Improved backpropagation training algorithm using conic section functions

Conference Paper · July 1997

DOI: 10.1109/ICNN.1997.616178 · Source: IEEE Xplore

2 authors:

Tülay Yildirim
Yildiz Technical University
149 PUBLICATIONS **1,327** CITATIONS
SEE PROFILE

John Marsland
Liverpool John Moores University
47 PUBLICATIONS **249** CITATIONS
SEE PROFILE

Improved Back Propagation Training Algorithm Using Conic Section Functions

TÜLAY YILDIRIM[1] and JOHN S. MARSLAND
Dept. of Electrical Eng. and Electronics
University of Liverpool
Brownlow Hill, PO.BOX.147
Liverpool L69 3BX, UK
e-mail:T.Yildirim@liv.ac.uk and marsland@liv.ac.uk

ABSTRACT

A new training algorithm composed of a propagation rule which contains MLP and RBF parts to improve the performance of back propagation is proposed. The network using this propagation rule is known as a Conic Section Function Network (CSFN). This network allows to convert the open decision boundaries in an MLP to closed ones in an RBF, or vice versa. It reduces the number of centres needed for an RBF and the hidden nodes for an MLP. It is important since this work is aimed at designing a VLSI hardware neural network. Furthermore, it converges to a determined error goal at lower training epochs than an MLP. The performance of an MLP trained back propagation and also fast back propagation using adapted learning rates, an RBFN, and the proposed algorithm is compared using Iris plant database. The results show that the introduced algorithm is much better than the others in most cases, in terms of not only training epochs but also the number of hidden units and centres.

1.Delta rule for a conic section function network (CSFN)

The general equation for a Conic Section Function Network (CSFN) can be expressed as follows [2,5,6]

$$y_{pj}=\sum_{i=1}^{n+1}(a_{p_i}-s_{ij})w_{ij}-\cos\omega_j\,\|a_p-s_j\| \qquad (1)$$

where $a_{pi}=x_{pi}$ if unit i is an input unit, s_{ij} are the centres for the RBF network, w_{ij} are the weights in an MLP, 2ω is an opening angle which can be any value in the range $[-\pi/2,\pi/2]$ and determines the different forms of the decision borders, i and j are the indices referring to units in the input and the hidden layer and p refers to the number of the patterns. In the work presented here the output layer has weights only and no RBF contribution.

2.Derivation of the parameters

2.1.Updating weights:

The general formula for delta training/learning weight adjustment for a single layer network and the chain rule is applied [4]. Assuming a unipolar activation function, the weight update can be expressed for output units as follows

$$\Delta_p w_{ji}=\eta a_{pj}(1-a_{pj})(t_{pj}-a_{pj})(a_{p_i}-s_{ij}) \qquad (2)$$

and for hidden layer units as:

$$\Delta_p w_{ji}=\eta a_{pj}(1-a_{pj})A.(a_{p_i}-s_{ij}) \qquad (3)$$

where

$$A=-\sum_k \delta_{pk}\left[w_{ik}-\cos\omega.\frac{a_i-s_i}{\|a_p-s_j\|}\right] \qquad (4)$$

2.2.Updating Centres:

The same procedure is applied for updating the centres. The centre update for hidden layer units can be written as

[1]On leave from Yıldız Technical University , İstanbul, Türkiye.

0-7803-4122-8/97 $10.00©1997 IEEE

$$\Delta_p s_{ij} \propto a_{pj}(1-a_{pj}).A.(-w_{ij}+\cos\omega\frac{a_i-s_i}{\|a_p-s_j\|}) \qquad (5)$$

2.3.Updating Opening Angle:

The rule to update the opening angle ω is the same as the procedure for the weight and centre adjustment. The angle update is given for the hidden layer units by

$$\Delta_p \omega_j \propto a_{pj}(1-a_{pj}).A.\sin\omega_j\|a_p-s_j\| \qquad (6)$$

3.Training

3.1.IRIS Plant Database

Simulations were run with the Iris plant database [3] which is perhaps the best known database to be found in the pattern recognition literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of Iris plant, Setosa, Versicolor, and Virginica. One class, Setosa, is linearly separable from the other two; the latter are not linearly separable from each other. A training set and a test set were formed using the 150 samples. The training set contains 120 patterns, 40 from each pattern class. The remaining 30 patterns were used to test the training algorithms.

3.2.Summary of the basic training algorithm

The training occurs in two distinct stages: Stage A placing the centres and Stage B updating the parameters by back propagation.

Step A1: The number of centres and Sum-Squared Error (SSE) chosen. First layer weights are set to zero and the opening angle, ω is started from π/4.

Step A2: Training step starts here. A new centre is determined from the input set using the orthogonal least squares algorithm [1]. The output of the hidden layer is computed and used with the training set to initialize the output layer weights. The output of the second layer is computed.

Step A3: Sum-Squared Error is calculated.

Step A4: Steps A2 and A3 are repeated for the required number of centres.

Step B1: The error signal vectors δ, of both layers are calculated.

Step B2: Output layers weights are adjusted by the back propagation algorithm (Eq.2).

Step B3: Hidden layer weights, centres, and opening angle are updated (Eq.3,4,5,6).

Step B4: The outputs of the layers are calculated.

Step B5: New SSE is computed. If this is larger than the error goal, go to step B1, otherwise, terminate the training session.

3.3.Methods for training

Two different algorithms for training the CSFN were proposed. The first algorithm is based on updating the weights, the centres, and the opening angle ω in the same epoch as described above. The second algorithm uses the same Stage A (placement of centres) but differs in the second stage as follows.

Step C1: A predetermined number (typically 8) of training epochs (i.e. Stage B) is executed but only weights are updated.

Step C2: One training epoch is executed where only the opening angles ω are adjusted.

Step C3: Another training epoch is performed to update the centres.

4.Software simulation results

Fig 1. gives a comparison of different training algorithms with the learning rate lr=0.05 and error goal eg=2.5 for 6 centres using Iris plant database and Fig.2 shows the training results for the first algorithm using different number of centres (no centre updating).

Table 1 and 2 show the results of first and second algorithm for CSFN. As can be seen from Table 2, second algorithm is better than the first because too many parameters are changing in each step in first algorithm and training error sometimes increases due to this. Indeed the first algorithm converges faster if no centre updating occurs indicating that the centre initialization is more than adequate. For 6 centres, standard MLP needs 4000 to 30000 epochs while the second algorithm needs only 561 epochs. First algorithm is also better with 2951 epochs for the same number of centres. MLP trained with fast back propagation gives some better results than first algorithm, but none better than second one. It requires a minimum of 1000 epochs depending on random start (convergence sometimes does not occur) while the new network is not dependent on random initialization. CSFN needs fewer centres than RBF. It requires only 6 centres to reach the same error goal as an RBF with 32 centres. The number of centres and the hidden units used is important since this work is aimed at designing a VLSI hardware neural network.

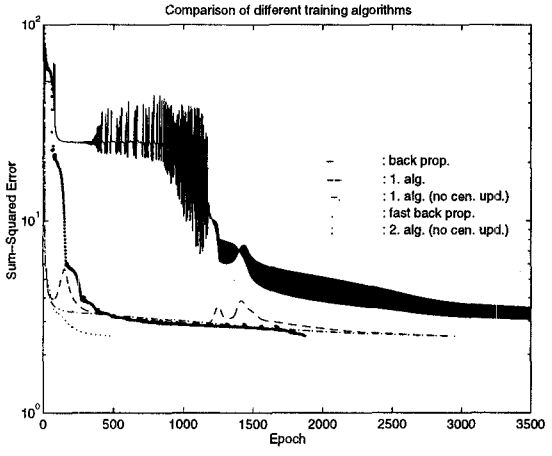

Figure 1. comparison of training algorithms

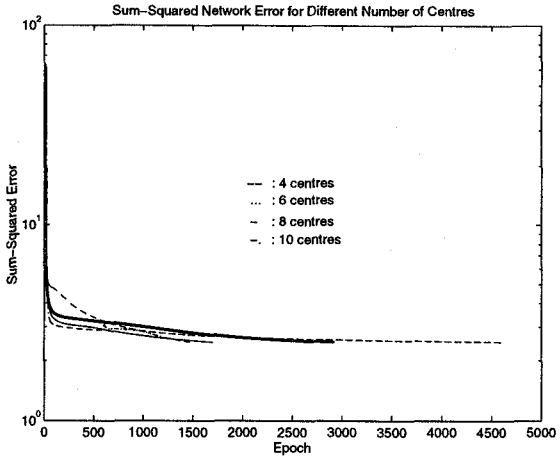

Figure 2. First training algorithm using different number of centres.

Number of centres	Epochs (centre updated)	Epochs (no centre updated)
4	Does not converge	4622
5	7903	4316
6	2951	2912
7	8479	2813
8	8230	1688
9	4172	1877
10	Not tried	1467

Table 1. Training epochs with different number of centres of CSFN (learning rate lr=0.05) for the first algorithm.

Number of centres	lr=0.01		lr=0.03		lr=0.04		lr=0.05		lr=0.07	
	cent. upd.	no c.up.	cent. upd.	no c.up.	cent. upd.	no c.up.	cent. upd.	no c.up.	cent. upd.	no c.up
5	1824	2490	663	914	592	707	663	503	868	1984

Table 2. Training epochs with different number of centres and different learning rates for the second algorithm

5.Conclusions

The performance of MLP trained with back propagation and fast backpropagation with adaptive learning rates and an RBFN using MATLAB Neural Network Toolbox, and the proposed algorithm is compared using Iris plant database. The results show that the new algorithm is much better than the others in most cases, in terms of not only training epochs but also the number of hidden units and centres since the decision boundaries can match the real data more closely. The CSFN converges to a determined error goal at lower training epochs than an MLP and is more stable than an MLP with random initialization.

References

[1] S.Chen, C.F.N.Cowan, P.M.Grant. " Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks", *IEEE Transactions on Neural Networks*, Vol.2, No.2, March 1991, 302-309.

[2] G.Dorffner. "Unified framework for MLPs and RBFNs: Introducing conic section function networks", *Cybernetics and Systems: An International Journal*, 25: 511-554, 1994.

[3] R.A.Fisher. "The use of multiple measurements in taxonomic problems", *Annual Eugenics*, 7, Part II, 179-188 (1936); also in "Contributions to mathematical Statistics", John Wiley, NY, 1950.

[4] D.E.Rumelhart, G.E.Hinton, R.J.Williams. "Learning internal representations by error propagation", in *Parallel Distributed Processing*, ed. D.E. Rumelhart and J.L. McClelland, Vol.1, Cambridge, MA: MIT Press, 1987, 318-362.

[5] T.Yıldırım, J.S.Marsland. "An RBF/MLP hybrid neural network implemented in VLSI hardware", *Conf. Proc. of NEURAP'95 Neural Networks and Their Applications*, Marseilles, March 20-22, 1996, pp.156-160.

[6] T.Yıldırım, J.S.Marsland. "Conic section function network synapse and neuron implementation in VLSI hardware", *Proc. of the IEEE Int. Conf. on Neural Networks*, ICNN, 1996, Washington DC, USA, Vol. 2, pp. 974-979.