Neurocomputing 207 (2016) 78–94

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Self-adjusting feature maps network and its applications

Dong-Lin Li [a], Mukesh Prasad [b], Chin-Teng Lin [a,c], Jyh-Yeong Chang [a]

[a] Department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan
[b] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
[c] Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, Australia

ARTICLE INFO

Article history:
Received 26 October 2015
Received in revised form
22 January 2016
Accepted 20 March 2016
Available online 10 May 2016

Keywords:
Unsupervised learning
Self-adjusting
Statistics
Quantization
Self-organizing map
Artificial neural networks

ABSTRACT

This paper, proposes a novel artificial neural network, called self-adjusting feature map (SAM), and develop its unsupervised learning ability with self-adjusting mechanism. The trained network structure of representative connected neurons not only displays the spatial relation of the input data distribution but also quantizes the data well. The SAM can automatically isolate a set of connected neurons, in which, the used number of the sets may indicate the number of clusters. The idea of self-adjusting mechanism is based on combining of mathematical statistics and neurological advantages and retreat of waste. In the training process, for each representative neuron has are three phases, growth, adaptation and decline. The network of representative neurons, first create the necessary neurons according to the local density of the input data in the growth phase. In the adaption phase, it adjusts neighborhood neuron pair's connected/disconnected topology constantly according to the statistics of input feature data. Finally, the unnecessary neurons of the network are merged or remove in the decline phase. In this paper, we exploit the SAM to handle some peculiar cases that cannot be handled easily by classical unsupervised learning networks such as self-organizing map (SOM) network. The remarkable characteristics of the SAM can be seen on various real world cases in the experimental results.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

In machine learning techniques, unsupervised learning involves a class of problems that determine how data are organized. Unsupervised learning is distinguishable from supervised learning and reinforcement learning, where the learner is provided only unlabeled examples. Unsupervised learning is closely related to the problem of density estimation in statistics, and also encompasses many other techniques that seek to summarize and explain key features of the data. For this reason, it is more meaningful to find common criteria when defining the correlation in data automatically. However, determining key features among higher dimensional data is not always easy. With some network architectures, it is possible to map patterns of arbitrary dimensional (the pattern space) onto a lower-dimensional structure of neurons (the feature space) in such a manner that similarity relationships present in the original data are remain the same after mapping. Such feature mapping can be useful for data visualization. A well-known approach to designing a self-organizing map (SOM) network that always uses Kohonen's learning rule [1]. Kohonen's SOM provides a topology-preserving map that retains neighborhood relationships of the input pattern, able to ask for proximal outputs of a feature map corresponding to proximal input patterns. Updated weights proceed to the winning neuron itself and the neighbors of the winning neuron.

However, two critical issues exist regarding the SOM. First, the network becomes unstable when input patterns exhibit a complex structure. Because the neighborhood relationships are constantly fixed in the training process, the previous topology-preserving map restricts the growth of the network structure. Weight vectors lying in zero-density areas are affected by input vectors from all surrounding parts of the non-zero distribution, which often occur in an SOM. The fluctuation ceases as the neighborhoods are shrunk; but a residual effect of the rigid neighborhoods is that some nodes usually remain outliers, even if causing many dead neurons is easy [2]. In general, the stability of competitive learning algorithms largely depends on initial random weights. In extreme situations, numerous dead nodes, which do not learn under any circumstances and to solve such problems, various ideas have been proposed. The parallel distributed processing (PDP) model was developed by Rumelhart et al. [3]. The PDP model has three basic principles: (a) the representation of information is distributed (not local); (b) memory and knowledge for specific items are not stored explicitly, but stored in the connections between units; and (c) learning can occur by experience with gradual changes in connection strength.

E-mail addresses: lazybones000@yahoo.com.tw (D.-L. Li), mukeshnctu.cs99g@nctu.edu.tw (M. Prasad), ctlin@mail.nctu.edu.tw (C.-T. Lin), jychang@mail.nctu.edu.tw (J.-Y. Chang).

http://dx.doi.org/10.1016/j.neucom.2016.03.067
0925-2312/© 2016 Elsevier B.V. All rights reserved.

In addition to the PDP model, W. Hu et al. proposed a fuzzy self-organizing neural network [4] that does not update weight according to neighborhood topology, but to a whole trajectory as an input to the network according to the fuzzy set theory. The second, difficulty with the SOM is that not only Kohonen's SOM, but also numerous networks fail to overcome the so-called stability plasticity dilemma. In other words, the weight vector set cannot adapt neatly to new input regions in situations, when the sequence of input vectors are non-stationary. To contend with this problem, some self-creating models are proposed [5–15]. The self-creating and organizing neural network (SCON) is proposed by Choi et al. [5], which uses an activation level to decide automatically whether to adapt the weights of existing nodes or to create a new "son node". The growing cell structure (GCS) proposed by Fritzke [6] replaces the grid network topology by building triangular blocks. The GCS both adds new nodes to and removes existing nodes from the network during the training process and the connections among nodes are adjusted to maintain the triangular connectivity. Fritzke also proposed the growing neural gas (GNG) algorithm [7] that combines the GCS and the neural gas algorithm [8]. A conservation principle proposed by Wang et al. [9] is incorporated with the competitive learning algorithm to harmonize equi-probable and equi-distortion criteria [10]. The growing self-organizing map (GSOM) proposed by Alahakoon et al. grows new nodes on the boundary based on a heuristic by using the value called Spread Factor. The branching competitive learning (BCL) network proposed by Xiong et al. [12] adopts a special node-splitting criterion, which is mainly based on the geometrical measurements of the movement of the synaptic vectors in the weight space. Liu proposed a robust energy artificial neuron based incremental self-organizing neural network with a dynamic structure (REISOD). It can adjust the scale of network automatically to adapt the scale of the data set and learn new data incrementally with preserving the former learnt results [13]. Yang proposed a novel self-constructing least-Wilcoxon generalized Radial Basis Function Neural-Fuzzy System (LW-GRBFNFS), which creates a self-constructing scenario for generating antecedent part of RBFNFS with particle swarm optimizer (PSO) [14]. Miche proposed two new clustering techniques based on Extreme Learning Machine (ELM). These clustering techniques can incorporate a priori knowledge (of an expert) to define the optimal structure for the clusters [15].

We cannot ensure the similarity of network structures after completing each training process, where identical input patterns are set in random input order. Training with unsupervised learning or SOMs usually requires clustering using supervised or unsupervised learning once more to obtain desired output va1ues and cluster numbers. To handle this problem, the feature map can be extended to a multilayer network by adding an association layer (output layer) to the output of the self-organizing map layer. The output nodes associate desired output values with certain input vectors. This structure is referred to as a self-organizing motor map, originally proposed by Ritter et al. [16]. The added layer is called a "motor map", in which movement commands are mapped to two-dimensional locations of excitation. In addition to the motor map, Brugger et al. proposed an enhanced version of the Clusot algorithm [17] that consists of two main steps: the computation of the Clusot surface utilizing the information contained in a trained SOM and the automatic detection of clusters in this surface.

Furthermore, the visualization of most proposed networks cannot directly represent the data structure or inter-neuron distance in its preserved network topology mapping. Several previous studies have proposed visualizing SOM for various concepts [18–24]. Visualization induced SOM (ViSOM) [18] is proposed to enhance SOM's visualization to preserve data structures and inter-neuron distances.

In this paper, a self-organizing and self-varying feature map network with a novel learning rule is proposed. To obtain a more adaptive neural network, the learning rule combines the topology-preserving map with the self-adjusting mechanism. The self-adjusting mechanism is based on statistics that evaluate what size and structure the network should contain without imposing excessive parameters. The proposed neural learning rule can dynamically change the size of the neural network and constantly adjust the neighborhood relationships of neurons according to input data in the training process. Therefore, it can ensure that the networks of different training trials are similar when training data are identical. In addition, visualizing the training result can obviously show distribution relationships existing in the input feature space using the neighborhood relationships among connected output neurons. In other words, we can automatically evaluate how many classes of the data samples are suitable without requiring any post process.

The remainder of this paper is organized as follows. In Section 2, we describe how to dynamically change the size of a neural network and adjust the neighborhood relationships of neurons in the training process using the self-adjusting mechanism. Furthermore, it introduces the learning rule and the parameters of the proposed neural network. In Section 3 describes the effect on different parameter selection. Section 4 shows the experimental results of our neural network. Finally, in Section 5 brings conclusions and discuss some issues of the proposed method along with some noteworthy topics for future work.

2. Self-adjusting mechanism

In this paper, a self-adjusting mechanism to dynamically adjust the network size and structure, according to input data distribution is proposed. The SAM network starts with a small number of neurons (chosen 4 in this paper) with initial random weights. The self-adjusting mechanism involves two aspects: the adaptive network size and dynamic neighborhood topology.

Assume the input vector is $\vec{x}_i(t)$ for $i=1,\cdots,M$ and the weight vector of the neuron is $\vec{w}_j(t)=\left[\vec{w}_{j1},\vec{w}_{j2},\cdots,\vec{w}_{jq}\right]^T$ for $j=1,\cdots,N$, where M and N represent the total number of training data and the neuron number, respectively. Parameters q and t represent the dimension of the input vector and the total time elapsed during learning process, respectively. Since the learning time scale is preceded with an epoch-based approach, thus the learning time t elapsed can be written as $t=nT_{ep}$, in which T_{ep} is the training time needed for an epoch learning of all input data learned once. Hereafter, we can equivalently write $t=nT_{ep}=n$ (discrete domain) for brevity. The input vector is compared with all the weight vectors of the neurons. A best-matching unit (BMU) $C_i(t)$ can be determined by calculating the Euclidean distance between the input vector and the weight vector:

$$C_i(t)=\underset{j}{\operatorname{argmin}}\{\|\vec{x}_i(t)-\vec{w}_j(t)\|\} \tag{1}$$

After the training iteration, the standard deviation $STD_j(t)$ of data subset whose winner is neuron j is defined as follows:

$$STD_j(t)=\sqrt{\frac{\sum_{i=C_i\in \text{Neuron}_j}\left(\vec{x}_i-\vec{w}_j(t)\right)^2}{m}} \tag{2}$$

where m and t represent the number of the input belong to neuron j and current iteration.

Download English Version:

https://daneshyari.com/en/article/494441

Download Persian Version:

https://daneshyari.com/article/494441

Daneshyari.com