Modular Network SOM (mnSOM): From Vector Space to Function Space

Tetsuo Furukawa Kazuhiro Tokunaga Kenji Morishita Syozo Yasui

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— **Kohonen's Self-Organizing Map (SOM), which performs topology-preserving transformation from a high-dimensional data vector space to a low-dimensional map space, provides a powerful tool for data analysis, classification and visualization in many application fields. Despite its powerfulness, SOM can only deal with vectorized data, although many expansions have been proposed for various data-type cases. This study aims to develop a novel generalization of SOM called *modular network SOM (mnSOM)*, which enables users to deal with general data classes in a consistent manner. mnSOM has an array structure consisting of function modules that are trainable neural networks, e.g. multi-layer perceptrons (MLPs), instead of the vector units of the conventional SOM family. In the case of MLP-modules, mnSOM learns a group of systems or functions in terms of the input-output relationships, and at the same time mnSOM generates a feature map that shows distances between the learned systems. Thus, mnSOM with MLP modules is an SOM in function space rather than in vector space. From this point of view, the conventional SOM of Kohonen's can be regarded as a special case of mnSOM, the modules consisting of fixed-value bias units. In this paper, mnSOM with MLP modules is described along with some application examples.**

I. Introduction

Kohonen's Self-Organizing Map (SOM) is an unsupervised learning algorithm for generating topology-preserving transformation from a high-dimensional data vector space to a low-dimensional map space, and is a powerful tool in many areas such as data mining, analysis, classification, and visualization. Applications of SOM have spread into numerous areas such as web-based search, bioinformatics and finance, and its importance continues to increase. Although a great deal of variations and modifications have been described, a conventional SOM can only deal with vectorized data for individual application paradigms[1]. This study aims to present a novel framework that generalizes the conventional SOM family; we call this *modular network SOM (mnSOM)*. It allows us to deal with general classes of data in a consistent manner. The idea of mnSOM is simple; each vector unit of a conventional SOM, which is arrayed on a 2-dimensional lattice, is replaced by a function module of a neural network[2]. mnSOM inherits all other properties of the conventional SOM. The mnSOM strategy has several advantages. First, the application targets are widely expanded from fields involving just vectorized data to those dealing with more general classes of datasets relevant to functions, systems, time series and so on. Here, what is needed when mnSOM is applied to a specific problem is basically just a choice between module types. Thus, mnSOM provides a high degree of freedom and design flexibility to users. Basically, the essential parts of the mnSOM algorithm are formulated in a consistent manner that does not depend on the module type. From this point of view, the conventional SOM stemming from Kohonen can be regarded as a special case of an mnSOM, the function modules of which are Hebbian-learning vector units. This suggests that a vast amount of work done in the past by conventional SOM can be easily imported into mnSOM. This is another important advantage of mnSOM.

While function modules of mnSOM can be neural networks of any trainable type, in this paper we consider cases of MLP modules. Thus, each nodal module of mnSOM represents a function or a system in terms of the input-output relationship. Given input-output data observed from a group of systems, mnSOM modules are expected to learn and to identify each of the underlying systems, while at the same time mnSOM generates a feature map that allocates those systems according to their mutual distances in function space. Thus, mnSOM with MLP modules is an SOM in function space rather than one in vector space. Furthermore, an MLP-module-mnSOM gives output to SOM. Consequently, the targets of mnSOM are widened to tasks that require outputs, such as control problems.

A similar idea has been proposed as Operator Map and ASSOM by Kohonen [3], [4], but our purpose is to establish a more general and more flexible framework that takes over the operator case. One may also find a resemblance to the Local Linear Map[5]; however, our mnSOM is essentially different. The purpose of a Local Linear Map is to represent a single nonlinear function as the combination of the local linear operators, whereas the purpose of an mnSOM is to generate a feature map of an assembly of nonlinear functions, which are represented by nonlinear function modules.

In this paper, ideas, architecture, algorithms of mnSOM are initially outlined with special attention given to the MLP-module-mnSOM, then simulation results of some application examples are presented.

0-7803-9048-2/05/$20.00 ©2005 IEEE

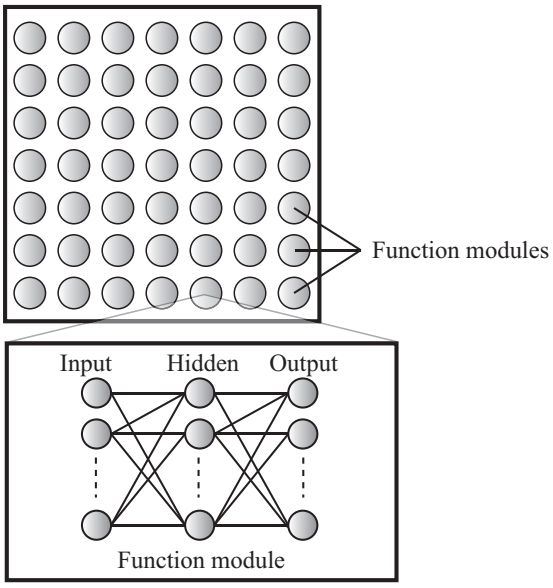


Fig. 1. The architecture of mnSOM.

II. Theoretical Framework

A. Architecture

The architecture of mnSOM is illustrated in Fig.1. Basically, the architecture is such that each nodal vector unit of the conventional SOM family is replaced by a function module such as an MLP. These modules are arrayed on a lattice that represents the coordinates of the feature map. In this paper, the function modules are assumed to be MLPs, though in reality the module architecture can be selected more flexibly. Differing from a conventional SOM, this MLP-module-mnSOM has input and output. The input is an n-dimensional vector that is fed into all input-layer units of MLP-modules, whereas the corresponding outputs are a collection of m-dimensional vectors from all modules. Thus, if the number of MLP modules is N, then mnSOM with MLP-module has one input vector (n-dimensional) and N output vectors (m-dimensional, each). Usually, the output vector of the best matching module (BMM), *i.e.* the winner module, for the input class is regarded as the output that represents the whole mnSOM network.

Suppose that there are M systems, which are unknown. But, input-output observation data are assumed to be available for them. The first task of an mnSOM is to identify these systems from such datasets. This means that the BMM of each system is expected to learn the input-output relationship of the system. The second task for the mnSOM is to generate a self-organizing map of the M systems. This means that if the i-th system and the i'-th system have similar characteristics, then two corresponding BMMs are near to each other by position on the lattice. Furthermore, the modules between the BMMs of the i-th and i'-th systems become "intermediate systems" by interpolation. These two tasks are processed in parallel.

B. Algorithm

Suppose that there are L sampled input-output vector pairs for each system. Thus, the total number of sample data is $L \times M$. Let us consider the case in which $D_i = \{(\mathbf{x}_{ij}, \mathbf{y}_{ij})\}$ $(j = 1, \ldots, L)$ is a set of observed input-output vector pairs of the i-th system. Let $f_i(\cdot)$ denote the function of the i-th system, so that $\mathbf{y}_{ij} = f_i(\mathbf{x}_{ij})$. The function $f_i(\cdot)$, which underlies the observation dataset, is not known *a priori*. Under such conditions, the aims of mnSOM are (i) to identify the unknown functions $f_1(\cdot), \ldots, f_M(\cdot)$, (ii) to interpolate between these functions, and (iii) to generate a feature map of these systems that shows the mutual distances between the systems in function space.

The algorithm for mnSOM consists of four processes: *evaluative process, competitive process, cooperative process*, and *adaptive process*.

In *the evaluative process*, the outputs of all mnSOM modules are evaluated for each input-output data vector pair. Suppose that an input data vector $\mathbf{x}_{ij}$ is picked up, then the output of the k-th modules $\tilde{\mathbf{y}}_{ij}^{(k)}$ is calculated for that input. This calculation process is repeated for $k = 1, \ldots, N$, using the same input $\mathbf{x}_{ij}$. After evaluating all outputs for all inputs, then the errors of all modules for each data class are evaluated. Now let $E_i^{(k)}$ be the error of the k-th module for the dataset from i-th system, *i.e.*,

$$E_i^{(k)} = \frac{1}{L} \sum_{j=1}^{L} \left\| \tilde{\mathbf{y}}_{ij}^{(k)} - \mathbf{y}_{ij} \right\|^2 \tag{1}$$

$$= \frac{1}{L} \sum_{j=1}^{L} \left\| g_k(\mathbf{x}_{ij}) - f_i(\mathbf{x}_{ij}) \right\|^2 . \tag{2}$$

Here, $g_k(\cdot)$ denotes the input-output function of the k-th MLP module. If L (the number of samples) is large enough, then the distance between the k-th module and the i-th system in function space is approximated by the error $E_i^{(k)}$ as follows.

$$L^2(g_k, f_i) = \int \| g_k(\mathbf{x}) - f_i(\mathbf{x}) \|^2 \, p_i(\mathbf{x}) d\mathbf{x} \simeq E_i^{(k)} \tag{3}$$

Here, $p_i(\mathbf{x})$ denotes the probability density function of the input data vectors $\{\mathbf{x}_{ij}\}$ of the i-th system. In this paper it is assumed that $\{p_i(\mathbf{x})\}$ for $i = 1, \ldots, M$ are approximately the same as $p(\mathbf{x})$, due to normalization of the data distribution for each class.

In *the competitive process*, the module which reproduces $\{\mathbf{y}_{ij}\}$ best is defined as *the best matching module (BMM)*, *i.e.* the winner module for the i-th system. Thus, let k_i^* be the module number of the BMM for the i-th system, then k_i^* is defined as

$$k_i^* = \arg\min_k E_i^{(k)}. \tag{4}$$

In *the cooperative process*, the learning rate of each module is calculated by using the neighborhood function. Usually, a BMM and its neighbor modules gain larger learning rates than other modules. Let $\psi_i^{(k)}(T)$ denote the learning rate of the k-th

module for the i-th system at time T. Then $\psi_i^{(k)}(T)$ is given by the following equations.

$$\psi_i^{(k)}(T) = \frac{h(l(k,k_i^*);T)}{\sum_{i'=1}^{M} h(l(k,k_{i'}^*);T)} \tag{5}$$

$$h(l;T) = \exp\left[-\frac{l^2}{2\sigma^2(T)}\right] \tag{6}$$

Here, $l(k,k_i^*)$ expresses the distance between the k-th module and the BMM for the i-th system in the map space, *i.e.* the distance on the lattice of mnSOM. $h(l;T)$ is a neighborhood function which shrinks with the calculation time T.

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

$$\Delta\mathbf{w}^{(k)} = -\eta\sum_{i=1}^{M}\psi_i^{(k)}\frac{\partial E_i^{(k)}}{\partial\mathbf{w}^{(k)}} = -\eta\frac{\partial E^{(k)}}{\partial\mathbf{w}^{(k)}} \tag{7}$$

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

$$g_k(\mathbf{x}) = \sum_{i=1}^{I}\psi_i^{(k)} f_i(\mathbf{x}). \tag{8}$$

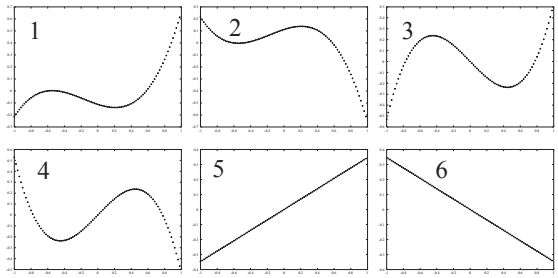
Therefore $g_k(\mathbf{x})$ is updated so as to converge to the interior division of $\{f_i(\mathbf{x})\}$ with the weights $\{\psi_i^{(k)}\}$. During training MLPs, each input vector $\mathbf{x}_{ij}$ is presented one by one as the input, and the corresponding output $\mathbf{y}_{ij}$ is presented as the desired signal.

Questions arise as to what are in common and what are different between mnSOM and SOM. In respect to algorithms, there are two main differences. First is the definition of the distance measured in the evaluative process. In the case of mnSOM, the distance between one module and one system is defined in function space, whereas the distance is defined in Euclidian vector space in a conventional SOM. It is a common point that both mnSOM and SOM measure the distance in terms of the square error between data and nodal modules/units. The other major difference lies in the learning algorithm of the adaptive process. Since the architectures are completely different, this difference between mnSOM and SOM is unavoidable. Nevertheless, the equation (7) is common for both mnSOM and SOM. Equation (7) is interpreted as the backpropagation algorithm when the modules are MLP, whereas (7) means the Hebbian-learning algorithm in the case of the conventional SOM. Consequently, in both cases the convergence points of each nodal module/unit are the weighted interior division points of a given dataset. The only difference is that a convergence point is defined in function space for mnSOM and in vector space for SOM. Therefore, the properties of the feature map generated by an mnSOM are the same as for a conventional SOM. This aspect provides good assurance when one uses mnSOM. Further comparisons between mnSOM and SOM are discussed later.

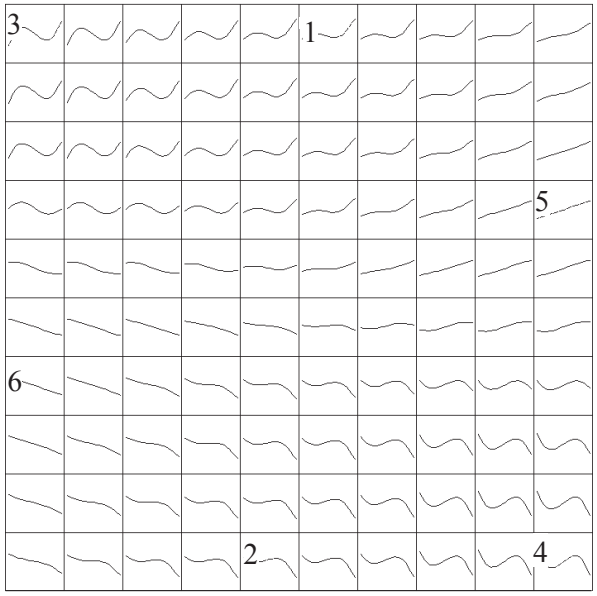
TABLE I

THE CUBIC FUNCTIONS

$f_1(x) = +\{5\sqrt{7}x^3 + 3\sqrt{5}x^2 + (2\sqrt{3} - 3\sqrt{7})x - \sqrt{5}\}/4$
$f_2(x) = -\{5\sqrt{7}x^3 + 3\sqrt{5}x^2 + (2\sqrt{3} - 3\sqrt{7})x - \sqrt{5}\}/4$
$f_3(x) = +\sqrt{7}(5x^3 - 3x)/2$
$f_4(x) = -\sqrt{7}(5x^3 - 3x)/2$
$f_5(x) = +\sqrt{3}x$
$f_6(x) = -\sqrt{3}x$



(a)



(b)

Fig. 2. (a) Training data vectors sampled from the six cubic functions of Table 1. (b) Feature map for the cubic function family generated by mnSOM. Labeled boxes represent the best matching modules (BMMs) of the six training data classes.

III. SIMULATIONS AND RESULTS

A. mnSOM of cubic functions

A simulation study is undertaken to examine the performance of mnSOM in some example cases. The first example is concerned with a family of cubic functions $y = ax^3+bx^2+cx$. The six cubic functions presented in Table 1 were used in this simulation. From each original function, 100 input-output data vector pairs $D_i = \{(x_{ij}, y_{ij})\}$ were sampled randomly (Fig.2a),

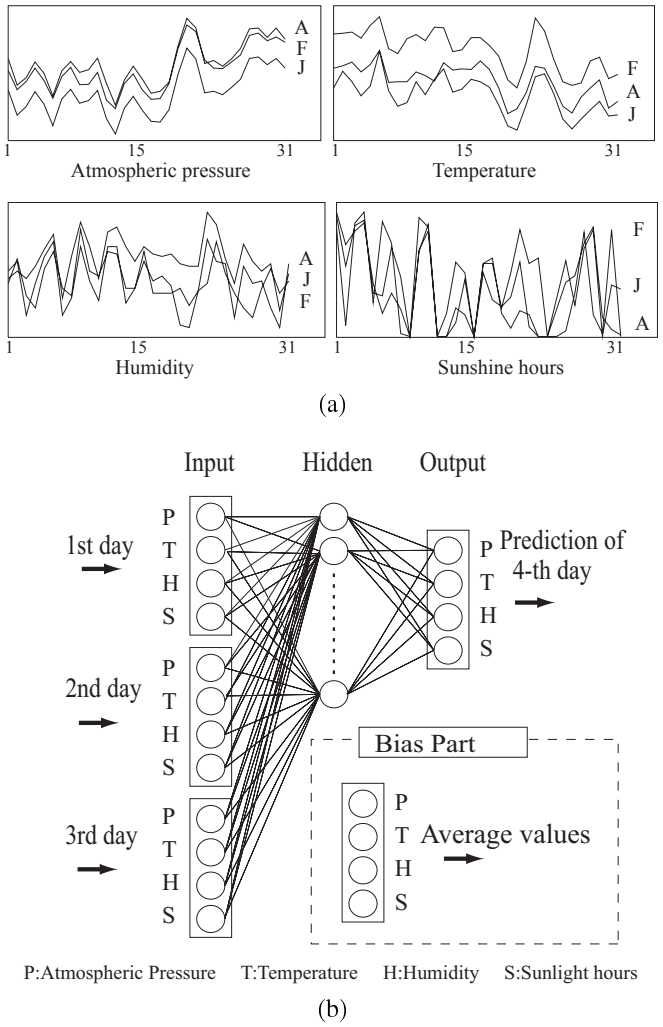


Fig. 3. (a) Examples of time series data of four weather attributes (from January 1–31) at cities A, F, J of Fig.4a. (b) The module architecture of the mnSOM for learning meteorological dynamics.

so that there were $M \times L = 6 \times 100$ pairs of data vectors. The probability density function of input $p(x)$ was uniformly distributed between $[-1, +1]$. These sampled data vectors were fed to the mnSOM in random order. The modules were the 3-layer MLPs with 5 hidden units.

Fig.2b shows a simulation result. The curve depicted in each box represents the function acquired by the corresponding module after training. The numbered boxes represent the BMMs for the training data of Fig.2a. These captured functions reproduced well the original functions. All other functions acquired by the rest of modules change continuously, so as to interpolate between the BMMs. Thus, the mnSOM succeeded in generating a feature map of the cubic functions. It is emphasized again that only randomly sampled data points were told to the mnSOM and no other information was used.

B. mnSOM of meteorological dynamics

The second material is a meteorological dataset available in a bulletin published by The Meteorological Agency of Japan.

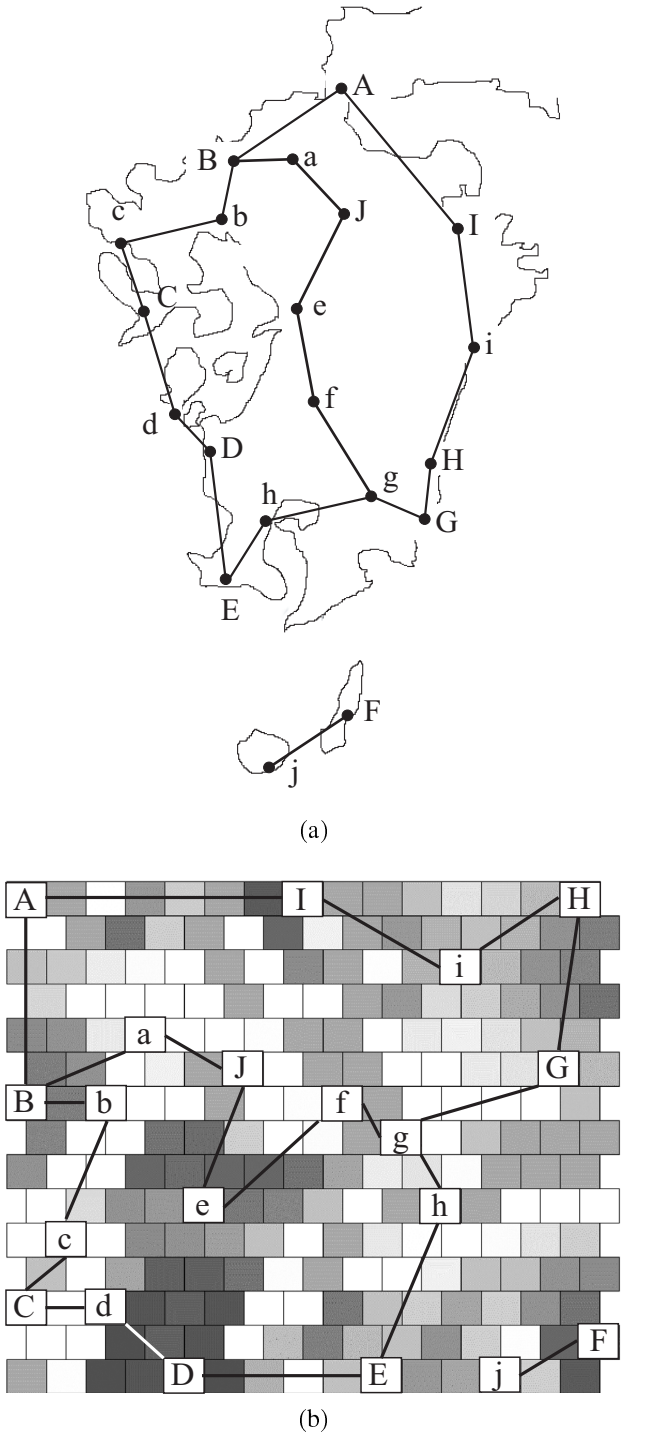


Fig. 4. (a) A map of Japan's Kyushu district. The weather data of cities A–Z were used for training, and the data of cities a–z were used for testing. (b) A weather map of the Kyushu district generated by mnSOM. A–Z and a–z correspond to the cities in Fig.4a. Gray scale represents average distance in function space between a module and its neighborhood.

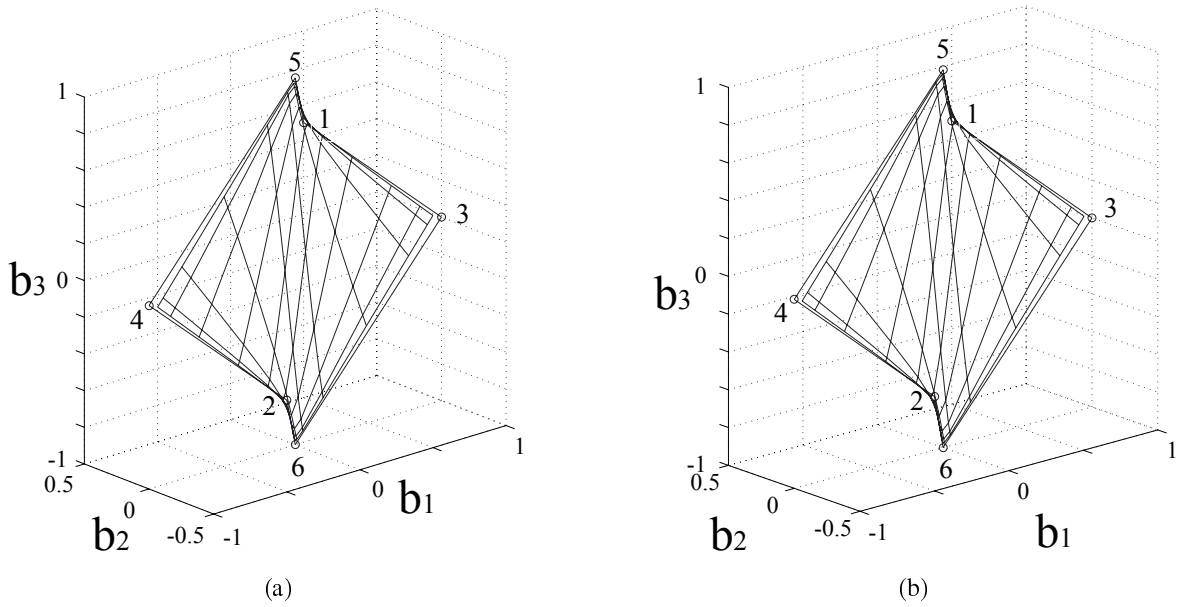


Fig. 5. The feature map of cubic functions plotted in the coefficient parameter space of Legendre expansion. (a) A map generated by mnSOM (b) A map generated by the conventional SOM. The labels 1–5 mean the BMMs of the systems (Table 1).

The dataset consists of daily records of weather attributes such as atmospheric pressure, temperature, humidity and sunlight hours, during January of the year 2000 at twenty cities in Kyushu district in western Japan. Example data are shown in Fig.3a. Such time-series data of ten out of twenty cities (A–J in Fig.4a) were given to the mnSOM for training, whereas the data of the remaining ten cities (a–j in Fig.4a) were used for testing.

The module architecture is shown in Fig.3b. The input data is a 12-dimensional vector that consists of the weather data of three consecutive days, and the desired output is the weather of the fourth day. Prior to the simulation, each time series record was normalized so that the DC component was removed. Thus, the MLP module is expected to learn weather dynamics fluctuating around the average. In addition to the MLP module, there is a four-dimensional static vector, which is expected to learn the average level (Fig.3b). For this reason, this architecture is a hybrid of a conventional SOM and an mnSOM.

During the learning phase, training data of the ten cities were given to the mnSOM. When learning had succeeded, the training was terminated. Then the test cities data were given to the mnSOM and the BMMs of those cities were identified. A simulation result is shown in Fig.4b. The labeled modules represent the BMMs determined by the training (A–J) and the test (a–j) cities of Fig4a. In the feature map, all cities for test were allocated to appropriate positions, in such a way as to interpolate between the BMMs of the training cities. Thus, mnSOM succeeded in generating "*a weather map of Kyushu district*," preserving the topology of the geographical map.

In this simulation, the mnSOM played three roles; (1) it identified the weather dynamics of the cities by training. (2) it estimated the weather dynamics of "intermediate cities," *i.e.,* tested, but not trained cities. (3) it generated "*a self-organizing weather map*" that reflected similarities in geo-meteorological dynamics.

IV. Discussion

A. Relations between mnSOM and SOM

As mentioned in the theory section, the algorithm for a mnSOM is not merely a modification of a conventional SOM, but rather it is a generalization that absorbs the conventional one. Thus, an mnSOM not only inherits many properties from a conventional SOM, but also adds several new original properties. Therefore, it is important to confirm what are common and what are different between an SOM and an mnSOM.

To this end, we re-visit the case of the cubic functions described in III.A. In the simulation, the mnSOM generated a feature map of the cubic function family from a set of input-output data. The data were sampled from the randomly given six systems, and they were presented to the mnSOM in random order. Practically, this learning style is very natural for real problems such as the case of weather data. However, in an artificial situation like a case of cubic functions, it is also possible to vectorize the function shapes. This means that the feature map of the cubic function family can be generated by the conventional SOM as well. One question then arises; is there any difference between the features map generated by the mnSOM and the one generated by the conventional SOM? To answer this question, we consider the case in which orthonormal functional expansion is employed for vectorization. Thus, let $\{\phi_j(\cdot)\}$ be a set of orthonormal functions. Then a function $f(\cdot)$ is transformed to a parameter

vector $\mathbf{a} = (a_0, \cdots, a_n)$, where

$$f(x) = a_0\phi_0(x) + a_1\phi_1(x) + \cdots + a_n\phi_n(x). \quad (9)$$

Under this condition, the distance in function space is identical to the one in coefficient vector space. Thus,

$$L^2(g_k, f_i) = (a_0 - b_0)^2 + \cdots + (a_n - b_n)^2 \quad (10)$$

$$= L^2(\mathbf{b}_k, \mathbf{a}_i). \quad (11)$$

Here, $\mathbf{a}_i$ and $\mathbf{b}_k$ denote the coefficient vectors of the given i-th system and the k-th module of mnSOM, respectively. In this situation, mnSOM and SOM should produce the same result, since the learning algorithm, *i.e.* (1)–(8) becomes identical for both types of SOM. That is, mnSOM and conventional SOM share the same properties of the feature map.

In the case of the cubic functions, $p(x)$ was set to be uniform between $[-1, +1]$. Therefore, normalized Legendre polynomials are the best for the orthonormal expansion in this case. Thus, the j-th orthonormal function $\phi_j(x)$ is described by j-th Legendre function $P_j(x)$ as

$$\phi_j(x) = \sqrt{j + 1/2}\, P_j(x). \quad (12)$$

Fig.5 shows a simulation result expressed by Legendre polynomials. The feature map generated by the mnSOM is shown in Fig.5a, whereas the map generated by the conventional SOM is shown in Fig.5b. Both graphs are plotted in the coefficient vector space $\mathbf{b} = (b_1, b_2, b_3)$ of the Legendre expansion. (In this case, the 0-th coefficient of the Legendre expansion is always zero). In the case of mnSOM (Fig.5a), all functions acquired by the modules (Fig.2b) are expanded by Legendre functions $\{g_k\} \rightarrow \{\mathbf{b}_k\}$ *after training*. In the case of the conventional SOM (Fig.5b), on the other hand, the functions corresponding to the given six systems (Fig.2a) are expanded $\{f_i\} \rightarrow \{\mathbf{a}_i\}$ *before training*, and then the coefficient vectors $\{\mathbf{a}_i\}$ were entered to the conventional SOM, the vector units of which are three-dimensional vectors $\{\mathbf{b}_i\}$. Both feature maps were always the same, like twins. This fact suggests that a great amount of previous work by conventional SOMs can be imported to mnSOMs.

However, it is important to note that since systems are usually unknown in most practical cases, such orthonormal expansion *before training* will not be possible. An mnSOM, however, does not require any such preprocessing. This is another important advantage of mnSOM.

B. mnSOM with other types of function modules

Although we have dealt with an MLP-module-mnSOM in this paper, the learning algorithm described by equation (1)–(8) is not specialized to MLP cases. Other module types are also possible, if the distances between systems and modules are defined, and if the learning algorithm for the modules is one that aims at minimizing the mean square error.

For example, if one employs the recurrent neural network (RNN), then the mnSOM acquires the ability to deal with dynamical systems. We have already reported several results of RNN-module-mnSOM, including a case of linear damped oscillatory systems and a case of nonlinear neuronal systems[6].

On the other hand, if one employs the auto-encoder MLP, then an mnSOM acquires the ability to deal with manifolds. When the auto-encoder modules have three layers, such a type of mnSOM becomes the same as an Adaptive Subspace SOM (ASSOM)[4], whereas it becomes a nonlinear-ASSOM when the modules have five layers.

While the algorithm for map formation remains untouched in this paper, many kinds of modifications for a map formation algorithm, such as hierarchical SOM, growing maps and SOM with recurrent feedback, can also be adopted without any other modification by mnSOM.

V. Conclusions

An extension of SOM using a modular network structure has been presented in this paper. Our mnSOM is a natural extension of a conventional SOM stemming from Kohonen[1], and we describe our mnSOM in a more general framework. The essential difference of an mnSOM from an SOM is not only the presence of functional modules in map space, but is also found in three different points. (1) In a conventional SOM, each data vector corresponds to a mapping object. In an mnSOM, a group of data vectors corresponds to a mapping object. Thus, the map of mnSOM represents the relationship between groups of data, instead of the relationship between individual data vectors. (2) In a conventional SOM, the position of each mapping entity, *i.e.* data vector, is known in vector space. In an mnSOM, the position of each mapping object, *i.e.* each system or function, is unknown in function space. An mnSOM can identify such entities through learning from a set of observed data while generating a feature map in parallel. (3) A conventional SOM does not have output, whereas an mnSOM does. From these aspects, mnSOM is believed to very much enlarge the application perspectives of a "SOM in general".

Acknowledgment

This work was supported by the 21th century KIT COE program 2002, "World of brain computing interwoven out of animals and robotics."

References

[1] T. Kohonen, *Self-Organizing Maps, 3.ed.*, Berlin: Springer, 2001.

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

[3] T. Kohonen, "Generalization of the Self-organizing map," *Proceedings of 1993 International Joint Conference on Neural Networks (IJCNN93)*, Nagoya, Japan, 1993, pp. 457–462.

[4] T. Kohonen, S. Kaski and H. Lappalainen, "Self-organized formation of various invariant-feature filters in the adaptive-subspace SOM," *Neural Computation*, Vol. 9, pp. 1321–1344, 1997.

[5] T. M. Martinetz, H. J. Ritter and K. J. Schulten, "Three-dimensional neural net for learning visuomotor coordination of a robot arm," *IEEE Transactions on Neural Networks*, vol. 1, No. 1, pp. 131–136, 1990.

[6] T. Furukawa, K. Tokunaga, S. Kaneko, K. Kimotsuki and S. Yasui, "Generalized self-organizing maps (mnSOM) for dealing with dynamical systems,", *Proceedings of 2004 International Symposium on Nonlinear Theory and its Applications (NOLTA2004)*, Fukuoka, Japan, 2004, pp. 231–234.