NEUROCOMPUTING

ELSEVIER Neurocomputing 48 (2002) 691–704

www.elsevier.com/locate/neucom

Training feedforward networks using simultaneous perturbation with dynamic tunneling

P. Thangavel*, T. Kathirvalavakumar

Department of Computer Science, University of Madras, Chepauk, Chennai 600 005, India

Received 5 October 2000; revised 15 March 2001; accepted 3 June 2001

Abstract

An efficient technique, namely simultaneous perturbation with dynamic tunneling for training single hidden layer feedforward network, is proposed. A sigmoidal hidden neuron is added to the single hidden layer neural network after training. Then the cascaded network is trained again using simultaneous perturbation. The dynamic tunneling technique is employed to detrap the local minima in training. The proposed technique is shown to give better convergence results for the selected problems, namely neuro-controller, encoder, adder, demultiplexer, XOR and L–T character recognition problem. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Single hidden layer; Cascade; Simultaneous perturbation; Dynamic tunneling

1. Introduction

Recently, neural networks have been extensively studied and used in many areas of science and engineering [1–11]. Feedforward neural networks are known to be universal approximators for nonlinear functions [2]. Training feedforward neural networks can be viewed as a nonlinear optimization problem in which the goal is to find a set of network weights that minimize an error function. The back-propagation approach is a well known approach for supervised learning for neural networks to minimize an error function. One of the major problems with back-propagation is local minimum entrapment. The dynamic tunneling technique [1,5] will be applied to detrap the local minima. Spall [6] has introduced a simultaneous

* Corresponding author.
E-mail address: uninet@md2.vsnl.net.in (P. Thangavel).

perturbation technique for training feedforward networks. He has applied this approach to problems such as optimization, stochastic approximation, adaptive control and neural network controller and their applications [7,8]. He has also pointed out that simultaneous perturbation type of methods are superior to the conventional optimization techniques. Maeda and De Figueiredo [4] have used simultaneous perturbation to train a neuro-controller for controlling a robot arm. Roychowdhury et al. [5] have proposed a new method for efficient training of multilayered perceptrons by combining error backpropagation to find local minima and a dynamic tunneling technique to detrap the local minima.

In this paper we propose an efficient technique by combining simultaneous perturbation and a dynamic tunneling technique for training single hidden layer feedforward networks by cascade learning. A single hidden neuron is added to the single layer of the neural network after training. Then the cascaded network is trained. Simultaneous perturbation is used to find the local minimum and to minimize the cost function, and the dynamic tunneling is used to detrap the local minimum. Thus the application of simultaneous perturbation and tunneling results in identifying the point of global minimum. The efficiency of the proposed method is demonstrated for selected problems namely neuro-controller, encoder, adder, demultiplexer, XOR and L–T character recognition problem. The proposed training technique and complexity of the algorithm is presented in Section 2. Section 3 describes the performance of the proposed algorithm for the selected examples.

2. Training neural network

A single hidden layer network is used as a prototype. Let $X=(x_i)$ be the input vector, $Y=(y_i)$ be the output vector, $W=(w_{ij})$ be the weight matrix between input layer and hidden layer, $V=(v_{ij})$ be the weight matrix between hidden layer and output layer, and C and D be the perturbation matrices with uniformly distributed random numbers in the interval $[-0.01,0.01]$ except $[-0.001,0.001]$. The characterstic of the hidden layer neuron is sigmoidal and the output layer neuron is linear or nonlinear depending on the problem; $d_i(t)$ represents the desired output of the output layer neuron of the tth set of input vector. Define $e_i(t)$, the error function, as the difference between the desired output and the actual output of the network

$$e_i(t)=d_i(t)-y_i(t). \tag{1}$$

A commonly used criterion for the cost function is sum of squared error (SSE) or mean square error (MSE) defined as

$$J=\frac{1}{2}\sum_i (e_i(t))^2, \tag{2}$$

$$J=E\left[\sum_i (e_i(t))^2\right], \tag{3}$$

where E is the statistical expectation operator. In training the ith component of the modifying vector of the weights Δw_t is defined as

$$\Delta w_t^i = \frac{J(Y(W_t + C_t, V_t + D_t)) - J(Y(W_t, V_t))}{c_t^i},$$

where $C_t = [c_t^1, c_t^2, \dots, c_t^n]^{\mathrm{T}}$ and $D_t = [d_t^1, d_t^2, \dots, d_t^k]^{\mathrm{T}}$ are perturbation vectors for adding small disturbances to all weights, $J(Y(W_t, V_t))$ is the error function of the neural network model for the weights associated with the connections and $J(Y(W_t + C_t, V_t + D_t))$ is the error function when all the weights of the connections in the network are disturbed, that is added, simultaneously by perturbation vectors. The properties of the perturbation vectors are assumed to be as described by Maeda and De Figueiredo [4]. Weights of the network are updated in the following manner

$$w_{t+1} = w_t - \alpha \Delta w_t, \tag{4}$$

where α is a positive learning coefficient. Maeda and De Figueiredo [4] have pointed out that this learning rule is a type of stochastic gradient method. It is expected that the network will converge in the statistical sense. Simultaneous perturbation is used in improving the network synaptic weights to minimize the cost function. Detailed strict convergence conditions of this simultaneous perturbation algorithm have been described by Spall [7]. The dynamic tunneling technique is applied to detrap the local minimum in training the network using simultaneous perturbation.

2.1. Simultaneous perturbation training

Initially assume that the network consists of only one neuron in the hidden layer. This neuron is used to model the relationship between input value and model residual. Initialize the synaptic weights W and V of the hidden neuron unit with uniformly distributed random numbers in $[-1, 1]$. Then, measure the value of the error function using forward operation of the network for all patterns of the input. In this error function, the desired value of the output neuron is the residual of the desired value of the network. Then add the perturbation matrices C and D to all synaptic weights W and V, respectively and observe the value of the error function using forward operation for all input patterns. If the MSE of the network after perturbation is less than that of the network before perturbation then this single hidden unit will be cascaded with the already created network. Otherwise dynamic tunneling technique is employed. After cascading, the output of the feedforward network for all input patterns and its error function are computed. Then do perturbation simultaneously with all the synaptic weights of the cascaded network and find its error function for all input patterns. If the MSE of the cascaded network before perturbation is less than that of the cascaded network after perturbation then perform dynamic tunneling. Otherwise update the weights using learning rule (4).

Then simultaneous perturbation is applied on synaptic weights and the network is trained repeatedly for the predetermined number of steps. Selection of the single hidden unit and cascading procedure is repeated until the cost function reaches the desired level.

2.2. Dynamic tunneling technique

The concept of dynamic tunneling is based on the fact that any particle placed at small perturbation from the point of equilibrium will move away from the current point to another within a finite amount of time [5]. The dynamic tunneling is implemented by solving the following differential equations

$$\frac{\mathrm{d}W1}{\mathrm{d}t} = \alpha(W1 - W)^{1/3}, \tag{5}$$

$$\frac{\mathrm{d}V1}{\mathrm{d}t} = \alpha(V1 - V)^{1/3}. \tag{6}$$

Here α represents the strength of learning. The value of $W1$ is $w_{ij} + \varepsilon_{ij}$ for all i, j; $V1$ is $v_{jk} + \varepsilon_{jk}$ for all j, k where $|\varepsilon_{ij}| \ll 1$ and $|\varepsilon_{jk}| \ll 1$ and the integration for (5) and (6) will be carried out for a fixed amount of time t with a small time step Δt. The systems (5) and (6) are of type

$$\frac{\mathrm{d}u}{\mathrm{d}t} = u^{1/3} \tag{7}$$

which has an equilibrium point at $u = 0$, which violates the Lipschitz condition at $u = 0$ since

$$\left| \frac{\mathrm{d}}{\mathrm{d}u}\left(\frac{\mathrm{d}u}{\mathrm{d}t}\right) \right| = |(-1/3)u^{-2/3}| \to \infty \quad \text{as } u \to 0.$$

This equilibrium point of the system is an attracting equilibrium point since the system reaches to 0 from any given initial condition. Similarly if the right-hand side function of Eq. (7) is $u^{-1/3}$ then the system has unstable repelling equilibrium point, at $u = 0$ [5]. To detrap the local minima, dynamic tunneling has been employed on the synaptic weights of the output layer and hidden layer alternatively.

2.3. Training algorithm

Simultaneous perturbation:

1. Generate random numbers for the synaptic weights W and V and for the perturbation matrices C and D.
2. $W1 = W + C$ and $V1 = V + D$.
3. For each set of input pattern find the error function corresponding to the synaptic weights and the perturbed weights, respectively.
4. Find diff $= \mathrm{MSE}(W1, V1) - \mathrm{MSE}(W, V)$.

```
5. If ( diff $< 0$ ) then $W = W1$; $V = V1$; else do dynamic tunneling.
6. Update the weights using Eq. (4) to obtain $W'$ and $V'$.
7. If $\mathrm{MSE}(W', V') > \mathrm{MSE}(W, V)$ then do dynamic tunneling then set $W = W'$;
   $V = V'$.
8. Train the network using steps 2–7 until the network is trained to the desired
   accuracy.
```

Dynamic tunneling: Dynamic tunneling does the following for the synaptic weights V and W alternatively.

```
9.  For $j = k$ (number of neurons in the layer) down to 1 do
    begin
10. Generate uniformly distributed random numbers $\varepsilon_{ij}$ in the range $[-1, 1]$.
11. Find $v1_{ij} = v_{ij} + \varepsilon_{ij}$ for all $i$.
12. For $l = 1$–$n$ (where $n$ is a user defined value) do
    begin
13. Integrate $\mathrm{d}v1_{ij}/(\mathrm{d}t) = \alpha(v1_{ij} - v_{ij})^{1/3}$ for all $i$.
14. Find diff $= \mathrm{MSE}(W1, V1) - \mathrm{MSE}(W, V)$.
15. If ( diff $< 0$ ) then
                $W = W1; V = V1;$ exit tunneling;
    end $l$; end $j$.
```

2.4. Complexity of the algorithm

In this subsection the worst case time complexity of the proposed algorithm is obtained in terms of the number of iterations performed. First we consider the training of a single hidden unit structure network. In a cycle, the total number of iterations for training all the N patterns will be N. Let $R * t/\Delta t$ be the maximum number of times tunneling will occur for a particular neuron. In the training process the number of output neurons will be always constant. The number of function evaluations using Runge–Kutta fourth order method is 4 per step. Let M be the sum of number of outputs and one hidden neuron in the structure. Therefore, the total number of iterations required during tunneling in training single hidden unit structure is $R*t/\Delta t*4*M$. If K hidden neurons are required to reach the termination condition then K different single hidden unit structures are trained. Therefore, the total number of iterations required in training single hidden unit structures is

$$4 * K * N * M * R * \frac{t}{\Delta t}.$$

In cascading phase, the whole network will be trained. Cascading will occur $K - 1$ times. The number of iterations required for training the cascaded network will be $N * 4 * Q * S * t/\Delta t * (M + 1)$, where Q is the maximum number of times the structure is trained repeatedly after cascading and $S * t/\Delta t$ is the maximum number of times the tunneling occurs in the cascading phase. The total number of iterations required in the training of whole network will be

$$Q*4*N*S*\frac{t}{\Delta t}*(M+1)+Q*4*N*S*\frac{t}{\Delta t}*(M+2)$$

$$+\cdots+Q*4*N*S*\frac{t}{\Delta t}*(M+K-1)$$

$$=Q*2*N*S*\frac{t}{\Delta t}*(K^2+(2*M-1)*K+2M).$$

The computational effort of the proposed algorithm will be

$$4*K*N*M*R*\frac{t}{\Delta t}+2*Q*N*S*\frac{t}{\Delta t}*(k^2+(2*M-1)*K+2M).$$

3. Simulation results and discussion

The performance of the proposed training scheme has been measured using the selected example problems, namely two-link planar arm, encoder, adder, demultiplexer, XOR and L–T character recognition problem. In the examples two-link planar arm, encoder, adder and demultiplexer, the activation function $1/(1+e^{-x})$ has been used for hidden and output layers. The convergence of the learning process is measured by using SSE for the two-link planar arm problem. For encoder, adder and demultiplexer MSE is used. The activation function $(1-e^{-x})/(1+e^{-x})$ is used for hidden layer and $1/(1+e^{-x})$ for output layer in the examples XOR and L–T character recognition. Here we use SSE. In all the examples the learning parameter α is fixed as 0.005. All the example problems considered were executed with five different initial weights and the results are tabulated in Tables 1 and 2. Table 3 shows the comparison of results for encoder, adder and demultiplexer between the proposed algorithm simultaneous perturbation with dynamic tunneling(SPDT) and error backpropagation with dynamic tunneling(EBPDT) of Roychowdhury et al. [5]. Comparison of results of examples XOR and L–T character recognition between the proposed algorithm and the conjugate gradient with line search and backpropagation with weight extrapolation(BPWE) methods of Kamarthi and Pittner [3] are shown in Table 4.

3.1. Two-link planar arm

First we consider a static problem of neuro-controller for two-link planar arm. A neural network as neuro-controller can learn an inverse of a plant without any information about the sensitivity function of the objective plant. We have prepared 10 positions of the top of the arm to be learned. Top of the arm (x,y) is represented as follows using arm lengths l_1, l_2 and the angles θ_1, θ_2

$$x=l_1\cos(\theta_1)+l_2\cos(\theta_1+\theta_2),$$

$$y=l_1\sin(\theta_1)+l_2\sin(\theta_1+\theta_2),$$

Table 1
Simulation results for selected example problems

	No. of hidden neurons	MSE/SSE	Epoch	Termination condition	Time step
	1	0.000096	150	0.0001	0.005
Encoder	1	0.000098	96	0.0001	0.005
2-1-4	1	0.000094	131	0.0001	0.005
	1	0.000082	161	0.0001	0.005
	1	0.000056	139	0.0001	0.005
	1	0.000055	135	0.0001	0.0005
Encoder	1	0.000090	148	0.0001	0.0005
4-1-4	1	0.000080	127	0.0001	0.0005
	1	0.000074	134	0.0001	0.0005
	1	0.000090	125	0.0001	0.0005
	1	0.000098	101	0.0001	0.05
	1	0.000098	109	0.0001	0.05
Half adder	1	0.000092	90	0.0001	0.05
	1	0.000047	133	0.0001	0.05
	1	0.000076	85	0.0001	0.05
	1	0.000092	79	0.0001	0.05
	1	0.000078	69	0.0001	0.05
Full adder	1	0.000098	97	0.0001	0.05
	1	0.000092	146	0.0001	0.05
	1	0.000092	141	0.0001	0.05
	1	0.000093	226	0.0001	0.05
	1	0.000086	194	0.0001	0.05
Demultiplexer	1	0.000094	148	0.0001	0.05
	1	0.000067	230	0.0001	0.05
	1	0.000084	173	0.0001	0.05
	2	0.000080	226	0.0001	0.0005
	2	0.000086	238	0.0001	0.0005
XOR	2	0.000088	276	0.0001	0.0005
	2	0.000081	225	0.0001	0.0005
	2	0.000054	184	0.0001	0.0005
	1	0.000098	162	0.0001	0.005
	1	0.000082	147	0.0001	0.005
L–T	1	0.000089	124	0.0001	0.005
	1	0.000093	115	0.0001	0.005
	1	0.000099	149	0.0001	0.005

where θ_1 and θ_2 are angles as shown in Fig. 1. We have used the above training method to find the output as θ_1 and θ_2 for the network input x_1 and x_2. The characteristic of each output neuron is the ordinary sigmoid function. The network has been constructed with two different data sets. It can be observed from Table 2

Table 2
Simulation results for two-link planar arm for different data sets

	No. of hidden neurons	SSE	Termination condition	Time step
	2	0.048187	0.05	0.005
Two-link	4	0.049983	0.05	0.005
planar arm	4	0.048084	0.05	0.005
I data set	3	0.049446	0.05	0.005
	4	0.049981	0.05	0.005
	2	0.046292	0.05	0.005
Two-link	4	0.047966	0.05	0.005
planar arm	7	0.049704	0.05	0.005
II data set	5	0.042975	0.05	0.005
	5	0.043825	0.05	0.005

Table 3
Comparison table for the proposed method SPDT with EBPDT for the encoder, adder and demultiplexer problems

	Encoder 2-1-4	Encoder 4-1-4	Half Adder	Full Adder	Demultiplexer
No. of hidden neurons					
EBPDT	5	2	2	3	3
SPDT	1	1	1	1	1
MSE					
EBPDT	0.001	0.005	0.0035	0.0017	0.0544
SPDT	0.000082	0.000074	0.000047	0.000092	0.000067
Epoch					
EBPDT	871	216	406	396	296
SPDT	161	154	133	146	230

that to reach SSE of 0.048084 for the first data set, the network requires 4 neurons and requires 7 neurons to reach the SSE of 0.049704 for the second set of data. Maeda and De Figueiredo [4] have used two hidden layer neural networks with 10 neurons in each hidden layer for learning the 10 desired positions of the top of the arm. The actual positions of the top of the arm and the positions predicted by the proposed training algorithm and the positions predicted by Maeda and De Figueiredo [4] algorithm are shown in Figs. 2 and 3. It can be seen that the positions predicted by the proposed algorithm are almost close to the actual positions.

Table 4
Comparison table for the proposed method SPDT with conjugate gradient with line search and BPWE for XOR and L–T problems

	Conjugate gradient with line search	Back propagation with weight extrapolation	Proposed algorithm SPDT
XOR			
No. of hidden neurons	3	3	2
SSE	0.012	0.00008	0.000088
Total epoch	14	4886	276
L–T			
No. of hidden neurons	2	2	1
SSE	0.011	0.0001	0.000098
Total epoch	5	1811	162

3.2. Encoder, adder and demultiplexer

Two types of encoders were simulated in this example. Training pattern for the first type is 0001 → 0001; 0010 → 0010; 0100 → 0100; 1000 → 1000; and for the second type is 00 → 1000; 01 → 0100; 10 → 0010; 11 → 0001. The network structure used for the first type is 4-1-4 and for the second type is 2-1-4. Roychowdhury et al. [5] have used the structure 4-2-4 and 2-1-4-4 for the first and second types of encoders. The training pattern used for half adder are 00 → 00; 01 → 01; 10 → 01; 11 → 10; and for the full adder are 000 → 00; 001 → 01; 010 → 01; 011 → 10; 100 → 01; 101 → 10; 110 → 10; 111 → 11. The network architecture used for the half adder is 2-1-2 and for the full adder is 3-1-2 where as Roychowdhury et al. [5] have used 2-2-2 and 3-3-2, respectively.

For demultiplexer, one input, two selection and four output lines were considered. The output line is controlled by selection lines. The input pattern (xyz) should be read as: $x =$ data input, $y, z =$ selection lines. If output pattern is represented by $(abcd)$ then the selection lines $yz = 10$ indicate that the output c will be the same as the data input x, while other outputs are maintained at one. The proposed algorithm requires 3-1-4 network structure whereas 3-3-4 structure has been used in Roychowdhury et al. [5]. Fig. 4 shows the plot of epoch versus MSE of the network.

Table 3 shows the total number of epochs and neurons required to reach the termination condition using the proposed algorithm and the results reported by Roychowdhury et al. [5]. It can be observed from the table that there is substantial improvement in number of neurons, errors and epochs.

3.3. XOR and L–T character recognition

The training pattern used for XOR are 00 → 0; 01 → 1; 10 → 1; 11 → 0. Our proposed algorithm needs 2-2-1 structure for SSE of 0.000088 within 276 epochs whereas Kamarthi and Pittner [3] have used 2-3-1 network structure for

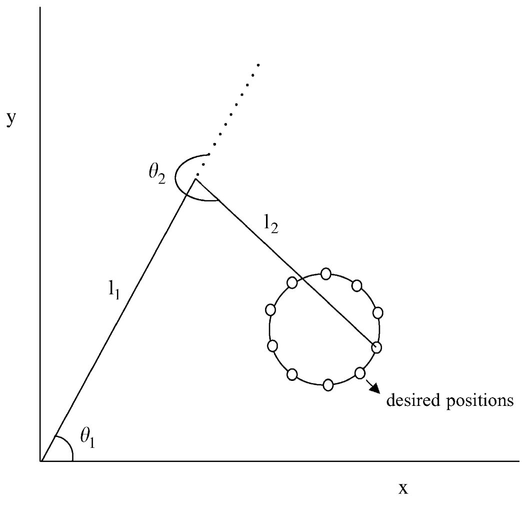

Fig. 1. Two-link planar arm.

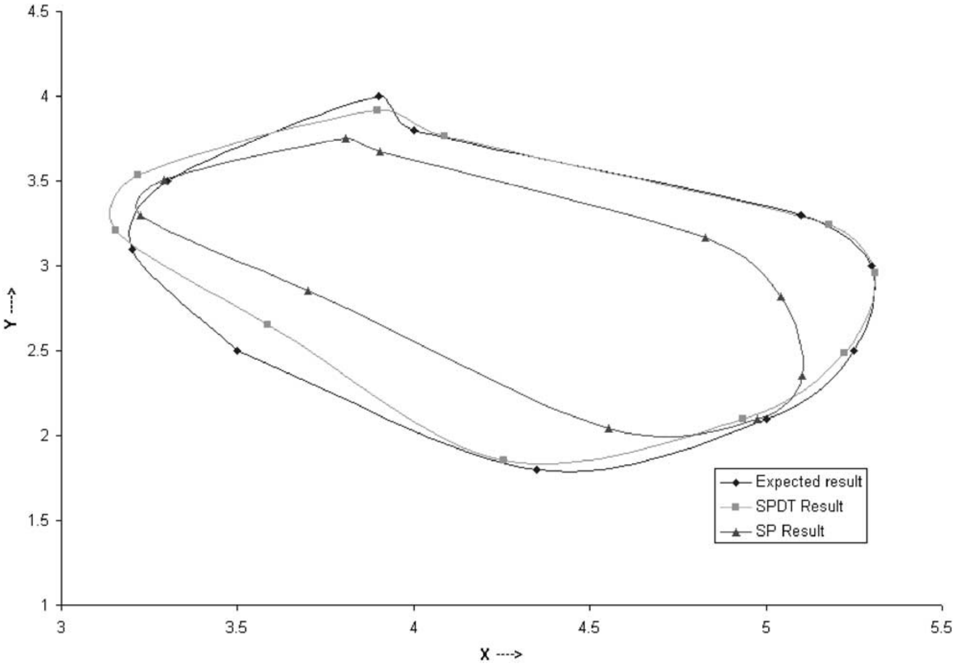

Fig. 2. Actual points and predicted points for the first data set of the two-link planar arm.

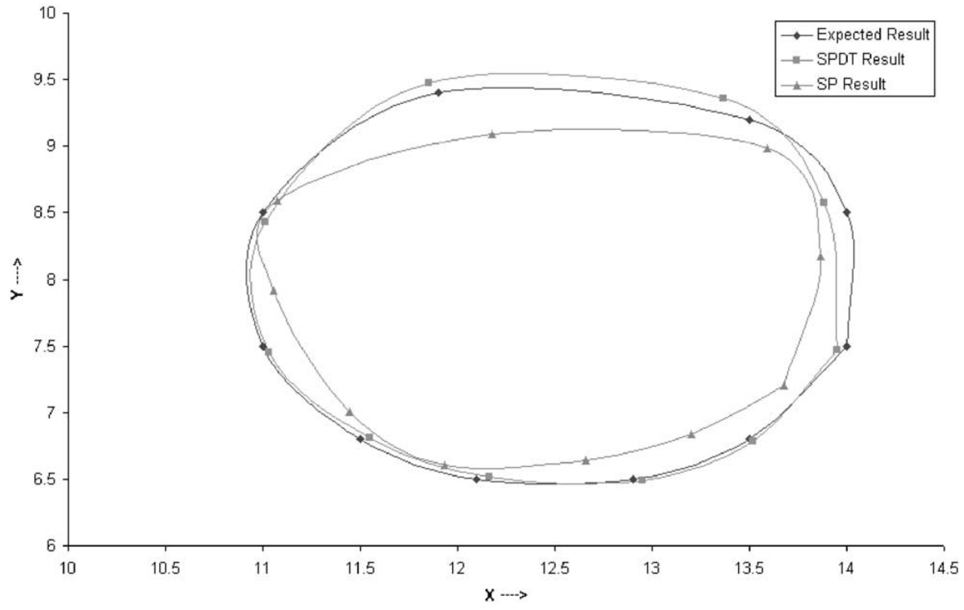

Fig. 3. Actual points and predicted points for the second data set of the two-link planar arm.

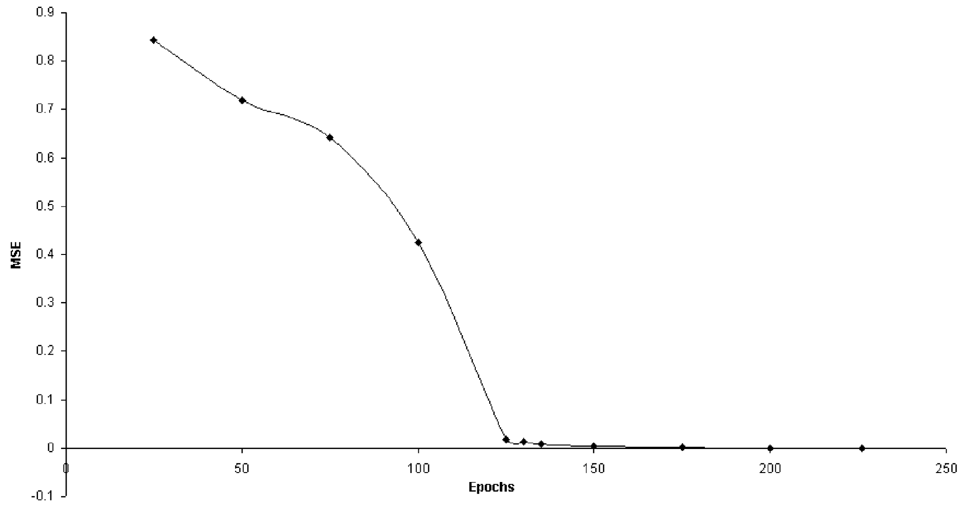

Fig. 4. Learning curve based on MSE and epoch for demultiplexer.

BPWE with SSE of 0.00008 and 4886 epochs and they have further shown that the conjugate gradient with line search method is able to train the network to an error value not smaller than 0.012, that is it could not train the network below this error for the termination condition of 10^{-4}.

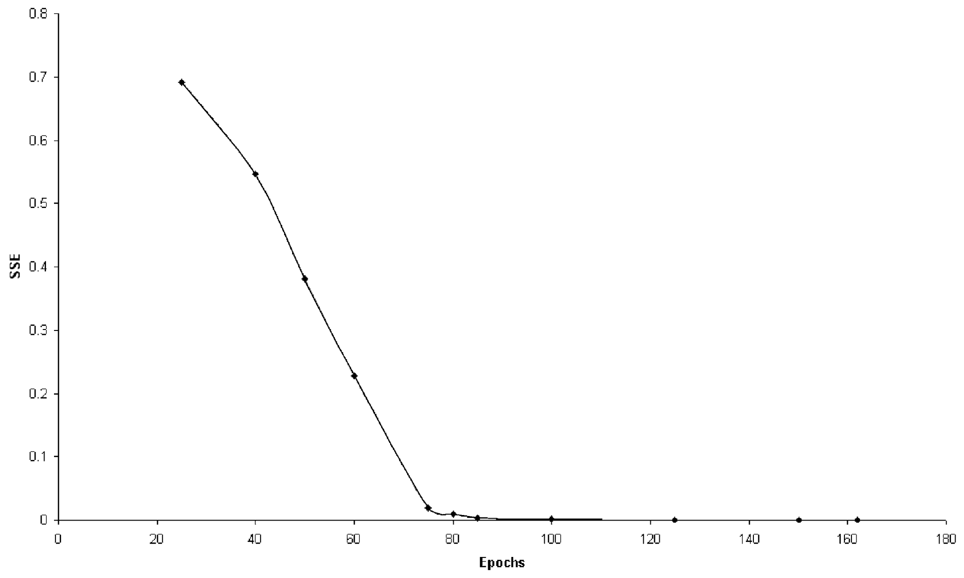

Fig. 5. Learning curve based on SSE and epoch of L–T character recognition.

In the letters L and T character recognition problem, the training set consisted of a 3×3 pixel binary image for each letter with its four orientations, in total eight patterns. The letters L and T are indicated by the target values 0.05 and 0.95 respectively for the output unit. The proposed SPDT requires 9-1-1 network structure to converge to SSE of 0.000098 using 162 epochs with a tolerance of 10^{-4} for termination. Kamarthi and Pittner [3] had 9-2-1 structure to converge to 0.0001 in 1811 epochs using the BPWE method whereas they have pointed out that their structure did not converge using conjugate gradient with line search method, and was locked in a local minima after reaching the SSE of 0.011 for the termination condition 10^{-5}. Fig. 5 shows the decrease in the error corresponding to epochs of the network.

4. Conclusion

Simultaneous perturbation with the dynamic tunneling approach for building single hidden layer networks is proposed. Sigmoidal single hidden neurons after training are cascaded to the network in such a way that it models the relationship between input data and model residual and then the whole network is trained. Simultaneous perturbation has been used to find the local minima and to minimize the cost function. Dynamic tunneling has been employed to detrap the local minima. It has been observed that the selected examples require a smaller number of neurons and epochs than the related results in the literature.

The proposed algorithm is very simple and needs lesser number of manipulations in learning. The value of the cost function, number of epochs and hidden neurons are significantly reduced in comparison to the error backpropagation with dynamic tunneling, conjugate gradient with line search and backpropagation with weight extrapolation methods. The proposed algorithm overcomes the local minimum point in learning using simultaneous perturbation procedure.

During simulation it has been noted that the number of epochs needed to converge to the global minimum depends on the initial choice of the weights. The results obtained for different initial weights show the robust learning of the proposed algorithm.

Acknowledgements

The authors are grateful to the editor and the anonymous referees for their constructive comments which helped to improve the quality of the paper.

References

[1] J. Barhen, V. Protopopescu, D. Reister, TRUST: a deterministic algorithm for global optimization, Science 276 (1997) 1094–1097.
[2] K. Hornik, M. Stinchombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.
[3] S.V. Kamarthi, S. Pittner, Accelerating neural network training using weight extrapolations, Neural Networks 12 (1999) 1285–1299.
[4] Y. Maeda, R.J.P. De Figueiredo, Learning rules for neuro-controller via simultaneous perturbation, IEEE Trans. Neural Networks 8 (1997) 1119–1130.
[5] P. Roychowdhury, Y.P. Singh, R.A. Chansarkar, Dynamic tunneling technique for efficient training of multilayer perceptrons, IEEE Trans. Neural Networks 10 (1999) 48–55.
[6] J.C. Spall, A stochastic approximation technique for generating maximum likelihood parameter estimates, Proceedings of the American Control Conference 1987, pp. 1161–1167.
[7] J.C. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, IEEE Trans. Automat. Control 37 (1992) 332–341.
[8] J.C. Spall, J.A. Christion, Nonlinear adaptive control using neural networks: estimation with a smoothed form of simultaneous perturbation gradient approximation, Statisca Sinica 4 (1994) 1–27.
[9] O. Stan, E. Kamen, A local linearized least squares algorithm for training feedforward neural networks, IEEE Trans. Neural Networks 11 (2000) 487–495.
[10] G.J Wang, C.C. Chen, A fast multilayer neural network training algorithm based on the layer-by-layer optimizing procedures, IEEE Trans. Neural Networks 7 (1996) 768–775.
[11] J. Zhang, A.J. Morris, A sequential learning approach for single hidden layer neural networks, Neural Networks 11 (1998) 65–80.

P. Thangavel received the M.Sc. degree in Mathematics from Bharathiar University, in 1985, M.Tech. degree in Computer Science and Data Processing from Indian Institute of Technology, Kharagpur in 1988 and the Ph.D. degree in Computer Science from Bharathidasan University in 1995. From 1988 to 1995 he worked as a Lecturer in Mathematics at Bharathidasan University. Since 1995 he has been working as Reader in Computer Science at University of Madras. His research interests include Parallel and Distributed Algorithms and Dynamic Chaotic Neural Networks.

T. Kathirvalavakumar received M.Sc. degree in Mathematics from Madurai Kamaraj University in 1986, Post Graduate Diploma in Computer Science and Applications from Bharathidasan University in 1987 and M.Phil. degree in Computer Science from Bharathiar University in 1994. Since 1987 he has been working as Lecturer in Computer Science at V.H.N.Senthikumara Nadar College, Virudhunagar. At present he is a doctoral candidate in the Department of Computer Science at University of Madras through Faculty Improvement Programme of University Grants Commission, India. His area of interest includes Neural Networks and Applications.