ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Design of Airfoil using Back propagation Training with Adaptive Learning Rate

[1]Thinakaran Kandasamy and [2]Rajasekar Rajendran

[1]*Sri Venkateswara College of Engineering & Technology, Computer Science, Thiruvallur, Tamil Nadu, India.*
[2]*Excel Engineering College, Aeronautical Engineering, Pallakkapalayam, Tamil Nadu, India.*

ARTICLE INFO

Article history:
Received 12 March 2015
Accepted 28 April 2015
Available online 5 May 2015

Keywords:
linear error, nonlinear error method, airfoil design, back propagation, adaptive learning rate

ABSTRACT

Here, we proposed new adaptive leaning rate algorithm to increase the convergence speed of Back propagation learning to design the airfoil. The aerodynamic force coefficients corresponding to series of airfoil are stored in a database along with the airfoil coordinates. A feed forward neural network is created with aerodynamic coefficient as input to produce the airfoil coordinates as output. In the proposed algorithm, adaptive learning rate for output layer determined by differential of output value and for the hidden layer, we calculated the adaptive learning rate by calculating differential of linear and non linear errors. Results indicate that this approach greatly enhances the training of artificial neural network and may accurately predict airfoil profile

© 2015 AENSI Publisher All rights reserved.

To Cite This Article: Thinakaran Kandasamy and Rajasekar Rajendran, Design of Airfoil using Back propagation Training with Adaptive Learning Rate. ***Aust. J. Basic & Appl. Sci.,*** *9(11): 117-123, 2015*

INTRODUCTION

The aim of this paper is to generate geometry for airfoil using inverse design method with high convergence speed and with minimal error. There are many inverse techniques in use, for example, hodo-graph methods for two-dimensional flows (Hoist *et al*,1982). The inverse design of back propagation takes long time to converge. Some focused on better function and suitable learning rate and momentum (R. A. Jacobs *et al*, 1988). To design fast algorithm, Abid *et al*. proposed a new algorithm by minimizing sum of squares of linear and nonlinear errors for all output (S. Abid *et al*,1991). Jeong *et al*. proposed learning algorithm based on first and second order derivatives of neural activation at hidden layers (Jeong S.Y *et al* ,2000). Usually two parameters called learning rate and momentum factor are used for controlling the weight adjustment along the descent direction.

If the learning rate is very large, then the learning may become unstable. If it is small, then often is very slow for practical applications which leads to finding of fast learning algorithms (Sha D *et al*,1999).One of the issues when designing a particular neural network is to calculate proper weight for neural activities. These are obtained from the training process of neural network. Specifically, the process of obtaining appropriate weight in a neural network design utilizes two set of equations. First, the feed forward equation is used to calculate the error function. The feedback equation is next used to calculate the gradient vector. This gradient vector is used for defining search directions in order to calculate weight change. The linear and nonlinear error introduces certain modification in directions in order to speed up the convergence. But searching for global minimum may be trapped at local minima during gradient descent, so fast convergence may not be guaranteed. To solve these problems adaptive learning algorithms have been developed. Jeong and Lee(2009) have proposed an adaptive algorithm based on first and second order derivatives of neural activation at hidden layers. Zhihong Man(2006) have proposed variable learning rate algorithm using Lyapunov method. Iranmanesh and Mahdavi(2009) have proposed a learning method using differential adaptive learning rate. In each iteration, the learning rate of the output layer computed by differentiating the error of the output layer. S.Jeyaseeli (2011) describes an adaptive learning rate algorithm based on differential linear and non linear errors to speed up the learning process of the neural network.

We proposed a new adaptive learning rate in which derivatives of the corresponding neuron's activation function are added and then divided by total number of output neurons to get the adaptive learning rate for the output layer. Linear and nonlinear error of each hidden neuron is multiplied with its corresponding output layer weights

Corresponding Author: Thinakaran, Sri Venkataeswara Engineering College & Technology, Thiruvallur, India,.
Ph: 9894475486, E-mail: thina85@rediffmail.com

separately and then added. Then the value is divided by number of hidden neurons. The differentiation of the sigmoidal function of this value is used as a learning rate for the hidden layer. The efficiency of the proposed algorithm in terms of time and epochs is shown by simulating the problems. And proved that the efficiency is highly improved using this adaptive learning rate method.

2. Neural Network Training method:

An ANN consists of a large number of highly interconnected neurons. Neural networks offer a very powerful and general framework for representing nonlinear mappings from several input variables to several output variables. The general architecture of a two layer neural network with feed-forward connections and one hidden layer is shown in Figure 1. The weighted sum of the inputs must reach the threshold value for the neuron to generate the output. It is normally very difficult to interpret the values of the connecting weights wij in terms of the task being implemented.

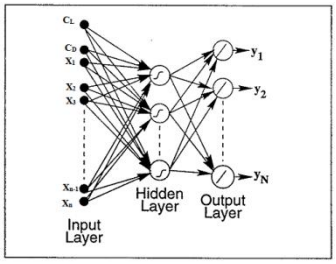

Fig. 1: The Neural-Network structure trained to predict surface Y-coordinates and CL, CD and X Coordinates are the inputs.

Figure 2 illustrates an airfoil profile which is defined by a set of x- and y-coordinates. The aerodynamic force coefficients corresponding to series of airfoil are stored in a database along with the airfoil coordinates. A feed forward neural network is created with the inputs - aerodynamic coefficients CL, CD and X Coordinates to produce the outputs - the airfoil Y coordinates.

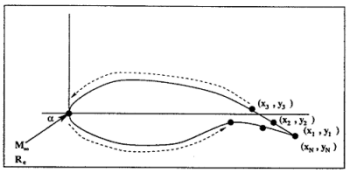

Fig. 2: Flow field and airfoil data.

Training generally involves minimization of error with respect to the training set. Learning algorithms such as the back-propagation algorithm for feed-forward multilayer networks [5] help us to find set of weights by successive improvement from an arbitrary starting point.

The below sigmoidal activation function is used to generate the output.

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

Abid *et al*. defines the linear output for a neuron j at output layer L as

$$u_j^L = \sum w_{ji} y_i^H \quad (2)$$

Where wji is the weight connection between the output neuron j and hidden neuron i. And y_i^H is the output of neuron i at hidden layer H.

And Abid *et al*. defines the non linear output as

$$f(u_j^L) = \frac{1}{1+e^{-u_j^L}} \quad (3)$$

The non linear error is given by

$$e_{1j}^L = d_j^L - y_j^L \quad (4)$$

Where d_j^L and y_j^L respectively is desired and current output for jth unit in the Lth layer. The linear error is given by

$$e_{2j}^L = ld_j^L - u_j^L \quad (5)$$

Where ld_j^L is defined as below

$ld_j^L = f^{-1}(d_j^L)$ (Inversing the output layer)

The learning function depends on the instantaneous value of the total error. As per the modified back propagation by Abid *et al*, the new cost term to increase the convergence speed is defined as

$$E_p = \frac{1}{2}(e_{1j}^L)^2 + \lambda\frac{1}{2}(e_{2j}^L)^2 \quad (6)$$

Where λ is Weighting coefficient

Now, this new cost term can be interpreted as below

Ep = (nonlinear error + Linear error)

Learning of the Output Layer:

Now, the weight update rule is derived by

applying the gradient descent method to Ep.

Hence we get weight update rule for output Layer L as

$$\Delta w_{ji} = -\mu_L \frac{\partial E_p}{\partial w_{ji}} \quad (7)$$

Where μ_L is the network learning parameter

Differentiating equation (6) based on equation (4) & (5) and applying it in equation (7), we get,

$$\Delta w_{ji} = \mu_L e_{1j}^L \frac{\partial y_j^L}{\partial w_{ji}} + \mu_L e_{2j}^L \frac{\partial u_j^L}{\partial w_{ji}}$$

$$\Delta w_{ji} = \mu_L e_{1j}^L \frac{\partial y_j^L}{\partial u_j} \frac{\partial u_j^L}{\partial w_{ji}} + \mu_L e_{2j}^L y_i^H$$

$$\Delta w_{ji} = \mu_L e_{1j}^L f'(u_j^L) y_i^H + \mu_L e_{2j}^L y_i^H \quad (8)$$

Learning of the hidden Layer:

And we get the weight update rule for the hidden layer H as

$$\Delta w_{ji} = -\mu_H \frac{\partial E_p}{\partial w_{ji}}$$

Where μ_H is the network learning parameter

Differentiating E_p and applying it in the above equation, we get

$$\Delta w_{ji} = \mu_H e_{1j}^H \frac{\partial y_j^H}{\partial w_{ji}} + \mu_H e_{2j}^H \frac{\partial u_j^H}{\partial w_{ji}}$$

$$\Delta w_{ji} = \mu_H e_{1j}^H \frac{\partial y_j^H}{\partial u_j} \frac{\partial u_j^H}{\partial w_{ji}} + \mu_H e_{2j}^H y_i^F \quad (9)$$

Where y_i^F is the output of neuron i at layer F which is previous to hidden layer H.

$$\Delta w_{ji} = \mu_H e_{1j}^H f'(u_j^H) y_i^F + \mu_L e_{2j}^H y_i^F$$

$$e_{1j}^H = \sum_{r=1}^{n_L} f'(u_r^L) e_{1r}^L w_{rj}^L$$

$$e_{2j}^H = f'(u_j^H) \sum_{r=1}^{n_L} e_{2r}^L w_{rj}^L$$

The network learning parameter μ is calculated, which plays an important role in minimizing the error. Then the network is trained with corresponding change of weight for both hidden and output layer. And the weight is determined by calculating linear, nonlinear error and adaptive learning rate.

Modified Back Propagation:

In the proposed technique the added derivative of the corresponding neuron's activation function is considered.

$$\varphi_{01} = \sum_{j=0}^{n} f'(u_j^L)$$

Now the result will be divided by the number of output neurons and it is known as μ(new learning rate)

$$\mu = \frac{\varphi_{01}}{n} \quad (10)$$

Hence applying the above learning parameter μ in equation (8), we get

$$\Delta w_{ji} = \mu e_{1j}^L f'(u_j^L) y_i^H + \mu e_{2j}^L y_i^H \quad (11)$$

Similarly for the hidden layer we have to calculate the adaptive learning rate μ_{hid}. First nonlinear errors and linear error of all hidden neurons are multiplied with weight of the output layer as below

$$\varphi_{h1} = \sum_{r=1}^{n_{L-1}} \sum_{j=1}^{n_L} e_{1j}^{L-1} w_{rj}^L$$

$$\varphi_{h2} = \sum_{r=1}^{n_{L-1}} \sum_{j=1}^{n_L} e_{2j}^{L-1} w_{rj}^L$$

Now φ is divided by the total number of hidden neurons n and the result is known as μ

$$\mu = \frac{\varphi_h}{n}$$

And then μ_{hid} is computed as follows

$$\mu_{hid} = f'(\mu) \quad (12)$$

Where f is sigmoidal activation functions. Then the change of weight is calculated using the following equation.

$$\Delta w_{ji} = -\mu_{hid} e_{1j}^H f'(u_j^H) y_i^F + \mu_{hid} e_{2j}^H y_i^F \quad (13)$$

Hence the above calculated weights from equation (13) and (11) for both hidden and output layer respectively are considered in the proposed algorithm.

Proposed algorithm:

In the Proposed Algorithm the network learning parameter μ is first initialized to some random value and then it is calculated using equation (10) and (12). Here, the change of weight for output layer and hidden layer is determined using new calculated learning rate. Its mean square error is calculated and compared with threshold value. Based on the comparison, the network is declared as trained. Now the test data is passed to the trained network to get the desired output.

Step1: Initialize the parameter μ to some random values

Step2: Assign Threshold value to a fixed value based on the sigmoid function.

Step3: Calculate linear output using equation (2)

Step4: Calculate Non-Linear output using sigmoid function as in the equation (3)

Step5: Calculate the below values for output layer

(i) Calculate the learning rate for output layer using equation (10)

(ii) Calculate weight change for output layer using the equation (11)

Step6: Calculate the below values for hidden layer

(i) Calculate the learning rate for hidden layer using equation (12)

(ii) Calculate the weight change for hidden layer using the equation (13)

Step7: Calculate the mean square error (mse) using below formula

$$mse = \frac{(\text{desired output} - \text{actual output})^2}{2}$$

Step8: The above calculated mean square error value is compared with threshold value.
Step9: If the mean square error value is greater than threshold value, then the above steps from 3 to 7 is repeated. That is, linear and non-linear outputs are calculated again using the newly calculated learning rate & weight change from step 5 and step 6.
Step10: If the mean square error value is less than threshold value, then declare that the network is trained
Step11: Now the test data is passed to trained network to produce the desired airfoil profile

RESULTS AND DISCUSSION

In our investigation of neural network models for inverse design of airfoil sections, we found that satisfactory results were obtained by adaptive learning rate.In our case, we considered ANN with twenty hidden neurons. We have a database comprised of 26 upper and lower-surface x and y coordinates, together with the corresponding coefficient of lift (CL) and the coefficient of drag (CD). There were 78 patterns in total. A pattern is nothing but the coordinates used to draw an airfoil profile. We are going to calculate the y coordinates using the proposed algorithm for given x coordinates and CL & CD. The calculated y coordinates should match with the y coordinates stored in the database. Hence we can say that the main goal is to determine the y coordinates of airfoil profile for given x coordinates and CL & CD.

The network was trained with 60 patterns to minimize error. Then network was tested with a test set comprising 18 patterns which was not used in the training process. In Table I, we have given the values of stored y coordinates, the values of calculated y coordinates for a pattern and also the difference between these values. Just for sample, we have given 8 coordinates out of 26 coordinates for a pattern. From this table, we can say that the computed profiles generated during the test process show good agreement with the actual profiles.

Table I: Profile comparison between calculated & stored Y coordinates.

Y coordinate in database	Y coordinate calculated using proposed algorithm in test phase	Difference
0.00301	0.00303	-0.00002
0.00800	0.00935	-0.00135
0.01560	0.01690	-0.00130
0.02492	0.02720	-0.00228
0.03497	0.03665	-0.00168
0.04470	0.04788	-0.00318
0.05294	0.05358	-0.00064

We can say that the computed profiles generated during the test process show good agreement with the stored database profiles .Now the airfoil profile generated by proposed algorithm and the profile stored in database are compared for Coefficient of Lift (CL) and Coefficient of Drag (CD). A tool called XFoil is used to compare CL and CD. This tool will produce the airfoil with CL and CD by taking the airfoil profile as input. Figure 3 contains the airfoils with corresponding CL and CD for the profiles naca0012, naca1012 and naca2012 which are generated by proposed algorithm in test phase. Thus we observe that the adaptive learning rate approach generated comparatively the correct airfoil profiles.

Next we compare the convergence rate at training phase between fixed learning rate approach and adaptive learning rate approach. To do this, we noted down the RMS error at each epoch and plotted it in the graph in Figure 4 for both approach. The blue line indicates the error in Fixed Learning rate and the red line indicates that of adaptive learning rate approach. Figure 4 shows how the training decreases root mean square (RMS) errors with the epoch. From this figure 4, it is obvious that adaptive learning rate approach converges quickly and in this approach the error is less at the converging stage.

Now we consider another attribute – the accuracy for our study. A measure of the accuracy of the results obtained can be inferred from examination of error which is defined as

$$Error = \frac{y_{i(actual)} - y_{i(computed)}}{airfoil\ thickness\ ratio} * 100$$

Where $y_{i(actual)}$ is the actual y-coordinate of the section at location i, $y_{i(computed)}$ is the computed y-coordinate. Table-II shows the maximum error in percentage for the airfoil profiles naca0012, naca1012, naca2012, naca3012 and naca3012 which are generated by proposed algorithm in test phase. From this table, we can conclude that the adaptive learning rate approach predicated comparatively the correct airfoil profiles.

Next the time taken and error in each epoch obtained from Fixed Learning rate and Adaptive learning rate approach are compared and tabulated in Table-III. From this table, it is clear that at epoch 14, the adaptive learning rate predicted approximately correct airfoil. But the fixed learning rate takes 22 epochs to predict approximately correct airfoil. The adaptive learning rate takes 515 clock ticks to predict correct airfoil, but the fixed learning rate takes 780

clock ticks to predict the correct airfoil. This proves that the proposed approach results in less error and takes less time to predict the airfoil for the given CL & CD. The time is denoted by number of clock tick.

Table II: Maximum error for airfoil profiles generated by proposed algorithm.

Airfoil	Maximum error (%)
NACA0012	0.001799
NACA1012	0.001493
NACA2012	0.003778
NACA3012	0.006585
NACA4012	0.013208

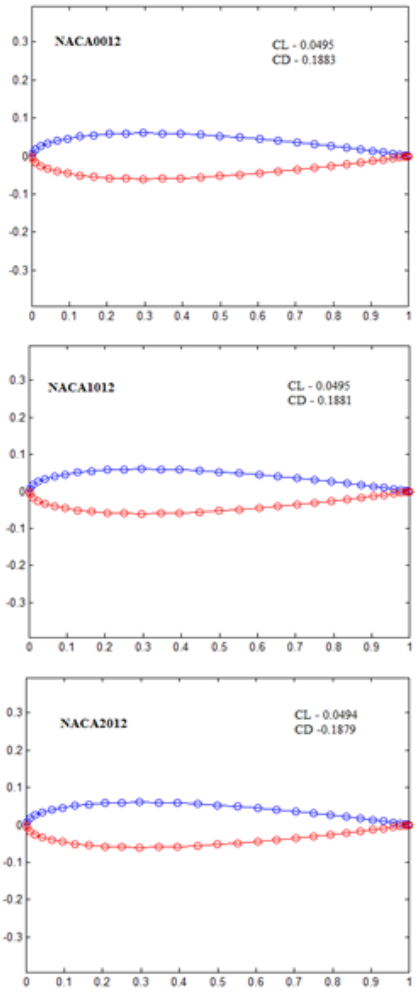

Fig. 3: Proposed Alg. generated Airfoils.

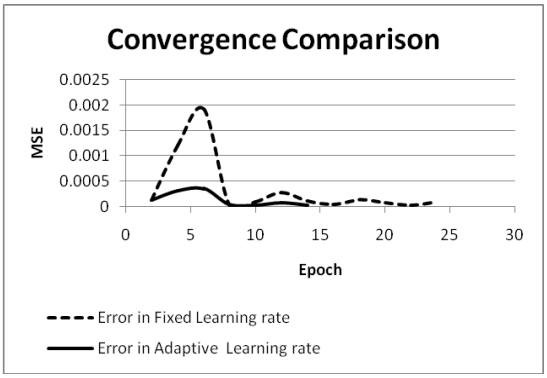

Fig. 4: Convergence comparison.

Table III: Comparison Table.

Epoch * 1000	Fixed Learning rate		Adaptive Learning rate	
	Error	clock ticks	Error	clock ticks
0	3.406344	0	4.493385	0
2	0.000124	78	0.000126	94
4	0.001202	141	0.000321	156
6	0.001922	219	0.000356	234
8	0.000038	281	0.000046	297
10	0.000089	359	0.000031	375
12	0.000275	421	0.000079	453
14	0.000105	499	0.00003	515
16	0.000045	562	-	-
18	0.000132	640	-	-
20	0.000078	702	-	-
22	0.000032	780	-	-

4. Conclusions:

In this paper, we have used an inverse design methodology in artificial neural networks for the design of airfoil. Here to get the adaptive learning rate for hidden layer, linear and nonlinear errors of hidden layer is multiplied with it corresponding output layer weights separately and then added. Then the value is divided by number of hidden neuron. And to get Adaptive learning rate for the output layer, the derivatives of the corresponding neuron's activation function are added then divided by total number of output neurons. The most advantage of this approach is that the time required to fix the learning rate by trial and error is saved. The results indicate that our adaptive learning rate approach in modified backpropagation algorithm will converge at comparatively high speed with fewer errors. We used several approaches which are discussed in the section "Results and Discussion" to prove this. Results indicate that optimally trained artificial neural networks using proposed algorithm may accurately predict airfoil profile at a faster pace.

Annexure:

The surface direction accurately estimated using Partial derivatives.

$f(x)$ is defined as below,

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

To find the accurate direction we need the partial derivatives of above function f. Differentiating the above function f, we get

$$f' = \frac{e^{-u}}{(1 + e^{-u})^2}$$

This f' is used in the equation (8) & (9) and the weight change is derived using it.

REFERENCES

Magoulas, G.D., V.P. Plagianakos, M.N. Vrahatis A. Van Ooyen, B. Nienhuis, 1992. "Improving the convergence of the backpropagation algorithm", Neural Networks, 5: 465-471.

Garabedian, P.R. and D.G. Korn, 1971. "Numerical Design of Transonic Airfoils", Numerical Solution of Differential Equations - II, Academic Press.

Hertz, J., A. Krogh and R.G. Palmer, 1991. "Introduction to the Theory of Neural Computation", Addison-Wesley Publishing Co., California.

Hoist, T.L., J.W. Slooff, H. Yoshihara and Jr. W.F. Ball-haus, 1982. "Applied computational transonic Aerodynamics", AGARD-AG-266.

Jeong S.Y., S.Y. Lee, 2000. "Adaptive learning algorithms to incorporate additional functional constraints into neural networks", Neurocomputing, 35: 73-90.

Behera, L., S. Kumar, A. Patnaik, 2006. "On adaptive learning rate that guarantees convergence in feedforward networks", IEEE Trans. Neural Networks, 17: 1116-1125.

Jacobs, R.A., 1988. "Increased rates of convergence through learning rate adaptation", Neural Networks, 1: 295-307.

Riedmiller, M. and H. Braun, 1993. "A Direct Adaptive Method for Faster Backpropagation Learning: the RPROP Algorithm", Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, March 28-April 1.

Abid, S., F. Fnaiech, M. Najim, 2001. "A fast feedforward training algorithm using a modified form of the standard backpropagation algorithm". IEEE Trans. Neural Networks, 12: 424-430.

Iranmanesh, S. and M.A. Mahdavi, 2009. "Differential adaptive learning rate method for back propagation neural networks, World Academy of Science", Engineering and Technology, 50: 285-288.

Jeyaseeli Subavathi, S. and T. Kathirvalavakumar, 2011. "Adaptive modified backpropagation algorithm based on differential errors", International Journal of Computer Science, Engineering and applications, 1: 21-34.

Sha, D., V.B. Bajic, 1999. "Adaptive on line ANN learning algorithm and application to identification of non-linear systems", Informatica, 23: 521-529.

Yu, X.H., G.A. Chen, S.X. Cheng, 1993. "Acceleration of backpropagation learning using optimized learning rate and momentum", Electron. Lett., 29: 1288-1289.

Zhihong Man, Hong Ren Wu, Sophie Liu and Xinghuo Yu, 2006. "A new daptive backpropagation algorithm based on Lyapunov stability theory for neural networks", IEEE Trans. Neural Networks, 17: 1580-1591.