COMPARISON BETWEEN LEVENBERG-MARQUARDT AND SCALED CONJUGATE GRADIENT TRAINING ALGORITHMS FOR IONOSPHERE CONDITION USING MULTILAYER PERCEPTRONS

Mohd Sharif Ibrahim
Faculty of Electrical Engineering
Universiti Teknologi MARA
40450 Shah Alam, Selangor
Malaysia
*email:syarif_p3@yahoo.com

Abstract - Ionosphere is one of layer at earth's atmosphere. Ionosphere can be described a many layer and have own functional of systems at atmosphere. This paper described the examinations of two training algorithms which are Levenberg-Marquardt (LM) and Scaled Conjugate Gradient (SCG) of Multilayer Perceptrons (MLPs) using MATLAB R2009a. Based on the result, we conclude that both algorithms were comparable in terms of accuracy and speed. However, the SCG algorithm has shown better advantage in terms of accuracy and speed on the best MLP structure (with 25 hidden units).

Keywords: **Ionosphere, Leverberg-Marquardt (LM) and Scaled Conjugate Gradient (SCG)**

I. INTRODUCTION

Ionosphere is the uppermost part of the atmosphere, distinguished from other layers because due to ionization characteristics done by solar radiation. It plays an important part in atmospheric electricity and forms the inner edge of the magnetosphere. It has practical importance because, among other functions, it influences radio the propagation to the distant places on the Earth. Ionosphere has a tendency to change due to the influence from the solar activity, which make it important to study the ionosphere layer. Ionosphere has the important effects on the propagation of radio waves links between satellite and ground stations, telecommunications, and guidance and surveillance radars. There are primarily three distinct layers in the ionosphere as like D, E and F layer. Each of the layers has its own significance to the ionosphere. This paper described the F2 layer is suitable for high frequency (HF) radar. The F2 region is the most important region for HF radio propagation because it is present 24 hours of the day, high altitude the longest communications paths and reflects the highest frequencies in the HF range. This paper also described the design of classification of radar returns from the ionosphere that have been investigated using Neural Network. The radar data is obtained from John Hopkins university ionosphere database [1].

In this paper, an examination of two popular training algorithms (Levenberg-Marquardt and Scaled Conjugate Gradient) were presented for Multilayer Perceptron (MLP) simulated of received signals. The performance of the training algorithms was tested using the Ionosphere Database [1]. The database consists of features such as bad and good conditions. The Ionosphere Database has been extensively used in literature as a benchmark for testing the performances of various classification algorithms [12, 13, 16, 17].

II. RELATED WORKS

In previous work on using various classifier and represented for variety techniques of multilayer perceptron neural network on the ionosphere database in literature. In [2], a Radial Basis Function Neural Network (RBF) classifier was used to train the ionosphere. The result is showed 100% accuracy on "bad" and 99.1935% accuracy on "good" instances.

Works by [3] compared between Multi Layer Perceptron (MLP) NN and Radial Basis Function Neural Network (RBF) NN on the ionosphere database. RBFNN is operates as an excellent classifier for the given task with accuracy 99.596%.

In [4], the investigated using back propagation and the perceptron training algorithm with a back propagation an average of over 96% accuracy on the test set.

In [5], the Genetic Algorithms (GA) was used to 96% optimize the MLP network structure, as well a training the MLP. Test on the ionosphere dataset yielded 91.4% in classification accuracy.

III. THEORETICAL BACKGROUND

A. ANN and MLP

ANNs are problem-solving tools that have become an alternative modeling method to some physical and non-physical systems with scientific or

mathematical basis [6]. It mimics the process of human learning using relatively crude electronic models [7]. Just as human brains can be trained to master some tasks through experiential knowledge and training, ANNs can be trained to recognize patterns and perform optimization through a training process.

ANN offers several advantages over conventional computing [8]:

1. Flexibility. The network automatically adjusts to a new environment without using any preprogrammed instructions.
2. Non-linearity. ANNs can compute nonlinear, nonparametric functions of their input, enabling them to perform arbitrarily complex transformations of data.
3. Robustness. ANNs are tolerant of both physical damage and noisy data.

MLP is defined as a system of massively distributed parallel processor (consisting of simple processing units called neurons) that have natural tendency for storing and utilizing experiential knowledge [9]. Normally, the MLP learns the relationship between a set of inputs and outputs by updating internal interconnections called weights using the back-propagation algorithm.

In MLP, the units are arranged in interconnected layers: one input layer, one or more hidden layers, and one output layer. The numbers of input and output units are typically fixed, since they depend on the input and desired output(s). However, the training algorithm and the number of hidden units are adjustable, and can be set so that it maximizes the performance of the MLP.

The interconnections between the MLP layers (weights) are typically initialized at random prior to training. The initialized weights represent the initial points in which the MLP begins the search for the solution. It is because of this, the value of the random numbers affects the network convergence. A too large or too small initial weight values would slow down or prevent convergence.

To solve the problem, the Nguyen-Widrow (NW) algorithm [10] generates initial weight and bias values for a layer, so that the active regions of the layer's units will be distributed approximately evenly over the input space. The ANN is trained as usual, where each hidden unit still having the freedom to adjust its weights during training. However, most of the adjustments will be small since the majority movement is eliminated by the improved initial values. Selecting weights so that the hidden units are scattered in the input space will substantially improve learning speed of ANNs.

B. Early Stopping (ES)

A common problem in MLP training is over-generalization, referring to a condition where the MLP has been trained until it has memorized the data it's given, rendering it unable to adapt and generalize to new cases [11].

In order to obtain the optimum MLP generalization, the Early Stopping (ES) method divides the dataset into three sets – the training set, validation set, and the testing set. The training set is used to update the MLP weights during the training phase, and the error in the independent validation set is monitored. Early stopping chooses a point along this path that optimizes an estimate of the generalization error computed from the validation set. Since the validation set does not participate in the weight update (training) process, it can be used as a performance gauge to measure the generalization capabilities of the ANN when it encounters previously untrained cases. If the training error continues to decrease, but the validation set error has started to increase, this indicates that over-generalization has occurred, thus training is stopped. ES is widely used because it is simple to implement and understand, and has been reported to be superior to regularization methods in many cases [11].

Early stopping has several advantages:

1. It is fast.
2. It can be applied successfully to networks in which the number of weights far exceeds the sample size.
3. It requires only one major decision by the user: what proportion of validation cases to use.

C. The Levenberg-Marquardt Algorithm

The LM algorithm [12] attempts to solve a nonlinear least square minimization problem in the form of:

$$f(x) = \frac{1}{2}\|r(x)\|^2 \tag{1}$$

where r is the residual vector. The application of LM optimization to train ANNs was introduced in [13]. When searching for the minimum on the error surface, a good learning rule will logically take larger steps in flat areas to skip plateaus quickly, and takes smaller steps when it encounters a large gradient to avoid overstepping the local minima. The LM does this by combining curvature based on two update rules, the Vanilla Gradient Descent (VGD), and the Gauss-Newton (GN) rule. The LM rule is:

$$x_{i+1} = x_i - \left(H - \mu\, diag(H)\right)^{-1} \nabla f(x_i) \tag{2}$$

where H is the approximated Hessian matrix, $\nabla f(x_i)$ is the gradient of the error function, and μ is the mediating factor between GN and VGD rules. The LM algorithm was designed to approach second-order training speed without having to compute the Hessian matrix [14].

When μ is zero, the LM algorithm becomes the GN method using the approximate Hessian matrix. When μ is large, the LM algorithm becomes the VGD algorithm with a small step size. The GN method is faster and more accurate near an error minimum, so the aim is to shift towards Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm [14].

The LM update rule is such that large steps are taken in the direction of low curvature to skip past plateaus quickly, and small steps taken in the direction of high curvature to slowly converge to minima.

The algorithm for the LM rule is as follows.

1. Do an update directed by rule (2).
2. Evaluate r(x) as the new parameter vector.
3. If r(x) has increased as a result of the update, then retract the step by resetting the weights to their previous values. Increase μ is by some significant factor, α. Then, redo step 1.
4. If r(x) has decreased as a result of the update, then accept the step by keeping the weights at their new values. Decrease μ by a factor of β, and repeat step 1.
5. Continue until any stopping condition is met.

D. The Scaled Conjugate Gradient Algorithm

SCG is a supervised learning algorithm which has been show to handle large-scale problems effectively [15]. Similar to the LM algorithm, it utilizes second order information (curvature) from the neural network, but has modest memory requirements due to inexpensive calculations of the gradient information [15].

The final SCG algorithm is detailed below. The interested reader is referred to [15] for the derivation of this algorithm.

1. Initialize the weight vector at the first iteration, x_1, and set the values of $\sigma > 0$, $\lambda_1 > 0$, and $\overline{\lambda_1} > 0$. Set the initial conjugate solution, p_1, and the steepest descent direction, r_1, equal to the error surface gradient, $p_1 = r_1 = -\nabla f(x_1)$.Set $success = true$.
2. If success=true, then calculate the curvature information, ∇^2 f(x_k):

$$\sigma_k = \frac{\sigma}{|p_k|} \quad (3)$$

$$\nabla^2 f(x_k) = \frac{\nabla f(x_k + \sigma_k p_k) - \nabla f(x_k)}{\sigma_k} \quad (4)$$

$$\delta_k = p_k^T \quad (5)$$

3. Scale $\nabla^2 f(x_k)$ and δ_k:

$$\nabla^2 f(x_k) = \nabla^2 f(x_k) + \left(\lambda_k - \overline{\lambda_k}\right) p_k \quad (6)$$

$$\delta_k = \delta_k + \left(\lambda_k - \overline{\lambda_k}\right) |p_k|^2 \quad (7)$$

4. If $\delta_k \leq 0$, make the Hessian matrix positive definite:

$$\nabla^2 f(x_k) = \nabla^2 f(x_k) + \left(\lambda_k - 2\frac{\delta_k}{|p_k|^2}\right) p_k \quad (8)$$

$$\overline{\lambda_k} = 2\left(\lambda_k - \frac{\delta_k}{|p_k|^2}\right) \quad (9)$$

$$\delta_k = -\delta_k + \lambda_k |p_k|^2, \lambda_k = \overline{\lambda_k} \quad (10)$$

5. Calculate the step size, μ_k:

$$\mu_k = p_k^T r_k, \alpha_k = \frac{\mu_k}{\delta_k} \quad (11)$$

6. Calculate the comparison parameter, Δk:

$$\Delta k = \frac{2\delta_k \left(f(x_k) - f(x_k + \alpha_k p_k)\right)}{\mu_k^2} \quad (12)$$

7. If $\Delta k \geq 0$, then a successful reduction in error can be made. Update the weight vectors, x_{k+1}, and the steepest descent direction, r_{k+1}:

$$x_{k+1} = x_k + \alpha_k p_k \quad (13)$$

$$r_{k+1} = -\nabla f(x_{k+1}) \quad (14)$$

$$\overline{\lambda_k} = 0,\ success = true \quad (15)$$

a. Check whether the direction is still acceptable with $k\ mod\ N = 0$. If acceptable:

$$p_{k+1} = r_{k+1} \quad (16)$$

b. Else, create a new conjugate direction:

$$\beta_k = \frac{|r_{k+1}|^2 - r_{k+1} r_k}{\mu_k} \quad (17)$$

$$p_{k+1} = r_{k+1} + \beta_k p_k \quad (18)$$

8. If $\Delta k \geq 0.75$ then reduce the scale parameter $\lambda_k = \frac{1}{2}\lambda_k$,
9. Else a reduction in the error is not possible:

$$\overline{\lambda_k} = \lambda_k, \text{ success = false} \quad (19)$$

10. If $\Delta k < 0.25$,then increase the scale parameter $\lambda_k = 4\lambda_k$

11. If the steepest descent direction $r_k \neq 0$, then set $k = k + 1$ and go to step 2. Else, terminate optimization and return x_{k+1} as the desired weights.

IV. METHODOLOGY

A. Dataset Description

The dataset was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The complete dataset contained of 351 instances. The targets were free electrons in the ionosphere. The output of simulated from MATLAB is of binary with either is good or bad as the result."Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not shown any changes their signals pass through the ionosphere.

Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this dataset are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal.

The first stage for this project is load data and preprocess from MATLAB. Preprocess was done to between parameters between -1 and 1. After preprocessing the data was separated into 3 parts. There are testing, validations, and training set divided by 60:20:20 ratios.

Training was stop when the performance test is max iterations reached, error criterion or over-fitting had occurred. If process is false, the process will be continue and rotate until finished.

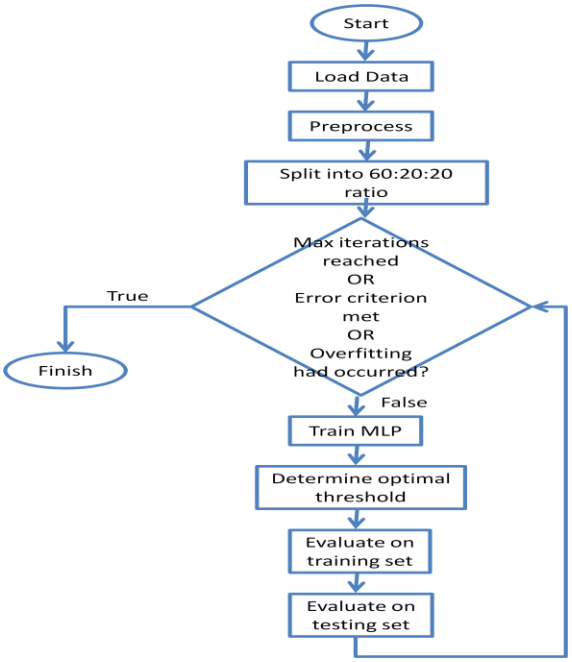

Fig 1 Flowchart of ionosphere database classification using MLPs with LM and SCG algorithms

B. MLP Structure and Parameters

For the MLP classifier presented in this paper, a fully-connected MLP structure was used. The MLP structure is shown in Fig. 2. Several different MLP structures to determine the optimal structure were evaluated for the given problem. The evaluated structures are shown in Table 1.

TABLE I
MLP STRUCTURES TESTED

Network	MLP Structure
1	15
2	20
3	25
4	30
5	35
6	40

The transfer function is responsible to transform the inputs entering the unit into output(s). This output, in turn, serve as inputs to other units, or to an outside connection, as dictated by the structure of the MLP [7]. Because the problem presented here is a pattern classification problem, the tangent-sigmoid (TANSIG) transfer function was used in the hidden and output layers. The NW method was used to initialize the MLP weights, while ES was used to prevent the MLP from over-generalizing.

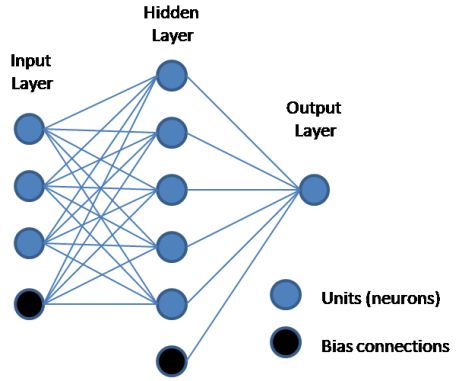

Fig. 2: Three-layer MLP structure. Hidden units were varied from one to 40. Input units not fully depicted due to spatial constraints.

C. *Random Number Generation*

MLP convergence depends on the initial value of the weights prior to training. To test the effectiveness of the proposed MLP structures, each training run performed on each structure was repeated four times with different random NW initialization values. The initialization values were generated using a pseudo-random number generator called the Mersenne-Twister algorithm (MTA) [16, 17].

In MTA, the sequence of random numbers generated is determined by the internal state of the random number generator. Setting the generator to different states leads to unique computations and outcomes for each state. The unique computations result in the generation of unique series of random numbers based on the state. To ensure repeatability of the experiments, the generator state is set to some fixed value each time the optimization executes to ensure that the same set of random numbers are generated. These random numbers were then assigned to the MLP weights, and training was performed. After training, the classification accuracy of the MLPs were averaged out and taken as the accuracy of that particular MLP structure. The random seeds tested are shown in Table 2.

TABLE II
RANDOM SEED VALUES FOR INITIALIZING MLPS

Seed	Value
1	0
2	50
3	100
4	150

D. *LM Algorithm Parameters*

The parameters shown in Table II and Table III were used for LM and SCG training algorithms, respectively.

Table III
LM PARAMETERS USED FOR TRAINING

Parameter	Value
Maximum epochs	1000
Training goal	0
Minimum $\nabla f(x_i)$	$1.00\text{x}10^{-10}$
μ	$1.00\text{x}10^{-3}$
α	0.10
β	10
Maximum μ	$1.00 \text{ x } 10^{10}$

For the LM algorithm, training stops if the number of iterations exceed the maximum epochs, if the performance function drops below the training goal, if the magnitude of the gradient is less than minimum $\nabla f(x_i)$ or if μ becomes larger than the maximum μ, the training is stopped.

V. RESULTS AND DISCUSSIONS

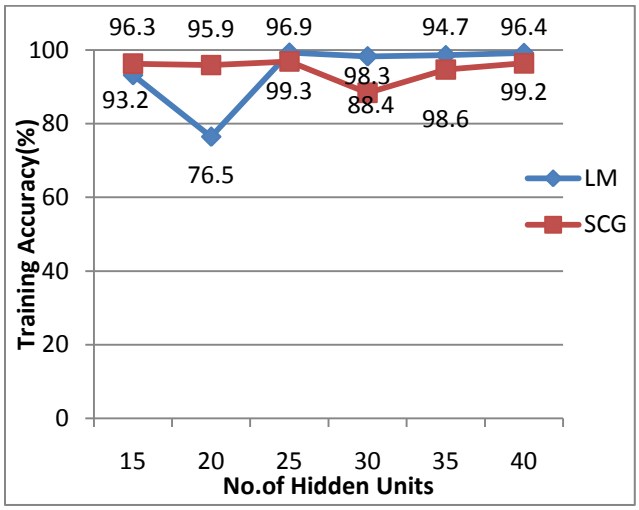

Fig.3: Average Training Accuracy versus Number of Hidden Units using LM and SCG training algorithms

In this section the simulation from MATLAB can be obtained using the LM and SCG algorithms in four conditions. Prior to training, the dataset rescaled to between -1 and 1 before being split into 60:20:20 (training: validation: testing) ratio. The average accuracy was obtained by averaging out the classification accuracy of the MLP with different initialization values. As can be seen, the accuracy of both algorithms was very high, as all tests showed above 75% accuracy. In some cases, the classification accuracy almost reached 100%. In the Fig.3 it can be seen that the number of hidden 25 units showed that the highest values for the both of algorithms.

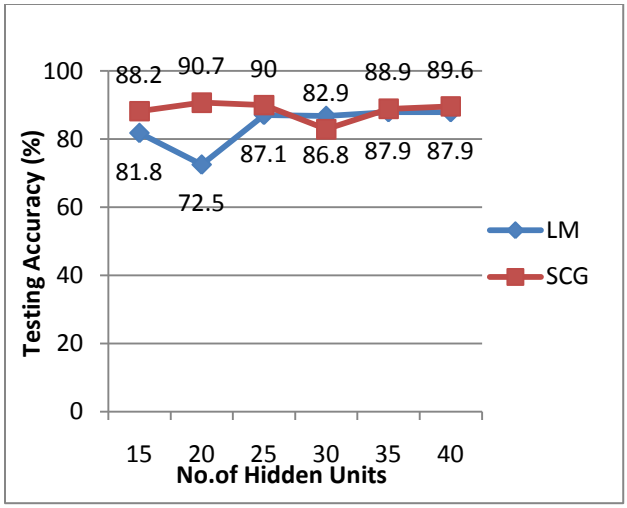

Fig.4: Average Testing Accuracy versus Number of Hidden Units using LM and SCG training algorithms

After the training process was completed the accuracy of the MLP classifiers were examined on the test set, in the shows. The results are shown in Fig. 4. The averaged classification accuracies were generally lower compared to the training set (ranging from 70% to 90% using LM, and 82% to 91% using SCG) as expected. This confirmed that for both training and testing phase, SCG algorithm performed higher than LM algorithms.

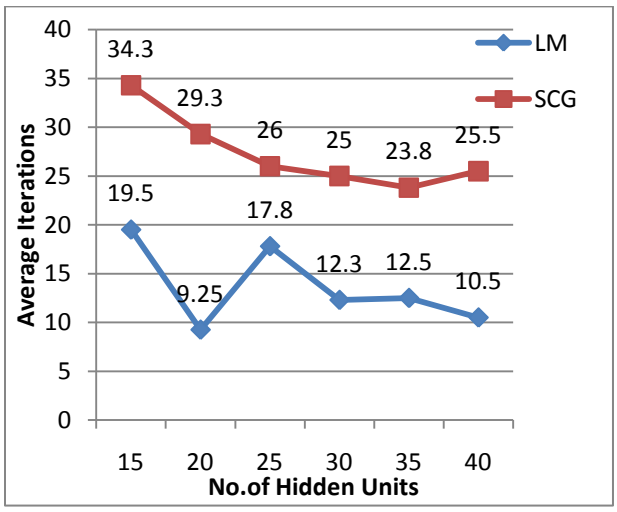

Fig.5: Average Iterations versus Number of Hidden Units using LM and SCG training algorithms

Fig. 5 show average training iterations versus number of hidden units by using both algorithms. During testing and training accuracy was stopped when ES has detected that over fitting has occurred. In the figure above, the LM algorithms better than SCG algorithms because average iterations has lower, but based on accuracy the SCG algorithms is priority.

The average MSE versus number of hidden units are shown in Fig. 6. A smaller MSE value indicates that the residuals are small, meaning that the particular MLP had fitted the data well. As can be seen in Fig.6 at hidden layer sizes 25 is the best value in the systems which showed very good accuracy, and economy of network size. At this point showed small value of MSE (0.106) while for SCG is 0.010. For SCG result the value had reached almost zero.

In this chart the figure showed that at the number of hidden units 25 is the minimum average MSE in the simulated. This is because during the training and testing perform the fast training speed.

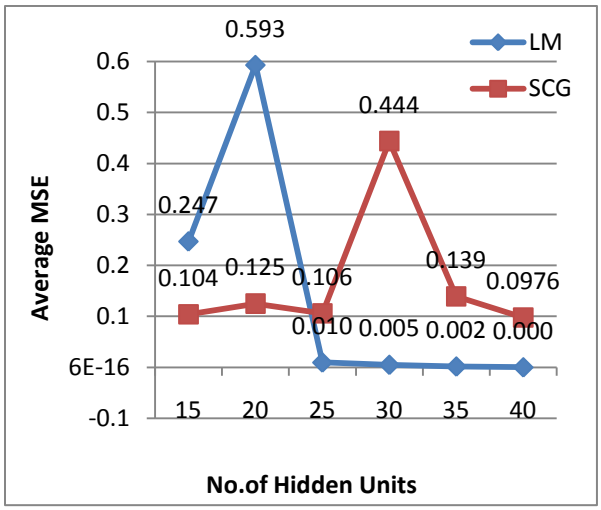

Fig.6: Average MSE versus Number of Hidden Units using LM and SCG training algorithms

VI. CONCLUSIONS

As a conclusion, comparison between two popular MLP training algorithms (LM and SCG) for classifier ionosphere data presented. Based on the results, it can be concluded that both algorithms were comparable in terms of accuracy and speed. However, the SCG algorithm has shown better advantage in terms of accuracy (as evidence in the average training accuracy and MSE) and speed (as evidence in the average training iterations) on the best MLP structure (with 25 hidden units).

VII. FUTURE WORKS

There are several other intelligent classifiers that can be used to replace the MLP such as Support Vector Machine (SVM) and adaptive neuro -fuzzy inference system (ANFIS). Also, the parameters of the Levenberg-Marquardt (LM) can be adjusted better convergence.

VIII. REFERENCES

[1] W. H. Wolberg, *et al.*, "UCI Machine Learning Repository [online], University of Wisconsin, WI,"*Available*,[*http://archive.ics.uci.edu/ml/datasets/Ionosphere*], 1989.

[2] Dr.S V Dudul, "*Classifications of Radar Returns From The Ionosphere Using RBF Neural Network*," July 2007

[3] Suresh S.Salankar,Dr.Balasaheb M.Patre, "*RBF Neural Network Based Model as an Optimal Classifier for the Classification of Radar Returns from the Ionosphere*"

[4] Sigillito, V. G., Wing, S. P., Hutton, L. V., & Baker, K. B. (1989), "*Classification of radar returns from the ionosphere using neural Networks"* Johns Hopkins APL Technical Digest, 10, 262-266.

[5] Mrityunjay Gautam,"*Evolution of Architecture and Weights of ANN using Variable Length GA*"2005

[6] V. B. Rao, *C++ neural networks and fuzzy logic*: MTBooks, IDG Books Worldwide, Inc., 1995.

[7] D. Anderson and G. McNeill, "*Artificial neural network technology*," Rome Laboratory, New York1992.

[8] J. Tebelskis, "Speech recognition using neural networks," PhD, Carnegie Mellon University, Pittsburgh, Pennysylvania, 1995.

[9] I. M. Yassin, "*Face detection using artificial neural network trained on compact features and optimized using particle swarm optimization*," M. S. thesis M. S. Thesis, Faculty of Electrical Engineering, Universiti Teknologi MARA, Shah Alam, 2008.

[10] D. Nguyen and B. Widrow, "*Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights*," in Proc. International Joint Conference on Neural Networks, 1990, pp. 21-26.

[11] L. Pretchelt, "*Early stopping - but when?*," Neural Networks: Trick of the Trade, vol. 1524, pp. 55-69, 1996.

[12] A. Ranganathan, "*The Levenberg-Marquardt Algorithm*," 2004.

[13] M. T. Hagan and M. B. Menhaj, "*Training feedforward networks with the Marquardt algorithm*," *IEEE Trans. on Neural Networks,* vol. 5, pp. 989 - 993, 1994.

[14] H. Demuth and M. Beale. (2005) MATLAB Neural Network Toolbox v4 User's Guide. *Mathworks Inc.*

[15] M. F. Møller, "*A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning*," *Neural Networks,* vol. 6(4), pp. 525–533, 1993.

[16] M. Matsumoto and T. Nishimura, "*Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudorandom Number Generator*," *ACM Transactions on Modeling and Computer Simulation,* vol. 8(1):3, pp. 3-30, 1998.

[17] *MATLAB Function Reference Documentation*. Mathworks Inc., Natick, MA, 2008.