Neural Networks 97 (2018) 62–73

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Modular representation of layered neural networks☆

Chihiro Watanabe *, Kaoru Hiramatsu, Kunio Kashino

NTT Communication Science Laboratories, 3-1, Morinosato Wakamiya, Atsugi-shi, Kanagawa Pref. 243-0198, Japan

ARTICLE INFO

Article history:
Received 2 March 2017
Received in revised form 29 August 2017
Accepted 29 September 2017
Available online 12 October 2017

Keywords:
Layered neural networks
Network analysis
Community detection

ABSTRACT

Layered neural networks have greatly improved the performance of various applications including image processing, speech recognition, natural language processing, and bioinformatics. However, it is still difficult to discover or interpret knowledge from the inference provided by a layered neural network, since its internal representation has many nonlinear and complex parameters embedded in hierarchical layers. Therefore, it becomes important to establish a new methodology by which layered neural networks can be understood.

In this paper, we propose a new method for extracting a global and simplified structure from a layered neural network. Based on network analysis, the proposed method detects communities or clusters of units with similar connection patterns. We show its effectiveness by applying it to three use cases. (1) Network decomposition: it can decompose a trained neural network into multiple small independent networks thus dividing the problem and reducing the computation time. (2) Training assessment: the appropriateness of a trained result with a given hyperparameter or randomly chosen initial parameters can be evaluated by using a modularity index. And (3) data analysis: in practical data it reveals the community structure in the input, hidden, and output layers, which serves as a clue for discovering knowledge from a trained neural network.

1. Introduction

Layered neural networks have recently been applied to various tasks (Bengio, Courville, & Vincent, 2013; LeCun, Bengio, & Hinton, 2015), including image processing (Krizhevsky, Sutskever, & Hinton, 2012; Tompson et al., 2014), speech recognition (Hinton et al., 2012; Sainath et al., 2013), natural language processing (Collobert, 2011; Sutskever, Vinyals, & Le, 2014), and bioinformatics (Leung et al., 2014; Xiong et al., 2015). Although they have simple layered structures of units and connections, they outperform other conventional models by their ability to learn complex nonlinear relationships between input and output data. In each layer, inputs are transformed into more abstract representations under a given set of the model parameters. These parameters are automatically optimized through training so that they extract the important features of the input data. In other words, it does not require either careful feature engineering by hand, or expert knowledge of the data. This advantage has made layered neural networks successful in a wide range of tasks, as mentioned above.

However, the inference provided by a layered neural network consists of a large number of nonlinear and complex parameters, which makes it difficult for human beings to understand it. More complex relationships between input and output can be represented as the network becomes deeper or the number of units in each hidden layer increases, however interpretation becomes more difficult. The large number of parameters also causes problems in terms of computational time, memory and over-fitting, so it is important to reduce the parameters appropriately. Since it is difficult to read the underlying structure of a neural network and to identify the parameters that are important to keep, we must perform experimental trials to find the appropriate values of the hyperparameters and the random initial parameters that achieve the best trained result.

In this paper, to overcome such difficulties, we propose a new method for extracting a global and simplified structure from a layered neural network (For example, Figs. 5 and 11). Based on network analysis, the proposed method defines a modular representation of the original trained neural network by detecting communities or clusters of units with similar connection patterns. Although the modular neural network proposed by Azam (2000) and Jacobs et al. (1991) has a similar name, it takes the opposite approach to ours. In fact, it constructs the model structure before training with multiple split neural networks inside it. Then, each small neural network works as an expert of a subset task. Our proposed method is based on the community detection algorithm. To date, various methods have been proposed to express the characteristics of diverse complex networks without layered

☆ This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

* Corresponding author.
E-mail address: watanabe.chihiro@lab.ntt.co.jp (C. Watanabe).

https://doi.org/10.1016/j.neunet.2017.09.017
0893-6080/

structures (Estrada & Velázquez, 2005; Meunier & Paugam-Moisy, 2006; Newman, 2006; Newman & Girvan, 2004; Newman & Leicht, 2007), however, no method has been developed for detecting the community structures of trained layered neural networks.

The difficulty of conventional community detection from a layered neural network arises from the fact that an assumption commonly used in almost all conventional methods does not hold for layered neural networks: to detect the community structure of network, previous approaches assume that there are more intra-community edges that connect vertices inside a community than inter-community edges that connect vertices in mutually different communities. A network with such a characteristic is called assortative. This seems to be a natural assumption, for instance, for a network of relationships between friends. In layered neural networks, however, units in the same layer do not connect to each other and they only connect via units in their parent or child layers. This characteristic is similar to that of a bipartite graph, and such networks are called disassortative. It is not appropriate to apply conventional methods based on the assumption of an assortative network to a layered neural network. A basic community detection method that can be applied to either assortative or disassortative networks has been proposed by Newman and Leicht (2007). In this paper, we propose an extension of this method for extracting modular representations of layered neural networks.

The proposed method can be employed for various purposes. In this paper, we show its effectiveness with the following three applications.

1. **Decomposition of layered neural network into independent networks**: the proposed method decomposes a trained neural network into multiple small independent neural networks. In such a case, the output estimation by the original neural network can be regarded as a set of independent estimations made by the decomposed neural networks. In other words, it divides the problem and reduces the overall computation time. In Section 4.1, we show that our method can properly decompose a neural network into multiple independent networks, where the data consist of multiple independent vectors.
2. **Generalization error estimation from community structure**: modularity (Newman & Girvan, 2004) is defined as a measure of the effectiveness of a community detection result. Section 4.2 reveals that there is a correlation between modularity and the generalization error of a layered neural network. It is shown that the appropriateness of the trained result can be estimated from the community structure of the network.
3. **Knowledge discovery from modular representation**: the modular representation extracted by the proposed method serves as a clue for understanding the trained result of a layered neural network. It extracts the community structure in the input, hidden, and output layer. In Section 4.3, we introduce the result of applying the proposed method to practical data.

The remaining part of this paper is composed as follows: we first describe a layered neural network model in Section 2. Then, we explain our proposed method for extracting a modular representation of a neural network in Section 3. The experimental results are reported in Section 4, which show the effectiveness of the proposed method in the above three applications. In Section 5, we discuss the experimental results. Section 6 concludes this paper.

2. Layered neural networks

We start by defining $x \in \mathbb{R}^M$, $y \in \mathbb{R}^N$ and a probability density function $q(x, y)$ on $\mathbb{R}^M \times \mathbb{R}^N$. A training dataset $\{(X_i, Y_i)\}_{i=1}^n$ with a sample size n is assumed to be generated independently from $q(x, y)$. Let $f(x, w)$ be a function from $x \in \mathbb{R}^M$, $w \in \mathbb{R}^L$ to $\mathbb{R}^N$ of a layered neural network that estimates an output y from an input x and a parameter w.

For a layered neural network, $w = \{\omega_{ij}^d, \theta_i^d\}$, where ω_{ij}^d is the weight of connection between the ith unit in the depth d layer and the jth unit in the depth $d+1$ layer, and θ_i^d is the bias of the ith unit in the depth d layer. A layered neural network with D layers is represented by the following function:

$$
\begin{aligned}
f_j(x, w) &= \sigma\Big(\sum_i \omega_{ij}^{D-1} o_i^{D-1} + \theta_j^{D-1}\Big),\\
o_j^{D-1} &= \sigma\Big(\sum_i \omega_{ij}^{D-2} o_i^{D-2} + \theta_j^{D-2}\Big),\\
&\vdots\\
o_j^{2} &= \sigma\Big(\sum_i \omega_{ij}^{1} x_i + \theta_j^{1}\Big),
\end{aligned}
$$

where a sigmoid function is defined by

$$\sigma(x) = \frac{1}{1+\exp(-x)}.$$

The training error $E(w)$ and the generalization error $G(w)$ are, respectively, defined by

$$
\begin{aligned}
E(w) &= \frac{1}{n}\sum_{i=1}^{n} \|Y_i - f(X_i, w)\|^2,\\
G(w) &= \int \|y - f(x, w)\|^2 q(x, y)\,dxdy,
\end{aligned}
$$

where $\|\cdot\|$ is the Euclidean norm of $\mathbb{R}^N$.

The generalization error is approximated by

$$G(w) \approx \frac{1}{m}\sum_{j=1}^{m} \|Y_j' - f(X_j', w)\|^2,$$

where $\{(X_j', Y_j')\}_{j=1}^m$ is a test dataset that is independent of the training dataset.

To construct a sparse neural network, we adopt the LASSO method (Ishikawa, 1990; Tibshirani, 1994) in which the minimized function is defined by

$$H(w) = \frac{n}{2} E(w) + \lambda \sum_{d,i,j} |\omega_{ij}^d|,$$

where λ is a hyperparameter.

The parameters are trained by the stochastic steepest descent method,

$$
\begin{aligned}
\Delta w &= -\eta \nabla H_i(w)\\
&= -\eta\left(\frac{1}{2}\nabla\{\|Y_i - f(X_i, w)\|^2\} + \lambda\,\mathrm{sgn}(w)\right),
\end{aligned}
\tag{1}
$$

where $H_i(w)$ is the training error computed only from the ith sample (X_i, Y_i). Here, η is defined for training time t such that

$$\eta(t) \propto \frac{1}{t},$$

which is sufficient for convergence of the stochastic steepest descent. Eq. (1) is numerically calculated by the following procedure, which is called error back propagation (Rumelhart, Hinton, & Williams, 1986; Werbos, 1974): for the Dth layer,

$$
\begin{aligned}
\delta_j^D &= (o_j^D - y_j)\, o_j^D\,(1 - o_j^D),\\
\Delta\omega_{ij}^{D-1} &= -\eta(\delta_j^D o_i^{D-1} + \lambda\,\mathrm{sgn}(\omega_{ij}^{D-1})),\\
\Delta\theta_j^D &= -\eta\delta_j^D.
\end{aligned}
$$

For $d = D-1,\ D-2, \ldots, 2$,

$$\delta_j^d = \sum_{k=1}^{l_{d+1}} \delta_k^{d+1} \omega_{jk}^d\, o_j^d\, (1 - o_j^d),$$
$$\Delta\omega_{ij}^{d-1} = -\eta(\delta_j^d o_i^{d-1} + \lambda\ \mathrm{sgn}(\omega_{ij}^{d-1})),$$
$$\Delta\theta_j^d = -\eta\delta_j^d.$$

Algorithm 1 is used for training a layered neural network based on error back propagation. With this algorithm, we obtain a neural network whose redundant weight parameters are close to zero.

Algorithm 1 Stochastic steepest descent algorithm of a layered neural network

for $i = 1$ to $a_1 * n$ **do**
  Randomly sample k from uniform distribution on $\{1, 2, \cdots, n\}$.
  $x_j \leftarrow x_j^k$,
  $y_j \leftarrow y_j^k$,
  where x_j^k and y_j^k is the j-th element of k-th sample.
  $\eta = 0.8 \times \frac{a_1 \times n}{a_1 \times n + 5 \times i}$,
  where $a_1 \times n$ is the number of iterations. (Here, we defined η so that it gets smaller and smaller as the iteration proceeds, to accelerate convergence of the algorithm.)
  (1) Output calculation of all layers: let o_j^d be an output of the j-th unit in the depth d layer.
  $o_j^1 \leftarrow x_j$.
  for $d = 2$ to D **do**
    $o_j^d \leftarrow \sigma(\sum_i \omega_{ij}^{d-1} o_i^{d-1} + \theta_j^d)$.
  end for
  (2) Update weight ω_{ij}^d and bias θ_j^d based on back propagation, where $\epsilon > 0$ is a small constant.
  $\delta_j^D \leftarrow (o_j^D - y_j)(o_j^D(1 - o_j^D) + \epsilon)$.
  $\Delta\omega_{ij}^{D-1} \leftarrow -\eta(\delta_j^D o_i^{D-1} + \lambda\ \mathrm{sgn}(\omega_{ij}^{D-1}))$.
  $\Delta\theta_j^D \leftarrow -\eta\delta_j^D$.
  for $d = D-1$ to 2 **do**
    $\delta_j^d \leftarrow \sum_{j'} \delta_{j'}^{d+1} \omega_{jj'}^d (o_j^d(1 - o_j^d) + \epsilon)$.
    $\Delta\omega_{ij}^{d-1} \leftarrow -\eta(\delta_j^d o_i^{d-1} + \lambda\ \mathrm{sgn}(\omega_{ij}^{d-1}))$.
    $\Delta\theta_j^d \leftarrow -\eta\delta_j^d$.
  end for
  $\omega_{ij}^d \leftarrow \omega_{ij}^d + \Delta\omega_{ij}^d$.
  $\theta_j^d \leftarrow \theta_j^d + \Delta\theta_j^d$.
end for

3. Modular representation of layered neural networks

Here we propose a new community detection method, which is applied to any layered neural networks (Fig. 1(A)). The proposed method is an extension of the basic approach proposed by Newman and Leicht (2007). It detects communities of assortative or disassortative networks. The key idea behind our method is that the community assignment of the units in each layer is estimated by using connection with adjacent layers.

As shown in Fig. 1(B), a partial network consisting of the connections between every layer and its adjacent layers is represented in the form of two matrices: $A^d = \{A_{ij}^d\}$ and $B^d = \{B_{ij}^d\}$. The matrix A^d and B^d represent the connections between two layers of depth $d-1$ and d, and two layers of depth d and $d+1$, respectively. In this paper, an element A_{ij}^d is given by

$$A_{ij}^d = \begin{cases} 1 & (|\omega_{ij}^{d-1}| \geq \xi), \\ 0 & (otherwise), \end{cases} \tag{2}$$

where ξ is called a weight removing hyperparameter. In a similar way, an element B_{ij}^d is given by

$$B_{ij}^d = \begin{cases} 1 & (|\omega_{ij}^d| \geq \xi), \\ 0 & (otherwise). \end{cases} \tag{3}$$

For simplicity, we denote A^d and B^d as A and B, respectively, in the following explanation.

Our method is based on the assumption that units in the same community have a similar probability of connection from/to other units. This assumption is almost the same as that in the previous method (Newman & Leicht, 2007), except that our method utilizes both incoming and outgoing connections of each community, and it detects communities in individual layers. Therefore, the community detection result is derived in a similar way to the previous method (Newman & Leicht, 2007), as explained in the rest of this section. As shown on the right in Fig. 1(B), the statistical model for community detection has three kinds of parameters. The first parameter $\pi = \{\pi_c\}$ represents the prior probability of a unit in the depth d layer that belongs to the community c. The conditional probability of connections for a given community c is represented by the second and third parameters $\tau = \{\tau_{c,i}\}$ and $\tau' = \{\tau'_{c,j}\}$, where $\tau_{c,i}$ represents the probability that a connection to a unit in the community c is attached from the ith unit in the depth $d-1$ layer. Similarly, $\tau'_{c,j}$ represents the probability that a connection from a unit in the community c is attached to the jth unit in the depth $d+1$ layer. Here, we omit the index d for these parameters $\pi,\ \tau,\ \tau'$ for simplicity. These parameters are normalized so that they satisfy the following condition:

$$\sum_c \pi_c = 1. \quad \sum_i \tau_{c,i} = 1. \quad \sum_j \tau'_{c,j} = 1. \tag{4}$$

Our purpose is to find the parameters $\pi,\ \tau,\ \tau'$ that maximize the likelihood of given matrices $A,\ B$. To solve this problem, we introduce the community assignment $g = \{g_k\}$, where g_k is the community of the kth unit in the depth d layer. The parameters are optimized so that they maximize the likelihood of $A,\ B$ and g:

$$\Pr(A, B, g|\pi, \tau, \tau') = \Pr(A, B|g, \pi, \tau, \tau')\Pr(g|\pi, \tau, \tau'),$$

where

$$\Pr(A, B|g, \pi, \tau, \tau') = \prod_k \left\{ \prod_i \left(\tau_{g_k,i}\right)^{A_{i,k}} \right\} \left\{ \prod_j \left(\tau'_{g_k,j}\right)^{B_{k,j}} \right\},$$
$$\Pr(g|\pi, \tau, \tau') = \prod_k \pi_{g_k}.$$

Then, the log likelihood of $A,\ B$ and g is given by

$$\mathcal{L} = \ln \Pr(A, B, g|\pi, \tau, \tau')$$
$$= \sum_k \left\{ \ln \pi_{g_k} + \sum_i A_{i,k} \ln \tau_{g_k,i} + \sum_j B_{k,j} \ln \tau'_{g_k,j} \right\}.$$

Here, the community assignment g is a latent variable and is unknown in advance, so we cannot directly calculate the above $\mathcal{L}$. Therefore, we calculate the expected log likelihood $\bar{\mathcal{L}}$ over g instead.

$$\bar{\mathcal{L}} = \sum_{g_1} \cdots \sum_{g_I} \Pr(g|A, B, \pi, \tau, \tau') \sum_k \left\{ \ln \pi_{g_k} + \sum_i A_{i,k} \ln \tau_{g_k,i} + \sum_j B_{k,j} \ln \tau'_{g_k,j} \right\}$$
$$= \sum_{k,c} \Pr(g_k = c|A, B, \pi, \tau, \tau')$$

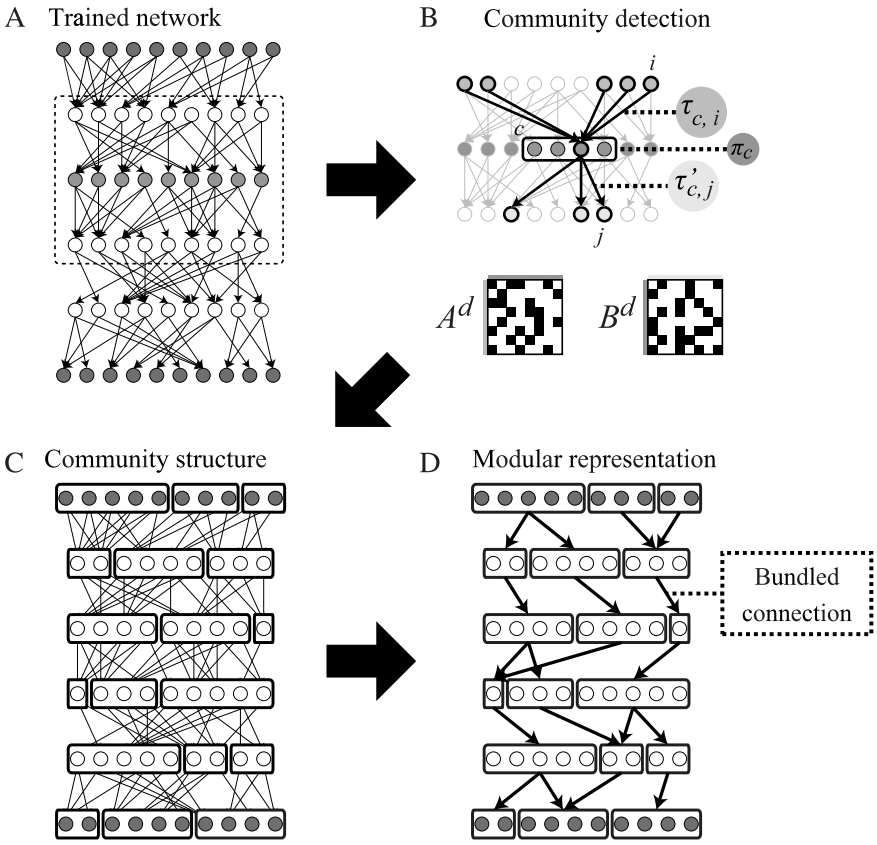

Fig. 1. Proposed method. (A) Trained network: a layered neural network is trained by the stochastic steepest descent method. (B) Community detection: the connections between every layer and its adjacent layers are represented by partial network matrices A^d and B^d. Communities in each layer are extracted by using network analysis. (C) Community structure: the community assignments of all units are determined from the estimated parameters in (B). (D) Modular representation: bundled connections are defined that summarize multiple connections between pairs of communities.

$$\times \left\{ \ln \pi_c + \sum_i A_{i,k} \ln \tau_{c,i} + \sum_j B_{k,j} \ln \tau'_{c,j} \right\},$$

where l is the number of units in the depth d layer. By defining

$$q_{k,c} = \Pr(g_k = c|A, B, \pi, \tau, \tau') = \frac{\Pr(A, B, g_k = c|\pi, \tau, \tau')}{\Pr(A, B|\pi, \tau, \tau')}, \tag{5}$$

the above equation can be rewritten as follows:

$$\bar{\mathcal{L}} = \sum_{k,c} q_{k,c} \left\{ \ln \pi_c + \sum_i A_{i,k} \ln \tau_{c,i} + \sum_j B_{k,j} \ln \tau'_{c,j} \right\}. \tag{6}$$

The parameter $q_{k,c}$ represents the probability that the kth unit is assigned to the community c. In other words, the community detection result is given by the estimated $\{q_{k,c}\}$. The optimal parameters for maximizing $\bar{\mathcal{L}}$ of Eq. (6) are found with the EM algorithm. The parameters π, τ, τ' with given $\{q_{k,c}\}$ are iteratively optimized.

Theorem 3.1. *If* $\{q_{k,c}\}$, $\{\pi_c\}$, $\{\tau_{c,i}\}$, $\{\tau'_{c,j}\}$ *maximizes* $\bar{\mathcal{L}}$, *then they satisfy*

$$q_{k,c} = \frac{\pi_c \left[\prod_i \tau_{c,i}^{A_{i,k}}\right] \left[\prod_j \tau'^{B_{k,j}}_{c,j}\right]}{\sum_s \pi_s \left[\prod_i \tau_{s,i}^{A_{i,k}}\right] \left[\prod_j \tau'^{B_{k,j}}_{s,j}\right]}, \ (\forall k,\ c) \tag{7}$$

and

$$\pi_c = \frac{\sum_k q_{k,c}}{l},$$

$$\tau_{c,i} = \frac{\sum_k q_{k,c} A_{i,k}}{\sum_{k,i} q_{k,c} A_{i,k}},$$

$$\tau'_{c,j} = \frac{\sum_k q_{k,c} B_{k,j}}{\sum_{k,j} q_{k,c} B_{k,j}}. \ (\forall c,\ i,\ j). \tag{8}$$

Proof. The denominator and numerator in the last term of Eq. (5) are given by

$$\begin{aligned}
\Pr(A, B, g_k = c|\pi, \tau, \tau') &= \sum_{g_1} \cdots \sum_{g_l} \delta_{g_k,c} \Pr(A, B, g|\pi, \tau, \tau') \\
&= \sum_{g_1} \cdots \sum_{g_l} \delta_{g_k,c} \prod_h \left\{ \pi_{g_h} \left[\prod_i \tau_{g_h,i}^{A_{i,h}} \right] \left[\prod_j \tau'^{B_{h,j}}_{g_h,j} \right] \right\} \\
&= \left\{ \pi_c \left[\prod_i \tau_{c,i}^{A_{i,k}} \right] \left[\prod_j \tau'^{B_{k,j}}_{c,j} \right] \right\} \\
&\quad \times \left\{ \prod_{h \neq k} \sum_s \pi_s \left[\prod_i \tau_{s,i}^{A_{i,h}} \right] \left[\prod_j \tau'^{B_{h,j}}_{s,j} \right] \right\},
\end{aligned}$$

and

$$\begin{aligned}
\Pr(A, B|\pi, \tau, \tau') &= \sum_{g_1} \cdots \sum_{g_l} \Pr(A, B, g|\pi, \tau, \tau') \\
&= \prod_k \sum_s \pi_s \left[\prod_i \tau_{s,i}^{A_{i,k}} \right] \left[\prod_j \tau'^{B_{k,j}}_{s,j} \right],
\end{aligned}$$

where $\delta_{i,j}$ is the Kronecker delta. Therefore, $q_{k,c}$ is given by Eq. (7).

The problem is to maximize $\bar{\mathcal{L}}$ of Eq. (6) with a given $\{q_{k,c}\}$ under the condition of Eq. (4). This is solved with the Lagrangian undetermined multiplier method, which employs

$$f = \bar{\mathcal{L}} - \alpha \sum_c \pi_c - \sum_c \beta_c \sum_i \tau_{c,i} - \sum_c \gamma_c \sum_j \tau'_{c,j},$$

and

$$\frac{\partial f}{\partial \pi_c} = \frac{\partial f}{\partial \tau_{c,i}} = \frac{\partial f}{\partial \tau'_{c,j}} = 0. \ (\forall c,\ i,\ j). \tag{9}$$

From Eq. (9), the following equations are derived:

$$\frac{\partial \bar{\mathcal{L}}}{\partial \pi_c} = \alpha, \quad \frac{\partial \bar{\mathcal{L}}}{\partial \tau_{c,i}} = \beta_c, \quad \frac{\partial \bar{\mathcal{L}}}{\partial \tau'_{c,j}} = \gamma_c. \ (\forall c,\ i,\ j). \tag{10}$$

Using Eq. (6) and Eq. (10), we obtain

$$\pi_c = \frac{1}{\alpha} \sum_k q_{k,c}, \quad \tau_{c,i} = \frac{1}{\beta_c} \sum_k q_{k,c} A_{i,k},$$

$$\tau'_{c,j} = \frac{1}{\gamma_c} \sum_k q_{k,c} B_{k,j}. \ (\forall c,\ i,\ j). \tag{11}$$

From Eq. (11) and the condition of Eq. (4), Lagrange's undetermined multipliers α, $\{\beta_c\}$, $\{\gamma_c\}$ are determined, and Eq. (11) is rewritten as Eq. (8). □

From the above theorem, the optimal parameters π, τ, τ' and the probability of community assignment q for the optimized parameters are iteratively estimated based on Eqs. (7) and (8). In this paper, the community assigned to the kth unit is determined by the c that maximizes $q_{k,c}$ (Fig. 1(C)).

Finally, we use the following methods to determine a modular representation of a layered neural network that summarizes multiple connections between the pairs of communities (Fig. 1(D)).

Four algorithms for determining bundled connections:

- Method 1: Community a and b have a bundled connection iff there exists at least one connection between the pairs of units $\{i, j\}$, $i \in a$, $j \in b$.
- Method 2: Let the number of units in communities a and b be l_a and l_b, respectively, and let the number of connections between the pairs of units $\{i, j\}$, $i \in a$, $j \in b$ be $l_{a,b}$. Communities a and b have a bundled connection iff $r_{a,b} \equiv \frac{l_{a,b}}{l_a l_b} \geq \zeta$ holds, where ζ is a threshold.
- Method 3: Among the bundled connections defined by Method 2, only those that satisfy the following (1) OR (2) are kept and the others are removed. (1) for any community a' in the same layer as community a, $r_{a,b} \geq r_{a',b}$. (2) for any community b' in the same layer as community b, $r_{a,b} \geq r_{a,b'}$.
- Method 4: Among the bundled connections defined by Method 2, only those that satisfy the above (1) AND (2) are kept and the others are removed.

By these procedures, we obtain the modular representation of a layered neural network.

4. Experiments

In this section, we show three applications of the proposed method: (1) the decomposition of a layered neural network into independent networks, (2) generalization error estimation from a community structure, and (3) knowledge discovery from a modular representation. Here we verify the effectiveness of the proposed method in the above three applications.

The following processing was performed in all the experiments:

(1) The input data were normalized so that the minimum and maximum values were $x_{\min}$ and $x_{\max}$, respectively.
(2) The output data were normalized so that the minimum and maximum values were 0.01 and 0.99, respectively.
(3) The initial parameters were independently generated as follows: $\omega_{ij}^d \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.5)$. $\theta_j^d \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.5)$.
(4) As in Eq. (2), the connection matrix $A_{ij}^d = 0.99$ if the absolute value of the connection weight between the ith unit in the depth $d-1$ layer and the jth unit in the depth d layer is larger than a threshold ξ, otherwise $A_{ij}^d = 0.01$. Note that 0.99 and 0.01 are used instead of 1 and 0 for stable computation. Similarly, B_{ij}^d is defined from the connection weight between the ith unit in the depth d layer and the jth unit in the depth $d+1$ layer (Eq. (3)). All units were removed that had no connections to other units.
(5) For each layer in a trained neural network, 10 community detection trials were performed. We defined the community detection result as one that achieved the largest expected log likelihood in the last of 200 iterations of the EM algorithm.
(6) In each community detection trial, the initial values of the parameters π, τ, τ' were independently generated from a uniform distribution on (0, 1), and then normalized so that Eq. (4) held.
(7) In visualization of modular representation, all communities with no output bundled connections from them were regarded as unnecessary communities. In the output layer, the communities with no input bundled connections were regarded in the same way as above. The bundled connections with such unnecessary communities were also removed. These unnecessary communities and bundled connections were detected from depth D to 1, since the unnecessary communities in the shallower layers depend on the removal of unnecessary bundled connections in the deeper layers.

4.1. Decomposition of independent layered neural networks

We show that the proposed method can properly decompose a neural network into a set of small independent neural networks, where the dataset consists of multiple independent dimensions. For validation, we made the synthetic data of three independent parts, merged them, and applied the proposed method to decompose them into the three independent parts.

4.1.1. Generation method of synthetic data

The method we used to generate the synthetic data is shown in Fig. 2. In the following, we explain the experimental settings in detail.

First, three sets of input data were independently generated. All the sets contained input data with 15 dimensions, and their values followed: $x_j^n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 3)$. Then, three neural networks were defined, each of which has independent weights and biases. In each neural network, all the layers consisted of 15 units, and the number of hidden layers was set at one. The sets of weights and biases for the first, second and third neural networks are denoted as $\{\omega,\ \theta\}$, $\{\omega',\ \theta'\}$, and $\{\omega'',\ \theta''\}$, respectively. These parameters were randomly generated as follows:

$$\omega_{i,j}^d,\ \omega'^d_{i,j},\ \omega''^d_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 2),$$

$$\theta_j^d,\ \theta'^d_j,\ \theta''^d_j \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.5).$$

For the weights ω, ω' and ω'', the connections with absolute values of one or smaller were replaced by 0.

Finally, three sets of output data were generated by using the above input data and neural networks by adding independent noise following $\mathcal{N}(0, 0.05)$. The three generated sets of input and output data were merged into one set of data, as shown in Fig. 2.

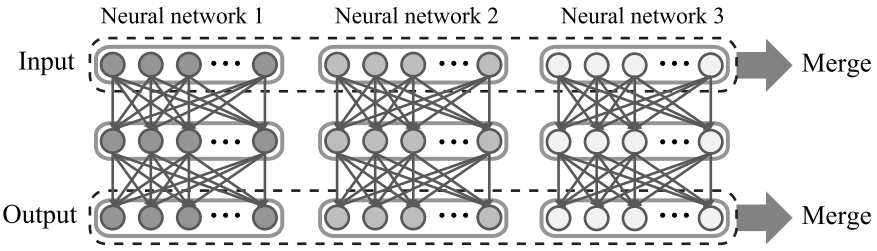

Fig. 2. Method for generating synthetic data. First, three vectors of input data were independently generated. Then, each input vector was connected to a different layered neural network with independent weights and biases. Independent noises were added to the resulting three output vectors, to generate the output data. These three sets of input and output data were merged into one.

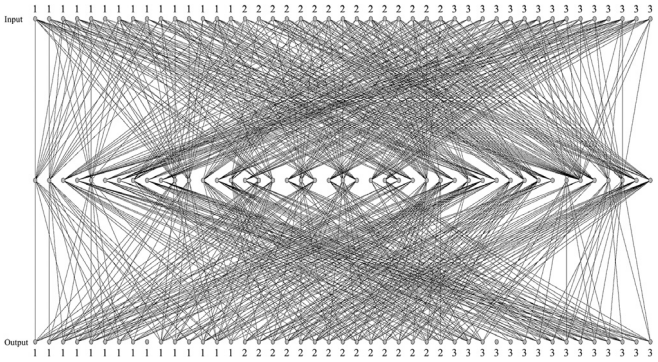

Fig. 3. Neural network trained using the synthetic data. The numbers above the input layer and below the output layer are the indices of the three datasets.

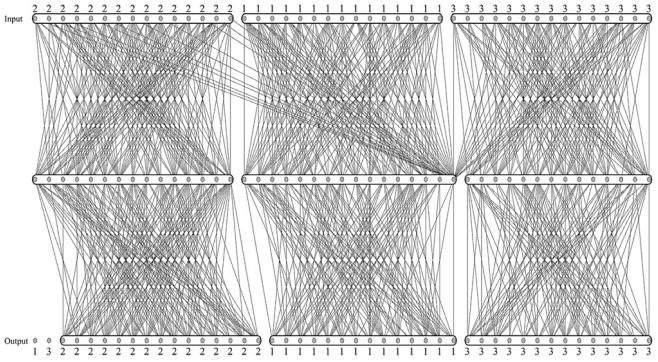

Fig. 4. Result of community detection with proposed method.

4.1.2. Neural network training and modular representation extraction

We trained another neural network with 45 dimensions for input, hidden and output layer using the merged data. Then, a modular representation of the trained neural network was made with the proposed method. The results of the trained neural network, its community structure, and its modular representation are shown in Figs. 3, 4, and 5, respectively. The numbers above the input layer and below the output layer are the indices of the three sets of data. These results showed that the proposed method could decompose the trained neural network into three independent networks.

4.1.3. Modular representation extraction using data generated by mutually dependent neural networks

We also conducted an experiment to extract modular representations from neural networks trained with data generated by mutually dependent neural networks. To control the extent of independence between three neural networks, we added κ bundled connections between randomly chosen pairs of communities in

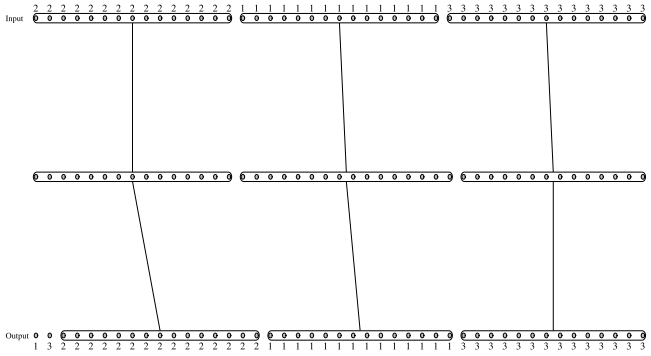

Fig. 5. Extracted modular representation of trained neural network. These results showed that our method could decompose the trained neural network into three independent networks.

mutually adjacent layers, where κ is set at $1, 2, \ldots, 10$. For example, in Fig. 2, the community in the input layer of neural network 1 and the community in the hidden layer of neural network 2 are randomly chosen and a bundled connection is added between them. The connection weights between a pair of communities with a bundled connection are independently generated from $\mathcal{N}(0, 2)$. As in the experiment described in Section 4.1.1, the output data of three communities were generated by using the above neural network with additional bundled connections and three sets of input data that were independently generated.

As in Section 4.1.2, a modular representations of the trained neural networks were extracted with the proposed method, with varying number of true additional bundled connections ($\kappa = 1, 2, \ldots, 10$). The results of the modular representations are shown in Fig. 6. These results showed that our proposed method could almost properly decompose the units in all layers when $\kappa \leq 7$ holds. As the number of additional bundled connections increases ($\kappa \geq 8$), a pair of communities in the same layer is more likely to share much connections to other units in the ground truth neural network, resulting that the units in such ground truth communities cannot be decomposed properly.

4.2. Generalization error estimation from community structure

In general, a trained result of a layered neural network is affected by different hyperparameters and initial parameter values. Here, we show that the appropriateness of a trained result can be estimated from the extracted community structure by checking the correlation between the generalization error and the modularity (Newman & Girvan, 2004).

The modularity is defined as a measure of the effectiveness of the community detection result, and it becomes higher with more intra-community connections and fewer inter-community connections. In other words, a network can be divided into different communities more clearly, as the modularity becomes higher.

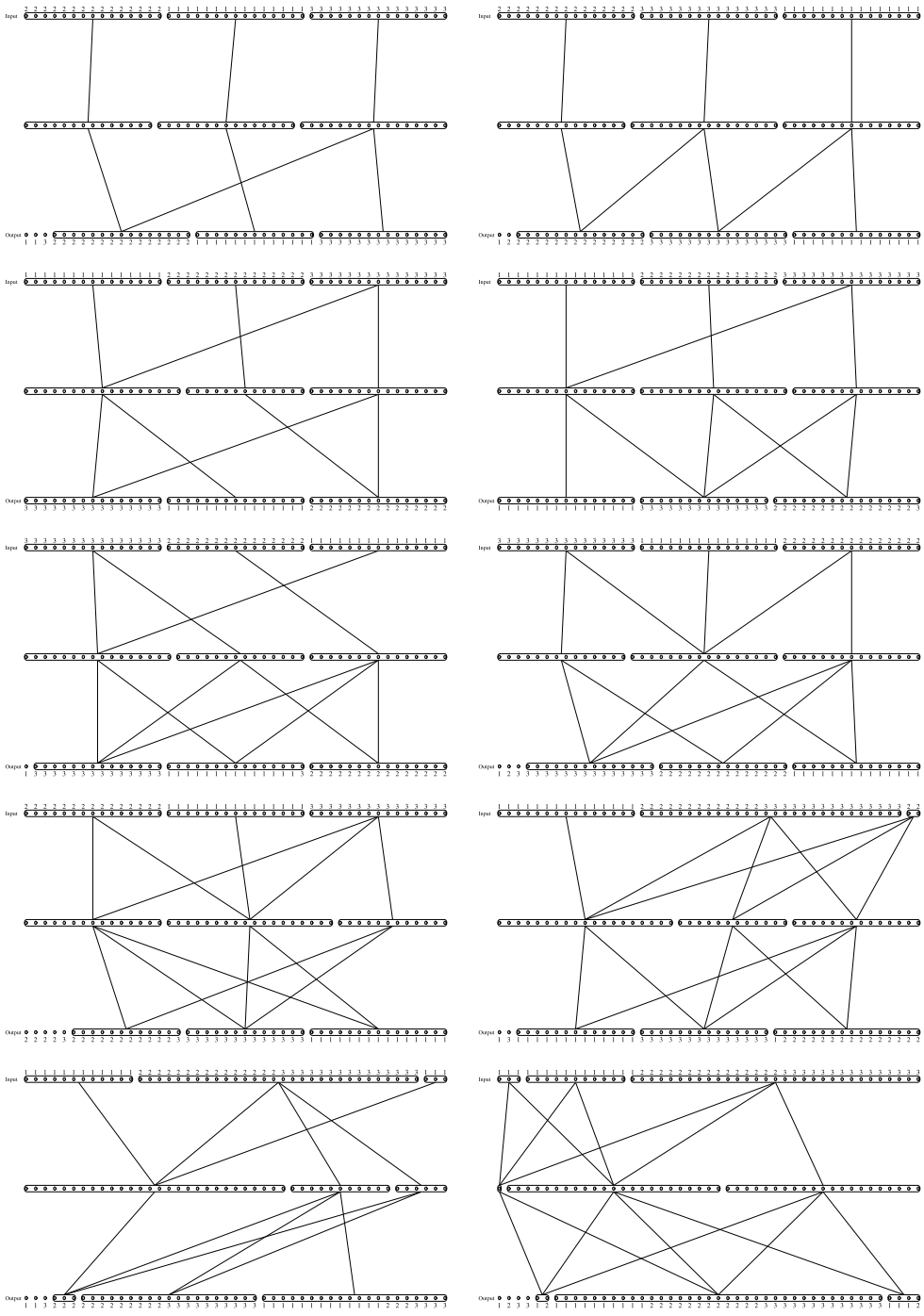

Fig. 6. Extracted modular representation of neural network trained using the data with varying true structure. From the top left, the number of true additional bundled connections is 1, 2, . . . , 10.

Let the number of communities in the network be C, and $\bar{A} = \{\bar{A}_{ij}\}$ be a $C \times C$ matrix whose element $\bar{A}_{ij}$ is the number of connections between communities i and j, divided by the total number of connections in the network. The modularity Q of the network is defined by

$$Q = \sum_i \left(\bar{A}_{ii} - \left\{ \sum_j \bar{A}_{ij} \right\}^2 \right).$$

This is a measure for verifying the community structure of assortative networks, so it cannot be applied directly to layered neural networks. In this paper, we define a modified adjacency matrix based on the original adjacency matrix of a layered neural network, and use it for measuring modularity. In the modified adjacency matrix, an element indexed by row i and column j represents the number of common units that connect with both the ith and jth units (Fig. 7). We set the diagonal elements of modified adjacency matrix at 0, resulting that there are no self-loops.

4.2.1. Correlation between modularity and generalization error when using input data of mutually independent dimensions

If the data consist of multiple independent dimensions like the synthetic data used in the experiment described in Section 4.1, the

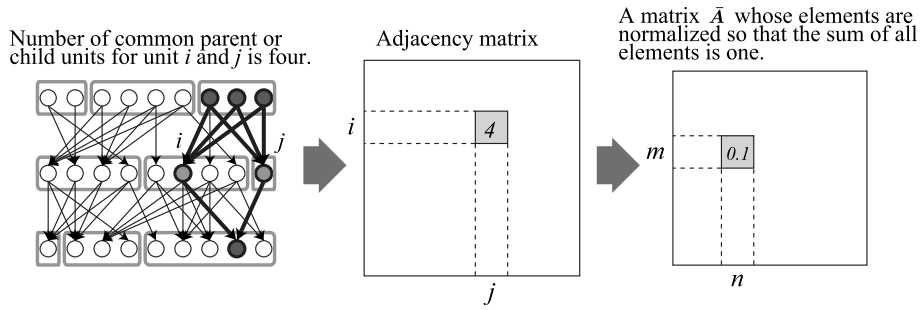

Fig. 7. Left and center: method for defining the modified adjacency matrix of a layered neural network for calculating modularity. Right: a matrix $\bar{A}$ whose elements indicate the fraction of the connection weights between two communities in the network.

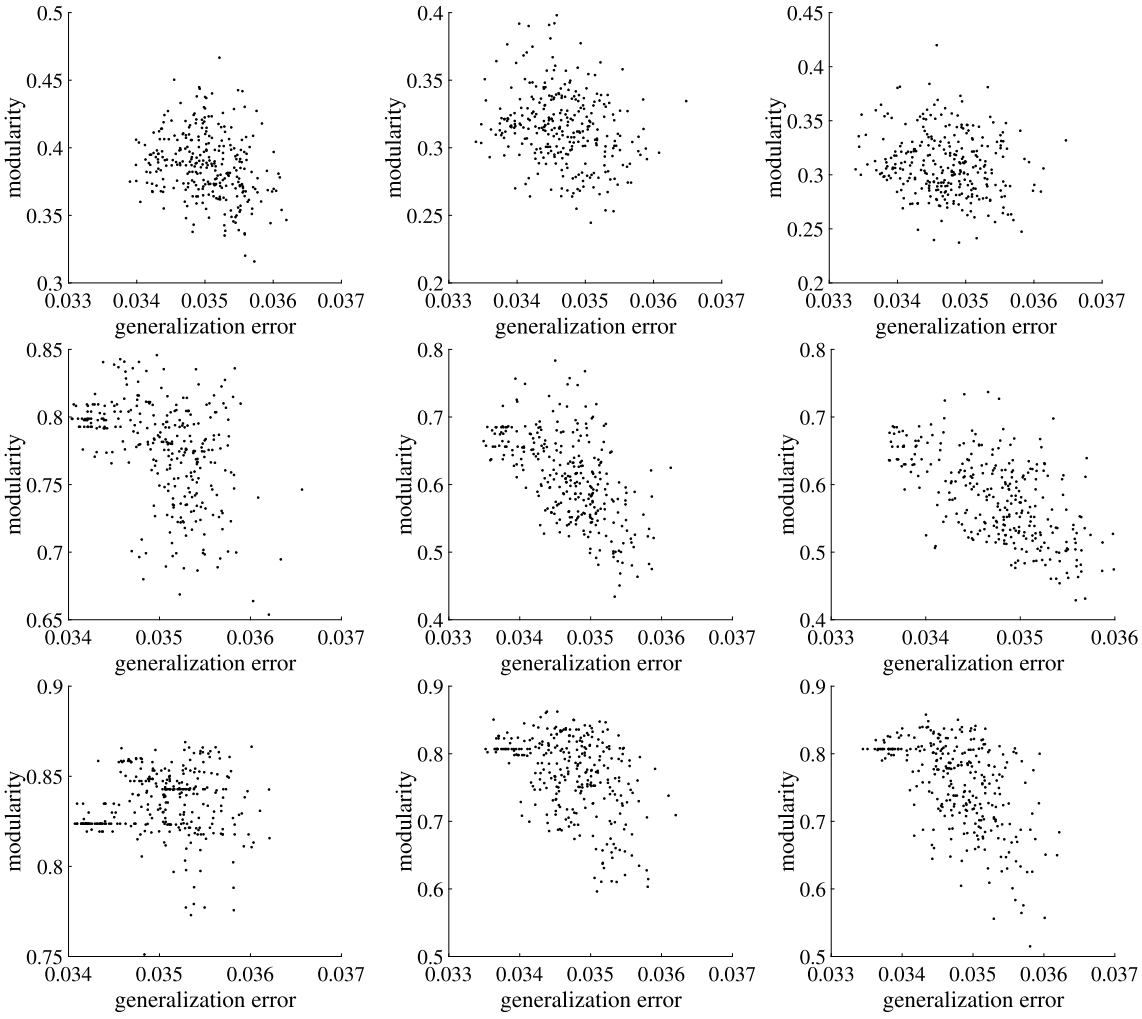

Fig. 8. Relationship between generalization error and modularity. Community detection was performed with layered neural networks trained by multiple independent sets of data. The LASSO hyperparameter λ is 1.0×10^{-5} (left), 1.0×10^{-6} (center) 1.0×10^{-7} (right). The weight removing hyperparameter ξ is 0.1 (top), 0.3 (center) 0.6 (bottom). Better trained results were obtained (with smaller generalization errors) when the trained neural networks had clearer community divisions (with high modularity) except when $(\lambda, \xi) = (1.0 \times 10^{-5}, 0.6)$.

generalization error is expected to be smaller when the weights of the connections between independent sets of input and output are trained to be smaller. Therefore, a higher modularity indicates a smaller generalization error.

In the experiment, we iterated the neural network training and community detection from the trained network 300 times, using 15 dimensional data that are generated in the same way as the experiment described in Section 4.1.1. The generalization error and modularity results for nine pairs of hyperparameters $\{\lambda, \xi\}$ are shown in Fig. 8, where λ is the LASSO hyperparameter and ξ is the weight removing hyperparameter. For the smaller λ and ξ, the overall modularities were lower, which indicates that there were more connections between mutually independent neural networks. It was experimentally shown for some hyperparameters that better trained results were obtained (with smaller generalization errors) when the trained neural networks had clearer community divisions (with higher modularity). Table 1 shows the correlations and the p-values for given $\{\lambda, \xi\}$.

Table 1
The correlation R and the p-value p for the generalization errors and the modularities.

ξ	λ		
	1.0×10^{-5}	1.0×10^{-6}	1.0×10^{-7}
0.1	$R: -0.24, p: 2.1 \times 10^{-5}$	$R: -0.23, p: 5.0 \times 10^{-5}$	$R: -0.17, p: 3.6 \times 10^{-3}$
0.3	$R: -0.43, p: 3.5 \times 10^{-15}$	$R: -0.58, p: 1.5 \times 10^{-28}$	$R: -0.61, p: 1.3 \times 10^{-31}$
0.6	$R: 0.040, p: 0.49$	$R: -0.44, p: 7.9 \times 10^{-16}$	$R: -0.55, p: 5.1 \times 10^{-25}$

Table 2
The correlation R and the p-value p for the generalization errors and the modularities with varying dependence between input data. The parameter α represents the strength of dependence.

α	0	0.1	0.2	0.3	0.4
R	−0.58	−0.72	−0.58	−0.62	−0.56
p	1.5×10^{-28}	5.4×10^{-49}	3.1×10^{-28}	4.9×10^{-33}	1.5×10^{-26}
α	0.5	0.6	0.7	0.8	0.9
R	−0.59	−0.33	−0.32	−0.014	0.14
p	2.6×10^{-29}	6.2×10^{-9}	1.5×10^{-8}	0.81	0.018

4.2.2. Correlation between modularity and generalization error when using input data of correlated dimensions

We also evaluated the relationship between modularity and generalization error, when there is dependence between dimensions of input data. Values of each dimensions of input data were given by

$$x_j^n = \begin{cases} z_j^n & (1 \le j \le 5), \\ (1-\alpha) \times z_j^n + \alpha \times z_{j-5}^n & (6 \le j \le 10), \\ (1-\alpha) \times z_j^n + \alpha \times z_{j-10}^n & (otherwise), \end{cases} \tag{12}$$

where α is a control parameter of dependence in input data and $z_j^n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 3)$.

We varied the parameter α from 0 to 0.9, and checked the correlation between modularity and generalization error for each setting. Here, we set the hyperparameters at $(\lambda, \xi) = (1.0 \times 10^{-6}, 0.3)$, and used the same experimental settings other than the generation method of input data and the hyperparameters. Table 2 shows the correlations and the p-values for varying control parameter α. It was shown that there was a correlation between generalization error and modularity when the dependence between input data was not so strong. Roughly, for the larger α, modularity and generalization error have the weaker correlation. This is because the three sets of input data were not necessarily be decomposed into different communities, even if the training result of neural network was appropriate. If the input data contain strongly dependent dimensions, it is necessary to remove such dimensions in advance, for analyzing the extracted modular structure properly. To construct a method for improving input data appropriately based on their dependency is a future work.

4.3. Knowledge discovery from modular representation

In order to show that the modular representation extracts the global structure of a trained neural network, we applied the proposed method to a neural network trained with practical data. We used data that represent the characteristics of each municipality in Japan (e-stat, 0000). The characteristics shown in Table 3 were used as the input and output data, and the data of municipalities that had any missing value were removed. There were 1905 data, and we divided them into 952 training data and 953 test data. Before the neural network was trained, all the dimensions for all sets of data were converted through the function of $\log(1 + x)$, because the original data are highly biased. The results are shown in Fig. 9.

We iterated the neural network training and community detection from the trained network 300 times, using the above data. The trained neural network and the modular representation with minimum generalization error are shown in Figs. 10 and 11, respectively. The correlation R between modularity and generalization error was −0.028. Fig. 11 shows, for example, that the number of births, deaths, marriages, divorces, people who engage in secondary industry work, and unemployed people (A4) were inferred from the population of transference, the number of outmigrants, households, secondary industry establishments and so on (A1).

From the extracted modular representation, we found not only the grouping of the input and output units, but also the relational structure between the communities of the input, output and hidden layers. For instance, the third community from the right in the depth 2 layer (B2) and the second community from the right in the depth 3 layer (B3) only connected to partial input and output units: they were used only for inferring the number of nuclear family households, single households, nuclear family households with members aged 65 and older, and elderly households (B4), from the population between 15 and 64 years of age, the number of general households and executives (B1).

5. Discussion

In this section, we discuss the proposed algorithm from four viewpoints, the community detection method, the validation of the extracted result, the scalability of our method, and the application.

First, to extract a modular representation from a trained neural network, we employed a basic iterative community detection method for each layer. It is possible to modify this method, for example, by using the weights of connections or the connections in further layers. Utilizing the output of each unit might also improve preciseness of the community detection result. In general, connection weights of a neural network can be trained the more appropriately with the more training data, resulting in the more valid modular representation extraction. For a given set of data, the optimal hyperparameters λ and ξ in the sense of the smallest generalization error can be found by cross-validation, but it takes heavy computational cost. To seek a method for determining the sufficient number of training data, or for optimizing community detection methods and hyperparameters according to the task with small computational complexity is future work.

Second, knowledge discovered from a modular representation depends on both the data and the analyst who utilizes the proposed method. For quantitative evaluation of extracted modular structure, statistical hypothesis test or statistical model selection method is required. However, such method has not been constructed in the field of network analysis, so it is also an important mathematical task in the future. Experimentally, there are both sensitive and robust communities which do and do not depend on them. Therefore, it becomes important to separate the essential

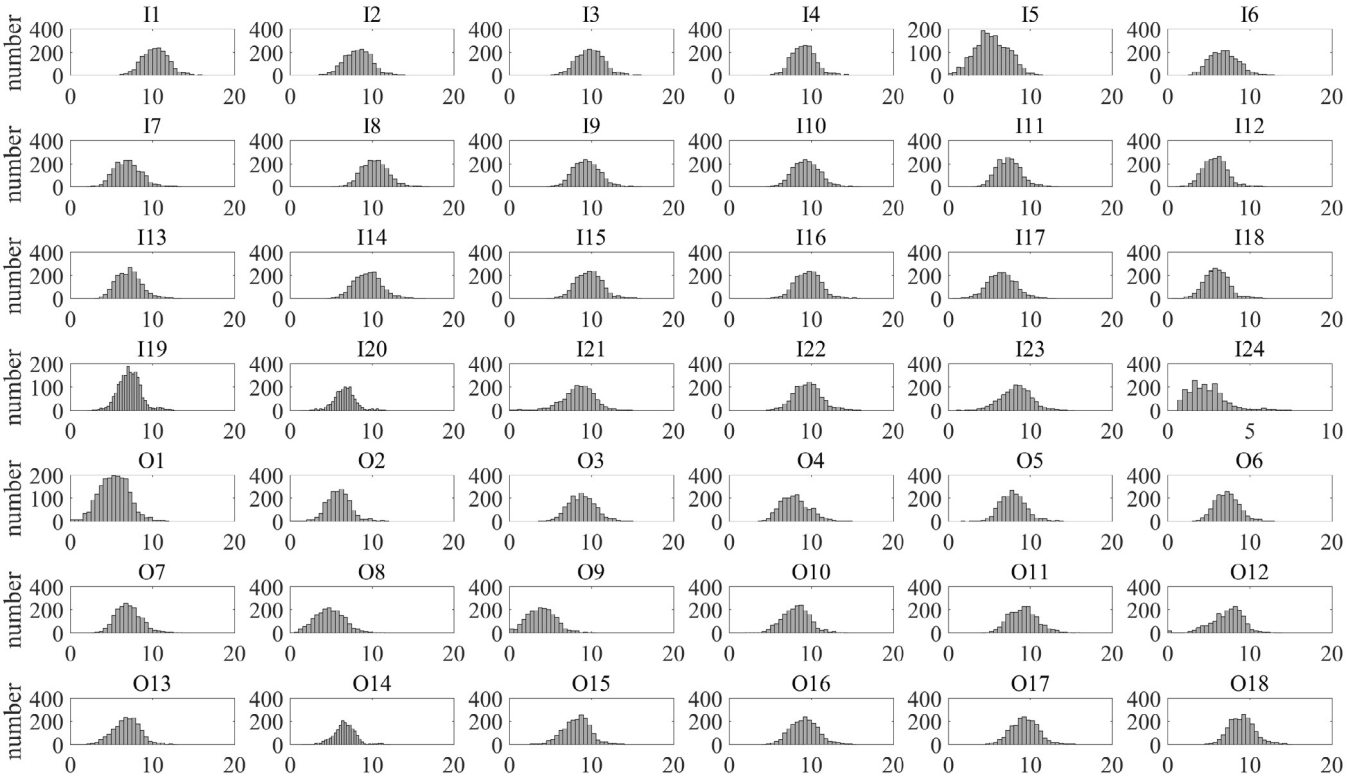

Fig. 9. Histogram of each dimension of data that contain the characteristics of each municipality in Japan. The data were converted through the function of $\log(1+x)$. The notations are shown in Table 3.

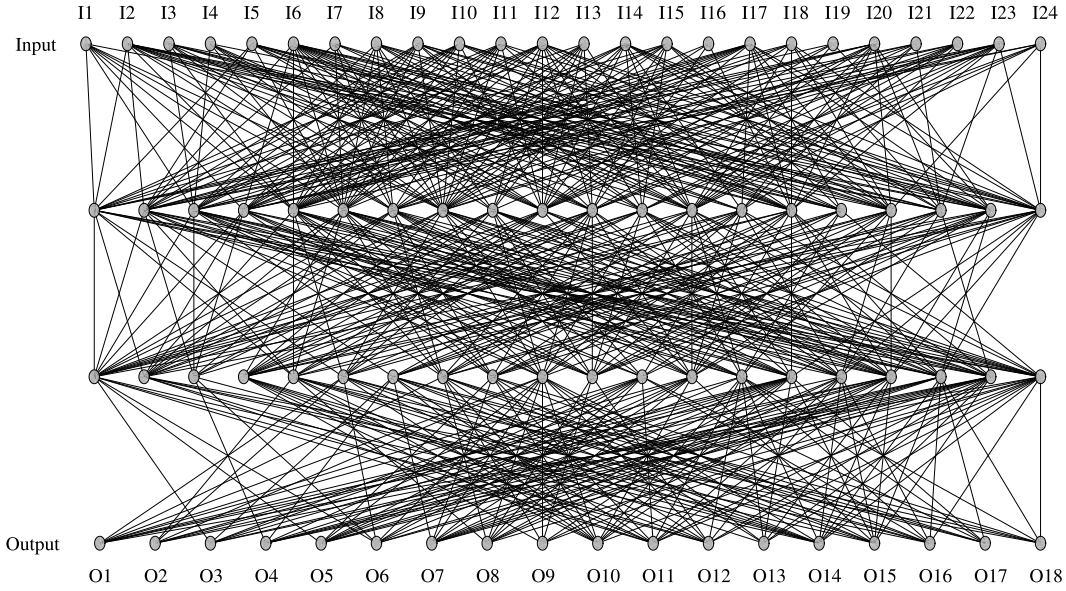

Fig. 10. Trained neural network for practical data. The input and output data notations are shown in Table 3.

results from fluctuations. We anticipate that our method will form the basic analytic procedure of such a study.

Third, in this paper, we experimentally evaluated the relationship between modularity and generalization error. It is well known that a community detection technique can be employed for large size networks. The analysis of larger datasets with higher dimensions would provide further information on layered neural networks. For such large datasets, it would also be important to evaluate the effectiveness of parallel computation, using the independent neural networks extracted with our proposed method.

And lastly, our proposed community detection method can be used for various applications, such as neural network compression. For instance, it would be possible to use modularity index of a resulting community structure as a penalty term for neural network regularization.

6. Conclusion

Layered neural networks have achieved a significant improvement in terms of classification or regression accuracy over a wide

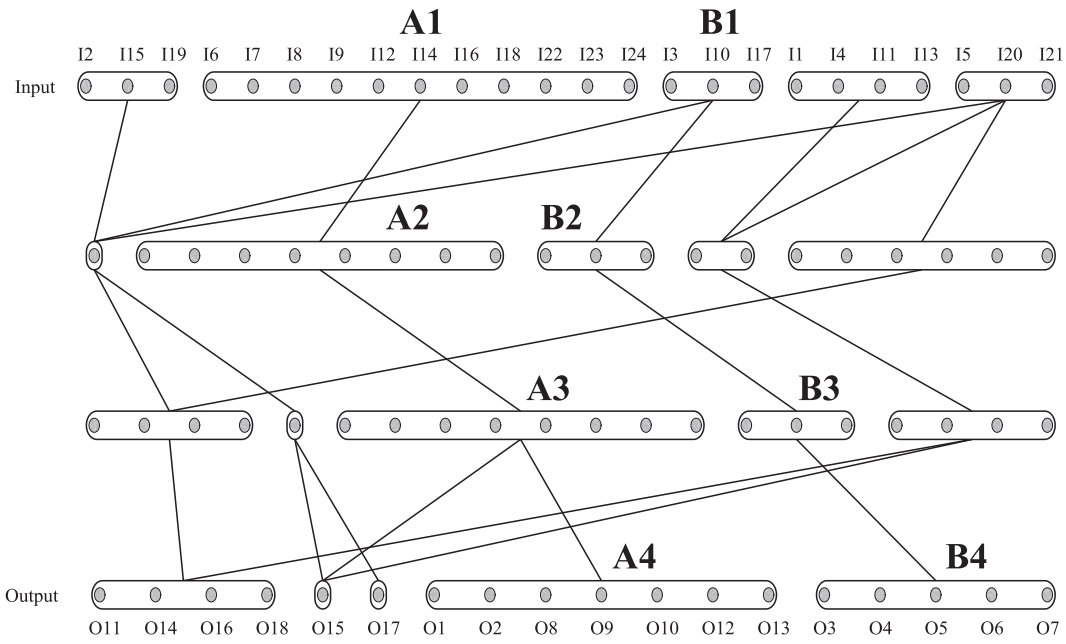

Fig. 11. Extracted modular representation of trained neural network. This figure shows, for example, that the number of births, deaths, marriages, divorces, people who engage in secondary industry work, and unemployed people (A4) were inferred from the population of transference, the number of out-migrants, households, secondary industry establishments and so on (A1).

Table 3
Notations of the data.

Name	Meaning	Name	Meaning
I1	Total population	I22	Number of employed people by business location
I2	Population under 15 years of age	I23	Number of commuters from other municipalities
I3	Population between 15 and 64 years of age	I24	Number of post offices
I4	Population aged 65 and older	O1	Number of births
I5	Foreign population	O2	Number of deaths
I6	Population of transference	O3	Number of nuclear family households
I7	Number of out-migrants	O4	Number of single households
I8	Daytime population	O5	Number of nuclear family households with members aged 65 and older
I9	Number of households	O6	Number of elderly couple households
I10	Number of general households	O7	Number of elderly single households
I11	Number of establishments	O8	Number of marriages
I12	Number of secondary industry establishments	O9	Number of divorces
I13	Number of tertiary industry establishments	O10	Number of secondary industry workers
I14	Number of workers	O11	Number of tertiary industry workers
I15	Labor force population	O12	Number of employees in manufacturing industry
I16	Number of employed people	O13	Number of unemployed people
I17	Number of executives	O14	Number of primary industry employees
I18	Number of employees with employment	O15	Number of secondary industry employees
I19	Number of employees without employment	O16	Number of tertiary industry employees
I20	Number of family workers	O17	Number of employees
I21	Number of commuters to other municipalities	O18	Number of employed workers in their municipalities

range of applications by their ability to capture the complex hidden structure between input and output data. However, the discovery or interpretation of knowledge using layered neural networks has been difficult, since its internal representation consists of many nonlinear and complex parameters.

In this paper, we proposed a new method for extracting a modular representation of a trained layered neural network. The proposed method detects communities of units with similar connection patterns, and determines the relational structure between such communities. We demonstrated the effectiveness of the proposed method experimentally in three applications. (1) It can decompose a layered neural network into a set of small independent networks, which divides the problem and reduces the computation time. (2) The trained result can be estimated by using a modularity index, which measures the effectiveness of a community detection result. And (3) providing the global relational structure of the network would be a clue to discover knowledge from a trained neural network.

Acknowledgment

We would like to thank Akisato Kimura for his helpful comments on this paper.

Appendix

Table 3 shows the notations of the data (e-stat, 0000) used in the experiment described in Section 4.3. The experimental settings of the parameters are shown in Table 4.

Table 4
The experimental settings of the parameters.

Name	Meaning	Exp.1	Exp.2	Exp.3
a_1	Mean iteration number of neural network training per set of data	4000	2000	
n	Number of training datasets	3000	500	952
m	Number of test datasets	0	500	953
$\{l_d\}$	Number of units in depth d layer	{45, 45, 45}	{15, 15, 15}	{24, 20, 20, 18}
D	Number of layers including input, hidden, and output layers	3		4
λ	Hyperparameter of LASSO	1.0×10^{-6}	[a]	1.0×10^{-7}
ϵ	Hyperparameter for convergence of neural network	0.001		
ξ	Weight removing hyperparameter	0.3	[a]	0.5
C	Number of communities per layer	3		5
Method	Method for defining bundled connections	2		3
ζ	Threshold for defining bundled connections	0.3		
$x_{\min}$	Minimum value of normalized input data	-3		-1
$x_{\max}$	Maximum value of normalized input data	3		1

[a] The nine parameters shown in the caption of Fig. 8 are used.

References

Azam, F. (2000). Biologically inspired modular neural networks, Tech. rep.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*, 1798–1828.

Collobert, R., et al. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*, *12*, 2493–2537.

e-Stat, http://www.e-stat.go.jp/SG1/estat/GL08020103.do?_toGL08020103_&tclassID=000001073038&cycleCode=0&requestSender=search .

Estrada, E., & Velázquez, J. (2005). Spectral measures of bipartivity in complex networks. *Physical Review E*, *72*.

Hinton, G., et al. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, *29*, 82–97.

Ishikawa, M. (1990). A structural connectionist learning algorithm with forgetting. *Journal of Japanese Society for Artificial Intelligence*, *5*, 595–603.

Jacobs, R., et al. (1991). Regression shrinkage and selection via the lasso. *Neural Computation*, *3*(1), 79–87.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural network. In *Advances in neural information processing systems* (pp. 1097–1105).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444.

Leung, M., et al. (2014). Deep learning of the tissue-regulated splicing code. *Bioinformatics*, *30*, i121–i129.

Meunier, D., & Paugam-Moisy, H. (2006). Cluster detection algorithm in neural networks, In *European symposium on artificial neural networks*, (pp. 19–24).

Newman, M. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, *103*(23), 8577–8582.

Newman, M., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, *69*.

Newman, M., & Leicht, E. (2007). Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences*, *104*(23), 9564–9569.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

Sainath, T., et al. (2013). Deep convolutional neural networks for LVCSR. In *Acoustics, speech and signal processing* (pp. 8614–8618).

Sutskever, I., Vinyals, O., & Le, Q. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B.*, *58*(1), 267–288.

Tompson, J. J., et al. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems* (pp. 1799–1807).

Werbos, P. (1974). *Beyond regression : New tools for prediction and analysis in the behavioral sciences*. Harvard University, (Ph.D. thesis).

Xiong, H., et al. (2015). The human splicing code reveals new insights into the genetic determinants of disease. *Science*, *347*.