Neurocomputing 207 (2016) 310–321

Contents lists available at ScienceDirect

Neurocomputing

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

A novel progressive learning technique for multi-class classification

Rajasekar Venkatesan *, Meng Joo Er

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 11 November 2015
Received in revised form
4 March 2016
Accepted 1 May 2016
Communicated by: Ning Wang
Available online 11 May 2016

Keywords:
Classification
Machine learning
Multi-class
Sequential learning
Progressive learning

ABSTRACT

In this paper, a progressive learning technique for multi-class classification is proposed. This newly developed learning technique is independent of the number of class constraints and it can learn new classes while still retaining the knowledge of previous classes. Whenever a new class (non-native to the knowledge learnt thus far) is encountered, the neural network structure gets remodeled automatically by facilitating new neurons and interconnections, and the parameters are calculated in such a way that it retains the knowledge learnt thus far. This technique is suitable for real-world applications where the number of classes is often unknown and online learning from real-time data is required. The consistency and the complexity of the progressive learning technique are analyzed. Several standard datasets are used to evaluate the performance of the developed technique. A comparative study shows that the developed technique is superior.

1. Introduction

The study on feedforward neural network (FNN) has gained prominence since the advent of back propagation (BP) algorithm [1]. Several improved and optimized variants of the BP algorithm have then been developed and analyzed [2–7]. In the past two decades, single hidden layer feedforward neural networks (SLFNs) has gained significant importance due to its widespread applications in recognition, classification and function approximation area [8–12]. Several learning techniques have been proposed since then for effective training of the SLFN [13,14]. The learning techniques can be grouped under two basic categories; Batch learning and Sequential learning [15].

Batch learning algorithms require pre collection of training data. The collected data set is then used for training the neural network. The network parameters are calculated and updated by processing all the training data together. There are several batch learning algorithms in the literature. One of the relatively new batch learning scheme called Extreme Learning Machines (ELM) is proposed by Huang et al. [16]. The special nature of ELM is that the input weights and the hidden node biases can be chosen at random [17]. A key feature of ELM is that it maintains the universal approximation capability of SLFN [17–19]. It has gained much attention to it and several research works are further made in it due to its special nature of random input weight initialization and its unique advantage of extreme learning speed [20]. The advantages of ELM over other traditional feedforward neural network are analyzed in the literature [9,21]. Many new variants and developments are made to the ELM and significant results are achieved in the approximation, classification and regression areas [22–24]. Batch learning involves processing of the complete data set concurrently for updating the weights. This technique is limited to its applications as batch learning techniques are more time consuming and requires the complete data set prior to training. On the other hand, in online/sequential learning algorithms, the network parameters are updated as and when a new training data arrives. To overcome the shortcomings of the batch learning techniques, several sequential and/or online learning algorithms are developed [25–28].

In many cases, sequential learning algorithms are preferred over batch learning algorithms as they do not require retraining whenever a new data sample is received [8]. Online-sequential learning method that combines ELM and Recursive Least Square (RLS) algorithm is later developed and is called online-sequential extreme learning machine (OS-ELM) [15]. Several variants of ELM and OS-ELM were developed and proposed in the literature [8,9,17,22,23,29].

The issue with existing multi-class classification techniques such as ELM and SVM is that, once they are trained to classify a specific number of classes, learning of new classes is not possible. In order to learn new class of data it requires retraining all the classes anew again.

The existing techniques require a priori information on the number of classes that will be present in the training dataset. The information on the number of classes is required to be either specified directly or is identified by analyzing the complete training data set. Based on this parameter, the network model will be designed and only the parameters or the weights of the networks are updated

* Corresponding author.
E-mail addresses: RAJA0046@e.ntu.edu.sg (R. Venkatesan), EMJER@ntu.edu.sg (M.J. Er).

http://dx.doi.org/10.1016/j.neucom.2016.05.006
0925-2312/

depending on the sequential input data. This makes the existing techniques "static" with respect to the number of classes it can learn.

While the existing techniques are suited for applications with pre-known dataset, it might not be well suited for applications such as cognitive robotics or those involving real-time data where the nature of training data is unknown. For such real-world and real-time data, where the number of classes to be learnt is often unknown, the learning technique must be self-developing to meet the dynamic needs. To overcome this shortcoming, a novel learning paradigm is proposed, called the "progressive learning".

Progressive learning is the next stage of advancement to the online learning methods. Existing online sequential techniques only learn to classify data among a fixed set of classes which are initialized during the initialization phase of the algorithm. They fail to dynamically adapt when introduced to new class/classes on the run. The progressive learning technique is independent of the number of class constraint and it can learn several new classes on the go by retaining the knowledge of previous classes. This is achieved by modifying the network structure by itself upon encountering a new class and updating the network parameters in such a way that it learns the new class and retains the knowledge learnt thus far.

The existing online sequential learning methods do not require retraining when a "new data sample" is received. But it fails when a "new class of data" which is unknown to the existing knowledge is encountered. Progressive learning technique overcomes this shortcoming by allowing the network to learn multiple new classes' alien to existing knowledge, encountered at any point of time.

2. Preliminaries

This section gives a brief review of the ELM and the OS-ELM techniques to provide basic background information.

2.1. Extreme learning machines

A condensed overview of the batch learning ELM technique as proposed by Huang et al. [16] is given below.

Consider there are N training samples represented as $\{(\mathbf{x}_j,\mathbf{t}_j)\}$ where j varies from 1 to N, $\mathbf{x}_j$ denotes the input data vector: $\mathbf{x}_j=[x_{j1},x_{j2},\ldots x_{jn}]^{\mathrm{T}}\in R^n$ and $\mathbf{t}_j=[t_{j1},t_{j2},\ldots,t_{jm}]^{\mathrm{T}}\in R^m$ denotes the target class labels. Let there be P number of hidden layer neurons in the network, the output of the standard SLFN can be given as

$$\sum_{i=1}^{P}\boldsymbol{\beta_i}g_i(\boldsymbol{x_j})=\sum_{i=1}^{P}\boldsymbol{\beta_i}g(\boldsymbol{w_i}.\boldsymbol{x_j}+b_i)=\boldsymbol{o_j} \tag{1}$$

where $j=1, 2\ldots N$, $\mathbf{w_i}=[w_{i1},w_{i2},\ldots w_{in}]^{\mathrm{T}}$ denotes the weight vector from input nodes to ith hidden node, $\boldsymbol{\beta}_i=[\beta_{i1}, \beta_{i2},\ldots \beta_{im}]^{\mathrm{T}}$ denotes the weight vector connecting ith hidden node to the output nodes and b_i is the hidden layer bias value.

For the standard SLFN mentioned in the equation above to perform as a classifier, the output of the network should be equal to the corresponding target class of the input data given to the classifier. Hence, for the SLFN in Eq. (1) to be a classifier, there exist a $\boldsymbol{\beta}_i$, $g(x)$, $\mathbf{w}_i$ and b_i such that

$$\sum_{j=1}^{P}\|\boldsymbol{o_j}-\boldsymbol{t_j}\|=0 \tag{2}$$

Therefore, the equation for the output of the network can be written as

$$\sum_{i=1}^{P}\boldsymbol{\beta_i}g(\boldsymbol{w_i}.\boldsymbol{x_j}+b_i)=\boldsymbol{t_j} \tag{3}$$

where $j=, 2,\ldots N$, and $\mathbf{t}_j$ denotes the target class corresponding to the input data vector $\mathbf{x}_j$. This equation can be written in compact form as

$$H\boldsymbol{\beta}=\mathbf{T} \tag{4}$$

where

$$\boldsymbol{H}(w1,\ldots wP,b1,\ldots,bP,x1,\ldots.,xN)=\begin{bmatrix} g(w_1.x_1+b_1) & \cdots & g(w_P.x_1+b_P)\\ \vdots & \ddots & \vdots\\ g(w_1.x_N+b_1) & \cdots & g(w_P.x_N+b_P)\end{bmatrix}_{NXP} \tag{5}$$

$$\boldsymbol{\beta}=\begin{bmatrix}\beta_1^T\\ \vdots\\ \beta_P^T\end{bmatrix}_{PXm} \tag{6}$$

$$\boldsymbol{T}=\begin{bmatrix}t_1^T\\ \vdots\\ t_N^T\end{bmatrix}_{NXm} \tag{7}$$

$\mathbf{H}$ is called the hidden layer output matrix of the neural network where each column of $\mathbf{H}$ gives corresponding output of the hidden layers for a given input $\mathbf{x}_i$. The mathematical framework and the training process are extensively described in the literature [9]. The key results are restated.

Lemma 1. *[9] Given a standard SLFN with N hidden nodes and activation function g: R → R which is infinitely differentiable in any interval, for N arbitrary distinct samples $(\boldsymbol{x}_i,\boldsymbol{t}_i)$, where $\boldsymbol{x}_i\in R^n$ and $\boldsymbol{t}_i\in R^m$, for any $\boldsymbol{w}_i$ and b_i randomly chosen from any intervals of R^n and R, respectively, according to any continuous probability distribution, then with probability one, the hidden layer output matrix $\boldsymbol{H}$ of the SLFN is invertible and $\|\boldsymbol{H\beta}-\boldsymbol{T}\|=0$.*

Lemma 2. *[9] Given any small positive value $\varepsilon>0$ and activation function g: R → R which is infinitely differentiable in any interval, there exists $P\le N$ such that for N arbitrary distinct samples (x_i,t_i), where $\boldsymbol{x}_i\in R^n$ and $\boldsymbol{t}_i\in R^m$, for any $\boldsymbol{w}_i$ and b_i randomly chosen from any intervals of R^n and R, respectively, according to any continuous probability distribution, then with probability one, $\|\boldsymbol{H}_{NxP}\boldsymbol{\beta}_{PXm}-\boldsymbol{T}_{NXm}\|<\varepsilon$.*

Thus it can be seen that for an ELM, the input weights w_i, and the hidden layer neuron bias bi can be randomly assigned. Training of the ELM involves estimating the output weights $\boldsymbol{\beta}$ such that the relation $\mathbf{H}\boldsymbol{\beta}=\mathbf{T}$ is true.

The output weight $\boldsymbol{\beta}$ for the ELM can be estimated using the Moore-Penrose generalized inverse as $\boldsymbol{\beta}=\mathbf{H}^{+}\mathbf{T}$, where $\mathbf{H}^{+}$ is the Moore-Penrose inverse of the hidden layer output matrix $\mathbf{H}$.

The overall batch learning algorithm of the ELM for training set of form $\{(\mathbf{x}_i,\mathbf{t}_i)|\mathbf{x}_i\in R^n, \mathbf{t}_i\in R^m, i=1,\ldots N\}$ with P hidden layer neurons can be summarized as,

STEP 1: Random assignment of input weights $\mathbf{w}_i$ and hidden layer bias b_i, $i=1,\ldots P$.
STEP 2: Computation of the hidden layer output matrix $\mathbf{H}$.
STEP 3: Estimation of output weights using $\boldsymbol{\beta}=\mathbf{H}^{+}\mathbf{T}$ where $\mathbf{H}^{+}$ is the Moore–Penrose inverse of $\mathbf{H}$ and $\mathbf{T}=[t_1,\ldots t_N]^{\mathrm{T}}$.

2.2. Online sequential – extreme learning machine

Based on the batch learning method of the ELM, sequential modification is performed and Online Sequential-ELM (OS-ELM) is proposed in literature [15]. OS-ELM operates on online data.

In the batch learning method ELM the output weight β is estimated using the formula $\boldsymbol{\beta}=\mathbf{H}^{+}\mathbf{T}$, where $\mathbf{H}^{+}$ is the Moore–Penrose inverse of the hidden layer output matrix $\mathbf{H}$. The $\mathbf{H}^{+}$ can