Modular Network SOM:
The Architecture, the Algorithm and Applications to Nonlinear Dynamical Systems

S. Kaneko **K. Tokunaga** **T. Furukawa**
Department of Brain Science and Engineering, Kyushu Institute of Technology
2-4 Hibikino, Wakamatsu-Ku, Kitakyushu 808–0196, Japan
kaneko-syuji@edu.brain.kyutech.ac.jp
tokunaga-kazuhiro@edu.brain.kyutech.ac.jp
furukawa@brain.kyutech.ac.jp

Abstract - *This paper proposes a generalized framework of SOM applicable to more extended data classes rather than vector data. The generalization is realized by adopting the idea of a modular network; thus it is called a modular network SOM (mnSOM), in which each nodal vector unit in a conventional SOM is replaced by a function module such like a neural network or a trainable algorithm. First, we introduce the basic idea of an mnSOM, and then by focusing on a class of mnSOM that consists of multi-layer perceptron modules, the architecture and the algorithm are presented. Finally, some applications to nonlinear dynamical systems are reported.*

Key words - modular network, mnSOM, Operator Map, ASSOM, bifurcation map

1 Introduction

Kohonen proposed the idea of the self-organizing map more than 20 years ago, but fields of application are still spreading in today's information driven society [1,2]. Despite its increasing importance, the conventional SOM and the most of its extensions can only deal with vectorized data. If one wishes to deal with a non-vector dataset, then one needs to make the data vectorized in advance, otherwise one needs to modify the SOM itself to adapt the data type. Therefore, generalizing the SOM family is one of the unavoidable problems, in which the SOM algorithm is described independently of the data type.

Our study aims to develop the generalized framework of the conventional SOM by adopting the idea of a modular network, in what we call a *modular network SOM (mnSOM)* [3,4]. The idea of an mnSOM is simple; every vector unit of the conventional SOM is replaced by a trainable function module such like a neural network. Thus the architecture of the mnSOM is an assembly of the function modules arrayed on a lattice (Fig. 1). The function modules can be designed to suit each application, while keeping the backbone algorithm of the SOM untouched. This generalization strategy provides both a high-degree of design flexibility and reliability to SOM users, because the mnSOM allows one to choose the function modules from the great number of neural architectures already proposed; and at the same time, the consistent extension method assures the theoretical consistency, e.g. statistical properties, of the results. Furthermore, this strategy gives the functional capability of data processing

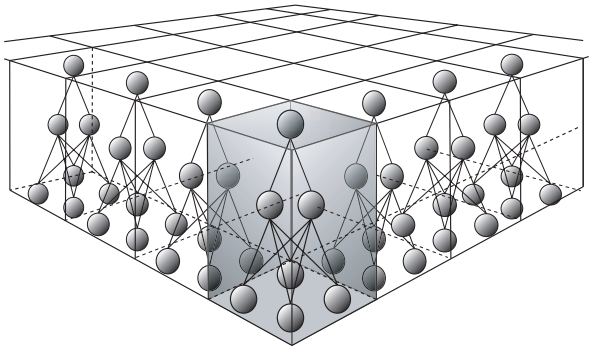

Figure 1: The architecture of mnSOM

to all nodal units of the SOM. Thus, our mnSOM can be used as an assembly of data processors after training. This is another advantage of an mnSOM.

As an example, let us consider the case that someone is trying to make a map of the weather dynamics of the world's cities [3,4]. In this case, the task is to generate a feature map that visualizes the degree of similarities or differences between the weather dynamics of each city; if the weather of New York and Tokyo are described by similar dynamics, then both cities are expected to be the neighbors in the weather map, whereas if London and Singapore have different dynamics, then they will be mapped far apart. Of course one could make a map by using a conventional SOM after vectorizing all the weather data; however, such map would provide no information about the relationships between the dynamics. To achieve this task, one should identify the weather dynamics of every city, and then generate a feature map by comparing the dynamics. This is what one can do with an mnSOM. By using an mnSOM, all one has to do is; (i) design a neural network that can learn the weather dynamics of a single city, (ii) employ the neural network as the function module of the mnSOM, and (iii) train the mnSOM by entering world wide weather data. In addition, the mnSOM can be used as an assembly of weather forecast systems after training.

Similar ideas have already been proposed as the Operator Map and ASSOM [5,6], but our purpose is to establish a more general and more flexible framework that takes over the operator case. Indeed, the multi-layer perceptrons (MLPs) [3,4], the recurrent neural networks (RNNs) [7], SOMs, Neural Gases (NGs), etc. are all available to act as function modules of an mnSOM. Operator Map and ASSOM can be regarded as mnSOMs with linear operator modules and PCA modules respectively, whereas an mnSOM with hebbian neuron modules becomes a conventional SOM.

In this paper, first the general idea of our mnSOM is introduced. Then focusing on a class of mnSOM, i.e. the MLP-module-mnSOM, the architecture and the algorithm are presented along with some applications to nonlinear dynamical systems.

2 Theoretical Framework

The architecture of our mnSOM is shown in Fig. 1. Basically, the architecture is such that each vector unit of the conventional SOM is replaced by a function module. These modules are arrayed on a lattice that represents the coordinates of the feature map. Many trainable algorithms can be employed as function modules, though Fig. 1 illustrates an MLP-module case. Since each module represents a certain '*functional feature*' determined by the module architecture, the entire mnSOM represents

a map of functions. Therefore, one might draw analogies between the mnSOM and the functional column structure or the functional map of the cortex.

Suppose that an mnSOM user is trying to map a set of I objects, $\{C_1, \ldots, C_I\}$. Here we call the mapping object '*atom*', since it is the most elementary unit in the map space. In the conventional SOM, each data vector is the atom, whereas in the previous example of the weather map, each atom corresponds to each of the weather dynamics. It is not necessary that the entities of atoms are identified in advance, i.e., they are generally assumed to be unknown. For example, the equation describing each of the weather dynamics is the entity of the atom, which is usually unknown. Instead, a sample dataset observed from each atom is assumed to be available. Thus, let $D_i = \{\boldsymbol{r}_{i,1}, \ldots, \boldsymbol{r}_{i,J}\}$ be the dataset observed from the ith object C_i. In this paper, it is assumed that $\{\boldsymbol{r}_{i,j}\}$ are vector data.

Now let the mnSOM have K function modules $\{M^1, \ldots, M^K\}$, which are designed to have an ability of regenerating, or mimicking the atoms. (Here, let the subscripts represent the indexes of mapping objects or datasets, whereas the superscripts mean the indexes of mnSOM.) Suppose that the property of each function module M^k is determined by a parameter set $\boldsymbol{w}^k$; thus in the case of MLP-module-mnSOM, $\boldsymbol{w}^k$ means the weight vector of the kth MLP module. Suppose further that each function module M^k is given a fixed position $\boldsymbol{\xi}^k$ in the map space. Therefore, $\boldsymbol{\xi}^k$ assigns the coordinates of M^k in the map space, while $\boldsymbol{w}^k$ determines the position in the data space.

Under such a situation, the tasks for mnSOM are (i) to identify the entities of $\{C_i\}$ from the observed datasets $\{D_i\}$ by adapting the function modules $\{M^k\}$, and (ii) to generate a map that shows the degree of similarities and differences between the atoms, in parallel. Note that the map generated by the mnSOM is expected to show the relationships between the entities of the atoms; therefore, direct comparisons between the datasets are meaningless for that purpose.

mnSOM users only need to give a definition of the measure representing the similarity between an atom and a function module. Suppose that S_i^k represents the degree of similarity between C_i and M^k, described as follows.

$$S_i^k \triangleq s\left(L(C_i, M^k)\right) \tag{1}$$

Here $L(C_i, M^k)$ is the distance measure and $s(\cdot)$ is a monotonically decreasing function. Both $L(\cdot,\cdot)$ and $s(\cdot)$ can be defined by the user depending on the module architecture. Since C_i is unknown, the similarity measure can be estimated from the observed dataset D_i. One of the natural definitions is to use the mean square error.

$$S_i^k \triangleq -\int \left\|\boldsymbol{r} - \boldsymbol{r}^k\right\|^2 p(\boldsymbol{r}) d\boldsymbol{r} \simeq -\frac{1}{J}\sum_{j=1}^{J}\left\|\boldsymbol{r}_{i,j} - \hat{\boldsymbol{r}}_{i,j}^k\right\|^2 \tag{2}$$

Here, $\hat{\boldsymbol{r}}^k$ is the approximation of $\boldsymbol{r}$ by the module M^k.

The algorithm of the mnSOM consists of four processes: *evaluative process, competitive process, cooperative process* and *adaptive process*. In the *evaluative process*, the similarity measure S_i^k is evaluated for every combination of C_i and M^k. In the following *competitive process*, the module that maximizes S_i^k is determined as the best matching module (BMM) of C_i. Thus, the index of the BMM, denoted as k_i^* here, is defined as follows.

$$k_i^* \triangleq \arg\max_k S_i^k \tag{3}$$

In the *cooperative process*, the degree of C_i belonging to M^k is given by the neighborhood function;

$$P(M^k|C_i) = h\left(\left\|\boldsymbol{\xi}^k - \boldsymbol{\xi}_i^*\right\|; T\right). \tag{4}$$

Here $h(\cdot\,; T)$ is the neighborhood function that shrinks by the calculation time T, whereas $\boldsymbol{\xi}_i^*$ is the position of M_i^* in the map space. By regarding $P(M^k|C_i)$ as a conditional probability, the posterior probability $P(C_i|M^k)$ can be evaluated by Bayes' rule.

$$\psi_i^k \triangleq P(C_i|M^k) = \frac{h\left(\left\|\boldsymbol{\xi}^k - \boldsymbol{\xi}_i^*\right\|; T\right)}{\sum_{i'=1}^{I} h\left(\left\|\boldsymbol{\xi}^k - \boldsymbol{\xi}_{i'}^*\right\|; T\right)} \tag{5}$$

ψ_i^k is calculated for all combinations of $\{C_i\}$ and $\{M^k\}$. At last, all modules are trained to maximize the expectation of the similarity measure in the adaptive process. Let $\langle S^k \rangle$ be the expectation of similarity measure of the kth module, which is given by

$$\langle S^k \rangle = \sum_{i=1}^{I} P(C_i|M^k) S_i^k = \sum_{i=1}^{I} \psi_i^k S_i^k. \tag{6}$$

Then the module parameters $\{\boldsymbol{w}^k\}$ are innovated so as to maximize $\{\langle S^k \rangle\}$ while $\{\psi_i^k\}$ are fixed. If a direct calculation of the best $\{\boldsymbol{w}^k\}$ for maximizing the expectation is impossible, then $\boldsymbol{w}^k$ can be innovated by using the gradient method as follows.

$$\Delta \boldsymbol{w}^k = \eta \frac{\partial \langle S^k \rangle}{\partial \boldsymbol{w}^k} = \eta \sum_{i=1}^{I} \psi_i^k \frac{\partial S_i^k}{\partial \boldsymbol{w}^k} \tag{7}$$

Here η is a small constant that determines the learning rate. If the similarity measure is defined as (2), then the adaptation process described in (7) becomes as follows.

$$\Delta \boldsymbol{w}^k = -\frac{\eta}{J} \sum_{i=1}^{I} \sum_{j=1}^{J} \psi_i^k \frac{\partial \left\|\boldsymbol{r}_{i,j} - \hat{\boldsymbol{r}}_{i,j}^k\right\|^2}{\partial \boldsymbol{w}^k} \tag{8}$$

These 4 processes are iterated, shrinking the neighborhood function area until the network gets to a steady state. Therefore, the algorithm of the mnSOM described above is the expectation maximization (EM) algorithm.

2.1 Algorithm of an MLP-module-mnSOM

To consider the practical aspects of an mnSOM, we hereafter consider the case that the function modules are the MLPs. In this case, each module represents an input-output relation, namely, a function or a system. Consequently, the atoms are also functions or systems. Now suppose that there are I functions $\{f_i(\cdot), \ldots, f_I(\cdot)\}$, which are unknown initially. Instead, the sample datasets $\{D_1, \ldots, D_I\}$ are assumed to be given. Here, $D_i = \{(\boldsymbol{x}_{i,1}, \boldsymbol{y}_{i,1}), \ldots, (\boldsymbol{x}_{i,J}, \boldsymbol{y}_{i,J})\}$ is a set of input-output vectors sampled from the ith function. Thus, the vector pairs satisfy $\boldsymbol{y}_{i,j} = f_i(\boldsymbol{x}_{i,j})$. The tasks of the MLP-module-mnSOM are (i) to identify the unknown functions, and (ii) to generate a feature map of those functions. In other words, the task of the mnSOM is to undertake a topological mapping from a function space to a map space. Therefore, the MLP-module-mnSOM is a SOM in the function space rather than one in the vector space.

To complete the mnSOM algorithm, all one needs to do is to give an appropriate definition of the similarity measure. In this case, the mean square error is employed as the measure. Thus,

$$S_i^k \triangleq -L^2(f_i, g^k) = -\int \left\| f_i(\boldsymbol{x}) - g^k(\boldsymbol{x}) \right\|^2 p_i(\boldsymbol{x}) d\boldsymbol{x} \tag{9}$$

Here $g^k(\cdot)$ denotes the function represented by the kth module M^k, and $p_i(\boldsymbol{x})$ is the probability density function (pdf) of the input $\boldsymbol{x}$ of the ith data class. Since $\{f_i(\cdot)\}$ are unknown, (9) cannot be evaluated directly. However, it is approximated by the mean square error between the output vectors $\{\boldsymbol{y}_{i,j}\}$ and the outputs of modules. Thus,

$$E_i^k \triangleq \frac{1}{J} \sum_{j=1}^{J} \| \boldsymbol{y}_{i,j} - \hat{\boldsymbol{y}}_{i,j}^k \|^2 \simeq -S_i^k \tag{10}$$

Here, $\hat{\boldsymbol{y}}_{i,j}^k$ is the output of M^k for the input $\boldsymbol{x}_{i,j}$, i.e., $\hat{\boldsymbol{y}}_{i,j}^k = g^k(\boldsymbol{x}_{i,j})$. In the competitive process, the module that maximizes S_i^k is determined as the BMM of C_i. It means that the least mean square error module for each class is determined as the BMM. In the following cooperative process, $\{\psi_i^k\}$ are calculated by using the neighborhood function. If a Gaussian function is chosen as the neighborhood function, then ψ_i^k is given as follows.

$$\psi_i^k = \frac{\exp\left[-\left\| \boldsymbol{\xi}^k - \boldsymbol{\xi}_i^* \right\|^2 / 2\sigma^2 \right]}{\sum_{i'=1}^{I} \exp\left[-\left\| \boldsymbol{\xi}^k - \boldsymbol{\xi}_{i'}^* \right\|^2 / 2\sigma^2 \right]} \tag{11}$$

In the adaptive process, the weight vectors $\{\boldsymbol{w}^k\}$ of all modules are innovated as follows.

$$\Delta \boldsymbol{w}^k = \eta \sum_{i=1}^{I} \psi_i^k \frac{\partial S_i^k}{\partial \boldsymbol{w}^k} \simeq -\frac{\eta}{J} \sum_{i=1}^{I} \sum_{j=1}^{J} \psi_i^k \frac{\partial \| \boldsymbol{y}_{i,j} - \hat{\boldsymbol{y}}_{i,j}^k \|^2}{\partial \boldsymbol{w}^k} \tag{12}$$

Therefore, the learning algorithm of MLP modules is given by the modified backpropagation algorithm. (12) means that the BMM and its neighbors learn the class at a larger learning rate than other modules. As the result, the map of functions is organized gradually through iterative learning.

3 Applications to Nonlinear Dynamical Systems

One of the application fields of an MLP-module-mnSOM would be the nonlinear dynamical phenomena. To validate the effectiveness of our mnSOM, two simulations involved with nonlinear dynamical systems were carried out.

Before describing the results, let us consider a situation in which there is a set of unknown nonlinear systems, and the observed time series data are available to be used. These nonlinear systems are assumed to be derived from a single system, the hidden parameters of which continuously alter the system. Therefore, if there are two systems determined by similar parameters, they are expected to have similar dynamics. In such a situation, the tasks of the mnSOM are (i) to identify the unknown systems from the observed time series data, and (ii) to arrange the systems in the order of the hidden parameters in the map space. In other words, the mnSOM is expected to generate a map space as an alternative expression of the hidden parameter space.

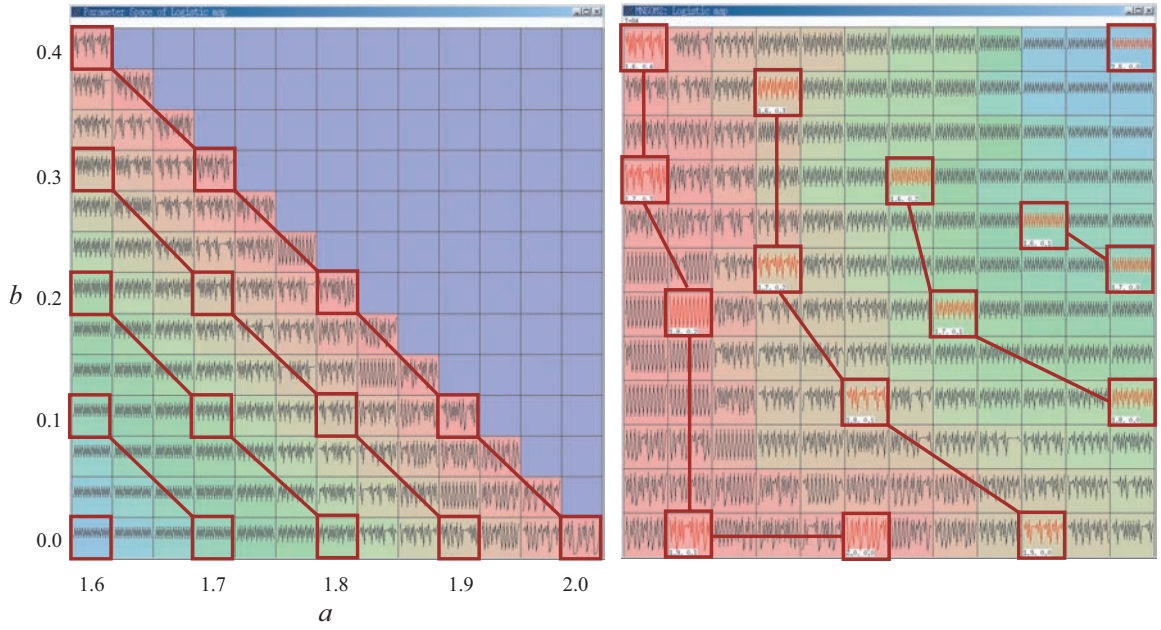

Figure 2: Simulation result of logistic maps. (a) The parameter space (a, b) and the corresponding time series. The boxes with thick frames were used for training the mnSOM (b) The map generated by the mnSOM. Each box represents the time series generated by the corresponding module

The mnSOM of Logistic Maps The first simulation was to deal with time series data generated by the logistic maps. In this simulation, the ith time series was generated by the following equation.

$$x_i(t+1) = f_i(x_i(t)) = -a_i\, x_i(t)^2 + a_i - 1 + b_i \tag{13}$$

Here, a_i and b_i were the hidden parameters that determined the dynamics of the ith time series. Since the waveforms are drastically changed by the parameters a and b, as shown in Fig. 2(a), the desired map could not be generated from direct comparisons between the time series. Therefore, the tasks of the mnSOM were to identify the dynamics $\{f_i(x)\}$ from the observed time series, and to map them while preserving the topology of the hidden parameter space (a, b). The MLP module structure was 1 input–4 hidden–1 output units. During the training phase, 15 time series produced by different parameter sets (indicated by the thick frames in Fig. 2(a)) were presented to the mnSOM.

The result is shown in Fig. 2(b). Each box corresponds to the module in which the waveform produced by the module is depicted. The modules indicated by the thick frames were the BMMs of the training data. As the figure shows, the topology of the parameter space (Fig. 2(a)) was preserved in the map space (Fig. 2(b)). Furthermore, the intermediate modules showed intermediate properties due to the interpolation of given systems.

The mnSOM of Neuron Models The second simulation was involved with the nonlinear dynamical phenomena of a biological neuron model. Here the Bonhöffer-van der Pol model (BVP model or FitzHugh-Nagumo model) was used. The BVP model is regarded theoretically as a reduction model of Hodgkin-Huxley equations, and it qualitatively reproduces various firing patterns of neurons, including the case of chaotic oscillation [8]. The dynamics of the BVP model is described by

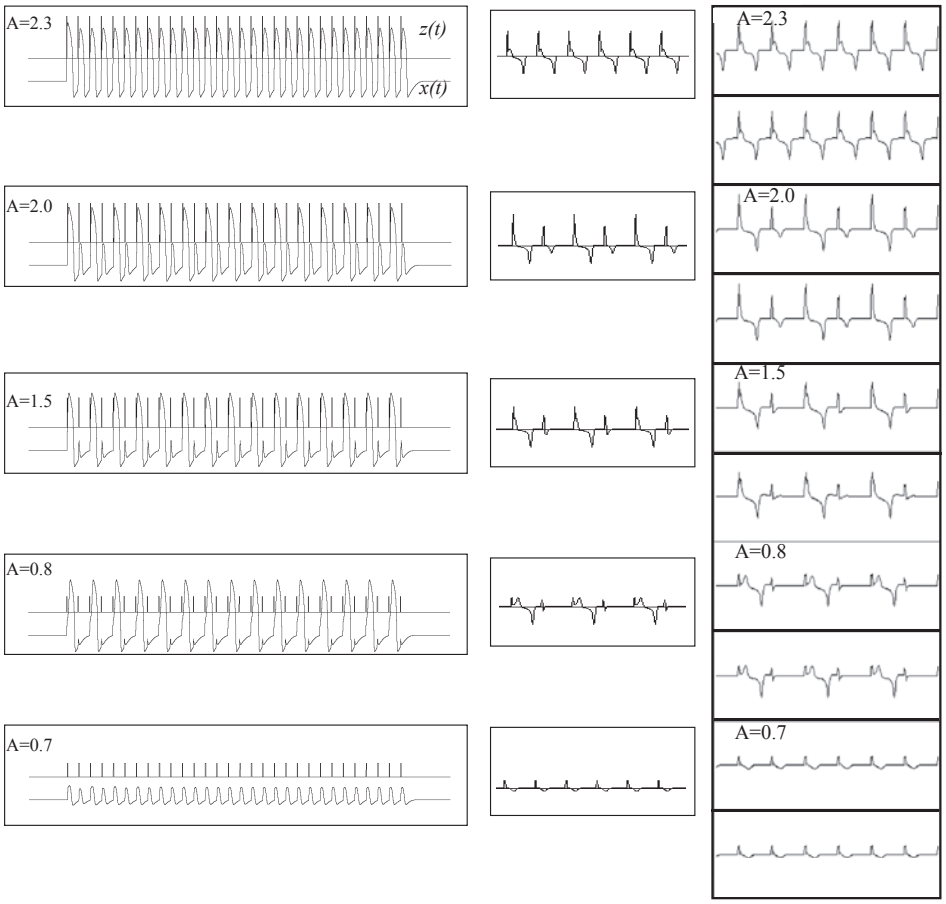

Figure 3: Simulation result of BVP neuron models. (a) Firing patterns $x(t)$ generated by a BVP model for the periodical pulse train $z(t)$ with different synaptic strengths A. (b) The time series of $\Delta x(t)$ of each class (c) The one dimensional map generated by the mnSOM and predicted $\Delta x(t)$ by the modules

the following differential equations.

$$\dot{x}(t) = c\left(x(t) - \frac{x^3(t)}{3} + y(t) - Az(t)\right) \tag{14}$$

$$\dot{y}(t) = -\frac{x(t) + by(t) - a}{c} \tag{15}$$

Here $x(t)$ and $z(t)$ represent the membrane potential and the input pulse respectively, and $y(t)$ is a hidden variable. a, b and c are constants, whereas A is the parameter representing the input intensity. In this simulation, it was assumed that $z(t)$ was the stimulus pulse to the presynaptic neuron, while A meant the synaptic strength as the hidden parameter. When A was changed, BVP model showed various firing patterns as shown in Fig. 3(a). The 5 pairs of time series $x(t)$ and $z(t)$ shown in Fig. 3(a) were regarded as the observed recordings, and they were entered to an mnSOM for training.

The tasks of the mnSOM were (i) to identify the dynamics from observed recordings, and (ii) to allocate these 5 systems in order according to the hidden parameter A. The mnSOM had a one dimensional map space with 10 modules, which were Elman type recurrent MLPs. The inputs of the

modules were $x(t)$ and $z(t)$, and the desired output was $\Delta x(t) = x(t + \Delta t) - x(t)$ as shown in Fig. 3(b). Thus, the learning task of each module was to predict the membrane potential change at Δt sec later. Each module had two input and one output units corresponding to $x(t)$, $z(t)$ and $\Delta x(t)$, while the number of the hidden (recurrent) units were set to 3.

Fig. 3(c) shows the result. Each box represents the module, and the waveform depicted in each box represents the $\Delta x(t)$ predicted by the module. The labeled modules are the BMMs of the given time series. The mnSOM succeeded in aligning the 5 waveforms in order of the hidden parameter A as well as in predicting $\Delta x(t)$.

The results of the logistic maps and the BVP models suggest that our mnSOM works as a '*Self-Organizing Bifurcation Map*' in applications involved with nonlinear dynamical systems.

4 Conclusion

In this paper, we have proposed a generalized framework of a SOM, which we named as an mnSOM. The results of applications to nonlinear dynamical systems showed that our mnSOM had good abilities; it worked as a parallel system identifier, a hidden-parameter finder and a system interpolator between given systems. The mnSOM does not directly generate a map of observed data , but extracts the essence e.g. the underlying dynamics or functions, and then generates the relevant map. These advantages of our mnSOM are showing in just one of its aspects, because the mnSOM can show another aspect by employing other types of modules. Considering this fact, our mnSOM is expected to be a powerful tool in many application fields, being a generalization of the SOM family.

Acknowledgement

This work was supported by a COE program (Center #J19) granted to the Kyushu Institute of Technology by MEXT of Japan.

References

[1] Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics **43**, 59–69, 1982

[2] Kohonen, T.: Self-Organizing Maps, 3.ed., Springer, 2001

[3] Tokunaga, K., Furukawa, T., Yasui, S.: Modular network SOM: Extension of SOM to the realm of function space. Proc. of WSOM2003, 173–178, 2003

[4] Furukawa, T., Tokunaga, K., Morishita, K., Yasui, S.: Modular network SOM (mnSOM): From vector space to function space. Proc. of IJCNN2005, 2005 (*accepted*)

[5] Kohonen, T.: Generalization of the Self-organizing map. Proc. of IJCNN93, 457–462, 1993

[6] Kohonen, T., Kaski, S., Lappalainen, H.:Self-organized formation of various invariant-feature filters in the adaptive-subspace SOM. Neural Computation **9**, 1321–1344, 1993

[7] Furukawa, T., Tokunaga, K., Kaneko, S., Kimotsuki, K., Yasui, S.: Generalized self-organizing maps (mnSOM) for dealing with dynamical systems. Proc. of NOLTA2004, 231–234, 2004

[8] Doi, S., Sato, S.: The global bifurcation structure of the BVP neuronal model driven by periodic pulse trains. Mathematical Biosciences **125**, 229–250, 1995