Radius Parallel Self-Organizing Map (RPSOM)*

Masato MASUDA**, Yasushi NAKABAYASHI*** and Genki YAGAWA**

** School of engineering, Toyo University

2100 Kujirai, Kawagoe, Saitama 350-8585, Japan

E-mail: masuda@cse.eng.toyo.ac.jp

*** Faculty of Information Sciences and Arts, Toyo University

2100 Kujirai, Kawagoe-shi, Saitama 350-8585, Japan

Abstract

Self-Organizing Map (SOM) proposed by Kohonen is a kind of the Neural Networks and using for the competitive learning. It is difficult to specify the learning parameters for SOM, because the neighborhood radius, which is a one of learning parameter, depends on the scale of the problem or map space, and the users of SOM code usually find the proper value through a trial and error process. In this circumstance, the authors propose a SOM that has several neighborhood radii, which is similar to the temperature parallel Simulated Annealing (SA) method. In this method, several neighborhood radii conduct competitive learning in parallel, and a highly evaluated neighborhood radius is dynamically selected. Furthermore, the authors propose the new evaluation function of the map, in order to evaluate the proposed method quantitatively. The results of the map evaluation with the proposed evaluation function were consistent with those of visual evaluation. The results are compared with those of the conventional SOM, to demonstrate the effectiveness of the proposed method.

Key words : Self-Organizing Map, Neighborhood Radius, Temperature Parallel SA, Parallelization

1. Introduction

Self-Organizing Map (SOM) is one of the neural networks proposed by T. Kohonen⁽¹⁾⁻⁽⁴⁾. This is a two-layered unsupervised competitive learning model that does not have a hidden layer. It is possible to visualize the model by mapping high-dimensional input vectors on a two-dimensional map space. The SOM is characterized by clustering and data mining, etc. through competitive learning. Competitive learning is the base for unsupervised learning. When data is provided, the activity values of map units are calculated from the weights among input vectors, input units, and map units. In competitive learning, inter-unit weight is changed so that activity value increases when it is high and decreases when it is low. Learning only the maximum activity value is called winner-take-all. This is simple and easy to understand, but there is a risk that most map units will turn out to be useless. There is also neighborhood learning, in which the winner learns pulling surrounding units. At this time, neighboring units correct weights according to the distance from the winner. Since not only the winner but also surrounding units conduct learning, useless units hardly appear and the interpolation between input vectors is good. In neighborhood learning, it is important to specify learning parameters, which are neighborhood radius and learning coefficient. The setting of learning parameters depends on problems and map size. Therefore, it is very difficult to set these parameters.

In this study, map optimization is carried out, by dynamically changing the learning parameters, such as the neighborhood radius and the learning coefficient. In order to dynamically change learning parameters, the authors formulate a function for evaluating a map, and demonstrate its appropriateness. In addition, a map optimization method is proposed, and its effectiveness is demonstrated.

*Received 26 Oct., 2011 (No. 11-0642)
[DOI: 10.1299/jcst.6.16]

2. Self-Organizing Map

SOM, which is one of neural networks and was proposed by T. Kohonen, is a two-layered unsupervised competitive learning model that does not have a hidden layer. In the learning algorithm of SOM, the characteristics of input data are learned through neighborhood learning. A map is formed so that similar kinds of data are located in the neighborhood and other data are located at distant places. Accordingly, it is possible to visualize high-dimensional vector data, and understand the relations among data intuitively.

2.1. Algorithm of SOM

SOM has a map with the $M \times N$ grid, and weighted units exist (Fig.1).

Fig. 1 Conceptual Diagram of SOM (U is map unit, m and n are unit number, w is uniting weight, X is input signal)

Unit weights are updated in neighborhood learning. The algorithm of SOM is as follows:
 Step 1. Initialize the weight vector w of an unit.
 Step 2. Input the vector $\mathbf{x}_{class} = (x_{class}^1, x_{class}^2, x_{class}^3, \dots, x_{class}^I)$ into input layer. Here, “class” represents the number of input vectors, and “I” denotes the dimension of the input vector.
 Step 3. Calculate the Euclidean distance “Dist” between an input vector and the weight vector of a unit in the map layer (competitive layer).

$$Dist_m = \sqrt{\sum_i (x_{class}^i - w_m^i)^2} \quad (1)$$

Step 4. Define the unit that minimizes the Euclidean distance obtained at Step 3 as the winning unit BMU (Best Matching Unit).

$$BMU = \arg \min_m Dist_m \quad (2)$$

Step 5. Update the weights of the winning unit and surrounding units only.

$$\Delta w_m^i = h(l(BMU, (m))) (x_{class}^i - w_m^i) \quad (3)$$

$$h(l) = \alpha \exp\left(-\frac{l^2}{2\sigma^2}\right) \quad (4)$$

Step 6. Repeat Steps 2 to 5 sufficiently.

Here, l represents the distance between the winning unit and the unit whose weight is updated. h and σ denote the neighborhood function and the range of the influence of the neighborhood function (neighborhood radius), respectively. This is decreased according to learning steps. α is learning coefficient, ranging from 0 to 1. The learning coefficient α decreases as learning steps proceed.

2.2. Evaluation of SOM

Generally, SOM is evaluated visually, because high-dimensional data are expressed as low-dimensional ones and so they can be seen intuitively. Namely, qualitative evaluation is common. Another method is to evaluate the performance of cluster classification. Cluster boundaries are selected and evaluated, based on the amount of data selected as the winners of map units, the similarity of code vectors between adjacent units, and the histogram using cluster density⁽⁵⁾⁻⁽⁷⁾.

However, in the evaluation of cluster classification, it is impossible to evaluate the performance of interpolation between units of SOM. In addition, this evaluation method expresses cluster boundaries with the post-learning map, and so it cannot be used dynamically. In this circumstance, the authors propose a method for evaluating a map. With this method, it is possible to evaluate SOM, which has been evaluated qualitatively or statically, in a quantitative or dynamic manner. In the proposed method, map evaluation is expressed by evaluation functions. The evaluation functions calculate the difference between adjacent units and the difference between an input vector and BMU (Eqs.5,6). The former represents the variation of adjacent units, and as this value is smaller, map interpolation is better. The first term in the right-hand side is the sum of vertical variations, while the second term is the sum of horizontal variations. The latter represents the difference between an input vector and the winning unit. As this value is smaller, the compatibility of the input vector with respect to the winning unit is better.

$$Eval_{Int} = \sum_{map-h}^{(M-1) \times N} \sum_1^I (w_{map-h}^i - w_{map-h+1}^i)^2 + \sum_{map-v}^{M \times (N-1)} \sum_1^I (w_{map-v}^i - w_{map-v+1}^i)^2 \quad (5)$$

$$Eval_{Fit} = \sum_{class}^{CLASS} \sum_1^I (w_{BMU}^i - x_{class}^i)^2 \quad (6)$$

Here, "CLASS" represents the total number of input vectors.

In order to examine the effectiveness of this evaluation formula, the clustering of the following input vectors is carried out.

Table 1 Input Vectors of Animal

Name	Small	Middle	Large	Nocturnality	2 Legs	4 Legs	Whiskers	Ungulata	Mane	Feather	Stripe	Hunt	Run	Fly	Swim	Herbivory
Dove	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0.5
Fox	0	1	0	0.5	0	1	1	0	0	0	0	1	0	0	0	0
Hen	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0.5
Lion	0	0	1	0	0	1	1	0	1	0	0	1	1	0	0	0
Goose	1	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0.5
Eagle	0	1	0	0	1	0	0	0	0	1	0	1	0	1	0	0
Dog	0	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0
Wolf	0	1	0	1	0	1	1	0	1	0	0	1	1	0	0	0
Zebra	0	0	1	0	0	1	1	1	1	0	1	0	1	0	0	1
Duck	1	0	0	0	1	0	0	0	0	1	0.3	0	0	1	1	0.5
Cat	1	0	0	0.5	0	1	1	0	0	0	0	1	0	0	0	0
Owl	1	0	0	1	1	0	0	0	0	1	0	1	0	1	0	0
Tiger	0	0	1	0.5	0	1	1	0	0	0	1	1	1	0	0	0
Horse	0	0	1	0	0	1	1	1	1	0	0	0	1	0	0	1
Hawk	1	0	0	0	1	0	0	0	0	1	0	1	0	1	0	0
Cow	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	1

Table 1 expresses 16 kinds of animals with 16-dimensional characteristics. As the setting of learning of SOM, map size was set at 20×20, neighborhood radius was decreased from 5 to 1, and learning coefficient was decreased from 0.5 to 0.1. This learning was repeated 1,000 times. Fig.3 shows the classification of these 16 kinds of animals with SOM. Color thickness indicates the distance from an input vector. As it is nearer, the color whitens, and vice versa.

The thin line denotes the distance between adjacent units. It also whitens, as the unit is nearer. In Fig.3, birds are concentrated in the lower region, and carnivorous animals are concentrated in the lower right region. Herbivorous animals are concentrated in the upper region. The units at the edge of a class have thick colors, and there appeared units distant from this class.

Figs.3 and 4 show the results of learning as neighborhood radius decreased from 2 to 0 and from 8 to 2. In Fig.3, the winner is obviously independent, but differs in weight from the unit of the adjacent class. Therefore, it can be considered that map interpolation is not so high. Namely, neighborhood radius is small, and a cluster was formed with the radius that does not influence other classes. In Fig.4, inter-unit distance is short at the edge of a class and interpolation is good, but the compatibility of the winner is lower than that in Fig.3, and there are two winners in one unit in some cases. Therefore, it can be considered that the compatibility with respect to this map inputting is not so good. Namely, neighborhood radius is large, and learning influences other classes, and so unit weights linger from some aspects, degrading the compatibility with the winning unit.

Fig. 2 Result of SOM at Neighborhood Radius 2 to 0

Fig. 3 Result of SOM at Neighborhood Radius 5 to 1

Fig. 4 Result of SOM at Neighborhood Radius 8 to 2

Table 2 Evaluation Value of 3 Radii

	Case 1 (2 → 0)	Case 2 (5 → 1)	Case 3 (8 → 2)
$Eval_{Int}$	3.82×10^2	2.48×10^2	1.19×10^2
$Eval_{Fit}$	0.00	0.00	1.99×10^{-1}

Table 2 shows the expression of the evaluation of the three maps with Eqs.5 and 6. It is assumed that neighborhood radius is decreased from 2 to 0 in Case 1, from 5 to 1 in Case 2, and from 8 to 2 in Case 3. In Case 1, the evaluation value $Eval_{Fit}$ is as good as 0.00, but $Eval_{Int}$ is the worst among the three cases. In Case 3, $Eval_{Int}$ is very good, but $Eval_{Fit}$ is the largest among the three cases. Namely, this result is consistent with that of visual observation. Therefore, it can be concluded that this evaluation function, which can expect quantitative evaluation without relying on visual observation, is appropriate and effective.

3. Radius Parallel SOM

There have been some researches into methods for solving SOM in parallel. There are two major methods. One is a method of dividing the competitive layer^{(8),(9)}. In this method, PE is allocated to each unit in the competitive layer. However, as a characteristic of the SOM algorithm, the units other than those for learning become idle. Therefore, each PE has inhomogeneous loading at the time of updating. The other is a method of dividing the input layer⁽¹⁰⁾. This was developed in order to solve the problem of inhomogeneous loading due to the division of the competitive layer. PE is allocated to one unit in the input layer, and has the weight for each unit in the competitive layer. No matter which unit in the competitive layer is selected as the winner or neighbor, all PEs make the same processing, and so loading becomes homogeneous. These are used for processing a large-scale map and heavy input vectors rapidly. Differing from these two parallel approaches, the authors conduct a new SOM parallelization.

The proposed method is not aimed at high-speed processing, but seeking optimal neighborhood radius in the duration for solving one map. It is difficult to set the neighborhood radius of the learning parameter of SOM, because it depends on a problem and map size. In the proposed SOM optimization method, learning can be conducted with the best neighborhood radius every time, by changing neighborhood radius dynamically. This method is similar to the temperature parallel SA (Fig.5).

The temperature scheduling of SA is very difficult, and requires empirical rules and pre-

Fig. 5 Temperature Parallel SA

liminary tests. In the temperature parallel SA^{(11)–(15)}, different temperature is set for each of different processors. Then, SA is processed in parallel in each processor. Namely, each processor has a different solution. Between adjacent processors, objective function values are compared at certain intervals. At this time, if the high-temperature solution is better than the low-temperature one, the solutions are exchanged between the processors. If the low-temperature solution is better, solutions are exchanged stochastically. Like this, in the temperature parallel SA, the high-temperature processor conducts search broadly, while the low-temperature one do so concentrically. As the processing progresses, better solutions are selected.

In this method, the very difficult management of neighborhood radius is conducted automatically, like the temperature management of SA. In this method, processors with different neighborhood radii operate in parallel, and the unit weight of the best processor according to Eqs. 5 and 6 is copied to all processors. Let us call this proposed method radius parallel SOM (RPSOM).

3.1. Algorithm of RPSOM

The RPSOM algorithm is as follows:

- Step 1. Initialize each processor, and unify the weight of the initial unit.
- Step 2. Produce a neighborhood function at each processor.
- Step 3. Start learning at each processor.
- Step 4. Calculate evaluation value at each processor.
- Step 5. Copy the weight of the processor whose evaluation value is the smallest to all processors.
- Step 6. Repeat Steps 2 to 5 sufficiently.

Fig.6 shows the flowchart of these steps.

Weight_{p*} in Fig.6 represents the weight of the processor whose evaluation value is the smallest among all processors. In RPSOM, neighborhood radius changes dynamically like the temperature parallel SA (Fig.7). Neighborhood radius is selected so as to decrease evaluation value every time.

Fig. 6 Flowchart of Radius Parallel SOM

Fig. 7 Neighborhood Radius Transition

3.2. Test of RPSOM

A map of animals is produced like Chapter 2. Map size, etc. are the same. Processing is conducted in parallel with 3 neighborhood radii. The 3 neighborhood radii used in Chapter 2 are adopted. Fig.8 shows the graph of neighborhood radius.

In RPSOM, evaluation value is calculated as follows:

$$Evaluation = \overline{Eval_{Int}} + \overline{Eval_{Fit}} \quad (7)$$

$$\overline{Eval_{Int}} = \frac{Eval_{Int}}{(M-1) \times N + M \times (N-1)} \quad (8)$$

$$\overline{Eval_{Fit}} = \frac{Eval_{Fit}}{CLASS} \quad (9)$$

$\overline{Eval_{Int}}$, $\overline{Eval_{Fit}}$ represent the averages of inter-unit difference and inter-class compatibility, respectively. Evaluation value is independent, and so effort per unit is adopted. In RPSOM, neighborhood radius is selected so as to minimize Eq.7. Fig.9 shows the graph of changed neighborhood radius. At the beginning of learning, there are no changes in neighborhood, but later the weight of the processor with large radius is selected to improve interpolation.

Fig.10 shows the map of learning results, and Table 3 shows evaluation value. Fig.10 indicates that each class is independent. In addition, the thin line, which represents the distance between units, whitens around the inter-class boundary. Namely, units at the edge of a

Fig. 8 Three Neighborhood Radii

Fig. 9 Trace of Neighborhood Radius Transition

class are similar to each other. In other words, similarity of input vectors is high, and the interpolation between winning units is high. In Table 3, RPSOM shows the minimum for all of $Eval_{int}$, $Eval_{Fit}$ and $Eval_{val}$. Namely, the merits of each neighborhood radius are reflected in good balance.

Table 3 Evaluation of Single-Processor and RPSOM

	Case 1	Case 2	Case 3	RPSOM
$Eval_{int}$	3.82×10^2	2.48×10^2	1.19×10^2	7.62×10^1
$Eval_{Fit}$	0.00	0.00	1.99×10^{-1}	0.00
$Eval_{val}$	5.03×10^{-1}	3.25×10^{-1}	1.69×10^{-1}	1.00×10^{-1}

Next, the learning of neighborhood radius is processed for 5 cases. The 5 radii include the above three and the radii between them. Learning coefficient and frequency are the same.

The results are shown in Fig.11, and Table 4, and the change in neighborhood is shown in Fig.12. Evaluation value improved when the 5 neighborhood radii were used. Since the number of neighborhood radii increased, it became possible to select a better radius.

By using this method, it is possible to simplify the setting of neighborhood radius. In

Fig. 10 Result of RPSOM

Fig. 11 Result of RPSOM when 5 Neighborhood Functions

addition, this method does not require preliminary tests for seeking optimal neighborhood radius, and neighborhood radius does not have complex function forms. This method is very easy and convenient, because the optimal radius is selected dynamically among given neighborhood radii.

3.3. Test of Iris Data

Next, the learning in RPSOM is evaluated with the data of iris⁽¹⁶⁾. Table 5 shows the data of iris. This data indicates the size and width of the sepals and petals of 3 kinds of iris. The data is for 50 irises for each kind, that is, a total of 150 irises.

The mapping of this iris problem is conducted with 3 neighborhood radii. Its learning conditions are the same as the mapping of animals, and neighborhood radii of Cases 1, 2, and 3 are used. Since the data of as many as 150 irises was used, they were classified with colors. Setosa, Versicolor, and Virginica are represented by red, green, and blue, respectively. Learning results, changes in neighborhood radius, and evaluation value are shown in Fig.13, Fig.14, and Table 6.

Table 4 Evaluation of 5 Neighborhood Functions RPSOM

	3 Neighborhoods	5 Neighborhoods
$Eval_{Int}$	7.62×10^1	7.38×10^1
$Eval_{Fit}$	0.00	0.00
$Evaluation$	1.00×10^{-1}	9.71×10^{-2}

Fig. 12 Trace of Neighborhood Radius Transition when 5 neighborhood functions

Table 5 Data of Iris

Kind	No.	Sepal		Petal	
		Length	Width	Length	Width
Setosa	1	5.1	3.5	1.4	0.2
	⋮	⋮	⋮	⋮	⋮
	50	5.0	3.3	1.4	0.2
Versicolor	1	7.0	3.2	4.7	1.4
	⋮	⋮	⋮	⋮	⋮
	50	5.7	2.8	4.1	1.3
Virginica	1	6.3	3.3	6.0	2.5
	⋮	⋮	⋮	⋮	⋮
	50	5.9	3.0	5.1	1.8

With this classification of iris, good results were obtained. In this iris data, Setosa had smaller sepals and petals than the others, and so class boundaries appeared obviously. Versicolor and Virginica data resemble each other, and so class boundaries are vague in the map. However, Versicolor and Virginica are not mixed in a class in the map.

Therefore, with RPSOM, it is possible to conduct cluster classification and interpolation with relatively similar data.

4. Conclusion

In this study, the authors proposed new evaluation and learning methods for SOM. In the evaluation method, a map is evaluated in parallel with learning. This evaluation method showed the same effects as visual evaluation. The learning method is the SOM with more than one neighborhood radius, which is called RPSOM. By using the above evaluation method, it is possible to dynamically select neighborhood radius. With RPSOM, clustering and the interpolation between winning units were conducted more efficiently than the case where learning is carried out with a single neighborhood radius or learning coefficient. This is because learning

Fig. 13 Iris Data Clustering by RPSOM

Fig. 14 Transition of Neighborhood Radius by RPSOM at Iris Data

Table 6 Evaluation value of Iris Data Map

	Case 1	Case 2	Case 3	RPSOM
$Eval_{Int}$	2.35×10^2	9.46×10^1	5.21×10^1	8.94×10^1
$Eval_{Fit}$	2.46×10^{-1}	4.35	1.10×10^1	1.01
$Evaluation$	3.11×10^{-1}	1.53×10^{-1}	1.42×10^{-1}	1.24×10^{-1}

parameters are dynamically adjusted to satisfy compatibility and interpolation in mapping. Therefore, just by setting some neighborhood radii, efficient learning becomes possible. In addition, it can be well evaluated by seeing the resultant map.

Lastly, it can be concluded that the proposed method is effective, because the setting of neighborhood radius of SOM can be simplified and satisfactory results can be obtained.

References

- (1) T.Kohonen, "Self-Organizing Maps", pringer-Verlag, 1995.
- (2) T.Kohonen, "Self-Organized formation of topologically correct feature maps", Biological Cybernetics, Vol.43, pp.59-69, 1982.
- (3) T.Kohonen, P. Somervuo, "Self-organizing maps of symbol strings", Neurocomputing, 19-30, 1998.
- (4) T.Kohonen, "Things you haven't heard about the self-organizing map", Proc. IEEE Int. Joint Conf. Neural Networks, San Francisco, 1147-1156, 1993.
- (5) X. Zhang and Y.Li, "Self-organizing map as a new method for clustering and data analysis", Proc. Int. joint Conf. on Neural Networks, pp.2448-2451, 1993.
- (6) R.H.Nielsen, "Competitive learning", in Neurocomputing, pp.63-70, Addison Wesley, 1989.
- (7) M.Terashima, F.Shiratani and K.Yamamoto, "Unsupervised Cluster Segmentation Method Using Data Density Histogram on Self-Organizing Feature Map", The transactions of the Institute of Electronics, Information and Communication Engineers, vol.J79-D-2(7), pp.1280-1290, 1996.(in Japanese)
- (8) K.Obermayer, H.Ritter and K.Schulten, "Large-scale simulations of self-organizing neural networks on parallel computers : application to biological modeling", Parallel Computing, Vol.14, pp.381-404, 1990.
- (9) C.H.Wu, R.E.Hodges and C.J.Wang, "Parallelizing the Self-Organizing Feature Map on multiprocessor system", Parallel Computing, Vol.17, pp.821-832, 1991.
- (10) G.Myklebust and J.G.Solheim, "Parallel Self-organizing Maps for actual applications", Proc.Int. Conf. Neural Networks 1995, 1995.
- (11) K.Nitta, "Multiple sequence alignment by parallel simulated annealing", Icot Technical Report, pp.TR-730, 1992.
- (12) Kirkpatrick, S., Gelett Jr. C. D., and Vecchi, M. P., "Optimization by Simulated Annealing", Science, 220(4598), 671-680, 1983.
- (13) K.Konishi, K.Taki and K.Kimura "Temperature Parallel Simulated Annealing Algorithm and Its Evaluation", Transactions in Information processing Society of Japan, 36, 797-807, 1995
- (14) K.Konishi, K.Taki and K.Kimura, "Temperature Parallel Simulated Annealing to the Traveling Salesman Problem and its Efficient Implementation in the Distributed Memory Parallel Machine", Joint Symposium of Parallel Processing , 153-160, 1996.(in Japanese)
- (15) M.Miki, T.Hiroyasu, M.Kasai and M.Ikeuchi, "Temperature Parallel Simulated Annealing with Adaptive Neighborhood for Continuous Optimization Problem", Proceedings of the LASTED International Conference, Parallel and Distributed Computing and Systems, 1999.
- (16) UCI Machine Learning Repository: <http://www.ics.uci.edu/mllearn/MLRepository.html>