SPC

Journal of Advanced Computer Science & Technology, 3 (1) (2014) 18-27
©Science Publishing Corporation
www.sciencepubco.com/index.php/JACST
doi: 10.14419/jacst.v3i1.1753
Research Paper

VSS SPU-EBP: Variable step size sequential partial update error back propagation algorithm

Maryam Rahmaninia

Department of Computer Engineering, Ghasre-shirin Branch, Islamic Azad University, Iran
**Corresponding author E-mail: ma.rahmaninia@gmail.com*

Abstract

In MLP networks with hundreds of thousands of weights which must be trained on millions of samples, the time and space complexity may become greatly large and sometimes the training of network by EBP algorithm may be impractical.
Sequential Partial Updating is an effective method to reduce computational load and power consumption in implementation. This new approach is very useful for the MLP networks with large number of weights in each layer that updating of each weight in each round of execution of EBP algorithm will be costly. Although this idea reduces computational cost and elapsed CPU time in each round but sometimes maybe increases number of epochs required to convergence and this leads to increase time of convergence. That is, to speed up more the convergence rate of the SPU−EBP algorithm, we propose a Variable Step Size (VSS) approach. In VSS SPU−EBP algorithm, we use a gradient based learning rate in SPU-EBP algorithm to speed up the convergence of training algorithm. In this method we derive an upper bound for the step size of SPU_EBP algorithm.

Keywords: *Neural Network, Error Back Propagation, MLP (Multi-Layered Perceptron), Sequential Partial Update Algorithm.*

1. Introduction

It is well known that the Multi-Layered Perceptron (MLP) artificial neural network is a nonparametric model for accomplishing a broad variety of prediction tasks in many areas of science and engineering. Gradient-based algorithms, particularly Error Back Propagation (EBP) algorithm [1] are well-known as a type of supervised learning algorithms which widely used for training the weights of MLP neural networks. Generally, the training process in EBP algorithm is done through iterative renewing of weights according to the error signal, which is the negative gradient of the mean-squared error (MSE) function. The error signal of each neuron in the output layer is defined as a difference between desired target value and the actual output value of the neuron multiplied by the gradient of its activation function. Then the error signal is back propagated to the lower layer.
The performance of MLP trained with EBP algorithm depends on several factors, including initial values, choice of learning parameters network topology, train set etc. [2], [3]. In order to overcome these drawbacks, early researches have been focused on estimation of the optimal initial values to speed up the convergence rate [4], [5], [6], and [7]. If the dynamic range of the activation function is entirely used, then this approach guarantees that the outputs of the hidden layers and output layer will be within the active region [8] [9]. Consequently, it can efficiently speed up the convergence of the EBP algorithm.
Caruana in [11] presented a tutorial at NIPS 93 with generalization results on a variety of problems as the size of the networks was varied from “too small” to “too large”. “Too small” and “too large” are related to the number of parameters in the model (without consideration of the distribution of the data, the error surface, etc.). Caruana reported that large networks rarely do worse than small networks on the problems he investigated. The results in this paper partially correlate with that observation. Caruana suggested that “back propagation ignores excess parameters”.
Crane, Fefferman, Markel and Pearson in [10] used real-valued data generated by a random target network, and attempted training new networks on the data in order to approximate the number of minima on the error surface under

varying conditions. The use of random target networks in this fashion has been referred to as the student teacher problem (Saad and Solla, 1995).
In order to find an optimal network topology for MLP neural networks, numerous approaches have been put forward in literatures, which make it depend on the size of the train set or the number of neurons in input and output layers. [11].
Despite recent improvements, the EBP algorithm for training MLP with large-scale data has still very challenging task. Particularly, in areas such as video classification, where the training set is huge and MLPs with hundreds of thousand weights must be trained on millions of samples, the time and space complexity may become greatly large and sometimes the training of network may be impractical. In this paper we present two generalizations of EBP algorithm to overcome these problems. The first is Sequential Partial Updating method. Partial updating is an effective method that reducing the computational load and power consumption in adaptive LMS filters design [12] [13]. Although this idea reduces the computational complexity, but may increases the time of convergence. Then to speed up more the time of convergence of the proposed Sequential Partial Update EBP (SPU-EBP) algorithm we propose secondary method. In VSS SPU-EBP algorithm, we use a gradient based learning rate to speed up more the convergence of SPU-EBP training algorithm. The maximum step size is expressed in terms of the gain in the step size parameter of the SPU-EBP algorithm, defined as the ratio between the upper bounds that ensure convergence when only a subset of the weights of the filter is updated every iteration. The convergence of these algorithms is proven mathematically. Also, the experiments reported at the end of paper obviously exhibit that VSS SPU-EBP yield very efficient (computational cost and space) and acceptable convergence time. The main contribution of this paper can be summarized as follow:
Section 2 describes the standard EBP algorithm. Section 3 presents the proposed SPU_EBP and VSS SPU-EBP algorithms. In Section 4, we present some metrics to evaluate the efficiently of the proposed algorithms. Experiential results of VSS SPU-EBP and SPU-EBP algorithms on the Iris data set, Oil price data set and Video classification are presented in Section 5. Section 6 presents our conclusions and considers some ideas that will improve the performance of our current algorithms.

2. Review of standard EBP algorithm

In this section we review the original EBP algorithm [1]. Without loss of generality, we consider a special MLP neural network with three layers which has m neuron(s) in input layer, L neuron(s) in hidden layer, and one neuron in output layer. Assume that $W^0 = (w_1^0, w_2^0, ...w_L^0)$ is the weight vector between all the hidden neuron(s) and the one output neuron, and $W^i = (w_1^i, w_2^i, ...w_m^i)$ is the weight vector between all the input neuron(s) and the i[th] hidden neuron for i=1,...,L. For compact representation of weight parameters we denote

$W = (W^0, W^1, ...W^L)$,

$V = (W^1, ..., W^L)^T$,

And for each sample of data set $x = (x_1, ..., x_m)^T$ we define vector function as

$\Phi(V.x) = (\varphi(\mathrm{W}^1.x), ..., \varphi(\mathrm{W}^L.x))$.

Let $\{x^j, d_j\}_{j=1}^Z$ be a set of training samples and $\varphi: R \to R$ be an activation function for both hidden and output layers. The actual output for each input $x^j \in R^m$ for $j = 1, ..., Z$ will be computed as follows:

$$y_j^0 = \varphi(W^0.\Phi(V.x^j)). \quad (1)$$

Where $\Phi(V.x)$ is a vector function whose transfer function in each dimension is the activation function of MLP.
The training error of the neuron in the output layer is defined as:

$$e_j = d_j - y_j^0 \quad (2)$$

And, the total training error of network is defined as:

$$E(W) = \frac{1}{2}(e_j)^2 = \Psi_j[W^0.\Phi(V.x^j)]. \quad (3)$$

Where $\Psi_j[t] = \frac{1}{2}(d_j - \varphi(t))^2$.

The EBP learning algorithm is designed to change the current weights W iteratively such that the system error function E (W) is minimized. The renewing of weights is made proportional to the partial derivation of E (W). The details are given in the following.
By differentiating E (W) with respect to W, we will get:

$$E_W(W) = (E_{W^0}(W), (E_{W^1}(W), ..., (E_{W^L}(W)) \quad (4)$$

Where:

$$E_{W^i}(W)=\begin{cases}\Psi'_j[W^0.\Phi(V.x^j)]\Phi(V.x^j) & i=0\\ \Psi'_j[W^0.\Phi(V.x^j)]W^0\Phi'(V.x^j)x^j & i=1,...,L\end{cases} \quad (5)$$

Now, if we consider $W^{(n)}=(W^{0(n)},W^{1(n)},...W^{L(n)})$, is the value of W in round n then by starting with an arbitrary initial value for W(0), we will be able to update the weights $\{W^{(n)}\}$ iteratively as follows:

$$W^{i(n+1)}=W^{i(n)}+\Delta W^{i(n)} \quad i=0, 1\ldots L, \quad n=0, 1\ \ldots \quad (6)$$

Where:

$$\Delta W^{0(n+1)}=-\mu E_{W0}(W^{(n)})\Phi(V^{(n)}.x^j) \quad (7)$$

And,

$$\Delta W^{i(n+1)}=-\mu E_{W^i}(W^{(n)})\ x^j, \qquad i=1,...,L \quad (8)$$

Here, μ is the learning rate of algorithm which is a positive real number that determines the step size of algorithm and has an effect on the speed of the convergence rate.

3. The proposed VSS SPU-EBP algorithm

In this section we will present the proposed VSS SPU-EBP algorithm. At first, we focus on SPU-EBP algorithm [14], which at each epoch choices a set of weight parameters in a sequential manner and update the weight using standard EBP algorithm. After that, a variable step size version of SPU-EBP algorithm is studied. The proposed VSS SPU-EBP algorithm will be achieved by integrating these two approaches. Details of SPU-EBP and VSS SPU-EBP algorithms are presented in the following subsections.

3.1. The SPU-EBP algorithm

Consider a full connected MLP neural network with k layer(s), where the number of neurons in layer h is N_h , for h=1… k. For given p assume that the number of weights for each neuron, in each layer is a multiple of p≤ $\{N_1,\ldots,N_h\}$. For convenience, we define the index set $S_{h,l}=\{1,2,\ldots,N_{h-1}\}$ for each neuron l in layer h.

Now, for each S_{hl} in layer h, we divide the index set $S_{h,l}$ into p mutually exclusive subsets $S_1^{h,l},S_2^{h,l},...,S_p^{h,l}$. Let I be an identity matrix. In round n of execution of algorithm, if we choose t^{th} subset of index set in layer h, we define the matrix $I_t^{(h)}$ by zeroing out the l^{th} row of $I_{N_{h-1}\times N_{h-1}}$ if $l\notin S_t^{h,l}$. In this case, for l^{th} neuron in layer h, where the input vector for this neuron is Y^l, $I_t^{(h)}Y^l$ will have at most N_l/p nonzero entries. Let the sentence "choosing $S_t^{h,l}$ at round n" stand to mean "choosing the weights that their indices is in $S_t^{h,l}$ for update at round n". Then, we describe SPU_EBP algorithm as follows.

At a given round n, for each neuron l in layer h, one of the sets $S_t^{h,l}$, is selected in a predefined manner and the update is performed. Without loss of generality, it can be assumed that at iteration n in layer h, for each neuron l, the set $S^{h,l}_{n\%p+1}$ is chosen to be updated, where n%p indicates the operation "n modulo p". Hence, the renewing of weights in the proposed SPU-EBP algorithm will be as follows

$$W^{i(n+1)}=W^{i(n)}-\mu E_{Wi}(W^{(n)})\ I^h_{n\%p+1}\ Y^i \qquad i=0\ldots L. \quad (9)$$

Y^i is an input vector and $E_{Wi}(W^{(n)})$ is the error value for i^{th} neuron in iteration n. For simplicity, according to Section 2, we consider a special MLP neural network with three layers which has m neuron(s) in input layer, L neuron(s) in hidden layer, and one neuron in output layer. Also, we assume a special case of SPU-EBP algorithm with p=2. We consider the sets $S_1^{1,l}$ and $S_2^{1,l}$ which consist of all even and odd weights for neuron l that are in hidden layer(layer 1) respectively and the sets $S_1^{2,l}$ and $S_2^{2,l}$ consist of all even and odd weights for neuron l in output

layer(layer 2) respectively. Now, according to the Equation (9), if at nth iteration the algorithm selects even weights, for an arbitrary input vector x in the hidden layer, we have:

$$W^{i(n+1)} = W^{i(n)} - \mu E_{W^i}(W^{(n)})\, I_1^{(1)}\, x^j \quad , \quad i=1,\ldots,L, \tag{10}$$

And, if it selects odd weights, then we have:

$$W^{i(n+1)} = W^{i(n)} - \mu E_{W^i}(W^{(n)})\, I_2^{(1)}\, x^j \quad ,$$

$$i=1\ldots L \tag{11}$$

Also, in output layer, if the algorithm selects even weights in this iteration, then we obtain:
And, if it selects odd weights, then we obtain:

$$W^{0(n+1)} = W^{0(n)} - \mu E_{W^0}(W^{(n)})\, I_2^{(2)}\, \Phi(V.x^j) \quad , \tag{12}$$

$$W^{0(n+1)} = W^{0(n)} - \mu E_{W^0}(W^{(n)})\, I_1^{(2)}\, \Phi(V.x^j) \quad , \tag{13}$$

The following theorem discusses about the convergence of this algorithm.

Theorem 1: *Let $\varphi : R \to R$ be an activation function and E (W) being the error function defined in Equation (3). For this network If μ is satisfactory small and for each $x \in R^m$, $|\varphi(x)|$, $|\varphi'(x)|$ and are uniformly bounded, then for SPU-EBP algorithm the following results are hold:*

$E(W^{(n+2)}) < E(W^{(n)})$ When n is sufficiently large;

There exists a nonnegative real number $\varepsilon^* \geq 0$, such that:

$\lim_{n \to \infty} E(W^{(n)}) = \varepsilon^*$;

$\lim_{n \to \infty} E(W^{(n)}) = 0$;

Furthermore, we have the strong convergence:

There exists $W^{0^*} \in \phi_0$ such that $\lim_{n \to \infty} W^{0(n)} = W^{0(*)}$.

As we will see in the experiments, this algorithm reduces the computational load and space complexity in each epoch but may increase the time of convergence.

Proof: See [14].

As we will see in the experiments, this algorithm reduces the computational cost and space complexity in each iteration but may increases the number of rounds until convergence and this leads to increase time of convergence. In the next subsection we propose a variable step approach to speed up the convergence of this algorithm by decreasing number of rounds.

3.2. The VSS SPU-EBP algorithm

The results of the previous subsections can be combined to create the VSS SPU-EBP training algorithm that has low convergence time and reduces the computational and space complexity in each epoch. The detail of this algorithm is summarized as follows.

Consider a MLP neural network with k layer(s), where the number of neurons in in layer h is N_h , for h=1… k. Also, let the number of weights in each layer be a multiple of p. In order to partial updating, the weights of each layer are partitioned into p subsets and the weighs of each subset in all layers are updated simultaneously. This renewing of weights is done for the other subsets sequentially. In this case we define a stage includes onetime updating of all weights of MLP network. In this algorithm the learning rate is updated at the end of each stage using Equation (14). Consequently, the algorithm updates the weights partially with a variable step size learning rate parameter.

Similar to Section 2 let we consider a special MLP neural network with three layers which has m neuron(s) in input layer, L neuron(s) in hidden layer, and one neuron in output layer. As mentioned before, an Estimated Posteriori Error is used to compute learning rate adaptively. This modification is being demonstrated in the following equation:

$$W^{i(n+1)} = W^{i(n)} + [-\mu_n^* E_{w^i}(W^{i(n)}) I^{(h)}_{n\%p}\, x^j] \quad , \tag{14}$$

μ_n^* , Is computed as follows:

$$\mu_n^* = \frac{2}{E(I_t^{(h)} x^j)} \tag{15}$$

Where, x^j is input vector for nth iteration.

As above, we have

For simplicity we denote E_{rn} instead of $E_{w^i}(W^{i(n)})$ and this value can be calculated as follow

$$Er_n = W^{i(n)} - W^i_{opt} \quad , i=0, 1, 2\ldots L.$$

This leads to the following coefficient error vector update for this network,

When n is even

$$Er_{n+1} = \begin{cases} (I - \mu^*_n I^{(1)}_1 x^j) Er_n \\ (I - \mu^*_n I^{(2)}_1 \Phi(\mathrm{V.x^j})) Er_n \end{cases}$$

And the following when n is odd:

$$Er_{n+1} = \begin{cases} (I - \mu^*_n I^{(1)}_2 x^j) Er_n \\ (I - \mu^*_n I^{(2)}_2 \Phi(\mathrm{V.x^j})) Er_n \end{cases}$$

Now, we analyze the convergence of the mean coefficient error vector $E[Er_{n+1}]$. We make the standard assumption that Er_n and x^j are independent of each other. For this algorithm, for any layers, the recursion for $E\ [Er_{n+1}]$ is given by

$$Er\ [Er_{n+1}] = (I - \mu R) E\ [Er_n] \tag{16}$$

Where I am the N-dimensional identity matrix and $R = E[I^h_k Y^i]$ is the input signal correlation matrix for those output and input layers. The well-known necessary and sufficient condition for to converge is given by

$$\rho(I - \mu R) < 1 \tag{17}$$

Where $\rho(B)$ denote the spectral radius of B ($\rho(B) = \max |\lambda_i(B)|$) . This leads to

$$0 < \mu < \frac{2}{\lambda_{\max}(R)} \tag{18}$$

Where is the maximum eigen-value of the input signal correlation matrix R. Note that this need not translate to be the necessary and sufficient condition for the convergence assumption which is not true in general. Taking expectation under the same assumption as above and using the independence assumption on the sequence x^j, which is the independence assumption on xj and Er_n , we obtain, when n is even

$$Er_{n+2} = \begin{cases} (I - \mu^*_{n+1} I^{(1)}_2 x^j) Er_{n+1} \\ (I - \mu^*_{n+1} I^{(2)}_2 \Phi(\mathrm{V.x^j})) Er_{n+1} \end{cases}$$

And when n is odd

$$Er_{n+2} = \begin{cases} (I - \mu^*_{n+1} I^{(1)}_1 x^j) Er_{n+1} \\ (I - \mu^*_{n+1} I^{(2)}_1 \Phi(\mathrm{V.x^j})) Er_{n+1} \end{cases}$$

Simplifying the above two sets of equations, we obtain, for even-odd VSS SPU-EBP when n is even

$$E[Er_{n+2}] = \begin{cases} (I - \mu^*_{n+1} I^{(1)}_2 x^j)(I - \mu^*_n I^{(1)}_1 x^j) E[Er_n] \\ (I - \mu^*_{n+1} I^{(2)}_2 \Phi(\mathrm{V.x^j}))(I - \mu^*_n I^{(2)}_1 \Phi(\mathrm{V.x^j})) E[Er_n] \end{cases} \tag{19}$$

And when n is odd

$$E[Er_{n+2}] = \begin{cases} (I - \mu^*_{n+1} I^{(1)}_1 x^j)(I - \mu^*_n I^{(1)}_2 x^j) E[Er_n] \\ (I - \mu^*_{n+1} I^{(2)}_1 \Phi(\mathrm{V.x^j}))(I - \mu^*_n I^{(2)}_2 \Phi(\mathrm{V.x^j})) E[Er_n] \end{cases} \tag{20}$$

It can be shown that under the above assumptions on x^j , Er_n , the convergence conditions for even and odd update equations are identical. We therefore focus on the odd type.

As you know, the necessary condition for convergence of $E[Er_{n+2}]$ is

$$\begin{cases} \rho((I - \mu^*_{n+1} I^{(1)}_1 x^j)(I - \mu^*_n I^{(1)}_2 x^j)) < 1, \\ \rho((I - \mu^*_{n+1} I^{(2)}_1 \Phi(\mathrm{V.x^j}))(I - \mu^*_n I^{(2)}_2 \Phi(\mathrm{V.x^j}))) < 1 \end{cases}$$

That, this leads

$$\begin{cases} \rho(I-\mu_{n+1}^{*}I_1^{(1)}x^j)<1, \quad \rho(I-\mu_{n+1}^{*}I_1^{(2)}\Phi(\mathrm{V}.\mathrm{x}^j))<1, \\ \rho(I-\mu_{n}^{*}I_2^{(1)}x^j)<1, \quad \rho(I-\mu_{n}^{*}I_2^{(2)}\Phi(\mathrm{V}.\mathrm{x}^j))<1 \end{cases}$$

Then, we have

$$0<\mu_{n+1}^{*}<\frac{2}{E[I_1^{(1)}x^j]} \text{ Or } 0<\mu_{n+1}^{*}<\frac{2}{E[I_1^{(2)}\Phi(\mathrm{V}.\mathrm{x}^j)]}$$

$$0<\mu_{n}^{*}<\frac{2}{E[I_2^{(1)}x^j]} \text{ Or } 0<\mu_{n}^{*}<\frac{2}{E[I_2^{(2)}\Phi(\mathrm{V}.\mathrm{x}^j)]}$$

So, $\mu_n^{*}=\min(\frac{2}{E[I_2^{(1)}x^j]},\frac{2}{E[I_2^{(2)}\Phi(\mathrm{V}.\mathrm{x}^j)]})$ and

So $\mu_{n+1}^{*}=\min(\frac{2}{E[I_1^{(1)}x^j]},\frac{2}{E[I_1^{(2)}\Phi(\mathrm{V}.\mathrm{x}^j)]})$

That this leads

$$\mu_n^{*}<\frac{2}{E[I_1^{(1)}x^j]},\ \mu_{n+1}^{*}<\frac{2}{E[I_2^{(1)}x^j]}$$

Now, if instead of just two partitions of odd and even coefficients (p=2), there are any number of arbitrary partitions (p≥2), and then, the update equations can be similarly written as above, whit p<2. Basely

$$E(Er_{n+p})=\prod_{i=1}^{p}(I-\mu_{n+i}^{*}I_{(i+n)\%p+1}^{h}Y^{i+n})E[Er_n]$$

That this leads

$$\forall n\geq 0, \mu_n^{*}<\frac{2}{E[I_1^{(1)}x^j]} \tag{21}$$

4. Evaluation metrics

To evaluate the proposed algorithms, we use four different metrics. These metrics are listed in bellows.

Number of Rounds (NR): A round is defined as one unit of an algorithm progress in which, one sample of data set is applied to MLP neural network for training and performs any required computation in all layers, and it sends back its error value to MLP. The number of rounds until MLP neural network convergence to MSE is shown by NR.

Computational Cost (CC): In each round every node compute output value for itself and forward to the next layer. This metric is the amount of computation that is executed in every round. This parameter can be calculated as follow

CC= t * Number of nodes.

Where, t is the amount of computation that is done in a node.

Elapsed CPU time (ECT): This parameter shows time requirement until a round of algorithm is executed.

Time of Convergence: This is the time that requires for reaching MLP neural network to considered MSE. We can obtain this parameter by the equation below:

Time of convergence=ECT * NR.

The main contribution of this paper is to reduce the computational cost (CC), Elapsed CPU Time (ECT), NR and time of convergence. This metrics are considered to prove that VSS SPU-EBP algorithm decrease ECT, CC, NR and Time of Convergence.

5. Experimental results

In order to demonstrate the efficiency of the proposed algorithms empirically, we organize three types of experiments. First, one well-known data set including Iris data set [15] has been used to satisfy the effectiveness of proposed algorithms on aspects of time of convergence, ECT, computation and space complexity. The second is Oil price data set [21]. This data set consists of 6000 data records reflecting the period between 2000 until 2006 with price of Oil in these years. Subsequently, we utilize the MLP neural network with VSS SPU-EBP algorithm as a classifier in a video classification system. The details of these experiments have been investigated in the following subsections. As you see these results confirm the efficiency of the VSS SPU-EBP algorithm in comparing to the SPU-EBP and standard EBP algorithms. According to the previous works, we utilize these metrics for comparing of the training algorithms.

5.1. Classifying iris data set

In this section, the performance of the proposed algorithms is shown on the Iris data set. The Iris data set is one of the best known databases in the pattern recognition literatures. The data set contains three classes. Each class has 50 instances, totally 150 patterns are used. All the values are normalized by dividing the value by 10. The method of [16] has been utilized to determine the topology of MLP neural network and SMI approach in [17] to initialize the weights. The structure has five input neurons including bias and one output neuron. We assume the minimum square error (MSE) is the stopping criterion in all training algorithms, and the maximum expected MSE is assumed to be 0.01 in all tests.

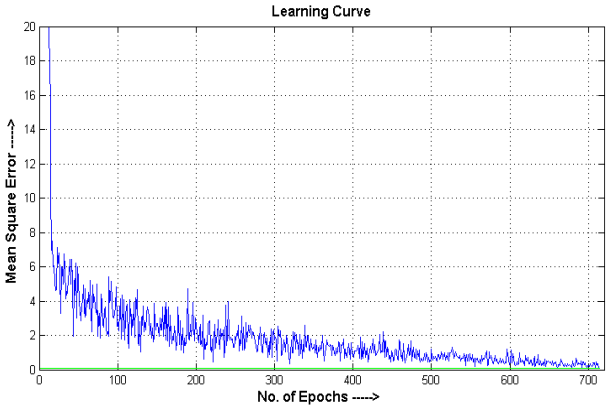

Fig. 1: Standard EBP algorithm

At first, the standard EBP algorithm is utilized for training of the MLP network. The learning curve of this training algorithm has been shown on Figure 1. It follows that the standard EBP algorithm has been reached the maximum expected MSE after 716 iterations and 149.05 seconds. Furthermore, the SPU-EBP algorithm with p=4 is applied to train the MLP network. Figure 2 depicts its learning curve and indicates that the proposed SPU-EBP learning algorithm has been achieved to the maximum expected MSE after 923 iterations and 176.75 seconds. It is concluded that the SPU-EBP algorithm requires more NR and time of convergence in comparing to the standard EBP algorithm. But, it must not be disregarded that the CC and ECT of SPU-EBP algorithm are approximately quarter of memory usage of standard EBP algorithm. Also, the same experiments are repeated for VSS SPU-EBP algorithm and the result has been illustrated in Figure 3. It is observed that the VSS SPU-EBP algorithm with 76.09 seconds has been reached the maximum expected MSE after 596 iterations.

These results follow that the VSS SPU-EBP algorithm has low time of convergence and NR in comparing the SPU-EBP algorithm. In other words, it has low convergence time, ECT, CC and space complexity in comparing the SPU-EBP and standard EBP algorithms.

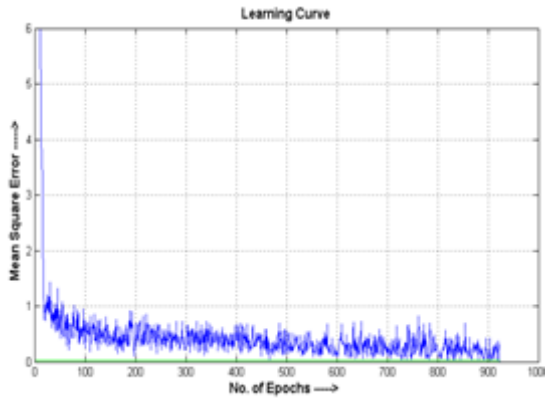

Fig. 2: SPU-EBP algorithm p=4

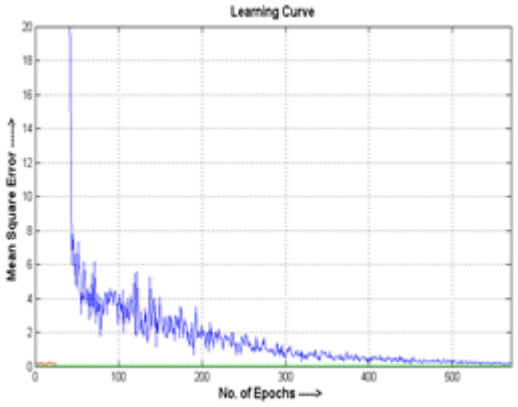

Fig. 3: VSS SPU-EBP algorithm

5.2. Oil price prediction system

In this section, the performance of the proposed algorithms is shown on the Oil price data set. In the experiments, the data set is partitioned into two equal subsets for training set and testing set. In all experiments, the MLP neural network have 10 neurons in input layer, 2 hidden layers with 9 and 8 neurons and 7 neurons in output layer. We assume maximum expected MSE is 0.3 in all tests.

At first, the standard EBP algorithm is utilized for training of the MLP network. The learning curve of this training algorithm has been shown in Figure 4. It follows that the standard EBP algorithm has been reached the maximum

expected MSE after 1700 iterations and 1018.11 seconds. Furthermore, the SPU-EBP algorithm with *p=4* is applied to train the MLP network. Figure 5 depicts its learning curve and indicates that the proposed SPU-EBP learning algorithm has been achieved to the maximum expected MSE after 2500 iterations and 1312.49 seconds. Also, the same experiments are repeated for VSS SPU-EBP algorithm and the result has been illustrated in Figure 6. It is observed that the VSS SPU-EBP algorithm after 237.24 seconds and 648 iterations has been reached the maximum expected MSE.
These results follow that the VSS SPU-EBP algorithm has low convergence time compared the SPU- EBP algorithm. In other words, it has lower convergence time, ECT, CC and space complexity compared the SPU-EBP and standard EBP algorithms.
Next, we consider the parameter *p*=6. Figure 7 depicts learning curve of SPU-EBP algorithm, and this curve converge maximum expected MSE after 5194 iterations with 3012.341 seconds and in the next figure (figure 8), by applied fit variable step size, VSS SPU_EBP algorithm reaches MSE after 2100 iterations and 492.31 seconds.
Since the two proposed algorithms based parameter *p*=4 and *p*=6 just update 25% and 16% percent of coefficients in each round, then ECT and CC consequently are decreased 75 and 83 percent than standard EBP. Also the results show the VSS SPU-EBP algorithm increases time of convergence in substantial amount.
As you see by increasing parameter *p* from 4 to 6, time of convergence and NR in two proposed algorithms increase but ECT and CC parameters decrees more.

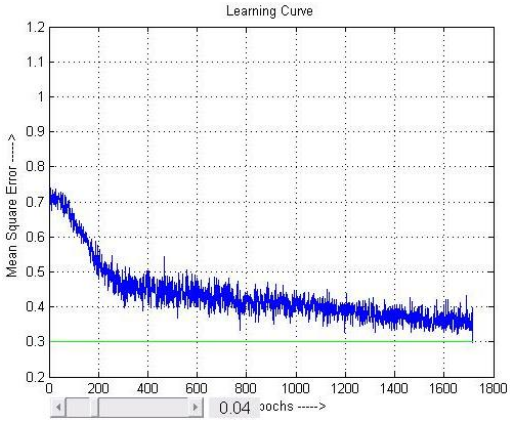

Fig. 4: Standard EBP algorithm.

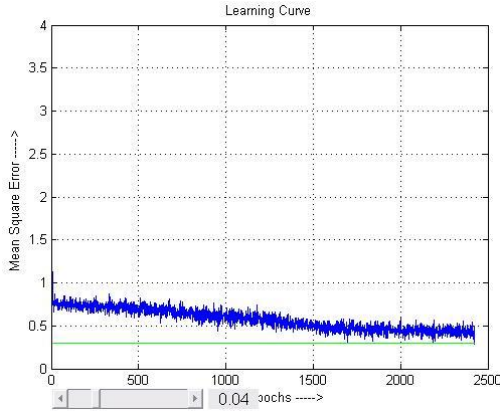

Fig. 5: SPU-EBP algorithm with p=4.

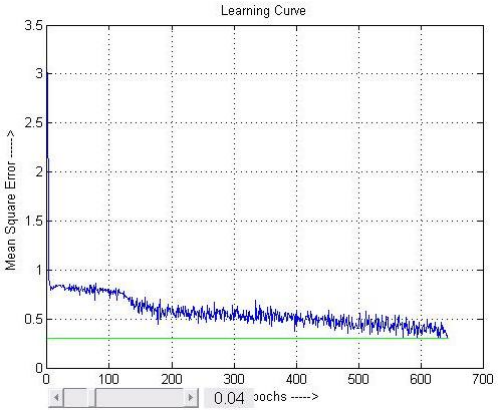

Fig. 6: VSS SPU-EBP algorithm with p=4.

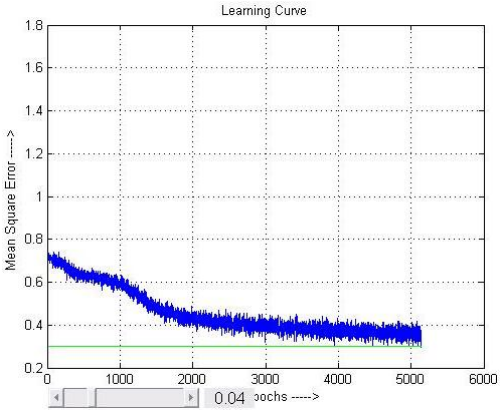

Fig. 7: SPU-EBP algorithm with p=6.

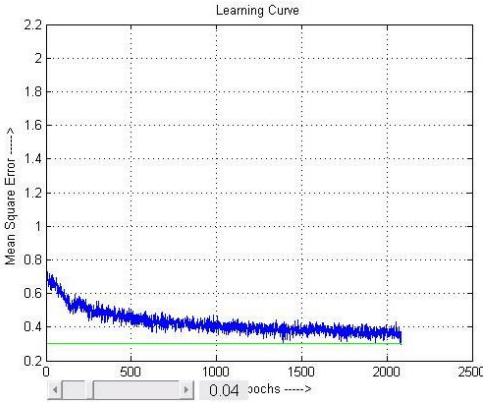

Fig. 8: VSS SPU-EBP algorithm with p=6.

5.3. Video classification system

Automatic video classification is an attractive and active research area in the field of video processing domain. It is an essential step in studying content and semantic of video data sets and in developing techniques for efficient retrieve, categorization and analyzing of large-scale video databases.
To date, numerous approaches have been put forward for the classification of video shots, and have produced highly acceptable results. After studying the literatures, we discovered that in these methods many of standard classifiers, such as Bayesian [18], Support Vector Machines (SVMs) [19], and neural networks [20] have been used. In addition, recently the Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMM) [21] are utilized popularly in the classification of video shots. Here, our objective is not to devise some novel works in the video classification area. We mainly focus on analyzing of the generalization property of the proposed training algorithms through classifying unseen video samples. Also, for the reason that typically video has huge set of dissimilar and noisy features, so the use of these data can be used to explain better the advantages and disadvantages of the proposed algorithms.
For lack of space we present the results in a compact table. The first column of the table shows the names of algorithms, the second column shows time of convergence and the third column shows NR. In these cases we choose parameter p=8 and MSE is considered to 0.4.
The gain results are depicted in table 1. As you see, standard EBP converges after 6,310 iterations and 1,539,703.1 seconds and the SPU-EBP algorithm converges after 11,278 iterations and 1,986,409.6 seconds and these parameters are 4,235 iterations and just 769,400 seconds for VSS SPU-EBP algorithm. The results show, by the VSS SPU-EBP algorithm time of convergence is decreased 49% than standard EBP and 60% than SPU-EBP algorithm. Also since, we update in each round just 1/8 of coefficients of input vectors then we can say CC and ECT is decreased 87% percent by SPU-EBP and VSS SPU-EBP algorithms than the standard EBP algorithm.

Table 1:

Name of algorithm	Time of Convergence	NR
Standard EBP Algorithm	1,539,703.1(s)	6,310
SPU-EBP algorithm p=6	1,986,409.6(s)	11,278
VSS SPU-EBP algorithm with p=6	769.400(s)	4,235

These results present our proposed algorithms have appropriate performance in similar conditions and the results are acceptable. We can deduce form the results of Figures 1, 2, 3, 4 and 5; the type of data set and structure of neural network has direct effect on convergence speed and computation load of algorithms.

6. Conclusion

In this paper, we present two new algorithms, SPU-EBP and VSS SPU-EBP algorithms. In the SPU-EBP algorithm we use a sequential partial update Error back propagation approach that updates coefficient vectors partially in a sequential manner. The results show SPU-EBP certainly, reduces the computation complexity (CC) and elapsed CPU time (ECT) in each round but it is concluded that the SPU-EBP algorithm require more NR and time of convergence in comparing to the standard EBP algorithm. But, it must not be disregarded that the CC, ECT and space complexity of SPU-EBP algorithm is approximately quarter of memory usage of standard EBP algorithm. To speed up more the convergence rate of the proposed SPU-EBP algorithm we propose VSS SPU-EBP algorithm. In the VSS SPU-EBP algorithm, we use a gradient based learning rate to speed up more the convergence time of training algorithm. These results follow that the VSS SPU-EBP algorithm has low convergence time, NR, CC and ECT in comparing to the SPU- EBP and standard EBP algorithms. Then we can deduce from the experiments, the VSS SPU-EBP algorithm has acceptable performance in compared to SPU-EBP and EBP algorithms.

References

[1] D. E Rumelhart and J. L. McClelland, 1986, "*Parallel distributed Processing",* Cambridge, MA, MIT Press.
[2] L.W. Chan and F. Fallside, 1987, "*An adaptive training algorithm for back propagation networks*", Computer Speech and Language, Vol. 2, pp. 205-218.
[3] R. Jacobs, 1988, "*Increased rates of convergence through learning rate adaptation*" , Neural Networks, 1:295-307.
[4] D. E. Rumelhart, G. E. Hinton, R. J. Williams, 1986, "*Parrallel Distributed Processing Exploration of the Micro-Structure of Cognition*". Cambridge, MA: MIT Press.
[5] P. Wolfe, 1969, "*Convergence conditions for ascent methods*", SIAM Mathematical Review 11 pp. 226–235.
[6] H.Leung and V.Zue, 1990, "*Phonetic classification using multilayer perceptions*," in ICASSP.
[7] T. Masters, 1993, "*Practical Neural Network Recipes in C++",* Academic Press, Inc.
[8] Hampshire, A. Waibel, 1990, "*A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks"*, IEEE, Transactions on Neural Networks, June, Vol. 1, No. 2.
[9] B. A. Telfer, H. H. Szu, 1994 "*Energy functions for minimizing misclassification error with minimum complexity networks*", Neural Networks, vol. 7, pp. 809-818.

[10] Crane, R.; Fefferman, C.; Markel, S.; and Pearson, J. 1995, "*Characterizing neural network error surfaces with a sequential quadratic programming algorithm*", In Machines That Learn.
[11] Caruana, R., 1993, "Generalization vs. Net Size", Neural Information Processing Systems, Tutorial, Denver, CO.
[12] Mahesh Godavarti And, Mahesh Godavarti , Alfred O. Hero Iii, 2011, "*Stability Analysis Of The Sequential Partial Update Lms Algorithm*", Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc, pp. 3857—3860.
[13] Douglas, S.C1997, 1997, "*Adaptive filters employing partial updates*". IEEE Trans. on Circuits and Systems II, 44(3):209–216.
[14] Maryam Rahmani nia, Ali Amiri, Mahmood Fathy, 2010, "*Partial Update Error Back propagation Algorithm*," 15th Iranian Computer society Conference, pp.287-294, Tehran, Iran.
[15] R. A. Fisher, 1936, "*The Use of Multiple Measurements in Taxo- nomic Problems*," Annual Eugenics, Vol. 7, No. 2, pp. 179-188.
[16] F.H.F. Leung, H.K. Lam, S.H. Ling, and P.K.S. Tam, 2013 *"Tuning of the structure and parameters of neural network using an improved genetic algorithm*," IEEE Trans. Neural Networks, vol. 14, no. 1, pp. 79-88, Jan.
[17] Y. F. Yam and T. W. S. Chow, 2000, "*A Weight Initialization Method for Improving Training Speed in Feed forward Neural Network*," Neurocomputing, Vol. 30, No. 1-4, pp. 219-232.
[18] Werbos, P., Beyond Regression: 1974, "*New Tools for Prediction and Analysis in the Behavioral Sciences*", PhD thesis, Harvard University.
[19] V.Vapnik, 1995, "*The Nature of Statistical Learning Theory*", Chapter 5. Springer-Verlag, New York.
[20] Krose, B. and van der Smagt, P., eds, 1993, "*An Introduction to Neural Networks*", fifth edn, University of Amsterdam.
[21] H.Bourlard and N.Morgan, 1994, "*Connectionist Speech Recognition: A Hybrid Approach*", lower Academic Publishers.