"ONE-STEPSECANT" A GEOMETRICAL SHAPE RECOGNITION ALGORITH

Anjali Gangwar[1], Amrita Kaur[2], Deep Kumar Sharma[3]

[1,2,3]PG Scholar, SRMSCET, Bareilly (India)

ABSTRACT

The purpose of this paper is to recognise the geometrical shapes likesquare and ellipse by using the "one-step-secant" algorithm of neural network. Firstly, a neural network with two layers and two input vectors is built. The first layer has twenty neurons, while the second one includes only two neurons. Then a training base and a test base through generating "rand" function is created . Each of the base consists of one hundred shapes: fifty squares and fifty ellipses. Lastly, testing of the network with the help of a performance function (MSE=Mean Squared Error) and "one-step-secant" algorithm takes place.

Keywords: Neural Network, Epoch, MSE

I. INTRODUCTION

There is no commonly accepted definition of a neural network. But, most people in the field would agree that a neural network is a network of many simple mainframes ("units"), each possibly able with a small extent of local memory. The units are linked by communication networks ("connections"), which typicallytransmit numeric (as opposed to symbolic) data, determined by several means. The units function only on their local data and on theinputs they receive via the networks. The control to local actions is often comfortable during training [1], [2].

Neural network can be divided into three architectures, namely single layer, multilayer network and competitive layer. In a net, the layers number can be defined on the basis of a number of interconnected weights in a neuron. A single layer network consists in only one layer of connection weights, whereas, a multilayer network consists in more than one layer of connection weights. The network also contains an additional layer called hidden layer. Multilayer networks can be used to solve more complicated problems compared to single layer network. Both of the network are also called feed-forward network where the signal flows from the input units to the output units in a forward direction [2].

The inverse error propagation algorithm has been created through simplification of a learning rule of Widrow-Hoff of multitask networks and differential and nonlinear transferring functions. In this paper we used a variant of the opposite error propagation algorithm.

Input vectors and equivalent "target" vectors are used to train the network until this can estimated a function, connecting input vectors with specific output vectors, or categorize the input vectors in a user mode specification.

Standard algorithm of opposite error propagation is related to the gradient decrease. The notion of inverse propagation of error is similar to the manner in which the gradient is computed for nonlinear multitask networks. There are numerousoperations for the standard algorithm that are based on other standard optimization techniques, like the conjugate gradient method or the Newton method.

Networks with opposite training error propagation tend to offer practicalanswers. This is happening in a suitable way, when input values which were not seen before, are introduced. Generally, new sets of input values are leading to similar outputs such as correct output (target output) for input vectors used in training. Those are similar to the new sets. Thissimplification property allows entertaining a network on a characteristic set of input/output pairs. It is also conducting to satisfactory results without network training on all the other possible input/output pairs.

II. BACKGROUND

In most cases, Training a neural network is an exercise in numerical optimization of a usually nonlinear objective function ("objective function" means whatever function you are trying to optimize and is a slightly more general term than "error function" in that it may include other quantities such as penalties for weight decay.

Methods of nonlinear optimization have been studied for hundreds of years, and there is aenormousworks on the subject in fields such as mathematical analysis, operational research, and statistical computing (e.g. [3], [4]). Masters in [5] has a good basicconversation of conjugate gradient and Levenberg- Marquardt algorithms in the environment of neural networks.

There is no single best method for nonlinear optimization. You need to choose a method based on the features of the problem to be solved. For independent functions with continuous second derivatives (which would include feed-forward nets with the most popular differentiable activation functions and error functions), three universal algorithms have been found to be operative for most practical purposes:

- For a small number of weights, stabilized Newton and Gauss-Newton algorithms, including various Levenberg-Marquardt and trust-region algorithms, are efficient. The memory required by these algorithms is proportional to the square of the number of weights.
- For a moderate number of weights, various quasi-Newton algorithms are efficient. The memory required by these algorithms is proportional to the square of the number of weights.
- For a large number of weights, various conjugate-gradient algorithms are efficient. The memory required by these algorithms is proportional to the number of weights.

In most applications, it is suitable to train several networks with different numbers of hidden units. Rather than train each network, start with completely random weights, it is usually more effective to use constructive learning. Constructive learning can be done with any of the predictable optimization methods or with the various "prop" methods, and can be very effective at finding good local optima at less expense than full-blown global optimization methods.

III. METHOD

This paper presents the first part of ancomprehensive project, which we need to achieve. Thus, we mean to discover the geometrical shapes described by a person's movement through the air. However, to achieve this, first we have to create a transferable device which could offer the necessary plots on the path. In this paper we used geometrical shapes in the xOy (bidimensional) plane, i.e. we used the bidimensional case. The next step would be to spread these to the xyz plane, for the tridimensional plane.

In this paper we recommend the operation of geometrical shape recognition: Square and geom. ellipse, for a given number of points. This paper starts from the concept of recognition of the geometrical shapes traced by one person through the air.

Quasi-newton method involves generating a sequence of matrices $G^{(k)}$that represents gradually accurate estimates to the inverse hessian $H^{(-1)}$.Using only the first derivative information of E, the updated expression is asfollows:

$$G^{(k+1)} = G^{(k)} + \frac{pp^T}{p^T v} - \frac{\left(G^{(k)} v\right) v^T G^{(k)}}{v^T G^{(k)} v} + \left(v^T G^{(k)} v\right) u u^T \qquad (1)$$

where

$$p = w^{(k+1)} - w^{(k)},$$

$$v = g^{(k+1)} - g^{(k)},$$

$$u = \frac{p}{p^T v} - \frac{G^{(k)} v}{v^T G^{(k)} v} \qquad (2)$$

and T represents transpose of a matrix. The problem with this approach is the necessity of calculation and storage of the estimated Hessian matrix for every iteration. The One-Step-Secant (OSS) is an approach to bridge the gap between the conjugate gradient algorithm and the quasi-Newton (secant) approach. The OSS approach doesn't store the complete Hessian matrix; it assumes that at each iteration the previous Hessian was the identity matrix. This has the advantage that the new search direction can be calculated without calculating a matrix inverse [2].

Newton's method is ansubstitute to the conjugate gradient methods for fast optimization. The basic step of Newton's method is

$$x_{k+1} = x_k - A_k^{-1} g_k \qquad (3)$$

where*k A* is the Hessian matrix (second derivatives) of the performance index at the current values of the weights and biases. Newton's method frequently converges quicker than conjugate gradient methods. Inappropriately, it is complex and costly to calculate the Hessian matrix for feed forward neural networks. There is a class of algorithms that is based on Newton's method, but which doesn't need calculation of second derivatives. These are called quasi-Newton (or secant) methods. They update an approximate Hessian matrix at each iteration of the algorithm. The update is calculated as a function of the gradient.

The quasi-Newton method has been most effective is the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update. This algorithm has been implemented in the „trainbfg" routine. The BFGS algorithm is described in [6].

Since the BFGS algorithm needs more storage and calculationin each iteration than the conjugate gradient algorithms, there is need for a secant approximation with smaller storage and calculationnecessities. The one step secant (OSS) method is an effort to link the gap between the conjugate gradient algorithms and the quasi-Newton (secant) algorithms. This algorithm does not store the complete Hessian matrix; it assumes that at each iteration, the previous Hessian was the identity matrix. This has the additional advantage that the new search direction can be calculated without computing a matrix inverse [1].

This algorithm needs more calculation in each iteration and more storage than the conjugate gradient methods, although it usually converges in less iteration. The approximate Hessian must be stored, and its dimension is n*n, where n is equal to the number of weights and biases in the network.

For very large networks it may be better to use strong back-propagation (Rprop) (in the „trainrp" routine) or one of the conjugate gradient algorithms. For smaller networks, however, „trainbfg" (BFGS quasi-Newton back-propagation) can be an efficient training function.

However, for complex networks, where number of connection is great (large), this algorithm is not very fast because it needscontrol and the notice Hessian approximate matrix. Full view handled problem in which we have twenty neurons on first layer and two input vectors, is necessitating a secant approximation with small requirements of control and notice. Therefore, in this case we used One-Step-Secant algorithm.

General description of method:

To pretend the giving out of the coordinate To pretend the processing of the coordinate points taken from the above-mentioneddevice,square and geom. ellipses have been generated in a random manner. Meaningful this and the fact that a person will not define, usually, perfect geometrical shapes, some expectations were measured for obtaining a real case:

- The shapes are traced anywhere in a requirement area, angle in down left (for

geom. ellipse, angle in down left of square what framing) full of random coordinates with a uniform distribution (abscissa respective angle there is between 0 and 100; likewise and ordinate), thus permitting a wide-ranging position of the traced form, like in the real case, when outlining is made inside a room with magnitudes from conditions.

- The shapes are drawn anywhere inside a specified area, the left lower corner (for geom. ellipse, the left lower corner of thesquare which surrounds it) having random coordinates with a constantdelivery (abscissa of the respective corner is between 0 and 100; likewise the ordinate). Thus a variedplacing of the drawn shapes is possible, like in the real case, when the outlining is done inside a room with quantifiedmagnitudes.

- The magnitudes of the shapes are createdcasually (again with an uniform distribution, so as not to create preferential dimensions).s

- Each coordinate of each point is affected by a constant noise, thus, allowing a tracing with limitations, exactly like in the real case.

The project was realized in Matlab version 7.0 and it is based on the idea of neural networks. The operation of the algorithms specific to neural networks was made with the use of the Neural Network Toolbox in Matlab. Neural networks are composed of simple elements which function in equivalent. These elements are encouraged from the biological nervous systems. As in nature, the function of network is determined in large by the connections between elements. A neural network may be skilled to realize certain function by set the values of the connections (synapses or weights) between elements. Generally, the neural networks are set or skilled, so that a certain set of input values would lead to a value of expected output (a target). Such a situation is presented in Fig. 1. The network is set through the association between the output value and the target (the expectedvalue), until the output of network approaches the target, with a given equalizer. In universal, many such input/target pairs are used for

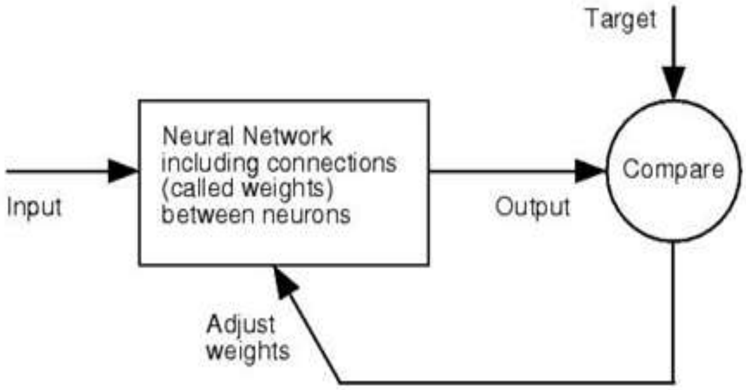

Fig. 1 - The comparison between the output value and the target

exercisenetwork.Neural networks have been trained to understand complex functions in wide-rangingapplications such as: shapes recognition (as in this case), classification, speech and voice signal recognition, control systems, medical imagistic and many others.

The area of neural networks has a history of about five eras, but has found solid application only in the last 20 years and continues to develop in fastermeasure. Thus, the idea of neural network is completely dissimilar from the traditional ideasindirect in areas such as control systems or the optimization of the systems where the terminology (mathematical statistics) and the designing actions were established and applied for many years.

Implementation procedure:

The first step is to build up a neural network with two layers and two input vectors. The first input vector contains abscissa points, and the second one contains ordinate points. Regarding the layers; the first one has twenty neurons, while the second one includes only two neurons.

The second step is to create a training base and a test base using the “rand” function. Each base contains one hundred shapes: fifty squares and fifty geom. ellipses.

The third step is testing the network by using a performance function (MSE=Mean Squared Error, where the error value is the amount by which the value output by the network differs from the training value. For example, if we required the network to output 0 and it output a 1, then Err = -1) and “one-stepsecant” algorithm. To assure the convergence towards the expected value while on any training set we realize a “while” loop that iterates the network initialization.

The result of the testing made one the exercise base must be under the value 1e-5; while for the test base is under the value 1e-4. These results are the errors that are reasonable enough for a correct classification. Further to these errors, the function also displays a value in which the development parameters are stored during the training. This is called epoch (An epoch is the presentation of the entire training set to the neural network. For example, in the case of the AND function an epoch consists of four sets of inputs being presented to the network (i.e. [0,0],
[0,1], [1,0], [1,1])).

The gradient of a function of two variables $F(x, y)$ is defined as:

$$\nabla F = \frac{\partial F}{\partial x}\hat{i} + \frac{\partial F}{\partial y}\hat{j}$$

and can be supposed of as a collection of vectors pointing in the direction of growing values of *F*. In Matlab, numerical gradients (differences) can be calculated for functions with any number of variables.

Algorithm “trainoss” [1] can train any network as long as its weight, net input, and transfer functions have derived functions. Back-propagation is used to calculate derivatives of performance perf with respect to the weight and biasvariables X. Each variable is adjusted according to the following [9]:

$$X = X + a * dX \quad (5)$$

where$dX$is the search direction. The parameter is selected to minimize the performance along the search direction. The line search function searchFcn is used to locate the minimum point. The first search direction is the negative of the gradient of performance. In successive iterations the search direction is calculated from the new gradient and the previous steps and gradients according to the following formula:

$$dX = -gX + Ac * X_{step} + Bc * dgX \quad (6)$$

where$gX$is the gradient, *step X* is the change in the weights on the previous iteration, and $dgX$is the change in the gradient from the last iteration. (For a more detailed discussion of the one step secant algorithm see [7]).

Training stops when any of these conditions occur:

1. The maximum number of epochs (repetitions) is reached.
2. The maximum amount of time has been exceeded.
3. Performance has been minimized to the goal.
4. The performance gradient falls below min-grad.
5. Validation performance has increased more than max_fail times since the last time it decreased (when using validation).

The program can be called through the function “rec_form”. This has a facultative parameter which represents the number of points upon which the latter training and recognition are made. The understood value of this parameter is 48. To be used for recognition, the trained network is used as a parameter for the „sim” function which verifies the behavior of the network on a shape inserted from the keyboard. If the function returns the value 0 1, the network has recognized the introduced shape with a square but if the returned value is 0 0, the network has recognized the introduced shape as being an ellipse.

Training the network is time consuming. It generallyacquires after several epochs, depending on how large the network is. Thus, large network essential more training time compared to the smaller one. Basically, the network is trained for several epochs and stopped after reaching the maximum epoch. For the same reason minimum error tolerance is used provided that the differences between network output and known outcome are less than the specified value. We could also stop the training after the network encounters certain stopping criteria. During training the network powerstudy too much.

For this project during training, authentication set is used in its place of training data set. After a few epochs the network is tested with the authentication data. The training is stopped as soon as the error on authentication set increases quickly higher than the last time it was checked [8].

IV. EXPERIMENTAL RESULTS

In this paper we analyzed three cases. In first case, the performance has met at epoch number 52 of 500 (500 is number maxim of epochs in our case), in the second case the performance has attained at epoch number 146 of

500 and the third case the performance has met at epoch number 148 of 500. Results obtained in those there cases analyzed in this article are:

In Table 1, 2 and 3 we have MSE and gradient at some epochs of those analyzed, until met the performance. Fig. 2 (a), (b) and (c) shows network when met the performance that is at

maximum epoch.

(a) Represent the first case;

(b) Represent the second case;

(c) Represent the third case.

Fig. 3 (a), (b) and (c) shows network after training. Fig. 4 (a), (b) and (c) shows results obtained with this method.

(a) For the first case we have the following results:

Table 1

The performance system after epoch number 52

Epoch	MSE /1e-005	Gradient
0 of 500	0. 377322	0. 505532
25 of 500	0. 0381433	0. 648829
50 of 500	7.70679e-005	0.0338334
52 of 500	5.45577e-006	0.000132798

In first case the performance has met after epoch number 52.

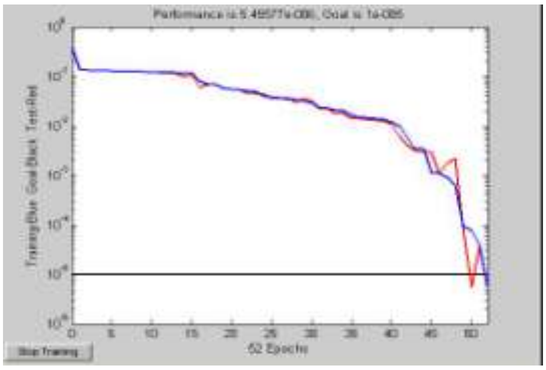

Fig. 2 (a) Maximum Epoch (in First Case is 52)

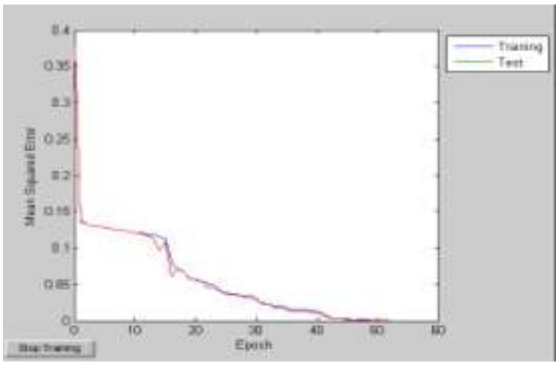

Fig. 2 (b) MSE After Training Networks in First Case

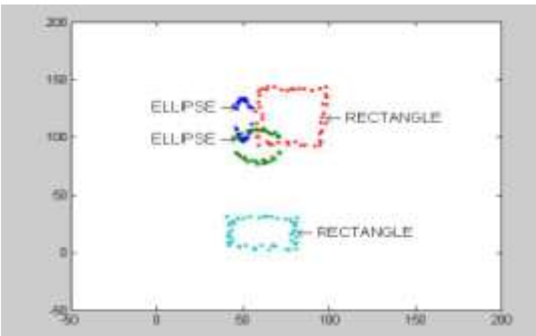

Fig. 2 (C) Results Obtained After Training Network in the First Case

(b) For the second case we have:

Table 2

The performance system met after epoch number 146

Epoch	MSE /1e-005	Gradient
0 of 500	0.511977	0.568587
25 of 500	0.0560792	0.16273
50 of 500	0.00763521	0.132888
75 of 500	0.000238334	0.000582092
100 of 500	0.000123981	0.000801856
100 of 500	2.34438e-005	5.7238e-005
146 of 500	9.38845e-006	5.50573e-005

In second case the performance has met after epoch number 146.

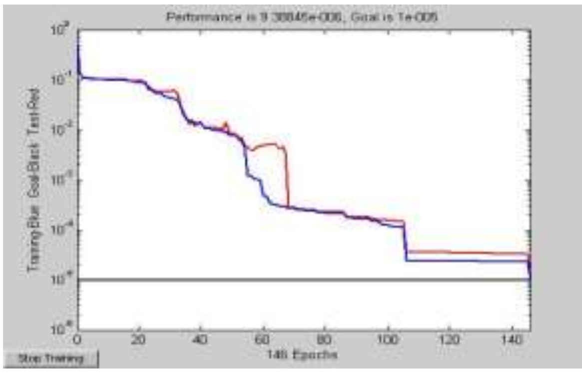

Fig. 3 (A) Maximum Epoch (In Second Case Is 146)

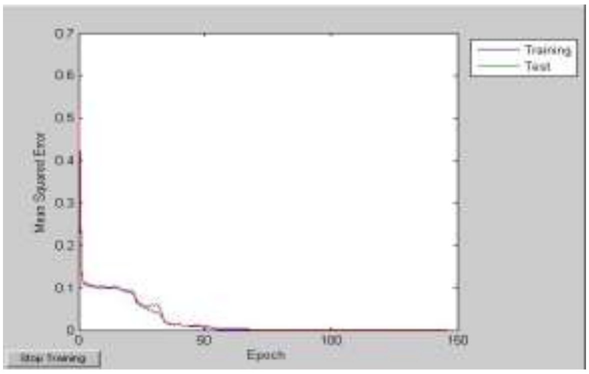

Fig. 3 (B) MSE After Training Networks In Second Case

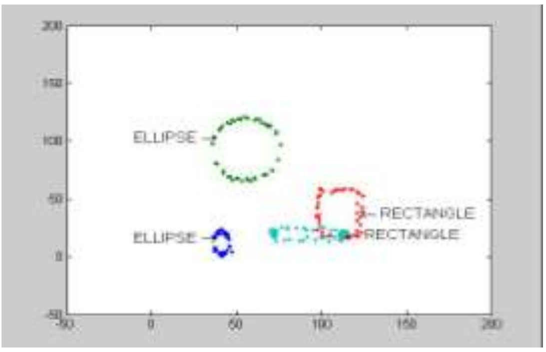

Fig. 3 (C) Results Obtained After Training Network in Second Case

(c) For the third case we have:

Table 3

The performance system met after epoch number 148

Epoch	MSE /1e-005	Gradient
0 of 500	0.292855	0.511591
25 of 500	0.029315	0.275339
50 of 500	0.00040364	0.00105794
75 of 500	1.58091e-005	3.21924e-005

100 of 500	1.53045e-005	4.51943e-005
125 of 500	1.50264e-005	3.37123e-005
141 of 500	2.71875e-006	1.39385e-005

In third case the performance has met after epoch number 148.

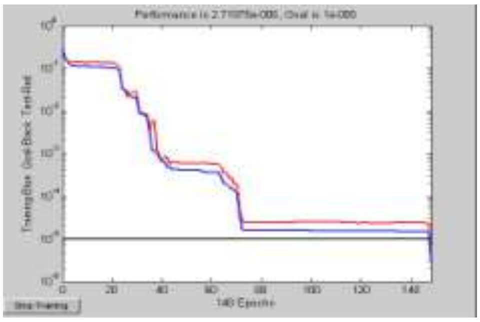

Fig. 4 (A) Maximum Epoch (In Third Case Is 148)

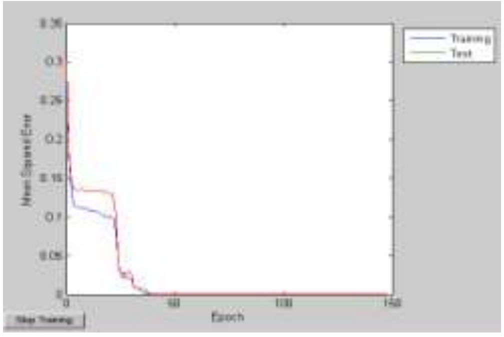

Fig. 4 (B) MSE After Training Networks In Third Case

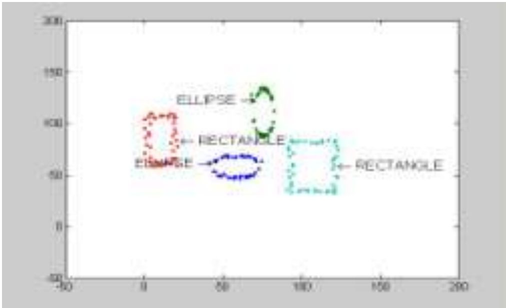

Fig. 4 (C) Results Obtained After Training Network in Third Case

V. CONCLUSIONS

For this project, the conjugate gradient algorithm influences a similar performance in a shorter time similar with "one-step-secant" method, with the variance that in some of the training cases the program blocks during the training time. Also, sometimes during the training the error "Divide by zero" arises. Therefore, one-step-secant algorithm is a sufficient algorithm that may achieve training without blockage during the running time of the program.

One-Step-Secant method representing a cooperation solution between conjugate gradient algorithms (methods with requirements low calculation) and quasi-Newton algorithms and this method no stocking complete Hessian matrix, but suppose that at each iteration previous Hessian matrix is identity matrix. This thing have additional advantage that new pursuit direction can be computing without computing inverse matrix, so in the case of Quasi-Newton algorithms.

This method may deal a helpful support for designing different geometrical shapes in many applications of current interest.

On the other hand, there are some points that should be better in additional work, such as improving the network algorithm, improving the simplification ability, etc. If the values of the performance function on training set is over 1e-5 value and on test set is over 1e-4, our program would be blocked. Helped by this program, we'll obtain the best results using the values of text frontstated.

In the future we propose to develop this application for more values of the performance function. In the near days to come, we will get new and more complete results for the performance function. Also we will spread the functionality of our algorithms so as to be able to make online plot of geometrical shapes with the use of satisfactorytransferable devices.

REFERENCES

[1] ***Matlab Help – Neural Network Toolbox

[2] Bishop C. M., Neural Networks for Pattern Recognition, Oxford Press, 1995.

[3] Bertsekas, D. P. (1995), Nonlinear Programming, Belmont, MA: Athena Scientific, ISBN 1- 886529-14-0.

[4] Bertsekas, D. P. and Tsitsiklis, J. N. (1996), Neuro-Dynamic Programming, Belmont, MA: Athena Scientific, ISBN 1-886529-10-8.

[5] Masters, T. (1995) Advanced Algorithms for Neural Networks: A C++ Sourcebook, NY: John Wiley and Sons, ISBN 0-471-10588-0

[6] Dennis, J.E., and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Englewood Cliffs, NJ: Prentice-Hall, 1983.

[7] Battiti, R., First and second order methods for learning: Between steepest descent and Newton's method, Neural Computation, **Vol. 4**, No. 2, 1992, pp. 141-166.

[8] Prechelt, L. (1998). Early Stopping-but when? Neural Networks: Tricks of the trade, (pp. 55- 69). Retrieved March 28, http://www.ipd.ira.uka.de/~prechelt, 2002.

[9] C. Aykanat, K. Oflazer, R. Tahboub. “Parallel Implementation of the Back-propagation Algorithm on Hypercube Systems, in Proceedings of the NATO Advanced Study Institute on Parallel and Distributed Computing, (Springer Verlag), Ankara, Turkey, July 1991.