International Journal of Neural Systems, Vol. 1, No. 3 (1990) 269–283

ADDING LEARNED EXPECTATION INTO THE LEARNING PROCEDURE OF SELF-ORGANIZING MAPS

Lei Xu
Department of Information Technology, Lappeenranta University of Technology, Box 20, 53851 Lappeenranta, Finland
Permanent address: Dept. of Mathematics, Peking University, P.R. China

Received 27 November 1989
Revised 2 April 1990

The self-organizing topological map is generalized by adding a learned expectation to its learning procedure, in order to improve its stability in nonstationary environments with unexpected inputs and abnormal noise. Such a learned expectation is realized through a compatibility test which checks whether an input is compatible with the earlier learned patterns on a map unit before the unit starts to adapt to the input. The generalized map consists of multi-maps with a pipeline architecture, equipped with a parallel search strategy which allows for the implementation of the learned expectation to be fulfilled without extra computing costs. Computer experiments on several examples are given to show the characteristics of the generalized map in comparison with the original map.

1. Introduction

As one of the most notable recent developments in competitive learning, Kohonen's self-organizing topological map has been widely investigated in its applications, theoretical analysis, extensions and variations.[1–3] Two variations were recently given.[4,5] One of them[4] made a modification on computing speed and data representation by adding a "conscience" to the competition. In the other variation,[5] the ordering of data with different variances in each dimension is improved by using accordingly weighted distances, and better and faster approximations of prominently structured density functions are obtained by introducing the neighbourhoods specified by a minimal spanning tree.

As opposed to those approaches, it is suggested in this paper that the maps be generalized by including a learned (top-down like) expectation into its learning procedure. The importance of top-down expectation in human perception has been emphasized by Grossberg.[6] Particularly, in Grossberg's ART model[11,12] (another most notable development in competitive learning), top-down expectation is used for self-stabilizing adaptive pattern recognition in real-time nonstationary input environments. In this paper, a learned expectation, with a similar function of top-down expectation in ART, is introduced into topological maps by checking whether the present input is compatible with the previously learned information of its best-matching unit, and if not, by finding another candidate for the best matching unit. This results in a generalized self-organizing map which consists of multi-maps, with a pipeline architecture equipped by a parallel search strategy. With little extra computing costs, such a generalized map not only can retain the properties of the original map, but can also stabilize its learning in a nonstationary environment with unexpected inputs and abnormal noise. Furthermore, its discriminative ability could be adjusted via an external parameter to tune to different needs.

In Sec. 2, a brief comparative review is made on the original Competitive Learning (CL), Adaptive Resonance Theory (ART) and Self-Organizing Map (SOM). Hereafter, an analysis is given to show that SOM learning has inherited some constraints of CL, and a new motivation is proposed for SOM to break these constraints by using a learned expectation, with a function similar to top-down expectation in ART. Section 3 starts with a discussion on where and how to add a learned expectation into the learning procedure of SOM, and then the aspects for practical implementation are investigated. Next, a generalized SOM with an architecture of multi-maps and parallel search is presented. Furthermore, the characteristics of the generalized learning proce-

dure are analyzed. Finally in Sec. 4, several computer experiments on both the original SOM and the generalized SOM are given to show the characteristics of the generalized SOM.

2. CL, ART and SOM: From Review To New Motivation

2.1. *Competitive learning (CL)*

CL model was first proposed by Malsburg[9] and Grossberg,[10] and has subsequently been further analyzed and developed by a number of authors.

The basic principle of CL is briefly described by the model given in Fig. 1(a). Layer L_1 consists of n input units $x_1, x_2, \ldots, x_n$, each x_i takes value 0 or 1. Layer L_2 consists of M units $z_1, z_2, \ldots, z_M$, each of which inhibits every other unit such that L_2 is winner-takes-all, i.e. the unit receiving the largest input obtains its maximum value 1 while all other units are pushed to the minimum value 0. From each x_i to each z_j, there is a weight w_{ij} as shown in Fig. 1(b). When an input pattern X (for convenience denoted by $X = [x_1, x_2, \ldots, x_n]^t$) appears on L_1, each unit z_j receives an input of sum

$$\eta_j = \sum_{i=1}^{n} w_{ij}x_i = m_j^t X, \qquad m_j^t = [w_{1j}, \ldots, w_{nj}]^t, \tag{1a}$$

and units $z_1, z_2, \ldots, z_M$ compete with each other to get activated, which results in

$$z_j = \begin{cases} 1 & \text{if } \eta_j = \max\{\eta_k, k = 1, \ldots, M\}. \\ 0 & \text{otherwise.} \end{cases} \tag{1b}$$

Then the weights adapt to the input X in the following way

$$\Delta m_j = \begin{cases} \alpha(\theta - m_j) & \text{if } z_j = 1; \\ 0 & \text{if } z_j = 0 \end{cases} \tag{1c}$$

where α is a parameter representing the learning rate, and θ is the normalized input $\theta = [x_1, \ldots, x_n]^t / \sum_{i=1}^{n} x_i$.

2.2. *Adaptive resonance theory (ART)*

Grossberg[10] has pointed out that although CL and variants are useful in many situations, the learning becomes unstable in response to a variety of input environments, especially in a nonstationary environment. To solve this kind of problem, he proposed the ART model[11,12] which could self-stabilize its learning in response to arbitrary input environments.

One key idea of ART is to introduce the mechanism of learned top-down expectation into CL. Roughly speaking, the idea is as follows:

In Fig. 2, after one of the units in layer L_2 (say unit c) won the competition, i.e. $z_c = 1$, m_c does not immediately start to learn X unless the unit c is uncommitted (i.e. it never won in all the previous competitions), otherwise, the unit c sends out a prototype pattern $P^c = [p_1^c, \ldots, p_n^c]^t$ to match the present input X under a given criterion (Fig. 2c). The prototype pattern is learned from all the previous inputs which let this unit c win. If a good match between P^c and X is obtained, the weight m_c adapts to X (Fig. 2b). If a mismatch occurs, the unit c is forced to be inhibited at $z_c = 0$ and another winner-takes-all process occurs within all the other units in layer L_2 (Fig. 2d). Then the new winner repeats the same procedure as the early winners did, until a good match is obtained or an uncommitted unit is found or all units of layer L_2 have been used up.

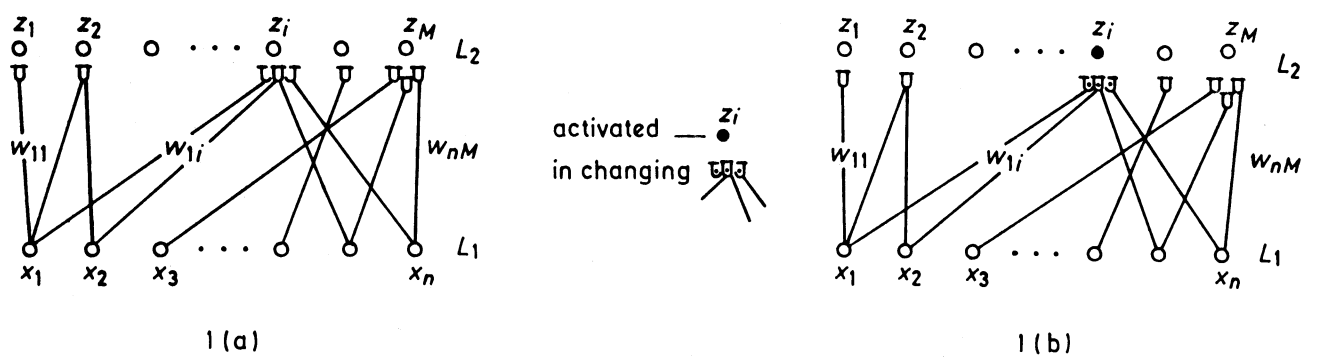


Fig. 1. Competitive learning. (a) A two-layer network. (b) Changing the weights connected to the winner.

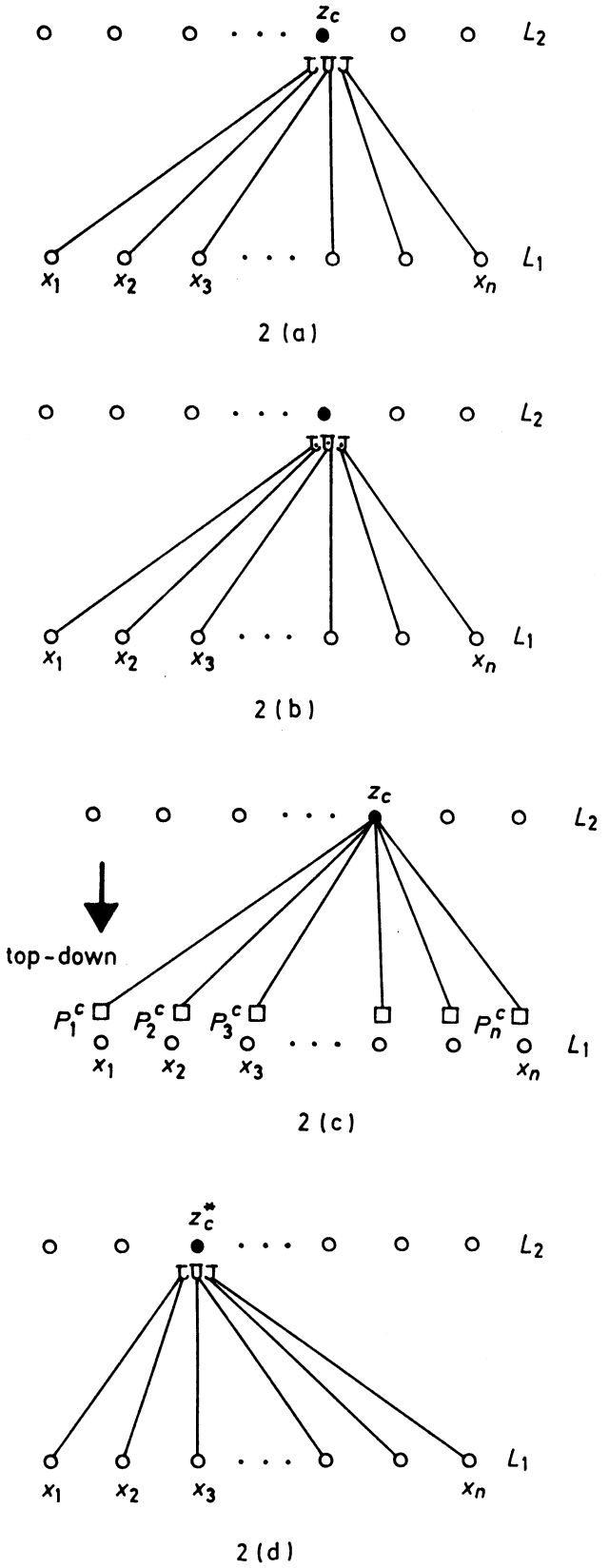


Fig. 2. Top-down expectation in ART. (a) Unit c wins the competition. (b) Changing the weights connected to unit c. (c) z_c sends out a prototype P^c to match X. (d) Searching a new winner if P^c mismatches X.

The whole ART process is completed via the interaction and coordination of several subsystems, the details of which can be found in the original papers.[6,10,11,12] What we are interested in is how the learned top-down expectation mechanism can be introduced into CL for self-stabilizing its learning in a nonstationary environment.[11,12]

2.3. *Self-organizing map (SOM)*

Based on the anatomical and physiological evidence that in mammalian brains there exists a Mexican-hat-like lateral interaction (Fig. 3a) between neuron units, Kohonen developed CL into his SOM[1] in a way different from ART, i.e. introducing a topological neighbourhood mechanism into CL.

We still take the 1-D case as in Fig. 1 to conceptually explain the key idea behind SOM and its connection with the original CL. In layer L_2 of SOM, each unit laterally interacts with other units in a form of Mexican hat (Fig. 3a), not as in CL where each unit of layer L_2 inhibits all the other units. As a result, when an input X appears on L_1 and each unit z_j receives an input η_j (see Eq. (1a)), units $z_1, z_2, \ldots, z_M$ compete and also cooperate with each other, the result is that the units within a neighbourhood N_c are activated (Fig. 3b) which results in the weights adapting to X in the following way:

$$\Delta m_j = \begin{cases} \alpha(\theta - m_j) & \text{if } j \in N_c; \\ 0 & \text{if } j \notin N_c, \end{cases} \tag{2}$$

where $N_c = \{c - k_t, \ldots, c - 1, c, c + 1, \ldots, c + k_t\}$, k_t is a constant integer, and unit c, which is usually called the best matching unit, satisfies $\eta_c = \max\{\eta_k, k = 1, \ldots, M\}$.

While for CL the learning occurs only on the weights connected to the winning unit (i.e. the best matching unit), the learning in the SOM occurs on the weights connected to all the units within a topological neighbourhood centred at the best matching unit.

We should point out that in the usual implementation of SOM there are several additional features[1]:

(1) As shown in Fig. 3c, layer L_2 is usually a two-dimensional array, thus the topological neighbourhood N_c is also a 2-D area instead of a 1-D interval.

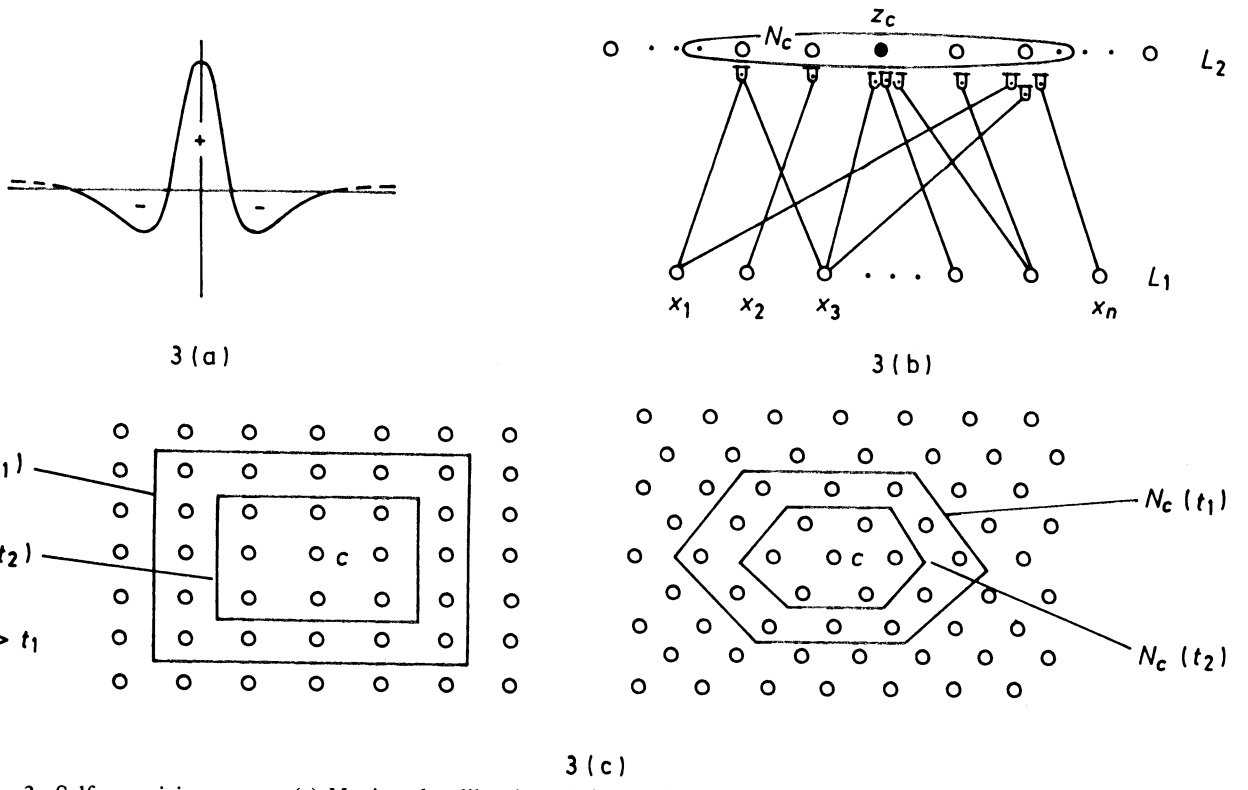


Fig. 3. Self-organizing map. (a) Mexican-hat-like lateral interaction. (b) Changing the weights connected to all the units in N_c. (c) Topological neighbourhood which changes with time.

(2) The variables x_i of input pattern X are continuous variables in the real domain R, instead of only taking value 0, 1.

(3) To ensure that learning is stabilized, the learning rate α and the neighbourhood N_c (see Fig. 3c) are both functions of time t: $\alpha(t)$ and $N_c(t)$, which monotonically decrease towards zero with increasing t.

(4) Equations (1a) (1b) and (2) are replaced by

$$\|X - m_c\| = \min\{\|X - m_j\|,\ j = 1, \ldots, M\} \quad (3a)$$

$$N_c(t) = \{\text{a set of units topologically centred at unit } c\} \quad (3b)$$

$$\Delta m_j = \begin{cases} \alpha(t)(X - m_j) & \text{if } j \in N_c; \\ 0 & \text{if } j \notin N_c. \end{cases} \quad (3c)$$

The following summarizes the general procedure of the SOM[1]:

step 1: For each input X, find the best-matching unit c, with its weight vector m_c being the best one to match X among all the m_j, according to a matching criterion such as Eq. (3a) (or other variants).

step 2: Increase the match between X and the weight vectors m_j of units within a topological neighbourhood N_c of unit c according to Eq. (3c).

2.4. *Analysis on SOM learning and a new motivation*

As indicated in Sec. 2.3, SOM came from incorporating a topological neighbourhood mechanism into CL: the winner-takes-all competing is replaced by the Mexican-hat-like competing and cooperating. This important development makes SOM more useful than the original CL both theoretically (e.g. for interpreting a variety of maps in the brain[1]) and for practical applications.[2,3] But, SOM still suffers from the unstable learning problem of the original CL.

When an input X induces a neighbourhood N_c by Eqs. (3a) and (3b), all the weights m_j of the units within N_c will learn X. As a result, when X is compatible with the knowledge previously stored in these weights, the learning will refine these weights to cover the new input without erasing the previous useful knowledge. However, when X is considerably different from what these weights learned earlier, the adaption to X will at least partly wash away the

prior knowledge. This adaptability is useful at the beginning, for rearranging the configuration of the map and for jumping out of the local optimal configurations. But if every unit keeps such adaptability, the learning will become unstable and repeat a cycle of adapting to the present input and forgetting the prior knowledge.

To handle the problem externally, SOM lets the learning rate $\alpha(t)$ slowly decrease with time from a given initial value to zero, such that the adaptability of the map is gradually switched off to force the learning to stabilize. In a stationary environment, every pattern will be repeatedly presented according to a probability distribution. So, what is forgotten one time still has a chance to be relearned next time. If $\alpha(t)$ decreases slowly and the learning procedure lasts long enough, the learning will be gradually stabilized in response to the probability distribution of inputs.[1] In a nonstationary environment, unexpected changes in the probabilities of inputs, or in the deterministic sequence of inputs, will unrecoverably erase the prior knowledge of the map, and make it difficult to stabilize unless the adaptability is shut off externally (e.g. let $\alpha(t)$ be zero or stop the learning procedure). Even when the map has already organized and converged in a stationary environment, some unexpected noisy inputs will also disturb or destroy the map, as long as the adaptability (i.e. the learning rate) is not frozen.

As is the case for CL, SOM needs an external adjustment on the adaptability of the learning. The adjustment is achieved through a teacher, externally designing how to decrease the learning rate $\alpha(t)$ and watching when to stop the learning. In a stationary environment, a teacher can approximately handle this task through slow learning. However, in a nonstationary environment, a teacher cannot catch up with those unexpected changes unless he is omniscient. SOM is thus unable to work in the nonstationary environment and not robust against noise.

It is interesting to recall how ART solves the stability problem. Instead of externally adjusting the adaptability of the learning, ART uses a learned top-down expectation to internally self-check whether the present bottom-up input is appropriate for the weights of the winner in L_2 to learn it. If not, the interaction between the bottom-up input and the top-down expectation will automatically shut off (in help of other subsystems in ART) the adaptability of the winner, and produces a new winner. Our motivation has been to find a way to introduce both the top-down expectation mechanism and the topological neighbourhood mechanism into CL, to combine the advantages of SOM and ART.

There are two possible choices. One is to introduce the topological neighbourhood to the ART model, which we do not intend to discuss here. The other one is to add to SOM a learned expectation mechanism similar to the learned top-down mechanism used in ART, so that learning could be self-stabilized through checking whether the present best matching unit should learn the input or a new best matching unit should be found. This approach to overcoming the limitations of SOM is discussed in the following sections.

3. A Generalized SOM With Learned Expectation

3.1. *Adding learned expectation into the learning procedure of SOM*

As stated in Sec. 2.2, in ART, the top-down expectation starts its action just after the competition of L_2 units produces a winner unit. Its function is to check if the present input is compatible with the previously learned code in the weights of the winner, and thus to decide either to learn the input by this winner or to find a new winner. That "an input is compatible with a learned code on a unit" means that the previously learned code will not be considerably (according to some criterion) erased after the unit learns the new input.

Similarly, in SOM, after an input X finds its best matching unit c in the 2D-lattice (which corresponds to layer L_2 in Fig. 1), an expectation could be introduced to check whether X is compatible with the earlier learned pattern on unit c and thus to decide either to learn X by unit c or to find a new best matching unit. Since it relates the present X to the previous knowledge of a specific unit, the expectation should not be nonspecific and primed externally, but should be individual and learned adaptively from the input patterns previously matched to the unit. So we call it the learned expectation of a unit, or simply the learned expectation. We omit the term "top-down" as used in ART since conventionally only the 2D-lattice is regarded as a layer in the SOM case and it is still not necessary to specify terminologies as "top-down" and "bottom-up".

Within the general procedure of SOM given in Sec. 2.3, a learned expectation should be inserted between step 1 and step 2. Specifically, the implementation of a learned expectation consists of three parts:

(1) Call out a learned pattern from the best matching unit, the pattern should be a prototype which represents the previous knowledge of the unit.

(2) Check whether the present input is compatible with the learned prototype. The test depends on two factors. One is the similarity between the input and the prototype under a given criterion. The more similar the input to the prototype, the less the prior knowledge will be erased by adapting to the new input. The other factor is the degree to which the best matching unit has approached its stabilized state. When the unit is far from its stabilized state, its stored knowledge is not so reliable; the test should be weak and should permit certain dissimilarity so that the unit is still sensitive to adapt to new input patterns. As the unit becomes closer to its stabilized state and its stored knowledge becomes more reliable, the test should be tight and should only permit high similarity, so that the unit can learn the new subtle information contained in the input pattern, but at the same time strongly resist the input from erasing the previously learned code.

(3) Find a new best matching unit if the input is not compatible with the learned pattern on the present best matching unit.

The above three parts are repeated until a best matching unit is uncommitted or is compatible with the present input, or until all the units of the 2D-lattice have been tested (in this case, further discussed in the next subsection, the input is rejected).

Now, we discuss the practical design aspects of parts (1) and (2), but leave part (3) to Sec. 3.2.

Consider part (1): how to obtain the prototype pattern stored on a unit. In ART, this pattern is stored in its top-down LTM traces[12] and learned from those inputs which let the unit win the competition in layer L_2. It is interesting that the learning equation of the top-down LTM traces in ART[12] is quite similar to the SOM learning equation (3c). The weight m_j of unit j in SOM learned by Eq. (3c) is just a kind of exponentially weighted mean of those inputs which let the unit fall in the neighbourhood of a best matching unit. Thus, for simplicity, we can directly regard the m_j as the learned prototype pattern of unit j.

Consider part (2): how to realize the compatibility test. There are two problems to be solved. One is how to know when a unit is close to its stabilized state, which requires clarification of the meaning of "the stabilized state of a unit". Strictly speaking, when the number of input patterns is considerably larger than the number of the units in the 2D-lattice, given a constant learning rate $\alpha \neq 0$, the weight m_j of each unit j will never absolutely converge to a constant vector. However, if inputs are from a stationary probability distribution, it could be argued (although without a strict proof) that the location of every m_j will reach a probabilistic equilibrium: each m_j is most probably to wander randomly in the weight space within a limited range, and the larger the value of α, the larger the range. When α slowly dereases, such a range gradually shrinks. After α finally reaches zero, each m_j is frozen at a fixed point. In this paper, the meaning of "the stabilized state" varies according to circumstances; it means "reached a probabilistic equilibrium" when α is fixed at a value larger than zero, or it means "stabilized at a fixed point" when α decreases towards and finally reaches zero. For the purpose of practical implementation, we use a scalar measure for each unit j, denoted σ_j, to describe the degree of stability. At each step, these scalars are updated according to

$$\sigma_j = \begin{cases} (1-\beta)\sigma_j + \beta\|\Delta m_j\| & \text{if } j \in N_c; \\ \sigma_j & \text{if } j \notin N_c. \end{cases} \tag{4}$$

There $0.5 \leq \beta \leq 1$ is a parameter, and Δm_j is the change in the value of m_j during the last updating. In effect, σ_j is a kind of exponentially weighted mean of all the previous differences $\|\Delta m_j\|$. In Eq. (3c), as $\alpha(t)$ tends to zero, $\|\Delta m_j\|$ tends to zero as the unit tends to its stabilized state, and σ_j tends to zero. When $\alpha(t)$ does not tend to zero but to a positive value, σ_j also becomes small in average, as the unit is close to its stabilized state.

The other problem is what criterion to use for the compatibility test. In accordance with Eq. (3a) and for simplicity of computation, we use $\|\cdot\|$ to measure the similarity between the present input X and the learned expectation pattern m_c. By taking the measure σ_c into consideration, we use the following inequality as a criterion for the compatibility test:

$$\alpha\|x - m_c\| < 4\frac{1-\nu}{\nu}\sigma_c. \tag{5}$$

X is regarded as being compatible with m_c if the inequality holds. The parameter $0 < \nu < 1$ has some similarity with the attentional vigilance parameter in ART[12]: for ν close to one the test is very sensitive, only a small difference $\|x - m_c\|$ can pass the compatibility test. The opposite effect is achieved if ν is close to zero. Since it follows from Eq. (3a) that $\Delta m_c = \alpha[x - m_c]$, checking whether (5) holds is in fact checking whether the scalar increment of the weight vector $\|\Delta m_c\|$ by learning the present input X is not larger than $4(1 - \nu)/\nu$ times the weighted mean of all the previous increments. The closer the unit to its stabilized state, the smaller σ_c is, thus the increment of the weight vector Δm_c by learning this X must be more strictly constrained to be quite small to avoid erroneous erasures.

3.2. *The generalized SOM with multi-maps and parallel search*

Consider the case when an input is rejected by all the units of the 2D-lattice, so that some input patterns remain unlearned. A single lattice has a limit capacity. The capacity could be extended by using a number of lattices to construct a multi-map architecture in which each lattice is a building block. All the blocks are connected to form a pipeline (see Fig. 4(a)) and each block has its own clock so that $N_c(t)$ and $\alpha(t)$ decrease according to their own time. A new pattern is presented to the first block; if it is rejected, it goes to the second block, . . ., and so forth until it is accommodated (learned) by some block, or it goes through and out of the pipeline. More blocks could be appended to increase the capacity, if needed.

In principle, both within a block and between blocks, the process of an input flowing (i.e. searching a new best matching unit after one compatibility test fails) is sequential. One may thus think that the computation time will increase considerably due to the use of the learned expectation. However, this is true only if the computation of Eq. (3a) and Eq. (5) is repeated each time that a test on the present best matching unit fails. In fact, Eq. (3a) and Eq. (5) need not be re-computed: the computation for searching the best-matching unit is approximately the same in both the original and the generalized SOM. The reason is as follows:

Consider first one block. When an input comes, the computation of searching the best matching unit consists of three parts:

(1) calculate its distance to all the units on the 2D-lattice: $d_j = \|X - m_j\|$, $j = 1, \ldots, M$, (see Eq. (3a)).

(2) make M comparisons (see Eq. (5)):

$$d_i < 4 \frac{(1 - \nu)}{\nu\alpha} \sigma_i$$

(where $4(1 - \nu)/\nu\alpha$ is constant to i), and then put each unit j with its d_j satisfying the inequality into a set U_C called the compatible subset.

(3) choose the unit c with $d_c = \min\{d_j, j \in U_C\}$ as the best matching unit. If U_C is empty, among the subset U_u of those presently uncommitted units in the lattice, choose the unit c with $d_c = \min\{d_j, j \in U_u\}$. If U_u is also empty, send this input to the subsequent block. It is not difficult to see that the result of such a search process is equivalent to a sequential search and for the best-matching unit by repeatedly calculating Eq. (3a) and Eq. (5).

The implementation of the original SOM also requires the computation of point (1) above. The M comparisons needed for deciding the smallest d_c require comparable computation to point (2) above. So the only extra computation the generalized SOM needs is point (3) above, and it is quite small since U_C contains usually only a few units. In hardware implementation and (see Fig. (4b)), each unit could calculate d_i and do its correspondent comparison in parallel with all the other units. The result is that a few units (i.e. those in U_C) are pre-activated, and then the winner-takes-all occurs only among these units. The computing speed of the generalized SOM in parallel implementation may even be faster (at least not slower) than that of the original SOM.

The search process between blocks can also be implemented in a parallel way. In the pipeline of Fig. 4(b) when an input is rejected by one block, the block can at the same time send it to the next block and get a new input from the preceding block: a systolic-like parallel computation could be used here.

The learning procedure in one block of the generalized SOM is summarized as follows:

put all the units in set U_u, *set all* σ_j *being a large constant.*
begin *get new* X, *set* $U_C = \emptyset$;
for $j = 1, 2, \ldots, M$;
calculate $d_j = \|X - m_j\|$;
if $\alpha d_j < 4(1 - \nu)\sigma_j/\nu$, **then** *put unit* j *into set* U_C;

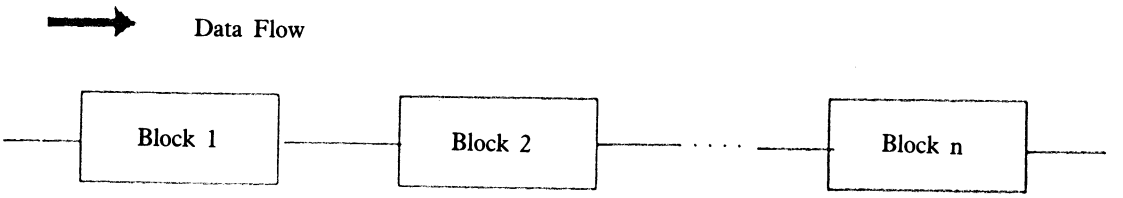


Fig. 4(a)

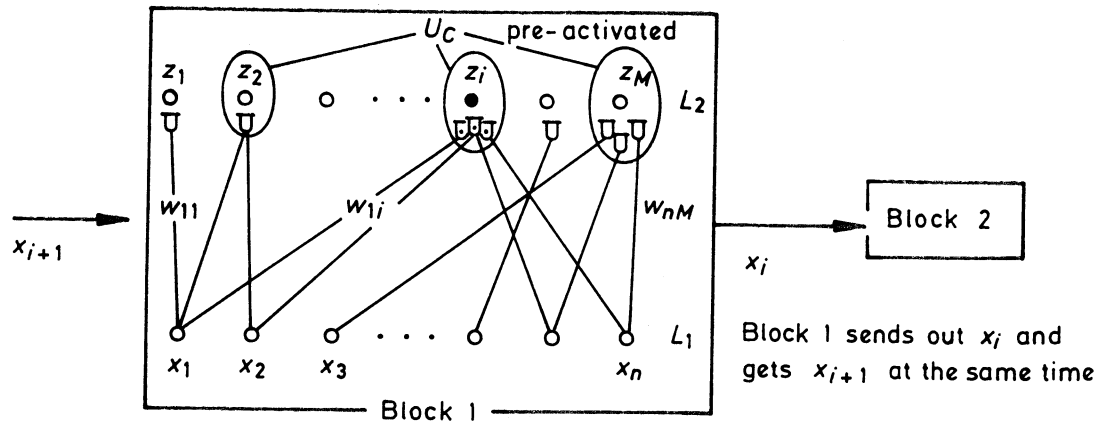


Fig. 4(b)

Fig. 4. Multi-maps and parallel search. (a) A pipeline architecture. (b) (i) units in U_C and pre-activated in parallel, (ii) blocks work in systolic way.

```
    endfor;
  if U_C = ∅ and U_u = ∅
    then send out X;
    else begin
      if U_C ≠ ∅  then find c with
                       d_c = min{d_j, j ∈ U_C};
                  else find c with
                       d_c = min{d_j, j ∈ U_u};
      for each j ∈ N_c(t)
        begin
        Δm_j = α(t)(X − m_j);
        if j ∈ U_u  then σ_j = d_j;
                    else σ_j := (1 − β)σ_j + βd_j;
      U_u = U_u − {j};
      end;
    endelse;
end.
```

Some Remarks:

(1) Although all σ_j are set at a large constant in the very beginning, it is just for informing the compatibility test which units are uncommitted and thus need not be tested. Once a unit is updated, it is immediately re-initialized by its first d_j. An alternative way is to omit this re-initialization, which is in effect equivalent to the way of running the original SOM procedure for a while and then gradually switching on the compatibility test as the initial σ_j is exponentially forgotten by the forgetting parameter β in Eq. (4).

(2) Equation (5) is designed for the case that α is a constant value or reduces slowly. After one unit, as well as its σ_j, has been updated, there will be a certain time interval before the unit can become the best-matching unit again. If α decreases sharply, the present learning rate will be considerably smaller than that of the last updating on the unit, while σ_j remains unchanged during the interval. It follows from Eq. (5) that the present compatibility test on the unit will be much easier to pass, even though X may be quite dissimilar to m_c. This means that the test does not follow the α's change. A solution is to replace Eq. (5) by:

$$\alpha_c' \|x - m_c\| < 4\frac{1-\nu}{\nu}\sigma_c \qquad (6)$$

where α_c' is the value of $\alpha(t)$ at the last updating on this unit c.

(3) In some cases (as in subspace methods of pattern recognition[7]), the magnitude differences of

patterns are not taken into consideration. The similarity for finding the best-matching unit is best measured by

$$\frac{X^t m_c}{\|X\|\|m_c\|} = \max\left\{\frac{X^t m_j}{\|X\|\|m_j\|},\ j = 1, \ldots, M\right\}. \quad (7a)$$

Accordingly, Eqs. (4) and (5) are replaced by

$$\sigma_j = \begin{cases} \beta\sigma_j + (1-\beta)0.5\left(1 + \dfrac{m_j^T m_j'}{\|m_j\|\|m_j'\|}\right) & \text{if } j \in N_c; \\ \sigma_j & \text{if } j \notin N_c; \end{cases} \quad (7b)$$

$$0.5\left(1 + \frac{m_j^T x}{\|m_j\|\|x\|}\right) > \nu\sigma_c \quad (7c)$$

where $0 < \nu < 1$, and all the σ_j are initialized to zero at the very beginning of the procedure. Here m_j' denotes the weight vector obtained from updating the present m_j.

3.3. *Adjustments on adaptability, stability, discriminative ability*

The two variable parameters $N_c(t)$ and $\alpha(t)$ in the original SOM are controlled externally and they decrease slowly with time. $N_c(t)$ controls the topological organization of the map, and $\alpha(t)$ adjusts the adaptability of the map. The choice of these parameters has already been discussed in Ref. 1 and Ref. 8.

In the generalized SOM, two additional constant parameters $0 < \nu < 1$ and $0.5 < \beta \leq 1$ are needed for the compatibility test. It follows from Eq. (4) that σ_j is actually a kind of exponential average of variations in m_j. When $\beta = 1$, it is just the most recent variation (i.e. the scalar increment of $\|\Delta m_c\|$ by learning the previous input). β is a weight factor for using a few recent variations and these smooth out the fluctuation of the last increment. As stated in Sec. 3.1, the parameter ν as a whole controls the sensitivity of the compatibility test. The compatibility test, as a whole, adjusts the adaptability of the map, similarly to $\alpha(t)$.

Although both $\alpha(t)$ and ν adjust the adaptability of the map, their functions are different. $\alpha(t)$ controls the gain or magnitude of the adaptability of every unit. The smaller the value, the smaller the change is made on every unit by adapting to any kind of inputs: α constraints "how much every unit can learn from an input". For convenience, we call it M-type constraint. The effect of α is nonspecific. In contrast, ν controls the scope of the adaptability of each unit: the larger the value, the smaller the set of inputs on which each unit concentrates its learning. Each unit j only has adaptability to learn a small set of inputs specified by its prior weight m_j: ν constraints "what kind of inputs each unit can learn". For convenience we call it S-type constraint. The effect of ν is specific for each unit, and is governed by the prior knowledge of that unit.

Learning stability is achieved through constraining the adaptability. M-type constraints and S-type constraints result in different kinds of stability. In the original SOM, any incoming input will be assigned to one unit by a winner-takes-all process. Since there is no S-type constraint on adaptability, the unit will learn it by a quantity controlled by M-type constraint, regardless of whether the input is considerably different from the prior knowledge of the unit. As analyzed in Sec. 2.4, this results in a repeated cycle of learning the present input and forgetting that previously learned. Although the cycle could be externally broken by gradually switching of the adaptability through slowly reducing $\alpha(t)$, SOM becomes unstable in nonstationary environments and unable to resist abnormal noise, since the prior knowledge will be unrecoverably erased by unexpected new inputs. For the generalized SOM in contrast, when an input is assigned to a unit, the internal S-type constraint on the unit's adaptability will allow the unit to learn the input only if it can refine the prior knowledge of the unit. The input will otherwise be sent to another unit or another block. An unexpected new input will thus not disturb any unit which is incompatible with it. Instead, it is learned by some unit which can learn it, or sent to another block if the present block has no unit which can learn it. As a result, the generalized SOM can work in nonstationary environments and resist abnormal noise.

The S-type constraint on the adaptability in the generalized SOM allows the learning to be stabilized even when $\alpha(t)$ remains constant or does not vanish to zero. The reason is that the S-type constraint only allows a unit to adapt to an input within a class consisting of similar patterns. Even if the unit adapts to the input with a high learning rate $\alpha(t)$, the learned pattern is still within the class. Consequently, the generalized SOM has less requirements on how to prearrange $\alpha(t)$ and its reduction schedule. If the S-type constraint is strong enough, the

learning could even be stabilized without $\alpha(t)$. It follows from Eqs. (4) and (5) that

$$\sigma_{j+1} < (1 - \beta)\sigma_j + 4\frac{1 - \nu}{\nu}\beta\sigma_j = q\sigma_j \quad (8)$$

where $q = (1 - \beta) + 4(1 - \nu)\beta/\nu$. For $0.8 < \nu < 1$, $q < 1$ and $\lim_{j\to\infty}\sigma_j = 0$. Thus for any α value (see Eq. (5)), the learning on every unit can be stabilized to learn only the pattern which is almost identical to that previously learned. Since $q = 1 - (1 - 4(1 - \nu)\beta/\nu$, β has no influence on the stability of learning but on the stabilizing speed.

Another interesting thing is that although the S-type constraint is internal and specific to each unit, the parameter ν supplies an external interface to adjust its strength. As a result, the discriminative ability of every unit is adjustable to the needs of the outside world. The higher the value of ν, the smaller the range of patterns that a unit can adapt to, so that only quite similar patterns could reach a given unit of the resulting map through winner-takes-all. Each unit has fine discriminative ability. In contrast, a lower ν value permits a large range of patterns for a unit to adapt to, so that each unit has coarse discriminative ability.

It should be pointed out that the learning stability and the discriminative ability of the learned map are coupled together; they are just two outcomes of the S-type constraint on the adaptability. For a special case, if $\nu \to 0$, the S-type constraint is totally removed, and the generalized SOM reduces to the original SOM. In using Eq. (5) the interval [0.8, 1) represents the strong S-type constraints; the interval [0.5, 0.8) represents the usual S-type constraints, equivalent to allowing the increment to adapt to the present input to be between one to four times the weighted mean of the previous increments; and the interval (0.0, 0.5) represents the weaker S-type constraint under which the generalized SOM behaves rather close to the original SOM.

4. Simulation Results

Computer simulations were conducted for both the original SOM and the generalized SOM. Four examples are given in this section to illustrate the characteristics of the generalized SOM.

In these examples, the components of the weight vectors are initialized by $0.45 + 0.1x$, x being a random number from a uniform distribution on the [0.0, 1.0) interval. The parameters $N_c(t)$ are decreased in the way of Ref. 8 for both the original and the generalized SOM. During the first n_1 steps, the radius r of $N_c(t)$ decreases from r_0 (generally including the majority of units in the map) to one (including the best matching unit and its 8 neighbours) linearly with time; thereafter r stays at the value of one. As for the parameter α, we simply take it as a constant, i.e. let $\alpha = 0.1$, for the following reasons:

(1) There are a variety of possibilities for choosing the schedule for the parameter $\alpha(t)$. Outcomes of the simulations are difficult to attribute to the compatibility test or the schedule $\alpha(t)$.

(2) For the original SOM, simulations performed reducing $\alpha(t)$ in the usual way,[8] lead to results similar to those obtained with fixed $\alpha = 0.1$. SOM can already stabilize quite well with $\alpha = 0.1$ if there are no unexpected inputs.

(3) Here our main interest is to observe if an organized map could be washed out by unexpected inputs. In the original SOM this could not be avoided for any given choice of $\alpha(t)$ unless there is an omniscient supervisor who knows when unexpected inputs come and externally shuts off the learning.

In all the figures for the following examples, the ordinals at the bottom of every box (e.g. 250) express that the map is obtained from presenting input samples during such time interval.

Example 1: The input data are shown in Fig. 5 and Fig. 6. m_T pattern samples come from class C_1, a 2D-uniform distribution with mean $(-0.5, 0.5)$ and variance 0.4 and other m_T pattern samples from class C_2, a 2D-uniform distribution with mean (1.5, 0.5) and variance 0.4. As shown in Fig. 5, an input sequence x_t is formed as follows: m_T samples of C_1 are sequentially presented as inputs to train the map. Then m_T samples of C_2 are presented sequentially. The m_T-th sample of C_2 is followed by the 1st sample of C_1 in a new repeated circle. In this example, $m_T = 500$.

For $t < 500$ such input sequence could be regarded as a stationary sequence, so we can observe the behaviour of both SOMs in a stationary environment. For $t > 500$, the whole sequence should be regarded as a nonstationary sequence, since its statistical properties change at $t = 500$, $t = 1000$, . . ., $t = 2500$, (e.g. for $t > 500$, $E(x_t) \neq$ constant, $E(x_t x_{t+r})$ is not independent of t). At $t = 500$, 1000, . . ., $t = 2500$, the new inputs can be regarded as "unexpected inputs", since there is an abrupt change into another class. The behaviour of

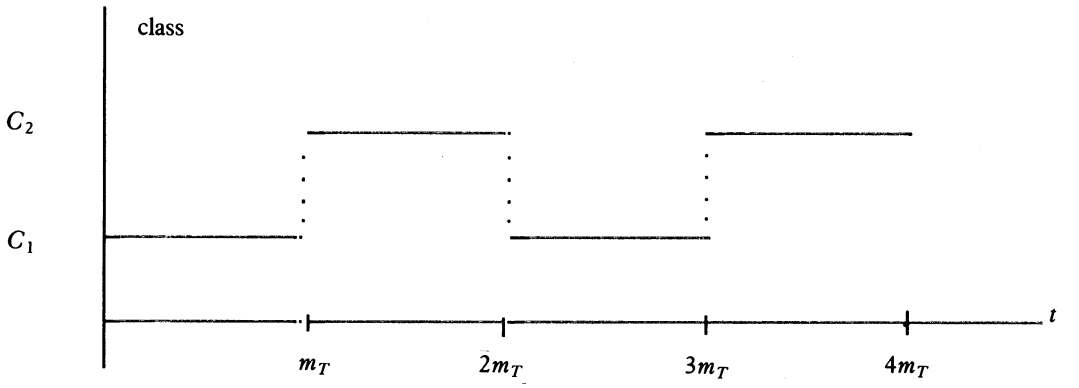


Fig. 5. The input patterns come alternatively from two classes every m_T step.

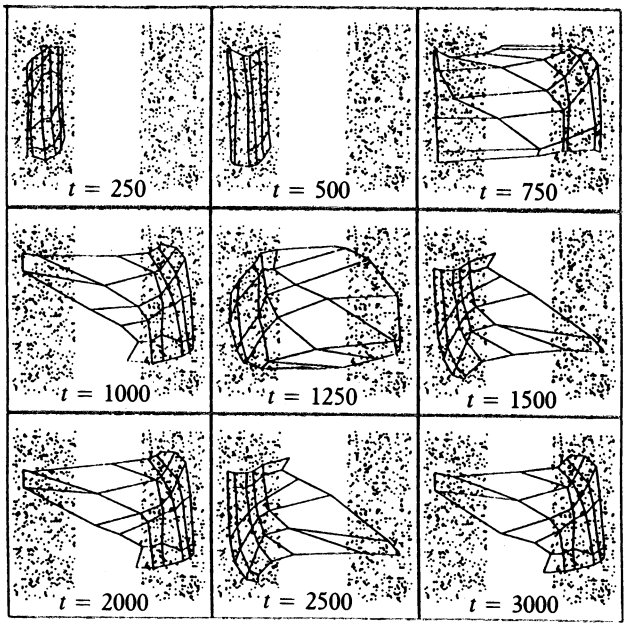
both SOMs in a nonstationary environment can thus be observed.

The parameters of N_c are $r_0 = 3$ and $n_1 = 400$ for both SOMs. Figure 6(a) is the result of the original SOM with a 6×6 map. For the first two boxes, we see that the map gradually adapts to inputs of C_1 during the first 500 samples, and has nearly well stabilized. However, after 250 samples of C_2 at $t = 750$, most of the previously learned map has been erased by adapting to C_2, and at $t = 1000$ nearly all the units have relearned the patterns of C_2. Hereafter, due to inputs again from C_1, the map relearns C_1 and gradually forgets C_2 until $t = 1500$ at which nearly all previously learned information about C_2 has been forgotten. Such an adapting and forgetting cycle is repeated every 500 steps, and the learning cannot be stabilized.

Figure 6(b) is the result of the generalized SOM of two blocks with parameters $\beta = 0.6$, $\nu = 0.6$. Each block is a 6×6 map. During the first 500 steps, the first two boxes show results similar to those of Fig. 6(a): the map of the 1st block gradually adapts to inputs of C_1 and has well stabilized at $t = 500$, while the map of the 2nd block (located at the centre with initial values) has learned nearly nothing since almost all the inputs have been absorbed by the 1st block and need not be sent to the 2nd block. As the samples of C_2 are presented as inputs, they are incompatible with the map previously learned by the 1st block; they are rejected by the compatibility test and sent to the 2nd block. As a result, the prior knowledge of C_1 is well retained. The map of the 2nd block gradually adapts to the C_2 samples (see 3rd and 4th boxes), and the learning has been nearly well stabilized at $t = 1000$. After $t > 1000$, input samples come from C_1, C_2 alternatively every 500 steps. Due to the compatibility test, these samples only refine the map which previously learned patterns of their own class. As a result, the learning has no cycle of adapting-and-forgetting and has nearly well stabilized just after 1000 steps.

In Fig. 6(b) two 6×6 maps are used, but only one 6×6 map is used in Fig. 6(a). One may wonder if the original SOM can get results similar to those of Fig. 6(b) by using two 6×6 maps. The answer is negative, since for each input there is not mechanism for deciding which map should learn it. Each sample is simultaneously presented to both maps, which results in two maps with similar behaviour, as shown in Fig. 6(c) (where the dashed lines represent one map and the solid lines represent the other map).

Example 2: This example is similar to example 1, but with $m_T = 100$. The parameters of $N_c(t)$ are $r_0 = 3$ and $n_1 = 200$ for both SOMs.



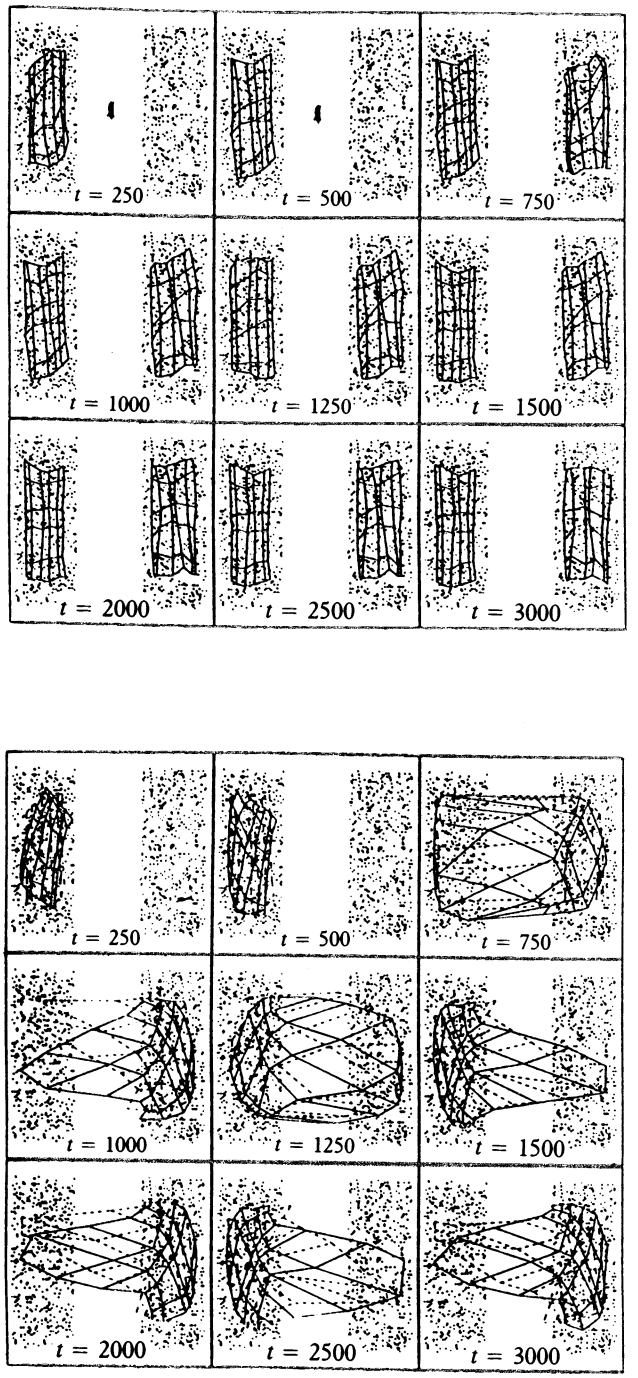


Fig. 6. Example 1: pattern samples are input in the way of Fig. 5, with $m_T = 500$. (a) When the origin SOM was used,the learning of each 500-step-cycle erased what had been learned in the earlier 500-step-cycle, and the map could not be stabilized. (b) When the generalized SOM with two maps was used, $\beta = 0.6$, $\nu = 0.6$. There is no learning-and-forgetting cycle and the maps of both blocks have been nearly well stabilized at $t = 1000$. (c) When the original SOM with two maps was used. The two maps have similar behaviour. The dashed lines represent one map, the solid lines represent the other map.

Figure 7(a) is the result of the original SOM. As in example 1, the patterns learned within the first 100 steps have been nearly erased during the second 100 steps. After $t = 200$, since m_T is short, before the previously learned C_2 (or C_1) has been totally washed away by adapting to the samples of C_1 (or C_2), the samples of C_2 (or C_1) are presented again. In comparison with Fig. 6(a), we see two points:

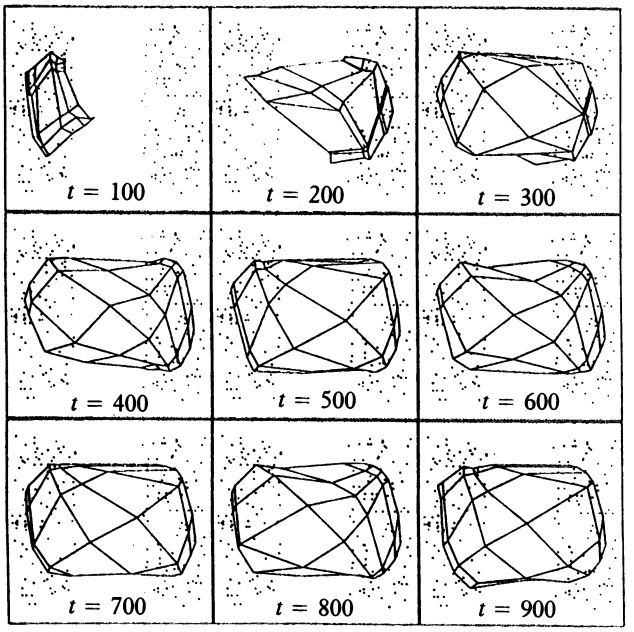
(1) In learning with the original SOM, there exists an unstable cycle of adapting to the present inputs and forgetting the prior knowledge.

(2) The unstable effect of such a cycle becomes weaker when the interval m_T becomes smaller, since in this case patterns could be relearned before they are totally erased.

Figure 7(b) is the result of the generalized SOM with two blocks and parameters $\beta = 0.6$, $\nu = 0.7$. Similar to Fig. 6(b), there is no unstable cycle of adapting-and-forgetting, and the maps of both blocks have been nearly well stabilized after $t = 500$.

The above two examples show that the original SOM exhibits unstable learning in a nonstationary environment. In contrast, the generalized SOM can stabilize its learning via the compatibility test. In the next example, we will show the behaviour of both SOMs when some unexpected noise is introduced to an already stabilized map.

Example 3: As shown in Fig. 8, 800 samples from class C_1, a 2D-uniform distribution with mean (0.5, 0.5) and variance 0.7, form a stationary sequence to train the maps of both SOMs. The N_c parameters of both SOMs are $r_0 = 3$, $n_1 = 400$. The original SOM has a 6×6 map, and the generalized SOM with $\beta = 0.6$, $\nu = 0.5$ has two blocks, with a 6×6 map each. As shown in Fig. 8(a) and Fig. 8(b), after training by the 800 samples, both the map of the original SOM and the map of the 1st block of the generalized SOM have been nearly well stabilized, and the map of the 2nd block of the generalized SOM has learned nothing and still remains at the centre of the box. Other 100 samples from class C_2, a 2D-uniform distribution with mean (2.0, 2.0) and variance 0.05, are presented as abnormal noisy samples to disturb the two stabilized maps. As shown in Figs. 8(a) and 8(b), the earlier learned map of the original SOM, after being disturbed by the 100 noisy samples, has been erased by 25%, while for the generalized SOM, the map of the 1st block has been kept intact and the noisy samples have been absorbed by some units of the 2nd block.



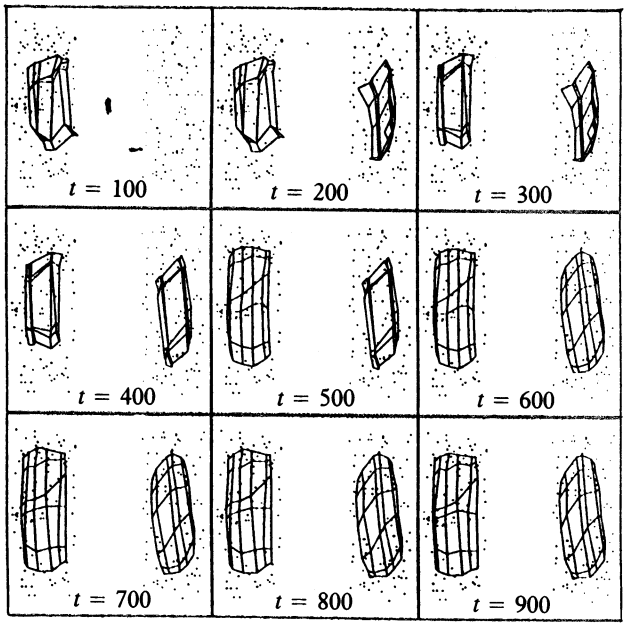
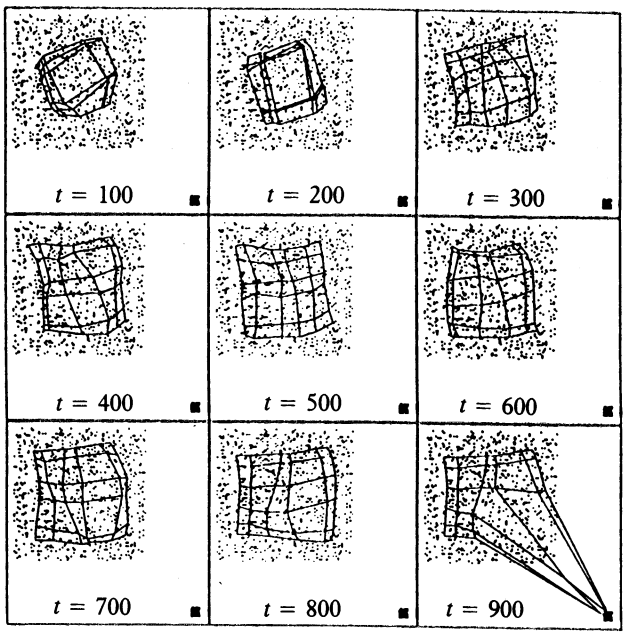


Fig. 7. Example 2: pattern samples are input again in the way of Fig. 5, but with $m_T = 100$. (a) When the original SOM was used, the unstable cycle of learning-and-forgetting still exists, but it becomes weaker since the interval m_T is smaller.(b) When the generalized SOM with two maps was used, $\beta = 0.6$, $\nu = 0.7$. There is no learning-and-forgetting cycle, and the maps of both blocks have been nearly well stabilized at $t = 500$.



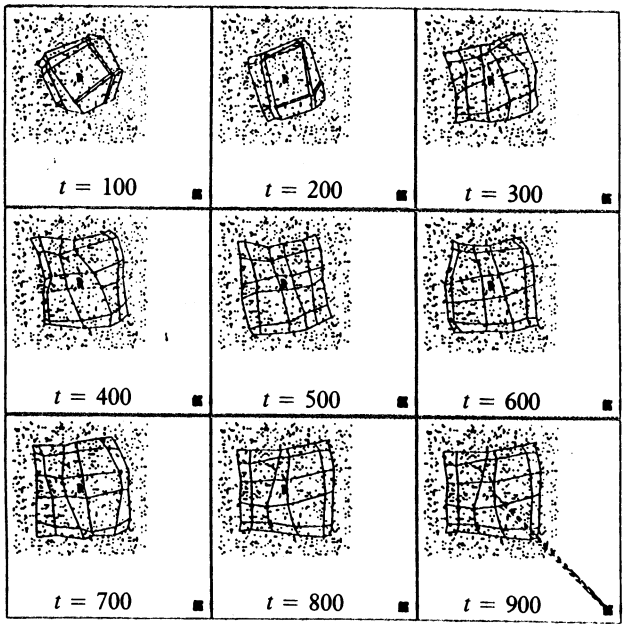


Fig. 8. Example 3: The 800 samples of C_1 are input to train the maps of both SOMs, then 100 abnormal noise samples are primed to disturb the learned maps of both SOMs. (a) For the original SOM, the earlier learned map has been erased by 25% after the presentation of 100 abnormal noise samples. (b) For the generalized SOM with two maps, $\beta = 0.6$, $\nu = 0.5$, during the 800 input samples, the map of the 2nd block has learned nothing and still remains at the centre of the box. The 100 abnormal noise samples have no effect on the map of the 1st block; they are absorbed by some units of the 2nd block.

Example 4: In this example, we show that the discriminative ability of the generalized SOM can be adjusted via the parameter ν. As shown in Fig. 9, 100 samples are generated in such a way that 50 samples come from class C_1, a 2D-normal distribution with mean $(-0.5, 0.5)$ and variance 0.3 and they are followed by 50 other samples from class C_2, a 2D-normal distribution with mean (1.5, 0.5) and variance 0.3. The 100 samples are presented repeatedly to train the maps of the generalized SOM with parameters $\beta = 0.6$, $r_0 = 3$, $n_1 = 400$. In Fig. 9, the two top boxes are the results with $\nu = 0.77$ and the two bottom boxes are results with $\nu = 0.6$. For the two top boxes, the maps of both blocks are used to represent all the samples, each unit responds to only a few samples, and the fine discriminative ability is obtained. For the two bottom boxes, only the map of the 1st block is used to represent all the samples and the map of the 2nd block is still at the centre of the box. Thus each unit of the map of the 1st block will respond to approximately two times as many samples as in the former case and a rougher discriminative ability is obtained.

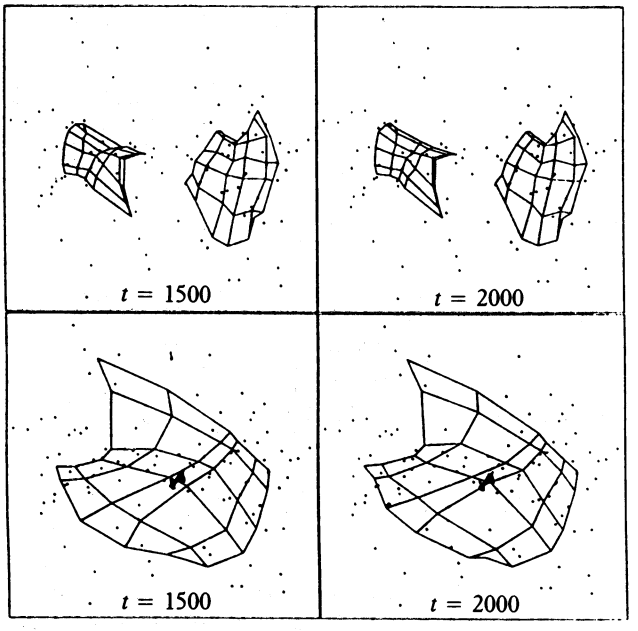


Fig. 9. 100 samples are generated in such a way that 50 samples come from class C_1, and they are followed by other 50 samples from C_2. The 100 samples are repeated to train the maps of the generalized SOM with parameters $\beta = 0.6$, $r_0 = 3$, $n_1 = 400$. The two top boxes are the results with $\nu = 0.77$; the maps of both blocks are used to represent all the samples. The two bottom boxes are the results with $\nu = 0.6$; only the map of the 1st block is used to represent all the samples and the map of the 2nd block is still at the centre of the box.

5. Summary

The conventional competitive learning method has some constraints. One is that it cannot stabilize its learning in a nonstationary environment because of a cycle of learning-and-forgetting. The original self-organizing map inherited this constraint, which makes it difficult to extend the original map for use in nonstationary environments, since the learning-and-forgetting cycle will destabilize its learning when unexpected inputs and abnormal noise appear. As with the top-down expectation of ART, a learned expectation is added to the learning procedure of self-organizing maps to improve its learning in a nonstationary environment. Such a learning expectation has been realized through the compatibility test, which checks whether an input is compatible with the prior knowledge on a map unit before the unit starts to adapt to it. The compatibility test is based on the similarity between the input and the learned weights of the unit, as well as how near the unit is to its stabilized state. Instead of externally and nonspecifically adjusting the adaptability of every unit in the original map, the compatibility test adjusts the adaptability of each unit internally and specifically according to the previously learned patterns. As a result, the unstable cycle of learning-and-forgetting can be avoided and the generalized map can stabilize its learning in an environment with unexpected inputs and abnormal noise. The compatibility test also supplies an external parameter for adjusting the discriminative ability to different needs.

The generalized map consists of multi-maps with a pipeline architecture equipped with a parallel search strategy, so that the learned expectation can be implemented with nearly no extra computing costs in comparison with the original map.

Several comparative experiments have been made between the original and the generalized map. As shown in the examples given in this paper, the preliminary results confirm the expected features of the generalized map. Future work includes experiments on more complex input data, investigating how the parameter ν affects performance, as well as organizing multi-maps into a hierarchical architecture.

Acknowledgments

The work was supported by Tekes Grant 4196/1988 under the Finsoft project. The author wishes to thank Prof. E. Oja for discussion and

support. Thanks also go to J. Lampinen for his help with the author's work on C language programming and computer simulations. The author would also like to thank the referees for their helpful comments and for editing the manuscript.

References

1. T. Kohonen, *Self-Organization and Associative Memory*, Springer, Berlin, 1988.
2. Self-Organization Section, *Proc. IJCNN '89*, Washington, D.C., 1989.
3. Self-Organization Section, *Proc. IEEE ICNN '88*, San Diego, 1988.
4. D. De Sieno, "Adding a conscience to competitive learning", *Proc. 1988 IEEE ICNN*, Vol. I, San Diego, California, July 1988, 117–124.
5. J. Kangas *et al.*, "Variants of self-organizing maps", *Proc. 1989 IJCNN*, Vol. II, Washington D.C., June 1989, 517–522.
6. S. Grossberg, "Nonlinear neural networks: princples, mechanisms and architectures", *Neural Networks* (1988) 17–61.
7. E. Oja, *Subspace Methods of Pattern Recognition*, RSP and J. Wiley, 1983.
8. T. Kohonen, "Competitive learning and self-organization", *Proc. Nordic Symp. Neural Computing*, Espoo, Finland, April 1989.
9. C. von der Malsburg, "Self-organization of orientation sensitive cells in the striate cortex", *Kybernetik* **14** (1973) 85–100.
10. S. Grossberg, "Adaptive pattern classification and universal recording, I: parallel development and coding of neural feature detectors", *Biol. Cybern.* **23** (1976) 121–134.
11. S. Grossberg, "Adaptive pattern classification and universal recording, II: feedback, expectations, olfaction, and illusions" *Biol. Cybern.* **23** (1976) 187–202.
12. G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine", *Computer Vision, Graphics, and Image Processing* **37** (1987) 54–115.