IEICE TRANS. INF. & SYST., VOL.E92–D, NO.3 MARCH 2009

PAPER

An Efficient Initialization Scheme for SOM Algorithm Based on Reference Point and Filters

Shu-Ling SHIEH[†], ***Student Member***, **I-En LIAO**[††*a)], **Kuo-Feng HWANG**[†††], ***and*** **Heng-Yu CHEN**[†], ***Nonmembers***

SUMMARY This paper proposes an efficient self-organizing map algorithm based on reference point and filters. A strategy called Reference Point SOM (RPSOM) is proposed to improve SOM execution time by means of filtering with two thresholds T_1 and T_2. We use one threshold, T_1, to define the search boundary parameter used to search for the Best-Matching Unit (BMU) with respect to input vectors. The other threshold, T_2, is used as the search boundary within which the BMU finds its neighbors. The proposed algorithm reduces the time complexity from $O(n^2)$ to $O(n)$ in finding the initial neurons as compared to the algorithm proposed by Su et al. [16]. The RPSOM dramatically reduces the time complexity, especially in the computation of large data set. From the experimental results, we find that it is better to construct a good initial map and then to use the unsupervised learning to make small subsequent adjustments.

key words: *clustering methods, self-organizing map, unsupervised learning, reference point, filters*

1. Introduction

Self-organizing map (SOM), originally suggested by Kohonen [7], is a very popular unsupervised neural network model for the analysis of high-dimensional patterns in data mining applications. It has been used to map high-dimensional input space and reduce it to two dimensions. Unfortunately, SOM suffered from the trial-and-error method requiring a large amount of iterations for searching a neighborhood preserved feature map [2], [20]. In this paper, we propose an efficient self-organizing map method to improve the performance of SOM. This paper proposes a new algorithm that calculates the distance from all the input points and distance from neuron to reference point in advance. In this way, it can filter out the neurons, which have no chance to become chosen winning neurons and neighborhood neurons. We used a threshold as the search boundary in which to input and conduct vector searches for the Best-Matching Unit (BMU). Another threshold is used as the search boundary in which the BMU finds its neighbors. We ran our algorithm on the data set of a Yeast database and the UCI KDD Archive to illustrate the performance of the proposed method.

In our method, the usage of a reference point is introduced not only for improving the search for the winners among the input neurons but also in finding candidates among the neighbors, which are adjacent to the winners. We propose a reference point to improve efficiency of SOM. The method reduces the time complexity from $O(n^2)$ to $O(n)$ in the steps of finding the initial neurons as compared to the algorithm proposed by Su et al.

The remaining sections of this paper are organized as follows: the next section briefly presents the Kohonen's SOM algorithm and related work on efficient SOM algorithm; in the third section, we give a detailed description of our algorithm and experimental results of the dataset from the Yeast database and UCI KDD Archive are provided in the fourth section. The conclusions are given in the last section.

2. Related Work

SOM algorithm is applicable to large data sets. The goal of SOM is to transform patterns of high dimensionality into a low-dimensional topological map. The training algorithm proposed by Kohonen for forming a feature map is stated as follows [5].

- **Initialization:** Choose random values for the initial weights w_i.
- **Winner Finding:** Find the winning neuron c at time t, using the minimum Euclidean distance criterion

$$c = arg \min_i \| x - w_i \|, i = 1, 2, \ldots, M \tag{1}$$

where $x = [x_1, \ldots, x_m] \in R^m$ represents an input vector at time t, M is the total number of neurons, and $\| \cdot \|$ indicates the Euclidean norm.

- **Weights Updating:** Adjust the weights of the winner and its neighbors, using the following rule:

$$w_i(t+1) = w_i(t) + \eta(t) h_{ci}(t)[x_j(t) - w_i(t)], \tag{2}$$

$$h_{ci} = \exp\left(-\frac{\| r_c - r_i \|^2}{2\sigma^2(t)}\right), \tag{3}$$

where $x_j(t)$ represents an input data at time t, $h_{ci}(t)$ is the topological neighborhood function of the winner neuron c at time t, $\eta(t)$ is a positive constant called

---

Manuscript received April 24, 2008.
Manuscript revised October 9, 2008.

[†] The authors are with the Department of Computer Science and Engineering, National Chung-Hsing University 250, Kuo-Kuang Road, Taichung, Taiwan, and Department of Information Management, Ling-Tung University, Taichung, Taiwan.

[††] The author is with the Department of Computer Science and Engineering, National Chung-Hsing University 250, Kuo-Kuang Road, Taichung, Taiwan.

[†††] The author is with the Department of Information Management, Overseas Chinese Institute of Technology, Taichung, Taiwan.

[*] Corresponding author.

a) E-mail: ieliao@nchu.edu.tw

DOI: 10.1587/transinf.E92.D.422

"learning-rate factor", $r_c \in R^2$ and $r_i \in R^2$ are the location vectors of nodes c and i, respectively. $\sigma(t)$ defines the width of the kernel. Both $\eta(t)$ and $\sigma(t)$ will decrease with time. It should be emphasized that the success of the map formation is critically dependent on the values of the main parameters (i.e., $h_{ci}(t)$ and $\eta(t)$), initial values of weight vectors, and the pre-specified number of iterations.

In the case of a discrete data set and fixed neighborhood kernel, the sum of squared-error of SOM can be defined as follows:

$$SSE = \Sigma_{j=1}^{n}\Sigma_{i=1}^{M} h_{ci} \parallel x_j - w_i \parallel^2 \tag{4}$$

Where n is the number of training samples, and M is the number of map units. Neighborhood kernel h_{ci} is centered at unit c, which is the BMU of input vector x_s, and evaluated for unit i.

Several research papers [1], [3], [14], [19], [21] have attempted to shorten the processing time of SOM. Kohonen originally identified three speedup approaches, namely, Shortcut Winner Search, Increasing the Number of Units in SOM, and Smoothing [8]. The Shortcut Winner Search approach refers to the algorithms for fast identification of BMU. The Shortcut Winner Search scheme can greatly reduce the time for the winner search process for some big maps. The basic idea of the second approach is to estimate good initial values for a map that contains plenty of neurons, based on a much smaller SOM. Finally, the smoothing approach adopts an early stopping heuristic in which the data driven fine tuning process is substituted by a smoothing technique of the neural locations.

Several different approaches have been proposed to improve the conventional SOM algorithm. Jun et al. [4] proposed a self-organizing feature map algorithm based on incremental ordering. Lo and Bavarian [13] addressed the effect of neighborhood function selection on the rate of convergence of SOM algorithm. Fritzke [3] proposed a new self-organizing neural network model that can determine shape as well as size of the network during the simulation in an incremental fashion. Kiang et al. [6] developed a "circular" training algorithm that tries to overcome some of the ineffective or inaccurate topological representations caused by the "boundary" effect. Koike et al. [9] proposed a two-stage self-organizing map algorithm with a threshold operation. Koike's map algorithm showed that the two-stage SOM improves the classification capability for high dimensional data through numerical simulations.

If there were M map neurons in the Kohonen SOM, then the total number of comparison operations to be performed during learning by an exhaustive search of the winners is $O(M^2)$ [8]. Liu and Xu [12] proposed a topological principal component analysis model, which can enhance the learning speed. Koikkalainen and Oja [10] and Truong [18] proposed a tree-structured SOM to improve the winner search reducing the number of search operations to $O(M \log M)$, where M is the total number of neurons. A self-organizing topological tree with $O(\log M)$ was proposed by Xu and Chang [22]. Kusumoto and Takefuji [11] proposed SOM algorithms doing away with learning of neighborhood vectors with $O(\log M)$. Note that if initial weights of each neuron are random, it may not be possible to construct a topologically ordered map. As a result, we will probably need more repetitions to refine the initial map.

Su et al. [15]–[17] proposed an efficient initialization scheme to construct an initial map. They use the self-organizing feature map algorithm to construct a good initial map and make small subsequent adjustments. But their initial phase utilizes time complexity $O(n^2)$, where n is the total number of input points.

In this paper, we propose an efficient initialization scheme for SOM algorithm based on reference point and filters. The proposed algorithm reduces the time complexity from $O(n^2)$ to $O(n)$ in finding the initial neurons as compared to the algorithm proposed by Su et al. The initialization scheme proposed by Su et al. is described as follows:

- **Initialization of the neurons on the four corners**
  At first, select a pair of input data points whose point to point distance is the largest within the two-dimensional neural array of size $K \times L$. The coordinates of the two points are used to initialize the weights of the neurons on the lower left corner (i.e. $W_{K,1}$) and the upper right corner (i.e. $W_{1,L}$). Then, from the remaining training data set, the coordinates of the point which is farthest from the two selected points is used to initialize on the upper left corner (i.e. $W_{1,1}$). Thereafter, the fourth corner can also be found by seeking the farthest point to these three points among the rest of the data points (i.e. $W_{K,L}$).
- **Initialization of the neurons on the four edges**
  Initialize the weights of the neurons on the four edges according to the following equations:

$$\begin{aligned} W_{1,j} &= \frac{W_{1,L} - W_{1,1}}{L-1}(j-1) + W_{1,1} \\ &= \frac{j-1}{L-1}W_{1,L} + \frac{L-j}{L-1}W_{1,1} \\ for\ j &= 2, \ldots, L-1 \end{aligned} \tag{5}$$

$$\begin{aligned} W_{K,j} &= \frac{W_{K,L} - W_{K,1}}{L-1}(j-1) + W_{K,1} \\ &= \frac{j-1}{L-1}W_{K,L} + \frac{L-j}{L-1}W_{K,1} \\ for\ j &= 2, \ldots, L-1 \end{aligned} \tag{6}$$

$$\begin{aligned} W_{i,1} &= \frac{W_{K,1} - W_{1,1}}{K-1}(i-1) + W_{1,1} \\ &= \frac{i-1}{K-1}W_{K,1} + \frac{K-i}{K-1}W_{1,1} \\ for\ i &= 2, \ldots, K-1 \end{aligned} \tag{7}$$

$$W_{i,L} = \frac{W_{K,L} - W_{1,L}}{K-1}(i-1) + W_{1,L}$$

$$= \frac{i-1}{K-1} W_{K,L} + \frac{K-i}{K-1} W_{1,L}$$
$$for\ i = 2, \ldots, K-1 \qquad (8)$$

- **Initialization of the remaining neurons**
  Initialize the remaining neurons according to the following equations:

$$W_{i,j} = \frac{W_{i,L} - W_{i,1}}{L-1}(j-1) + W_{i,1}$$
$$= \frac{j-1}{L-1} W_{i,L} + \frac{L-j}{L-1} W_{i,1}$$
$$for\ i = 2, \ldots, K-1; j = 2, \ldots, L-1 \qquad (9)$$

3. Our Proposed Scheme

In this paper, a new strategy called Reference Point SOM (RPSOM) is proposed for improving SOM performance by using reference point and filters based on two thresholds.

Kohonen's SOM method constructs an initial map randomly. Even though it costs only $O(1)$, it may not be able to construct an ideal initial map. To cope with the initialization problem, a conventional approach is to restart the training procedure with other random weights. The price paid for the trial-and-error method is that we have to waste much time to solve the problem. Instead of using randomly initialized weights, Su et al. proposed a simple initialization scheme to construct a good initial map. However, the initial phase of their algorithm has time complexity of $O(n^2)$. Hence we propose an algorithm to improve Su's method by reducing the time complexity of the initialization phase from $O(n^2)$ to $O(n)$.

To construct a good initial map, the basic idea is to find an appropriate hypercube large enough to cover all the training data points and then to press the hypercube into a plane. In the initialization scheme of Su et al., they started from finding four corners for building a good initial map. In the proposed RPSOM algorithm, we improve Su's method by introducing a reference point to construct a good initial map. It is likely that the selected four corners by our algorithm are different from those by Su's method, but the more same corners there are, the more similar the two initial maps will be.

In the initialization scheme, Su et al. first select a pair of data points whose data distance is the longest one among the data set. This will require $\frac{n \times (n-1)}{2}$ comparisons. In the proposed RPSOM algorithm, we select the first corner point, which has the longest distance to the reference point, only with n comparisons. Each of the other three corner points can also be found in n comparisons. Hence, using a reference point to find the four corners among the data points will improve the time complexity of the initialization scheme proposed by Su et al. from $O(n^2)$ to $O(n)$.

We apply a preprocessing step to calculate the distances between every data point and the reference point, then we use the distances to determine if the distance of two points should be computed. By evaluating the distance of two arbitrary points P_1 and P_2 to the same reference point, we get two values d_1 and d_2 , respectively. If the difference between d_1 and d_2 is small enough, then the Euclidean distance between P_1 and P_2 is calculated. Otherwise, the Euclidean distance is not computed.

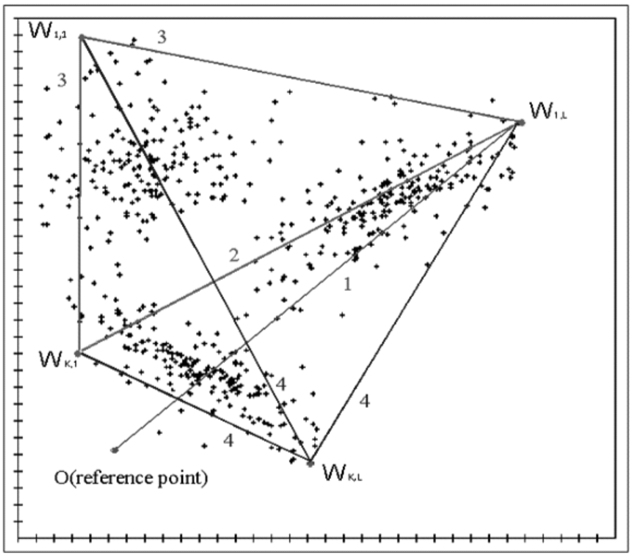

Fig. 1 Initialization scheme with the origin as reference point.

Using reference point is the main theme of this research to reduce the execution time of SOM algorithm. Theoretically, reference point can be any point in the data space. However, the origin is a good choice for its easy computation of the distance between the origin and any input data point. But using the origin as the reference point may affect the search space of finding winner neurons for those data sets that are distant from the origin. As the experimental results shown in the Subsection 4.5, the center of input data points is a better choice for reference point compared to the origin. In the following discussions, either the origin or the central point is used as the reference point.

3.1 Initialization Scheme Algorithm Based on Reference Point

In a large-scale database with high dimensional data points, determining the four corners of the input points as proposed by Su et al. will cost a lot of computation time. Therefore, we use the following method to make improvements to the initialization scheme proposed by Su et al., as shown in Fig. 1.

1. **Finding four corners of the input data set**
   We consider the point with the longest distance to the reference point as the first corner (i.e. $W_{1,L}$). Based on this point, find another point among the data that has the largest distance to $W_{1,L}$, denoted $W_{K,1}$. Then we locate the third corner (i.e. $W_{1,1}$) based on the two points known by searching the point with the largest distance to $W_{1,L}$ and $W_{K,1}$. Thereafter, the fourth corner can also be located by seeking the farthest point to these three

points among the rest of the data points (i.e. $W_{K,L}$). The proposed initialization scheme algorithm using reference point has the following steps:

a. Choose the reference point. For a data set consisting of m-dimensional input points of size n, let $O = (o_1, o_2, o_3, \dots, o_m)$ be the reference point. In this paper, the reference point could be the origin or the center of all data points. If the origin is the reference point, then $O_i = 0$ for $1 \le i \le m$. If the center is the reference point, then $O_i = \frac{1}{n}\sum_{k=1}^{n} x_{k,i}$, $1 \le i \le m$, where $X_k = (x_{k,1}, x_{k,2}, x_{k,3}, \dots, x_{k,m})$ represents the k^{th} input point.

b. Find a point X_p such that

$$p = arg \max_q \parallel X_q - O \parallel, q = 1, 2, \dots, n \quad (10)$$

Let $W_{1,L} = X_p$.

c. Choose X_p, using the following rule:

$$p = arg \max_q \|X_q - W_{1,L}\|, q = 1, 2, \dots, n \quad (11)$$

Let $W_{k,1} = X_p$.

d. Choose X_p, using the following rule:

$$\begin{aligned} p = arg \max_q (\parallel X_q - W_{1,L} \parallel \\ + \parallel X_q - W_{K,1} \parallel), q = 1, 2, \dots, n \end{aligned} \quad (12)$$

Let $W_{1,1} = X_p$.

e. Choose X_p, using the following rule:

$$\begin{aligned} p = arg\max_q (\|X_q - W_{1,L}\| + \|X_q - W_{K,1}\| \\ + \parallel X_q - W_{1,1} \parallel), q = 1, 2, \dots, n \end{aligned} \quad (13)$$

Let $W_{K,L} = X_p$.

2. **Initializing the neurons on the four edges**
After we get four corners on the plane, then the following step is to arrange the neurons on these four edges uniformly by applying the methods described on efficient initialization scheme, as Eqs. (5) to (8).
3. **Initializing the remaininig neurons**
After getting the initial values of the neurons on four edges, apply the Formula (9) to generate the initial values of the remaining neurons.

The first step of the Su et al. algorithm for an input data set of size n is to initialize the neurons on the four corners, Su's algorithm requires $\frac{n\times(n-1)}{2} + 2 \times n$ comparisons. Therefore, the time complexity of their algorithm is $O(n^2)$. However, by applying the reference point initialization scheme, the computation time is linear. The time complexity will be reduced to $O(n)$.

In our experimental results, our initialization scheme finds, the same four or three corners as those found by Su's scheme. But our scheme improves the efficiency by the order of magnitude. Therefore, the reference point initialization scheme algorithm is an efficient and time-saving way in initialization phase.

In the proposed scheme, the selection of the reference point may affect the initial map. However, as the results of our experiments show, the effect is insignificant because the four corners found by using different reference points are almost the same. Therefore, the dependence between accuracy and reference point is very weak. On the contrary, the accuracy is mainly dependent of the selections of two thresholds T_1 and T_2, which will be evident in the subsequent discussions.

3.2 RPSOM Algorithm

The RPSOM algorithm is introduced in this section to improve the performance of the traditional SOM. The proposed RPSOM improves SOM effiency by means of filters based on two thresholds. An input point uses the first threshold, T_1, to define the search boundary for finding the winning neuron. The second threshold, T_2, works as a boundary in which the winning neuron finds its neighborhood neurons. As shown in Fig. 2 and Fig. 3, we assume that the origin is chosen as the reference point.

If an input point searches for the nearest neuron, it just needs to notice the neurons whose distance to the reference point is between $|Id_j - T_1|$ and $|Id_j + T_1|$, where Id_j represents the distance of the input point x_j and the reference point as shown in Fig. 2. After finding the winning neuron for an input, a similar method with threshold T_2 is applied to look for neighborhood neurons of the winner for updating the weights, as shown is Fig. 3. The computation is speed up by using filters of T_1 and T_2.

The whole procedure referred to as the RPSOM algorithm is summarized as follows.

Step 1 Initialization Scheme
The initialization scheme is as described in Sect. 3.1.
Step 2 Winner Finding
We use the minimum Euclidean distance criterion to find the winner neuron c at time t. It consists of the

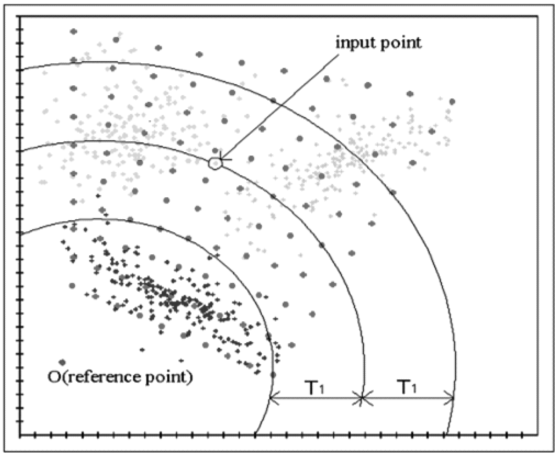

Fig. 2 The threshold T_1 works as search boundary in which input vector finds the winning neurons.

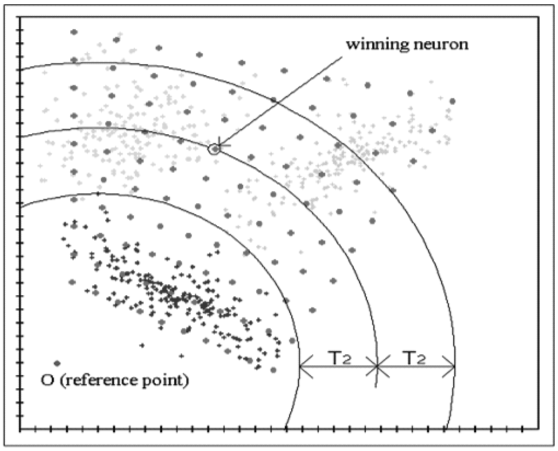

Fig. 3 The threshold T_2 works as a boundary in which the winning neuron finds its neighborhood neurons.

following two steps:

1. Compute the distance Id_j between the data point x_j and the reference point for every input point.

$$Id_j = \sqrt{\sum_{k=1}^{m}(x_{j,k} - o_k)^2}, 1 \leq j \leq n. \tag{14}$$

   Compute the distance Nd_i between the neuron i and the reference point for every neuron.

$$Nd_i = \sqrt{\sum_{k=1}^{m}(w_{i,k} - o_k)^2}, 1 \leq i \leq M. \tag{15}$$

2. Finding the BMU
   If the difference Id_j and Nd_i is not greater than the threshold T_1, then compute Euclidean distances between the neurons and the input point, and find the nearest neuron as the winning neuron c. The pseudo code is as follows:

```
For $j$=1 to $n$
    winner_min_distance=∞
    $i = 1$
      If $|Nd_i - Id_j| < T_1$ and $i \leq M$ do
        dist_inputdata_and_neuron
            $= \| x_j - w_i \|$
        If dist_inputdata_and_neuron
            <winner_min_diststance
          winner_min_distance
            =dist_inputdata_and_neuron
          $c = i$
        End if
        $i = i + 1$
      Else
        Execute Kohonen's SOM algorithm
      End if
Next $j$
```

The first threshold T_1 is to exclude neurons, which are far away from input point and are very unlikely to become the winner. According to Chebyshev's inequality and our experimental results, choosing $T_1 = \mu + \sigma$ as discussed in Sect. 4.5 will have the best results, where μ and σ represent the arithmetic mean and the standard deviation of the distances between every input point and the reference point, respectively.

We get a conclusion from the experiments that the greater that RPSOM thresholds (i.e. T_1 and T_2) are set, the more accuracy the result will be. However, the more computation would be required. Therefore, we try to change the thresholds when time elapses. We use larger thresholds at the very beginning of executing RPSOM, and deduce the thresholds gradually as the process continues, to achieve the balance of accuracy and computations.

Step 3 Updating Weights
Calculate the distance Nd_c between the winning neuron c and the reference point. For arbitrary neuron i, compute the distance between the neuron and the reference point as Nd_i. If the difference between Nd_i and Nd_c is not greater than a threshold T_2, the neuron i is thought to be a feasible neuron, and then the weight of i should be renewed by a proper heuristic function $H_{ci}(t)$. When the renewing happens, the distances of these feasible neurons to the reference point should be calculated and stored again. The pseudo code is as follows:

```
$i = 1$
If $|Nd_c - Nd_i| < T_2$ and $i \leq M$ do

    $H_{ci}(t) = \exp(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)})$

    $W_i(t+1) = W_i(t) + H_{ci}(t)(x_j(t) - W_i(t))$
    $i = i + 1$
Else
    Execute Kohonen's SOM algorithm
End if
```

In the pseudo code, $x_j(t)$ represents an input data at time t; $H_{ci}(t)$ is the topological neighborhood function of the winner neuron c at time t; $\eta(t)$ is monotonic decrease, $0 < \eta(t) < 1$.

The second threshold T_2 , works as search boundary in which the winning neuron finds its neighborhood neurons for updating weights. According to our experiments, choosing $T_2 = \mu$ will have best results.

Step 4 Go to **Step 2** until the pre-specified maximum number of epochs is reached.

4. Experimental Results

In our experiments, the program is written in Java using Bor-

Table 1 Execution time comparison of initialization scheme of RPSOM and that of Su's algorithm running on 2-D artificial dataset.

	Initialization scheme of RPSOM	Initialization scheme of Su's algorithm
Time	< 1 ms	78 ms
Results	These two methods generate the same four corners.	

Table 2 Execution time comparison of initialization scheme of RPSOM and that of Su's algorithm running on a Yeast database.

	Initialization scheme of RPSOM	Initialization scheme of Su's algorithm
Time	4.4 sec	1128 sec
Results	These two methods generate the same four corners.	

Table 3 Execution time comparison of initialization scheme of RPSOM and that of Su's algorithm running on a Census-Income database.

	Initialization scheme of RPSOM	Initialization scheme of Su's algorithm
Time	10.7 sec	16382 sec
Results	These two methods generate the same three corners.	

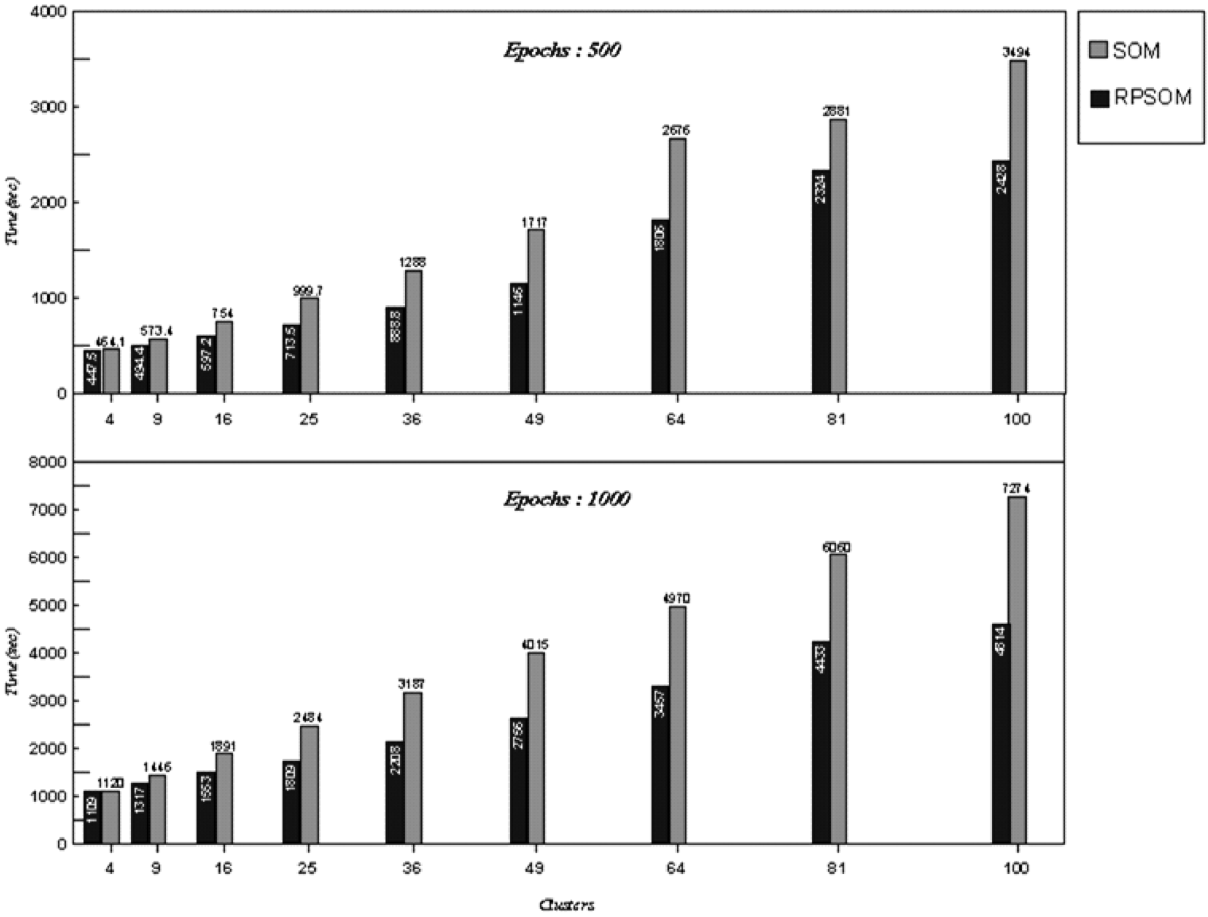

Fig. 4 Execution time comparison of SOM and RPSOM running on Yeast database in 500 and 1000 epochs.

land JBuilder 9 Enterprise Edition. The platform used is Intel Pentium 4 3.2 GHz with 512 MB DRAM and 80 GB hard disk running Windows 2000 Server Pack3.

All experiments ran our algorithms on the datasets from a Yeast database and UCI KDD Archive to illustrate the performance improvement of the proposed method. In the experiment, the execution time of the Kohonen's SOM algorithm is reduced in half in our scheme. At the same time, the sum of the squared error distance in our scheme is also smaller than that of the traditional SOM.

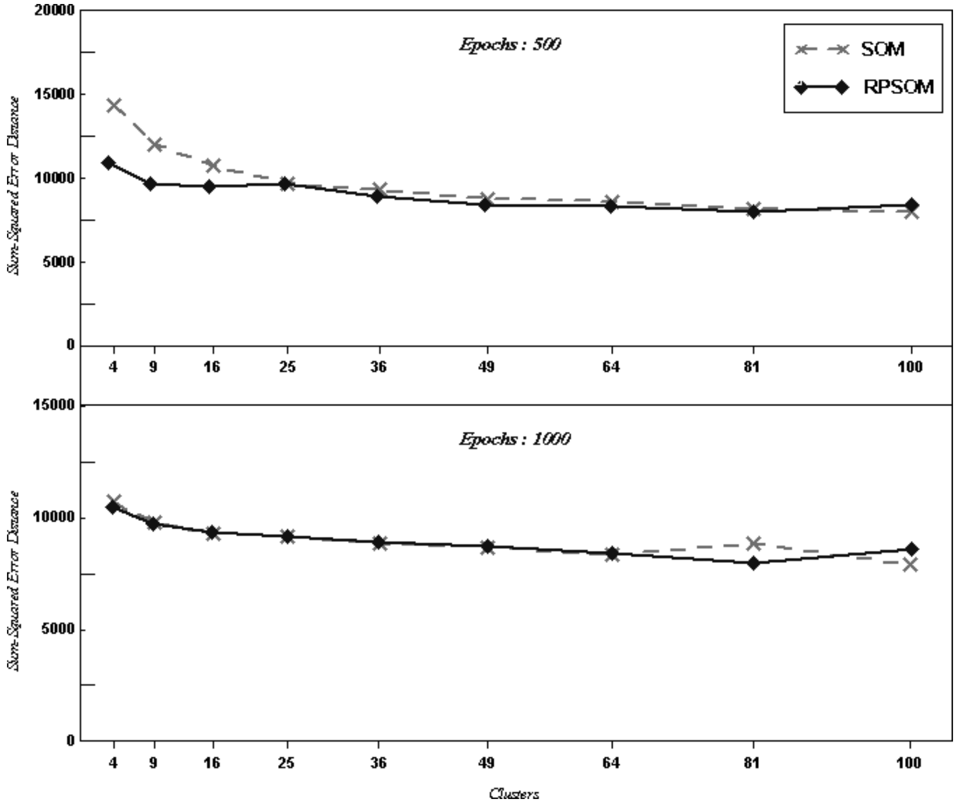

Fig. 5 Sum of squared-error distance comparison of SOM and RPSOM running on Yeast database in 500 and 1000 epochs.

4.1 The Data Sets Used in Our Experiments

We use three data sets to demonstrate the effectiveness of the proposed RPSOM algorithm in our experiments. To achieve a better clustering result and to avoid bad effects produced by noises and outliers, all data sets have been pre-processed using data cleaning normalization schemes.

1. **2-D artificial dataset**
   2-D artificial dataset (http://cilab.csie.ncu.edu.tw/course/cluster/FCM's Data.zip) is a computer simulation data set. The data set contains 579 artificial data points with 2 columns each. The data consists of three shallow elongated parallel clusters in the 2D plane. The dataset was trained by the RPSOM initialization scheme and Su's scheme, respectively. Obviously, our scheme reduces the time complexity dramatically.
2. **Yeast database**
   There are 2467 data records and each record has 81 columns in the Yeast database (http://genome-www.stanford.edu/clustering/Figure2.txt). Thus far there are 6,000 yeast genes that have been sequenced. In this database, there are 79 different experiments for 2467 yeasts genes. For example, the alpha factor is added at different periods to observe the fluorescence of performances. Positive values mean the experimental team has shown a rapid increasing effect than the relevant comparable team. Higher values, result in higher effects. Negative values mean the experimental team has produced a reducing effect. Lower values result in lower effects.
3. **Census-Income database**
   In this experiment, we take the Census-Income Database of UCI KDD Archive (http://kdd.ics.uci.edu) as our input data set. This data set contains census data extracted from the 1994 and 1995 Current Population Surveys conducted by the U.S. Census Bureau. The data contains 199523 records with 41 demographic and employment related fields each. We select 20000 data points, each of which consists of 12 non-numeric fields, randomly from the Census-income database.

4.2 Comparison of Execution Time in the Initialization Phase

The following tables are the results we got based on three datasets, which are a 2-D artificial dataset, a Yeast database, and a Census-Income database, respectively. We used the initialization scheme by Su et al. and RPSOM to look for the four corners from all data points of the three databases. The experimental results show that the proposed RPSOM finds either the same four corners (as shown in Tables 1 and Table 2) or same three corners (as shown in Table 3) compared to Su's method, but the RPSOM has about two orders

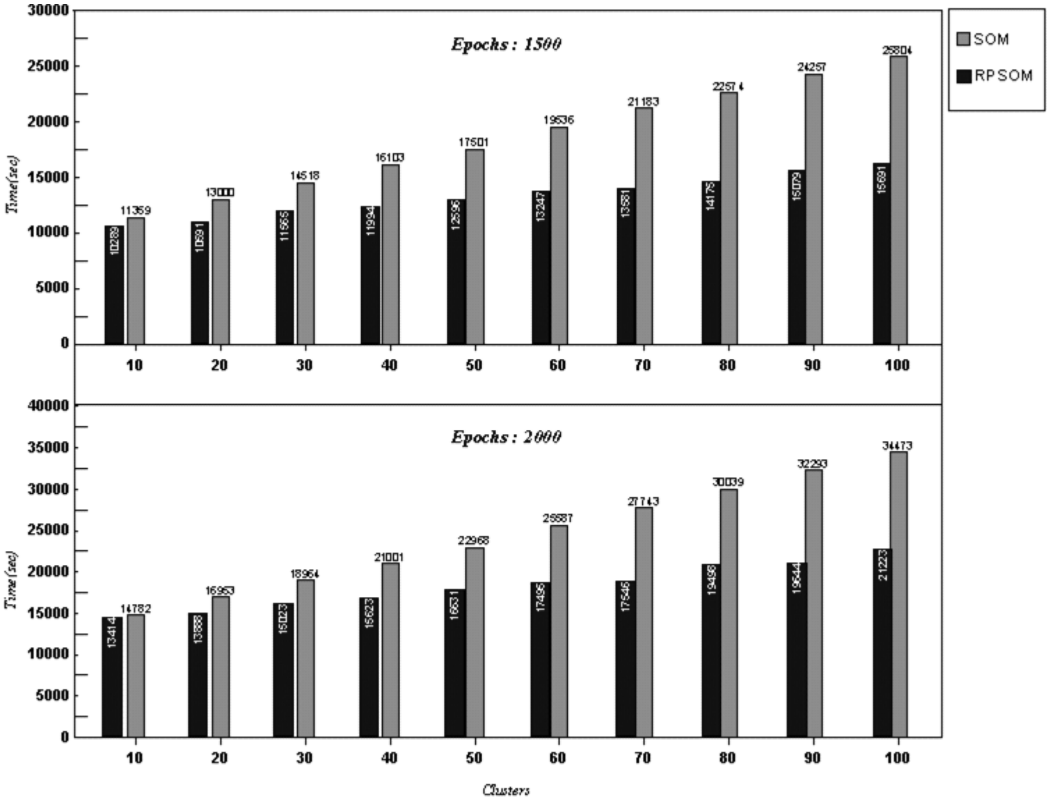

Fig. 6 Execution time comparison of SOM and RPSOM running on 20000 input data in 1500 and 2000 epochs.

of magnitude improvement in execution time for the initialization phase.

4.3 Comparison of Execution Time and Clusters between SOM and RPSOM Using the Yeast Database

In this experiment, we ran the traditional SOM and the RPSOM on the Yeast database, and measured the results on execution time and error distance. The numbers of neurons we used were 4, 9, 16, 25, 36, 49, 64, 72, and 100, in 500 and 1000 epochs, respectively.

In the experimental results as shown in Fig. 4, the larger the clustering number, the more the reduction of execution time. By error distance, it is estimated that RPSOM is smaller than the traditional SOM for smaller clustering number in 500 epochs, whereas the difference is not obvious given larger clustering. However, the results of error distance as shown in Fig. 5 for two schemes are almost the same in 1000 epochs.

4.4 Comparison of Execution Time and Clusters between SOM and RPSOM Using the Census-Income Database

In this experiment, we ran the traditional SOM and the RPSOM on the Census-Income database, and measured the results on execution time and error distance. The numbers of neurons we used were 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100, in 1500 and 2000 epochs, respectively.

In the experimental results as shown in Fig. 6, the RPSOM can reduce the execution time by 40% when executing 100 neurons clustering in 1500 epochs for the Census-Income database. By error distance, it is estimated that RPSOM is smaller than the traditional SOM for smaller clustering number in 1500 and 2000 epochs, whereas the difference is not obvious given larger clustering as shown in Fig. 7.

4.5 The Execution Time with Different Reference Points in the Yeast Database

The choice of reference point influences the search time required. In some cases, if we choose the origin or the median as reference point, the result may cause the search area to be too large to achieve the expected results. Therefore, this experiment takes the central point of input data as the reference point and compares the execution time of these two schemes.

The results show that the center choice is faster than the origin choice in the Yeast database, as shown in Fig. 8. Let us take clustering number 64 as an example. It is 2.92% times faster. Choosing the central point as reference point can avoid the problem of too large a search area when constructing the threshold.

In the experiment of the Yeast database, the larger the number of clusters is, the more the ratio of computing is reduced. By error distance, this experiment takes the central

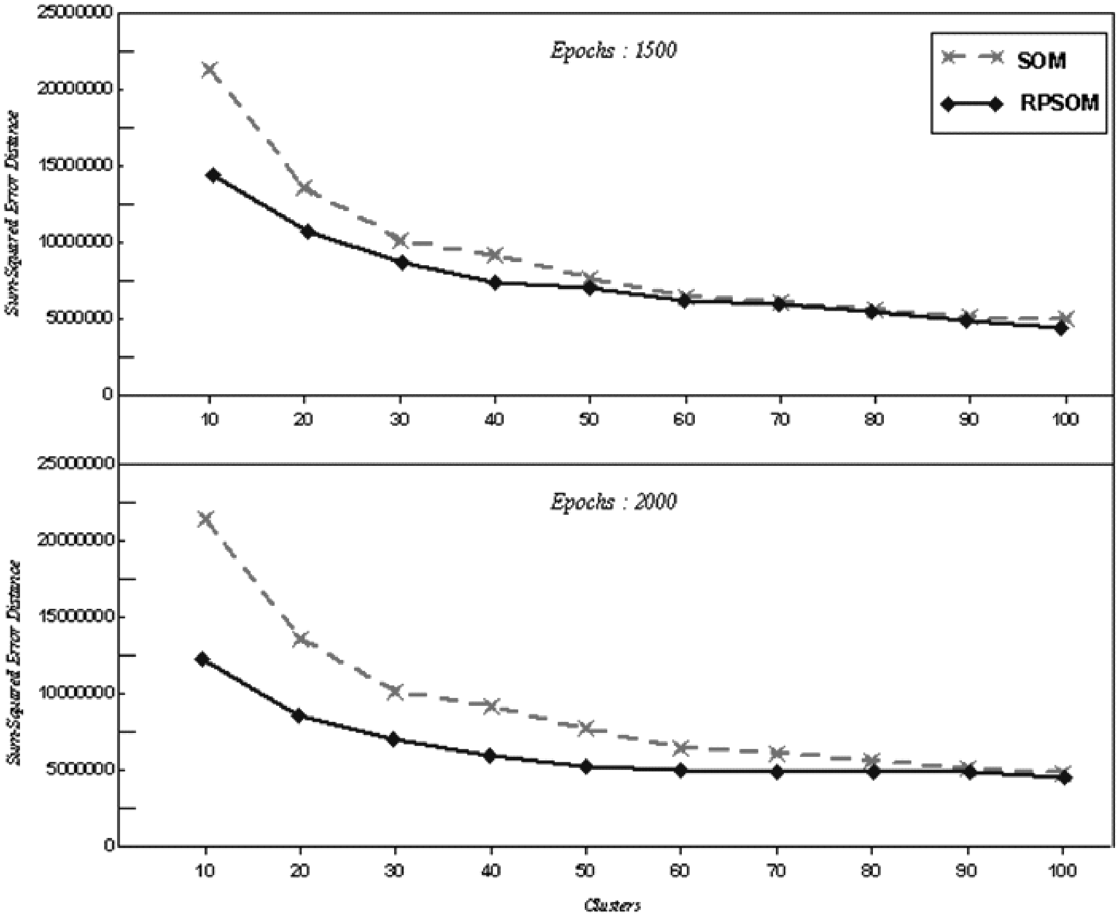

Fig. 7 Sum of squared-error distance comparison of SOM and RPSOM running on 20000 input data in 1500 and 2000 epochs.

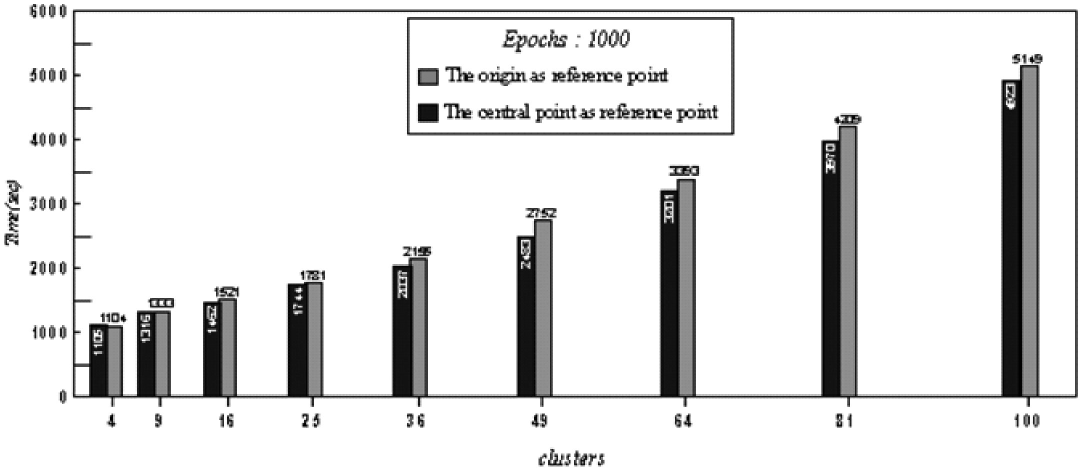

Fig. 8 The execution time of taking different reference point in 1000 epochs.

point and origin as the reference point within 1000 epochs. The results shown in Fig. 9 are that the error distances are almost the same.

In the next experiment, we focus on the accuracy of finding winning neurons and the quantities of reduced computation with different T_1 filter criteria. From the results as shown in Table 4 we found that the larger the filter is, the lower the ratio of computation reduced. For example, consider $T_1 = 6$, which approximates $\mu + \sigma = 6.24$, the ratio of reduced number of comparisons is 28.84% and the radio of matched winner neurons reaches as high as 99.94%. This experiment concludes that the central point as reference

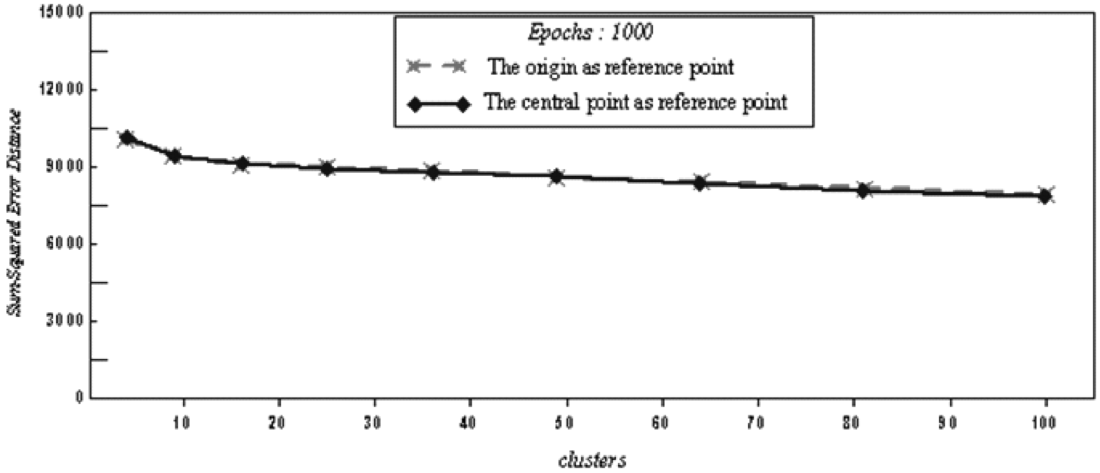

Fig. 9 The error distance of taking different reference point in 1000 epochs.

Table 4 The impact of different T_1 thresholds for clustering Yeast database ($\mu = 4.88$, $\sigma = 1.36$) into 64 clusters using center point as the reference point in 1000 epochs.

Threshold (T_1)	2	3	4	5	6	7	8	9	10
Number of input data entered (2467 × 1000)	2467000	2467000	2467000	2467000	2467000	2467000	2467000	2467000	2467000
Number of match winner neurons in traditional SOM and RPSOM	1864455	2351309	2440988	2459997	2465502	2466983	2466999	2466999	2467000
Number of mismatch winner neurons in traditional SOM and RPSOM	386456	80138	26012	7003	1498	17	1	1	0
Number of input without any neuron inside the range bounded by T_1	216089	35553	0	0	0	0	0	0	0
Number of comparisons for finding winner neurons by the traditional SOM (2467 × 1000 × 64)	157888000	157888000	157888000	157888000	157888000	157888000	157888000	157888000	157888000
Number of comparisons for finding winner neurons by RPSOM	100540498	120102910	96013850	97741474	112351520	130622061	140992545	147951657	153020861
Ratio of reduced number of comparisons	36.32%	23.93%	39.19%	38.09%	28.84%	17.27%	10.70%	6.29%	3.08%
Ratio of matched winner neurons	75.57580%	95.31046%	98.94560%	99.71613%	99.93928%	99.99931%	99.99996%	99.99996%	100%

point can speed up computation and avoid large threshold. It is a good choice to select the reference point to be the central point when the data is situated far from the origin.

5. Conclusions

In this paper, we propose a new SOM algorithm named Reference Point SOM (RPSOM) to speed up the execution time. It can effectively reduce the time for doing clustering in a large database. It can reduce time required by 20% when executing 100 neurons clustering in 1000 epochs for the Yeast database.

In summary, the method described in this paper offers three advantages. First, the initialization scheme of RPSOM is more efficient than Su's scheme. Our method reduces the time complexity from $O(n^2)$ to $O(n)$ in the steps for locating the initial neurons. Second, a reference point based method is proposed when SOM algorithm is looking for the winning neuron and thereafter the neighboring neurons. It can reduce the computation time for finding the winning and neighboring neurons.

Finally, it can calculate the distances from all the input points and neurons to the reference point in advance. In this way, it can filter out those impossible winning neurons and neighboring neurons. In addition, it can control the filtered area by two thresholds. We also show that choosing the central point of input data as the reference point is better than choosing the origin. From the experiments, we find that the performance with regard to improved execution time of the RPSOM is much better than traditional methods.

References

[1] H. Bauer and T. Villmann, "Growing a hypercubical output space in a self-organizing feature map," IEEE Trans. Neural Netw., vol.8, no.2, pp.218–226, 1997.
[2] B. Fritzke, "Let it grow-self-organizing feature maps with problem dependent cell structure," Proc. International Conference on Artificial Neural Networks, pp.403–408, 1991.
[3] B. Fritzke, "Growing cell structures-a self-organizing network for unsupervised and supervised learning," Neural Netw., vol.7, no.9, pp.1441–1460, 1994.
[4] Y.P. Jun, H. Yoon, and J.W. Cho, "L learning: A fast self-organizing feature map learning algorithm based on incremental ordering," IEICE Trans. Inf. & Syst., vol.E76-D, no.6, pp.698–706, June 1993.
[5] J.A. Kangas, T.K. Kohonen, and J.T. Laaksonen, "Variants of self-organizing maps," IEEE Trans. Neural Netw., vol.1, no.1, pp.93–99, 1990.
[6] M.Y. Kiang, U.R. Kulkarni, M. Goul, A. Philippakis, R.T. Chi, and E. Turban, "Improving the effectiveness of self-organizing map networksusing a circular Kohonen layer," Proc. 30th Hawaii International conference on System Sciences, pp.521–529, 1997.
[7] T.K. Kohonen, Self Organization and Associative Memory, 2nd ed., Springer, New York, 1995.
[8] T.K. Kohonen, Self-Organizing, 3rd ed., Springer, Berlin, 2001.
[9] K. Koike, S. Kato, and T. Horiuchi, "A two-stage self-organizing map algorithm with threshold operation for data classification," Proc. Society of In-strument and Control Engineers, pp.3097–3099, Osaka, 2002.
[10] P. Koikkalaninen and E. Oja, "Self-organizing hierarchical feature maps," Proc. International Joint Conference on Neural Network (IJCNN), pp.279–284, 1990.
[11] H. Kusumoto and Y. Takefuji, "$O(log_2 M)$ self-organizing map algorithm without learning of neighborhood vectors," IEEE Trans. Neural Netw., vol.17, no.6, pp.1656–1661, 2006.
[12] Z.Y. Liu and L. Xu, "Topological local principal component analysis," Neurocomputing, vol.55, pp.739–745, 2003.
[13] Z.P. Lo and B. Bavarian, "On the rate of convergence in topology preserving neural networks," Biological Cybernetics, vol.65, no.1, pp.55–63, 1991.
[14] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data," IEEE Trans. Neural Netw., vol.13, no.6, pp.1331–1341, 2002.
[15] M.C. Su and H.T. Chang, "Fast self-organizing feature map algorithm," IEEE Trans. Neural Netw., vol.11, no.3, pp.721–733, 2000.
[16] M.C. Su and H.T. Chang, "A new model of self-organizing neural networks and its application in data projection," IEEE Trans. Neural Netw., vol.12, no.1, pp.153–158, 2001.
[17] M.C. Su, T.K. Liu, and H.T. Chang, "An efficient initialization scheme for the self-organizing feature map algorithm," Proc. International Joint Conference on Neural Networks, pp.1906–1910, Washington, 1999.
[18] K.K. Truong, "Multilayer Kohonen image codebooks with a logarithmic search com-plexity," Proc. IEEE International Conference Acoust., Speech, and Signal, pp.2789–2792, Canada, 1991.
[19] I. Valova, D. Szer, N. Gueorguieva, and A. Buer, "A parallel growing architecture for self-organizing maps with unsupervised learning," Neurocomputing, vol.68, pp.177–195, 2005.
[20] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," IEEE Trans. Neural Netw., vol.11, no.3, pp.586–600, 2000.
[21] L. Wetmore, M.I. Heywood, and A.N. Zincir-Heywood, "Speeding up the self-organizing feature map using dynamic subset selection," Neural Processing Letters, vol.22, no.1, pp.17–32, 2005.
[22] P. Xu and C.H. Chang, "Self-organizing topological tree," Proc. IEEE International Symposium on Circuits and Systems (ISCAS), pp.732–735, 2004.

Shu-Ling Shieh received the MS degree in information management from National Yunlin University of Science & Technology, Taiwan, in 1996. Presently she is pursuing Ph.D. at the Department of Computer Science of National Chung Hsing University, Taiwan. She is currently a lecturer in the Department of Information Management of the Ling-Tung University, Taiwan. Her research interests are in data mining, neural network, self-organizing map, and information visualization.

I-En Liao received the BS degree in Applied Mathematics from National Cheng-Chi University, Taiwan, in 1978, and both the MS degree in Mathematics and the PhD degree in Computer and Information Science from the Ohio State University in 1983 and 1990, respectively. He is currently a professor in the Department of Computer Science of National Chung Hsing University, Taiwan. His research interests are in data mining, XML database, wireless networks, and bioinformatics. He is a member of the ACM and the IEEE Computer Society.

Kuo-Feng Hwang received the BS degree in Information Science from Tung Hai University, Taiwan, in 1994, and both the MS degree in Mathematics and the PhD degree in Applied Mathematics from National Chung Hsing University, Taiwan, in 1996 and 2006, respectively. He is currently an assistant professor in the Department of Information Management of the Overseas Chinese Institute of Technology, Taiwan. His research interests are in data mining, wireless networks, and bioinformatics.

Heng-Yu Chen received the BS degree in Information Management from National Yunlin University of Science & Technology, Taiwan, in 2003, and the MS degree in Computer Science from the National Chung Hsing University in 2006. He is currently an engineer in the Department of Research & Design of APEX ROBOT SYSTEMS, INC., Taiwan. His research interests are in data mining, neural networks, and self-organizing map.