Journal of Mathematical Research with Applications
Sept., 2016, Vol. 36, No. 5, pp. 621–630
DOI:10.3770/j.issn:2095-2651.2016.05.012
Http://jmre.dlut.edu.cn

Online Sequential Double Parallel Extreme Learning Machine for Classifications

Mingchen YAO, Chao ZHANG*, Wei WU
School of Mathematical Sciences, Dalian University of Technology, Liaoning 116024*, P. R. China*

Abstract Double parallel forward neural network (DPFNN) model is a mixture structure of single-layer perception and single-hidden-layer forward neural network (SLFN). In this paper, by making use of the idea of online sequential extreme learning machine (OS-ELM) on DPFNN, we derive the online sequential double parallel extreme learning machine algorithm (OS-DPELM). Compared to other similar algorithms, our algorithms can achieve approximate learning performance with fewer numbers of hidden units, as well as the parameters to be determined. The experimental results show that the proposed algorithm has good generalization performance for real world classification problems, and thus can be a necessary and beneficial complement to OS-ELM.

Keywords double parallel forward neural network; perception; extreme learning machine; classification problems

MR(2010) Subject Classification 92B20; 68T05; 68W27

1. Introduction

Double parallel forward neural network (DPFNN) model is a mixture structure of single-layer perception and single-hidden-layer forward neural network (SLFN) [1–3], in which there exists an additional connection from the input layer to the output layer, therefore DPFNN is also referred to as bypass neural network [4]. In DPFNN, the output nodes receive information not only through the hidden nodes but also directly through the input nodes, thus it is possible to improve the learning capacity via both reconstructed and original knowledge; From the mappings point of view, if the activation function used in hidden units is non-linear, while the activation function used in output units is linear, then DPFNN is a parallel mathematical model with both linear and nonlinear structure [1]. Since it has good learning capacity and generalization performance, the DPFNN model has been widely used in real world problems such as pattern recognition, function approximation and feature selection [1–3,5]. However, the frequently used learning algorithms in such a model are based on gradient descent of error function [3,6,7], which means all the parameters in the network have to be tuned iteratively by using such a slow method. As a result, the time expenditure is usually much higher than what we expected.

Received July 24, 2015; Accepted November 9, 2015
Supported by the National Natural Science Foundation of China (Grant Nos. 11401076; 61473328; 11171367; 61473059).
* Corresponding author
E-mail address: yaomingchen@hlju.edu.cn (Mingchen YAO); chao.zhang@dlut.edu.cn (Chao ZHANG); wuweiw@dlut.edu.cn (Wei WU)

Recently, an effective training algorithm for SLFNs called extreme learning machine (ELM), originated by Huang et al. [8–10], has aroused many research interests, where the input weights including hidden layer biases are randomly assigned and the output weights are determined by the Moore-Penrose generalized inverse of the hidden output matrix. The advantage of ELM lies in fast parameter learning speed for there exists no iteration process, and for many applications, the time consuming by ELM algorithms would be smaller by several orders of magnitude than that of by back prorogation (BP) or by support vector machine (SVM). According to Huang's results [8], to achieve an acceptable learning performance by using ELM algorithm, the network needs more hidden units than those of gradient-based algorithms, which will lead to a larger size of the network and adverse effect to the network generalization performance [11]. Huynh [12] then proposed regularized least square extreme learning machine (RLS-ELM) to overcome the above shortcomings, in which both input weights and outputs weights are calculated analytically by pseudo-inverse operation. For the same classification problem, this algorithm need few hidden units, and can offer good performance with compact network architecture. Yao [5] has successfully applied the idea of the ELM to the architecture of DPFNN, which results in the double parallel extreme learning machine algorithm (DP-ELM).

The training method of traditional ELM algorithm is usually of batch-learning type, which means the learning process is memory-consuming. In many applications, the training samples may arrive one by one or group by group, thus the latest samples are hard to utilize for batch training. To deal with these cases, Liang [13] proposed the online sequential extreme learning machine (OS-ELM) algorithm, and Huynh developed the OS-ELM algorithm, referred to as online RLS-ELM algorithm. All above methods can learn the data one by one or chunk by chunk with fixed or varying chunk size. Since these methods are based on ELM algorithm, the learning speed of them is much faster than that of traditional methods such as sequential stochastic gradient descent back propagation (SGBP) and Levenberg-Marquart algorithm [14–16]. When employing the idea of ELM, OS-ELM and RLS-ELM to the architecture of DPFNN, the online DPELM algorithm can be derived. Compared with OS-ELM and OS-RLS-ELM, similar learning speed and better generalization performance are achieved but fewer hidden units are used in the proposed algorithms.

The rest of the paper is organized as follows. Section 2 gives a brief review of structure DPFNN and DPELM algorithm. In Section 3, an illustration example is chosen to show that why DPELM algorithm can resolve classical XOR problem with least hidden units. Online sequential DEPLM algorithm is derived in Section 4. Performance evaluation for classification problem is shown in Section 5. The conclusion is given in last Section.

2. Structure of DPFNN and DPELM algorithm

Consider a three-layer DPFNN with d input nodes, N hidden nodes and c output nodes, as shown in Figure 1(a). We denote by $\mathbf{W} = (w_{ij})_{d\times N}$ the weight matrix connecting the input and hidden layers, where $\mathbf{w}^{(j)} = (w_{1j}, w_{2j}, \ldots, w_{dj})^\top \in \mathbb{R}^d$ is the weight vector connecting the input layer and j-th nodes of the hidden layer. Similarly, we denote the weight matrix connecting the hidden and the output layers by $\mathbf{U} = (u_{ij})_{N\times n}$, and $\mathbf{u}^{(k)} = (u_{1k}, u_{2k}, \ldots, u_{Nk})^\top \in \mathbb{R}^N$. The weight matrix directly connecting the input and output layers is denoted by $\mathbf{V} = (v_{ij})_{d\times c}$, and $\mathbf{v}^{(k)} = (v_{1k}, v_{2k}, \ldots, v_{dk})^\top \in \mathbb{R}^d$. For the hidden units with bias b_j, it is a common strategy to extend the dimension of $\mathbf{w}^{(j)}$ and the input pattern $\mathbf{x}$, that is, set $w_{d+1,j} = b_j$ and $x_{d+1} = 1$. In DPFNN, there exist no bias in the output units.

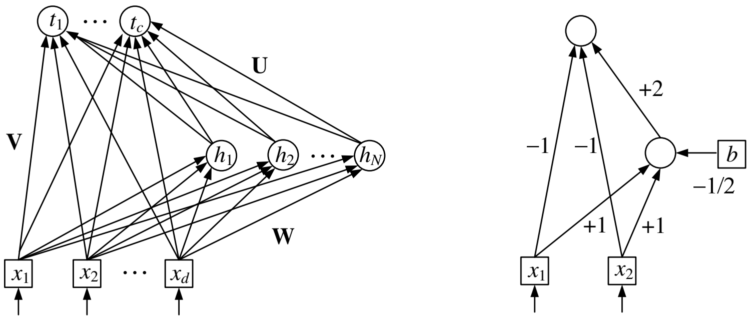

Figure 1 (a) Structure of DPFNN. (b) Network solving XOR problem.

For a given set of training examples $\{\mathbf{x}^{(i)}, \mathbf{t}^{(i)}\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}^c$, where $\mathbf{t}^{(i)}$ is the desired output of the input pattern $\mathbf{x}^{(i)}$, and $\mathbf{y}^{(i)}$ is the actual output of $\mathbf{x}^{(i)}$. The goal of the network learning is to determine the weight matrix $\mathbf{U}, \mathbf{V}$ and $\mathbf{W}$ minimizing the following error function,

$$E(\mathbf{U}, \mathbf{V}, \mathbf{W}) = \sum_{i=1}^{n} \left\|\mathbf{y}^{(i)} - \mathbf{t}^{(i)}\right\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{c} (\mathbf{h}^{(i)} \cdot \mathbf{u}^{(j)} + \mathbf{x}^{(i)} \cdot \mathbf{v}^{(j)} - \mathbf{t}^{(j)})^2, \tag{2.1}$$

where $\mathbf{h}^{(i)} = (g(\mathbf{x}^{(i)} \cdot \mathbf{w}^{(1)}), \ldots, g(\mathbf{x}^{(i)} \cdot \mathbf{w}^{(N)}))^{\mathrm{T}}$ represents the i-th hidden unit output and $g(\cdot)$ is the activation function of the hidden unit. The notation $\|\cdot\|$ represents the Euclidean norm, and the notation $\mathbf{x} \cdot \mathbf{y} = (\mathbf{x}, \mathbf{y})$ is the inner product of vector $\mathbf{x}$ and vector $\mathbf{y}$, respectively. If we denote

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}^{(1)} & \ldots & \mathbf{x}^{(n)} \\ 1 & \ldots & 1 \end{pmatrix}^\top, \quad \mathbf{P} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right)^\top, \quad \mathbf{T} == (\mathbf{t}_1, \ldots, \mathbf{t}_n)^\top, \tag{2.2}$$

$$\mathbf{W} = \begin{pmatrix} \mathbf{w}^{(1)} & \ldots & \mathbf{w}^{(N)} \\ b_1 & \ldots & b_N \end{pmatrix}, \quad \mathbf{U} = \left(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(c)}\right), \quad \mathbf{V} = \left(\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(c)}\right). \tag{2.3}$$

By (2.2) and (2.3), Eq. (2.1) can be converted into the form of the matrix,

$$\min E(\mathbf{U}, \mathbf{V}, \mathbf{W}) = \|g(\mathbf{XW})\mathbf{U} + \mathbf{PV} - \mathbf{T}\|^2, \tag{2.4}$$

where $g\,(\mathbf{XW})_{ij} = g([\mathbf{XW}]_{ij}) = g(\mathbf{x}^{(i)} \cdot \mathbf{w}^{(j)} + b_j)$. It is somewhat difficult to resolve all three unknown matrices $\mathbf{U}, \mathbf{V}$ and $\mathbf{W}$ simultaneously for the activation function $g(\cdot)$ of the hidden unit may be nonlinear. However, when the value of $\mathbf{W}$ is given, set $\mathbf{H} = (g(\mathbf{XW}), \mathbf{P})$ and $\mathbf{A} = (\mathbf{U}, \mathbf{V})^\top$, and Eq. (2.4) becomes

$$\min \|\mathbf{HA} - \mathbf{T}\|. \tag{2.5}$$

Since Eq. (2.5) is equivalent to the general least square problem, it suffices to determine matrix $\mathbf{A}$ if the values of matrices $\mathbf{H}$ and $\mathbf{T}$ are known.

In summary, given training set $\{\mathbf{x}^{(i)}, \mathbf{t}^{(i)}\}_{i=1}^{n} \subset \mathbb{R}^d \times \mathbb{R}^c$, activation function $g(\cdot)$ of hidden units, and the number of hidden units N, the double parallel ELM algorithm is described as follows:

```
Algorithm 1 Double Parallel Extreme Learning Machine Algorithm
Step 1. Randomly assign the values for weight matrix $\mathbf{W}$ of the input layer.
Step 2. Calculate the hidden-layer output weight matrix $\mathbf{H} = g(\mathbf{XW})$,
        set $\mathbf{H}_1 = (\mathbf{H}, \mathbf{P})$ and $\mathbf{A} = (\mathbf{U}, \mathbf{V})^\top$.
Step 3. Determine the output layer weight matrix $\mathbf{A}$ by $\mathbf{A} = \mathbf{H}_1^\dagger \mathbf{T}$,
        where $\mathbf{H}_1^\dagger$ denotes the pseudo-inverse of matrix $\mathbf{H}_1$.
```

Remark 2.1 Several methods can be used to calculate the pseudo-inverse of matrix $\mathbf{H}$. These methods may include orthogonal method, orthogonal projection method, iterative method, and singular value decomposition (SVD). There are two cases when using orthogonal projection methods, if $\mathbf{H}^\top\mathbf{H}$ is nonsingular, then $\mathbf{H}^\dagger = \left(\mathbf{H}^\top\mathbf{H}\right)^{-1}\mathbf{H}^\top$, and if $\mathbf{H}\mathbf{H}^\top$ is nonsingular, then $\mathbf{H}^\dagger = \mathbf{H}^\top\left(\mathbf{H}\mathbf{H}^\top\right)^{-1}$. Singular value decomposition method can always be used if $\mathbf{H}^\top\mathbf{H}$ or $\mathbf{H}\mathbf{H}^\top$ tends to become singular. For example, the function "pinv" used in MATLAB software is adopted the methodology of truncated singular value decomposition (TSVD).

3. An illustration example

Here, we will present a simple instance to show that why DPELM algorithm can deal with the classic exclusive-OR (XOR) problem perfectly with the minimum hidden units. There are four points (patterns) in the plane that correspond to the input patterns (0,0), (0,1), (1,0) and (1,1). The purpose is to construct a pattern classifier that produces the binary output 0 in response to the input pattern (0,0) or (1,1), and the binary output 1 in response to the input pattern (0,1) or (1,0). Figure 1(b) describes the network architecture involving a single hidden neuron for solving the XOR problem. We will show that DPELM algorithm indeed solves the XOR problem with only one hidden unit by constructing a truth table and the decision region.

For brevity, let $w_1 = w_2 = 1$ and $b = -\frac{1}{2}$. The net input of the hidden units is $\mathbf{XW} = (-\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{3}{2})^\top$, the hard limit activation function in the hidden units leads to the hidden output $(0, 1, 1, 1)^\top$. Furthermore, we have

$$\mathbf{H} = (g(\mathbf{XW}, \mathbf{P})) = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \text{ and } (\mathbf{U}, \mathbf{V}) = \left(\mathbf{H}^\top\mathbf{H}\right)^{-1}\mathbf{H}^\top\mathbf{T} = \begin{pmatrix} 2 \\ -1 \\ -1 \end{pmatrix},$$

Thus we obtain $\mathbf{U} = (2)$ and $\mathbf{V} = (-1, -1)$. The truth table of actual output of the network is listed as follows:

P	g(**XW**)**V**	**PV**	actual output
(0,0)	0	0	0
(0,1)	2	-1	1
(1,0)	2	-1	1
(1,1)	2	-2	0

Table 1 The truth table for XOR problem

The network that solves the XOR problem consists of two neurons. The decision boundary constructed by the hidden neuron is $x_1 + x_2 - \frac{1}{2} = 0$, as shown in Figure 2(a). For all points on the lower left side of the line, neuron outputs 0; for all points on the other side of the line, neuron outputs 1. Likewise, the decision boundary constructed by the output neuron is $x_1 + x_2 = 0$, as shown in Figure 2(b). The neuron outputs 0 only at point $(0, 0)$; otherwise, the neuron outputs 1. The final decision formed by the two neurons is $x_1 + x_2 - 2x_3 = 0$, as shown in Figure 2(c). All the points that lie in the hyper-plane output 0, whereas the other two points, say $(1, 0, 1)$ and $(0, 1, 1)$ output 1.

We have to note that when the output neuron has the bias term, the two different classification patterns lie exactly on the two sides of the hyper-plane. This accords with Cover's theory [17], i.e., a complex pattern-classification problem, casting in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated. For XOR problem, four nonlinearly separable patterns are mapped into three-dimensional space by the two neurons, which make it possible for them to be separated linearly.

The above example shows that the DPELM algorithm can cope with the XOR problem with one single hidden neuron. It is worthwhile to note that, for single hidden layer feed forward neural network, BP algorithm requires at least two hidden neurons, RLS-RLM [12] requires at least three hidden neurons, and original ELM [8] or online-ELM requires at least four hidden neurons to solve the XOR problem.

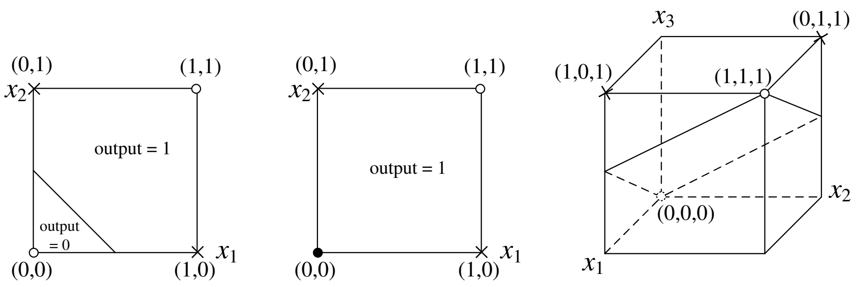

Figure 2 (a) Decision boundary constructed by hidden neuron. (b) Decision boundary constructed by output neuron. (c) Decision boundaries constructed by the complete network.

4. Online sequential double parallel ELM algorithm

For many real applications, batch training mode of DPELM where the entire data are required may results in overflow of memories. Sometimes, the data presented to the learning algorithm arrive one-by-one or chunk-by-chunk. Therefore the economical and reasonable way to train the samples is adopted by online method, in other words, one may utilize the data depending on their arrival time, or by partitioning the large dataset into small ones that are mutually disjoint.

In the original ELM algorithm, the input weight matrix $\mathbf{W}$ is first assigned random values, and then the output weight matrix $\mathbf{A}$ is determined by pseudo-inverse of hidden layer output matrix. While in RLS-ELM [12], the random matrix $\mathbf{C} \in \mathbb{R}^{c\times N}$ is first given, and by linear system

$$\mathbf{XW} = \mathbf{TC}, \tag{4.1}$$

the weight matrix $\mathbf{W}$ and $\mathbf{A}$ then can be solved, which achieves a compact network of small size. However, system (4.1) is a typical ill-posed problem, for which the Tikhonov regularization method is commonly used. In this way, (4.1) can be replaced by seeking $\mathbf{W}$ that minimizes

$$\|\mathbf{XW} - \mathbf{TC}\|^2 + \lambda\|\mathbf{W}\|^2, \tag{4.2}$$

where $\lambda > 0$ denotes the regularization parameter.

In order to solve (4.2), note that for any real matrix $\mathbf{A}$, we have

$$\|\mathbf{A}\|^2 = \operatorname{tr}\left(\mathbf{A}^\top\mathbf{A}\right), \tag{4.3}$$

where $\operatorname{tr}(\cdot)$ denotes the trace of a matrix. Now let $f(\mathbf{W}) = \|\mathbf{XW} - \mathbf{TC}\|^2 + \lambda\|\mathbf{W}\|^2$, by using (4.3), we arrive at

$$f(\mathbf{W}) = \operatorname{tr}\big\{\mathbf{W}^\top(\mathbf{X}^\top\mathbf{X})\mathbf{W} - (\mathbf{TC}^\top)\mathbf{XW} - \mathbf{W}^\top\mathbf{X}^\top\mathbf{TC} + \lambda\mathbf{W}^\top\mathbf{W}\big\}. \tag{4.4}$$

It is easy to prove, by the theories of matrix calculus, that the following properties hold for the matrix $\mathbf{A}$ and $\mathbf{B}$ with proper order,

$$\frac{\partial}{\partial\mathbf{A}}\operatorname{tr}(\mathbf{BA}) = \mathbf{B}^\top,\quad \frac{\partial}{\partial\mathbf{A}}\operatorname{tr}\left(\mathbf{A}^\top\mathbf{B}\right) = \mathbf{B},\quad \frac{\partial}{\partial\mathbf{A}}\operatorname{tr}\left(\mathbf{A}^\top\mathbf{A}\right) = 2\mathbf{A}, \tag{4.5}$$

and

$$\frac{\partial}{\partial\mathbf{A}}\operatorname{tr}\left(\mathbf{A}^\top\mathbf{BA}\right) = \mathbf{BA} + \mathbf{B}^\top\mathbf{A}. \tag{4.6}$$

By using (4.5) and (4.6), differentiating both sides of (4.4) with respect to $\mathbf{W}$ gives

$$\frac{\partial f}{\partial\mathbf{W}} = 2\left(\mathbf{X}^\top\mathbf{XW} + \lambda\mathbf{W} - \mathbf{X}^\top\mathbf{TC}\right). \tag{4.7}$$

Let $\frac{\partial f}{\partial\mathbf{W}} = 0$. We obtain $\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}\right)\mathbf{W} = \mathbf{X}^\top\mathbf{TC}$. For each $\lambda > 0$, $\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}\right)$ is positive definite, and therefore

$$\mathbf{W} = \left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^\top\mathbf{TC} \tag{4.8}$$

Let the initial training subset be $S_0 = \big\{(\mathbf{x}^{(j)}, \mathbf{t}^{(j)}) | j = 1, \ldots, n_0\big\}$. Substituting n_0 for n in (2.1) yields

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}^{(1)} & \ldots & \mathbf{x}^{(n_0)} \\ 1 & \ldots & 1 \end{pmatrix}^\top, \quad \mathbf{T} = (\mathbf{t}_1, \ldots, \mathbf{t}_{n_0})^\top. \tag{4.9}$$

Since the matrix $\mathbf{C}$ in (4.8) is independent of the arriving data, the initial weights (including hidden layer biases) are obtained as follows

$$\mathbf{W}_{(0)} = \left(\mathbf{X}_0^\top \mathbf{X}_0 + \lambda \mathbf{I}\right)^{-1} \mathbf{X}_0^\top \mathbf{T}_0 \mathbf{C}. \tag{4.10}$$

Set $\mathbf{L}_{(0)} = \mathbf{W}_{(0)} = \left(\mathbf{X}_0^\top \mathbf{X}_0 + \lambda \mathbf{I}\right)$, and equation (4.10) can be rewritten as

$$\mathbf{W}_{(0)} = \mathbf{L}_{(0)}^{-1} \mathbf{X}_0^\top \mathbf{T}_0 \mathbf{C}. \tag{4.11}$$

Let $\mathbf{H}_0 = g\left(\mathbf{X}_0 \mathbf{W}_{(0)}\right)$ and $\mathbf{A}_{(0)} = \left(\mathbf{U}_{(0)}, \mathbf{V}_{(0)}\right)^\top$. Applying **algorithm 1** yields

$$\mathbf{A}_{(0)} = \left\{ \left(\mathbf{H}_0, \mathbf{P}_0\right)^\top \left(\mathbf{H}_0, \mathbf{P}_0\right) \right\}^{-1} \left(\mathbf{H}_0, \mathbf{P}_0\right)^\top \mathbf{T}_0, \tag{4.12}$$

where the definition of $\mathbf{P}_0$ is shown in (2.2). If we set $\mathbf{Q}_{(0)} = \{(\mathbf{H}_0, \mathbf{P}_0)^\top (\mathbf{H}_0, \mathbf{P}_0)\}^{-1}$, the existence of initial $\mathbf{Q}_{(0)}$ is guaranteed for the size of n_0 is adjustable. Now (4.12) becomes

$$\mathbf{A}_{(0)} = \mathbf{Q}_{(0)} \left(\mathbf{H}_0, \mathbf{P}_0\right)^\top \mathbf{T}_0, \tag{4.13}$$

For n_1 observations of the next training subset $S_1 = \left\{(\mathbf{x}^{(j)}, \mathbf{t}^{(j)}) | j = n_0 + 1, \ldots, n_0 + n_1\right\}$, or in general, for the k-th training subset consisting of n_k patterns

$$S_k = \left\{(\mathbf{x}^{(j)}, \mathbf{t}^{(j)}) | j = \sum_{i=0}^{k-1} n_i + 1, \ldots, \sum_{i=0}^{k-1} n_i + n_k\right\}, \tag{4.14}$$

online updating the output weights based on recursive least-squares solution is given in [14], that is:

$$\mathbf{L}_{(k)} = \mathbf{L}_{(k-1)} + \mathbf{X}_k^\top \mathbf{X}_k, \tag{4.15}$$

and

$$\mathbf{W}_{(k)} = \mathbf{W}_{(k-1)} + \mathbf{L}_{(k)}^{-1} \mathbf{X}_k^\top \left(\mathbf{T}_k \mathbf{C} - \mathbf{X}_k \mathbf{W}_{(k-1)}\right). \tag{4.16}$$

For double parallel network structure described in Figure 1(a), one has to take the input matrix $\mathbf{P}_k$ into account when determining the output weights matrix. According to [13], substituting $(\mathbf{H}_k, \mathbf{P}_k)$ for the hidden weights output matrix $\mathbf{H}_k$ leads to

$$\mathbf{Q}_{(k)} = \mathbf{Q}_{(k-1)} - \mathbf{Q}_{(k-1)} \left(\mathbf{H}_k, \mathbf{P}_k\right)^\top \left\{\mathbf{I} + \left(\mathbf{H}_k, \mathbf{P}_k\right) \mathbf{Q}_{(k-1)} \left(\mathbf{H}_k, \mathbf{P}_k\right)^\top \right\}^{-1} \left(\mathbf{H}_k, \mathbf{P}_k\right) \mathbf{Q}_{(k-1)}, \tag{4.17}$$

and

$$\mathbf{A}_{(k)} = \mathbf{A}_{(k-1)} + \mathbf{Q}_{(k)} \left(\mathbf{H}_k, \mathbf{P}_k\right) \left\{\mathbf{T}_k - \left(\mathbf{H}_k, \mathbf{P}_k\right) \mathbf{A}_{(k-1)}\right\}. \tag{4.18}$$

In summary, given the training set $S = \{(\mathbf{x}^{(j)}, \mathbf{t}^{(j)}) | j = 1, \ldots, n\}$, the activation function $g(\cdot)$ of hidden layer, and the number of hidden units, the online sequential DPELM (OS-DPELM) scheme consists of two phases, namely an initialization phase (IP) and a sequential learning phase (LP), which is stated as in Algorithm 2.

Algorithm 2 Online Sequential DPELM Algorithm

```
IP. Initialization Phase for the first training subset $\mathbf{S}_0$.
Step 1. Assign random values uniformly from interval $(-1,1)$ for matrix $\mathbf{C}$.
Step 2. Calculate the initial input weights and biases $\mathbf{W}_{(0)}$ by (4.10) or (4.11).
Step 3. Determine the initial output weight matrix
        $\mathbf{A}_{(0)} = (\mathbf{U}_{(0)}, \mathbf{V}_{(0)})$ by (4.12) or (4.13).
LP. Learning Phase for the $k$-th ($k \geq 1$) training subset $S_k$ consisting of $n_k$ samples.
Step 4. Calculate the $k$-th input weights matrix $\mathbf{W}_{(k)}$ by (4.15) and (4.16).
Step 5. Compute the hidden output weight matrix $\mathbf{H}_k = g(\mathbf{H}_k \mathbf{W}_{(k)})$.
Step 6. Determine the $k$-th output weight matrix $\mathbf{A}_{(k)}$ by (4.17) and (4.18).
Step 7. Set $k = k + 1$ and repeat until all the training subsets are used for only one time.
```

Remark 4.1 Different from conventional online algorithms (e.g., online BP algorithm), OS-DPELM makes use of each sample once but not cyclically in the training set. Like online ELM or online RLS-ELM, the training examples can be used one-by-one or chunk-by-chunk. Once all the training data is included in the initialization phase, the online DPELM then becomes batch DPELM. Thus, batch DPELM can be considered as a special case of online DPELM.

5. Experimental results

Classification problems are very common in real world applications. For examples, it is necessary to diagnose whether a patient's tumor cell is malignant or not, or distinguish which kind of mushroom is edible in expert system. In order to verify the performance of online double parallel extreme learning machine, six data sets for classification are taken from UCI machine learning repository, and the learning and testing results will be compared with those of online ELM and online RLS-ELM.

All the simulations are carried out in MATLAB 7.8 environment running in Core2Duo, 2.53 GHZ CPU with 2GB RAM. For each dataset, one third of the whole data is used for training while the remaining data for blind test. The input feature values are normalized into the range $[-1,1]$ in case of inconsistency. The activation function adopted in the hidden layer of the network is the form of sigmoid. The number of hidden units is gradually increased from only one neuron and the nearly optimal number of units was chosen based on the best generalization performance. For every data set, we run 50 trials and treat the average values as the final results. In our experiments, to ensure the existence of $\mathbf{Q}_{(0)}$, we generally choose the number of initial dataset $n_0 \geq N$ (the number of hidden units), and thus the existence of $\mathbf{Q}_{(k)}$ is also guaranteed. Except the initialization process, every 10 samples are used in one learning iteration.

It can be seen from Table 2 that, to achieve similar or better performance, our proposed approach needs fewer hidden units than those of OS-ELM or OS-RLS-ELM. At the same time, we get a compact network of small size, which means the parameters to be adjusted are highly reduced. The regularization method used in OS-DPELM also gives rise to the promotion of the generalization performance according to Barlett's results [11]. Furthermore, our proposed method spends less time on training process than that of the other two methods.

Dataset	Specifications	Online methods	# hidden units	Training times(s)	Train Acc(%)	Test Acc(%)
Diabetes	# Features:8	DPELM	2	0.0042	76.63	78.13
	# Training:512	RLS-ELM	4	0.0048	76.39	77.73
	# Testing:256	ELM	20	0.0128	78.02	77.34
MUSK	# Features:166	DPELM	2	0.1305	95.36	**94.12**
	# Training:4398	RLS-ELM	10	0.1868	94.95	93.37
	# Testing:2199	ELM	160	1.3581	94.38	93.42
Australian Credit	# Features:6	DPELM	2	0.0036	76.11	**74.43**
	# Training:460	RLS-ELM	4	0.0038	77.02	72.87
	# Testing:230	ELM	16	0.0112	76.52	73.30
Liver	# Features:6	DPELM	2	0.0031	73.67	72.88
	# Training:230	RLS-ELM	8	0.0038	74.22	72.46
	# Testing:115	ELM	16	0.0052	74.35	72.17
Heart	# Features:10	DPELM	1	0.0012	86.83	**86.33**
	# Training:455	RLS-ELM	4	0.0012	85.68	84.44
	# Testing:228	ELM	16	0.0031	85.61	83.67
Breast Cancer	# Features:13	DPELM	2	0.0031	96.12	**98.68**
	# Training:180	RLS-ELM	4	0.0047	95.18	97.55
	# Testing:90	ELM	16	0.0094	95.61	97.36

Table 2 Comparisons of online sequential DPELM, ELM and RLS-ELM

Remark 5.1 The value of regularization parameter λ in our simulations is not sensitive to the results. Usually the positive parameter λ lies between 0 and 1, so we may choose a very small value such as $\lambda = 10^{-4}$ for it. Besides, we must point out that regularization parameter λ can smooth the peak of a function, which results in the ability of function approximation of both DPELM and RLS-ELM is no better than that of ELM. That is the reason why the approximation problems are excluded from our numerical experiments.

6. Conclusion

ELM is an easy but efficient learning mechanism for the generalized SLFNs. However, this algorithm often requires a large number of hidden units and thus slowly responds to new observations. Although RLS-ELM was proposed to overcome the problem, it did not take the direct influence of input patterns on the network outputs into account. In this paper, an online sequential DPELM scheme is proposed based on double parallel network structure, in which the linear component is involved and consequently the number of hidden units is greatly reduced.

Thus the proposed method can achieve good generalization performance with high speed for both learning and testing processes.

References

[1] O. K. ERSOY, D. HONG. *Parallel, self-organizing, hierarchical neural networks.* IEEE Trans. Neural Netw., 1990, **1**(2): 167–178.

[2] Rui HUANG, Mingyi HE. *Feature selection using double parallel feedforward neural networks and particle swarm optimization.* IEEE Congress on Evolutionary Computation, CEC2007, 692–696.

[3] Jian WANG, Wei WU, Zhengxue LI, et al. *Convergence of gradient method for Double parallel feedforward neural network.* Int. J. Numer. Anal. Mod., 2011, **8**(3): 484–495.

[4] M. T. HAGAN, H. B. DEMUTH, M. H. BEALE, et al. *Neural Network Design.* Pws Pub., Boston, 1996.

[5] Mingchen YAO, Wenting LI, Yan LIU. *Double parallel extreme learning machine.* Energy Proc., 2011, **13**: 7413–7418.

[6] D. E. RUMELHART, G. E. HINTON, R. J. WILLIAMS. *Learning representations by back-progagating errors.* Nature, 1986, **323**: 533–536.

[7] Wei WU, Guorui FENG, Zhengxue LI, et al. *Deterministic convergence of an online gradient method for BP neural networks.* IEEE Trans. Neural Netw., 2005, **16**(3): 533–540.

[8] Guangbin HUANG, Qinyu ZHU, C. K. SIEW. *Extreme learning machine: theory and applications.* Neurocomputing, 2011, **70**(1): 489–501.

[9] Guangbin HUANG, Dianhui WANG, Yuan LAN. *Extreme learning machines: A survey.* International Journal of Machine Leaning and Cybernetics, 2011, **2**(2): 107–122.

[10] Guangbin HUANG, Hongming ZHOU, Xiaojian DING, et al. *Extreme learning machine for regression and multiclass classification.* IEEE T. Syst. Man Cy. B., 2012, **42**(2): 513–529.

[11] P. L. BARLETT. *The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network.* IEEE T. Inform. Theory, 1998, **44**(2): 525–536.

[12] H. T. HUYNH, Y. WON, J. J. KIM. *An improvement of extreme learning machine for compact single-hidden-layer feedforward neural networks.* Int. J. Neural Syst., 2008, **18**(05): 433–441.

[13] Nanying LIANG, Guangbin HUANG, P. SARATCHANDRAN, et al. *A fast and accurate on-line sequential learning algorithm for feedforward networks.* IEEE Trans. Neural Netw., 2006, **17**(6): 1411–1423.

[14] H. T. HUYNH, Y. WON. *Online training for single hidden-layer feedforward neural networks using RLS-ELM.* IEEE International Symposium on Computational Intelligence in Robotics and Automation, 2009, (CIRA2009): 469–473.

[15] Y. A. LECUN, L. BOTTOU, G. B. ORR, et al. *Neural Networks: Tricks of the Trade.* Springer, 2012.

[16] V. S. ASIRVADAM, S. F. MCLOONE, G. W. IRWIN. *Parallel and separable recursive Levenberg-Marquardt training algorithm.* in Proc. 12th IEEE Workshop Neural Netw. Signal Process, 2002, **4**(6): 129–138.

[17] T. M. COVER. *Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition.* IEEE Transactions on Electronic Computers, 1965, **EC-14**(3): 326–334.