Alexandria Engineering Journal (2014) 53, 827–831

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

A novel Self-Organizing Map (SOM) learning algorithm with nearest and farthest neurons

Vikas Chaudhary [a,*], **R.S. Bhatia** [a], **Anil K. Ahlawat** [b]

[a] *National Institute of Technology (N.I.T.), Kurukshetra, Haryana, India*
[b] *Krishna Institute of Engineering & Technology, Ghaziabad, U.P., India*

Received 3 May 2013; revised 11 September 2014; accepted 16 September 2014
Available online 23 October 2014

KEYWORDS

Self-Organizing Map (SOM);
Farthest neuron;
Nearest neuron;
Winning frequency;
Neighborhood neurons

Abstract The Self-Organizing Map (SOM) has applications like dimension reduction, data clustering, image analysis, and many others. In conventional SOM, the weights of the winner and its neighboring neurons are updated regardless of their distance from the input vector. In the proposed SOM, the farthest and nearest neurons from among the 1-neighborhood of the winner neuron, and also the winning frequency of each neuron are found out and taken into account while updating the weight. This new SOM is applied to various input data sets and the learning performance is evaluated using three standard measurements. It is confirmed that modified SOM obtained a far better result and better effective mapping as compared to the conventional SOM, which reflects the input data distribution.

1. Introduction

The Self-Organizing Map (SOM) is an unsupervised learning algorithm introduced by Kohonen [1]. In the area of artificial neural networks, the SOM is an excellent data-exploring tool as well [2]. It can project high-dimensional patterns onto a low-dimensional topology map. The SOM map consists of a one or two dimensional (2-D) grid of nodes. These nodes are also called neurons. Each neuron's weight vector has the same dimension as the input vector. The SOM obtains a statistical feature of the input data and is applied to a wide field of data classification [3–6]. SOM is based on competitive learning. In competitive learning [7], neuron activation is a function of distance between neuron weight and input data. An activated neuron learns the most and its weights are thus modified. If a similar pattern is found again, then the same neuron may be activated again. This means that a particular neuron wins repeatedly. So this neuron would learn more. To prevent this, conscience learning is a way, which had been proposed by De Sieno [8]. Further, Rival penalized competitive learning (RPCL) [9] and its variant Rival penalized controlled competitive learning (RPCCL) [10–13] was also proposed. SOM preserves the topology of input data by assigning each datum to a neuron having the highest similarity, and data with similar attributes are mapped into adjacent neurons [14].

The remainder of this paper is organized as follows. In Section 2, we explain the conventional SOM learning algorithm. In Section 3, the proposed SOM learning algorithm

* Corresponding author.
E-mail address: vikas_2610@rediffmail.com (V. Chaudhary).
Peer review under responsibility of Faculty of Engineering, Alexandria University.

http://dx.doi.org/10.1016/j.aej.2014.09.007

is explained. In Section 4, we conduct the experiments to compare the performance. In Section 5, we discuss the conclusion.

2. Self-Organizing Map (SOM)

The SOM consist of m neurons located at a regular low-dimensional map, usually a 2-D map. These neurons [15] are connected with their neighbors according to topological connections. There are two common types of topologies rectangular and hexagonal [16,17] for SOM map. Each neuron i has a d-dimensional weight vector w = ((w_{i1}, w_{i2},....., w_{id}), where $i = 1, 2. \ldots m$, which has the same dimension as the input space.

The conventional SOM learning algorithm can be explained using the following steps:

(a) Initialize the weight vectors $w_i's$ of the $m \times n$ neurons.
(b) Randomly select an input vector $x(t)$ and it is input to all the neurons at the same time in parallel.
(c) Find the winner neuron c, i.e., BMU using the following equation:

$$c = \arg\left(\min_{1 \le i \le mn} \{\|w_i(t) - x(t)\|\}\right), \tag{1}$$

||.|| is the Euclidean distance measure. Where $x(t)$ and $w_i(t)$ are the input and weight vector of neuron i at iteration t respectively.

(d) The weight vector of the neurons is updated using the following equation:

$$w_i(t+1) = w_i(t) + h_{c,i}(t)[x(t) - w_i(t)], \tag{2}$$

where $h_{c,i}(t)$ is a Gaussian neighborhood function [16] given below:

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

where r is the coordinate position of the neuron on the map, $\alpha(t)$ is the learning rate and $\sigma(t)$ is the width of neighborhood radius. Both $\alpha(t)$ and $\sigma(t)$ decrease monotonically using the following equation:

$$\alpha(t) = \alpha(0)\left(\frac{\alpha(T)}{\alpha(0)}\right)^{t/T}, \tag{4}$$

$$\sigma(t) = \sigma(0)\left(\frac{\sigma(T)}{\sigma(0)}\right)^{t/T} \tag{5}$$

where T is the training length.

(e) For all the input data, steps (b) to (d) are repeated.

3. Modified SOM

For each input data, the neurons at minimum and maximum distance from among 1-neighborhood of the BMU are found out as shown in Fig. 1. These are then named nearest and farthest neuron for that particular input. The proposed learning algorithm of SOM can be summarized in the following steps:

(Step 1) All the weight vectors $w_i \in \mathcal{M}$ of $m \times n$ neurons are initialized, where $i = 1, 2, \ldots, mn$ and $\mathcal{M}$ is a set of $m * n$ weight vectors. Then the winning frequency $\eta_i = 0$ is initialized for all neurons and the connection value $C_{(i,j)} = 0$ is also initialized between each neuron.

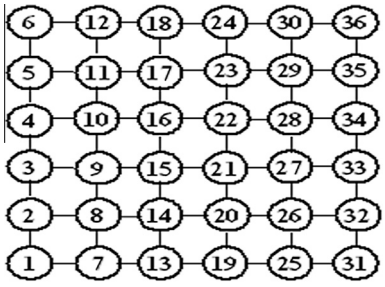

Figure 1 Neighborhood on the rectangular grid. Suppose $C = 16$, $Nc_1 = \{10, 15, 17, 22\}$. If $f = 15$, $S_f = \{13, 14\}$. If $f = 22$, $S_f = \{28, 34\}$.

(Step 2) An input vector $x(t)$ is selected randomly and given simultaneously to all the neurons.

(Step 3) The winner neuron c, i.e., BMU is found out using Eq. (1). Then, the distance between input $x(t)$ and weight vector is found and the rank $rank_i$ is assigned to each neuron, where $i = 0, 1, \ldots mn$. The rank $rank_i$ is taken to be 0 for the BMU, because of being nearest to the input vector. The winning frequency η_c of the winner neuron c is increased by 1.

(Step 4) The farthest neuron and the nearest neuron are found out from among the 1-neighborhood of BMU using Euclidean equation.

(Step 5) The connection value between BMU and neuron i is increased using the following equation:

$$C_{(c,i)} = C_{(c,i)} + 1, \tag{6}$$

where $i = f$ or $i \in S_f$.

Also, the relative winning frequency λ_i of the neuron i is calculated using the following equation:

$$\lambda_i = \eta_i \Big/ \sum_{j=1}^{\mathcal{M}} \eta_j \tag{7}$$

(Step 6) Except for the nearest neuron, the weight vectors of the winner neuron and its neighbors are updated using the following equation:

$$w_i(t+1) = w_i(t) + h_{c,i}(t)[x(t) - w_i(t)], \tag{8}$$

where the function $h_{c,i}(t)$ is the neighborhood function described as follows:

$$h_{c,i}(t) = \alpha(t) \cdot (1 - \lambda_i) \cdot \exp\left(-\frac{\gamma_{(c,i)}}{2\sigma^2(t)}\right), \tag{9}$$

$$\gamma_{(c,i)} = r_i + \left(||r_c - r_i||^2 + C_{(c,i)}\right), \tag{10}$$

Both $\alpha(t)$ and $\sigma(t)$ decrease consistently with time using Eqs. (4) and (5) respectively.

(Step 7) The weight vector of the nearest neuron is updated using the following equation:

$$w_q(t+1) = w_q(t) + h_{c,q}(t)[x(t) - w_q(t)], \tag{11}$$