Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)

Realized through a Marriage with Modular-Networks

Tetsuo Furukawa and Kazuhiro Tokunaga

Department of Brain Science and Engineering, Kyushu Institute of Technology
Hibikino, Wakamatsu-ku, Kitakyushu 808-0196, Japan
E-mail: furukawa@brain.kyutech.ac.jp

Abstract— **This paper presents a new development of self-organizing maps (SOM), realized by combining them with the idea of a modular network. This we called a modular network SOM (mnSOM) in which each reference vector unit of a conventional SOM is replaced by a functional module. Since users can choose the functional module from any trainable architecture such as neural networks, the mnSOM is very flexible as well as having a high data processing ability. In this paper, we first introduce the basic idea and then describe its theory. Finally we introduce some applications of mnSOMs.**

Keywords: Self-Organizing Map, SOM, Modular Network, mnSOM, Adaptive Control

I. Introduction

To develop intelligent agents such as autonomous robots, much higher functions must be given to them than those so far realized. For example, such agents need to have a large scale memory system that has an effective learning algorithm without interference between memories, a high adaptability to the changes of context and environment, and an ability to generalize their knowledge from a limited number of experiences. As well, it is important to transcend the dualism of supervised and unsupervised learning schemes. To develop such systems, we need fundamental architectures that provide good platforms on which to build these systems.

In this paper, we introduce a fundamental architecture we call *a modular network SOM* (mnSOM). This is an architecture that will provide good platforms on which to develop intelligent agents. The basic idea of an mnSOM is to combine Kohonen's self-organizing map (SOM) with modular networks. This idea was first proposed by Tokunaga *et al.* [1], [2], and many variations have been developed. Here, the concept and the theory of mnSOM are first described, and then some applications are introduced.

II. Architecture and Algorithm of mnSOM

A. Architecture

The concept of an mnSOM is simple. Every reference vector unit of the Kohonen's SOM is replaced by a functional module of neural network (Fig. 1). Thus, the mnSOM can also be regarded as a kind of modular network in which the modules are arrayed on a lattice. Therefore, the mnSOM has features of both an SOM and a modular network. This strategy has several advantages. First, the mnSOM allows users to deal with not only a set of vector data but also sets of functions, systems, time series, manifolds, and so on. Second, users can design the functional modules depending on their purpose. Users can choose an appropriate module type from a great number of already proposed trainable architectures. Therefore, the mnSOM provides users with a high degree of flexibility and freedom. Third, the theoretical aspects of the conventional SOM, e.g., statistical properties, are consistent with the mnSOM, because the backbone algorithm of an SOM is kept untouched. This aspect assures the theoretical reliability of an mnSOM to users [3].

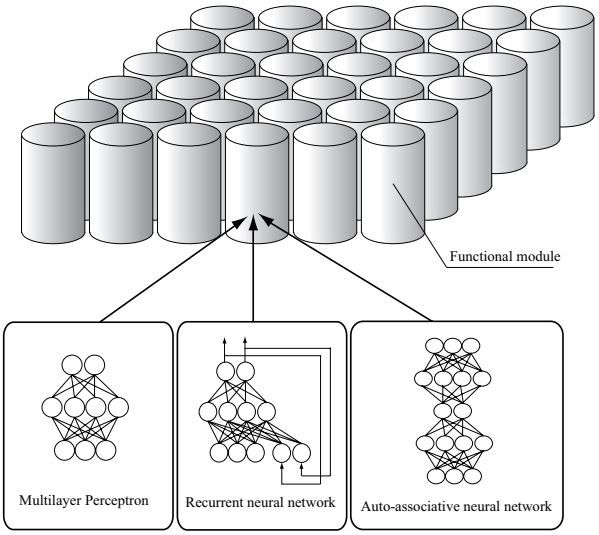

Fig. 1. The concept of modular network SOM (mnSOM)

As an example, let us consider the case in which an mnSOM user wants to make an adaptive controller system using an mnSOM. The user's purpose is to build an mnSOM with lots of controller modules, each of which is specialized to a different context. In such a case, all the user has to do is to (i) determine the architecture of the trainable controller modules, and (ii) define an appropriate distance measure that determines the distance between two controllers. The task of the mnSOM is to train those functional modules under various contexts, while at the same time generating a feature map that indicates similarities or differences between those controllers. If the required controllers under context A and B are similar, then the corresponding controllers should be located near each

other in the map space of the mnSOM, whereas if context C and D require quite different controllers, then those controllers should be arranged further apart. Additionally, the intermediate modules are expected to become controllers for intermediate contexts. The backbone algorithm of an SOM assures such continuity between modules. After the training has finished, the user can use the mnSOM as an assembly of controllers that can adapt to dynamic changes of the context. This is a new aspect that is not found in a conventional SOM, which only generates a static map. These advantages are realized through the synergy of an SOM and a modular network.

B. Framework: Episode and Class

There are two important concepts to describe the algorithm of an mnSOM, *episode* and *class*. *Episode* is a set of data vectors observed together. It is the minimum unit and cannot be divided. Thus, data vectors belonging to the same episode should be processed by the same functional module. On the other hand, *class* is a set of data vectors observed from the same object, e.g., the same system, but this does not mean that they should be observed together.

As an example, suppose that there are static systems A, B, C, $\cdots$. In such a case, *class A* means a set of input-output data vectors observed from system A, i.e., $C_A = \{(\mathbf{x}_{A_1}, \mathbf{y}_{A_1}), \ldots, (\mathbf{x}_{A_n}, \mathbf{y}_{A_n})\}$. Suppose further that there are *episodes* D_1, D_2, $\cdots$, and $D_i = \{(\mathbf{x}_{i,1}, \mathbf{y}_{i,1}), \ldots, (\mathbf{x}_{i,m}, \mathbf{y}_{i,m})\}$. If the episode D_i is observed from system A, then D_i should be a subset of C_A, but there may be other episodes that are observed from system A as well.

It is worth stressing that an episode does not necessarily have *label*, which indicates the class it belongs to. In the case of *unlabeled episode*, the data vectors of an episode should be members of the same class, but there is no information about which class they should belong to. In other words, every data vector has a *tag* that indicates the episode it belongs to, but they do not have labels. In the case of *labeled episodes*, the class information is given to every episode. In this case, every class can be regarded as a sum of episodes. The important point is that the mnSOM can deal with both cases. It is sometimes misunderstood that an mnSOM needs class labels, but this is not true.

Another typical case is where there is only one data sequence observed from an object, e.g., a system, and the property of the object is slowly modulated. In this case, the sequence is assumed to consist of a lot of short period episodes in which the property of the system is assumed to be fixed. The mnSOM can deal with this case as well [4].

C. Algorithm of MLP-mnSOM

As a representative case, let us consider the case in which the mnSOM consists of multilayer perceptron (MLP) modules (MLP-mnSOM). Since each MLP module can represent a nonlinear function, i.e., an input-output relation, the entire mnSOM can generate a map of functions or static systems.

Now suppose that there are I episodes $D_1, \ldots, D_I$ are given, and each episode has J data vectors. Here, $D_i = \{(\mathbf{x}_{i,1}, \mathbf{y}_{i,1}), \ldots, (\mathbf{x}_{i,J}, \mathbf{y}_{i,J})\}$ are a set of input-output vectors sampled from the same static system, and they are assumed to satisfy $\mathbf{y}_{i,j} = f_i(\mathbf{x}_{i,j})$. The tasks of the MLP-mnSOM are; (i) to identify the unknown functions $\{f_i(\cdot)\}$, (ii) to generate a feature map of those functions, and (iii) to map the episodes to the map space of the mnSOM.

The algorithm of an mnSOM consists of four processes: *evaluative process, competitive process, cooperative process* and *adaptive process* [3].

In *the evaluative process*, the outputs of all mnSOM modules are evaluated for each input-output pair. Suppose that an input data vector $\mathbf{x}_{i,j}$ is picked up; the outputs of all modules $\{\tilde{\mathbf{y}}^1_{i,j}, \ldots, \tilde{\mathbf{y}}^K_{i,j}\}$ are then calculated for that input. (Here K denotes the number of modules of the mnSOM. Subscripts mean the indexes of data vectors and episodes, while a superscript indicates the index of modules). This calculation is repeated for all input-output data vector pairs of all episodes. After evaluating all outputs, then the errors of all modules for each episode are evaluated. Now, let E^k_i be the error of the k-th module for the i-th episode, then it is given as

$$E^k_i = \frac{1}{J}\sum_{j=1}^{J} \left\| \tilde{y}^k_{i,j} - y_{i,j} \right\|^2 \tag{1}$$

$$= \frac{1}{J}\sum_{j=1}^{J} \left\| g^k(x_{i,j}) - f_i(x_{i,j}) \right\|^2 . \tag{2}$$

Here, $g^k(\cdot)$ denotes the input-output function of the k-th MLP module. If J is large enough, the distance between $f_i(\cdot)$ and $g^k(\cdot)$ is approximated by the error E^k_i as follows.

$$L^2(f_i, g^k) = \int \left\| g^k(x_{i,j}) - f_i(x_{i,j}) \right\|^2 p(x) dx \simeq E^k_i \tag{3}$$

Here, $p(x)$ is the probability density function of the input. Please note that (3) means the definition of distance between two functions. Therefore, the distance between an episode and an MLP module is measured in function space.

In *the competitive process*, the MLP module that minimizes E^k_i is determined as *the best matching module* (BMM) or *the winner* of the i-th episode. Thus, the index of the BMM, denoted as k^*_i here, is defined as

$$k^*_i \triangleq \arg\min_k E^k_i . \tag{4}$$

Theoretically, k^*_i means the most likelihood estimation of the mapped position of the i-th episode. The BMM is determined for every episode.

In *the cooperative process*, the rate of how much each MLP module should learn for every episode is calculated. Here the rate is called *learning mass*. BMM and its neighbor modules gain larger learning masses than the other modules, as determined by the neighborhood function. Let $m^k_i(t)$ denote the learning mass of the i-th MLP module of the k-th episode at calculation time t. Then m^k_i is given by

$$m^k_i = h\left(d(k, k^*_i);\ t\right). \tag{5}$$

Here $d(k, k_i^*)$ expresses the distance between the k-th module and the BMM of the i-th episode in the map space, i.e., the distance on the lattice of mnSOM. $h(d;\ t)$ is a neighborhood function that shrinks with the calculation time t. *Normalized learning mass* μ_i^k is also calculated as follows.

$$\mu_i^k = \frac{m_i^k}{\sum_{i'=1}^{I} m_{i'}^k} \tag{6}$$

In *the adaptive process*, all MLP modules are updated by the backpropagation learning algorithm as follows.

$$\Delta \mathbf{w}^k = -\eta \sum_{i=1}^{I} \mu_i^k \frac{\partial E_i^k}{\partial \mathbf{w}^k} = -\eta \frac{\partial E^k}{\partial \mathbf{w}^k} \tag{7}$$

Here, $\mathbf{w}^k$ denotes the weight vector of the k-th MLP module and $E^k = \sum_i \mu_i^k E_i^k$. Note that E^k has the global minimum point at which $g^k(x)$ satisfies

$$g^k(x) = \sum_{i=1}^{I} \mu_i^k f_i(x) = \frac{\sum_i m_i^k f_i(x)}{\sum_{i'} m_{i'}^k}. \tag{8}$$

Thus, $g^k(x)$ is updated so as to converge to the mass center of $\{f_i(x)\}$ with the masses $\{m_i\}$. During the training of MLPs, each input vector $\mathbf{x}_{i,j}$ is presented one by one as the input, and the corresponding output $\mathbf{y}_{i,j}$ is presented as the teacher signal. These four processes are iterated until the network gets to a steady state with shrinking of the neighborhood size.

Theoretically, the above process is identical with the batch-learning algorithm of a conventional SOM (BL-SOM) with an exception. The exception is the definition of the distance; in the case of the conventional SOM, the distance is measured in Euclid space, whereas the distance is measured in function space in the case of an mnSOM. Fig. 2 shows a comparison between the maps generated by a conventional SOM and the MLP-mnSOM. The left map was generated by the mnSOM directly from the given episodes, whereas the right map was generated by the conventional BL-SOM by giving the coefficient vectors of orthonormal expansion, which are estimated from the observed episodes. Thus, in the case of the BL-SOM, each episode should be first fitted by linear sum of orthonormal basis functions $\{\phi_j(x)\}$, i.e.,

$$f_i(x) = a_{i,0}\phi_0(x) + a_{i,1}\phi_1(x) + \cdots + a_{i,n}\phi_n(x). \tag{9}$$

Then a set of coefficient vector $\{\mathbf{a}_i\} = \{(a_{i,0}, \ldots, a_{i,n})\}$ are given to the conventional SOM as a set of data vectors. Finally, the functions represented by the SOM are regenerated as

$$g^k(x) = v^{k,0}\phi_0(x) + \cdots + v^{k,n}\phi_n(x), \tag{10}$$

where $\mathbf{v}^k = (v^{k,0}, \ldots, v^{k,n})$ means the k the reference vector of the BL-SOM. Under this situation, the distance measure of function space is identical to the one in coefficient vector space.

$$L^2_{\text{function}}\left(f_i(x), g^k(x)\right) = \int_{-\infty}^{\infty} \left(f_i(x) - g^k(x)\right)^2 p(x)dx \tag{11}$$

$$= \sum_{j=0}^{n} \left(a_{i,j} - v^{k,j}\right)^2 \tag{12}$$

$$= L^2_{\text{coefficient vector}}(\mathbf{a}_i, \mathbf{v}^k) \tag{13}$$

Therefore, the equivalence between the mnSOM and the BL-SOM with orthonormal expansion are theortically assured. In the case of Fig. 2, episodes are assumed be observed from a set of polynomial functions, and normalized Legendre functions are used as the orthonormal basis functions. The two maps are identical, like twins, because of the theoretical equivalence. However, it is important to note that such orthonormal expansion *before training* is difficult in many practical cases, since the functions are usually unknown. In the case of the mnSOM, these functions are estimated in parallel with generating a map of them, and there is no necessary to identify them in advance. This is an important advantage of an mnSOM.

D. Naive generalization of mnSOM algorithm

Now let us consider more generalized module architecture cases of an mnSOM. Suppose that an mnSOM user has a set of episodes $\mathcal{D} = \{D_1, \ldots, D_I\}$, and each of these has J data vectors, i.e., $D_i = \{\mathbf{r}_{i,1}, \ldots, \mathbf{r}_{i,J}\}$. In the previous case, $\mathbf{r}_{i,j} = (\mathbf{x}_{i,j}, \mathbf{y}_{i,j})$. To simplify the situation, let us assume that these episodes are observed from a set of objects $O = \{O_1, \ldots O_I\}$. In a conventional SOM, each data vector is a mapping object, whereas in the case of an MLP-mnSOM, each object corresponds to each of the nonlinear functions.

There is an essential difference between a conventional SOM and the mnSOM. In the conventional case, all the mapping objects, i.e., the data vectors, are known and there is no need to estimate the objects. On the other hand, it often happens that the entities of the objects are unknown in the case of a generalized mnSOM. Therefore, the user should identify those objects in parallel with generating their self-organizing map. Thus the mnSOM should solve the simultaneous estimation problem.

Let us suppose that the mnSOM has K functional modules $\{M^1, \ldots, M^K\}$, which are designed to have the ability to regenerate, or mimic the objects. In other words, a module is capable of approximating an object O_i after training by the episode D_i. Suppose further that the property of each functional module M^k is determined by a parameter vector $\mathbf{w}^k$. Under this situation, the tasks of the mnSOM are; (i) to identify the objects $\{O_i\}$ from the episodes $\{D_i\}$, and (ii) to generate a map of those objects. These two tasks should be processed in parallel.

The most straightforward and naive generalization of the mnSOM algorithm is as follows. Let $\tilde{\mathbf{r}}_{i,j}^k$ be an approximation of $\mathbf{r}_{i,j}$ by the k-th module. Then the average error between the

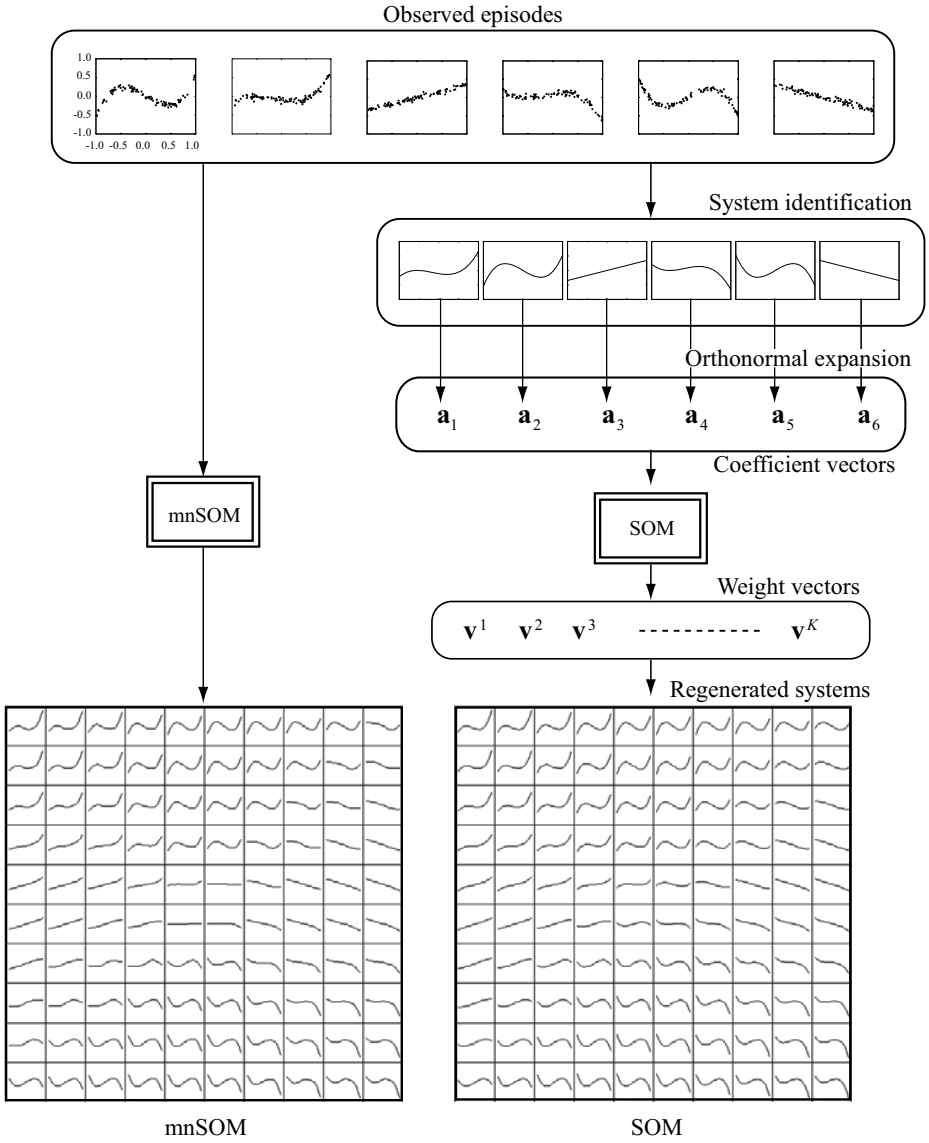

Fig. 2. A comparison of polynomial function maps generated by mnSOM (left) and SOM (right).

i-th episode and the k-th module E_i^k is measured by

$$E_i^k = \frac{1}{J}\sum_{j=1}^{J} \left\| \tilde{\mathbf{r}}_{i,j}^k - \mathbf{r}_{i,j} \right\|^2 . \tag{14}$$

The BMM is then determined by (4), and the learning mass $\{m_i^k\}$ and the normalized learning mass $\{\mu_i^k\}$ are calculated by (5) and (6), respectively. Finally, every module is trained by using episodes with the learning mass $\{\mu_i^k\}$. If the module algorithm is described by the gradient descendent method, then the modules are updated as follows.

$$\mathbf{w}^k = -\eta \sum_{i=1}^{I} \mu_i^k \frac{\partial E_i^k}{\partial \mathbf{w}^k} \tag{15}$$

This is the *naive* algorithm of a generalized mnSOM. This naive generalization may look all right, and actually it has been often used in the past works. In fact the naive algorithm would work appropriately in many cases, but not always. Though it is worth trying this naive version, users are recommended to examine the relevance in the following way.

E. Generalized mnSOM Algorithm

Now let us consider the true generalization of the mnSOM algorithm, which includes naive cases. In the naive case, the distance measure is defined by the average error between a data vector and a module output. To obtain a more theoretically plausible algorithm, we must consider the distance measured between an object O_i and a module M^k. Thus, users need to define an appropriate distance measure $L(O_i, M^k)$ that signifies the difference between an entire object O_i and an entire model obtained by M^k, instead of the difference between

an individual data vector and the corresponding output of a module. Since the distance depends on how the user wants to define the differences between two objects, the measure should be defined depending on the user's purpose.

By using the distance measure, the definition of *mass center* can be determined as follows.

$$\bar{O} \triangleq \arg\min_{O} \sum_{i=1}^{I} m_i L^2(O_i, O). \tag{16}$$

Here $\bar{O}$ is the mass center of the given objects $\{O_1, \dots O_I\}$ with the masses $\{m_i\}$. If O belongs to a vector space, then $\bar{O}$ is given by

$$\bar{O} = \frac{m_1 O_1 + \cdots m_I O_I}{m_1 + \cdots m_I} = \sum_{i=1}^{I} \mu_i O_i. \tag{17}$$

Here μ_i^k denotes the normalized mass given by $\mu_i = m_i / \sum_{i'} m_{i'}$.

Since each object O_i is assumed to be unknown, we can measure only the distance between an estimated object and a module, i.e., $L^2(\tilde{O}(D_i), M^k)$. Here $\tilde{O}(D_i)$ is the object estimated from the i-th episode D_i.

Each module is updated so as to be the mass center, the mass of which is given by the neighborhood function. Thus, the updated algorithm is formulated as

$$\mathbf{w}^k(t+1) = \arg\min_{\mathbf{w}} \sum_{i=1}^{I} m_i^k L^2\left(\tilde{O}(D_i), M(\mathbf{w})\right). \tag{18}$$

When the estimated distance $L^2(\tilde{O}(D_i), M^k)$ can be approximated by the mean square error, i.e.,

$$L^2\left(\tilde{O}(D_i), M^k\right) \simeq \frac{1}{J} \sum_{j=1}^{J} \left\| \tilde{\mathbf{r}}_{i,j}^k - \mathbf{r}_{i,j} \right\|^2, \tag{19}$$

then the naive algorithm is obtained.

Typical cases that require this generalization is an SOM-module-mnSOM, that is called SOM^2, and an mnSOM with autoassociative neural network modules.

III. Variations of mnSOMs and their Applications

In this section, we introduce some representative module architectures of mnSOMs and their applications.

A. MLP-mnSOM

MLP-module-mnSOM (MLP-mnSOM) is a typical architecture of the mnSOM family. Tokunaga *et al.* applied an MLP-mnSOM to the analysis of weather dynamics [2], [3]. In their case, the task of the mnSOM was to predict tomorrow's weather from the weather of the past three days. Therefore, their mnSOM can be regarded as a huge assembly of weather forecast systems. They showed that the geographic topology was preserved in the map of the weather dynamics.

An MLP-mnSOM is also applied to adaptive controllers. Minatohara *et al.* proposed a concept of *self-organizing adaptive controller* (SOAC) [5] as an extension of MOSAIC proposed by Wolpert and Kawato [6]. In the SOAC, each module consists of a pair of a controller and a predictor; and BMM is determined by the predictor block. Thus, the module that predicts best becomes the winner, i.e., the BMM, and the target object is controlled by the controller block of the winner module. They applied a SOAC to controlling an inverted pendulum, the mass and the length of the pendulum suddenly changed. Nishida *et al.* also proposed architecture similar to that of an SOAC based on an MLP-mnSOM. He applied the mnSOM to an underwater autonomous vehicle [7].

B. RNN-mnSOM

Recurrent neural networks (RNN) such as Elman Net and Jordan Net are available as functional modules of an mnSOM. In this case, the RNN-mnSOM can generate a map of dynamical systems. Kaneko *et al.* reported that an RNN-mnSOM generates a map of neuron models [8]. They gave some firing patterns generated by a Bonhoefer-van der Pol (BVP) model with different synaptic parameters. As the result, the mnSOM generated an appropriate map that indicates the changes of the synaptic parameter value. They also proposed a concept of *self-organizing bifurcation map* (SOBM).

Aziz *et al.* applied an RNN-mnSOM to a mobile robot, trying to articulate a sequence of sensor data when a robot moves [9]. In their model, a continuous sequence is articulated by the discrete transition of BMM.

C. ANN-mnSOM

A mnSOM with autoassociative neural network (ANN) modules is a variation of an MLP-mnSOM. An ANN has a sand clock like symmetrical structure, the task of which is to regenerate input vectors in the output layer. It is known that a 3-layer ANN can represent a data distribution in a linear subspace, while a 5-layer ANN can approximate a nonlinear subspace, i.e., a manifold. Thus, an ANN-mnSOM is an SOM of manifolds.

Tokunaga *et al.* showed that a 5-layer-ANN-mnSOM generated a map of periodical waveforms. The ANN-mnSOM generated a map indicating the differences of shapes and the frequencies of the waveforms [10]. They also applied an ANN-mnSOM to texture classification task.

D. SOM^2: SOM-module-mnSOM

SOM-module-mnSOM proposed by Furukawa is a main member of mnSOM, and it also forms another family of generalized SOM. He calls this architecture *SOM*^{2}, because an SOM^2 represents a product manifold, namely, SOM×SOM [11], [12], [13]. Like the case of the ANN-mnSOM, SOM^2 also generates a map of manifolds, but the ability of the SOM^2 is much higher than that of the ANN-mnSOM.

The essence of the algorithm of SOM^2 is as follows. Suppose that $\mathbf{w}^{k,l}$ is the l-th reference vector of the k-th SOM module, and $\mathbf{x}_{i,j}$ is the j-th data vector of the i-th episode. Then the reference vectors of SOM^2 are updated as

$$\mathbf{w}^{k,l}(t+1) = \sum_{i=1}^{I} \sum_{j=1}^{J} \alpha_i^k(t) \beta_{i,j}^l(t) \mathbf{x}_{i,j}. \tag{20}$$

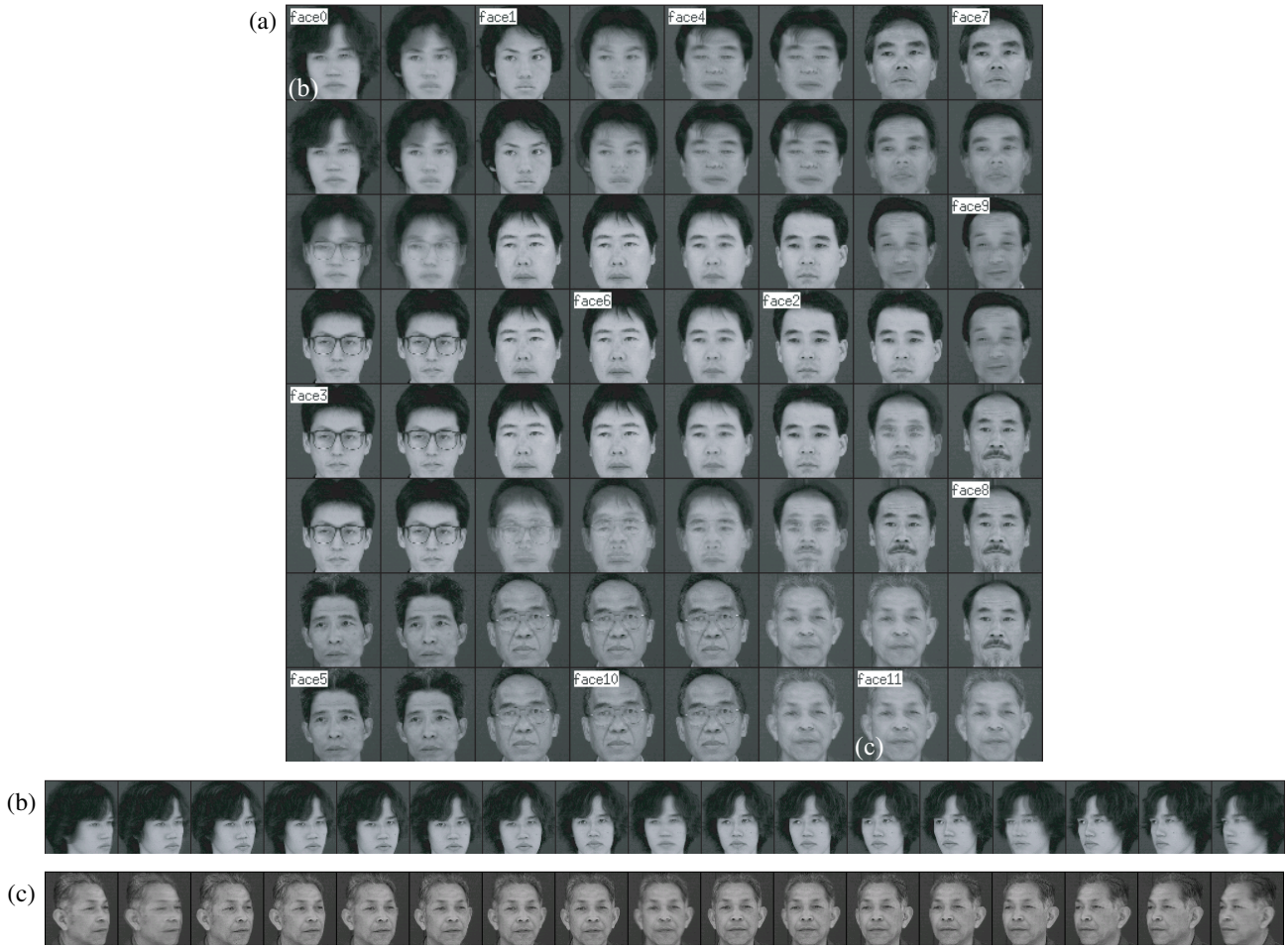

Fig. 3. 'Face map' generated by SOM^2. The parent map (a) and the child maps of face0 (b) and face11 (c).

Here α_i^k is the normalized learning mass of the k-th SOM module for the i-th episode, while $\beta_{i,j}^l$ denotes the normalized learning mass of the l-th reference vector for $\mathbf{x}_{i,j}$, i.e.,

$$\alpha_i^k(t) = \frac{h_\alpha\left(d(k, k_i^*),\, t\right)}{\sum_{i'=1}^{I} h_\alpha\left(d(k, k_{i'}^*),\, t\right)} \tag{21}$$

$$\beta_{i,j}^l(t) = \frac{h_\beta\left(d(l, l_{i,j}^{*,*}),\, t\right)}{\sum_{j'=1}^{J} h_\beta\left(d(l, l_{i,j'}^{*,*}),\, t\right)}. \tag{22}$$

Here $h_\alpha(\cdot)$ and $h_\beta(\cdot)$ are the neighborhood function between SOM modules and the one between reference vectors within an SOM module respectively. $l_{i,j}^{*,*}$ is the index of the winner unit for the $\mathbf{x}_{i,j}$ within the winner module of the i-th episode. Theoretically, SOM^2 organizes a higher-order map, namely, homotopy. Therefore SOM^2 can be regarded as a 'Self-Organizing Homotopy Network'. From another point of view, a conventional SOM represents a data distribution by a manifold, whereas an SOM^2 represents a set of data distributions by a fiber bundle [13].

Furukawa showed that an SOM^2 generated a map of 3D objects from a set of 2D projected images, e.g., photographs [11]. Fig. 3 shows a map of various faces generated by an SOM^2. In this case, every episode consists of a set of photographs taken from a person. Raw image data without any preprocessing are used as the data vectors, and no pose information is given. In this case, each SOM modules (child SOM) had one-dimensional topology, while the entire mnSOM (parent SOM) had two dimensional map space. As shown in Fig. 3, the SOM^2 succeeded to make a map of faces, in which each SOM module represented continuous pose change of a person, while the entire SOM^2 represented a map of various faces. Furthermore, the reference vectors of SOM modules with same index represent face images with same pose. Therefore, it is expected that SOM^{2}s will be applied to simultaneous estimations of face and pose.

A prominent property of SOM^2 is that it is allowed to make a nested structure like a Russian doll. For example, SOM^2 can be a module of a meta-mnSOM. Thus, SOM^2-module-

mnSOM, i.e., SOM^3, is also possible. SOM^3 represents a product manifold of SOM×SOM×SOM. In the case of SOM^3, the update algorithm is described as follows.

$$\mathbf{w}^{l,m,n}(t+1) = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} \alpha_i^l(t)\,\beta_{i,j}^m(t)\,\gamma_{i,j,k}^n\,\mathbf{x}_{i,j,k} \tag{23}$$

$$\alpha_i^l(t) = \frac{h_\alpha\left(d(l,l_i^*),\,t\right)}{\sum_{i'=1}^{I} h_\alpha\left(d(l,l_{i'}^*),\,t\right)} \tag{24}$$

$$\beta_{i,j}^m(t) = \frac{h_\beta\left(d(m,m_{i,j}^{*,*}),\,t\right)}{\sum_{j'=1}^{J} h_\beta\left(d(m,m_{i,j'}^{*,*}),\,t\right)} \tag{25}$$

$$\gamma_{i,j,k}^n(t) = \frac{h_\gamma\left(d(n,n_{i,j,k}^{*,*,*}),\,t\right)}{\sum_{k'=1}^{J} h_\gamma\left(d(n,n_{i,j,k'}^{*,*,*}),\,t\right)} \tag{26}$$

It is easy to extend the n-th order cases, namely, SOM^n as SOM^{n-1}-module-mnSOM that represents $n-1$ order homotopy.

E. NG×SOM: NG-module-mnSOM

It is also possible to employ other types of vector quantization algorithms such as neural gas (NG) networks. Furukawa also proposed a NG-module-mnSOM called NG×SOM [11], [13].

As a variation of an mnSOM, a "modular network NG (mnNG)" is also possible [11]. In this case, all the reference vectors of a NG network are replaced by functional modules of neural networks. If one employs MLPs as the functional modules, then one obtains a MLP-mnNG. When the module architecture is SOM or NG, then SOM×NG and NG^2=NG×NG are obtained, which mean "SOM-module-mnNG" and "NG-module-mnNG".

F. Other module types

Many other neural architectures are also possible. For example, radial basis function modules would be a better substitute for MLP-mnSOM. Stochastic type networks such as a Boltzmann machine and a Hopfield network make up another group. These are expected to generate a 'map of memories'.

The mnSOM includes some variations of SOM that have been proposed previously. If one employs a linear operator module (e.g., single layer perceptron), then it becomes an Operator Map as proposed by Kohonen [14]. When a principle component analysis (PCA) module is used, then the mnSOM becomes an adaptive subspace SOM (ASSOM) [15]. If one employs Hebbian neurons as the functional modules, then the mnSOM becomes a conventional SOM. Therefore, a mnSOM is a generalization of a SOM rather than an extension because it includes the conventional cases.

IV. Conclusion

In this paper, the concept and theory of mnSOMs are described and some applications introduced. To return to the issue of how we can develop fundamental architectures to realize intelligent agents. From this point of view, the mnSOM has several advantages. (i) It can process much larger tasks than single neural networks. (ii) It has less interference of memories because of its modular structure. (iii) The mnSOM has a manner that reflects both aspects of supervised and unsupervised learning . (iv) Users can produce many variations by designing the functional modules depending on their purposes. In addition, the architecture of mnSOM is suitable for parallel processing, and it would be a solution of computational time problem.

While the mnSOM can be powerful tool in many fields as standalone architecture, the mnSOM can also possible to be a part of a larger architecture of artificial intelligence. For example, one can add a sequence learning algorithm, which memorizes the transition of the BMM of an mnSOM. Considering these points, the mnSOM is expected to be a good platform for realizing intelligent agents.

Acknowledgments

This work was supported (in part) by the 21th Century Center of Excellence Program to Kyushu Institute of Technology (Center #J19) by the Japanese MEXT. The facial data in this paper are used by permission of Softopia Japan, Research and Development Division, HOIP Laboratory.

References

[1] K. Tokunaga and T. Furukawa, "Modular network SOM: Extension of som to the realm of function space," in *Proceedings of Workshop on Self-Organizing Maps 2003 (WSOM2003)*, Kitakyushu, Japan, 2003, pp. 173–178.

[2] K. Tokunaga, T. Furukawa, and S. Yasui, "Modular network SOM: Self-organizing maps in function space," *Neural Information Processing – Letters and Reviews*, vol. 9, no. 1, pp. 15–22, 2005.

[3] T. Furukawa, K. Tokunaga, K. Morishita, and S. Yasui, "Modular network SOM (mnSOM): From vector space to function space," in *Proceedings of International Joint Conference on Neural Networks 2005 (IJCNN2005)*, Montreal, Canada, 2005, pp. 1581–1586.

[4] K. Tokunaga and T. Furukawa, *Modular network SOM: Theory, algorithm and applications*, ser. Lecture Notes in Computer Science. Heidelberg: Springer Berlin, 2006, vol. 4232, pp. 958–967.

[5] T. Minatohara and T. Furukawa, "Self-organizing adaptive controllers: Application to the inverted pendulum," in *Proceedings of Workshop on Self-Organizing Maps 2005 (WSOM2005)*, Paris, France, 2005, pp. 41–48.

[6] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*, vol. 11, pp. 1317–1329, 1998.

[7] S. Nishida, K. Ishii, and T. Furukawa, *An Online Adaptation Control System Using mnSOM*, ser. Lecture Notes in Computer Science. Heidelberg: Springer Berlin, 2006, vol. 4232, pp. 935–942.

[8] S. Kaneko, K. Tokunaga, and T. Furukawa, "Modular network som: The architecture, the algorithm and applications," in *Proceedings of Workshop on Self-Organizing Maps 2005 (WSOM2005)*, Paris, France, 2005, pp. 537–544.

[9] M. A. Muslim, M. Ishikawa, and T. Furukawa, "A new approach to task segmentation in mobile robots by mnSOM," in *Proceedings of 2006 IEEE World Congress on Computational Intelligence (IJCNN2006 Section)*, Vancouver, Canada, 2006, pp. 6542–6549.

[10] K. Tokunaga and T. Furukawa, "Nonlinear ASSOM constituted of autoassociative neural modules," in *Proceedings of Workshop on Self-Organizing Maps 2005 (WSOM2005)*, Paris, France, 2005, pp. 637–644.

[11] T. Furukawa, "SOM^2 as SOM of SOMs," in *Proceedings of Workshop on Self-Organizing Maps 2005 (WSOM2005)*, Paris, France, 2005, pp. 545–552.

[12] ——, *SOM of SOMs: Self-organizing map which maps a group of self-organizing maps*, ser. Lecture Notes in Computer Science. Heidelberg: Springer Berlin, 2005, vol. 3696, pp. 391–396.

[13] ——, *SOM of SOMs: An extension of SOM from map to homotopy*, ser. Lecture Notes in Computer Science. Heidelberg: Springer Berlin, 2006, vol. 4232, pp. 950–957.

[14] T. Kohonen, "Generalizations of the self-organizing map," in *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, vol. 1, 1993, pp. 457–462.

[15] T. Kohonen, S. Kaski, and H. Lappalainen, "Self-organization formation of various invariant-features filters in the adaptive subspace SOM," *Neural Computation*, vol. 9, pp. 1321–1344, 1993.