Some Learning Properties of Modular Network SOMs

Manabu TAKEDA *, Kazushi IKEDA **, and Tetsuo FURUKAWA ***

Abstract : The Modular Network Self-Organizing Map (mnSOM) is a generalization of the SOM, where each node represents a parametric function such as a multi-layer perceptron or another SOM. Given datasets are, in general, fewer than nodes, some nodes never win in competition and have to update their parameters from the winners in the neighborhood. This is a process that can be regarded as interpolation. This study derives the interpolation curve between winners in simple cases and discusses the distribution of winners based on the neighborhood function.

Key Words : neural networks, Self-Organizing Map, function learning, Modular-Network SOM.

1. Introduction

A Self-Organizing Map (SOM) is a learning algorithm for projecting high-dimensional given data into a lower (usually one or two) dimensional space in an unsupervised manner [1], [2]. A generalization of the SOM, called the Modular Network SOM (mnSOM), exists, in which each node represents a parametric function such as a multi-layer perceptron or another SOM [3]–[6].

Each node in a conventional SOM is given input vectors to be adapted/quantified through self-organization. In the case of an mnSOM, however, the nodes are given sets of learning data, to which they then have to adapt, i.e., estimate the parameter vector which produced each of the datasets. At the same time, the nodes cooperate and compete in the same way as in a conventional SOM. This extension gives mnSOMs a high degree of design flexibility without changing the backbone algorithm of the SOM. One of the applications of mnSOMs is a map of the weather dynamics of the world’s cities [5], while another is a classifier of 3D objects from 2D images [6].

Another difference from conventional SOMs is that in mnSOMs given datasets are fewer than the nodes. In many cases, conventional SOMs work as vector quantizers where a lot of input vectors are located in a continuous space and fewer nodes are representative vectors. However, input “vectors” for an mnSOM are much fewer than the nodes it has, since each of the input vectors is represented as categorized data for training. This means that some (actually many) of the nodes never win in competition. Such a node has to update its parameter vector by relying on neighboring winners. The purpose of this study is to elucidate how an mnSOM learns, or what function the mnSOM realizes, from given data, which was never an issue in the conventional SOM framework.

Since the parameter vector of a node is determined by the winners surrounding it, an mnSOM interpolates the function space based on winner nodes from a mathematical viewpoint. In fact, many of the applications of mnSOMs are based on interpolation. To make mnSOMs more reliable, however, we need to clarify the properties of the interpolation theoretically. Hence, as the first step of the study, we derive the explicit function given by the interpolation in a simple case.

This paper is organized as follows. Sections 2 and 3 show the formulation and some properties of mnSOMs, respectively. In Section 4, we explicitly derive the interpolation function in simple cases in which the nodes have a one-dimensional structure. Section 5 extends the results from the simplest case to a two-dimensional case which is more common in applications. The results are confirmed by computer simulations in Section 6. The discussions so far assume that the two winners are fixed, although they actually move during the learning process. In Section 7, we give a qualitative analysis of the distribution of the winners to relax the assumption. Finally, our conclusions are presented in Section 8.

2. Modular Network SOM

We formulate the mnSOM as given below, where \( x_m \) and \( y_m \) denote, respectively, the \( m \)th input vector and its corresponding label in the \( m \)th dataset. Note that the relationship differs from dataset to dataset.

1. Assume \( M \) given datasets

\[
D_m = \{(x_m, y_m) | n = 1, \ldots, N_m\}
\]

\( m = 1, \ldots, M \),

and \( K \) nodes, each of which represents a parametric function \( g_k(x, w_k) \). The winner \( k_m^*(t) \) for the \( m \)th dataset \( D_m \) at the \( t \)th step is defined as the node that has the minimum loss for \( D_m \) among the \( K \) nodes, that is,

\[
k_m^*(t) = \arg \min_{1 \leq k \leq K} L_m^k(t),
\]

\[
L_m^k(t) = \sum_{(x, y) \in D_m} L(g_k(x, w_k(t), y)
\]

where \( L(g, y) \) represents a differentiable loss function with respect to \( g \), such as the squared Euclidean norm \( \|y - g\|^2 \).

2. Every node \( k \) updates its parameter vector, denoted by \( w_k(t) \) to \( w_k(t + 1) \) in a supervised manner so that its loss

* Graduate School of Informatics, Kyoto University, Sakyo, Kyoto 606-8501, Japan
** Nara Institute of Science and Technology, Ikoma, Nara 630-0192, Japan
*** Kyushu Institute of Technology, Kitakyushu 808-0196, Japan
E-mail: kazushi@is.naist.jp
(Received March 11, 2009)
(Revised July 22, 2009)
$L^k_m(t)$ for $D_m$ decreases. Its learning rate, however, depends on the winner $k^*_m$ and the neighborhood function $\Phi(D(k,k^*_m))$, which is a decreasing function of a distance $D(\cdot,\cdot)$ in a low-dimensional structure such as a lattice or a torus, introduced a priori.

3. Repeat steps 1 and 2.

An example of an mnSOM with a lattice-structure and employing the stochastic descent method for updating its parameter vectors, is expressed as

$$w_k(t+1) = w_k(t) - \eta_m \frac{\partial L^k_m}{\partial w_k}$$  \hspace{1cm} (4)$$

$$\Phi^k_m(t) = \Phi(D(k,k^*_m(t))) = \exp\left[-\frac{D(k,k^*_m(t))}{2\sigma^2_t}\right],$$  \hspace{1cm} (5)

where $D(k,k')$ is the Manhattan distance in the lattice, and $\sigma_t > 0$ and $\eta > 0$ are decreasing functions of time $t$ for convergence of the algorithm.

3. Properties of mnSOMs

Suppose $\eta$ is a constant $\eta$ for simplicity. Then, each node learns all the datasets $D_m$ with different weights $\Phi^k_m$ by repeating the steps given above in the limit $t \to \infty$. In other words, node $k$ minimizes the weighted loss,

$$L^k = \sum_m \Phi^k_m w^k_m,$$  \hspace{1cm} (6)

where $(t)$ is omitted since $k^*_m$ does not depend on $t$ as $t \to \infty$. We denote the parameter vector of node $k$ in the limit by $w_k$.

When two nodes are close to one another, that is, they have a small $D(k,k') \approx 0$, the learning rates hold $\Phi^k_m \approx \Phi^k_m$ for any dataset $D_m$, which leads to $L^k \approx L^k$, and hence $w^k \approx w_k$. This means that the set of nodes in the mnSOM forms a low-dimensional manifold in the parameter vector space, and hence the functions realized by such parameters change smoothly as seen in Fig. 1 [7].

It should be noted that in many applications of mnSOMs, the number $M$ of datasets is much less than the number $K$ of nodes. This means that the winners are sparsely located in the structure and that the parameters of the remaining nodes are determined by the winners, only if the winners are fixed in advance. The nodes that cannot win for any dataset are called losers. This property can be regarded in the sense that an mnSOM interpolates the function space, in other words, estimates the missing values for losers. In fact, mnSOMs have been applied to such fields as weather forecasting, where the weather for a particular node is forecasted from the data of its neighbors [5].

4. Interpolation Curves

From a mathematical viewpoint, an mnSOM determines the parameters of the losers by interpolating those of the winners, where the interpolation is based on both the neighborhood function of the mnSOM and the learning machine for each node. In this section, we derive the explicit interpolation formula of an mnSOM in the simple case, in which each node is a linear function,

$$g_k(x, w_k) = w^T_k x,$$  \hspace{1cm} (7)

and winners are fixed in a one-dimensional lattice structure (called the line below). Note that the linearity makes it possible to express the optimal parameter for (6) explicitly and that any loser in the line is located between two winners.

In the following, we denote the winners for datasets $D_1$ and $D_2$ by A and B, and their parameter vectors by $w_A$ and $w_B$, respectively, and consider the parameter vector $w_X$ of loser X to be between A and B. By introducing the distance function into the line as the segment length of two nodes, we express the length of AX and XB as $a$ and $b$, respectively (Fig. 2). Our purpose is to express $w_X$ as a function of $w_A$, $w_B$, $a$, and $b$.

Since the equilibrium condition for the learning (4) is expressed as

$$0 = \sum_m \Phi^k_m \frac{\partial L^k_m}{\partial w_k}$$  \hspace{1cm} (8)

for any $k$, $w_A$ and $w_B$ must satisfy

$$0 = \Phi(0) \frac{\partial L^A_A}{\partial w_A} + \Phi(a + b) \frac{\partial L^A_B}{\partial w_A},$$  \hspace{1cm} (9)$$

$$0 = \Phi(a + b) \frac{\partial L^B_B}{\partial w_B} + \Phi(0) \frac{\partial L^B_A}{\partial w_B},$$  \hspace{1cm} (10)

when the mnSOM converges, where

$$\frac{\partial L^k_m}{\partial w_k} = \frac{\partial}{\partial w} \sum_{(x,y) \in D_m} ||y - w^T_k x||^2$$

$$= -\hat{p}_m + \hat{R}_m w_k,$$  \hspace{1cm} (11)$$

$$\hat{p}_m = \frac{1}{L_m} \sum_{(x,y) \in D_m} xy,$$  \hspace{1cm} (12)$$

$$\hat{R}_m = \frac{1}{L_m} \sum_{(x,y) \in D_m} xx^T,$$  \hspace{1cm} (13)

which leads to

$$0 = \Phi(0)(-\hat{p}_A + \hat{R}_A w_A) + \Phi(a + b)(-\hat{p}_B + \hat{R}_B w_B),$$  \hspace{1cm} (14)$$

$$0 = \Phi(a + b)(-\hat{p}_A + \hat{R}_A w_B) + \Phi(0)(-\hat{p}_B + \hat{R}_B w_B).$$  \hspace{1cm} (15)
When there are sufficient examples in each dataset, the experimental auto-correlation matrix $R_m$ can be approximated to the true one, $R$, based on the essential assumption in the mSOM, that the pdf $p(x)$ of any dataset is the same [5]. Therefore, the cross-correlation vectors $\hat{p}_m$ are expressed as

$$\hat{p}_A = -\Phi(0)Rw_A + \Phi(a + b)Rw_B,$$

$$\hat{p}_B = \Phi(a + b)Rw_A - \Phi(0)Rw_B. \quad (16)$$

As for $w_X$, (8) leads to

$$0 = \Phi(a)\frac{\partial L_X^p}{\partial w_A} + \Phi(b)\frac{\partial L_X^p}{\partial w_B}, \quad (17)$$

or

$$w_X = \frac{\Phi(0)\Phi(a) - \Phi(0)\Phi(a + b)}{(\Phi(0) - \Phi(a + b))(\Phi(a) + \Phi(b))}w_A - \frac{\Phi(0)\Phi(b) - \Phi(0)\Phi(a + b)}{(\Phi(0) - \Phi(a + b))(\Phi(a) + \Phi(b))}w_B. \quad (19)$$

This means that $w_X$ is essentially a linear interpolation since the sum of the coefficients of $w_A$ and $w_B$ is always unity.

From (19), the winners are expressed as

$$w_A = R^{-1}\Phi(0)\hat{p}_A + \Phi(a + b)\hat{p}_B,$$

$$w_B = R^{-1}\Phi(a + b)\hat{p}_A + \Phi(0)\hat{p}_B, \quad (20)$$

respectively, which can be substituted for (19), to give the explicit form of $w_X$.

Let us show two examples of interpolation curves by explicitly giving the neighborhood functions. In the first example, the neighborhood function is a Gaussian with constant variance,

$$\Phi(d) = \exp\left(-\frac{d^2}{2\sigma^2}\right). \quad (22)$$

Then, (19) is given as

$$w_X = \frac{\exp[-a^2/2\sigma^2] - \exp[-b^2/2\sigma^2] - (a + b)^2}{(1 - \exp[-(a + b)^2/2\sigma^2])(\exp[-a^2/2\sigma^2] + \exp[-b^2/2\sigma^2])}w_A \quad \text{or} \quad \text{in a matrix form as}$$

$$Pf = RWD, \quad (28)$$

where

$$P = \{\hat{p}_U, \hat{p}_U^2, \ldots, \hat{p}_U^M\},$$

$$W = [w_U, w_U^2, \ldots, w_U^M],$$

$$F = [\Phi(d_{UmUk})]_{m,k}, \quad (31)$$

$$D = \text{diag}\left\{\sum_m \Phi(d_{UmUk})\right\}, \quad (32)$$

In the same way, the parameter vector $w_X$ of a loser $X$ satisfies

$$0 = \sum_m \Phi(d_{UmX})\frac{\partial L_X^p}{\partial w_X}, \quad (33)$$

where

$$L_X^p(t) = \sum_{(x,y) \in D_u} L(g_t(x, w_X), y), \quad (34)$$

or in a matrix form as

$$Pf_X = d_X Rw_X, \quad (35)$$

where

$$f_X = [\Phi(d_{U1X})\Phi(d_{U2X})\ldots\Phi(d_{UMX})]^T,$$

$$d_X = \sum_m \Phi(d_{UmX}). \quad (37)$$

Combining (28) and (35), $w_X$ is expressed as

$$w_X = WD^{-1}f_X/d_X, \quad (38)$$

which means that $w_X$ is a linear combination of $w_{Um}, m = 1, \ldots, M.$

### 5. Extension to General Cases

The discussion in the previous section can easily be extended to the general case, that is, with more dimensions in a low-dimensional structure and more winners affecting a loser.

We denote the winner for dataset $D_k$ by $U_k$, which has a parameter vector $w_{Uk}$, and consider the parameter vector $w_X$ of a loser $X$. From (8), $w_{Uk}$ must satisfy

$$0 = \sum_m \Phi(d_{UmUk})\frac{\partial L_{UmUk}^p}{\partial w_{UmUk}}, \quad (26)$$

for any $k$, where $d_{UmUk}$ is the distance between $U_m$ and $U_k$ in the low-dimensional structure, that is, $D(U_m, U_k)$.

By introducing the same assumption that the experimental auto-correlation matrix can be approximated to the true one, $R$, as in the previous section, (26) is rewritten as

$$\sum_m \Phi(d_{UmUk})\hat{p}_{Um} = \sum_m \Phi(d_{UmUk})Rw_{Uk}, \quad (27)$$

or in a matrix form as

$$PF = RWD, \quad (28)$$

where

$$P = \{\hat{p}_U, \hat{p}_U^2, \ldots, \hat{p}_U^M\},$$

$$W = [w_U, w_U^2, \ldots, w_U^M],$$

$$F = [\Phi(d_{UmUk})]_{m,k}, \quad (31)$$

$$D = \text{diag}\left\{\sum_m \Phi(d_{UmUk})\right\}, \quad (32)$$

In the same way, the parameter vector $w_X$ of a loser $X$ satisfies

$$0 = \sum_m \Phi(d_{UmX})\frac{\partial L_{UmX}^p}{\partial w_{UmX}}, \quad (33)$$

where

$$L_{UmX}^p(t) = \sum_{(x,y) \in D_u} L(g_t(x, w_{UmX}), y), \quad (34)$$

or in a matrix form as

$$Pf_X = d_X Rw_X, \quad (35)$$

where

$$f_X = [\Phi(d_{U1X})\Phi(d_{U2X})\ldots\Phi(d_{UMX})]^T,$$

$$d_X = \sum_m \Phi(d_{UmX}). \quad (37)$$

Combining (28) and (35), $w_X$ is expressed as

$$w_X = WD^{-1}f_X/d_X, \quad (38)$$

which means that $w_X$ is a linear combination of $w_{Um}, m = 1, \ldots, M.$
Table 1 Parameters for the computer simulations.

| # of nodes N | 11 |
| # of datasets M | 2 |
| # of items in datasets L | 100 |
| Dimension of x and w | 2 |
| Distribution of x | Uniform in \([-5,5] \times [-5,5]\) |
| Neighborhood function | Gaussian with variance 3 |
| Step size \(\eta\) | 0.1 |
| # of learning steps | 5000 |
| Noise | free |
| \(w^*_1, w^*_2\) | \((5,-5)^T, (-5,5)^T\) |

---

6. Computer Simulations

To verify the results in Section 4, various computer simulations were carried out. Each node \(k\) is a linear function, \(y = w_k^T x\), and each example in the \(m\)th dataset \(D_m = \{ (x_{mn}, y_{mn}) | n = 1, \ldots, N_m \}\) is constructed as \(y_{mn} = w_m^T x_{mn}\) where \(w_m^*\) is the true parameter vector for \(D_m\). Other parameters are listed in Table 1.

After self-organizing and learning according to (4), the leftmost node is the winner for \(D_1\) and the rightmost for \(D_2\) in the line of eleven nodes (Fig. 3).

Since \(x\) obeys the uniform distribution in \([-5,5] \times [-5,5]\), we obtain the auto-correlation matrix \(R\) and the cross-correlation vectors explicitly as

\[
R = \begin{bmatrix}
25/3 & 0 \\
0 & 25/3
\end{bmatrix}, \\
p_1 = \begin{bmatrix} 25/3 \\ 5 \end{bmatrix}, \\
p_2 = \begin{bmatrix} 25/3 \\ -5 \end{bmatrix}.
\]

By substituting these for (19), we get the theoretical solid curves for the first and second elements of the parameter vector in the upper and lower plots of Fig. 4, respectively. The circles in these plots denote the experimental results.

7. Distribution of Winners

In this section, we discuss the distribution of winner nodes since these are not fixed in general. Actually, any winner can be influenced by other datasets if there is a node in the neighborhood that wins for another dataset. Since the effect depends on the neighborhood function, the result of this analysis suggests how to determine the width of the neighborhood function.

Suppose that the width of the neighborhood function is small, e.g., \(\sigma^2 \ll 1\) in (22). Then, each winner matches the corresponding dataset best. Moreover, the other datasets cannot affect the winner due to the small width, and thus, the winner never loses for that particular dataset (Fig. 5). This situation also applies to the node next to the winner, where the difference of the neighborhood function is regarded as the difference of the learning coefficient, which does not affect the equilibrium condition in (8). Hence, the next node also becomes a winner, and so on, one after the other, until the set of winners interferes with another winner-set (Fig. 6).

Around the boundary of two winner-sets, the nodes between the winner-sets will be interpolated as discussed in the previous section (Fig. 7).

Suppose that winners \(w_A\) and \(w_B\) for the datasets \(D_1\) and \(D_2\) are closer than the width of the neighborhood function (Fig. 8). Then, the current winner, \(w_A\) for example, cannot be the winner for \(D_1\) in the future. Since \(w_A\) is more affected than the node to the left of \(w_A\) when \(D_2\) is given and \(w_B\) wins, \(w_A\) matches \(D_1\) less, while the left node will ultimately match it better. This means that the boundaries of winner-sets have the same width, the radius of the neighborhood function, with finite support of the neighborhood function. In the case of infinite support, the winners will align equidistantly from one another, though a more strict analysis of this is necessary in the future.

8. Conclusions

In this paper, we analyzed the mnSOM and derived an explicit form for the interpolation between two winner nodes, where each node is a linear function. The results showed that the interpolation is essentially linear, that is, the parameter vector of a node between two winners is expressed as a weighted combination of the vectors of the winners, where the weights are determined by the neighborhood function. In addition, we discussed the distribution of winner nodes.
The results implied that the boundaries of winner-sets have the same width, the radius of the neighborhood function, in the case of finite support of the neighborhood function, and otherwise, the winners align equidistantly from one another. More strict analysis is a future work of the authors.

References


Manabu Takeda
He received his B.E. degree from Kyoto University, Japan, in 2008. He is currently a master-course student of the Graduate School of Informatics, Kyoto University.

Kazushi Ikeda (Member)
He received his B.E., M.E., and Ph.D. degrees from the University of Tokyo, Japan, in 1989, 1991, and 1994, respectively. In 1994, he joined Kanazawa University and moved to Kyoto University in 1998. He has been a Professor of the Graduate School of Information Science, Nara Institute of Science and Technology since 2008. His research interests include machine learning, neural networks and adaptive filters. He is a member of IEICE, JNNS, ISCIE, and IEEE.

Tetsuo Furukawa
He received his B.E. and M.E. degrees from Osaka University, Japan, in 1987 and 1989, respectively. In 1989, he joined the faculty of the Kyushu Institute of Technology, where he is currently a Professor of the Department of Brain Science and Engineering. He is a member of IEICE, JNNS, and IEEE.