279

Software Toolkit for Designing an Artificial Neural Network

M. A. Ahmad[1], J Mohamad-Saleh[2]

[1]Electrical & Electronics Engineering School, Engineering Campus, Universiti Sains Malaysia, Seri Ampangan, 14300 Nibong Tebal , Pulau Pinang, Malaysia.
[1]Email: done2k1@hotmail.com, Tel.: 012-534 5055
[2]Email: jms@eng.usm.my, Tel.: 04-593 7788 ext 6027

ABSTRACT

Basically, there are two kinds of artificial neural network (ANN), which can be classified into supervised and unsupervised. Commonly, supervised neural networks are trained or weights adjusted, so that a particular input leads to a specific target output. Generally, the supervised training methods are commonly used in solving most problems. An ANN can be designed, trained, validated and tested by means of the Neural Network Toolbox in Matlab. Thus far, there exists an interface to run these processes in the Matlab Neural Network toolbox. The interface is called *nntool*. Unfortunately *nntool* is not user-friendly. Moreover it lacks information and normally gives help commands that are too simple. The difficult to follow and understand procedures for developing an artificial neural network model using *nntool* make it a rather complicated software. Therefore, it is very difficult for beginner users to use *nntool* for developing an ANN. For that reason, *AnnSol* (Artificial Neural Network Simulator), a newly designed toolkit, has been developed in this project.

Keywords: Neural Network, Computer Simulation, Matlab nntool.

1. INTRODUCTION

Artificial Neural Network sometimes referred as neurocomputers, connectionist network or parallel distributed processors (Haykin, 1994). It was first proposed several decades ago, but the interest in its use is relatively new (Lippmann,1987). It is different from all other kinds of existing systems because of its ability to "learn" or gain "knowledge" based on a set of examples given to it. The basic idea behind neural network is to allow machine to copy the way the human brain learns, that is through experience or examples. Neural networks are composed of computational elements operating in parallel and the arrangements are reminiscent of the biological neuron of the brain. The computational elements or neurons are connected via links, whose weights that are adapted during the learning phase of a neural network model, another similarity to the biological neurons. Hence, the name of Artificial Neural Network.

ANN are generally used when most step-by-step programming fail to solve a complicated problem. Thus far, ANN has been used in many fields ranging from engineering to environmental to medicine. For example, ANN has been used to classify objects (Pope, 1994), make predictions (Simon et al., 2000; Richard et al., 2003; Billings et al., 1991; Gomm et al., 1997), recognize pattern (Karsten, 2001; Büker and Hartmann, 1996) and estimate parameter values (Mohamad-Salleh and Hoyle, 2003).

It is normal that a simulation or modeling of an ANN is first carried out to investigate its feasibility at solving a certain problem. Once the simulation is successful, then only a dedicated ANN microprocessor system is designed. One of an neural network simulation tool is called *nntool* (Demuth,2002)

Although the *nntool* Toolbox can be used to design, train, validate and test a neural network in Matlab, the interface has not been found to be user-friendly. In *nntool,* for example, there are help sections for users to get information on how to use the interface but the help

does not contribute much to user. This is because all these enlightenment are given too simple explanation, that it becomes less informative for new users to understand its usage.

Basically, the processes of an ANN using *nntool* are not efficient for some reasons. First, the users who are not familiar with Matlab they will face a lot of difficulties in designing, testing, validate and training processes due to little proficiency in using the Matlab programming language. Most probably they will find difficulties in keeping the data, choosing the desired network models and other processes, which is involved when designing neural network because the usage of *nntool* interface is quite intricate. This may force a user to use trial and error methods during these processes, causing wasted time.

Second, in *nntool* users can look at the processes or activities that have been executed. Therefore, they have to repeat the same processes all over again for the next ANN design. If the processes are too long or too complicated, the user may forget what have been done and this could lead to failure, and waste of the user's time. Moreover, in *nntool*, all network commands or functions are placed in a small, convoluted and ugly interface.

Third, if the user wants to start with a new designing process using *nntool*, he has to delete all previous data or close Matlab and reopen it. At this stage, it shows that *nntool* is not sophisticated. Moreover, having to do this actually wastes a user's time.

Fourth, users are not allowed to set percentage values for testing, training and validation data sizes while designing an ANN using *nntool*. Hence, they have to do it by giving a command in the Matlab command window. For those who are new with Matlab, they may face difficulties in doing this.

Discussed above are the weaknesses and disadvantages of using *nntool*. For those users who are not familiar in using Matlab they will meet with all these difficulties. These drawbacks motivate the development of this project, whose objective is to design a toolkit, which is far more uer-friendly and easy to use by any user, being at any level, compared to *nntool*. This new toolkit is called *AnnSol*.

2. METHODOLOGY AND APPROACH

This paper focuses on supervised networks i.e. Perceptron, Multi-layer Perceptron (MLP), Learning Vector Quantization (LVQ) and Hopfield. Tools such as Matlab, GUIDE and Neural Network Toolboxes are used in carrying out this project. The reason for using Matlab as the main connector to the new interface is its common use in many simulations or modeling.

There are two approaches to building this interface. One way is by using programming methods and the other is by using GUIDE. GUIDE has been used to create this interface because plenty of time can be saved using this method. Besides using GUIDE in Matlab, Visual Basic or C++ Builder can also be used to build the interface. However, it will make the task difficult particularly in creating functions for designing ANN models. Matlab already has built-in functions to design ANN, within the Neural Network Toolbox. Therefore it is a lot easier to use GUIDE to develop this interface as the necessary function can only be called from the interface programs instead of having to create new function.

GUIDE, the Matlab graphical user interface development environment, provides a set of tools for creating GUIs. These tools greatly simplify the process of laying out and programming a GUI. When GUI in GUIDE is opened, it will display the Layout Editor, which is the control panel for all the GUIDE tools. The Layout Editor enables users to layout a GUI quickly and easily by dragging components, such as push button, pop-menus, static text, checkbox, fram or axes from the component palette into the layout area. GUI tools used for this paper are:

1. Axes.
   Used to display figures, images or flatten the output result for this ANN in graphical methods such as graph
2. Frame.
   This component is used to decorate the interface windows
3. Edit Text.
   This function is used to allow users to enter data or values for creating an ANN. Command is use in this callback section, in doing so, an error message diolog box

may pop up notify users about wrong values which have been entered. For example, if an edit text requires a numeric value but a user enters an alphabet, an error message will be displayed.

4. Checkbox.
   A checkbox allows users to choose preferred parameters. For this paper the choices are epoch, show, goal, hard limiter and iteration stop, which maybe needed in a training process.
5. Static text.
   This function is used to display filenames when a user would like to load a file such as a data file. This function also displays the number of data in a loaded file, the processing time for training, testing and validation processes. In this paper, static text function is also used to display MSE values for testing and validation process as when all processes involved in an ANN development are completed.
6. Pushbutton.
   A pushbutton is clickable. For the designed interface, a pushbutton is clicked when users would like to begin a training or adaptation process, saving, open, reset, loading, editing data or to get help on using the interface.

In *nntool*, if a user requires variable to be displayed in the Matlab workspace browser, he needs to export the variables in the export section. In contrary, *AnnSol* will automatically display the necessary variables once the users start a training process by clicking on the [Train] button. The "Evalin" and "assignin" Matlab functions are used to achieve this. The "Assignin" function is used to export any variable values from the GUI to the workspace while "evalin" is used to import any variable values from the workspace back to the GUI.

3. RESULTS

Below are 6 figures, which show the results of this project. Figure 1 shows the main window of *AnnSol*. Users can click on the pushbutton to select a desired network i.e. Multi-layer Perceptron (MLP), Learning Vector Quantization (LVQ), Perceptron and Hopfield. Figure 2 shows the MLP interface window. Figure 3 shows the LVQ interface window and Figure 4 shows the Perceptron interface window. A user is able to enter data or values, or select preferred parameters from each of the network's interface window.

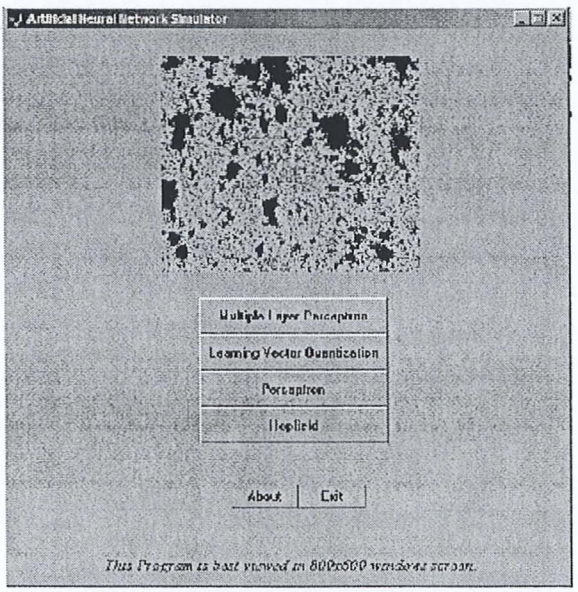

Figure 1: Artificial Neural Network Simulator interface

Figure 5 shows the two-dimensional Hopfield network interface and finally, Figure 6 shows the three-dimensional Hopfield network interface.

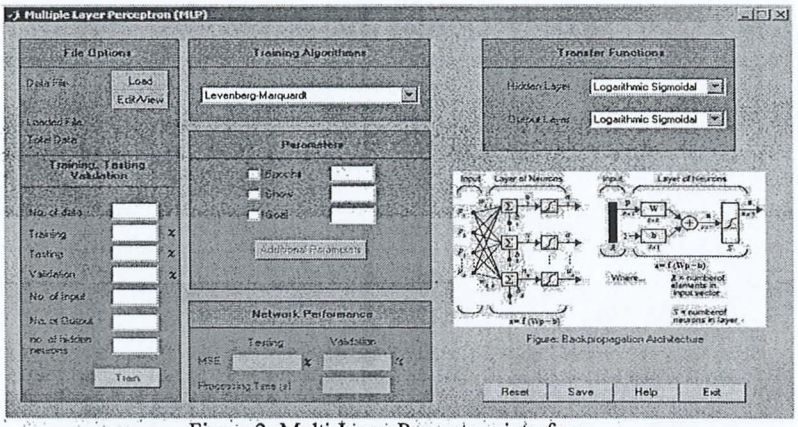

Figure 2: Multi-Layer Perceptron interface

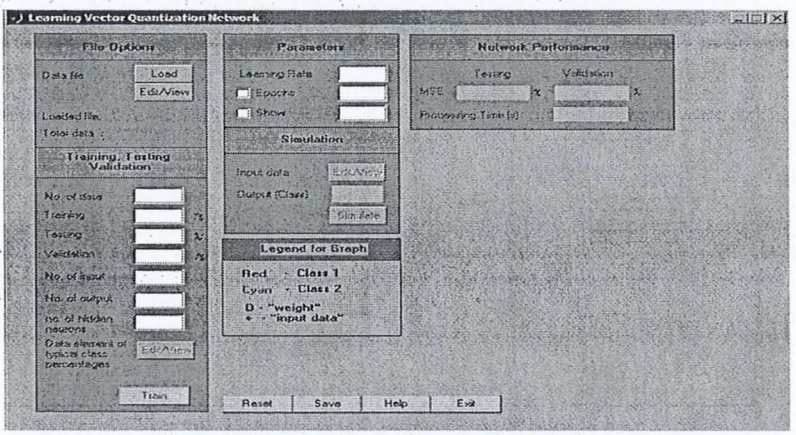

Figure 3: Learning Vector Quantization interface

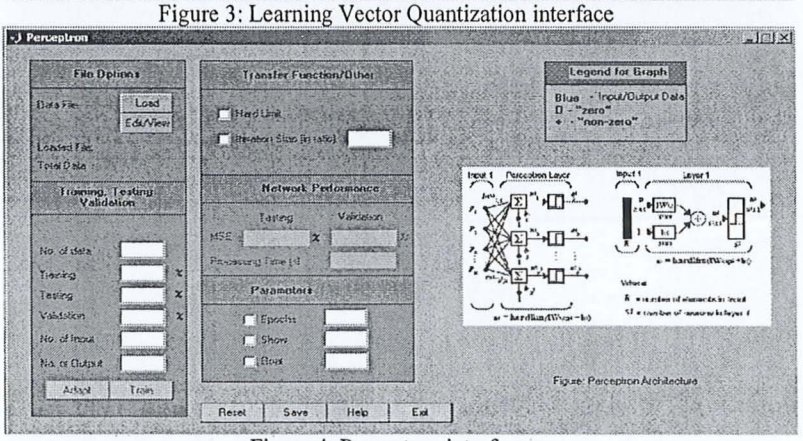

Figure 4: Perceptron interface

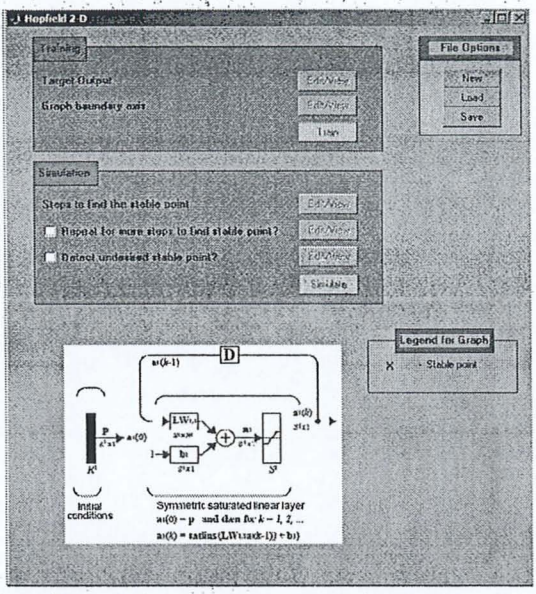

Figure 5: 2-dimensional Hopfield interface

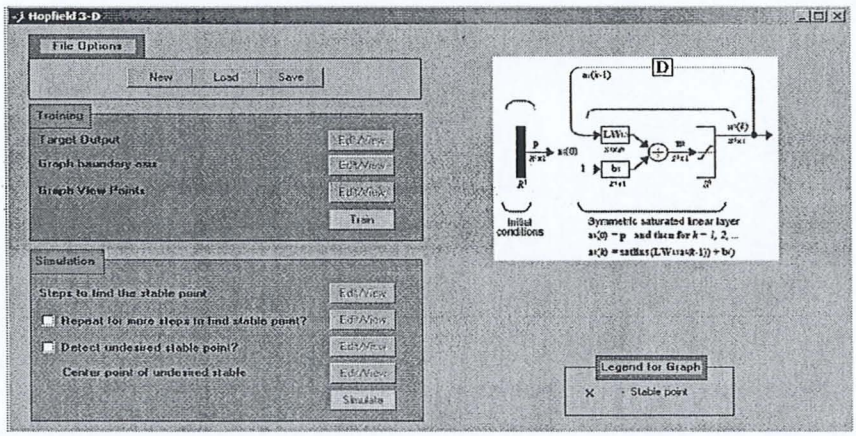

Figure 6: 3-dimensional Hopfield interface

4. DISCUSSION

The *AnnSol* interface is far more user-friendly compared to *nntool*. As has been discussed in section 2, there are many weaknesses in *nntool* and all the weaknesses have been overcome by*AnnSol* . The advantages offered by *AnnSol* are:

1. With *AnnSol*, the MSE percentage for testing and validation will be shown automatically after the training, testing and validation processes are completed. This automatically not supported in *nntool*.
2. The Parameters to create a backpropagation network are very limited in *nntoo*. In *AnnSol* however, all the necessary parameters are provided and can be changed as necessary by users.

3. *nntool* only provides two kinds of stopping conditions for a training process i.e. error goal and epochs. In *AnnSol* however, three kinds of stopping condition are provided. There are the validation state, error goal and epoch. In default, *AnnSol* use the validation stop condition, which this is the best stopping condition in developing a robust ANN (Bishop, 1994).

5. CONCLUSION

A new neural network simulation toolkit called the Artificial Neural Network Simulator (*AnnSol*) has been successfully developed in this project. This interface can be used hand-in-hand with Matlab to design an artificial neural network. It is capable of accomplishing processes that *nntool* can i.e. testing, training and validation whilst providing some extra functions to further facilitate the tasks.

Overall, *AnnSol* provides the convenience of creating four artificial neural network models i.e. Multi-Layer Perceptron, Learning Vector Quantization, Perceptron and Hopfield. The task of designing, training and testing a neural network is made simple by using this interface.

REFERENCES

Billings S. A., Jamaluddin H. B. and Chen S. (1991), "A Comparison of The Backpropagation and Recursive Prediction Error Algorithms for Training Neural Networks", Mechanical Systems and Signal Processing, vol. 5, no. 3, 233-255.

Bishop C.M. (1994), "Neural Networks and Their Application", Review of Scientific Instruments 65(6), 1803-1832.

Büker, U. and Hartmann, G. (1996). "Knowledge-Based View Control of Neural 3D Object Recognition System, 24-29, In Proceeding of International Conference on Pattern Recognition.

Demuth H., and Beale M., (2002), "Neural Network Toolbox for Use With Matlab ®, The MathWorks.

Gomm J. B., Williams D., Evans J. T. and Doherty S. K. (1997), "Neural Network Applications in Process Modelling and Predictive Control", Transactions of the Institute of Measurement and Control, vol. 19, No. 4, 175-184.

Haykin S. (1994), "Neural Networks - A Comprehensive Foundation", McMillan Publishing Com. Ltd.

Karsten S. (2001), "Automatic Pattern Recognition in ECG time series", Institute für Strahlen- und Kernphsik, University of Bonn, Nussalle 14-16, D-53115, Germany.

Lippmann R. P. (1987), "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, 4-22.

Mohamad-Salleh J. and Hoyle B.S (2002), "Determination of Multi-Component Flow Process Parameters Based On Electrical Capacitance Tomography Data Using Artificial Neural Networks", Measurement Science and Technology, 13(2), 1815-1821.

Pope, A. R. (1994), "Model Based Object Recognition: A Survey of Recent Research. Technical Report TR-94-04. University of British Columbia.

Simon A. (2000), "Using Artificial Neural Network methods to predict forest characteristics in southeast Alaska", 4th International Conference On Integrating GIS and Environmental Modeling (GIS/EM4: Problem, Prospects and Research Needs. Banff, Alberta, Canada.

Richard L. (2003), "Prediction of traumatic wound infection with a neural network derived decision model", Department of Emergency Medicine, Michigan State University/Kalamazoo Centre for Medical Studies, Kalamazoo, MI