A Fast Self-Organizing Map Algorithm for Handwritten Digit Recognition

Yimu Wang, Alexander Peyls, Yun Pan, Luc Claesen and Xiaolang Yan

Abstract This paper presents a fast version of the self-organizing map (SOM) algorithm, which simplifies the weight distance calculation, the learning rate function and the neighborhood function by removing complex computations. Simplification accelerates the training process in software simulation and is applied in the field of handwritten digit recognition. According to the evaluation results of the software prototype, a 15–20 % speed-up in the runtime is obtained compared with the conventional SOM. Furthermore, the fast SOM accelerator can recognize over 81 % of handwritten digit test samples correctly, which is slightly worse than the conventional SOM, but much better than other simplified SOM methods.

Keywords Neural network · Self-organizing map · Handwritten digit recognition · Simplification

1 Introduction

The self-organizing map (SOM) also called Kohonen neural network is a competitive learning artificial neural network proposed by Kohonen in 1982 [1]. It is an unsupervised learning method which has both visualization and clustering properties by discovering the topological structure hidden in the data sets. Essentially the goal of a self-organizing map is to map continuous high-dimensional data onto a discrete low (typically one- or two-) dimensional feature map.

Y. Wang (✉) · Y. Pan · X. Yan
Institute of VLSI Design, Zhejiang University, Hangzhou, People's Republic of China
e-mail: wym85511@gmail.com

A. Peyls · L. Claesen
EDM, Hasselt University, Diepenbeek, Belgium

As a clustering algorithm, the SOM has been applied widely in various fields including pattern recognition, defect inspection and as a data-mining tool to perform classification of high-dimensional data [2, 3]. Research on improving the performance of the SOM has been going on for decades. One of the key issues to overcome is the low speed learning process while obtaining a well trained map. A SOM is well trained if clustering is achieved in a short time and, at the same time, it creates a projection of data into the map strongly related to the distribution of data in the input space. One of the main reasons for this continued research effort is that the amount of data which is to be analyzed can be huge, for instance thousands of high-dimensional image vectors. The simulation of extensive networks with thousands of neurons, each with high-dimensional weights takes relatively much time on state of the art general purpose computers. To solve this problem, this paper presents a fast version of the SOM algorithm and software simulation proves that the SOM has been accelerated to some extent.

The remainder of this paper is organized as follows: Sect. 2 gives a brief overview of related works. Next, Sect. 3 presents the conventional self-organizing map. Section 4 presents our proposed fast self-organizing map algorithm. Section 5 discusses the experimental results on handwritten digit recognition and finally in Sect. 6 the conclusions are drawn.

2 Related Work

To improve both the efficiency and effectiveness of the conventional SOM algorithm, many approaches have been proposed. A first possibility to reduce the runtime of the SOM is to compute initial values for the feature map instead of choosing them randomly in such a way that the training will be accelerated. In [4] the K-means clustering algorithm is used to select initial values for the weight vectors of the neurons, which subsequently reduces the required amount of training steps. Because the SOM offers multiple opportunities to exploit the parallel computing [5], a second way of handling the computational complexity is to transform the SOM algorithm into a distributed algorithm. Lobo et al. developed a distributed SOM in order to speed up the training of the SOM [6]. In order to shorten the processing time the batch version of the SOM has been used by Yu and Alahakoon [7], this version of the SOM is also more suitable for parallel implementation. A third way of accelerating the neural computations is to design simplified SOM algorithm. The weight update step is simplified by removing the non-linear functions in the following papers [8, 9] and therefore results in a more hardware-friendly version of the SOM algorithm. Nevertheless, these simplified methods suffer from a low recognition accuracy and are hardly effective in complex applications. In this context, a fast SOM algorithm is proposed in this paper which not only speeds up the training process but also promises a similar recognition accuracy with the conventional SOM.

3 Self-Organizing Map Algorithm

1. Initialization step: At the start of the SOM algorithm, typically all the weights w_j of the neurons are initialized with random values.
2. Compute the distance between the training vector $X = \{x_1, \ldots, x_M\}$ and each neuron N_j with weight w_j, using the Euclidean distance function:

$$D_j = \sqrt{\sum_{i=1}^{M} (x_i - w_{ji})^2} \tag{1}$$

3. Define the winning neuron as the neuron with the minimum distance.
4. Update each neuron according to the following update function:

$$w_j(t+1) = w_j(t) + \alpha(t) \cdot N_{j,I(X)}(t) \cdot (X - w_j(t)) \tag{2}$$

$w_j(t+1)$ is the updated weight vector, $\alpha(t)$ the learning rate and $N_{j,I(X)}(t)$ the topological neighbourhood value at training step t. $I(X)$ is the winning neuron.

5. Update the neighborhood function and the learning rate.
6. Repeat steps 2–5 for the next training vector.

4 Fast Self-Organizing Map Algorithm

Distance calculation The conventional self-organizing map uses the Euclidean norm as the distance calculating function (see Eq. 1), however because it involves the squaring of values and a square root, the Euclidean distance computation is time-consuming for software prototype and also resource-intensive for hardware implementation. Following [8, 10], we use the Manhattan distance which is computationally simpler for calculating the distance between vectors.

$$D_j = \sum_{i=1}^{M} |(x_i - w_{ji})| \tag{3}$$

Learning Rate Function The learning rate is typically defined as the following exponential function.

$$\alpha(t) = \alpha_0 \cdot exp\left(\frac{-t}{\tau_\alpha}\right) \tag{4}$$

Note that because of the multiplication, division and exponential function, this function will cost too much learning time. To reduce the computational complexity imposed by the exponential calculation of the conventional SOM, an alternative formula is selected to substitute the conventional Eq. 4. Actually, this alternative is the first term of the Taylor series expansion of Eq. 4.

$$\alpha = \alpha_0 \left(1 - \frac{t}{T}\right) \tag{5}$$

Neighborhood Function When using Kohonen's self-organizing map, the distance in the feature map between the neurons influences the learning process. A typical neighborhood function is shown in Eq. 6, which decreases not just over time, but also depends on the topological distance of the two neurons in the net.

$$N_{x,y}(t) = exp\left(\frac{-d_{x,y}^2}{2\sigma^2(t)}\right) \tag{6}$$

Here $d_{x,y}$ is the distance between node x and node y, more specifically it is the physical distance between the nodes in the feature map and $\sigma(t)$ the time dependent value responsible for decreasing the neighborhood size over time.

However our proposed neighborhood function ignores any influence of time and only depends on the topological distance between two neurons, which is computed by the Euclidean norm. For each neuron within the neighborhood size *ns*, the neighborhood parameter is calculated as shown in Eq. 7. Neurons outside this neighborhood area will not be updated.

$$N_{x,y} = \begin{cases} e^{-2d_{x,y}^2} & if\ d(x,y) \leq ns \\ 0 & if\ d(x,y) > ns \end{cases} \tag{7}$$

The weight update function depends on both the learning rate function and the neighborhood function. In Fig. 1, respectively the results of the conventional weight update function and our proposed weight update function are shown in the condition of $\alpha_0 = 1, d_{x,y} = [0,4], t = [0,10000]$. Note that the shapes of the 3D charts based on these functions are similar, which motivates the similarity in performance between both versions. The performance results will be given in the last section.

5 Case Study: Handwritten Digit Recognition

The performance of the SOM was tested in the field of handwritten digit recognition and the MNIST database was chosen to train and test the feature map [11]. We evaluated the proposed fast SOM by a software simulation on a PC with a general purpose processor clocked at 2.1 GHz and 2 GB of SDRAM. In Fig. 2, the runtime with varying amounts of iterations and varying amounts of neurons is

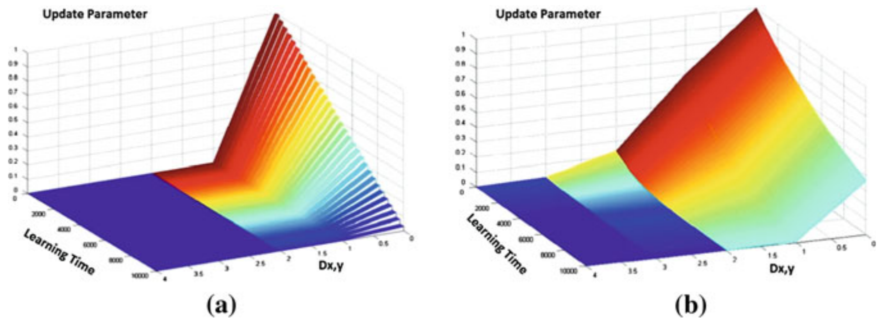
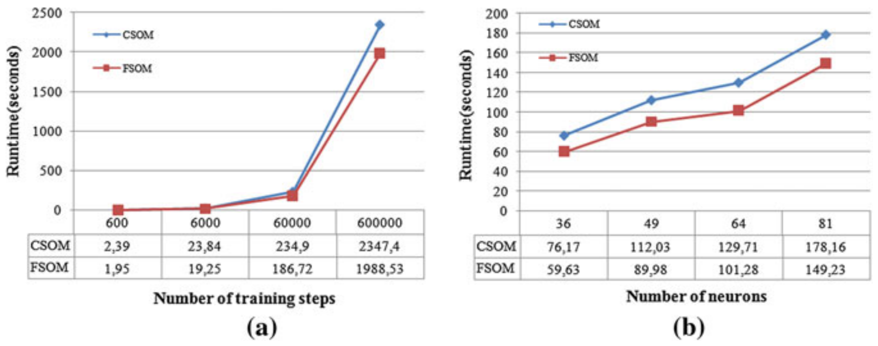


Fig. 1 Comparison of weight update functions. **a** Proposed weight update function. **b** Conventional weight update function



	600	6000	60000	600000
CSOM	2,39	23,84	234,9	2347,4
FSOM	1,95	19,25	186,72	1988,53

	36	49	64	81
CSOM	76,17	112,03	129,71	178,16
FSOM	59,63	89,98	101,28	149,23

Fig. 2 Runtime comparison between conventional and fast SOM. **a** Runtime with different training steps. **b** Runtime with varying numbers of neurons

compared between the proposed fast SOM and the conventional SOM respectively. By using the Manhattan distance metric, a simplified learning rate and neighborhood function, no multiplications and exponential computations is required, due to this a reduction in time ranging from 15 to 20 % can be achieved.

Furthermore, it is also important to note that the proposed fast SOM algorithm is able to obtain this speed-up while maintaining an accuracy which is similar compared to the conventional SOM. Our fast SOM algorithm outperforms other simplified SOM algorithms such as [8, 9].

In Table 1 the recognition accuracy of the conventional, our fast version and also Pena's [10] SOMs are shown. These were obtained by various numbers of iterations and each iteration equals training the feature map with 60,000 input vectors. Afterwards, the SOM is tested with 10,000 test samples. Finally the feature maps of the conventional and proposed SOMs are shown in Fig. 3. The neurons of both maps clearly organized themselves and clusters can be distinguished.

Table 1 Recognition accuracy on MNIST database

Iterations	Conventional SOM (%)	Proposed fast SOM (%)	Pena's SOM (%)
1	82.34	81.83	64.96
10	84.93	83.24	66.03
100	85.01	83.79	66.81
200	85.74	84.13	67.24

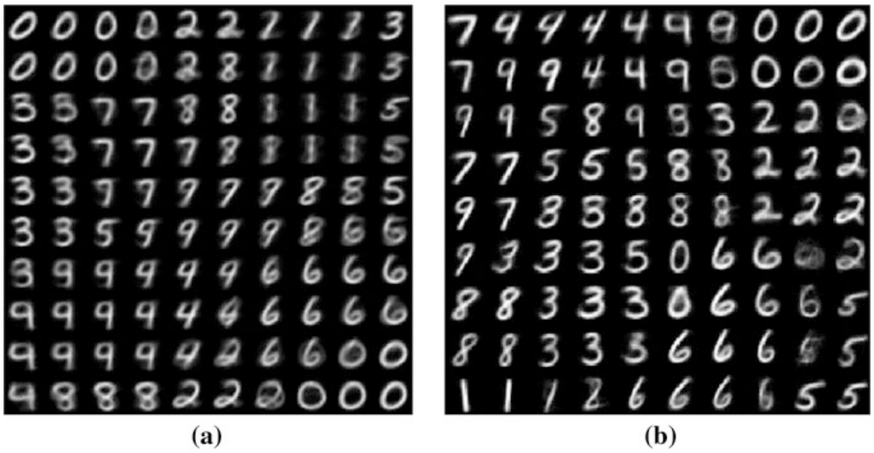


Fig. 3 Feature map after training with MNIST dataset. **a** Feature map of proposed SOM. **b** Feature map of conventional SOM

6 Conclusion

This paper proposes a fast SOM algorithm for handwritten digit recognition which simplifies the conventional SOM by removing complex computations in the weight distance calculation, the learning rate function and the neighborhood function. After evaluating the performance in software simulation, we conclude that the proposed fast SOM algorithm can reach the goal of accelerating to some extent, maintain similar recognition accuracy compared to the conventional SOM and performs much better than other simplified SOM methods.

References

1. Kohonen T (1990) The self-organizing map. Proc IEEE 78(1):1464–1480
2. Kohonen T, Kaski S, Lagus K et al (2000) Self organization of a massive document collection. IEEE Trans Neural Netw 11(3):574–585
3. Silven O, Niskanen M, Kauppinen H (2003) Wood inspection with non-supervised clustering. Mach Vis Appl 3:275–285

4. Mu-Chun S, Hsiao-Te C (2000) Fast self-organizing feature map algorithm. IEEE Trans Neural Netw 11(3):721–732
5. Nordström T (1992) Designing parallel computers for self organizing maps. In: Fourth Swedish workshop on computer system architecture
6. Lobo VJ, Bandeira N, Moura-Pires F (1998) Distributed Kohonen networks for passive sonar based classification. In: International conference on multisource-multisensor information fusion, Las Vegas
7. Yaohua Y, Damminda A (2006) Batch implementation of growing self-organizing map. In: International conference on computational intelligence for modelling control and automation, and international conference on intelligent agents, web technologies and internet commerce
8. Pena J, Vanegas M (2006) Digital hardware architecture of Kohonen's self organizing feature maps with exponential neighboring function. In: IEEE international conference on reconfigurable computing and FPGA
9. Agundis R, Girones G, Palero C, Carmona D (2008) A mixed hardware/software SOFM training system. Computaciny Sistemas 4:349–356
10. Porrmann M, Witkowski U, Ruckert U (2006) Implementation of self-organizing feature maps in reconfigurable hardware. In: FPGA implementations of neural networks. Springer, Heidelberg, pp 247–269
11. LeCun Y, Cortes C, The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/