ELSEVIER

Neurocomputing 50 (2003) 193–209

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Simultaneous perturbation for single hidden layer networks — cascade learning

P. Thangavel*,[1], T. Kathirvalavakumar[2]

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

Received 8 June 2000; accepted 23 November 2001

Abstract

A simultaneous perturbation approach for cascade learning of single hidden layer neural network is presented. A sigmoidal hidden neuron is added to the single layer of hidden neurons after training until the error has stopped decreasing after a certain limit. Then, the cascaded network is again trained using simultaneous perturbation. Perturbation employed on the weights connecting to hidden neurons are changed to detrap the local minima in training. The proposed technique gives better convergence results for the selected problems, namely neuro-controller, XOR, L–T character recognition, two spirals, simple interaction function, harmonic function and complicated interaction function.

Keywords: Cascading; Feedforward neural network; Local minima; Simultaneous perturbation

1. Introduction

Recently, neural networks have been well studied and widely used in many fields to model complex systems. Its capability has been well demonstrated both in practical applications [5] and in theoretical study. Feedforward neural networks are powerful models for solving non-linear mapping problems. Despite advances in neural networks, determining the most appropriate network size for solving a specific task is yet to be solved. Feedforward neural network model selection techniques can be classified into three groups: (i) perform a selection through arbitrary models, (ii) begin with a

* Corresponding author.

E-mail address: thangavelp@yahoo.com (P. Thangavel).

[1] The work of P. Thangavel is partially supported by CSIR, Government of India sponsored project;

[2] The work of T. Kathirvalavakumar is funded by UGC, Government of India, Faculty Improvement Programme.

0925-2312/03/$ - see front matter
PII: S0925-2312(01)00704-4

complex model and then simplify, and (iii) begin with a simple model then increase its complexity.

In the first method, several arbitrary structures are tried and the one giving the best performance is selected. Such neural networks could have more hidden neurons than necessary [4,11,14,23]. In the second method, a large network is trained first and its size is reduced by removing redundant hidden neurons using the network pruning technique [2,8,15–17]. Castellano et al. [2] have described an iterative pruning algorithm to remove such redundant hidden neurons. Pruning requires advanced knowledge of what size is large for the problem at hand but this is not a serious concern as upper bound on the number of hidden units have been established [8]. In the third one, start with a small network and then add additional hidden units and weights until a satisfactory solution is found. This is called constructive method [9,19,24,25]. The small network to start with has no hidden units. If prior knowledge of the problem is available, an alternative initial state may be supplied by the user. Constructive algorithm searches for small network solution. Smaller networks are more efficient in feedforward computation and can be described by a simpler set of rules. If the initial network selected is too small, it may not converge to a good solution and hence underfit the data. On the other hand, selecting an initial network that is much larger than required makes training computationally expensive. Constructive algorithm will spend the majority of their time training the networks smaller than the final networks as compared to algorithms that start training with an oversize network. This method has been proved to be powerful for training feedforward neural networks.

Recently, several researchers [3–8,10,13,20–25] have proposed different approaches to find the structure of the neural network. Chen et al. [3,4] have proposed a procedure for determining the neurons in radial basis function networks using orthogonal least squares regression. Holcomb and Morari [7] have proposed a local training method for radial basis function network training and to find the number of hidden neurons. Wang et al. [21] have described a procedure to find the number of hidden layers and hidden neurons in it. Yam and Chow [23] have proposed a training algorithm for feedforward neural networks based on the linear least squares and modified gradient descent method. Fahlman and Lebiere [6] have proposed the cascade-correlation learning architecture for feedforward neural networks. The cascaded-correlation learning network [1,6] adds a new candidate hidden unit that receives weighted connections not only from the input, but also from any hidden unit already present in the network. These cascaded connections enable the new candidate unit to approximate the complex mapping of residual error which cannot be accomplished either by existing hidden units or by the new hidden sigmoidal unit without cascaded connections. Riedmiller and Braun [15] have proposed a learning algorithm Resilient backpropagation [RPROP] for multilayer feedforward neural networks. RPROP performs a local adaptation of the weight updates according to the behaviour of the error function. Cybenko [5] has shown that a continuous function can be approximated by a feedforward neural network with only a single hidden layer, where each unit in the hidden layer has sigmoidal non-linear activation function. Hush and Horne [9] have presented a constructive algorithm with piecewise linear sigmoidal nodes for non-linear function approximation. Young and Downs [24] have described an algorithm that constructs a feedforward network with

a single hidden layer of threshold units which implements the task for any consistant classification problem on real valued training vectors. Treadgold and Gedeon [20] have proposed an algorithm in which, once new hidden neuron is connected to the network, all weights are trained using RPROP algorithm [15]. SARPROP regularization term [15] is added to the algorithm in order to test the effect of regularization on a constructive algorithm. Zhang and Morris [25] have described a sequential orthogonal training method for single hidden layer network with sigmoidal activation function and also for the neural networks with mixed type of hidden neurons and hybrid models.

Hwang et al. [10] have examined the structural and algorithmic similarities and differences between a cascade-correlation learning network (CCLN) and a projection pursuit learning network (PPLN) and carried out a comparative study for regression and classification applications in feedforward networks. A universal acceleration technique for the backpropagation algorithm based on extrapolation of each individual interconnection weight was proposed by Kamarthi and Pittner [11]. Kwok and Yeung [12] have studied a number of objective functions for training new hidden units in constructive algorithms for multilayer feedforward networks. In [12] modified quickprop algorithm has been used to train the hidden layers while output layer was trained by computing pseudoinverse exactly. They have proposed few computational tricks that can be used to improve the optimization of the objective functions under practical computations. Maeda and De-Figueiredo [14] have used simultaneous perturbation to train a neuro-controller for controlling a robot arm. They have used neural network as a neuro controller to learn an inverse of a plant without any information about the sensitivity function of the objective plant. Simultaneous perturbation technique for training feedforward technique has been introduced by Spall [18].

In this paper, we propose a technique by combining cascade learning and simultaneous perturbation approach for constructing a single hidden layer neural networks. In this method, a sigmoidal hidden neuron is added to the single layer of hidden neurons after a period of training when the error has stopped decreasing below a certain limit. After the addition of new hidden neuron the whole network is again trained with simultaneous perturbation technique. Simultaneous perturbation is used to find the local minimum and to minimize the cost function. To detrap the local minima, added perturbed values on the connection weights are changed neuron by neuron starting from the output layer. The efficiency of the proposed method is demonstrated for the selected example problems namely neuro-controller, XOR, L–T character recognition, two spirals, simple interaction function, harmonic function and complicated interaction function. The next section describes the proposed method of simultaneous perturbation for cascade learning. Section 3 describes the performance of the proposed algorithm with different examples. Section 4 contains the concluding remarks.

2. Training of single hidden layer neural network

We consider a single hidden layer neural network. The activation function of hidden layer neurons are assumed to be sigmoidal. The output layer neuron(s) can have linear activation function or non-linear activation function depending on the problem. In this

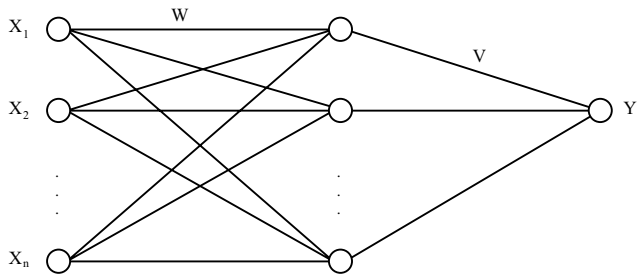


Fig. 1. Single hidden layer neural network.

paper for simplicity we consider networks with single output neuron having linear activation function. The input to the neural network may be received from outside the network. We use the cascading approach for constructing the network architecture.

2.1. Training method

We consider a single hidden layer neural network as shown in Fig. 1, in which $X = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$ is the input vector. Superscript T denotes transpose. $W_i = [w_i^1, w_i^2, \ldots, w_i^n]^{\mathrm{T}}$ is the weight matrix associated with the ith hidden neuron including thresholds as weights with input value 1. $W = [W_1, W_2, \ldots, W_k]^{\mathrm{T}}$ is a weight matrix associated with the hidden layer neurons. $V = [v^1, v^2, \ldots, v^k]^{\mathrm{T}}$ denotes the weight matrix associated with the output layer neuron. Superscript k denotes number of hidden neurons. The sigmoidal function

$$f(x) = \frac{1}{(1 + \mathrm{e}^{-x})}$$

is used for the hidden neuron to represent its input–output characteristic. Therefore, the output of the jth neuron corresponding to the input is

$$H_j = \frac{1}{(1 + \mathrm{e}^{-\sum_{p=1}^{n} w_j^p x_j^p})}.$$

$[H_1, H_2, \ldots, H_k]^{\mathrm{T}}$ is a vector which corresponds to the output of the hidden layer neurons.

$$Y_i = \sum_{j=1}^{k} V_j H_j$$

is the system output for the ith input pattern.

$$Y_{\mathrm{d}i} = \sum_{j=1}^{k} V_j H_j + E_i$$

is the desired output for the ith input pattern, where E_i is the model residual. Define the error function $J(W, V)$ as

$$J(W, V) = \sum_{i} (Y_i - Y_{\mathrm{d}i})^2.$$

Here, every hidden neuron is used to model the relationship between input data and model residuals. For the first hidden neuron the model residual is the desired output. The weights corresponding to the hidden neurons and the output neurons are modified by the learning rule via simultaneous perturbation technique.

2.2. Learning rule and simultaneous perturbation

We use the learning rule via simultaneous perturbation technique proposed by Maeda and De-Figueiredo [14] for cascade learning. The ith component of the modifying quantity corresponding to the tth iteration, of the weights Δw_t^i is defined as follows:

$$\Delta w_t^i = \frac{J(Y(W_t + C_t, V_t + D_t)) - J(Y(W_t, V_t))}{c_t^i}, \tag{1}$$

where $C_t = [c_t^1, c_t^2, \ldots, c_t^n]^{\mathrm{T}}$ and $D_t = [d_t^1, d_t^2, \ldots, d_t^k]^{\mathrm{T}}$ are the perturbation vectors with uniformly distributed random numbers in the interval $[-0.01, 0.01]$ except $[-0.001, 0.001]$ 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 of the neural network model 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 [14]. Weights of the network are updated using the following learning rule,

$$w_{t+1} = w_t - \alpha \Delta w_t, \tag{2}$$

where α is a positive learning coefficient. The error function is measured using forward operation of the neural network for all input patterns. Then, add small perturbation to all weights simultaneously and observe the value of the error function using forward operation of the neural network for all input patterns. Detailed strict convergence conditions of this simultaneous perturbation algorithm have been described by Spall [19].

2.3. Training via simultaneous perturbation

2.3.1. Training single hidden unit

Select one hidden neuron unit for training as in Fig. 2. This neuron is used to model the relationship between input value and model residual. The weights are uniformly distributed random numbers in $[-1, 1]$. The output of the feedforward neural network and its error function for all training patterns are calculated and accumulated. The random numbers for the perturbation vectors C_t and D_t are generated and added correspondingly with the weights W_t and V_t simultaneously. Again, calculate the output of the feedforward neural network and its error function for all desired input patterns and obtain the total error. If the mean squared error (MSE) of the network after perturbation is less than the MSE of the network before perturbation then use the learning rule to find the modified weight and update the weights otherwise change the perturbed weights of the network neuron by neuron from the output layer until the MSE after

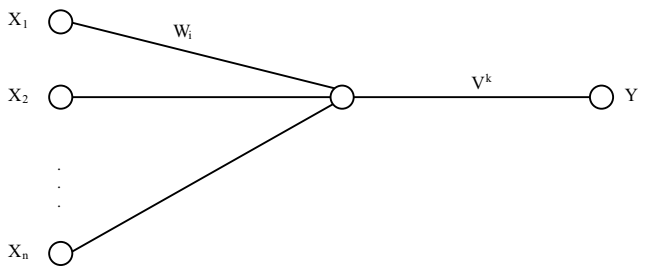


Fig. 2. Structure of the new hidden unit.

perturbation is less than the MSE before perturbation. If the MSE of the network after weight updation is greater than the MSE of the network before weight updation then do the modification by adding perturbation on the updated weights of the network neuron by neuron.

2.3.2. Training cascaded neural network

New hidden unit created as in Fig. 2 will be cascaded with the already created neural network structure. After cascading the new hidden unit, the whole structure will be trained by the simultaneous perturbation technique. During training after perturbation if the MSE of the network is greater than the MSE of the network before perturbation then change the perturbed weights of the network neuron by neuron in the output layer and then in the hidden layer. Using the learning rule find the modified weight and update the weights of the neural network. If the MSE of the network after updation is greater than the MSE of the network before updation then change the weights of the network by adding perturbation on the connection weights of the network neuron by neuron in the output layer and then in the hidden layer until the MSE of the network after updation is less than MSE of the network before updation. Selection of new hidden unit and cascading process will be continued until the training of the cascaded neural network structure results in giving small enough error.

2.3.3. Algorithm

- *Training*

  Step 1. Enter input values and expected output values.

  Step 2. Generate random numbers for the weight vectors.

  Step 3. For each input pattern
  compute output of the feedforward network, find the square error, accumulate $(\sum(Y - Y_d)^2)$ and then find MSE.

  Step 4. Generate perturbation vectors and add with weight vectors simultaneously.

  Step 5. For each input pattern
  compute output of the feedforward network, find the square error and accumulate $(\sum(Y - Y_d)^2)$ and then find $P1 = \text{MSE}$.

  Step 6. If the MSE before perturbation is less than the MSE after perturbation namely $P1$ then do Detrap local minima.

Step 7. Find the modifying quantities for all weights using Eq. (1) and update the weights of the network using Eq. (2).

Step 8. Find the output of the network and mean squared error namely $P1$. If this $P1$ is greater than the MSE obtained in step 3 then do Detrap local minima.

Step 9. Repeat steps 3–8 m times, where m is a user defined number.

- *Detrap local minima*

Step 10. Select neurons from output layer and then from hidden layer one by one and do the steps 11–15 repeatedly.

Step 11. Set the weights of the connection lines for the selected neuron with the new perturbation values.

Step 12. For $i = 1$–n times do steps 13–15, where n is a user defined number.

Step 13. Add the connection weights of the selected neuron with the new perturbation values.

Step 14. Find the output of the network for all training patterns and find the MSE.

Step 15. If this MSE is less than $P1$ then exit from this part.

3. Simulation results

All the example problems were simulated in C on a Pentium III with 550 MHz system. In all examples the activation function used in hidden neurons are sigmoidal. The network has been trained with different initial weights to test the robustness.

3.1. Two-link planar arm

First, we consider a static problem of Neuro-controller namely two-link planar arm which is described in Maeda and De Figueiredo [14]. We prepare 10 positions of the top of the arm to be learned. Top of the arm (x, y) is represented as follows using arm length l_1, l_2 and angles θ_1, θ_2 as follows:

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

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

where θ_1 and θ_2 are angles as shown in Fig. 3. We have used the above cascading approach to find θ_1 and θ_2 for the desired input x and y. The network has been trained with four different data sets. Each set of data has been trained with 10 different initial weights. The actual positions and the positions predicted by the network are shown in Figs. 4 and 5 for two sets of data. It has been found that four neurons are needed for each data set to reach the sum of squared error (SSE) 0.046898, 0.048605, 0.048949 and 0.049982, respectively, to the data set 1,2,3 and 4 to learn the 10 positions. The results are shown in Table 1. The learning rate α is assumed to be 0.00001. Whereas Maeda and De Figueiredo [14] have used two hidden layer neural network with 10 neurons in each hidden layer for learning the 10 desired positions of the top of the arm.

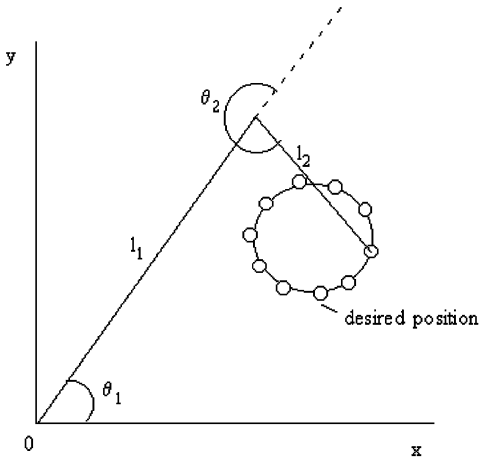


Fig. 3. Two-link planar arm.

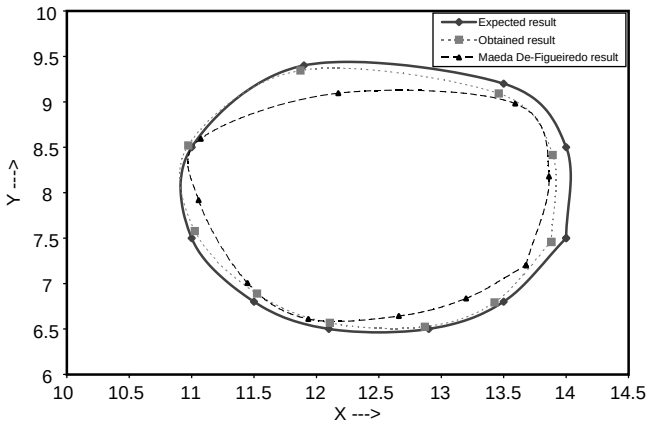


Fig. 4. Actual points and predicted points of the first set of data for the two-link planar arm.

3.2. XOR, L–T character recognition and two spirals

Next, we consider the standard XOR problem. The proposed algorithm has been executed with 25 different initial weights. It has been found that 3.32 hidden neurons and 3241.91 epochs are required on the average to reach the SSE 0.00008 within 5 s for training the single hidden layer network. The average time needed per epoch was 0.00146 seconds. The learning rate $\alpha = 0.001$ has been used. Whereas Kamarthi and Pittner [11] have used 2-3-1 network structure for backpropagation with weight extrapolation (BPWE) with SSE of 0.00008 and epochs 4886. They have further shown

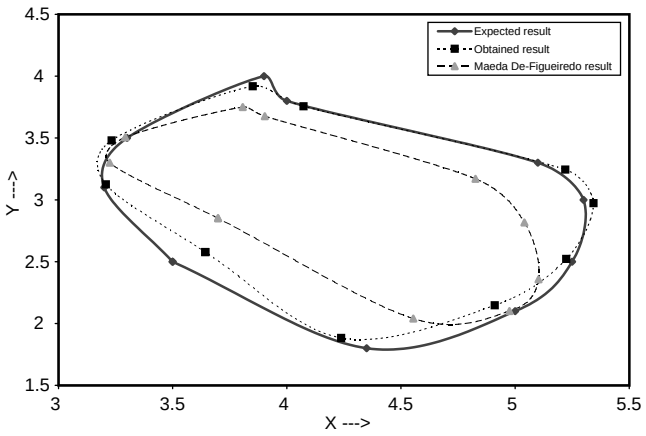


Fig. 5. Actual points and predicted points of the second set of data for the two-link planar arm.

that the conjugate gradient with line search method was not able to train the network below the error value 0.012 for the termination condition of 10^{-4}.

Next, we consider L–T character recognition problem. The training set consisted of a 3×3 pixel binary image for each letter with its four orientation, in total eight patterns. The target values for the letters L and T are choosen to be 0.05 and 0.95 corresponding to the output unit. The proposed algorithm requires 9-2-1 network structure to converge to SSE of 10^{-4}. The network is trained with 25 different initial weights. It has been found that the network converged with an average of 547.2 epochs and 3.2 s. The learning rate α is assumed to be 0.1. Kamarthi and Pittner [11] had 9-2-1 structure to converge in 1811 epochs using the BPWE method. Further, 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 0.011 for the termination condition 10^{-5}. Fig. 6 shows the decrease in the error corresponding to epochs of the network.

Then we consider two spirals problem with 82 original input patterns. The input patterns use points within a radius of 3 units and are randomly merged for training. The input coordinates represent the points of two interwined spirals in the two-dimensional plane. The network is trained to classify the points of two separate spirals. The points lying on the spirals are recognised with its corresponding target values 0 and 1. The proposed algorithm uses 11 hidden neurons with SSE 0.000097, and requires 7546.93 epochs and of time 1711.5 s on the average for 15 different initial weights in training. The average time needed per epoch was 0.2268 s. But Kamarthi and Pittner [11] have used 2-8-2-1 two hidden layer network to converge to SSE 0.0001. The BPWE [11] method converged within 3798 epochs and 78618 s whereas they have specified that the conjugate gradient with line search method did not converge after reaching SSE 18.16 in 17 epochs for the termination condition 10^{-4}. The decrease of SSE after addition of each neuron in the proposed algorithm is depicted in Fig. 7. Results

Table 1
Simulation results for the two link planar arm problem

	Number of hidden neurons	SSE	Termination condition
I data set	3	0.049694	0.05
	3	0.049148	0.05
	3	0.046875	0.05
	3	0.049234	0.05
	2	0.049932	0.05
	4	0.046898	0.05
	3	0.048200	0.05
	3	0.049911	005
	2	0.049574	0.05
	3	0.049421	0.05
II data set	3	0.045961	0.05
	3	0.048974	0.05
	3	0.049241	0.05
	3	0.048928	0.05
	3	0.048132	0.05
	2	0.049949	0.05
	3	0.048852	0.05
	3	0.047214	0.05
	4	0.048605	0.05
	3	0.049219	0.05
III data set	3	0.049953	0.05
	3	0.048601	0.05
	3	0.047015	0.05
	3	0.049886	0.05
	3	0.049790	0.05
	3	0.049961	0.05
	3	0.043492	0.05
	3	0.049612	0.05
	3	0.049569	0.05
	4	0.048949	0.05
IV data set	3	0.049217	0.05
	3	0.049975	0.05
	3	0.049814	0.05
	4	0.049982	0.05
	3	0.049218	0.05
	3	0.049104	0.05
	3	0.048048	0.05
	3	0.049628	0.05
	3	0.046983	0.05
	3	0.047699	0.05

of this section are tabulated in Table 2. Kamarthi and Pittner [11] have simulated the XOR, L–T and two spirals problems in C on a Sun SPARC-4 Ultra 5 workstation.

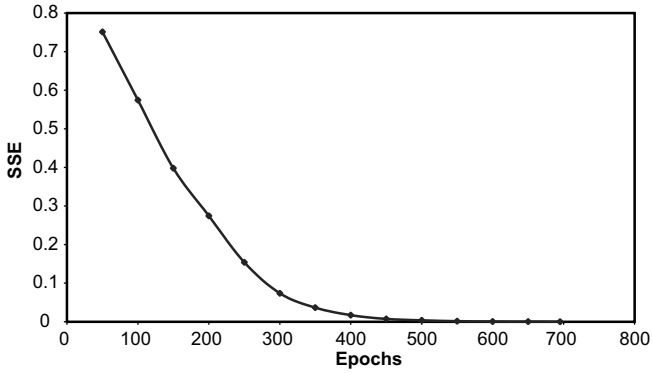


Fig. 6. Learning curve based on SSE and epochs for L–T.

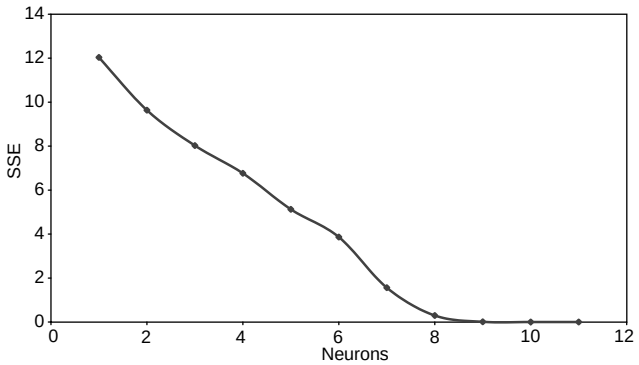


Fig. 7. Learning curve based on SSE and number of neurons for two spirals.

3.3. Two dimensional regression problems

In this subsection we consider the following two-dimensional regression problems

- *Simple interaction function*

$$f(x_1, x_2) = 10.391((x_1 - 0.4)(x_2 - 0.6) + 0.36).$$

- *Harmonic function*

$$f(x_1, x_2) = 42.659(0.1 + \tilde{x}_1(0.05 + \tilde{x}_1^4 - 10\tilde{x}_1^2\tilde{x}_2^2 + 5\tilde{x}_2^4)),$$

where $\tilde{x}_1 = x_1 - 0.5$, $\tilde{x}_2 = x_2 - 0.5$.

- *Complicated interaction function*

$$f(x_1, x_2) = 1.9(1.35 + e^{x_1}\sin(13(x_1 - 0.6)^2)e^{-x_2}\sin(7x_2)).$$

Table 2
Comparison table for the worst case results of the proposed method, conjugate gradient with line search and BPWE for XOR, L–T and two spirals problems

	Conjugate gradient with line search	Back propagation with weight extrapolation	Proposed algorithm
XOR			
No. hidden neurons	3	3	5
SSE	0.012	0.00008	0.00008
Total epochs	14	4886	3398
Time(s) per epoch	0.1	1.0	0.0015
L–T			
No. hidden neurons	2	2	2
SSE	0.011	0.0001	0.00009
Total epochs	5	1811	770
Time(s) per epoch	0.1	0.6	0.0026
Two spirals			
No. hidden neurons	10(8-2)	10(8-2)	11
SSE	18.16	0.0001	0.000099
Total epochs	17	3798	8891
Time(s) per epoch	1.6	20.4	0.3558

For training the above functions 225 points have been generated from uniform distribution $U[0,1]^2$. 2500 set of points have been generated for testing from a regularly spaced grid on $[0,1]^2$ for the above functions. The fraction of variance unexplained (FVU) on the test is used for comparison. It is defined as

$$\mathrm{FVU} = \frac{\sum_{i=1}^{N}(f(x_i) - f_n(x_i))^2}{\sum_{i=1}^{N}(f(x_i) - \bar{f})^2},$$

where $\bar{f} = 1/N \sum_{i=1}^{N} f(x_i)$, $N =$ number of training patterns. The learning rate α is assumed to be 0.00001 for the above three functions. Termination condition used and the hidden neurons needed are 0.0005, 0.05, 0.05 and 6,10,7, respectively, for the corresponding simple interaction, harmonic and complicated interaction functions. The obtained networks for the functions are tested with 10 different trials each containing 2500 sets of points. Mean testing FVU's of 10 different trials are 0.00092, 0.091284 and 0.092709 for the functions simple interaction, harmonic and complicated interaction, respectively. Whereas Kwok and Yeung [12] have obtained the minimum FVU's on mean testing in 100 trials for the simple interaction function, harmonic function and complicated interaction function are 0.00258, 0.24738 and 0.11126, respectively. They have selected this minimum FVU's among the networks ranging 1–15 hidden neurons with 10000 input patterns when testing but 225 input patterns at training.

Hwang et al. [10] have simulated complicated interaction function using cascade-correlation learning network. They have used 225 input patterns for training and 2500 input patterns for testing. Ten hidden neurons with 88 connection weights required for CCLN to reach the testing error (FVU) of 0.359. Whereas the proposed algorithm used

Table 3
Comparison table for the test mean FVU's of the proposed method with CCLN and Kwok and Yeung's result for selected problems

Function		Cascade-correlation learning	Kwok & Yeung's result	Proposed algorithm
Simple interaction function	Neurons	—	15	6
	FVU error	—	0.00258	0.00092
	connection weights	—	60	24
Harmonic function	Neurons	—	15	10
	FVU error	—	0.24738	0.091284
	connection weights	—	60	40
Complicated interaction function	Neurons	10	15	7
	FVU error	0.359	0.11126	0.092709
	connection weights	88	60	28

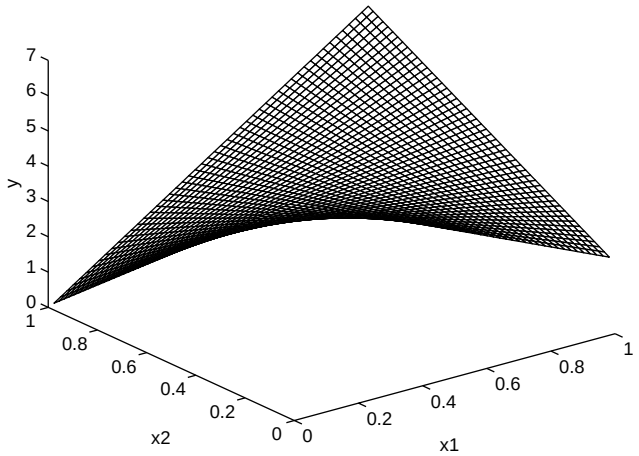


Fig. 8. Simple interaction function for the expected values.

only seven hidden neurons with 28 connection weights to reach the mean testing FVU 0.092709 with sigmoidal activation function for 10 different trials. Table 3 shows the comparative results for the above problem. Figs. 8–13 show the three-dimensional plots of the expected and the network results for the above three functions using 2500 test data sets.

4. Conclusion

A simultaneous perturbation training algorithm for building and training single hidden layer neural network is proposed. In this method, all hidden neurons are used to model the relationships between input data and model residuals. A sigmoidal hidden neuron

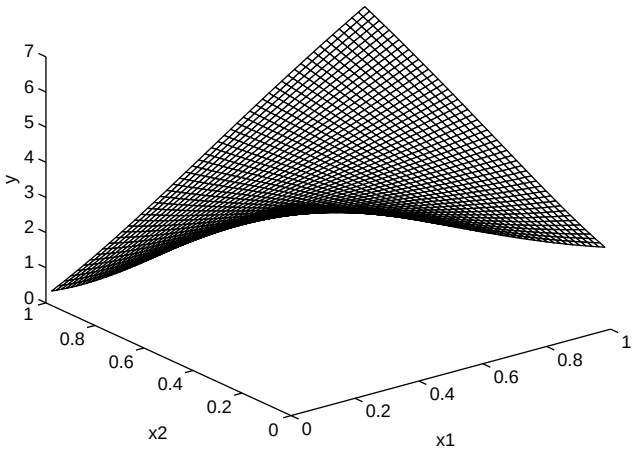


Fig. 9. Simple interaction function for the network values.

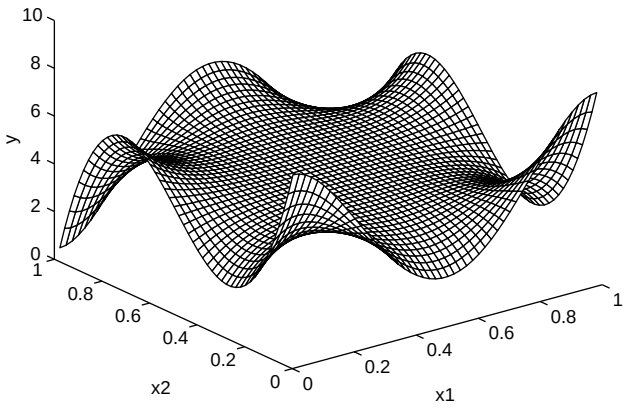


Fig. 10. Harmonic function for the expected values.

is added to the single layer of hidden neurons after a period of training when the error has stopped decreasing below a threshold value. After the addition of the new hidden neuron, the whole network is again trained with simultaneous perturbation. In training, perturbed values to the connection weights are changed to detrap the local minima. The method has been tested with selected examples. It has been observed that for the selected examples the number of neurons and epochs have been reduced compared to the related results in the literature.

The proposed algorithm is very simple and easy to implement and requires less number of manipulations. The value of the cost function, number of epochs, the number of neurons and number of layers are reduced in the proposed work when compared with the Maeda and De Figueiredo [14]'s work in the two-link planar arm problem. In

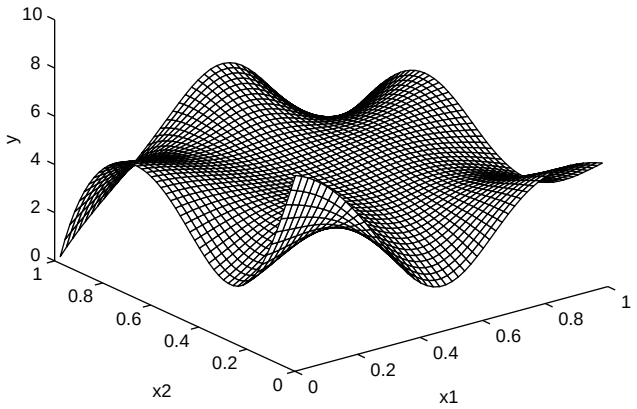


Fig. 11. Harmonic function for the network values.

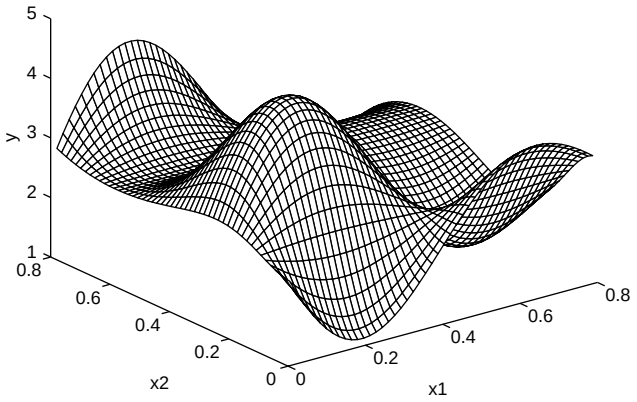


Fig. 12. Complicated interaction function for the expected values.

the XOR and L–T character recognition problems number of epochs and time needed for the proposed algorithm is less than the Backpropagation with weight extrapolation algorithm and conjugate gradient with line search method. In the two spirals problem eventhough the number of neurons and epochs needed for the proposed algorithm are greater than the Backpropagation with weight extrapolation algorithm, the proposed algorithm needs only single hidden layer network, less connection weights and less time to converge to the expected result.

The proposed algorithm needs less number of neurons, connection weights to reach the minimum cost function than the cascade-correlation learning [10] and the constructive algorithm [12] using different objective functions for training hidden units in two-dimensional regression problems. The results obtained for different initial weights show the robust learning of the proposed algorithm.

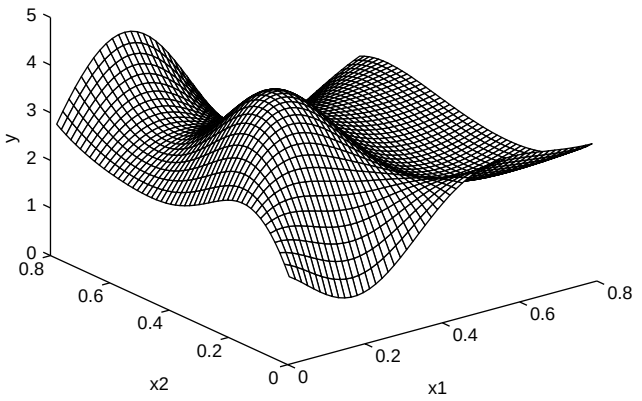


Fig. 13. Complicated interaction function for the network values.

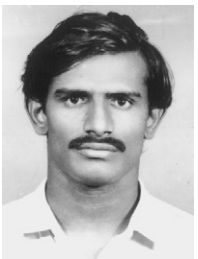
Acknowledgements

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

References

[1] C. Bishop, Neural Networks for Pattern Recognition, Clarendon, Oxford, 1995.
[2] G. Castellano, A.M. Fanelli, M. Pelillo, An iterative pruning algorithm for feedforward neural networks, IEEE Trans. Neural Networks 18 (1997) 519–531.
[3] S. Chen, S.A. Billings, P.M. Grant, Non-linear system identification using neural networks, Int. J. Control 51 (1990a) 1191–1214.
[4] S. Chen, S.A. Billings, C.F.N. Cowan, P.M. Grant, Practical identification of Narmax models using radial basis functions, Int. J. Control 52 (1990b) 1327–1350.
[5] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (1989) 303–314.
[6] S.E. Fahlman, C. Lebiere, The cascade-correlation architecture, Adv. Neural Inform. Process. Systems 2 (1990) 524–532.
[7] T. Holcomb, M. Morari, Local training for radial basis function networks towards solving the hidden units problem, Proceedings of the American Control Conference, Boston, MA, USA, 1991, pp. 2331–2336.
[8] S.C. Huang, Y.F. Huang, Bounds on the number of hidden neurons in multilayer perceptrons, IEEE Trans. Neural Networks 2 (1991) 47–55.
[9] D.R. Hush, B. Horne, Efficient algorithms for function approximation with piecewise linear sigmoidal networks, IEEE Trans. Neural Networks 9 (1998) 1129–1141.
[10] J. Hwang, S. You, S. Lay, I. Jou, The cascade-correlation learning: a projection pursuit learning perspective, IEEE Trans. Neural Networks 7 (1996) 278–289.
[11] S.V. Kamarthi, S. Pittner, Accelerating neural network training using weight extrapolations, Neural Networks 12 (1999) 1285–1299.
[12] T. Kwok, D. Yeung, Objective functions for training new hidden units in constructive neural networks, IEEE Trans. Neural Networks 8 (1997) 1131–1147.
[13] M. Lehtokangas, Fast initialization for cascade-correlation learning, IEEE Trans. Neural Networks 10 (1999) 410–414.

[14] Y. Maeda, R.J.P. De Figueiredo, Learning rules for neuro-controller via simultaneous perturbation, IEEE Trans. Neural Networks 8 (1997) 1119–1130.
[15] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: the RPROP algorithm, Proceedings of the International Conference on Neural Networks, Vol. 1, San Franscisco, CA, 1993, pp. 586–591.
[16] J. Sietsma, R.J.F. Dow, Neural net pruning—why and how, Proceedings of the IEEE International Conference on Neural Networks, Vol. 1, San Diego, CA, USA, 1998, pp. 325–333.
[17] J. Sietsma, R.J.F. Dow, Creating artificial neural networks that generalise, Neural Networks 4 (1991) 49–67.
[18] J.C. Spall, A stochastic approximation technique for generating maximum likelihood parameter estimates, Proceedings of the American Control Conference 1987, pp. 1161–1167.
[19] J.C. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, IEEE Trans. Automat. Control 37 (1992) 332–341.
[20] N.K. Treadgold, T.D. Gedeon, Exploring constructive cascade networks, IEEE Trans. Neural Networks 10 (1999) 1335–1350.
[21] Z. Wang, C.D. Massimo, A.J. Morris, A procedure for determining the topology of feedforward neural networks, Neural Networks 7 (2) (1994) 291–300.
[22] Z. Wang, M.T. Tham, A.J. Morris, Multilayer neural networks approximated canonical decomposition of nonlinearity, Int. J. Control 56 (1992) 655–672.
[23] J.Y.F. Yam, T.W.S. Chow, Extended least squares based algorithm for training feedforward networks, IEEE Trans. Neural Networks 8 (1997) 806–810.
[24] S. Young, T. Downs, CARVE—a constructive algorithm for real-valued examples, IEEE Trans. Neural Networks 9 (1998) 1180–1190.
[25] 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 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, Government of India. His area of interest include Neural Networks and Applications.