Modular Network SOM and Self-Organizing Homotopy Network as a Foundation for Brain-like Intelligence

Tetsuo Furukawa
Department of Brain Science and Engineering, Kyushu Institute of Technology
2–4 Hibikino, Wakamtatsu-ku, Kitakyushu 808–0196, Japan
e-mail: furukawa@brain.kyutech.ac.jp

Keywords: homotopy, fiber bundle, SOM^2, mnSOM

***Abstract*—** In this paper, two generalizations of the SOM are introduced. The first of these extends the SOM to deal with more generalized classes of objects besides the vector dataset. This generalization is realized by employing modular networks instead of reference vector units and is thus called a modular network SOM (mnSOM). The second generalization involves the extension of the SOM from 'map' to 'homotopy', allowing the SOM to deal with a set of data distributions rather than a set of data vectors. The resulting architecture is called SOM^n, where each reference unit represents a tensor of rank n. These generalizations are expected to provide good platforms on which to build brain-like intelligence.

1 Introduction

In order to develop intelligent agents such as autonomous robots, we must give them much higher functions than those realized thus far. For example, such agents need to have a large scale memory that has an effective learning algorithm without interference between memories, a high adaptability to changes in context and environment, and an ability to generalize their knowledge from a limited number of experiences. In addition, 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 two fundamental architectures: a modular network SOM (mnSOM) and SOM^n. These architectures will provide, as a starting point, good platforms on which to develop intelligent agents. The basic idea of the mnSOM is to combine Kohonen's self-organizing map (SOM) with a modular network. This idea was first proposed by Tokunaga *et al.* [1, 2], and many variations have since been developed. On the other hand, the SOM^n is an extension of the SOM from 'map' to 'homotopy' [3, 4, 5], and is thus also called a *Self-Organizing Homotopy* (SOH). A SOM^n represents the continuous change of a set of data distributions. Next, we describe the concepts and theory of the mnSOM and SOM^n, and then we introduce some applications thereof.

2 Concepts of the generalizations

2.1 Architecture of mnSOM

The concept of an mnSOM is simple. Every reference vector unit of Kohonen's SOM is replaced by a functional module of a neural network (Figure 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 the SOM and 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 according to their purpose. Users can choose an appropriate module type from a great number of existing trainable architectures [6]. 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 those of the mnSOM, because the backbone algorithm for the SOM is left untouched. This ensures the theoretical reliability of the mnSOM for users [7].

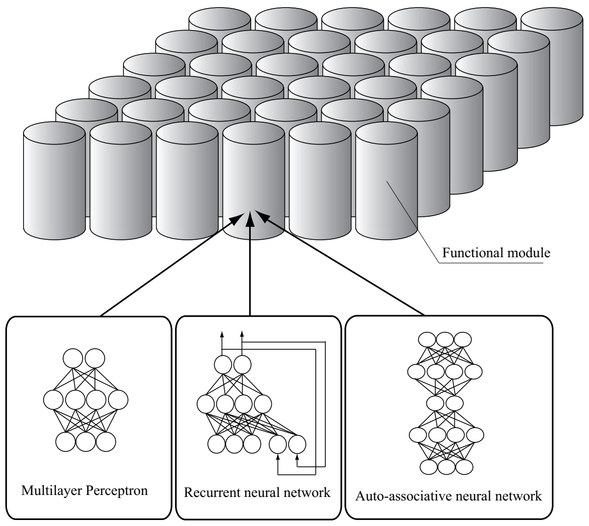


Figure 1: The concept of a modular network SOM (mnSOM).

As an example, let us consider the case in which a user wishes to implement an adaptive controller system using an mnSOM. The user's purpose is to build an mnSOM with multiple controller modules, each of which is specialized within a different context. In this case, the user only has 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 the functional modules within various contexts, while at the same time generating a feature map that indicates similarities and differences between the controllers. If the required controllers in contexts A and B are similar, then the corresponding controllers should be located near each other in the map space of the mnSOM. If contexts C and D, however, require quite different controllers, these controllers should be arranged further apart. Additionally, the intermediate modules are expected to become controllers for intermediate contexts. The backbone algorithm for a SOM ensures 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 in context. This is a novel aspect that is not found in a conventional SOM, which generates only a static map. These advantages are realized through the synergy of the SOM and modular network.

In the above case, the mnSOM represents the continuous change in a set of input-output relations, i.e., a set of functions. This mathematical concept is known as a 'homotopy', and thus, the mnSOM can be regarded as a learning machine representing a homotopy.

2.2 Architecture of SOM^n

By employing a SOM itself as the functional module in an mnSOM, we obtain a SOM-module-mnSOM. This type of mnSOM provides us with another extension of the SOM, the SOM^2 (Figure 2). A SOM^2 consists of an assembly of basic SOM modules arrayed on a lattice, which are replacements for the reference vectors of the basic SOM. Thus a SOM^2 is also regarded as a 'SOM of SOMs'. Though the name may sound a bit eccentric, the SOM^2 is a straightforward extension of the conventional SOM. Further nesting of SOMs, as in Russian dolls, is also possible, e.g., SOM^3 and SOM^4. Theoretically, the reference units of a SOM^n represent tensors of rank n. This means that the reference units of a SOM^1 represent tensors of rank 1, i.e., ordinary vectors, and thus a SOM^1 is just a conventional SOM.

Since the basic SOM represents a mapping from a high-dimensional data space to a low-dimensional feature one, the actual task of the SOM^2 is to represent the continuous change in these maps, i.e., a homotopy. Thus, the SOM^2 is an extension from a 'self-organizing map' to a 'self-organizing homotopy'. From another viewpoint, the SOM^2 has the ability of representing a set of distributions of given datasets by a fiber bundle, whereas the conventional SOM

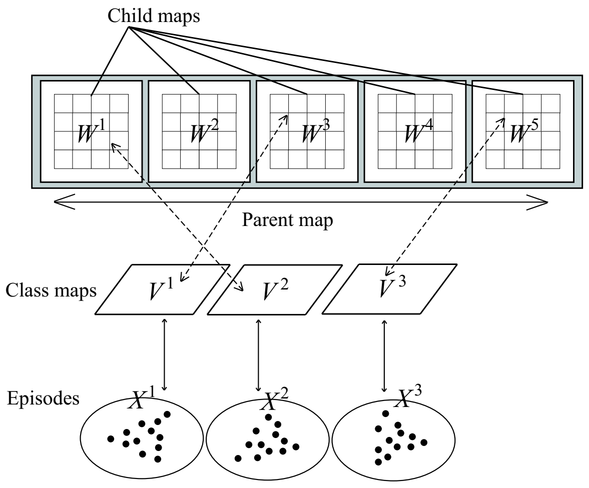


Figure 2: The concept of the SOM^2.

represents a data distribution by a manifold. Therefore the SOM^2 can be regarded as a fiber bundle learning machine rather than a manifold learning one.

When a group of datasets is given, the SOM^2 approximates their distributions by using a set of *child SOMs*, and the *parent SOM* simultaneously generates a map of the child maps. If two distributions of datasets are comparatively similar (or different), these two datasets are located closer (or further apart) in the parent map.

Such an architecture is useful when a set of data vectors observed from the same object forms a corresponding manifold in the data space. A typical example is face classification from a set of 2-dimensional photographs. In this case, a set of photographs taken of a single person from various viewpoints forms a manifold that is unique to that person. Therefore, if there are n people, one obtains n face image manifolds that can be classified by a SOM^2 [4].

2.3 Framework: Episode and Class

Two important concepts are used to describe the algorithms for the mnSOM and SOM^2: *episode* and *class*. An *episode* is a set of data vectors observed at the same time and is the minimum unit in the generalized algorithms. Thus, data vectors belonging to the same episode should be processed by the same functional module or the same child SOM. On the other hand, a *class* is a set of data vectors observed from the same object, e.g., the same system, but not necessarily at the same time.

As an example, suppose that there are static systems A, B, C, $\cdots$. Here *class A* represents 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 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 also observed from system A.

It is worth stressing that an episode does not necessarily have a *label*, which indicates the class to which it belongs. In the case of an *unlabeled episode*, the data vectors of the episode should be members of the same class, but there is no information about the class they belong to. In other words, every data vector has a *tag* that indicates the episode to which it belongs, but they do not have labels. In the case of *labeled episodes*, the class information is given to every episode, so that every class can be regarded as a sum of episodes. The important point is that the mnSOM and SOM^2 can deal with both cases. It is sometimes assumed that these extensions need class labels, but this is not true.

There is another episode category. A *complete episode* is one that has enough data vectors to cover the manifold of the class. In contrast, a *partial episode* provides only limited information about the class. For example, an episode of face images with limited view angles is a partial episode. The algorithm for the generalized SOM needs to be modified depending on the episode type.

2.4 Naive extension of SOM

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., $D_i = \{\mathbf{r}_{i,1}, \ldots, \mathbf{r}_{i,J}\}$. To simplify the situation, let us assume that these episodes are complete and labeled, and 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 one of the nonlinear functions.

There is an essential difference between the conventional SOM and our generalized SOMs. In the former 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, in the case of a generalized SOM it often happens that the entities of the objects are unknown. Therefore, the user needs to identify the objects at the same time as generating their self-organizing map. Thus the generalized SOM should solve the simultaneous estimation problem.

Let us suppose that the mnSOM has K functional modules $\{M^1, \ldots, M^K\}$, which are designed with 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 $\mathbf{w}^k$. In 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 these objects. These two tasks should be processed in parallel.

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

$$E^k_i = \frac{1}{J}\sum_{j=1}^{J}\left\|\tilde{\mathbf{r}}^k_{i,j} - \mathbf{r}_{i,j}\right\|^2. \quad (1)$$

The BMM is then determined by

$$k^*_i \triangleq \arg\min_k E^k_i, \quad (2)$$

and the learning mass $\{m^k_i\}$ and the normalized learning mass $\{\mu^k_i\}$ are calculated by

$$m^k_i = h\left(d(k, k^*_i);\ t\right) \quad (3)$$

$$\mu^k_i = \frac{m^k_i}{\sum_{i'=1}^{I} m^k_{i'}}. \quad (4)$$

Finally, every module is trained using episodes with the learning mass $\{\mu^k_i\}$. If the module algorithm is described by the gradient descendent method, then the modules are updated as follows.

$$\mathbf{w}^k = -\eta\sum_{i=1}^{I}\mu^k_i\frac{\partial E^k_i}{\partial\mathbf{w}^k} \quad (5)$$

This is the *naive* algorithm for a generalized mnSOM. This naive generalization may appear correct, and indeed it has often been used in past research. In fact the naive algorithm works appropriately in many cases, but not always. Though this naive version is worth trying, users are advised to examine the relevance as discussed below.

2.5 Natural extension of SOM

Now let us consider the true generalization of the mnSOM algorithm, which includes the naive case, in which 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 $L(O_i, \hat{O}^k)$ that reflects the difference between an entire object O_i and an entire model $\hat{O}^k$ 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 according to the user's purpose.

By using the distance measure, the definition of *mass center* can be determined by

$$\bar{O} \triangleq \arg\min_O \sum_{i=1}^{I} m_i L^2(O_i, O). \quad (6)$$

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

$$\bar{O} = \frac{m_1 O_1 + \cdots m_I O_I}{m_1 + \cdots m_I} = \sum_{i=1}^{I} \mu_i O_i. \tag{7}$$

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., $L^2(\tilde{O}(D_i), \hat{O}^k)$. Here $\tilde{O}(D_i)$ is the object estimated from the i-th episode D_i, and it is updated in parallel with $\hat{O}^k$.

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

$$\mathbf{w}^k(t+1) = \arg\min_{\mathbf{w}} \sum_{i=1}^{I} m_i^k L^2 \left(\tilde{O}(D_i, t), \hat{O}(\mathbf{w}) \right). \tag{8}$$

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

$$L^2 \left(\tilde{O}(D_i), \hat{O}^k \right) \simeq \frac{1}{J} \sum_{j=1}^{J} \left\| \tilde{\mathbf{r}}_{i,j}^k - \mathbf{r}_{i,j} \right\|^2, \tag{9}$$

the naive algorithm is obtained.

Typical cases that require this generalization are the SOM-module-mnSOM, known as SOM^2, and the mnSOM with auto-associative neural network modules.

2.6 Algorithm for SOM^2

The algorithm for the SOM^2 can be derived from the natural algorithm described above. Let $\mathbf{w}^{kl}$ denote the l-th reference vector of the k-th child SOM, then $\mathbf{W}^k = (\mathbf{w}^{k1}, \dots, \mathbf{w}^{kL})$ refers to the joint reference vectors of the k-th child SOM. Thus $\mathbf{W}^k$ represents the i-th section of the fiber bundle, while $\mathbf{F}^l = (\mathbf{w}^{1l}, \dots, \mathbf{w}^{Kl})$ expresses the l-th fiber. Besides the parent and child maps, another set of SOMs, called *class maps*, are prepared to describe the class manifolds. Let $\mathbf{v}^{nl}$ and $\mathbf{V}^n$ be the reference vectors and the joined reference vectors of the class maps, respectively. Note that $\mathbf{V}$ and $\mathbf{W}$ correspond to $\tilde{O}$ and $\hat{O}$, respectively.

Now let us consider an example with I episodes $\{\mathbf{x}^{ij}\}$, where $\mathbf{x}^{ij}$ is assumed to belong to the i-th class. In this situation, the SOM^2 algorithm for the parent map is formulated as follows.

$$k^*(\mathbf{V}^n) = \arg\min_k \|\mathbf{V}^n - \mathbf{W}^k\| \tag{10}$$

$$A_n^k = \frac{h_p\left[d(k, k^*(\mathbf{V}^n))\right]}{\sum_{n'} h_p\left[d(k, k^*(\mathbf{V}^{n'}))\right]} \tag{11}$$

$$\mathbf{W}^k = \sum_n A_n^k \mathbf{V}^n \tag{12}$$

The algorithm for child maps is described by

$$l^*(\mathbf{x}^{ij}) = \arg\min_l \|\mathbf{x}^{ij} - \mathbf{w}^{k^*(\mathbf{V}^i)l}\| \tag{13}$$

$$B_{ij}^{nl} = \begin{cases} \dfrac{h_c\left[d(l, l^*(\mathbf{x}^{ij}))\right]}{\sum_{j'} h_c\left[d(l, l^*(\mathbf{x}^{ij'}))\right]} & \text{if } n = i \\ 0 & \text{if } n \neq i \end{cases} \tag{14}$$

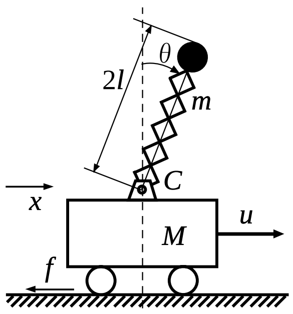
$$\mathbf{v}^{nl} = \sum_i \sum_j B_{ij}^{nl} \mathbf{x}^{ij}. \tag{15}$$

By combining Eqs. (12) and (14), the following updated SOM^2 algorithm is obtained.

$$\mathbf{w}^{kl} = \sum_n \sum_i \sum_j A_n^k B_{ij}^{nl} \mathbf{x}^{ij} \tag{16}$$

Note that the class maps $\{\mathbf{V}^n\}$ have disappeared in Eq. (16), because $\{\mathbf{V}^n\}$ is defined for convenience of explanation and is therefore not necessary in the practical implementation.

The above algorithm can be interpreted as follows. At the parent level, the conventional SOM algorithm is executed by regarding $\mathbf{V}^n$ and $\mathbf{W}^k$ as the data and the reference vectors, respectively. At the child level, the reference vectors of the class maps $\mathbf{v}^{nl}$ are updated by executing the con-



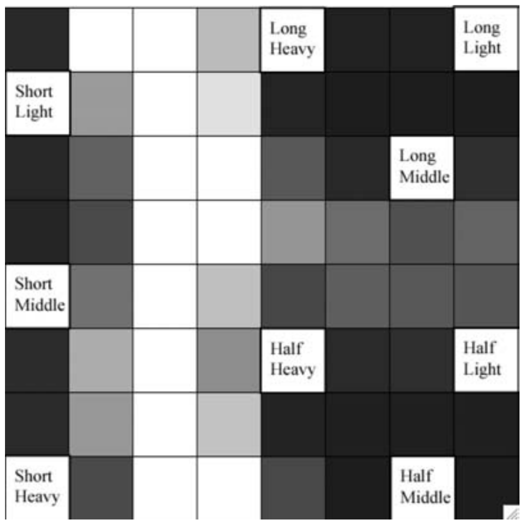


Figure 3: A map of controllers of inverted pendulums generated by an mnSOM.

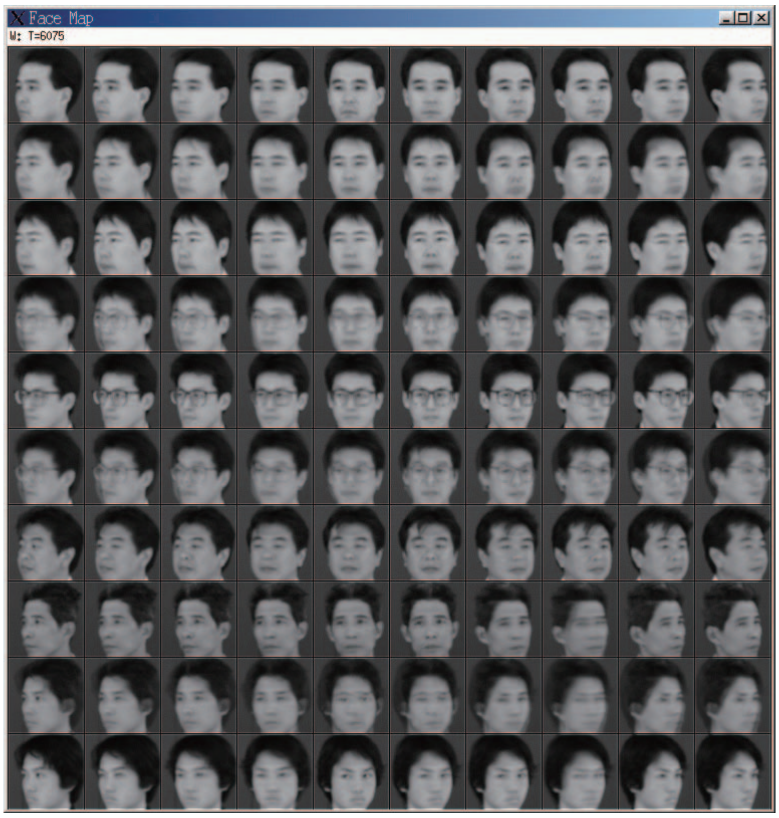


Figure 4: A map of facial images generated by a SOM^2. Each row represents a child map, whereas each column represents a fiber of the SOM^2.

ventional SOM algorithm, in which the reference vector of the winner, i.e., $\mathbf{w}^{k^*(\mathbf{V}^n)l}$, is regarded as the initial state of $\mathbf{v}^{nl}$. Consequently the child maps are organized by affecting one another via the parent map, thus representing the fibers naturally. Note that the k-th reference unit $\mathbf{W}^k$ is a tensor of rank 2, and the entire SOM^2 is represented by a tensor of rank 3.

3 Applications

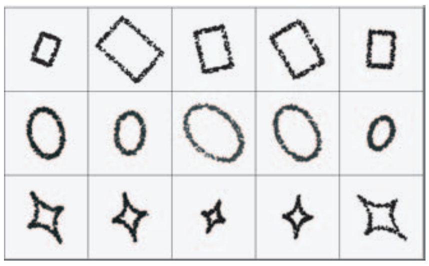
Adaptive control is one of the typical application fields for the mnSOM. In this case, the mnSOM consists of an assembly of neural network controllers. By training the mnSOM using various parameters of the target object, the mnSOM generates a map of controllers. Figure 3 shows the map of controllers for an inverted pendulum, the mass and length of which vary. Since this map also represents the parameter space of the pendulums, the mnSOM can select the appropriate controller module before taking over control. For example, if a given pendulum looks long and heavy, the mnSOM can commence control using the controller module that appears most appropriate. Once after taking over control, the best matching controller is selected as the winner. Further details of this method are given in [8].

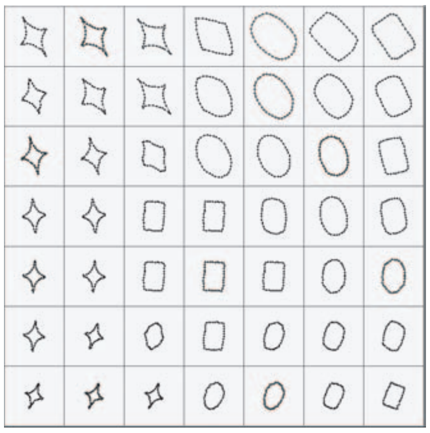
The main application field for the SOM^2 is manifold classification. The representation and classification of face image sets are examples of this. Figure 4 shows the map of faces generated by a SOM^2. In this case, every episode consists of a set of facial images taken from various angles. As a result, the continuous change in camera angle is represented by the child maps, whereas each fiber represents a set of face images taken from the same angle.

Shape classification is another typical application for the SOM^2. Figure 5 shows an example, where every contour is regarded as an episode, each data vector of which represents the $x - y$ coordinate of a dot. Figure 5 (b) shows the map of the contours generated by a SOM^2. The result shows that the intermediate child maps represent intermediate shapes, and the entire SOM^2 represents the continuous change in the contours. Figure 5 (c) shows a more practical case, in which a set of neural gas (NG) networks are employed instead of child SOMs.

4 Conclusion

In this paper, two generalizations of the SOM are introduced. The two methods can easily be combined, enabling larger networks to be built. If a user wishes to represent a set of continuously changing input-output functions, the RBF$\times$SOM^2 would be a good solution. In addition, these

(a)


(b)


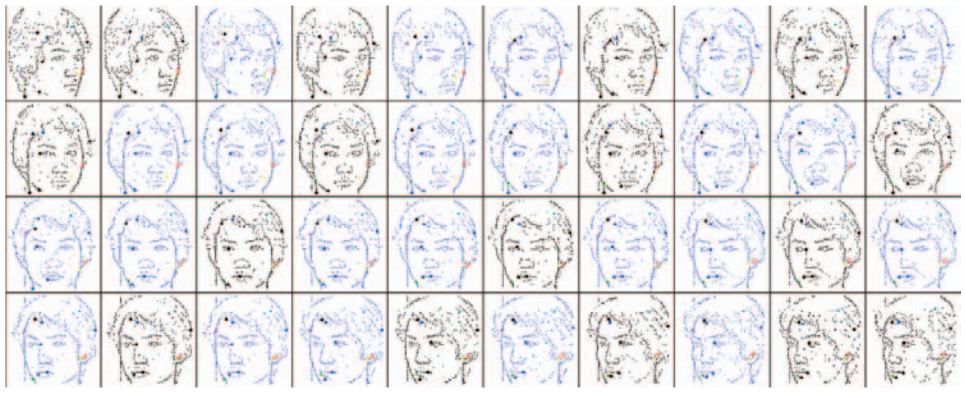
(c)


Figure 5: (a) The given set of contours of a map of shapes (b) The map of shapes generated by a SOM^2 (c) A map of faces represented by a set of dots (The original map is one dimensional; due to space constraints it is however displayed over 4 lines.)

generalizations can be applied to many extensions of the SOM, such as the growing type, hierarchical type and so on. The combination with the growing type of SOMs provides a more flexible representation of higher-order functions. We are now applying this method to building intelligence for autonomous agents.

Acknowledgements

This work was partially supported by the 21st Century Center of Excellence Program at Kyushu Institute of Technology (Center #J19) by MEXT of Japan. The work was also partially supported by a Grant-in-Aid for Scientific Research (C) granted by MEXT of Japan. The facial images in this paper have been used with permission of Softopia Japan, Research and Development Division, HOIP Laboratory.

References

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

[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, "SOM of SOMs: Self-organizing map which maps a group of self-organizing maps," *Lecture Notes in Computer Science*, Vol.3696, pp.391–396, 2005.

[4] T. Furukawa, "SOM^2 as SOM of SOMs," *Proc. of WSOM2005*, pp.545–552, 2005.

[5] T. Furukawa, "An extension SOM from 'map' to 'homotopy'," *Lecture Notes in Computer Science*, Vol.4232, pp.958–967, 2006.

[6] T. Furukawa, "Generalization of the self-organizing map: From artificial neural networks to artificial cortexes,", *Lecture Notes in Computer Science*, Vol.4232, pp.943–949, 2006.

[7] T. Furukawa, K. Tokunaga, K. Morishita and S. Yasui, "Modular network SOM (mnSOM): From vector space to function space," *Proc. of IJCNN2005*, pp.1581–1586, 2005.

[8] T. Minatohara and T. Furukawa, "Self-organizing adaptive controllers: Application to the inverted pendulum," *Proc. of WSOM2005*, pp.537–544, 2005.