriminate between the inference and reproduction part, and if they can extract linguistic rules for the inference part.

6.2 Improved generalization performance

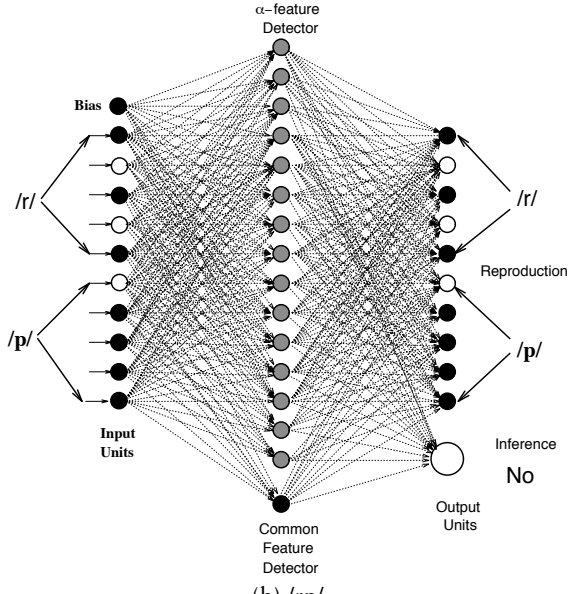
We have shown that the neural α-feature detector can clearly extract important features from input patterns. We think that the explicit extraction of salient features is concerned with improved generalization. In this section, we examine whether generalization performance can be improved by α-entropy minimization.

First, we examined whether the α-entropy minimization is correlated with improved generalization. In experiments, the number of input, hidden, and output units were 10, 15, and 11 respectively, as shown in Figure 13. The number of training, validation, and generalization data is 50. Learning was considered to be finished when the absolute error between targets and outputs is smaller than 0.1 for all input patterns and all output units. In addition, generalization performance was evaluated by using validation and test sets.

Figure 14(a) shows the α-entropy as a function of the parameter β. The parameter α was set to 0.3 as the first approximation because experimental results in the previous sections confirmed better performance with $\alpha = 0.3$. As can be seen in the figure, α-entropy is naturally decreased as the parameter β is increased. Figure 14(b) shows validation, generalization, and training errors as a function of the parameter β. It can be seen in the figure that generalization errors are quickly decreased



(a) /pr/



(b) /rp/

Figure 13. Inference of well-formedness of consonant clusters and reproduction of consonant clusters by NNs. (a) To this network, a well-formed input /pr/ (present) is given. Thus, the network is trained to produce *yes*. (b) Since /rp/ is ill-formed, the network must produce *no*.

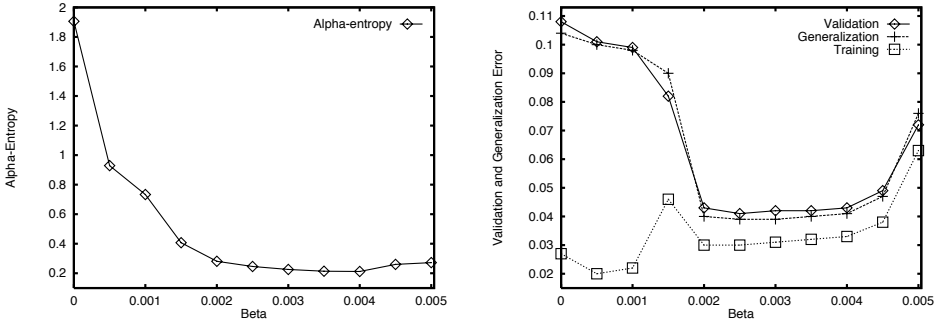


Figure 14. α-entropy, validation, generalization, and training error: (a) α-entropy as a function of the parameter β; (b) validation, generalization, and training error as a function of the parameter β. The parameter α is 0.3.

in direct proportion to the decrease of α-entropy. However, it can also be observed that generalization and validation errors are inversely increased when the parameter β is much larger. This is explained by the fact that the training error is significantly increased in this case. Thus, we can say that generalization performance can be improved in direct proportion to the decrease of α-entropy under the condition that the training error is sufficiently small.

Then, we compared the generalization performance of the neural α-feature detector with the performance of other methods. Table 2 shows generalization comparisons. Average values were averages over 10 runs with 10 different initial conditions. The weight elimination method used was developed in [19] and has a good reputation for improved generalization. In all methods, including the standard BP method ($\beta = 0$), validation sets were used to control learning.

As shown in Table 2, generalization errors in the root mean squared (RMS) error are significantly decreased from 0.090 by a standard BP method to 0.054 by weight decay, and to 0.047 by weight elimination. By using the neural α-feature detector, generalization errors are further decreased. As the parameter α is zero, namely, the simple information minimization is used, the generalization error is decreased to 0.048, which is approximately equivalent to the performance of weight elimination. When the parameter α is further increased to 0.2 or 0.3, the best performance (0.40) can be obtained. Then, when the parameter is further increased, generalization performance is inversely decreased, and the standard deviation is significantly increased. This means that as the parameter α is increased, results are dependent upon initial conditions.

In addition to the performance comparison by the RMS, the error rate was computed for the intuitive interpretation of experimental results. The error rate denotes the number of incorrectly estimated input

Method	Generalization Error			
	RMS		Error Rate	
	Average	Standard Deviation	Average	Standard Deviation
Standard Method	0.090	0.010	0.160	0.068
Weight Decay	0.054	0.004	0.038	0.015
Weight Elimination	0.047	0.012	0.036	0.035
α-feature Detector				
$\alpha = 0.0$	0.048	0.007	0.030	0.025
$\alpha = 0.1$	0.041	0.004	0.016	0.013
$\alpha = 0.2$	0.040	0.003	0.014	0.010
$\alpha = 0.3$	0.040	0.002	0.018	0.006
$\alpha = 0.4$	0.041	0.002	0.018	0.006
$\alpha = 0.5$	0.043	0.005	0.024	0.013
$\alpha = 0.6$	0.050	0.014	0.040	0.033
$\alpha = 0.7$	0.051	0.016	0.050	0.055

Table 2. Comparison of generalization by different methods. RMS denotes the root mean squared error. The error rate shows the number of incorrectly estimated patterns divided by the total number of patterns.

patterns divided by the total number of patterns. For simplicity, outputs larger than or equal to 0.5 were set to one, and outputs less than 0.5 were set to zero. We now examine the error rates in Table 2. When the standard BP method is used, the error rate is 0.160, a significantly large error rate. In other words, only 84% of total input patterns are correctly estimated. When weight decay and weight elimination are used, the error rate is decreased to 0.038 and 0.036 respectively. When the neural α-feature detector is used, the error rate is further decreased to 0.014, the minimum error rate for $\alpha = 0.2$. These results confirm that generalization performance can be significantly increased by using the neural α-feature detector.

6.3 Feature detection and interpretation

In this section, we examine whether the neural α-feature detector can discriminate between the reproduction part and the inference part, and that it can extract the main features of consonant cluster formation.

Figure 15(b) shows information ($D_j(0)$) for 15 α-feature detectors. As can be seen in the figure, just one α-feature detector; that is, the fifteenth detector, has much larger information. The value of information of all other detectors is close to zero. Thus, we only have to examine this feature detector with sufficiently large information. On the other hand, the values of information of hidden units by the standard method are relatively higher than the neural feature detector, as shown in Figure 15(a).

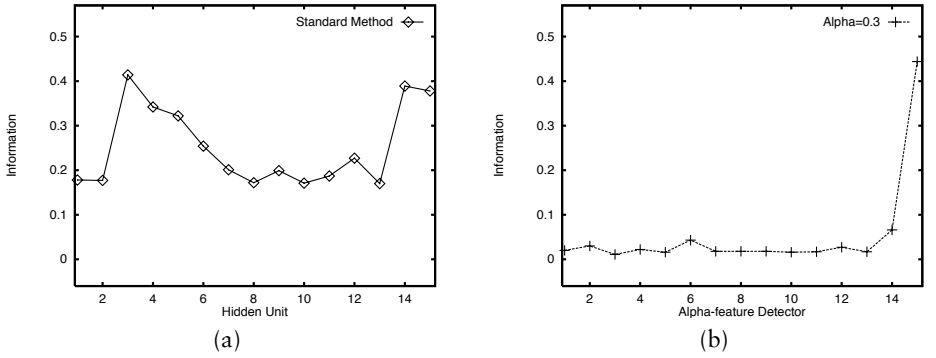


Figure 15. Information by the standard method and the neural α-feature detector; (a) information for 15 hidden units by the standard method ($\beta = 0$), and (b) information for 15 α-feature detectors. The parameter α was 0.3.

We have shown that one α-feature detector with large information can be detected. Figure 16 shows a skeleton network obtained by the neural α-feature detector. For simplicity, only an input component, α-entropy controllers, and a feature detector are plotted. In making a skeleton network as shown in Figure 16(b), only α-entropy controllers or input-hidden connections greater than 1 ($|w_{jk}| > 1$) are plotted, and all other controllers are completely eliminated. As can be seen in the figure, only five α-entropy controllers or input-hidden connections are extracted in the skeleton network. This reduction in the number of α-entropy controllers or input-hidden connections are surely concerned with improved generalization. Then, by examining α-entropy controllers closely, we can see that the inference on consonant cluster formation is based upon the *manner* and the *voicing* of consonants, which is compatible with linguistic analysis [20, 21] and with our intuition. For further explanation of the inference mechanism, see Appendix A.3.

7. Conclusion

A neural α-feature detector as been proposed. The neural α-feature detector is used to detect main or essential features observed in input patterns. A neural α-feature detector is basically composed of an input component, a feature detector (α-feature detectors and a common feature detector), an α-entropy controller, a feature generator (α-feature generators and common feature generators), and an output component. The α-entropy controller is used to control α-entropy in α-feature detectors. Controlling α-entropy corresponds to information maximization and minimization. If an α-feature detector is considered to be an important detector, information contained in the detector is maximized. On the other hand, if an α-feature detector is not considered to be impor-

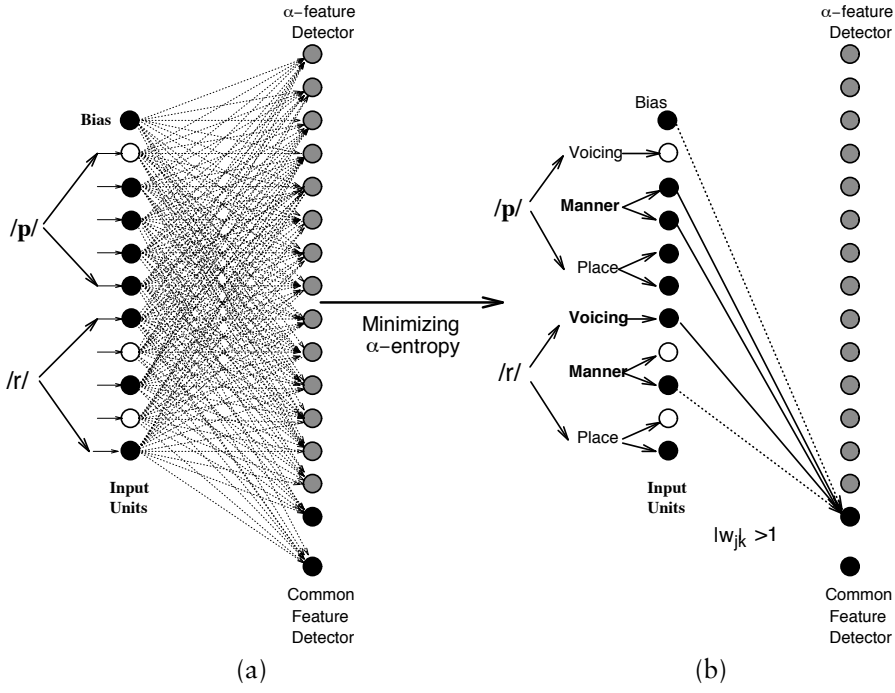


Figure 16. Skeletonization by minimizing α-entropy: (a) original network and (b) a skeleton network obtained by minimizing α-entropy.

tant, the information of the α-feature detector is decreased as much as possible.

The neural α-feature detector has been applied to four problems: an artificial *F-H* problem, recognition of six alphabet characters, recognition of 26 alphabet characters, and the inference of the well-formedness of consonant clusters. In all problems, salient features in input patterns have been successfully extracted, which can be interpreted intuitively. In addition, in the last experiments on the inference of consonant cluster formation, results confirmed that generalization performance is improved in direct proportion to the decrease of α-entropy.

Many attempts to describe and explain human cognitive processes have been made from the information theoretic point of view. One of the main problems with these attempts is that information is exclusively maximized or minimized, depending on the situation. Our neural α-feature detector is concerned with information maximization and minimization. The brain is naturally expected to do different mechanisms of optimization on different levels. Thus, we can say that our feature detector is a much better model for representing complex optimization in human intellectual activities in general.

Appendix

A. Well-formedness by sonority principle

A.1 Sonority order

As briefly discussed in section 6, we suppose that consonant cluster formation must strictly observe a sonority principle. The sonority principle states that each consonant has its own sonority value, as shown in Table 3 [20, 21]. The principle can be summarized as follows. Suppose that two consonants C_1C_2 must be placed at the beginning of words. If the sonority of the precedent consonant C_1 is smaller than that of the following consonant C_2, then consonant clusters are well-formed or permitted at the beginning of words. Otherwise, consonant clusters are not permitted at the beginning of words, that is, ill-formed. For example, a consonant cluster /pr/ is well-formed, because the sonority of /p/ (sonority=1) is less than the sonority of /r/ (sonority=7). On the other hand, a consonant cluster /rp/ is ill-formed, because the sonority (sonority=1) of /p/ is less than the sonority (sonority=7) of /r/.

A.2 Phonological representation

The number of input units was 10 for the experiments of the consonant formation discussed in section 6. Each consonant was represented in five bits by using a phonological representation from [18], as shown in Table 4.

A.3 Interpretation of internal representation

The basic mechanism of consonant formation inference is briefly explained in section 6. We now explain the inference mechanism in greater detail. Figure 17 shows a skeleton network obtained by minimizing α-entropy. It can be immediately seen in the figure that when a given

Order	Features	Examples
1	Voiceless Stop	[p, t, k]
2	Voiced Stop	[b, d, g]
3	Voiceless Fricative	[f, θ, s]
4	Voiced Fricative	[v, δ, z]
5	Nasal	[m, n, η]
6	Lateral	[l]
7	Retroflex	[r]
8	Semivowel	[y, w]

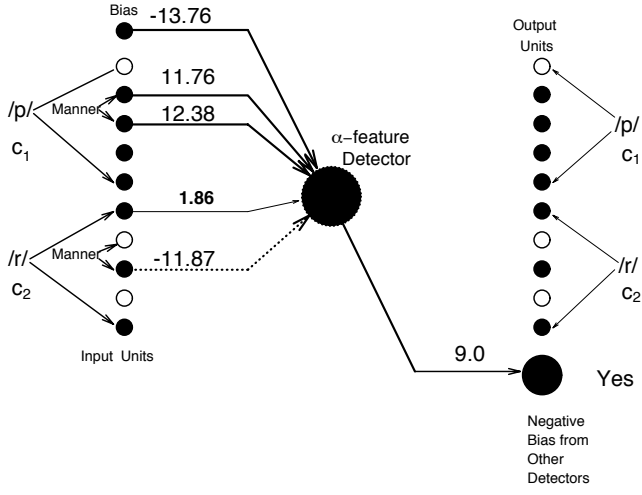
Table 3. An example of the sonority order of consonants [20]. Our artificial language is supposed to strictly observe this sonority principle.

		Phonological Features				
Sonority	Phoneme	Voicing	Manner		Place	
1	/p/	0	1	1	1	1
1	/t/	0	1	1	1	0
1	/k/	0	1	1	0	0
2	/b/	1	1	1	1	1
2	/d/	1	1	1	1	0
2	/g/	1	1	1	0	0
3	/f/	0	1	0	1	1
3	/θ/	0	1	0	1	0
3	/s/	0	1	0	0	1
4	/v/	1	1	0	1	1
4	/δ/	1	1	0	1	0
4	/z/	1	1	0	0	1
5	/m/	1	0	0	1	1
5	/n/	1	0	0	1	0
5	/η/	1	0	0	0	0
6	/l/	1	0	1	1	0
7	/r/	1	0	1	0	1
8	/y/	1	0	1	0	0
8	/w/	1	0	1	1	1

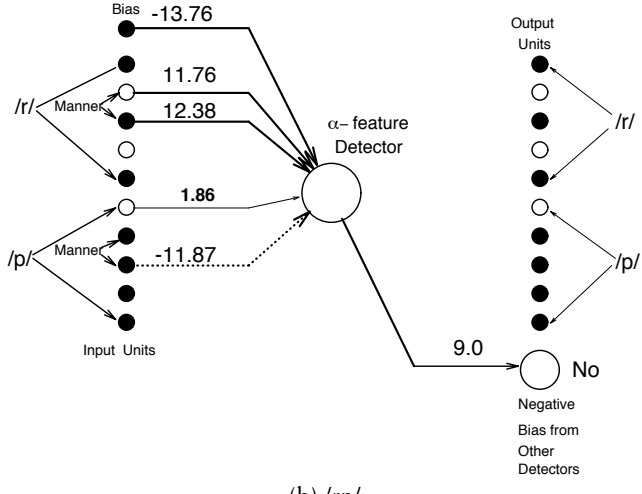
Table 4. A phonological representation used in our experiments. This representation was made by following the representation in [18].

consonant cluster is well-formed, the α-feature detector must be turned on, which makes the corresponding output unit on by the strongly positive α-feature generator or hidden-output connections (9.0). One of the easiest ways to turn the α-feature detector on is to turn on two *manner* input units, which are combined with strongly positive α-feature detectors or input-hidden connections (11.76 and 12.38) to turn the α-feature detector on. This means that the *stop* consonants such as /p,t,k,d,b/ should be placed at the front part of the consonant cluster, because the *manner* units are represented in 11 for these consonants. These *stop* consonants have lower sonority values, as shown in Table 3. Thus, the inference by the neural α-feature detector is compatible with our intuition.

Then, let us examine the second part of the input unit. It can be seen in the figure that one of the *manner* inputs of the second consonant C_2 is strongly negative (−11.87). If this input is on, the α-feature detector has the possibility of being turned off by the bias (−13.76). As can be seen in Tables 3 and 4, the *manner* of the *fricative* and *nasal* consonants such as /f,θ,s,v,δ,z,w,n,η/ are represented in 10 or 00. Thus, these consonants cannot use the strongly negative α-controller (−13.76), and the α-feature detector has a high possibility of being turned on. This means that the *fricative* and *nasal* consonants tend to be the second consonant C_2. As



(a) /pr/



(b) /rp/

Figure 17. A skeleton network obtained by minimizing α-entropy: (a) the well-formed cluster /pr/, and (b) the ill-formed cluster /rp/.

the sonority values of these consonants are higher than the sonority of the *stop* consonants, this inference corresponds to our intuition.

The *manner* of the *lateral*, *retroflex*, and *semivowel* consonants such as /l,r,y,w/ is represented in 01, meaning that by the strongly negative α-controller (-13.76) the α-feature detector tends to be turned off. However, these consonants have the highest sonority, meaning that these should be the second consonant. At this moment the *voicing* input unit

is used. By the positive α-entropy controller (1.86) from the *voicing* input unit, the α-feature detector tends to be turned on in spite of the strongly negative α-entropy controller (-11.87).

We now describe an example of the use of the *voicing* input unit. Figure 17(a) shows an internal state of the skeleton network for a well-formed consonant cluster /pr/. The first consonant is a *stop* consonant, whose *manner* input units are on ($11.76 + 12.38 = 24.14$). However, by the strongly negative α-entropy controller with the bias ($-13.76 - 11.87 = -25.63$) the α-feature detector tends to be turned off ($-25.63 + 24.14 = -1.49$). This is contrary to the sonority principle, because the consonant cluster /pr/ is well-formed. However, since the second consonant /r/ is a voiced consonant, by the α-entropy controller (1.86) from the *voicing* input unit the net input to the α-feature detector is positive ($1.86 - 1.49 = 0.37$), making the α-feature detector not fully turned on but slightly activated. Because of the strongly positive α-feature generator (9.0) the output is finally on.

Figure 17(b) shows an ill-formed case. This case is easy to understand and is one of the typical ill-formed cases. The first consonant /r/ cannot turn on two *manner* input units, meaning that the α-feature detector cannot be turned on because of the strongly negative bias (-13.76). In addition, the second consonant /p/ generates the strongly negative α-entropy controller (-11.87), making the α-entropy controller off.

References

[1] H. B. Barlow, "Unsupervised Learning," *Neural Computation*, **1** (1989) 295–311.

[2] H. B. Barlow, T. P. Kaushal, and G. J. Mitchison, "Finding Minimum Entropy Code," *Neural Computation*, **1** (1989) 412–423.

[3] R. Linsker, "Self-organization in a Perceptual Network," *Computer*, **21** (1988) 105–117.

[4] A. N. Redlich, "Redundancy Reduction as a Strategy for Unsupervised Learning," *Neural Computation*, **5** (1993) 289–304.

[5] Ryotaro Kamimura, "Hidden Information Maximization for Feature Detection and Rule Discovery," *Network*, **6** (1995) 577–622.

[6] Y. Akiyama and T. Furuya, "An Extension of the Back-propagation Learning which Performs Entropy Maximization as well as Error Minimization," Technical Report NC91-6, Institute of Electronics, Information, and Communication Engineers, 1991.

[7] G. Deco, W. Finnof, and H. G. Zimmermann, "Unsupervised Mutual Information Criterion for Elimination in Supervised Multilayer Networks," *Neural Computation*, **7** (1995) 86–107.

[8] Ryotaro Kamimura and Shohachiro Nakanishi, "Improving Generalization Performance by Information Minimization," *Institute of Electronics, Information, and Communication Engineers Transactions on Information and Systems*, **E78-D** (1995) 163–173.

[9] N. Abramson, *Information Theory and Coding* (McGraw-Hill, New York, 1963).

[10] R. Ash, *Information Theory*, (John Wiley & Sons, New York, 1965).

[11] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication* (University of Illinois Press, Champaign, 1949).

[12] L. Brillouin, *Science and Information Theory* (Academic Press, New York, 1962).

[13] L. L. Gatlin, *Information Theory and Living Systems* (Columbia University Press, New York, 1972).

[14] K. Kunisawa, *Information Theory* (Kyoritsu Publisher, Tokyo, 1983).

[15] Y. Kato, Y. Tan, and T. Ejima, "A Comparative Study with Feedforward PDP Models for Alphanumeric Character Recognition," *The Transaction of the Institute of Electronics, Information, and Communications Engineers*, **J73-DII** (1990) 1249–1254.

[16] S. A. Solla and M. Fleisher, "Accelerated Learning in Layered Neural Networks," *Complex Systems*, **2** (1988) 625–639.

[17] Y. Tan, Y. Kato, and T. Ejima, "Error Functions to Improve Learning Speed in PDP Models: A Comparative Study," *The Transactions of the Institute of Electronics, Information, and Communications Engineers*, **J73-DII** (1990) 2022–2028.

[18] K. Plunkett and V. Marchman, "U-shaped Learning and Frequency Effects in a Multilayered Perceptron: Implication for Child Language Acquisition," *Cognition*, **38** (1991) 43–102.

[19] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization by Weight-elimination with Application to Forecasting," in *Neural Information Processing Systems*, volume 3, (Morgan Kaufmann Publishers, San Mateo, CA, 1991).

[20] H. Nema, "Phonotactics and Sonority," *Senshu Journal of Foreign Language and Education*, **23** (1994) 65–94.

[21] H. Nema, "Phonotactics and Syllabification," *Journal of the Senshu University Research Society*, **56** (1995) 23–61.