

DESIGN OF AIRFOIL USING BACKPROPAGATION TRAINING WITH MIXED APPROACH

K.Thinakaran, Dr.R.Rajasekar

Sri Venkateswara College of Engineering & Tech., Thiruvallur, Tamil Nadu,
 Excel Engineering College, Pallakkapalayam, Tamil Nadu,
 thina85@rediffmail.com ,rajasekar080564@yahoo.com

ABSTRACT

Levenberg-Marquardt back-propagation training method has some limitations associated with over fitting and local optimum problems. Here, we proposed a new algorithm to increase the convergence speed of Backpropagation 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 feedforward neural network is created with aerodynamic coefficient as input to produce the airfoil coordinates as output. In the proposed algorithm, for output layer, we used the cost function having linear & nonlinear error terms then for the hidden layer, we used steepest descent cost function. Results indicate that this mixed approach greatly enhances the training of artificial neural network and may accurately predict airfoil profile.

Keyword: linear error, nonlinear error, steepest descent method, airfoil design, neural networks, backpropagation, Levenberg-Marquardt.

1. INTRODUCTION

The aim of this paper is to generate geometry for airfoil using inverse design method with high convergence speed and with minimal error. Hence we are supposed to enhance inverse design method which will converge fast and have minimal error. There are many inverse techniques in use, for example, hodograph methods for two-dimensional flows (1-3). The inverse design of backpropagation takes long time to converge. Some focused on better function and suitable learning rate and momentum (4-9). To design fast algorithm, Abid et al. proposed a new algorithm by minimizing sum of squares of linear and nonlinear errors for all output (10). Jeong et al. proposed learning algorithm based on first and second order derivatives of neural activation at hidden layers (11). Han et al. proposed two modified constrained learning algorithms - First new Learning Algorithm and Second new Learning Algorithm to obtain faster convergence rate. The additional cost terms of the first algorithm are selected based on the first derivatives of the activation functions of the hidden neurons and second order derivatives of the activation functions of the output neurons while the additional cost terms of second one are selected based on the first order derivatives of the activation functions of the output neurons and second order derivatives of the activation functions of the hidden neurons [12]. The notable differences of the accelerated modified LM method are that the line search for the approximate LM step [16].

The objective of this work is to show that using cost function having linear & nonlinear terms in output layer and using cost function of steepest descent [13-15] in hidden layer trains the network fast. This mixed approach trains the network in such a way that it converges fast with minimal error. 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. The process of obtaining appropriate weight in a neural network design utilizes two set of equations. First, the feedforward 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 gradient direction is the direction of steepest descent. But, the learning rate may be different in different directions. The linear and nonlinear error terms introduces certain modification in directions in order to speed up the convergence. Hence the mixed approach of using cost function having linear & non-linear error terms in output layer and using cost function of steepest descent in hidden layer results fast convergence. So, we are going to analyze this mixed approach in the proposed algorithm.

2. ANN TRAINING METHOD

Figure-1 illustrates an airfoil profile which is described by a set of x- and y-coordinates.

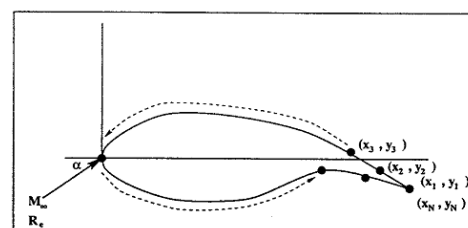


Figure-1. Flow field and airfoil data

The aerodynamic force coefficients corresponding to series of airfoil are stored in a database along with the airfoil coordinates. A feedforward neural network is created with the inputs - aerodynamic coefficients C_l , C_d and X Coordinates to produce the outputs - the airfoil Y coordinates.

Here, we considered an ANN with an input layer, hidden layer and an output layer as in Figure2. Each layer has its own

neurons which interconnect with neurons in the consecutive layer. The airfoil co-efficients and the 'x' co-ordinates are considered as neurons in the input layer. We are aimed to generate the airfoil 'y' co-ordinates. The input and the weight determine again network traverse happens till it meets threshold the output of each neuron in each layer. The actual output from the output layer is compared against targeted output. If it doesn't match the threshold value, then we need to find weight change based on the outputs and once value. This process is termed to be training the network. 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. Once network is trained with trained data set, then using feedforward function, it is tested with test data to verify whether it generates the desired output – the airfoil 'y' co-ordinates.

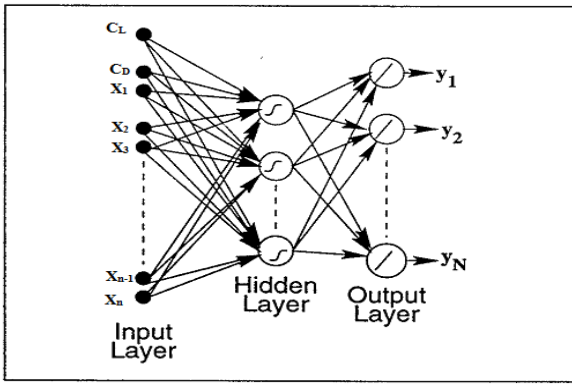


Figure-2. ANN architecture used

For our analysis, we considered the below sigmoidal activation function 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

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

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 j^{th} unit in the L^{th} 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

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

The learning function depends on the instantaneous value of the total error. As per the modified backpropagation 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

$$E_p = (\text{nonlinear error} + \text{linear error})$$

Learning of the Output Layer

Now, the weight update rule is derived by applying the gradient descent method to E_p . Hence we get weight update rule for output Layer L as

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

where μ_L is the network learning parameter.

Hence, differentiating E_p in equation (6) w.r.t w_{ji} , we get

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

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

$$\Delta w_{ji} = -\mu_L e_{1j}^L f'(u_j^L) y_i^H + \mu_L e_{2j}^L y_i^H \quad (7)$$

Learning of the hidden Layer Using Steepest Descent Algorithm

The learning algorithm is the repeated application of the chain rule to compute the influence of each weight in the network with respect to an arbitrary error function E. Here the Error function is,

$$E_k = \frac{1}{2}(T_k - O_{ok})^2 \quad (8)$$

And the weight change for hidden layer (ΔV) is,

$$\Delta V = \eta \frac{\partial E_k}{\partial V_{ij}}$$

$$\frac{\partial E_k}{\partial V_{ij}} = \frac{\partial E_k}{\partial O_{ok}} \frac{\partial O_{ok}}{\partial I_{ok}} \frac{\partial I_{ok}}{\partial O_{Hi}} \frac{\partial O_{Hi}}{\partial I_{HJ}} \frac{\partial I_{HJ}}{\partial V_{ij}}$$

By differentiating error function (8) w.r.to O_{ok} , we get

$$\frac{\partial E_k}{\partial O_{ok}} = -(T_k - O_{ok})$$

To get the term $\frac{\partial O_{ok}}{\partial I_{ok}}$, we first calculate the output of kth output neuron O_{ok} using below equation

$$O_{ok} = \frac{1}{1 + e^{-\lambda(I_{ok} - \theta_{ok})}}$$

$$O_{ok} = (1 + e^{-\lambda I_{ok}} e^{\lambda \theta_{ok}})^{-1}$$

Now, differentiate O_{ok} w.r.to I_{ok} gives,

$$\frac{\partial O_{ok}}{\partial I_{ok}} = \lambda e^{-\lambda I_{ok}} e^{\lambda \theta_{ok}} (1 + e^{-\lambda I_{ok}} e^{\lambda \theta_{ok}})^{-2}$$

$$\frac{\partial O_{ok}}{\partial I_{ok}} = \lambda O_{ok} (1 - O_{ok})$$

Hence,

$$\frac{\partial E_k}{\partial O_{ok}} \frac{\partial O_{ok}}{\partial I_{ok}} = -\lambda (T_k - O_{ok}) O_{ok} (1 - O_{ok}) = -d_k$$

$$\frac{\partial I_{ok}}{\partial O_{Hi}} = W_{ik}$$

$$\frac{\partial O_{Hi}}{\partial I_{HJ}} = \lambda (O_{Hi}) (1 - O_{Hi})$$

$$\frac{\partial I_{HJ}}{\partial V_{ij}} = I_{ij}$$

$$\frac{\partial E_k}{\partial V_{ij}} = -d_k \cdot W_{ik} \cdot \lambda (O_{Hi}) (1 - O_{Hi}) \cdot I_{ij}$$

$$\frac{\partial E_k}{\partial V_{ij}} = d_i^* I_{ij}$$

where,

$$d_i^* = -d_k \cdot W_{ik} \cdot \lambda (O_{Hi}) (1 - O_{Hi})$$

Hence the weight change for hidden layer,

$$\Delta V = \eta I_i d^* \quad (9)$$

LEVENBERG-MARQUARDT ALGORITHM

The Levenberg-Marquardt is one of the fastest and accurate learning algorithms for small to medium sized networks. The advantage of the LM algorithm decreases as the number of network parameters increases. The Levenberg-Marquardt algorithm (LM) can find a solution of a system of non-linear equations, $y = \phi x$, by finding the parameters, ϕ , that link variables, y , to variables, x . The LM algorithm in (10), finding the appropriate change, $\Delta \phi$, leading to smaller errors. The LM-algorithm depends on error, E , the Hessian matrix H , the gradient of the error, ∇J , a scalar μ which controls the trust region, and I is the identity matrix.

$$H = J \times J$$

$$\nabla J = J \times E$$

$$\Delta \phi = -(H + I\mu)^{-1} \nabla J \quad (10)$$

Proposed Mixed approach algorithm

In the Proposed Algorithm the learning parameters μ and λ are initialized according to the convergence of the problem. Here, the change of weight for output layer is determined using linear and nonlinear error and the hidden layer is trained with steepest descent method. 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 learning parameters μ, λ 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: Then calculate weight change for output layer using the equation (7) and calculate the weight change for hidden layer using the equation (9)

Step6: Calculate the mean square error using the equation (8)

Step7: The above calculated mean square error value is compared with threshold value.

Step8: If the mean square error value is greater than threshold value, then the above steps from 3 to 6 is repeated.

Step9: If the mean square error value is less than threshold value, then declare that the network is trained

Step10: Now the test data is passed to trained network to produce the desired airfoil profile.

3. RESULTS AND DISCUSSION

It is said that Levenberg-Marquardt is one of the fastest and accurate learning algorithms for small to medium sized networks. But our proposed algorithm is faster than Levenberg-Marquardt algorithm in FNN with the structure of 2-2-1 to address XOR problem. From the below Table 1 observation, you can see the time taken to converge in mixed approach to solve XOR problem is less when compared to the Levenberg-Marquardt algorithm. In general, the standard LM algorithm does not perform as well on pattern recognition problems as it does on function approximation problems. The advantage of the LM algorithm decreases as the number of network parameters increases.

TABLE 1- Performance comparison in XOR problem

Mixed Approach			LMBP		
epoch	MSE	Time (ms)	epoch	MSE	Time (ms)
0	0.108857	16	1	0.500698	109
480	0.124079	31	2	0.442961	111
840	0.124255	47	4	0.347800	114
1140	0.123867	62	6	0.314990	116
1500	0.106499	78	8	0.269975	119
1860	0.021690	82	10	0.192142	122
2280	0.000039	94	12	0.098765	125

If the Hessian is inaccurate this can greatly throw off Newton's Method. Higher lambda favors gradient descent, lower lambda favors Newton. Each cell in this matrix represents the second order derivative of the output of the neural network. Calculation of the Hessian accomplished by calculating the gradients.

$$\frac{\partial E}{\partial w_{(i)}} = 2(y - t) \frac{\partial y}{\partial w_{(i)}}$$

Actually the second derivative is element of the Hessian matrix. This is calculated with the following formula.

$$\frac{\partial^2 E}{\partial w_i \partial w_j} = 2(\frac{\partial y}{\partial w_i} \frac{\partial y}{\partial w_j} + (y - t) \frac{\partial^2 y}{\partial w_i \partial w_j})$$

Some of the LMA algorithm only supports a single output neuron. This is because the LMA algorithm has its roots in mathematical function approximation. So by the above we identify that Levenberg-Marquardt algorithm has some drawback that is the each epoch will take more time to complete the computation. In the Table I the Levenberg-Marquardt algorithm take more time to complete each epoch. In mixed approach for the output layer we use the steepest descent cost function to identify the error value but to increase the speed of convergence we use non linear approach in the hidden layer, so it will increase the convergence speed of backpropagation algorithm.

In our investigation of neural network models for inverse design of airfoil sections, we found that satisfactory results were obtained by using the mixed approach of steepest descent and linear and non linear error terms.

In our research, an airfoil is drawn with 26 x and y coordinates with corresponding coefficient of lift (C_L) and the coefficient of drag (C_D). We have database comprised of 78 NACA four series airfoils with different combinations of C_L, C_D, x & y coordinates. Here we are considering C_L, C_D and x coordinates as input values to the neurons in the input layer of ANN. Also we considered twenty hidden neurons in the hidden layer. The y coordinates are considered as output neurons in the output layer of ANN. We are going to calculate the y coordinates using the proposed algorithm for given x coordinates and C_L & C_D. 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 C_L & C_D. This is the "inverse" problem for airfoil design.

We trained the network with 60 airfoil sets. Then we tested the network for the remaining 18 sets. We gave C_L, C_D & x coordinates and our ANN generated its corresponding y coordinates which equals the y coordinates stored in our database at optimized speed. In Table 2, we have given the values of stored y coordinates, the values of calculated y coordinates (calculated using our algorithm) for a pattern NACA1017 and also the difference between these values. Just for sample, we have given 8 coordinates out of 26 coordinates for the pattern NACA1017. From this table, we can say that the computed profiles generated during the

test process show good agreement with the actual profiles.

Table 2 - Profile comparison between calculated & stored Y coordinates

Y coordinate in database	Y coordinate calculated using proposed algorithm in test phase	Difference
0.00350	0.003086	0.000414
0.00993	0.009320	0.000610
0.03173	0.030728	0.001002
0.04468	0.044099	0.000581
0.05721	0.056115	0.001095
0.07540	0.074828	0.000572
0.07835	0.078136	0.000214
0.07593	0.075784	0.000146

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. The new computed airfoil x and y coordinates from the testing process are passed to XFOIL tool. This tool creates the airfoil and generates its corresponding CL, CD. Four such airfoils from XFOIL tool are shown in Figure-3. The CL, CD from XFOIL tool and the CL, CD from our database are compared and found that it matches. This shows that our ANN is trained with minimal error. A measure of the accuracy of the results obtained can be inferred from examination of error which is defined as

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

Where $y_i(actual)$ is the actual y-coordinate of the section at location i, $y_i(computed)$ is the computed y-coordinate. We computed the error using the above formula considering the y coordinate we derived from our mixed approach as $y_i(computed)$ and the y coordinate from the database as $y_i(actual)$ and tabulated in Table-3 and this table shows the maximum error in percentage for the airfoils shown in Figure-3. From this table, we can conclude that the mixed approach predicated comparatively the correct airfoil profiles

Table 3- Maximum error for airfoils shown in Fig3

Airfoil	Maximum error (%)
NACA0012	0.00194
NACA0014	0.00265
NACA2013	0.00113
NACA2017	0.00238

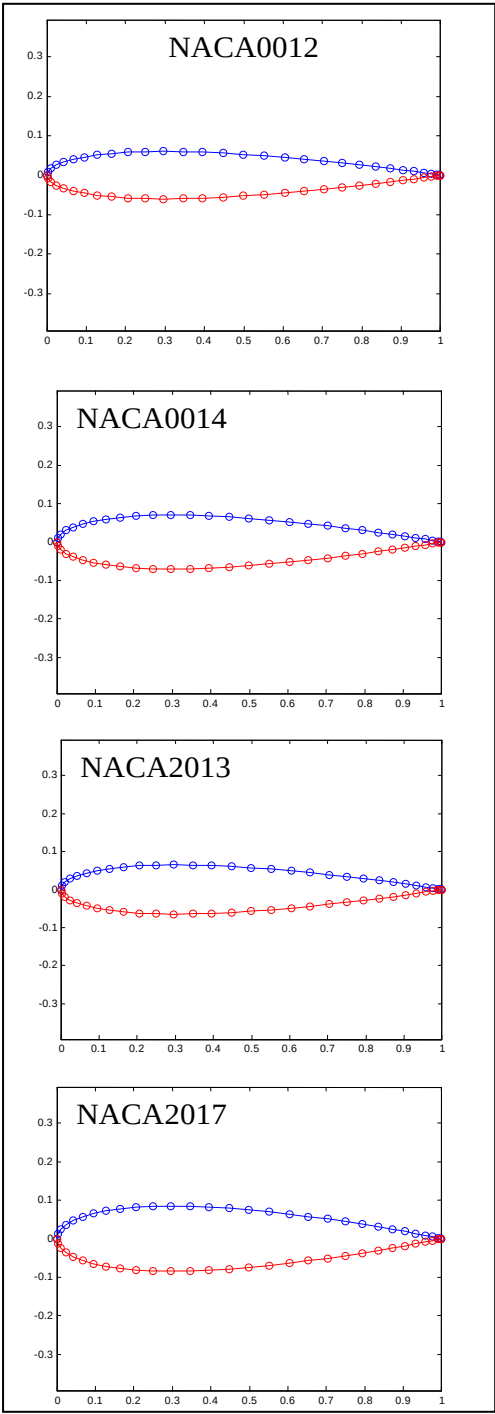


Figure-3. Airfoil profiles from XFOIL tool

Figure-4 shows how the training decreases mean square Error (MSE) with the epoch. The red dotted line indicates the error in linear and nonlinear approach. The green line (line with hyphen and dot) indicates the error in steepest descent. And the blue line indicates that of mixed approach. From the figure, it is obvious that mixed

approach converges quickly compared to steepest descent or nonlinear approach. Also the error is minimal for mixed approach compared to steepest descent and nonlinear approach.

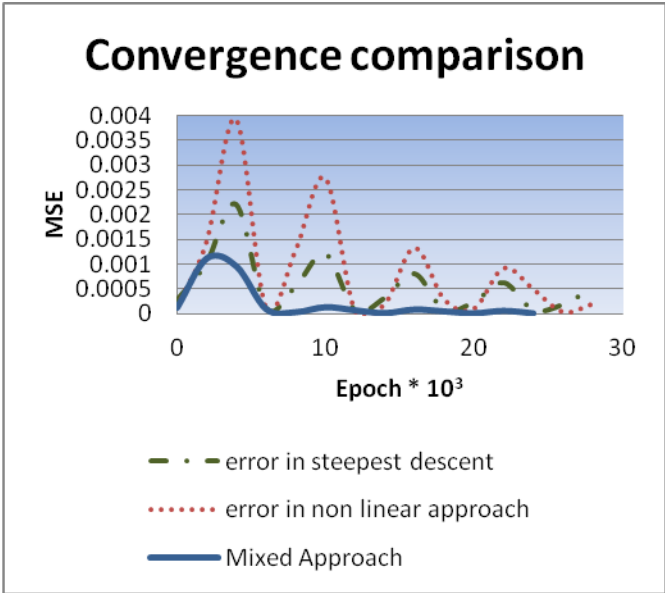


Figure-4. Convergence comparison

The results obtained from steepest descent, non-linear and mixed approach are compared and tabulated in Table-4. When you check the table, you can easily find that our proposed algorithm converges quickly than the other two approaches. This table clearly proves that the mixed 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 ticks in the table.

Table 4- Comparison Table

Epoch * 1000	error in steepest descent	error in non linear approach	Mixed Approach (Proposed Algorithm)
0	3.005668	3.20248	3.011418
2	0.000261	0.00013	0.000131
4	0.00102	0.001444	0.001125
6	0.002218	0.003933	0.000961
8	0.000116	0.00014	0.000101
10	0.000545	0.001281	0.000041
12	0.001216	0.002752	0.000141
14	0.00007	0.000065	0.000061
16	0.000315	0.00020	0.000021
18	0.000806	0.001336	0.000093
20	0.000064	0.000294	0.000056
22	0.000229	0.000106	0.000018

24	0.000623	0.000926	0.000068
26	0.000067	0.000499	0.000016
28	0.00018	0.000039	Converged & the aim attained
30	0.000518	0.000202	-
32	0.000068	0.000064	-
34	0.000148	0.000024	-
36	0.000446	0.000063	-
38	0.000066	0.000294	-
40	0.000126	0.000034	-
42	0.000391	0.000538	-
44	0.000064	0.000096	-
46	0.000111	0.000172	-
48	0.000347	0.000245	-
50	0.000061	0.000067	-
52	0.000099	0.000151	-
54	0.000312	0.000187	-
56	0.000058	0.000051	-
58	0.00009	0.000138	-
60	0.000282	0.000136	-
62	0.000055	0.000039	-
64	0.000082	0.000134	-
66	0.000257	0.000099	-
68	0.000053	0.000037	-
70	0.000076	0.00012	-
72	0.000235	0.000095	-
74	0.000051	0.000041	-
76	0.000071	0.000098	-
78	0.000216	0.000103	-
80	0.000049	0.000047	-
82	0.000066	0.000056	-
84	0.00020	Converged & the aim attained	-
86	0.000047		-

4. CONCLUSIONS

In this paper, we have used an inverse design methodology in artificial neural networks for the design of airfoil. We did a comparison between mixed algorithm and Levenberg-Marquardt algorithm. The results show that mixed algorithm is better than Levenberg-Marquardt algorithm in terms of convergence speed. The results indicate that our proposed mixed approach will converge at comparatively high speed with fewer errors. In the proposed algorithm for output layer we used the cost function having linear and nonlinear error terms then for the hidden layer steepest descent cost function is used. Results indicate that

optimally trained artificial neural networks using proposed algorithm may accurately predict airfoil profile at a faster pace without oscillation in learning.

Annexure

The surface direction accurately estimated using Partial derivatives. To find the non linear error we take inverse of $f(x)$.

$f(x)$ is defined as below,

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

Hence inverse of $f(x)$,

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

To find the accurate direction we need the partial derivatives of f .

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

This f' is used in the equation (7) and the weight change is derived using it.

The complete results obtained using our mixed approach is tabulated in Table-IV.

REFERENCES

- [1]. Garabedian, P.R. and Korn, D.G., "Numerical Design of Transonic Airfoils," Numerical Solution of Differential Equations - II, Academic Press, 1971
- [2]. Hoist, T.L., Sloof, J.W. Yoshihara, H. and all-haus Jr., W.F., "z" Applied computational Transonic Aerodynamics" AGARD-AG-266, 1982.
- [3]. Sloof, J.W., "Computational Procedures in Transonic Aerodynamic Design," Lecture presented at ITCS Short Course on Computational Methods in potential Aerodynamics, Amalfi, Italy, 1982 (also NLR MP 82020 U)
- [4]. Hertz, J., Krogh, A. and Palmer, R.G., Introduction to the Theory of Neural Computation, Addison-Wesley Publishing Co., California, 1991.

- [5]. Riedmiller, M. and Braun, H., "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, 1993.
- [6]. L. Behera, S. Kumar, A. Patnaik "On adaptive learning rate that guarantees convergence in feedforward networks", IEEE Trans. Neural Networks 17, 1116-1125, (2006).
- [7]. R. A. Jacobs "Increased rates of convergence through learning rate adaptation". Neural Networks 1, 295-307, (1988).
- [8]. G.D. Magoulas, V.P. Plagianakos, M.N. Vrahatis "Globally convergent algorithm with local learning rate" IEEE Trans. Neural Network 13, 774-779, (2002)
- [9]. X. H. Yu, G. A. Chen, S. X. Cheng "Acceleration of backpropagation learning using optimized learning rate and momentum", Electron. Lett. 29, 1288-1289, (1993).
- [10]. S. Abid, F. Fnaiech, M. Najim "A fast feedforward training algorithm using a modified form of the standard backpropagation algorithm". IEEE Trans. Neural Networks 12, 424-430, (2001).
- [11]. Jeong S.Y., Lee S.Y., (2000) "Adaptive learning algorithms to incorporate additional functional constraints into neural networks", Neuro computing 35, 73-90, (2000)
- [12]. F. Han, Q.H. Ling, d.S. Huang "Modified constrained learning algorithms incorporating additional functional constraints into neural networks", Information Sciences 178, 907-919, (2008).
- [13]. Dilip Sarkar, "Methods to speed up Error Back-propagation Learning Algorithm", ACM Computing surveys, Vol. 27, No. 4, December 1995.
- [14]. Thiang, Handry Khoswanto, Rendy Pangaldus, Artificial Neural Network with Steepest Descent Backpropagation Training Algorithm for Modeling Inverse Kinematics of Manipulator, World Academy of Science, Engineering and Technology 36, 2009.
- [15]. Richard S. Sutton, Two problems with backpropagation and other steepest-descent learning procedure for networks, GTE Laboratories.
- [16]. Jinyan Fan, Accelerating the modified Levenberg-Marquardt method for nonlinear equations "Math. Comp", 1173-1187, (2014).

ABOUT THE AUTHORS



K.Thinakaran received M.E., degree in computer science from Mahendra Engineering College which is affiliated to Anna University, Coimbatore, Tamilnadu in 2009. He is currently working toward the Ph.D. degree at the Anna University. He is currently an Associate Professor in Computer Science Engineering, Sri Venkateswara College of Engineering & Technology, Thiruvallur, India. His current research interests include Neural Network and Data Mining.



Dr.R.Rajasekar received his doctorate from Department of Aeronautics, Imperial College, London, UK. His aeronautical masters' degree was from IIT, Madras. He is currently working as the Professor and Head of Aeronautical Engineering Department, Excel Engineering College, Pallakkapalayam, Tamil Nadu. (an affiliated college under Anna University, Chennai). His specialization and research interests are aerodynamics and its applications.