Improving Conjugate Gradient method For Training Feed Forward Neural Networks

Luma. N. M.Tawfiq & Alaa. K. J. AL-Mosawi
Department of Mathematics, College of Education-Ibn Al-Haitham, University of Baghdad .

Abstract

In this paper, many modified and new algorithms have been proposed for training feed forward neural networks, many of them having a very fast convergence rate for reasonable size networks.

In all of these algorithms we use the gradient of the performance function (energy function, error function) to determine how to adjust the weights such that the performance function is minimized, where the back propagation algorithm has been used to increase the speed of training. The above algorithms have a variety of different computation and thus different type of form of search direction and storage requirements, and all the above algorithms applied in approximation problem.

1. Introduction

Back propagation (BP) process can train multilayer feed forward neural network (FFNN). With differentiable transfer functions, to perform a function approximation to continuous function $f \in R^N$, pattern association and pattern classification. The term of back propagation to the process by which derivatives of network error with respect to network weights and biases, can be computed. This process can be used with a number of different optimization strategies.

2. Training Algorithms for Neural Networks

Any non-linear optimization method, a local or global one, can be applied to the optimization of feed-forward neural networks weights. Naturally, local searches are fundamentally limited to local solutions, while global ones attempt to avoid this limitation. The training performance varies depending on the objective function (energy function or error function) and underlying error surface for a given problem and network configuration.

Since the *gradient* information of error surface is available for the most widely applied network configurations, the most popular optimization methods have been variants of *gradient* based back-propagation algorithms. Of course, this is sometimes the result of an inseparable combination of network configuration and training algorithm which limits the freedom to choose the optimization method.

Widely applied methods are, for example, modified back-propagation [1], back propagation using the conjugate-gradient approach [2], scaled conjugate-gradient and its stochastic counterpart [3], the Marquadt algorithm [4], and a concept learning based back-propagation [5]. Many of these gradient based methods are studied and discussed even for large networks in [6]. Several methods have been proposed for network configurations where the *gradient* information is not available, such as simulated annealing for networks with non-differentiable transfer functions [7].

In many studies only small network configurations are considered in training experiments. Many *gradient* based methods and especially the Levenberg-Marquadt method are extremely fast for small networks (few hundreds of parameters), thus, leaving no room or motivation for discussion of using evolutionary approaches in the cases where the required *gradient* information is available. The problem of local minima can be efficiently avoided for small networks by using repeated trainings and randomly initialized weight values. Nevertheless, evolutionary based global optimization algorithms may be useful for validation of an optimal solution achieved by a *gradient* based method.

For large FFNN´s, consisting of thousands of neurons, the most efficient training methods (Levenberg – Marquadt , Quasi – Newton, etc.) demand an unreasonable amount of computation due to their computational complexity in time and space. One possibility could be a hybrid of traditional optimization methods and evolutionary algorithms as studied in [8]. Unfortunately, it seems that none of the contemporary methods can offer superior performance over all other methods on all problem domains. It seems that no single solution appears to be available for the training of artificial neural networks.

2.1. Conjugate Gradient Algorithms *(traincg)*

The conjugate gradient algorithms perform a search along conjugate directions, which produces generally faster convergence than gradient descent directions [Hagan and Beale, 1996]. The CG algorithms start out by searching in the gradient descent direction (negative of the gradient) on the first iteration, $\rho_0 = -g_0$. Then the next search direction is determined so that it is conjugate to previous search directions, that is : [9]

$$W_{k+1} = W_k + \eta_k \rho_k \text{ . Where } \rho_k = -g_k + \beta_k \rho_{k-1}.$$

The various versions of CG are distinguished by the manner in which the β_k is computed.

In this paper, we will present different variations of CG algorithms. In most of the training algorithms a learning rate is used to determine the length of the weight update (step size).

In most of the CG algorithms, the step size is adjusted at each iteration. A search is made along the CG direction to determine the step size, which will minimize the performance function along that line search. The CG algorithms that usually used in ANN as a training algorithm is much faster than variable learning rate back propagation, and are sometimes faster than Resilient back propagation, although the results will vary from one problem to another.

2.1.1. Fletcher- Reeves update *(traincgf)*

The general procedure for determining the new search direction is to combine the new *gradient* descent direction with the previous search direction : $\rho_k = -g_k + \beta_k\rho_{k-1}$

For Fletcher-Reeves update procedure [10] : $\beta_k = \dfrac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$

2.1.2. Polak- Ribiere update (traincgp)

Another version of the conjugate gradient algorithm was proposed by Polak and Ribiere [11].

For the Polak - Ribiere update, the constant β_k is computed from :

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{g_{k-1}^T g_{k-1}}$$

2.1.3. Dixon update (traincgd)

We propose another version of the conjugate gradient algorithm, which derive from classical method proposed by Dixon [11].

For the Dixon update, the constant β_k is computed by : $\beta_k = \dfrac{-g_k^T g_k}{\rho_{k-1}^T g_{k-1}}$

2.1.4. Al-Assady and Al-Bayati update (traincga)

We use another version of the conjugate gradient algorithm, when the classical method proposed by Al-Assady and Al-Bayati [11].

For the Al-Assady and Al-Bayati update, the constant β_k is computed by :

$$\beta_k = \frac{-g_k^T \Delta g_{k-1}}{\rho_{k-1}^T g_k}$$

2.1.5. Hestenes - Stiefel update (traincgh)

We will consider another version of the CG algorithm, when the classical method proposed by Hestenes - Stiefel [9].

For the Hestenes-Stiefel update, the constant β_k is computed by :

$$\beta_k = \frac{g_k^T \Delta g_{k-1}}{\rho_{k-1}^T \Delta g_{k-1}}$$

2.1.6. Reyadh - Luma update (traincgr)

We propose a new version of the CG algorithm when the search direction at each iteration is determined by : $\rho_k = - g_k + \beta_k \rho_{k-1}$

Where the constant β_k is computed b y : $\beta_k = \dfrac{g_k^T \Delta g_{k-1}}{\rho_{k-1}^T g_{k-1}}$

Remarks

1. For all CG algorithms, the search direction will be periodically reset to the negative of the gradient. The standard reset point occurs

when the number of iterations is equal to the number of ANN parameters (weights and biases).

2. Each of the CG algorithms, which we have discussed so far, requires a line search at each iteration. This line search is computationally expensive, since it requires that the ANN response to all training inputs which should be computed several times for each search. But the other hand one can design an algorithm to avoid the time consuming for performing line search.

3. Improve Conjugate Gradients Method

3.1.1. Conjugate Search Directions

A series of directions $\{ \rho_k \}$ have to be found such that :

$$(g_{k+1})^T \rho_k = 0 \quad \text{and} \quad (g_{k+1} + \lambda \rho_{k+1})^T \rho_k = 0$$

And, by developing in series to the lowest order :

$$[\, g^T_{k+1} + \lambda \rho^T_{k+1} H \,] \rho_k = 0$$

$$\rho^T_{k+1} H \rho_k = 0 \text{ ....(1)}$$

Where H is the Hessian calculated at point W_{t+1}. Directions which respects condition (1) are named conjugate.

3.1.2. Quadratic Error Function

A quadratic error function is of the form :

$$E(W) = E_0 + b^T W + \frac{1}{2} W^T H W \quad , \quad b, H = \text{constant.}$$

And H (the Hessian) is symmetrical and positive definite.

The error *gradient* is :

$$g = b + H W \qquad \text{..................(2)}$$

and the minimum of E is achieved at the point W^* where :

$$g_{W^*} = 0 \longrightarrow b + H W^* = 0 \quad \text{..............(3)}$$

Let consider a set of $\{\rho_i\}_{i=\overline{1,NW}}$ conjugate with respect to H directions (N_W being the total number of weights) :

$$\rho_i^T H \rho_j = 0 \qquad \text{for} \quad i \neq j \qquad \text{.............(4)}$$

and, of course, $\rho_i \neq 0$, $\forall i$.

Proposition 1 [9]

The $\{\rho_i\}_{i=\overline{1,Nw}}$ set of conjugate directions is linearly independent.

Remark

Let assume that the starting point for the search is W_0 and the minimum point is W^*. Then it may be possible to write :

$$W^* - W_0 = \sum_{i=1}^{N_w} \gamma_i \rho_i \quad \ldots\ldots\ldots\ldots\ldots\ldots \text{(5)}$$

Where γ_i are some parameters. Finding W^* may be envisaged by successive steps (of length γ_i along directions ρ_i) in the form :

$$W_{i+1} = W_i + \gamma_i \rho_i \quad \ldots\ldots\ldots\ldots \quad (6)$$

where $i = 0,\ldots, N_W$ and $W_{Nw+1} = W^*$.

By multiplying (5) with $\rho_\ell^T H$ to the left and using (3) plus the property (4)

$$\rho_\ell^T H \ (W^* - W_0) = - \rho_\ell^T b - \rho_\ell^T H \ W_0 = \sum_{i=0}^{N_w} \gamma_i \rho_\ell^T H \rho_i = \gamma_\ell \rho_\ell^T H \rho_\ell$$

and then the γ_ℓ steps are :

$$\gamma_\ell = - \frac{\rho_\ell^T (b + HW_0)}{\rho_\ell^T H \rho_\ell} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(7)}$$

The γ_ℓ coefficients may be put into another form. From (6) :

$$W_{i+1} = W_1 + \sum_{j=1}^{i} \gamma_j \rho_j$$

Multiplying with $\rho_{i+1}^T H$ to the left and using again (4) :

$$\rho_{i+1}^T H W_{i+1} = \rho_{i+1}^T H W_0 \rightarrow$$

by (2) $$\rho_{i+1}^T g\,|_{W_{i+1}} = \rho_{i+1}^T (b + HW_{i+1}) = \rho_{i+1}^T (b + HW_0)$$

and by using this result in (7) we have another (new) form of γ_ℓ :

$$\gamma_\ell = - \frac{\rho_\ell^T g\,|_{W_\ell}}{\rho_\ell^T H \rho_\ell} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(8)}$$

Proposition 2 [9]

If the weight vector is updated according to the procedure (6) the gradient of the error function at step i+1 is orthogonal on all previous conjugate directions

$\rho_j^T g\,|_{W_i} = 0, \quad \forall\, i , j$ such that $j < i \leq N_w$ ……………(9)

The set of conjugate directions { ρ_i } may be built as follows :

1. The first direction is chosen as : $\rho_0 = - g|_{W_1}$

2. The following directions are built incrementally as :

$$\rho_{i+1} = - g|_{W_{i+1}} + \beta_i \rho_i \ldots\ldots\ldots\ldots\ldots\ldots(10)$$

Where β_i are coefficients to be found such that the newly build ρ_{i+1} is conjugate with the previous ρ_i, i.e. $\rho_{i+1}^T H\rho_i = 0$. By multiplying (10) with H ρ_i to the right, we get a new form of β_i :

$$(-g_{i+1} + \beta_i\rho_i)^T H\rho_i = 0 \quad \Rightarrow \quad \beta_i = \frac{g_{i+1}^T H\rho_i}{\rho_i^T H\rho_i} \quad \ldots\ldots\ldots\ldots(11)$$

Proposition 3 [9]

By using the above method for building the set of directions, the error *gradient* at step j is *orthogonal* on all previous ones :

$$(g|_{Wj})^T g|_{Wi} = 0 \quad ,\forall \text{ j,i such that } j < i \leq N_W \quad \ldots\ldots\ldots\ldots\ldots\ldots(12)$$

Proposition *4* [9]

The set of directions build by the above method are mutually conjugate.

Remarks :

1- The method described in this section gives a very fast converging method for finding the error minima (optimal weight), i.e. the number of steps required equals the dimensionality of the weight space.

2- The previous section give the general method for fast finding the minima of E(optimal weight). However there are 2 remarks to be made

- The error function was assumed to be quadratic.
- For a non-quadratic error function the Hessian is variable and then it has to be calculated at each W_i point which results into a very computational intensive process.

3- For the general algorithm it is possible to express the η_i and β_i coefficients without explicit calculation of Hessian. Also while in practice the error function is not quadratic the conjugate *gradient* algorithm still gives a good way of finding the error minimum point (optimal weight).

Now, we give several ways to express the β_k coefficients which speed the training algorithm :

1) Abd – Al-Jabbar – Luma update (**traincgAJ**) procedure $\beta_k = -\frac{g_k^T g_k}{\rho_{k-1}^T g_k}$

2) Layla – Luma update (**traincgll**) procedure $\beta_k = \frac{\Delta \mathbf{g}_{k-1}^{\mathbf{T}} \mathbf{g}_k}{\mathbf{g}_{\mathbf{k}}^{\mathbf{T}} \mathbf{g}_{\mathbf{k}}}$

3) Alaa update (**traincgak**) procedure $\beta_k = \frac{g_k^T \Delta g_k}{g_{k-1}^T \Delta g_k}$

4.Learning Parameter and Convergence for Conjugate Gradients Method

Let consider a quadratic error as function of γ_i :

$$E(W_i + \gamma_i \rho_i) = E_0 + b^T (W_i + \gamma_i \rho_i) + \frac{1}{2} (W_i + \gamma_i \rho_i)^T H (W_i + \gamma_i \rho_i)$$

The minimum if error along the direction given by ρ_i is found by imposing the cancellation of its derivative with respect to γ_i :

$$\frac{\partial E}{\partial \gamma_i} = 0 \quad \Rightarrow \quad b^T \rho_i + (W_i + \gamma_i \rho_i)^T H \rho_i = 0$$

and considering the property $x^T y = y^T x$ and the fact that $g = b + H W$, then we get a new form to γ_i :

$$\gamma_i = \frac{\rho_i^T g_i}{\rho_i^T H \rho_i} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(13)$$

The fact that formula (13) coincide with expression (8) indicate that the procedure of finding these coefficients may be replaced with any procedure for finding the error minima along ρ_i direction.

5. Application

We applied multilayer FFNN with ridge basis function have linear output units and a single hidden layer of hyperbolic tangent hidden units (nodes). The number of hidden nodes in all problems is 2N+1, where N is number of input nodes. The training problems used problem domains function approximation and we training each problems 9 different times and the weights of the networks computed by back propagation algorithm with training algorithm :

(traincgAJ , traincgll , traincgak)

Problem 1

$F(x) = 3x (x - 0.6)(x + 1.17)$; where $0 \le x \le 1$

The numerical results of ridge basis function FFNN with network structure 1–3 –1, introduced in table (1) ,figure (2) illustrate the target function of FFNN training by " traincgak " .

Figure (1), illustrate the deviation ΔF(x) of the approximate results by using traincgak and traincgll training algorithms from the exact function .

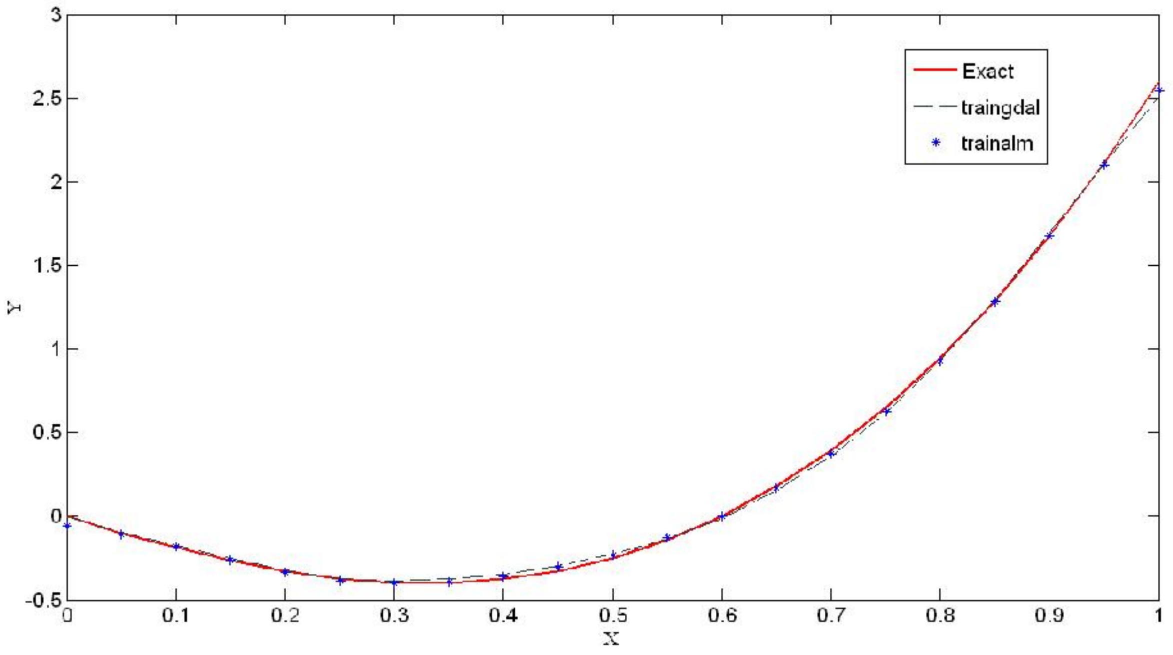

Figure (1) : the deviation ΔF(x) of the approximate results by using " traincgll " and " traincgak " from the exact function

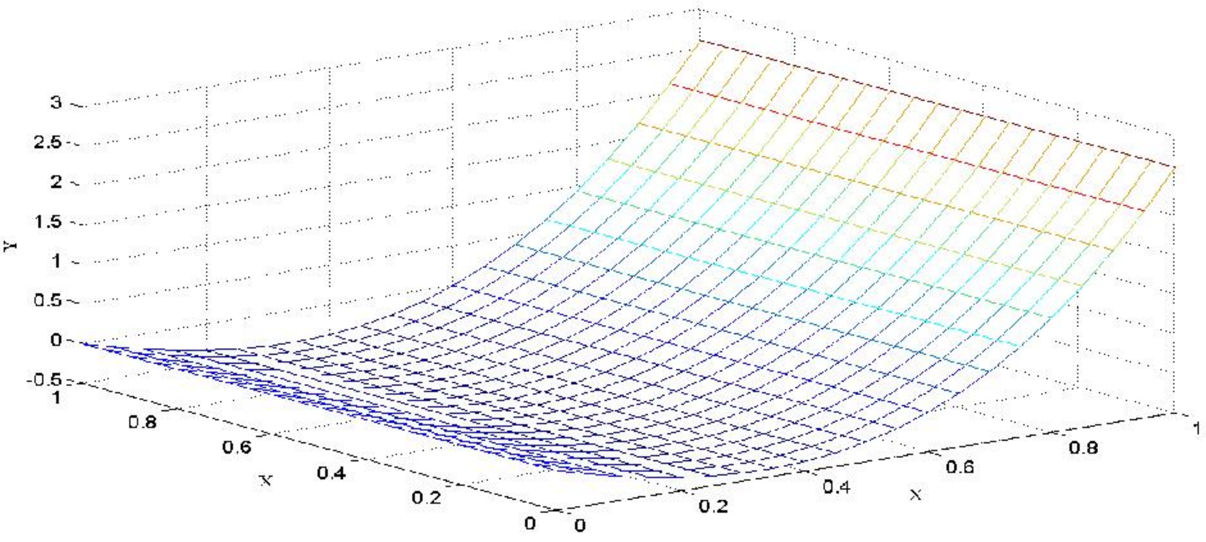

Figure (2) : illustrate the target function of FFNN training by " traincgak ".

The training Epochs and Performance				
NO.	TrainFcn	PerfTest	Time/S	Epochs
1	traincgak	0.00038264	0.39	79
2	traincgll	0.00055329	5.765	1159
3	traincgAJ	0.00055776	609.406	103391

The initial Weight and Bias value			
net.IW{1,1}	net.IW{1,1}	net.IW{1,1}	net.IW{1,1}
-8.4	-0.3944712	8.4	0.395796964
-8.4	0.08334771	4.2	
8.4	-0.69825405	0	

The parameters	
Lr	0.4
Mc	0.1
lr_inc	1.1
lr_dec	0.4
max_perf_inc	1.02
delta0	0.13
delt_inc	1.4
delt_dec	0.7
delt_max	50
Epochs	500000
Time	inf
Show	500
Goal	0.0005

Table (1) : Training results after several independent trials for problem 1

Problem 2

$F(x) = (x_1 - x_2)^3 + 1.8\, x_1 x_2 - x_1 + 7x_2$; where $0 \le x_1 \le 1$, $0 \le x_2 \le 1$.

The numerical results of ridge basis function FFNN with network structure 2 – 5 –1, introduced in table 2 .Figure (3) illustrate the exact function for problem 2 and the target function of FFNN training by " traincgak " illustrated by figure (4) .

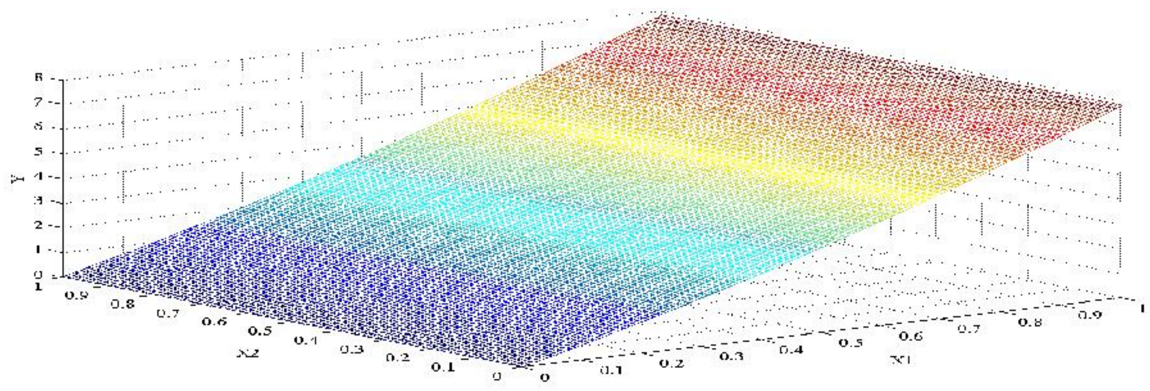

Figure (3): exact function for problem 2

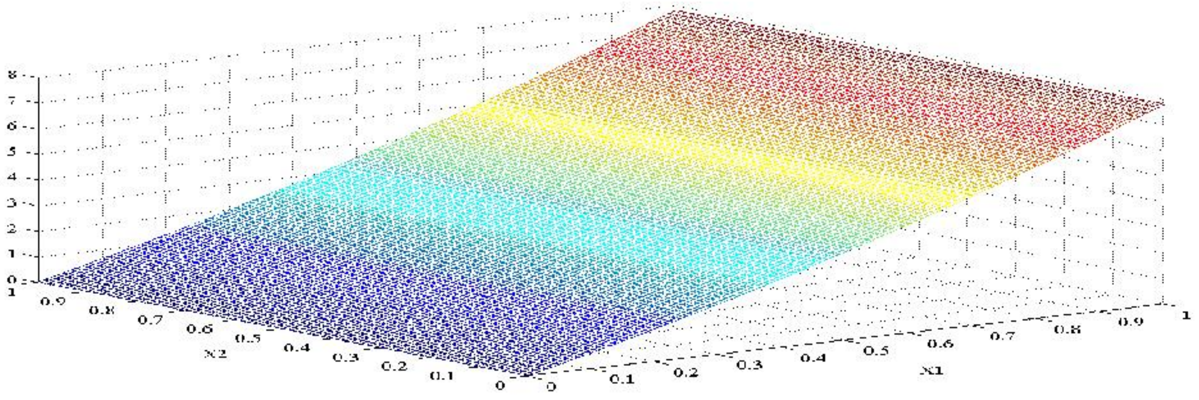

Figure (4) : illustrate the target function of FFNN training by " traincgak "

NO.	TrainFcn	PerfTest	Time/S	Epochs
1	traincgak	5.88E-05	0.75	178
2	traincgAJ	7.07E-05	701.969	145232
3	traincgll	9.50E-05	4.687	1081

The parameters	
Lr	0.24
Mc	0.4
lr_inc	1.01
lr_dec	0.8
max_perf_inc	1.01
delta0	0.2
delt_inc	1.3
delt_dec	0.7
delt_max	50
Epochs	500000
Time	inf
Show	1000
Goal	0.00005

The initial Weight and Bias value				
net.IW{1,1}		net.LW{2,1}	net.B{1,1}	net.B{2,1}
5.410613386	3.150438507	0.230864696	-7.411021115	-0.18858757
-6.1805004	-1.000707154	0.583874075	5.155851361	
1.356304956	-6.112318453	0.843625941	2.378006748	
-0.27280126	6.255044322	0.476414492	-4.556369115	
6.199395094	-0.876071039	-0.647467711	0.468833141	

Table (2) : Training results over several independent trials for problem 2

Problem3

$F(x_1, x_2) = (x_1^2 - x_2^2) \sin(5x_1)$ where ; $x_1 = -1 : 0.2 : 1$; $x_2 = -1 : 0.2 : 1$;

This problem solved in [12] by a modified structure that employs a set of locally weighted basis functions and use sum of product neural network (SOPNN) with gradient descent training algorithm 'traingd' with learning rate 0.1 . Figure (7) shows the result of the approximation of SOPNN where the exact function for problem 3 given in figure (5).

In this paper we use a multilayer FFNN with structure : 2 - 5 -1 with hyperbolic tangent hidden neuron and linear output neuron with ' traincgAJ ' training algorithm. It's clear that the method suggested in this paper (see figure (6)) is most accurate that the method used in [12] (see figure (7)).

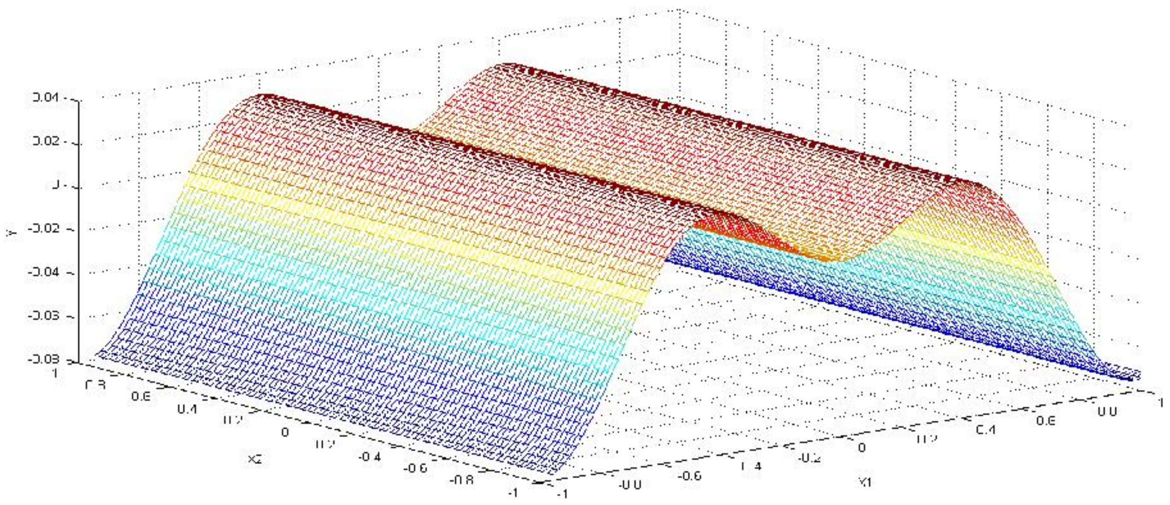

Figure 5 : exact function for problem 5

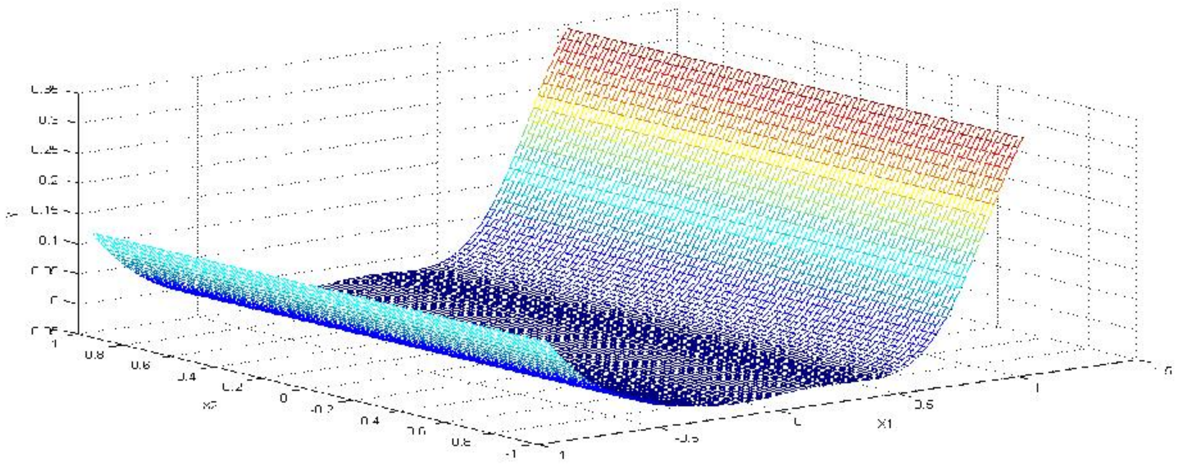

Figure (6): illustrate the target function of FFNN training by "**traincgAJ** "

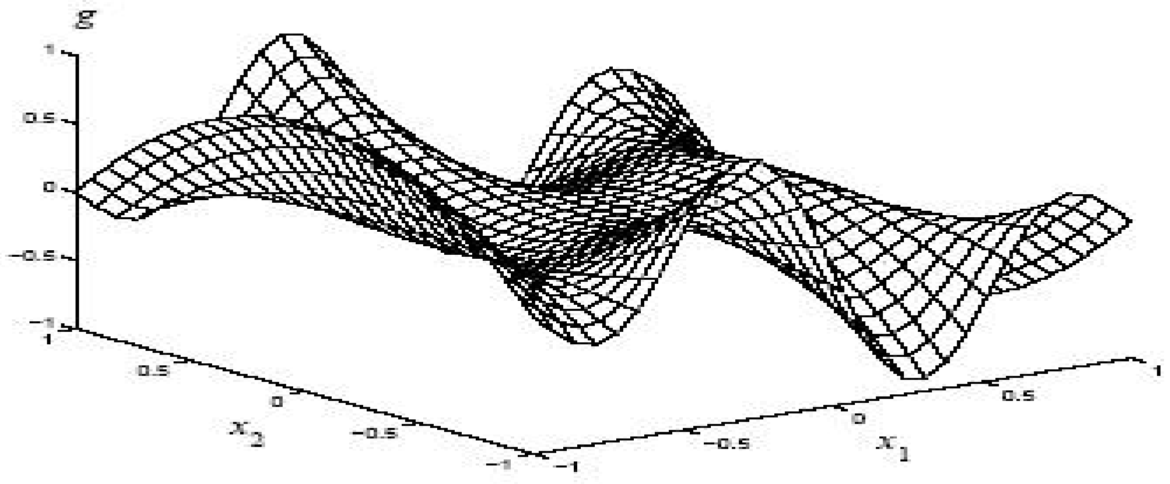

Figure (7): Target function generator by method in [12] of problem 3

References

[1] G.-B. Huang, "Real -Time Learning Capability of Neural Networks ", IEEE Transactions on Neural Networks, VOL. 17, NO. 4, JULY 2006 .

[2] W. W. Hsieh, "Machine Learning Methods in Environmental Sciences Neural Networks and Kernels", Cambridge University Press, August 31, 2008.

[3] G. Weir - Smith and C. A. Schwabe, " Spatial interpolation vs neural network propagation as a method of extrapolating from field surveys ", GIS Centre,HSRC (Human Sciences Research Council), Pretoria, 2002.

[4] J.M. Turmon ,"Assessing Generalization of Feed forward Neural Networks", phD. thesis, University of Cornell, August 1995.

[5] M.A.Ali, S.D.Gore and M. AL-Sarierah , " The Use of Neural Network to Recognize the parts of the Computer Motherboard ", Journal of Computer Sciences 1(4), pp. 477- 481, 2005.

[6] J.ILONEN, J.-K.KAMARAINEN and J.LAMPINEN, "Differential Evolution Training Algorithm for Feed-Forward Neural Networks", Neural Processing Letters 17: pp. 93 – 105, 2003.
[7] S. Breutel, "Analysing the Behaviour of Neural Networks", PhD thesis, Queensland University of Technology, Brisbane, 2004.
[8] T. Su , J. Jhang and C. Hou , " A Hybrid Artificial Neural Networks and Particle Swarm Optimization for Function Approximation", International Journal of Innovative Computing, Information and Control ICIC International, Volume 4, Number 9, September 2008, pp. 2363 — 2374.
[9] R. M. Hristev , " The ANN Book " , Edition 1, 1998.
[10] G.P.Jaya Prakash and TRBstaff Representative,"Use of Artificial Neural Networks In Geomechanical And Pavement System " , Transportation Research Circular, Number E-co12 ,December 1999.
[11] L.N.M.Tawfiq and R.S.Naoum, "On Training of Artificial Neural Networks", AL-Fath Jornal, No 23, 2005.
[12] K. A.H. Al-Hindi, "A Modified Sum-of-Product Neural Network Using Locally Weighted Basis Functions", Umm Al-Qura Univ. J. Sci. Med. Eng. Vol. 18, No.2, pp.191 -206 (2006).

تحسين طريقة انحدار الميل في تدريب الشبكات العصبية ذات التغـذية التقـدمية

د. لمى ناجي محمد توفيق و علاء كامل جابر
قسم الرياضيات – كلية التربية –أبن الهيثم – جامعة بغداد

المستخلص

في هذا البحث اقترحنا عدد من الخوارزميات المطورة والجديدة لتدريب الشـبكات العصـبية ذات التغذية التقدمية البعض منها تمتلك سرعة تقارب جيدة للشبكات ذات التركيب المعقول.
في كل تلك الخوارزميات انحدار دالة الأداء (دالة الخطأ، دالة الطاقة) لتحديد كيفية ضـبط الأوزان بحيث تكون دالة الأداء أقل ما يمكن . حيث استخدمنا خوارزمية الانتشـار المرتـد لتسريع التدريب ،
جميع الخوارزميات أعلاه تتنوع من حيث اختلاف الحسابات وبالتالي اختلاف الأنواع حسب الصيغ لاتجاه التفتيش والخزن الذي تقتضيه وكل الخوارزميات أعلاه طبقـت فـي مسـائل التقريب .