PARALLEL SELF-ORGANIZING MAP①

Li Weigang
Department of Computer Science-CIC, University of Brasilia-UnB,
C. P. 4466, CEP: 70919-970, Brasilia-DF, Brazil, E-mail: Weigang@cic.unb.br

ABSTRACT A new self-organizing map, parallel self-organizing map (PSOM), was proposed for information parallel processing purpose. In this model, there are two separate layers of neurons connected together, the number of neurons in both layer and connections between them is equal to the number of total elements of input signals, the weight updating is managed through a sequence of operations among some unitary transformation and operation matrixes, so the conventional repeated learning procedure was modified to learn just once and an algorithm was developed to realize this new learning method. With a typical classification example, the performance of PSOM demonstrated convergence results similar to Kohonen's model. Theoretic analysis and proofs also showed some interesting properties of PSOM. As it was pointed out, the contribution of such a network may not be so significant, but its parallel mode may be interesting for quantum computation.

Key words artificial neural networks competitive learning parallel computing quantum computing

1 INTRODUCTION

"Once saw, never forgotten" is a sentence which used to describe a human sense and learning sequence. For example, a boy glanced at a lovely girl in a party, on his way home girl's face appears again and again during his thinking. This is a distinct feature of the human brain. For a simple put, the brain is organized in many places in such a way that different sensor inputs are represented by topologically ordered computational maps[1]. In the field of artificial neural networks(ANN), this sequence is called pattern reorganization. Some kinds of artificial neural networks can simulate this sequence by repeated learning. Among the architectures and algorithms suggested for ANN, the self-organizing map(SOM) has the special property of effectively creating spatially organized "internal representations. Kohonen, the author of Ref.[2], was attempting to construct an artificial sytem, SOM, that can show the same behavior as boy's experience through various learning. Following Kohonen's principle of topographic map formation, the spatial location of an output neuron in the topographic map corresponds to a particular domain or feature of the input data[3]. In application, SOM has been particularly successful in various pattern recognition tasks. As mentioned by Grossberg[4], the conventional learning is in terms of serial processing and this slowed down the acceptance of a sampling operation. So, to simulate boy's behavior through just one time's learning is still difficult for SOM.

Recently developed parallel self-organizing map (PSOM)[5] tries to show the same behavior, in which the Kohonen's SOM is reconstructed in a parallel architecture, the number of neurons in both input/output layer and connections between them is equal to the number of total elements of input signals, the weight updating is managed through a sequence of operations among some unitary transformation and operation matrixes, so the conventional repeated training procedure is modified to learn just once. It is interesting to mention that even in parallel processing environment, the developed weight transformation makes PSOM to have the same competitive learning ability and convergence property as the conventional SOM.

In classical computing, PSOM is even less efficient than SOM because the formet needs ex-

① Supported by the CNPq under contract No.521442/97-4 (NV) Received Sep. 24, 1998

tra competitive operations and weight transformations. On the other hand, putting every data point of input pattern as a neuron of layer is almost impossible. Suppose there are signals $x(i) \in x$, $i = 1, 2, \cdots, M$); one input neuron is needed using SOM, but M input neurons are needed in PSOM. In a common case $M = 1\,000$, and no any conventional computer can process PSOM with 1 000 input neurons yet!

In quantum computing, the unique characteristics of quantum theory may be used to represent information with a neuron number of exponential capacity[6,7]. Using quantum representation $x(i)$, $i = 1, 2, \cdots, M$, the number of neurons is exponentially reduced to $\log_2 M$. When $M = 1\,000$, with quantum computing, PSOM just needs 10 neurons in both input and output layer. In quantum computing, the competitive operations and weight transformation will operate simultaneously[5]. Realization of a parallel SOM in quantum computing has motivated the development of this new model, PSOM.

Since Benioff[8] and Feynman[9] discovered the possibility of using quantum mechanical system for reasonable computing and Deutsch[10] defined the first quantum computing model, the quantum computation has been developed as a multidiscipline. Specially in recent years, the appearances of Shor's factoring algorithm[11] and Grover's search algorithm[12] speeded up the development in this area. As an index of quantum computation study situation, a statistical result of the numbers of e-print paper in quantum physics[13] maintained by Los Alamos National Laboratory shows this tendency: the increase in monthly averages reaches almost 40 %, and 108 papers were published only in June 1998. There are some selected literatures[10-12,8,14,15] which can help readers to get a basic conception of quantum computation.

In the field of artificial neural networks (ANN), some pioneers introduced quantum computation into analog discussion, quantum associative memory, parallel learning and empirical analysis[16-19,6,7]. These researches constructed the base for further study of quantum computation in artificial neural networks, specially Vantura and Martinez's quantum associative memory has attracted more attention[6].

When comparing the quantum computation with artificial neural networks, one may find that it is necessary to modify the structure and learning manner of ANN to combine quantum parallelism. So the main purpose of this paper is to study new structure and learning algorithm of self-organizing map(SOM).

2 KOHONEN'S MODEL AND LEARNING ALGORITHM

Kohonen's model is particularly useful for understanding and modeling cortical maps in the brain. The main objective of SOM is to transform an incoming signal pattern of arbitrary dimension into an one or two-dimensional array of postsynaptic neurons, it's structure is shown in Fig.1. The input vector represents the set of input signals: $x = [x(1), x(2), \cdots, x(M)]'$. The synaptic weight vector of neuron j is denoted by $\omega_j = [\omega_j(1), \omega_j(2), \cdots, \omega_j(M)]'$, $j = 1, 2, \cdots, M$.

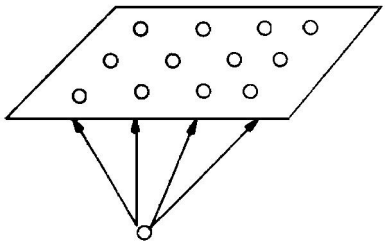

Fig.1 Kohonen's SOM

There are four basic steps involved in Kohonen's competitive learning algorithm:

(1) Initialization. Choose random values for the initial weight vectors $\omega_j(0)$. the only restriction here is that the $\omega_j(0)$ must be different for $j = 1, 2, \cdots, M$, where M is the number of neurons in the output layer. It may be desirable to keep small the magnitude of the weight.

(2) Sampling. Draw a current training time sample $x(t)$, $t = 1, 2, \cdots, T$, from the input distribution with a certain probability; The vec-

tor $x(x(i) \in x, i=1, 2, \cdots, M)$, represents the sensory signal. For $T > M$, it depends on the requirement of the training precision.

(3) Similarity matching. Find the best-matching(winning) neuron $I_c(x)$ at time t, using the minimum-distance Euclidean criterion:

$$I_c(x(t)) = \min \boldsymbol{d}_j(t) = \min \| x(t) - \omega_j(t) \|, \quad j=1, 2, \cdots, M \tag{1}$$

(4) Updating. Adjust the synaptic weight vectors of all neurons, using the update formula:

$$\omega_j(t+1) = \omega_j(t) + \eta(t)[x(t) - \omega_j(t)], \quad j \in \Lambda_{Ic(x)}(t) \tag{2}$$

$$\omega_j(t+1) = \omega_j(t), \text{ otherwise}$$

where $\eta(t)$ is the learning-rate parameter, and $\Lambda_{Ic(x)}(t)$ is the neighborhood function centered around the winning neuron $I_c(x)$; both η_0 and Λ_0 vary dynamically during learning for best results. For simplicity, $\eta(t) = \eta_0[1.0 - t/T]$ and $\Lambda_{Ic(x)}(t) = \Lambda_0[1.0 - t/T]$[19], where η_0 is the initial value of $\eta(t)$ and Λ_0 is the initial value of $\Lambda_{Ic(x)}(t)$.

In step 3, to find the best-matching (winning) neuron $I_c(x)$ at time t, $O(M-1)$ comparisons are needed. In step 4, to get a stable $\omega_j(t)$, the training iteration may take $O(T)$ times depending on the input distribution of $x(i)$, in many cases $T > M$. This means that the step 2 will take $\Omega(T \cdot (M-1))$ times.

3 PARALLEL SELF-ORGANIZING MAP AND LEARNING ALGORITHM

The structure of PSOM is based on the Willshaw-von der Malsburg's mode[11], which consists of a two-dimensional array of presynaptic neurons and a two-dimensional array of postsynaptic. Comparing with Willshaw-von der Malsburg's model, three main differences are: 1) the number of the presynaptic neurons equals the number of total elements of input vector; 2) there is just one connection between an input and output neuron; 3) the competitive learning is realized through a sequence of the matrix multiplication which is a facility for parallel processing; The structure of the PSOM is shown in Fig.2.

Suppose there is a signal $x' = (x(1), x(2), \cdots, x(M))$. The structure of PSOM is designed with a two-dimensional array of presynaptic neurons ($M^{1/2} \cdot M^{1/2}$) and a two-dimensional array of postsynaptic neurons ($M^{1/2} \cdot M^{1/2}$), $\boldsymbol{y}' = (y(1), y(2), \cdots, y(M))$. Every neuron $x(i)$ of input layer just has one link with the neuron $y(i)$ of ouput layer with the weight $\omega^t(i) \in (\omega^t)' = (\omega^t(1), \omega^t(2), \cdots, \omega^t(M))$, where t is the current operation time, $t = 0, 1, 2, \cdots, T$. At this moment, T equals $(M-1)$. The following are the main steps of PSOM's competitive learnng:

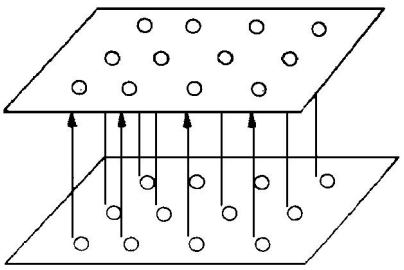

Fig.2 The structure of PSOM

(1) Weight Initialization. Choose random values for the initial weight vectors ω^0. The only restriction here is that the $(\omega^0)' = (\omega^0(1), \omega^0(2), \cdots, \omega^0(M))$ must be different for $i = 1, 2, \cdots, M$. It may be desirable to keep the magnitude of the weight small.

(2) Similarity matching. Repeat the steps 2, 3, 4 for T times. Let $(\omega^t)' = \boldsymbol{v}'$(initially, $\omega^t = \omega^0$) and calculate

$$(\boldsymbol{d}^t)' = \| (x - \omega^t)' \| = \| x(1) - \omega^t(1) \|, \| x(2) - \omega^t(2) \|, \cdots, \| x(M) - \omega^t(M) \| \tag{3}$$

and find the best-matching (winning) at time t, using the Euclidean criterion, if,

$$\boldsymbol{d}^t(k) = \min \boldsymbol{d}^t, \tag{4}$$

where $\boldsymbol{d}^t(k)$ is the minimum Euclidean distance. In this case, $y^t(k)$ is the winner from the competition. $(y^t)' = (y^t(1), y^t(2), \cdots, y^t(k-1), y^t(k), y^t(k-1), \cdots, y^t(M)) = (0, 0, \cdots, 0, 1, 0, \cdots, 0)$.

(3) Updating. Adjust the synaptic weight vectors of all neurons, using Equation(5):

$$\omega^{t+1}(i) = \omega^t(i) + \eta(t)[x(i) - \omega^t(i)], \quad i \in \Lambda_k(t) \tag{5}$$

$\omega^{t+1}(i) = \omega^{t}(i)$, otherwise

where $\eta(t)$ is the learning-rate parameter, and $\Lambda_k(t)$ is the neighborhood function centered around the winning neuron k; both η_0 and Λ_0 vary dynamically during the learning for best results. For simplicity, $\eta(t) = \eta_0[1.0 - t/T]$ and $\Lambda_k(t) = \Lambda_0[1.0 - t/T]$[19], where η_0 is the initial value of $\eta(t)$ and Λ_0 is the initial value of $\Lambda_k(t)$. The vector ($\omega^{t+1}(1)$, $\omega^{t+1}(2)$, …, $\omega^{t+1}(M)$) are generated from this step.

(4) Stop condition. Verificate the condition in Equation (6), if it is certified then go to step 6. A precision vector ε, ($\varepsilon(i) \in \varepsilon$, $i = 1, 2, \cdots, M$), is simply defined as $\varepsilon' = (\varepsilon, \varepsilon, \cdots \varepsilon)$, where ε is a certain small value depending on the precision requirement of the problem:

$$\omega^{t+1} - \omega^{t} < \varepsilon \tag{6}$$

(5) Reorganizing the order of vector ω. Multiply the weight transformation matrix Q & $M^{1/2} \cdot M^{1/2}$ with ω^{t+1}, where Q is an unitary matrix i.e. $QQ^{-1} = 1$. Then, get a new vector

$$v' = (\omega^{t+1})'Q = (\omega^{t+1}(1), \omega^{t+1}(2), \cdots, \omega^{t+1}(M)) \cdot \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ & & & \cdots & & & \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix} = (\omega^{t+1}(2), \omega^{t+1}(3), \cdots, \omega^{t+1}(1)) \tag{7}$$

(6) Registering. Save ω^{t+1} and stop.

4 CLASSIFICATION EXAMPLE

In order to compare the performance of the above two algorithms, a classification example is shown in Table 1, whose data are represented in Cartesian two dimension(2D) space[20].

Table 1 Cartesian 2D space data

X1	X2	Output
1.2	3.0	1
9.4	6.4	-1
2.5	2.1	1
7.9	8.4	-1

4.1 Kohonen's resolution

Fig.3 shows the architecture of the Kohonen based learning network for this classification task. Two prototypes (A and B) are selected, one to represent each data cluster. The weight vector in node A is randomly initialized to 2, 4; and in node B is to 8, 6.

Usually, Kohonen learning selects data points for analysis in random order. This paper takes the point of Table 1 in the order from top to bottom for easier comparison with other method. Table 2 shows the training results, where t means the current training time, ($d(1)^{1/2}$ & $(d(2)^{1/2}$) are the Euclidean distance. After four training iterations, the weight vector in node A converged to 2.05, 2.8; and in node B to 8.3, 7.3.

4.2 PSOM's resolution

Using the proposed model for two dimension data and two prototypes classification problem, 4·4 neurons are needed in both input and output layer. Fig.4 shows the distribution of input data, connections and the weights for each prototype. The initial weights are carefully selected in this example in order to show the sequence of the weight transformation in every competition. For example, the weights ω^0_{A11},

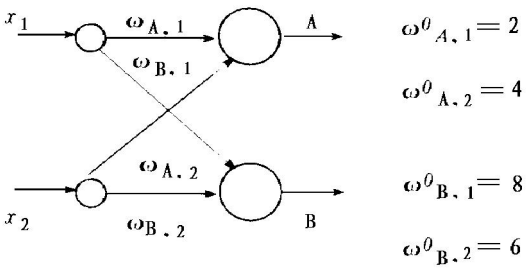

Fig.3 Kohonen model for classification

Table 2 Training results of using Kohonen's algorithm

T	$\omega_{A,1}$	$\omega_{A,2}$	$\omega_{B,1}$	$\omega_{B,2}$	$d(1)$	$d(2)$
0	2.0	4.0	8.0	6.0	1.64	55.24
1	1.6	3.5	8.0	6.0	69.25	2.12
2	1.6	3.5	8.7	6.2	2.77	55.25
3	2.05	2.8	8.7	6.2	65.58	5.48
4	2.05	2.8	8.3	7.3	12.04	122.2

ω^0_{A12}, ω^0_{A13}, ω^0_{A14} equal 4, 1, 2, 3, the others are also shown in Table 3. Where, $(d(1))^{1/2}$, $(d(2))^{1/2}$ are the Euclidean distance; i means original order of weights, e.g. 1, 2, 3, 4 means that the weight vector is in the order of ω^0_{A11}, ω^0_{A12}, ω^0_{A13}, ω^0_{A14}. After one Q transformation, the order is 2, 3, 4, 1 and the weight vector is in order of ω^1_{A12}, ω^1_{A13}, ω^1_{A14}, ω^1_{A11}. After four times Q transformation, the weight vector in node A is converged to 1.61, 2.18, and in node B is 8.63, 8.25.

Obviously, the convergence of the weights obtained from both models shows the same tendency. Through this example, the results demonstrated the equivalence of the two models.

5 PROPERTIES OF PSOM

SOM has received much attention in literature because the process of its organization is fundamental to the organization of the brain. Some intuitive principles of self-organization were summarized by von der Malsburg[22, 1] as:

(1) Limitation of resources leads to competition among synapse and therefore the selection of the most vigorously growing synapses at the expense of the others.

(2) Modifications in synaptic weights tend to cooperate.

Comparing PSOM with SOM, the developed algorithm shows the satisfaction of the above principles. The detail explanation and the main properties of PSOM are as follows.

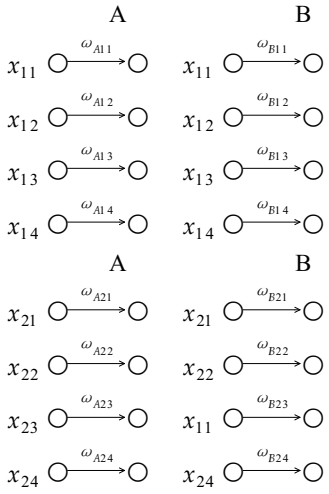

Fig.4 PSOM model for classification

Table 3 Training results using PSOM

i	$\omega_{A,1}$	$\omega_{A,2}$	$\omega_{B,1}$	$\omega_{B,2}$	$d(1)$	$d(2)$
1	4	2	7	9	8.84	69.64
2	1	3	8	6	82.12	2.12
3	2	4	9	7	3.86	66.26
4	3	1	6	8	78.77	3.77
2	1	3	8.7	6.2	0.04	66.49
3	2.25	3.05	9	7	62.35	0.52
4	3	1	6	8	1.46	47.06
1	4	2	7	9	56.17	1.17
3	2.25	3.05	9.2	6.7	1.11	77.69
4	3	1	6	8	70.12	14.12
1	4	2	7	9	2.26	67.86
2	1.1	3	8.7	6.2	75.4	5.48
4	3	1	6	8	7.24	48.04
1	4	2	7	9	48.52	12.52
2	1.1	3	8.3	7.3	2.77	60.68
3	1.72	3.03	9.2	6.7	67.02	4.58
1	4	2	7	9	8.84	69.64
2	1.1	3	8.3	7.3	72.58	2.02
3	1.73	3.02	8.55	7.55	1.46	66.31
4	3	1	6	8	78.77	3.77

5.1 Once learning mechanism

The learning mechanism of PSOM is different from SOM, but functionally equal. As the descriptions in the initial of section 3, when defining the structure of PSOM, the number of the presynaptic neurons equals the number of total elements of input vector and there is just one connection between a neuron from input and a neuron from output layer. PSOM draws total samples of x, ($x(i) \in x$, $i = 1, 2, \cdots, M$), just once. The competition and weight updating are realized via a sequence of the operations which is a facility for parallel processing, so the conventional repeated learning procedure is modified to learn just once in PSOM. From this point of view, **property 1** is introduced, that is, PSOM learns input signals x($x(i) \in x$, $i = 1, 2, \cdots, M$) just once and the weights are updated via a sequence of t times parallel operations.

The PSOM's algorithm of section 3 can be

resumed in following sequences:

Step 1, Input x, i.e. PSOM learns from outside.

Step 2, One operation of competition and updating, for t times:

1) $\omega^t = v$(in first operation, $\omega^t = \omega^0$);

2) $d^t = \| x - \omega^t \|$;

3) $d^t(k) = \min d^t$;

4) $\omega^{t+1}(i) = \omega^t(i) + \eta(t)[x(i) - \omega^t(i)]$, $i \in \Lambda_k(t)$;

$\omega^{t+1}(i) = \omega^t(i)$, otherwise;

5) If $\omega^{t+1} - \omega^t > \varepsilon$, go to 1), else go to step 3;

6) $v = \omega^{t+1} Q$, go to 1);

Step 3, Saving ω^{t+1} and stop.

The signals x is input to system only at the beginning of the algorithm at step 1, and the operations of competition and weight updating are executed through step 2. The step 1 just passes through one time, so the **property 1** is proved. At the same time, PSOM's competitive weight updating sequence shows the satisfaction of the principle 1 of SOM.

5.2 Weight transformation

By second principle of SOM, modifications in synaptic weights tend to cooperate. In PSOM, there is just one connection between neurons of input and output layers. Cooperation among neurons may be impossible when depending only on the map's structure. To satisfy this principle, weight transformation Q is introduced in PSOM aiming at getting information from every neuron for full competition during weight updating and avoid a local minimum. This transformation will be used $(T-1)$ times during the competitive and updating operations of PSOM. When using Q transformation, the position of all elements of ω^t will be changed after every repeated multiplication. Table 4 shows the training results of prototype B from PSOM using the data of Table 1, with and without transformation. In the case of no transformation, the minimum Euclidean distance $d_{\min}$ slides down toward the direction relating to point 9.4, 6.4 of Table 1. In this case, any time training is no more meaning due to the local minimum. And the weights of prototype A is converged to point 2.5, 2.1; the weights of prototype B is converged to point 9.4, 6.4. The situation is better with the transformation Q. The Table 4 gives the competitive and updating results of prototype B without local minimum using weight transformation. Information exchange using weight transformation makes PSOM to have the competitive learning ability and convergence property of the conventional SOM. These properties will be proved in the next two subsections.

Table 4 Efficiency of transformation Q

t	With Q			Without Q		
	$\omega_{B,1}$	$\omega_{B,2}$	$d_{\min}$	$\omega_{B,1}$	$\omega_{B,2}$	$d_{\min}$
0	8	6	2.12	8	6	2.12
1	9	7	0.52	8.7	6.2	0.52
2	8.7	6.2	5.58	9.05	6.3	0.133
3	9.2	6.7	4.58	9.255	6.35	0.033
4	8.3	7.3	2.02	9.313	6.375	0.001

5.3 Convergence property

Ritter and Schulten analyzed a Markovian algorithm for the formation of topologically correct feature maps proposed by Kohonen[23] and proved that the convergence to an equilibrium map can be ensured by a criterion for the time depending on the learning step size. The following **property 2** shows the general description of this convergence property of SOM.

Let X denote a spatially continuous input (sensory) space, the topology of which is defined by the metric relationship of the vector $x \in X$. Let A denote a spatially discrete output space, the topology of which is endowed by arranging a set of neurons as the computation nodes of a layer. Let Φ denote a nonlinear transformation called a feature map, which maps the input space X onto space A as shown by $\Phi: X \to A$. That is **property 2**. The self-organizing feature map Φ, represented by the set of synaptic weight vectors $\{\omega_j | j = 1, 2, \cdots, M\}$, in the output space A, provides a good approximation to the input space X.

This convergence property can also be described in following form:

Lemma 1 for a SOM, after t times training, specially $t \to \infty$, $\lim_{t\to\infty} \Delta\omega_j(t) = [x(t) - \omega_j(t)] = 0$, then, $\lim_{t\to\infty}(\omega_j(t+1) - \omega_j(t)) = 0$.

Using **property 2** and **lemma 1**, the following description try to demonstrate the equivalence of PSOM and SOM in the sense of the convergence. That is **property 3**, PSOM has the same convergence property as Kohonen's SOM.

Suppose a simple SOM as (a) and PSOM as (b) of Fig.5 where $M=3$, it can be easily generalized to arbitrary M situation. To verify the convergence of the map during weight updating, weights are noted $\omega^t(i, j)$, $i=1, 2, \cdots, M$, for SOM and $\omega_p^t(i, j)$ for PSOM. This notation means that the weight is updated from the connection between input $x(i)$ with output neuron $y(j)$ at iteration t. In case of PSOM, when i is not equal j there are no connection between the input $x(i)$ with output neuron $y(j)$. But the $\omega_p^t(i, j)$, $j \neq j$ really exists due to the Q transformation, it is shown in eqn.(8), considering $t = n \cdot M$ and n is a large integer number.

By PSOM, from Eqn.(5) in case (b) of Fig.5, there is

$$(\omega_p^{t+1})' = \begin{bmatrix} \omega_p^{t+1}(1,1) \\ \omega_p^{t+1}(2,2) \\ \omega_p^{t+1}(3,3) \end{bmatrix} \text{ or } = \begin{bmatrix} \omega_p^t(3,1) + \eta(t)[x(1) - \omega_p^t(3,1)] \\ \omega_p^t(1,2) + \eta(t)[x(2) - \omega_p^t(1,2)] \\ \omega_p^t(2,3) + \eta(t)[x(3) - \omega_p^t(2,3)] \end{bmatrix} \quad (8)$$

It should be noted that: $\omega_p^t(3,1) = \omega_p^{t-1}(2,1) + \eta(t-1)[x(3) - \omega_p^{t-1}(2,1)]$, $\cdots$. All of these calculations are in accordance with the weight transformation.

By SOM, for Equation (2) in the case (a) of Fig.5, assuming that, at $t-1$, the input is $x(3)$, and at t, the input is $x(1)$ (remember that the order of appearance of $x(i)$ has a probability of 1/3, so the sequence of input in this case is $x(3)$, $x(1)$ or $x(2)$ or $x(1)$):

$$(\omega^{t+1})' = \begin{bmatrix} \omega^{t+1}(1,1) \\ \omega^{t+1}(1,2) \\ \omega^{t+1}(1,3) \end{bmatrix} = \mathbf{T} \begin{bmatrix} \omega^t(3,1) + \eta(t)[x(1) - \omega^t(3,1)] \\ \omega^t(3,2) + \eta(t)[x(1) - \omega^t(3,2)] \\ \omega^t(3,3) + \eta(t)[x(1) - \omega^t(3,3)] \end{bmatrix} \quad (9)$$

When n is large enough, specially, $n \to \infty$, $t \to \infty$, from **Lemma 1**, $\Delta\omega^{t+1}(1,1) = [x(1) - \omega^t(3,1)]$, when $t \to \infty$, there is $\lim_{t\to\infty} \Delta\omega^{t+1}(1,1) = 0$, then, $\lim_{t\to\infty} \omega^{t+1}(1,1) - \omega^t(3,1) = 0$.

Comparing above result, there is the same situation for $\Delta\omega_p^{t+1}(1,1)$ in Eqn.(8), $\Delta\omega_p^{t+1}(1,1) = [x(1) - \omega_p^t(3,1)]$, when $t \to \infty$, there is $\lim_{t\to\infty} \Delta\omega_p^{t+1}(1,1) = 0$, then, $\lim_{t\to\infty} \omega_p^{t+1}(1,1) = \omega_p^t(3,1)$.

Similarly, the following two cases of PSOM can also be shown as the cases of SOM:

$$\lim_{t\to\infty} \omega_p^{t+1}(2,2) - \omega_p^t(1,2) = 0.$$

$$\lim_{t\to\infty} \omega_p^{t+1}(3,3) - \omega_p^t(2,3) = 0.$$

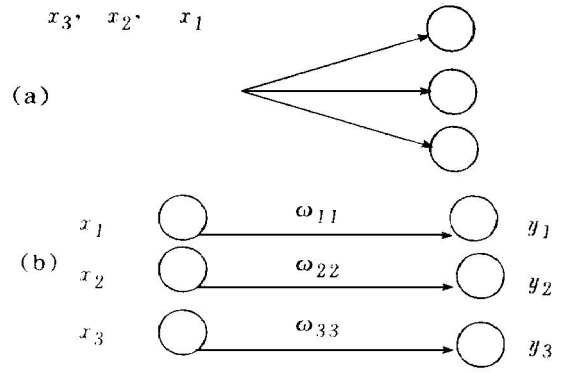

Fig.5 A simple SOM (a) and PSOM (b)

5.4 Equiprobability and participation of competition of input signal x

When Kohonen learning selects the data points for analysis in random order with the probability density ρ being constant, the probability of appearing $x(i)$, ($x(i) \in x$, $i=1, 2, \cdots, M$) is $1/M$. This sequence is equivalent to get data point from x, ($x(i) \in x$, $i=1, 2, \cdots, M$) once in PSOM. After the weight transformation, the opportunity of participation in competition of every input element is equiprobable. From this view point, theorem 4, **property 4**, is introduced: During the weight updating,

in the sense of the participation of competition for every element of input signal x, ($x(i) \in x$, $i = 1, 2, \cdots M$), PSOM is equivalent to SOM in case of the input distribution density ρ is a constant.

Suppose a simple SOM and PSOM as Fig. 5, where $M = 3$, it can be easily generalized arbitrary to M situation. To verify the opportunity of participation of competition of every input element $x(i)$ during weight updating, the weight is noted $\omega^t(i, j)$, $i = 1, 2, \cdots M$, and $j = 1, 2, \cdots, M$, for SOM and $\omega_p^t(i, j)$ for PSOM. This notation means that the weight is updated from the connection between input $x(i)$ and output neuron $y(j)$ at iteration t. In case of PSOM, when i is not equal to j there is no connection between input $x(i)$ and output neuron $y(j)$. But the $\omega_p^t(i, j)$, $i \neq j$ really exists due to the Q transformation. Considering $t = n \cdot M$, n is a large integer number.

In case of SOM, suppose the order of input of elements of x is in $x(1)$, $x(2)$, $x(3)$, the Euclidean distance $d^t = \| x - \omega^t \|$ is

$$d^{t+1} = \begin{bmatrix} [x(1) - \omega^{t+1}(1,1)] \\ [x(1) - \omega^{t+1}(1,2)] \\ [x(1) - \omega^{t+1}(1,3)] \end{bmatrix} \quad (10)$$

$$d^{t+2} = \begin{bmatrix} [x(2) - \omega^{t+2}(2,1)] \\ [x(2) - \omega^{t+2}(2,2)] \\ [x(2) - \omega^{t+2}(2,3)] \end{bmatrix} \quad (11)$$

$$d^{t+3} = \begin{bmatrix} [x(3) - \omega^{t+3}(3,1)] \\ [x(3) - \omega^{t+3}(3,2)] \\ [x(3) - \omega^{t+3}(3,3)] \end{bmatrix} \quad (12)$$

From Eqn.(10), it can be seen that, $x(1)$ appears three times and cooperates the weight of $\omega^{t+1}(1,1)$, $\omega^{t+1}(1,2)$ and $\omega^{t+1}(1,3)$ during the weight updating iterations of $t+1$, $t+2$, and $t+3$. From Eqns.(11), (12), there are similar results for $x(2)$ and $x(3)$.

In case of PSOM, from Equation (3), the Euclidean distance $d_t = \| v - \omega_t \|$ at t is

$$d^{t+1} = \begin{bmatrix} [x(1) - \omega_p^{t+1}(3,1)] \\ [x(2) - \omega_p^{t+1}(1,2)] \\ [x(3) - \omega_p^{t+1}(2,3)] \end{bmatrix} \quad (13)$$

$$d^{t+2} = \begin{bmatrix} [x(3) - \omega_p^{t+2}(3,3)] \\ [x(1) - \omega_p^{t+2}(1,1)] \\ [x(2) - \omega_p^{t+2}(2,2)] \end{bmatrix} \quad (14)$$

$$d^{t+3} = \begin{bmatrix} [x(2) - \omega_p^{t+3}(3,2)] \\ [x(3) - \omega_p^{t+3}(2,1)] \\ [x(1) - \omega_p^{t+3}(2,1)] \end{bmatrix} \quad (15)$$

From Eqns.(13), (14) and (15), it can be seen that $x(1)$ appears three times and cooperates the weights of $\omega_p^{t+2}(1,1)$, $\omega_p^{t+1}(1,2)$ and $\omega_p^{t+3}(1,3)$ during the weight updating iterations of $t+1$, $t+2$, and $t+3$. There are similar results for $x(2)$ and $x(3)$ too.

From **Property 2** and **Lemma 1**, when $t \to \infty$, $\omega_p^{t+2}(1,1)$ approximately equals $\omega_p^{t+1}(1,1)$ and $\omega_p^{t+3}(1,3)$ approximately equals $\omega_p^{t+1}(1,3)$. Then, the behaviors of Eqns.(13), (14) and (15), can be modified as the $x(1)$ appears three times and cooperates the weights of $\omega_p^{t+1}(1,1)$, $\omega_p^{t+1}(1,2)$ and $\omega_p^{t+1}(1,3)$ during the weight updating iterations of $t+1$, $t+2$, $t+3$. These same results can be seen from Eqn.(10) in case of SOM. For $x(2)$ and $x(3)$, the same results can be seen too. This conclusion verifies the equivalence of PSOM and SOM in the sense of participation of competition for every element of input signals when input distribution density ρ is a constant.

5.5 Input signals with multi-dimension

Section 3 described a PSOM's algorithm just for input signals with one dimension. In case of input signals with multi-dimension, the structure of PSOM will not change, but the number of neurons in both input and output layer should be $2 \cdot M$, for this example. For n dimensions input data, the number of neurons in both input and output layer should be $n \cdot M$.

5.6 Stop condition

In many cases, the input signal x, ($x(i) \in x$, $i = 1, 2, \cdots, M$), has a large number of samples, i.e. $M \gg 1$. The weights of PSOM may be converged within a satisfied precision after some iterations of the multiplication and operations. It is not necessary to repeat M times, i.e. $T < M$. But in other cases, M iterations are not enough for a small set of data, ($T > M$) times should be executed. This is the main reason to introduce the stop condition of Eqn.(6).

6 PERSPECTIVE OF PSOM IN QUANTUM COMPUTATION

Depending on the computation environment and the application property, the PSOM may change the opinion of researchers of ANN field, in special case of quantum self-organizing map (QuSOM)[5]. the following will present the general discussion on the comparison of PSOM with Kohonen's SOM.

(1) The most interesting feature of the PSOM is its parallelism property. Quantum mechanics computer can be in a superposition of states and carry out multiple operation at the same time. QuSOM is developed by a sequence of the operations of some transformation and operation matrixes. All the signals are input into the map just once and the output should be converged by repeating certain sequence.

(2) In the classical computation sense, to put all elements of the signals as neurons may be impossible and the operation of PSOM will also be time consuming. Fortunately, in quantum computing, this is not an important problem. The unique characteristics of quantum theory may be used to represent information with a neuron number of exponential capacity[7]. For the input patterns, $x(i)$, $i = 1, \cdots, M$, using quantum representation, the neurons number is exponentially reduced to $\log_2 M$.

7 CONCLUSION

Through comparison one can find a potential of PSOM for the future quantum computation. The future direction of the research is to combine PSOM with quantum computation to form a quantum self-organizing map (QuSOM).

REFERENCES

1 Haykin S. Neural Networks—a Comprehensive Foundation. Englewood Cliffs, USA: Macmillan College Publishing Company, Inc, 1994.

2 Kohonen T. In: Proceedings of the IEEE 78. 1990: 1464 - 1480.

3 Kohonen T. Biol Cybernetics, 1988, 43: 59 - 69.

4 Grossberg S. INNS/ENNS/JNNS Newsletter, Neural Networks, 1998, 11(1).

5 Li Weigang. A Quantum Self-organizing Map, to be published.

6 Ventura D and T Martinez. In: Proceedings of the International Joint Conference on Neural Networks. May 1998: 509 - 13.

7 Ventura D and Martinez T. Quantum Associative Memory, preprint submitted to IEEE Transactions on Neural Networks, June 16, 1998.

8 Bennett C H E, Bernstein G *et al*. SIAM Journal of Computing, 1997, 26(5): 1997.

9 Feynman R. International Journal of Theoretical Physics, 1982, 21: 467 - 488.

10 Deutsch D. In: Proceedings of the Royal Society of London A. 1985: 97 - 117.

11 Shor P W. In: Goldwasser S ed, Proceedings of the 35th Annual Symposium on the Foundations of Computer Sciences. IEEE Computer Society Press, 1994: 124 - 134.

12 Grover L K. In: Proceedings of the 28th Annual ACM Symposium on the Theory of Computing. ACM, New York, 1996: 212 - 19.

13 Los Alamos National Laboratory, e-Print Papers in Quantum Physics: htp://xxx.lanl.gov/archive/quant-ph, by Los Alamos National Laboratory, USA, 1998.

14 Bernstein E and Vazirani U. SIAM Journal of Computing, 1997, 26(5): 409 - 1410.

15 Grover L K. A Framework for Fast Quantum Mechanical Algorithms. LANL e-Print Quant-ph/9711043, 1997.

16 Chrisley R. In: Pylkkänen P and Pylkkö P eds, New Directions in Cognitive Science: Proceedings of the International Symposium, Saariselka. 4 - 9 August 1995, Lapland, Finland, Finnish Association of Artificial Intelligence: 77 - 89.

17 Menneer T and Narayanan A. Quantum-Inspired Neural Networks, Technical Report R329, Department of Computer Science, University of Exeter. Exeter, United Kingdom, 1995.

18 Behrman E C, Niemel J, Steck J E and Skinner S R. IEEE Transactions on Neural Networks, 1996.

19 Perus M. Informatica, 1996, 20: 173 - 183.

20 Darhoff J E. Neural Network Architectures: an Introduction. van Nostrand Reinhold, 1990.

21 Luger G F and Stubblefield W A. Artificial Intelligence. Reading, MA: Addison-Wesley Longman, Inc. 1998.

22 von der Malsburg C. In: SF Zornetzer *et al* eds, An Introduction to Neural and Electronic Networks. São Diego CA: Academic press, 1990.

23 Ritter H and Klaus Schulten. Biol Cybernetics, 1988, 60: 59 - 71.

(Edited by Lai Haihui)