Available online at www.sciencedirect.com

ScienceDirect

AASRI Procedia 6 (2014) 19 – 25

AASRI Procedia

www.elsevier.com/locate/procedia

2013 2nd AASRI Conference on Computational Intelligence and Bioinformatics

Interval Based Weight Initialization Method for Sigmoidal Feedforward Artificial Neural Networks

Sartaj Singh Sodhi[a]*, Pravin Chandra[a]

[a]*University School of ICT, GGS Indraprastha University, Sector 16C, Dwarka, Delhi - 110078, INDIA*

Abstract

Initial weight choice is an important aspect of the training mechanism for sigmoidal feedforward artificial neural networks. Usually weights are initialized to small random values in the same interval. A proposal is made in the paper to initialize weights such that the input layer to the hidden layer weights are initialized to random values in a manner that weights for distinct hidden nodes belong to distinct intervals. The training algorithm used in the paper is the Resilient Backpropagation algorithm. The efficiency and efficacy of the proposed weight initialization method is demonstrated on 6 function approximation tasks. The obtained results indicate that when the networks are initialized by the proposed method, the networks can reach deeper minimum of the error functional during training, generalize better (have lesser error on data that is not used for training) and are faster in convergence as compared to the usual random weight initialization method.

Peer-review under responsibility of Scientific Committee of American Applied Science Research Institute

Keywords: Weight Initialization; Sigmoidal Feedforward Artificial Neural Networks; Artificial Neural Networks.

1. Introduction

Weight and thresholds (collectively known as weights) initialization is an important aspect of sigmoidal feedforward artificial neural network (SFFANN) training. Weights are usually initialized to small uniform

* Corresponding author. Tel.: +919873348666;
E-mail address: [b]sartaj@ipu.ac.in, sartajsodhi@yahoo.com; [c]pchandra@ipu.ac.in, chandra.pravin@gmail.com

Peer-review under responsibility of Scientific Committee of American Applied Science Research Institute
doi:10.1016/j.aasri.2014.05.004

random values in the interval [-δ, δ] (where usually δ ε (0,1]). Training of SFFANN has been shown to be sensitive to initial weight choice [1]. The random initialization of weights was proposed by Rumelhart et al. 1987 [2]. They observed that if weights are initialized to equal values they move in tandem/groups during training, and to break this "weight-symmetry", the random weight initialization method was proposed. Moreover, it has been suggested in literature that hidden nodes act as feature detectors [3]. Thus, it is desirable that each hidden layer node act as a detector of a separate/distinct feature.

Thus, it becomes meaningful to initialize the weights leading into the hidden node (including the threshold of the node), in a manner that for distinct hidden nodes, the weights and thresholds belong to distinct region of the initialization interval. This would lead to the net input to distinct nodes being different by design, and during training allow the nodes to adapt to become detectors of distinct features. This, hypothesis is enshrined in the proposed weight initialization method. The proposed method is compared against four random weight initialization methods (specifically, δ = 0.25, 0.50, 0.75 and 1.00), on a set of 6 function approximation tasks.

The paper is organized as follows: Section 2 presents the design of experiments including the architecture of the networks used, the function approximation tasks, the random weight initialization method description and the proposed weight initialization method. Section 3 presents the results and discussion while conclusions are presented in Section 4.

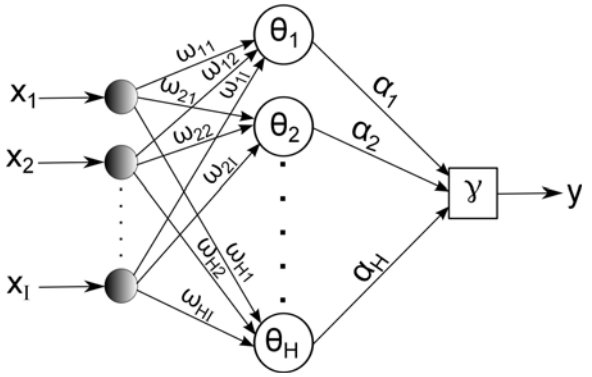

Fig. 1. Schematic diagram of the SFFANN.

2. Design of Experiment

The universal approximation results for SFFANNs assert that there exist network with sufficient number of sigmoidal output function hidden nodes can approximate any continuous function arbitrarily well [4-6]. Thus, the networks used in this study have use one hidden layer of sigmoidal nodes. A sigmoidal function, in the context of this paper, is a function that has the following property:

Definition 1: A continuous, monotonically increasing and differentiable function $\sigma(x)$, where x is a real quantity, is sigmodial if and only if it satisfies the following:

$$\lim_{x\to\infty}\sigma(x)=1;\quad \lim_{x\to-\infty}\sigma(x)=-1 \qquad (1)$$

A function (out of the many functions that satisfy (1) and (2) is the *hyperbolic tangent* function (another function is a scaled version of the *arc tangent* function). The sigmoidal function used in this study as the output function / activation function / squashing function of the hidden nodes is the *hyperbolic tangent* function:

$$\sigma(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2}$$

The hyperbolic tangent function is an anti-symmetric function, and is used as preference has been shown in literature for anti-symmetric function as compared to asymmetric activation function [7]. The schematic diagram of the networks used in this study is represented in Fig.1. The number of inputs to the network is represented by I, the number of nodes in the hidden layer (such a node is called hidden node) is represented by H, and the number of outputs of the network is taken as 1. The *j*th input is represented by x_j and the output of the network is represented by *y*. The connection strength between the *i*th hidden node and the *j*th input is represented by ω_{ij}, the threshold of the *i*th hidden node is represented by θ_i, then the net input to the *i*th hidden node is given by:

$$n_i = \sum_{i=1}^{I} \omega_{ij} x_j + \theta_i \tag{3}$$

Thus, the output from the *i*th hidden node can be represented as $h_i = \sigma(n_i)$. The connection strength between the *i*th hidden node and the output node is α_i, while the threshold of the output node is γ. The output function of the output node is a pure linear function, that is the net input to the output node is transferred as the output from the network:

$$y = \sum_{j=1}^{H} \alpha_i h_i + \gamma \tag{4}$$

2.1. Function Approximation Tasks

The approximation of the following 6 functions are used as the learning task in the experiments:

$$f_1(x) = \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x-0.9)^2 + 0.4} - 6; \quad x \in [0,1] \tag{5}$$

$$f_2(x_1, x_2) = 3(1-x_1)^2 e^{(-x_1^2 - (x_2+1)^2)} - 10(x_1/5 - x_1^3 - x_2^5) e^{-x_1^3 - x_2^3} - e^{-(x_1+1)^2 - x_2^3}/3; \quad x_1, x_2 \in [-3,3] \tag{6}$$

$$f_3(x_1, x_2) = e^{x_1 \sin(\pi x_2)}; \quad x_1, x_2 \in [-2,2] \tag{7}$$

$$f_4(x_1, x_2) = 1.3356(1.5(1-x_1) + e^{2x_1 - 1} \sin(3\pi(x_1 - 0.6)^2) + e^{3(x_2 - 0.5)} \sin(4\pi(x_2 - 0.9)^2); \quad x_1, x_2 \in [0,1] \tag{8}$$

$$f_5(x_1, x_2) = 1.9(1.35 + e^{x_1} \sin(13(x_1 - 0.6)^2) e^{-x_2} \sin(7x_2); \quad x_1, x_2 \in [0,1] \tag{9}$$

$$f_6(x_1, x_2) = \frac{1 + \sin(2x_1 + 3x_2)}{3.5 + \sin(x_1 - x_2)}; \quad x_1, x_2 \in [-2,2] \tag{10}$$

The function f_1 is the *humps.m* sample function in Matlab [8], function f_2 is the *peaks.m* function in Matlab [8] while the functions of eq. (7)-(10) have been taken as a benchmark problem from Cherkassky et al., 1996 [9].

The number of hidden nodes was decided on the basis of exploratory experiments conducted in which the number of hidden nodes was varied between 2 to 30 in steps of 1, for 100 epochs of training. The first network of the minimal size that gave satisfactory error during training was taken as the appropriate size for the experiment. The architecture of the networks used for the approximation of the 6 functions is summarized in Table 1.

Table 1. Network architecture summary for the function approximation tasks.

Function	Number of Inputs(I)	Number of Hidden Nodes (H)	Number of Outputs
f_1	1	8	1
f_2	2	15	1
f_3	2	10	1
f_4	2	10	1
f_5	2	12	1
f_6	2	25	1

2.2. Random Weight Initialization Methods

The proposed weight initialization method is compared against 4 random weight initialization routines. The random weight initializations are labeled as WTR_i, where i is in {1,2,3,4}, the i's correspond to the weight interval parameter δ in {0.25,0.50,0.75,1.00}.

2.3. Proposed Weight Initialization Method

The random weight initialization routines distribute all weights and the thresholds over the interval [-δ, δ]. The proposed method for weight initialization distributes the weights leading into the hidden nodes (including the hidden node threshold) in a manner such that no two distinct node have weights leading in that belong distinct regions of the interval used for initialization of the weights.

The proposed method is called the Interval Based Weight Initialization Method (IWI). The method IWI distributes the weights leading in to the ith hidden node in the interval [(2i-1)/(H-1), (2i+1)/(H-1)] (as uniform random numbers in the given interval). While the threshold of the ith hidden node is initialized to $2i/(H-1)$. The hidden nodes to the output node weights are initialized to deterministic values between [-C,C] where $C = H^{-1/2}$; that is, for the connection weight between ith hidden node and the output node the weight is $\alpha_i = -C + 2iC / (H-1)$. A similar mechanism, but in a random manner, for the weight initialization of weights based on the fan-in to a node is suggested in[7].

2.4. Experiment Procedure

The Resilient Backpropagation algorithm is used for training the network. For the purpose of training 200 input data sets are generated by uniform random sampling of the input domain of the function, and the corresponding output calculated from the function to create the training data set. For testing the generalization capability of the trained network(s), a similar set with 1000 data values is generated and called the test set.

For each task and each weight initialization method, 30 networks are trained for 1000 epochs. The average over the 30 networks of the mean squared error (MMSE) for the training set and the test set is reported together with the standard deviation of the mean squared error as a measure of the goodness of training (lower value of MMSE is better) and generalization capability, respectively.

Another set of experiments is performed to measure the convergence speed during training, wherein a goal equal to twice the worst MMSE achieved in the previous experiment is kept as a goal of training, and the number of epochs required is measured. Since the maximum epoch of training is kept at 1000, if a network does not converge during training, its epoch value is kept at 1001 (arbitrarily). This creates a small bias in the mean epoch value, but the number of non-convergent networks is also counted and reported for each instance of function approximation task and weight initialization method.

3. Result and Discussion

The result of the training experiment and the generalization experiment is summarized in Table 2. From the Table 2, it can be seen that among the random weight initialization methods (WTRs), there is no single method that gives the best result across the function approximation tasks. It can also be seen that the proposed weight initialization method (IWI) always leads to training and generalization errors that are smaller than any of the random weight initialization methods. We may infer that the proposed method of weight initialization leads to lower value of error after training on an average and up to a factor of 2 deeper minima of the error functional can be achieved on training by the proposed method (IWI), as the variation in the ratio of the best result for the WTR for a specific function approximation task to that of the proposed method lies in the interval [1.12, 2.86].

Moreover, from the generalization experiments we may infer that the networks trained after initialization by the proposed method, have better generalization behavior. That is, the error on data not used for training is lower for networks initialized by IWI. For the generalization experiment, the ratio of the best result for the random weight initialization method (WTR) to the result for the proposed method (IWI) lies between [1.08,2.71] across the function approximation task. Thus, for the generalization experiment also, the proposed method has error on an average across problems that is lower by a factor of about 2 for networks trained using the IWI method.

Table 2. Training and Generalization Data Summary. For the training and the generalization experiments the average mean squared error is shown and the figure in parenthesis is the standard deviation. All figures are $\times 10^{-3}$.

Function	Training					Generalization				
	WTR_1	WTR_2	WTR_3	WTR_4	IWI	WTR_1	WTR_2	WTR_3	WTR_4	IWI
f_1	2.44 (4.02)	0.89 (1.55)	0.90 (3.04)	0.40 (0.37)	0.14 (0.08)	2.65 (4.28)	0.98 (1.67)	0.98 (3.19)	0.46 (0.42)	0.17 (0.09)
f_2	13.43 (3.91)	13.08 (3.77)	14.46 (3.00)	14.74 (4.35)	7.84 (2.58)	31.17 (7.44)	27.84 (6.89)	28.77 (4.71)	26.25 (5.81)	24.27 (8.69)
f_3	2.70	2.88	2.73	2.68	1.28	3.91	4.18	3.94	3.85	2.13

	(0.82)	(0.66)	(0.78)	(0.80)	(0.35)	(1.18)	(0.94)	(1.04)	(1.12)	(0.59)
f_4	3.96	3.84	3.64	3.76	2.68	5.68	5.84	5.31	5.61	4.36
	(1.45)	(1.26)	(1.69)	(1.52)	(1.58)	(1.96)	(1.97)	(2.36)	(2.35)	(2.62)
f_5	7.89	8.80	6.66	5.98	3.97	13.43	13.98	12.14	11.88	9.10
	(5.06)	(5.77)	(4.14)	(3.36)	(3.24)	(6.72)	(7.13)	(5.97)	(5.02)	(4.72)
f_6	23.50	9.22	17.71	8.25	3.91	48.67	26.24	41.45	25.98	13.88
	(46.55)	(3.44)	(34.13)	(2.18)	(1.69)	(77.39)	(6.69)	(60.52)	(3.78)	(3.88)

The summary of the experiments conducted for measurement of speed of convergence is shown in Table 3. From the data, we may infer that the proposed method leads to faster convergence in all case. No single random weight initialization method out of the 4 WTRs can be preferred on convergence speed, as for different function approximation task, a different random weight initialization routine may give better results. The ratio of the minimum average epochs required by any of the 4 random weight initialization method (WTR) to the average epochs required by the proposed method (IWI) for convergence to the specified goal lies in the interval [1.00, 3.19]. Thus, we may infer that the networks trained after initialization by the proposed methods (IWI), have a convergence speed that can be faster by a factor of about 2 as compared to the random weight initialized networks on an average..

Table 3. Convergence Speed Experiment Summary. The goal represents the goal for convergence speed experiment. For the convergence speed experiment the average epoch for convergence is shown the Statistic column. The number of non-convergent networks is also shown as NCN. Goal and the Statistic figures are $\times 10^{-3}$.

		WTR_1		WTR_2		WTR_3		WTR_4		IWI	
Function	Goal	Statistics	NCN	Statistics	NCN	Statistics	NCN	Statistics	NCN	Statistics	NCN
f_1	2.44	400.30	5	272.50	1	185.80	1	158.80	0	49.80	0
f_2	14.74	92.83	0	111.03	0	131.00	0	223.20	0	80.23	0
f_3	2.88	299.33	0	301.97	0	293.87	0	248.53	0	129.07	0
f_4	3.96	275.50	0	249.23	0	274.73	0	248.20	0	247.57	0
f_5	8.80	522.83	2	522.33	2	394.40	1	334.00	0	196.73	0
f_6	23.50	502.13	3	426.60	0	442.93	2	311.93	0	173.17	0

4. Conclusions

In the current work, a proposal for distribution of SFFANN input to hidden weight and thresholds of the hidden weight is made in a manner such that these weights associated with distinct hidden nodes lie in disjoint intervals. On a set of 6 function approximation task, the efficiency and efficacy of the proposed method is demonstrated. That is, networks that are initialized by the proposed method, can be trained to achieve deeper minima as compared to random weight initialization method; they generalize better and are faster in training.

References

[1] Kolen J. F., Pollack J. B., “Back propagation is sensitive to initial conditions,” in *Proc. of the 1990 conference on Advances in neural information processing systems 3*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 860–867.

[2] Rumelhart D. E., Hinton G. E., Williams R. J., "Learning internal representations by error propagation," in *Parallel Distributed Processing: Volume 1: Foundations*, D. E. Rumelhart, J. L. McClelland, and The PDP Research Group, Eds. Cambridge: MIT Press, 1987, pp. 318–362.
[3] Haykin S., *Neural Networks: A Comprehensive Foundations*. Englewood Cliffs, NJ: Prentice–Hall, 1999.
[4] Cybenko G., "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals, and Systems*, vol. 2, pp. 303–314, 1989.
[5] Funahashi K., "On the approximate realization of continuous mapping by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.
[6] Hornik K., Stinchcombe M., White H., "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.
[7] LeCun Y., Bottou L., Orr G. B., Muller K.-R., "Efficient backprop," in *Neural Networks: Tricks of the trade*, ser. LNCS:1524, G. B. Orr and K.-R. Muller, Eds. Berlin: Springer, 1998, pp. 9–50.
[8] The MathWorks Inc., "Matlab version R2013a," 2013.
[9] Cherkassky V., Gehring D., Mulier F., "Comparison of adaptive methods for function estimation from samples," *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 969–984, 1996.