Modified Cascade-2 Algorithm with Adaptive Slope Sigmoidal Function

Jaswinder Kaur [1] and Sudhir Kumar Sharma[2]
[1] Research Scholar, Ansal University, Gurgaon, New Delhi, India
[2] Professor, Department of Computer Science, IITM Janakpuri, New Delhi,India
jasukaur@rediffmail.com, sudhir_sharma99@yahoo.com

Abstract

Cascade-2 algorithm is a variant of well-known cascade-correlation algorithm that is widely investigated constructive training algorithm for designing cascade feedforward neural networks. This paper proposes a modified Cascade-2 algorithm with adaptive slope sigmoidal function (MC2AASF). The algorithm emphasizes on architectural adaptation and functional adaptation during learning. This algorithm is a constructive approach of designing cascade architecture. To achieve functional adaptation, the slope of the sigmoidal function is adapted during training. One simple variant is derived from MC2AASF is where the slope parameter of sigmoidal function used at the hidden layers' nodes is fixed to unity. Both the variants are compared to each other on five function approximation tasks. Simulation results show that adaptive slope sigmoidal function presents several advantages over standard fixed shape sigmoidal function, resulting in increasing flexibility, smoother learning, better generalization performance and better convergence.

Keywords: Adaptive slope sigmoidal function; Cascade-Correlation algorithm; Cascade-2 algorithm; Constructive neural networks; Dynamic node creation.

1 Introduction

Artificial neural networks have been successfully applied to problems in data processing, robotics, and numerical control of computer, decision making, and function approximation, classification and regression analysis.

FeedForward Neural network is a layered neural network in which the neurons are organized in the form of layers and the neurons in one layer get the input from the previous layer and feed their output to the next layer. Among the various types of neural networks feedforward neural networks (FFNNs) is used most widely.

The generalization capability and convergence time of supervised learning in FNNs depends on various factors such as choice of network architecture (number of hidden nodes and network topology), the choice of training algorithm and the choice of activation function of each node. This suggests the need for an algorithm that can find appropriate size of the network architecture automatically and that also learns the weights during training.

Constructive neural networks (CoNN) consist of minimum architecture in which hidden nodes are added one at a time incrementally.

Many constructive neural networks (CoNN) proposals for regression problems are given in [1]-[4]. Kwok and Yeung [1] survey the major CoNN algorithms for regression problems. In their proposed taxonomy that is based on the concept of a state-space search, group the algorithms into six different categories. Among these, the most popular for regression problems is the Cascade-Correlation algorithm (CCA) proposed by Fahlman and Lebiere [5] and next is the dynamic node creation (DNC) algorithm proposed by Ash [6]. The latter algorithm constructs a single hidden layer FNN automatically, whereas the former constructs cascade architecture during training.

In each phase, CCA adds one hidden node in a separate hidden layer at a time and hidden node is connected to all inputs as well as previously trained hidden nodes. After the training of input weights of current hidden node gets completed, it is connected to output nodes with input weights frozen and all inputs of output nodes are trained again. This algorithm has inspired many new variants and also has been used in the reinforcement learning methods.

The several variants of CCA algorithm and similar type of algorithms have been proposed from time to time in the literature. These algorithms differ from each other in various aspects, the connectivity patterns of the current hidden node i.e. cascade architecture or single hidden layer FNN, activation function used at hidden layers' node, objective function used for candidate node training, the optimization method used for training the individual hidden node, the stopping criteria for candidate node training and halting criteria for the node addition. Lastly, they can also be classified on the basis of how the connection weights are frozen and once again trained.

Cascade-2 algorithm [7] was also first proposed by Fahlman, who proposed the idea of CCA. Cascade-2 algorithm differs from CCA by training current hidden node to directly minimize the residual error rather than to maximize its covariance with residual error. Besides this, hidden node has adjustable output connections to all of the output nodes and all other things are common in both algorithms. Several authors have demonstrated that CCA is effective for classification tasks but not very successful on regression problems. This is because for its correlation term tending to drive the hidden node activations to their extreme values, thereby, making it hard for the network to produce a smoothly varying output [8]- [10].

Logistic activation function is widely used at hidden nodes in FNNs due to its nonlinear capability. In general, the slope parameter of sigmoidal function is fixed to unity prior to training and cannot be adapted to suit different problems during training. We can achieve a great nonlinear mapping capability if slope parameter of the sigmoidal function is adapted by the training data. In past many researchers had used adaptive slope sigmoidal function (ASSF) for fixed size FNNs and reported better generalization performance and faster learning with less number of hidden nodes [11]-[15]. The ASSF defined as follows:

$$g(x, \beta) = \frac{1}{1 + e^{-\beta x}};\ x \in \mathrm{R} \qquad (1)$$

where R is assumed as the set of real numbers and β is called slope parameter. For each hidden node, the value of the slope parameter is updated during the learning process. If the slope parameter is unity, then this activation function is equivalent to standard log-sigmoidal function.

In this paper we propose a modified Cascade-2 algorithm with adaptive slope sigmoidal function (MC2AASF).

The paper is organized as follows: In Section-2, we propose MC2AASF. Section-3presents the experimental design to compare the efficiency of the two variants. In Section-4, the results are presented and discussed. In Section-5, conclusions are presented.

2 The proposed algorithm

This section presents the MC2AASF for designing cascade architecture and differs from Cascade 2 algorithm in these four aspects:

1) MC2AASF starts the network with one hidden node. Input and output nodes are not directly connected.
2) MC2AASF algorithm uses only one objective function (squared error criterion) to train (input and output connection weights simultaneously) of each hidden node in one stage. One practical advantage of MC2AASF is that we do not need to switch between two different optimizations.
3) MC2AASF freezes both input and output connection weights of the each trained hidden node.
4) Stochastic gradient descent method is used for training the individual hidden node, thus we are not restricted to using batch mode.

The proposed MC2AASF algorithm focuses on both architectural and functional adaptation during the learning. The number of input and output nodes is decided according to the characteristic of a given problem. We formulate MC2AASF for regression problem. Without loss of generality, we consider minimal architecture has N_i nodes in input layer, onenode in each hidden and output layer. The output node has a linear activation function, while the hidden layers' node has ASSF defined in (1). There is a hidden node which is added in the current network and trained at a time and it does not change its weights (input and output) after training gets completed for current hidden node. The currently added hidden node is connected to all the input nodes as well as previously trained hidden nodes and connected to the output nodes and makes separate hidden layer in the form of cascade architecture just as in the case of Cascade-2 algorithm. During the training of the current hidden node, the input and output connection weights, slope parameter and bias of output node are trained by using gradient descent method in sequential mode, minimizing the squared error objective function [16].

Let iw_{ni} represents weight between the n-th hidden node and i-thinput while ow_k represents weight between the k-thhidden node and output node. The connection weight hiw_{nj} represents weight between the n-th hidden node and j-th previously trained hidden node. The connections weights iw_{n0} and ow_0 act as the biases for the n-th hidden node and output node, respectively. The biases of the hidden nodes and output node are represented using the 0-th auxiliary input x_0 and 0-th auxiliary hidden node O_0, respectively. The values of x_0 and O_0 are set to unity. The training pairs are represented by (x^p, f^p); $p = 1,2,.....,P$, where P is the number of training exemplars. The index p is always assumed to be present implicitly [16].

If x_i is the i-thcomponent of the input, then the total input for the n-thhidden node is as follows:

$$net_n = \sum_{i=0}^{N_i} iw_{ni}\ x_i + \sum_{j=1}^{n-1} hiw_{nj}\ O_j \tag{2}$$

The output for the n-th hidden node is as follows:

$$O_n = g(net_n, \beta_n) = \begin{cases} 1 & n = 0 \\ 1/\left(1 + e^{-\beta_n \mathrm{net}_n}\right) & n \geq 1 \end{cases} \tag{3}$$

A cascade network, having n hidden nodes implements the function

$$f_n(x) = \sum_{k=0}^{n} ow_k\ O_k = f_{n-1}(x) + F_n(x) \tag{4}$$

where $f_{n-1}(x)$ is the function implemented by the cascade architecture that had (n – 1) hidden nodes and where

$$F_n(x) = ow_n\ O_n + ow_0 \tag{5}$$

We canspecify the objective function for training the current n-th hidden node by (6) that is the squared error function on a per example basis.

$$S = \frac{1}{2}(f - f_n(x))^2 = \frac{1}{2}(f - f_{n-1}(x) - F_n(x))^2$$

$$= \frac{1}{2}(e_{n-1} - F(x))^2 \tag{6}$$

where e_{n-1} is the residual error that is left from the previously added hidden nodes (i.e. it is the desired output for the current n-th hidden node).

The cascade network is trained by using gradient descent method applied to the minimization of the objective function defined in (6) on a per pattern basis.

If w is any trainable parameter of the network, then its weight increment with momentum term is defined as follows:

$$\Delta w(p) = \alpha_w\ \Delta w(p-1) - \eta_w \frac{\partial S}{\partial w}(p) \tag{8}$$

where, $\alpha_w \in (0,1)$ is a constant, also known as momentum parameter and $\eta_w \in (0,1)$ is a constant, also known as the learning rate. Let $e = (e_{n-1} - F(x))$ be the residual error, then weight increment without indices defined as follows:

$$\Delta w = \alpha_w\ \Delta w + \eta_w\ e \frac{\partial F}{\partial w} \tag{9}$$

We can easily derive the following results:

$$\frac{\partial F}{\partial ow_k} = O_k; \qquad k = 0, n \tag{10}$$

$$\frac{\partial F}{\partial iw_{ni}} = ow_n \frac{\partial O_n}{\partial net_n} x_i; \qquad i = 0,1,\mathrm{K}\ , N_i \tag{11}$$

$$\frac{\partial F}{\partial hiw_{nj}} = ow_n \frac{\partial O_n}{\partial net_n} O_j; \qquad j = 1,\mathrm{K}\ , n-1 \tag{12}$$

$$\hat{O}_n = \frac{\partial O_n}{\partial net_n} = \begin{cases} \beta_n\ O_n\ (1 - O_n); & n \geq 1 \\ 0; & n = 0, \end{cases} \tag{13}$$

$$\frac{\partial F}{\partial \beta_n} = ow_n \frac{\partial O_n}{\partial \beta_n} = ow_n \frac{\hat{O}_n}{\beta_n} net_n; \tag{14}$$

We are able to write the weight update rules as for $n = 1,2\ldots,N_h;$ where N_h is the maximum number of hidden nodes added in cascade network architecture and $p = 1,2,\ldots..,P$

$$\Delta ow_k = \alpha_w\ \Delta ow_k + \eta_w\ e\ O_k; \qquad k = 0, n \tag{15}$$

$$\Delta iw_{ni} = \alpha_w\ \Delta iw_{ni} + \eta_w\ e\ ow_n\ \hat{O}_n\ x_i; \quad i = 0,1,\mathrm{K}\ , N_i \tag{16}$$

$$\Delta hiw_{nj} = \alpha_w\ \Delta hiw_{nj} + \eta_w\ e\ ow_n\ \hat{O}_n\ O_j; \quad j = 1,2,\mathrm{K}\ , n-1 \tag{17}$$

$$\Delta \beta_n = \alpha_\beta\ \Delta \beta_n + \eta_\beta\ e\ ow_n \frac{\hat{O}_n}{\beta_n} net_n \tag{18}$$

The proposed algorithm is empirically compared against the algorithm in which the slope parameter is kept constant (equal to unity) i.e that is not updated at all (MC2A).

3 Experimental Design

The following five two-dimensional regression functions are used to compare the learning behavior of MC2AASF and MC2A. These functions have been studied in [2], [4]:

(a) Simple interaction function (SIF)

$$y = 10.391((x_1 - 0.4)(x_2 - 0.6) + 0.36). \tag{19}$$

(b) Radial function (RIF)

$$y = 24.234\left(\left((x_1-0.5)^2+(x_2-0.5)^2\right)(0.75 - (x_1-0.5)^2-(x_2-0.5)^2\right) \quad (20)$$

(c) Harmonic function (HF)

$$y = 42.659\left(0.1+(x_1-0.5)\left(0.05+(x_1-0.5)^4 - 10(x_1-0.5)^2(x_2-0.5)^2+5(x_2-0.5)^4\right)\right) \quad (21)$$

(d) Additive function (AF)

$$y = 1.3356\left(1.5(1-x_1)+e^{2x_1-1}\sin\left(3\pi(x_1-0.6)^2\right) + e^{3(x_2-0.5)}\sin\left(4\pi(x_2-0.9)^2\right)\right) \quad (22)$$

(e) Complicated interaction function (CIF)

$$y = 1.9\left(1.35+e^{x_1}\sin\left(13(x_1-0.6)^2\right)e^{-x_2}\sin(7x_2)\right) \quad (23)$$

For each function 1,450 uniformly distributed random points were generated in the two-dimensional space $0 \le x_1, x_2 \le 1$.

The used data was normalized in the interval [-1, +1] and then partioned into the training set (TRS), validation set (VS), and testing set (TS). The first 225 exemplars were used for TRS, the following 225 exemplars were used for VS and the final 1,000 exemplars for TS.

30 independent runs were performed for each regression function. For each trial, initial weights sets were generated in the interval [-1, +1] at random.

After a series of experiments, we set the values of the parameters as constant for all regression functions. Hidden nodes were added up to a maximum of 15. Each individual hidden node was trained up to a maximum of 300 epochs. The learning rate η_w and momentum constant α_w for the weights were 0.1 and 0.8, respectively. The learning rate η_β and momentum constant α_β for slope parameter were 0.1 and 0.8, respectively. We started slope parameter with a value of unity and updated it so that it reached its optimal value during training. Each trained hidden node acquired different optimal value in the range $(\beta_{\min}, \beta_{\max}) = (0.1,10)$ in our simulation.

The final performance of selected network (same configuration of the network, where validation MSE was minimum) was measured from the TS.

4 Results and discussions

The results of the 300 experiments conducted are presented in this section. For drawing summary, we considered all experiments that are executed. For brevity, we present summary data in Table 1. For comparing of the two variants of the discussed algorithm, the following measures are used.

1) The minimum of the MSE (MINMSE) on test set achieved in all the experiments for regression function is in the third column.
2) The maximum of the MSE (MAXMSE) on test set achieved in all the experiments for regression function is in the fourth column.
3) The mean of the MSE (MMSE) on test set achieved in all the experiments for regression function is in the fifth column.
4) The standard deviation of the MSE (STDMSE) on test set achieved in all the experiments for the regression function is in the sixth column.

TABLE 1. SUMMARY RESULTS OF THE TWO VARIANTS OF THE PROPOSED ALGORITHMS.

Function	Algorithm	MINMSE 10^{-2}	MAXMSE 10^{-2}	MMSE 10^{-2}	STDMSE 10^{-2}	MINHN	MHN	STDHN	RMMSE
SIF	MC2A	0.1855	0.7594	0.2694	0.1029	5	12.667	2.783	1.141
	MC2AASF	0.1147	0.3799	0.2361	0.0696	9	13.433	1.995	
RF	MC2A	0.8914	4.0576	1.8828	0.7022	6	13.600	2.078	1.413
	MC2AASF	0.7789	3.0431	1.3326	0.5474	7	13.000	2.304	
HF	MC2A	5.1611	6.7188	6.2061	0.4368	1	5.333	4.880	2.200
	MC2AASF	1.7654	4.4761	2.8211	0.7366	6	11.867	2.849	
AF	MC2A	1.5372	8.6172	3.0695	1.9604	8	13.200	2.265	1.079
	MC2AASF	1.3102	5.0254	2.8445	0.9277	6	13.200	2.578	
CIF	MC2A	2.8277	3.9131	3.1953	0.2713	3	11.433	3.936	1.309
	MC2AASF	1.6035	3.3129	2.4401	0.5368	8	13.233	1.888	

5) The minimum number of hidden nodes (MINHN) found by the algorithm in all the experiments for regression function is in the seventh column.
6) The mean number of hidden nodes (MHN) found by algorithm in all the experiments for regression function is in the eighth column.
7) The standard deviation in the number of hidden nodes (STDHN) found by algorithm in all the experiments for regression function is in the ninth column.
8) Ratio of the mean of the MSE (RMMSE) in all the experiments of the MC2A to the MC2AASF is in the tenth column.

The MC2AASF gives lower MSE than the variant MC2A for all the regression functions. In order to observe the significance difference in generalization performance, we performed a *t*-test. The null hypothesis is rejected with 95% confidence level for all the regression functions. It is inferred that there is a significant difference in MSE achieved by the two variants of the algorithm. The RMMSE is greater than one for all tasks. All these show that the generalization performance and convergence capability of MC2AASF is better than the MC2A.

The common feature of the both variants is the freezing of previously trained nodes for the sake of computational efficiency and avoids the moving-target problem. Since the local error vector is already computed as a necessary part of the weight update equation, to update the slope parameter,*it* does not impose any significant computational burden for the variant MC2AASF.

5 Conclusion

In this paper, we proposed a modified Cascade-2 algorithm with adaptive slope sigmoidal function. This algorithm is a constructive approach of building cascade architecture and thus obviating the need for *a priori* guessing the network architecture.The functional adaptation is achieved through the adaptive slope parameter of sigmoidal function that prevents the nonlinear nodes from saturation and increases their learning capabilities.The algorithm determines not only the optimum number of hidden nodes in cascade architecture, as also the optimum slope parameter for them. From the results obtained, we may conclude that the MC2AASF gives better generalization performance and smoother learning than the variant MC2A. The proposed constructive training algorithm can be used for forecasting of electric energy demand for a smart city.

References

[1] T. Y. Kwok, D. Y. Yeung, "Constructive Algorithms for Structure Learning in feedforward Neural Networks for Regression Problems," IEEE Transactions on Neural Networks, vol. 8, no. 3, pp 630-645, (1997).

[2] T.Y. Kwok, D.Y.Yenug, "Objective functions for training new hidden units in constructive neural networks," IEEE Transactions on Neural Networks, vol. 8, no. 5, pp 1131-1148 (1997).

[3] J. J. T. Lahnajarvi, M. I. Lehtokangas, and J. P. P. Saarinen, "Evaluation of constructive neural networks with cascaded architectures," Neurocomputing, vol. 48, pp 573-607 (2002).

[4] L. Ma and K. Khorasani, "New training strategies for constructive neural networks with application to regression problems," Neurocomputing, vol. 17, pp 589-609 (2004).

[5] S. E. Fahlman and C. Lebiere, "The cascade correlation learning architecture," Advances in Neural Information Processing System 2, D. S. Touretzky, Ed. CA: Morgan Kaufmann, pp 524-277 (1990).

[6] T. Ash, "Dynamic node creation in backpropagation networks," Connection Science, vol. 1, no. 4, pp 365-375 (1989).

[7] L. Prechelt, "Investigation of the cascor family of learning algorithms," Neural Networks 10 (5), pp 885-896 (1997).

[8] S. E. Fahlman and J. A. Boyan, "The Cascade 2 Learning Architecture," Technical Report(forthcoming), CMU-CS-94-100, Carnegie Mellon University (1994)

[9] M. C. Nechyba and Y. Xu, "Neural network approach to control system identification with variable activation functions," *IEEE International Symposium on Intelligent Control*, Columbus, Ohio, USA (1994).

[10] J. N. Hwang, S. Shien and S. R. Lay, "The Cascade – Correlation Learning: A Projection Pursuit Learning Perspective," In *IEEE Transactions on Neural Networks*, vol. 7, no. 2, (1996).

[11] T. Yamada, T. Yabuta, "Remarks on a neural network controller which uses an auto-tuning method for nonlinear functions," IJCNN, Vol. 2, pp 775-780 (1992).

[12] Z. Hu and H. Shao, "The study of neural network adaptive control systems," control and Decision, vol. 7, pp 361-366 (1992).

[13] C. T. Chen and W. D. Chang, "A feedforward neural network with function shape autotuning," Neural Networks, Vol. 9, issue (4), pp 627-641(1996).

[14] S. Xu and M. Zhang, "A novel adaptive activation function," In Proc. Int. JointConf. Neural Networks, vol. 4, pp 2779-2782 (2001).

[15] P. Chandra ,Y. Singh, "An activation function adapting training algorithm for sigmoidal feedforward networks," Neurocomputing, pp 429-437(2004).

[16] S. K. Sharma and P. Chandra. "An adaptive slope sigmoidal function cascading neural networks algorithm", *Emerging Trends in Engineering and Technology (ICETET), 2010* 3rd International Conference on IEEE, (2010).