Genetic-Algorithms-Based Approach to Self-Organizing Feature Map and its Application in Cluster Analysis

Mu-Chun Su and Hsiao-Te Chang
Department of Electrical Engineering, Tamkang University, Taiwan, R.O.C.
E-mail address : muchun@ee.tku.edu.tw

Abstract

In the traditional form of the self-organizing feature map (SOFM) algorithm, the criterion for stopping training is either to terminate the training procedure when no noticeable changes in the feature map are observed or to stop training when the number of iterations reaches a prespecific number. Unfortunately, there is no guarantee that the final map will be the most successful (i.e. topologically ordered) map of the whole maps formed during the training procedure. In this paper we propose an efficient method for measuring the degree of topology preservation. Based on the method we apply genetic algorithms (GAs) in two stages to form a topologically ordered feature map. We then use a special method to interprete an SOFM formed by the proposed genetic-algorithm-based method to estimate the number and the locations of clusters from a multidimensional data set without labeling information. Two data sets are used to illustrate the performance of the proposed methods.

1. Introduction

Cluster analysis is one of the basic tools for exploring the underlying structure of a given data set and is being applied in a wide variety of engineering and scientific disciplines. Cluster seeking is very experiment-oriented in the sense that clustering algorithms that can deal with all situations are not yet available. Extensive and good overview of clustering algorithms can be found in [1]-[2]. The performance of most clustering algorithms is greatly influenced by the number of clusters which can not always be defined *a priori*, the choice of initial cluster centroids, and the geometrical properties (e.g. shapes and distributions) of the data. Generally, there are two approaches to specifying the number of clusters. The first approach involves increasing the number of clusters, computing some certain performance measures in each run, until partition into optimal number of clusters is obtained [3]-[4]. This approach requires extensive computation. The second approach focuses on finding a good projection algorithm which maps a set of multi-dimensional patterns onto a two-dimensional space so as to allow one to cluster data directly by eyes [5]-[7]. The price paid for the possibility of visual examination of clusters is that we can not automate the specification of clusters. Besides, projection algorithms are usually expensive to use.

Lately neural networks, for example, competitive learning networks [8], adaptive resonance theory (ART) networks [9]-[10], and self-organizing feature map (SOFM) networks [11] also often have been used to cluster data. One problem associated with competitive learning networks is the number of neurons should be prespecified by the user, however, if the prespecified number of neurons does not match the real number of clusters of a data set the clustering results will be unacceptable. One way of preventing this is to have a finite (or infinite) supply of neurons, but not use them until needed. The ART-based networks behavior in just this way. However, the performance of the networks is highly dependent on a "*vigilance parameter*" which is prespecified by the user. If the value of the parameter is small the similarity condition is easier to meet, resulting in a coarse categorization. It means the number of the resulting clusters is small and the sizes of the clusters are large. On the other hand, if the value of the parameter is chosen to be close to 1, many finely divided clusters are formed. As for the SOFM network, the principal goal of the SOFM algorithm developed by Kohonen [11] is to convert patterns of arbitrary dimensionality into the response of one- or two-dimensional arrays of neurons, and to perform this transformation adaptively in a topological ordered fashion. The transformation makes topological neighborhood relationship geometrically explicit in low-dimensional feature map, therefore, an SOFM usually acts as a preprocessor to either extract features from a given data set or allow one to visually metric-topological relationships of input patterns. It should be emphasized that the interpretation of a formed feature map is not as straightforward as it appears to be. In fact, a trained map has to be calibrated by supervised labeling of array neurons in response to a specific known vector from the training set. Such labeling is usually achieved by the so-called "*voting method*" (i.e. a neuron is labeled class k if it

0-7803-4859-1/98 $10.00©1998 IEEE

responds to input patterns belonging to class k as a majority within the whole data set). After this we are then able to analyze the topological meaning of the labeled map. As we can see, if category information is not available, the inspection of the map does not reveal any information about the clustering characteristics of the data. In fact, the absence of category labels distinguishes cluster analysis from pattern recognition (and discriminate analysis), therefore, an unlabeled SOFM is not of much help in cluster analysis.

In [12], we have proposed a method of interpreting an unlabeled SOFM so as to provide us with clustering information. However, the performance of the method highly depends on whether the topological neighborhood relationship of the data set can be preserved in a SOFM. The success of feature map formation is critically dependent on how the main parameters of the algorithm, namely, the learning rate and the neighborhood function are selected. Unfortunately, they are usually determined by a process of trial and error. Moreover, the criterion for stopping training is usually either to terminate the training procedure when no noticeable changes in the feature map are observed or to stop training when the number of iterations reaches a prespecified number. As a result, there is no guarantee that the final map will be the most successful one. This motivated us to apply genetic algorithms in a two-stage procedure to search a topologically ordered feature map. In the first stage, we try to select N^2 (the size of the network to be trained) samples to most represent the population of the data set which contains M samples ($M \geq N^2$). We then enter the second stage to arrange the N^2 selected samples into the N^2 neural array in a topologically ordered fashion. Of course, we have to define a fitness function which reflects the degree of topology preservation. Various qualitative and quantitative methods for characterizing the degree of topology preservation have been proposed [13]-[14]. Each has its advantages and disadvantages. In this paper, we give a new and efficient measure for quantifying topology preservation. This new measure is especially applicable to the method of interpreting an unlabeled SOFMs. A vanishing value of the proposed measure indicates a perfect neighborhood preservation; positive values indicate violations of neighborhood relations.

The remaining of the paper is organized as follows. The next section briefly introduces the characteristics of SOFMs and the method of extracting clustering information from a trained SOFM. We then present the method of applying GAs to form a topologically ordered SOFM in Section 3. Simulation results of two data sets are provided in Section 4. Finally, a few concluding remarks are given to conclude this paper.

2. Interpretation of a Trained SOFM

The principal goal of self-organizing feature maps is to transform patterns of arbitrary dimensionality into the responses of one- or two-dimensional arrays of neurons, and to perform this transform adaptively in a topological ordered fashion. The transformation makes topological neighborhood relationship geometrically explicit in low-dimensional feature maps. The essential constituents of SOFMs are as follows [11] :

- an array of neurons that compute simple output functions of incoming inputs of arbitrary dimensionality,
- a mechanism for selecting the neuron with the largest output,
- an adaptive mechanism that updates the weights of the selected neuron and its neighbors.

The training algorithm proposed by Kohonen for forming an SOFM is summarized as follows :

Step 1: Initialization: Choose random values for the initial weights $\underline{w}_j(0)$.

Step 2: Winner Finding: Find the winning neuron j^* at time k, using the minimum-distance Euclidean criterion:

$$j^* = \arg\min_j \left\| \underline{x}(k) - \underline{w}_j \right\|, \qquad j = 1, \cdots, N^2. \tag{1}$$

where $\underline{x}(k) = [x_1(k), \cdots, x_n(k)]^T$ represents the k^{th} input pattern, and $\|\cdot\|$ indicates the Euclidean norm.

Step 3: Updating: Adjust the weights of the winner and its neighbors, using the following rule:

$$\underline{w}_j(k+1) = \begin{cases} \underline{w}_j(k) + \eta(k)(\underline{x}(k) - \underline{w}_j(k)) & if\ j \in N_{j^*}(k) \\ \underline{w}_j(k) & o.w. \end{cases} \tag{2}$$

where $\eta(k)$ is a positive constant and $N_{j^*}(k)$ is the topological neighborhood set of the winner neuron j^* at time k. It should be emphasized that the success of the map formation is critically dependent on how the main parameters (i.e. $\eta(k)$ and $N_{j^*}(k)$) are selected, initial values of weight vectors, and the number of iterations.

As mentioned, if no category information is available, the inspection of the map does not reveal any information about the clustering characteristics of the data set. From our previous work [12], we proposed a method to interpret an unlabeled SOFM so as to provide us with information of both the number of clusters and the locations of the

cluster centroids. The method is given as follows:

(1) Map forming : The whole training data set is used to form a SOFM.

(2) Response accumulation : The responses of each neuron are accumulated according to the following equations:

$$h_j = \sum_{k=1}^{M} Out_j(\underline{x}_k), \quad j = 1, 2, \cdots, N^2. \tag{3}$$

and

$$Out_j(\underline{x}_k) = \exp\left(-\left\|\underline{x}_k - \underline{w}_j\right\|^2\right), \quad j = 1, 2, \cdots, N^2. \tag{4}$$

where M denotes the total number of training patterns.

(3) Peak searching : We can view the accumulated responses of the neurons as an $N \times N$ digital image. The digital image is a matrix whose row and column indices identify a neuron in the array and the corresponding matrix element value identifying the gray level at that point is proportional to the value of the accumulated response of the corresponding neuron. The image provides a global structure of the given data set. Pixels with relatively brighter gray levels are the potential centroids of clusters. A 3 × 3 mask can be used to automate the specification of such pixels. We scan the image pixel by pixel, from top to bottom and from left to right. Let p_0 denote the pixel at any step in the scanning process and let $p_1, \cdots, p_8$ be the 8-neighbors of p_0. A peak exists at the pixel p_0 if the following condition is satisfied:

$$h_{p_0} > h_{p_i}, \quad i = 1, 2, \cdots, 8. \tag{5}$$

We have to point out that actually we can find out neurons whose accumulated response are greater than their 8-neighbors directly from the accumulated responses without transforming them into a digital image. The reason for doing this transformation is that the transformed image may help us to justify the estimates and even to correct the estimates.

3. GA-Based Algorithm for Self-Organizing Feature Maps

The Kohonen algorithm is a well established learning rule for SOFMs and can be easily implemented. However, the selection of the main parameters and the stopping criterion show a substantial impact on the results produced by the SOFM algorithm. In addition, Ritter and Schulten have shown that on average the Kohonen rule decreases the value of a cost function associated with the rule until we reach a local minimum [15]. These motivated us to propose a method based on GAs to form a feature map. GAs are derivative-free stochastic optimization methods based loosely on the concepts of natural selection and evolutionary processes. GAs are less likely to get trapped in local minima, which inevitably are present in any gradient-based optimization application. In GAs, the search for an optimal solution is achieved through the manipulation of a population of string structures known as chromosomes. Each chromosome is a simple coding of a potential solution to the problem domain and is assigned a "*fitness score*" according to how good the solution to the problem it is. Instead of a possible (trial) solution, GAs usually work on a population of possible solutions which is then evolved repeatedly toward a new population using genetic operators such as reproduction, crossover, and mutation. After a number of generations, the population contains chromosomes with better fitness values; this is analogous to Darwinian models of evolution by random mutation and natural selection.

Our object is to form a topologically ordered feature map, therefore, the fitness function that associates a performance value with each chromosome should reflect the degree of the neighborhood preservation of the resulting feature map. As mentioned, various qualitative and quantitative method for measuring the preservation or violation of neighborhood relations have been proposed [13]-[14]. Each approach has its own considerations. Since our goal is to extract clustering information from an unlabeled feature map, a new topology measure serving for this particular application is proposed here. Let Λ_m denote the set contains the 8-neighbors of neuron m. A feature map is topologically ordered if the following condition is satisfied :

$$\text{IF} \quad \left\|\underline{w}_i - \underline{w}_m\right\| \geq \left\|\underline{w}_{j(m)} - \underline{w}_m\right\| \tag{6}$$

$$\text{for} \quad 1 \leq m \leq N^2, \quad 1 \leq i \leq N^2, \quad i \neq m \quad \text{and} \quad i \notin \Lambda_m$$

where $j(m) = \arg\min_{k \in \Lambda_m} \sum_{l=1}^{2} |r_{kl} - r_{il}|$, $\underline{r}_k = [r_{k_1}, r_{k_2}]^T$ is the plane position vector of neuron k, and $\sum_{l=1}^{2} |r_{kl} - r_{il}|$ is the planar distance between neuron k and neuron i. We now define V as the measure for measuring the degree of the violation of neighborhood relations. The definition is given as follows :

$$V = \sum_{m=1}^{N^2} \sum_{i \in S_m} \left[1 - \exp\left(-\left\|\underline{r}_i - \underline{r}_m\right\|^2\right)\right] \frac{\left\|\underline{w}_{j(m)} - \underline{w}_m\right\| - \left\|\underline{w}_i - \underline{w}_m\right\|}{\left\|\underline{w}_{j(m)} - \underline{w}_m\right\|} \tag{7}$$

where S_m is referred to as the set containing neurons which violate the aforementioned condition with respect to

neuron m. The term, $1-\exp(-\|\underline{r}_i - \underline{r}_m\|^2)$, is a scaling factor which increases as the planar distance between neuron i and neuron m decreases. While $V = 0$ indicates a perfect neighborhood preservation, positive values of V indicate violations of neighborhood relations. We use an example shown in Fig. 1 to illustrate the effectiveness of the measure. They are two maps of a square input space onto a 5×5 neural array. Clearly, the map with $V = 0$, shown in Fig. 1(a), is more topologically ordered than the map (with $V > 0$), shown in Fig. 1(b).

Instead of directly applying GAs to search a topologically ordered feature map, we solve the problem in two stages. The reason is very obvious and straightforward. If we directly encode a possible solution into a string and then expect GAs to find out an acceptable solution for us, the evolutionary process will take a very long time to converge because the solution space is too huge. In the first stage, we try to select N^2 data points out of the training set containing M data points so that these N^2 data points can represent the whole data set in the sense that the covariance matrices of these two sets (the selected N^2 data points and the original M data points) are as similar as possible. We encode a possible solution into a M-bit binary string containing only N^2 1's and $(M-N)$ 0's. Each bit with value 1 represents the corresponding data point should be selected and vice versa. It should be emphasized that the traditional one-point or two-point crossover operators and mutation operator can not be applied directly in the evolutionary process because the constraint that the resultant chromosomes should contain exactly N^2 1's. In this paper we employ a crossover operator similar to the partially matched crossover (PMX) [16] to replace the traditional crossover operators. As for the mutation operator, if the number of 1's in the mutated chromosome is not equal to N^2, we have to randomly alter another bit until the number of 1's is equal to N^2. After we have selected N^2 data points, we enter the second stage where we try to arrange these N^2 data points into an $N \times N$ neural array so as to form a topologically ordered map. At this time, the genetic representations of trial solutions are in the form of integer numbers. We use the topology measure V as the fitness function for the problem. In our simulations, we have examined two different representation. The first option is illustrated by Fig. 2(a). In this representation, the neurons are indexed from row to row in the order of 1 to N^2. The sequence means that the second data point is assigned to be the synaptic weight vector of the neuron on the top left corner of the $N \times N$ neural array and so on. The second option is illustrated by Fig. 2(b). Obviously, it tell us that the fifth data point is assigned to be the synaptic weight vector of the neuron on the right bottom corner of the neural array. Each representation has its advantages and disadvantages. The good news concerning the first representation is that we can directly employ a simple crossover operation similar to the PMX to undergo the crossover process. The bad news is that it does not preserve the 2-D neighbooring information since it uses a one-dimensional representation. As a result, we usually can not find an acceptable solution in just a few generations. The second representation maintains the 2-D neighbooring information but simple crossover operator is not applicable. Therefore, we designed a special 2-D crossover operator for the second representation. The details of the crossover procedure can not be given here because of the space limitation. The basic idea was inspired by Grefenstette *et al.* [17]. The crossover operator constructs only one offspring from two parents. For each neuron on the inner $(N\text{-}2) \times (N\text{-}2)$ neural arrays of the parents, we compute the inner summation of Eq. (7). The computed values of the $2 \times (N\text{-}2) \times (N\text{-}2)$ neurons are sorted. In the increasing order, we then copy the whole 3×3 block containing the neuron with the smallest computed value and its 8-neighbors to the offspring in the way from top to bottom and from left to right. Note that the condition that each integer (e.g. from 1 to N^2) can only appear once must be hold. If the condition is violated we just randomly choose an integer from the set containing integers which have not yet been used to replace the integer. Finally, if the measure V for the resultant map is not equal to zero, we may then use the simple competitive learning algorithm to fine tune the map. This fine-tuning procedure usually converges very fast since we start from a very good initial map.

4. Computer Simulations

Two data sets are used to illustrate the effectiveness of the proposed methods. The artificial data set contains 579 2-dimensional data points. We chose this 2-D data set as the first example for two reasons. First the visual inspection can help us to qualitatively check the degree of preservation of neighborhood relations; second we can easily see the peaks of the transformed 2-D digital image correspond to the centers of the real clusters of the data set. The second data set is the well-known Iris data set which contains 150 4-dimensional data points. Since the data points are 4-dimensional we can not cluster the data directly by eye, therefore, it can be used to illustrate the effectiveness of our method.

Example 1 : Artificial Data Set

In order to test the performance of the proposed

methods, we generated a mixture of spherical and ellipsoidal clusters as shown in Fig. 3. In our simulations, a 5×5 network was trained to form a topological feature map. In each stage, the population size is 1000 and the evolutionary generations are 100. The value of the measure V decrease from 59.98 to 9.88. Table 1 tabulates the accumulated responses of the neural array. The transformed 2-D image is shown in Fig. 4. For a more distinct illustration, we flip the gray levels such that darker regions represent peaks instead of valleys and the gray levels are quantized to be 3 levels. We can easily find there are three peaks (i.e. three clusters) either by eye or by using the 3×3 mask to scan the image.

Example 2 : Iris Data Set

The Iris data set has three subsets (i.e. Iris Setosa, Iris Versicolor, and Iris Virginica), two of which are overlapping. The Iris data set are in a 4-dimensional space and each subset contains 50 patterns. The size of the network used in the simulations is still 5×5. The value of the measure V decrease from 69.75 to 7.63. Table 2 tabulates the accumulated responses of the neural array. The transformed 2-D digital image is given in Fig. 6. Obviously, there are three peaks in the image, therefore, there exist three clusters in the Iris data set.

5. Conclusions

In this paper we propose a method of applying GAs to form an SOFM and then use a special method to interpret an unlabeled map so as to provide estimates of the number and the locations of clusters from a data set. In order to accelerate the searching procedure, we adopt a two-stage training procedure. If the value of the measure V for the resultant map found by the GAs is not very small or equal to zero, we suggest to use the simple competitive learning algorithm to fine tune the map. The simulation results demonstrate the effectiveness of the proposed methods.

Acknowledgments

This work is supported by the National Science Council, Taiwan, R.O.C., under the Grant NSC87-2213-E-032-006

References

[1] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentic Hall, New Jersey, 1988.

[2] R. O. Duda and P. E. , *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.

[3] J.-C. Lin, "Multi-class clustering by analytical two-class formulas," *Pattern Recognition and Artifical Intelligence*, vol. 10, no.4, pp. 307-323, 1996.

[4] R. P. Li and M. Mukaidono, "A new approach to rule learning based on fusion of fuzzy logic and neural networks," *IEICE Trans. Inf. & Syst.*, vol. E78-D, No. 11, November, 1995.

[5] C. L. Chang and R. C. T. Lee, "A heuristic relaxation method for nonlinear mapping in cluster analysis," *IEEE Trans. on System, Man, and Cybernetics*, vol. 3, pp. 197-200, 1973.

[6] C. E. Pykett, "Improving the efficiency of Sammon's nonlinear mapping by using clustering archetypes," *Electronics Letters* 14, 799-800, 1978.

[7] R. C. T. Lee, J. R. Slagle, and H. Blum, "A triangulation method for the sequential mapping of points from N-space to two-space," *IEEE Trans. on Computers*, vol. 26, pp. 288-292, 1977.

[8] T. Kohonen, "The 'Neural' Phonetic Typewritter," *IEEE Computer*, vol. 27, no. 3, pp. 11-12, 1988.

[9] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Proc*, vol. 37, pp. 54-115, 1987.

[10] G. A. Carpenter and S. Grossberg, "ART2: self-organization of stable category recognition codes for analog input patterns," *Appl. Optics*, vol. 26, no. 23, pp. 4919-4930, Dec. 1987.

[11] T. Kohonen, *Self-Organization and Associative Memory*, 3*rd* ed. New York, Berlin: Springer-Verlag, 1989.

[12] M. C. Su, N. DeClaris, and T. K. Liu, "Application of neural networks in cluster analysis," IEEE Int. Conf. on Systems, Man, and Cybernetics, pp.1-6, Orlando, Oct. 12-15, 1997.

[13] H.-U. Bauer and K. Pawelzik, "Quantifying the neighborhood preservation of self-organizing feature maps," *IEEE Trans. Neural Networks*, vol. 3, pp. 570-579, 1992.

[14] T. Villmann, R. Der, M. Herrmann, and T. M. Martinetz, "Topology preservation in self-organizing feature maps : exact definition and measurement," IEEE Trans. on Neural Networks, vol. 8, no. 2, March, 1997.

[15] H. Ritter and K. Schulter, "Kohonen's self-organizing maps : exploring their computational capabilities," in IEEE Int. Conf. on Neural Networks, vol. 1, pp. 109-116, San Diego, 1988.

[16] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," *Proc. 1st ICGA*, pp. 154-158, 1985.

[17] J. J. Grefenstette, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Hillsdale, NJ : Lawrence Erlbaum Associates, 1985.

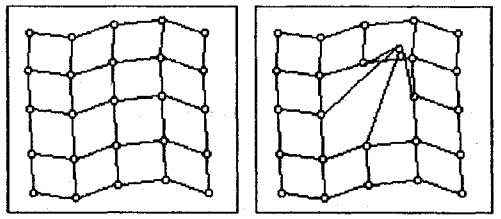
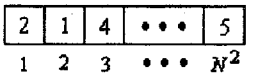


Figure 1. Illustrateion of neighborhood relations. (a) perfect neighborhood preservation; (b) violation of neighborhood preservation.



(a)

2	1	4	24	16
18	15	7	23	21
9	11	3	12	8
10	17	25	14	19
13	6	20	22	5

(b)

Figure 2. Two representations for the second stage of the crossover mechansim. (a) one-dimensional case; (b) two-dimension case.

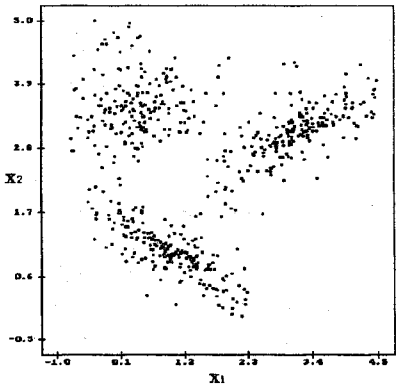


Figure 3. The artificial data set contains 579 two-dimensional data points.

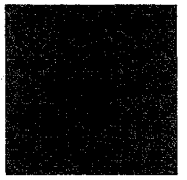


Figure 4. The transformed 2-D digital image of the artificial data set.

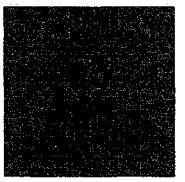


Figure 5. The transformed 2-D digital image of the Iris data set.

Table 1. The accumulated responses of the artificial data set in the 5 × 5 neural array.

59.75690	115.78004	46.53558	117.27979	72.08619
110.04392	114.07597	101.57908	108.90295	108.19028
75.19670	114.35085	90.68682	118.59007	76.39701
59.74305	89.34039	131.09433	120.30192	106.62796
129.01447	131.20070	130.45264	130.52867	95.60133

Table 2. The accumulated responses of the Iris data set in the 5 × 5 neural array.

1.13551	1.00019	1.60847	1.00686	1.01832
1.27416	1.47069	1.00000	1.32737	1.00000
1.01443	1.00003	1.00091	1.00005	1.00000
1.02544	1.13882	1.00674	1.04979	1.00091
1.00248	2.00098	1.00248	1.00684	1.00000