Modular Structure Generation by Greedy Network-Growing Algorithm

Ryotaro Kamimura and †Haruhiko Takeuchi
Information Science Laboratory
and Future Science and Technology Joint Research Center, Tokai University,
1117 Kitakaname Hiratsuka Kanagawa 259-1292, Japan
ryo@cc.u-tokai.ac.jp
† Human-Computer Interaction Group,
Institute for Human Science and Biomedical Engineering,
National Institute of Advanced Industrial Science and Technology,
1-1-1 Higashi, Tsukuba 305-8566, Japan
takeuchi.h@aist.go.jp

Abstract—In this paper, we propose a new method to generate modular structures. In the method, the number of elements, that is, the number of competitive units is gradually increased. To control a process of module generation, we introduce two kinds of information, that is, unit and modular information. Unit information represents information content obtained by individual elements in all modules. On the other hand, modular information is information content obtained by each module. We try to increase both types of information simultaneously. We applied our method to two classification problems: random data classification and web data classification. In both cases, we observed that modular structures were automatically generated.

I. INTRODUCTION
In this paper, we propose a new computational method to generate modular structures by using the network-growing algorithm called greedy network-growing. The main point of the new method is summarized by two points: (1) this is a new competitive learning method that is realized by maximizing information; and (2) a modular structure is gradually generated. Let us explain this point in more detail.

We have so far introduced a new type of information-theoretic competitive learning [1], [2]. In these methods, competition is realized not by the winner-take-all algorithm and the lateral inhibition but by maximizing information between input patterns and competitive units. When information is maximized, only one unit is turned on, while all the other units are off. Thus, information maximization can be used to simulate competitive processes. In addition, in maximizing information content, entropy of competitive units must be maximized. This means that all competitive units must be equally used. Thus, no dead neurons should not be generated in our information maximization method.

In the second place, modular structures are gradually generated by repeatedly maximizing information. In growing networks, we use a greedy network-growing algorithm [3], [4]. The greedy network-growing algorithm can increase the number of competitive units gradually. In the beginning, a module is composed of only one competitive unit. The number of competitive units is gradually increased. In growing a modular structure, we try to generate competitive units with properties similar to each other. For this, a new growing cycle starts with connections weights obtained by the previous growing cycle. Thus, in a module, all elements tend to behave similarly to each other.

II. MODULAR STRUCTURE GENERATION

Figure 1 shows a network architecture in which four units are grouped into two modules. In this architecture, two competitive units in each module are expected to behave similarly to each other. Figure 2 shows a process of a modular structure in which the greedy network-growing algorithm is used. In the greedy algorithm [3], [4], we suppose that a network attempts to absorb as much information as possible from the outer environment, because in the outer environment, there are many destructive forces against artificial systems. Therefore, we assume at least that artificial systems must absorb as much information as possible as defense against the destructive forces. To absorb information on the outer environment, the systems gradually increase their complexity until no more complexity is needed. When no more additional information can be obtained, the network recruits another unit, and then it again tries to absorb information.
We define two types of information, that is, unit information and modular information. Unit information is information content obtained for each competitive unit. On the other hand, modular information is information content for each module. Let us compute the first type of information, that is, unit information. We consider unit information content stored in each competitive unit. For this purpose, let us define information to be stored in a neural system. Information stored in the system is represented by a decrease in uncertainty [5]. Uncertainty decrease, that is, information \( I \) is defined by

\[
I = - \sum_{j} p(j) \log p(j) + \sum_{s} \sum_{j} p(s)p(j \mid s) \log p(j \mid s),
\]

where \( p(j) \), \( p(s) \) and \( p(j \mid s) \) denote the probability of the \( j \)th unit in a system, the probability of the \( s \)th input pattern and the conditional probability of the \( j \)th unit, given the \( s \)th input pattern, respectively.

Let us present update rules to maximize information content in every stage of learning. As shown in Figure 2, a network is composed of input units \( x_k^s \) and competitive units \( v_j^s \). The \( j \)th competitive unit receives a net input from input units, and an output from the \( j \)th competitive unit can be computed by

\[
v_j^s = \exp \left( - \frac{\sum_{k=1}^{L} (x_k^s - w_{jk})^2}{2\sigma^2} \right),
\]

where \( L \) is the number of input units, and \( w_{jk} \) denote connections from the \( k \)th input unit to the \( j \)th competitive unit. The output is increased as connection weights are closer to input patterns.

In the previous method, we used the inverse of the squares of Euclidean distance between input patterns and connection weights:

\[
v_j^s = \frac{1}{\sum_{k=1}^{L} (x_k^s - w_{jk})^2}.
\]

The conditional probability \( p(j \mid s) \) is computed by

\[
p(j \mid s) = \frac{v_j^s}{\sum_{m=1}^{M} v_m^s},
\]

where \( M \) denotes the number of competitive units. Since input patterns are supposed to be given uniformly to networks, the probability of the \( j \)th competitive unit is computed by

\[
p(j) = \frac{1}{S} \sum_{s=1}^{S} p(j \mid s).
\]

Information \( I \) is computed by

\[
I = - \sum_{j=1}^{M} p(j) \log p(j) + \frac{1}{S} \sum_{s=1}^{S} \sum_{j=1}^{M} p(j \mid s) \log p(j \mid s).
\]

As information becomes larger, specific pairs of input patterns and competitive units become strongly correlated. Differentiating information with respect to input-competitive connections \( w_{jk} \), we have

\[
\frac{\partial I}{\partial w_{jk}} = - \sum_{s} \left( \log p(j) - \sum_{m=1}^{M} p(m \mid s, t) \log p(m) \right) Q_j^s
\]

\[
+ \frac{1}{S} \sum_{s=1}^{S} \left( \log p(j \mid s) - \sum_{m=1}^{M} p(m \mid s) \log p(m \mid s) \right) \times Q_j^s
\]

where

\[
Q_j^s = \frac{x_j^s - w_{jk}}{S \sigma^2} \exp \left( - \frac{\sum_{k=1}^{L} (x_k^s - w_{jk})^2}{2\sigma^2} \right).
\]

Thus, we have update rules

\[
\Delta w_{jk} = -\beta \sum_{s=1}^{S} \left( \log p(j) - \sum_{m=1}^{M} p(m \mid s) \log p(m) \right) Q_j^s
\]

\[
+ \beta \sum_{s=1}^{S} Q_j^s \log p(j \mid s)
\]

\[
- \beta \sum_{s=1}^{S} Q_j^s \sum_{m=1}^{M} p(m \mid s, t) \log p(m \mid s).
\]

A. Modularization

In computing modular information, several units are combined to produce a module. For this, the output from a module can be computed by the sum of all outputs in each module:

\[
V_r^s = \exp \left( - \frac{1}{U} \sum_{j} \sum_{k=1}^{L} (x_k^s - w_{jk})^2 \right),
\]

where \( U \) is the number of competitive units in the \( r \)th module. The conditional probability \( p(j \mid s) \) is computed by

\[
p(j \mid s) = \frac{v_j^s}{\sum_{r=1}^{R} v_r^s},
\]

where \( R \) is the number of modules. Since input patterns are supposed to be given uniformly to networks, the probability of the \( r \)th module is computed by

\[
p(r) = \frac{1}{S} \sum_{s=1}^{S} p(r \mid s).
\]
Information $I$ is computed by

$$I = -\sum_{r=1}^{R} p(r) \log p(r)$$

$$+ \frac{1}{S} \sum_{s=1}^{S} \sum_{r=1}^{R} p(r \mid s) \log p(r \mid s). \quad (13)$$

Differentiating this information in the same way as unit information, we can have update rules to maximize modular information. Units composed of a module must behave similarly to each other. For this, we use our greedy network-growing algorithm. We modified the greedy algorithm such that a new unit must be added with connection weights similar to connection weights previously obtained. This mechanism assures that all competitive units in a module are similar to each other.

III. ARTIFICIAL DATA ANALYSIS

In this experiment, we try to generate a modular structure by using uniform random numbers with two different mean values. Figure 3 shows a network architecture for this problem. In the input layer, two paired values are given, and the number of competitive units is forced to be increased to six units for two modules.

![Greedy network-growing architecture](image)

Figure 4(a) shows an original data. Figure 4(b) shows unit information as a function of the number of epochs. Information is gradually increased for three growing epochs. Figure 4(c) shows modular information as a function of the number of epochs. Modular information is increased up to a maximum point immediately for any growing stage. Figure 5(c) shows modular structure obtained by information maximization. As can be seen in the figure, two groups of data are surrounded by connection weights generated by information maximization.

IV. WEB DATA ANALYSIS

We present experimental results on a web data analysis. In this problem, we identify the meaning of polysemous word from its context. Japanese word “kanazuchi” has two meanings. One meaning corresponds to a hammer, and another corresponds to a person who can not swim. We gathered web pages in which “kanazuchi” appeared. We used a search engine Google. In the beginning, we gathered 25 web pages where the both of “kanazuchi” and “kiru”, which correspond to “cut” in English, appered. And then we gathered other 25 web pages where the both of “kanazuchi” and “oyogu”, which correspond to “swim” in English, appered. We checked that the word “kanazuchi” was used for a hammer in the former 25 web pages, and that was used for a person who can not swim in the latter 25 web pages. The 50 web pages were parsed and nouns were extracted for the further analysis. Then we made a co-occurrence matrix. The line of the matrix corresponds to words and the column corresponds to web pages. The number of different words is 1126, and the number of web pages is 50. Figure 6 shows a modular structure for the web problem.

![Modular architecture for the web problem](image)

Figure 7(a) shows unit information as a function of the number of epochs. Information is gradually increased for each growing cycle. Figure 7(b) shows modular information as a function of the number of epochs. Modular information is gradually increased for each epoch, and close to a maximum point.

Figure 8 shows competitive unit activation patterns for the first to fourth cycle. We can clearly see two modules in which the number of competitive units is gradually increased. Figure 8(a) shows the activation patterns when each module has just one unit. As can be seen in the figure, the first competitive
unit responds to "hammer" in the meaning, and the second competitive unit responds to "a person who can not swim" in the meaning. Only one web page (No. 46) was not correctly characterized. Figure 8(b) shows the activation patterns when each module has two units. Among the "a person who can not swim" category, three web pages (No. 31, 35, 36) were shown to be strongly similar in the meaning. Those three web pages were diaries, and they showed how they learned to swim. Figure 8(c) shows the activation patterns when each module has three units. Here, again, detailed structure of the web pages were shown. For example, two web pages (No. 18 and 23) were shown to be similar in the meaning. Both of the pages were written about the experiences of making kitchen and wood decks. Figure 9 shows experimental results by ordinary competitive learning. Because no module structures are supposed, it is difficult to see important characteristics in connection weights.

V. CONCLUSION

In this paper, we have proposed a new method to generate modular structures by using the greedy network-growing algorithm. The algorithm can gradually create modular structure by recruiting new neurons. In growing networks, neurons in a module are forced to behave similarity to each other with similar connection weights. Thus, a module with similar elements is finally generated. We have applied our method

Fig. 4. Original artificial data (a), individual information (b), modular information (c) and a modular structure (d).

Fig. 7. Unit information (a) and modular information (b) as a function of the number of epochs.
to random data classification and web data classification. In both cases, clear modular structures are observed to grow in the course of learning. For simplicity reason, we have restricted the number of modules to two. However, the number of modules should be larger. In addition, we should take into account hierarchical structures between modules in the future studies.

Acknowledgment

The author is very grateful to Taeko Kamimura and Mitali Das for valuable comments and suggestions.

REFERENCES


Fig. 8. Connection weights after the first (a), the second (b), the third (c) and the fourth (b) growing cycle.

Fig. 9. Connection weights obtained by ordinary competitive learning.