DATA CLASSIFICATION WITH AN IMPROVED WEIGHTLESS NEURAL NETWORK

Weng Kin Lai[a] and George Coghill[b]

[a]MIMOS Bhd., Technology Park Malaysia, email: *lai@mimos.my*
[b]Department of Electrical & Computer Engineering, University of Auckland, New Zealand
email: *G.Coghill@auckland.ac.nz*

Abstract - **This paper examines the performance of an enhanced weightless neural network as a classifier. Like all earlier weightless neural network models, this network learns in one pass through the data. This new weightless neural network has shown significant gains in the classification accuracy over the earlier Deterministic RAM Network (DARN), on a variety of problems. In addition, some comparisons between the DARN and the proposed network are presented. This will also include some evidence on how a standard Multilayer Perceptron network would behave on the same data sets. Finally, hardware implementation issues are discussed.**

Keywords: - **Data classification, Weightless neural networks.**

1. INTRODUCTION

Classification is the process of labelling specific sets of input data. The classifier may choose a variety of ways to realize this goal. Whatever the method, classification is an important component for many problem-solving tasks. Traditionally, such classification problems have been handled by statistical techniques, although newer novel methods based on swarm intelligence have also been investigated in recent times (Deneubourg *et al.*, 1990). A contemporary study (Hoe *et al.*, 2002) categorised a set of documents involving 84 web pages from four different categories: *Business*, *Computer*, *Health* and *Science*. These documents were randomly retrieved from the *Google* web directory within the corresponding categories and processed by the text classifier.

There are two distinct classification techniques commonly in use – supervised and unsupervised (Richard *et al.*, 2001). In the case of unsupervised classification approaches, the algorithm clusters patterns without using any input-output data pairs as a reference. On the other hand, supervised classification, as the name suggests, begins with a training set of preclassified example transactions. E.g. For fraud detection applications, this would include complete records of both fraudulent and valid activities, determined on a record-by-record basis. The classifier-training algorithm then uses these pre-classified examples to determine the set of parameters required for proper discrimination, by encoding these parameters into the classifier model.

However, the focus of this paper lies with a class of networks known as *Weightless Neural Networks* (WNN). One advantage of this approach is that training only involves a single pass through the data. Another inviting prospect is that these networks may easily be implemented in digital hardware, where they will be able to give very high performance in terms of operating speed. This paper presents a new model, which has shown promising generalization properties.

The remainder of this paper is organised as follows. In the next section, the concept of using Artificial Neural Networks (ANN's) (Lippmann, 1989) for classification is introduced. In Section 3, a brief description of the evolution of the weightless neural network is introduced. Section 4 then describes the architecture and learning rules for the new weightless neural network model. Section 5 provides the major design issues concerning the new network as well as the experimental results. Finally, some conclusions are presented in Section 6.

2. ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANNs) or neural networks (NNs) as they are commonly known, are essentially parallel, distributed processing structures that consist of simple processing elements, usually interconnected via unidirectional signal channels (Robert, 1989). Furthermore, they can be taught by a process of learning from task examples, to store experiential knowledge and making it available for use later (Aleksander & Morton, 1990).

Today, there are many different neural network architectures, ranging from the simple single-layer, feed-forward network to the more sophisticated multi-layer networks with feedback (Robert, 1994). The adaptability of many of these networks is based upon the McCulloch and Pitts (MCP) model that seeks to represent the responses of the simple processing elements (neurons) by variable weights in their interconnections (Aleksander & Morton, 1990). This has been illustrated in Figure 1.

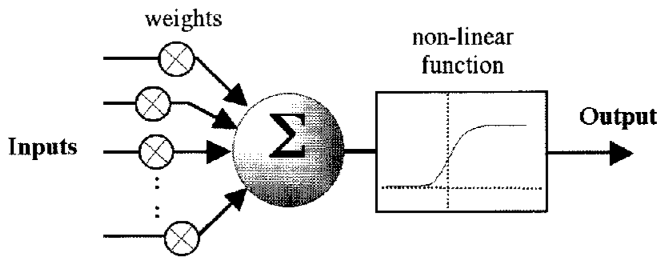


Figure 1: An example of a McCulloch and Pitts neuron

These weights may be adjusted through an appropriate learning rule. Even though neural networks were inspired by studies in neuroscience, they have now been utilised for their useful computation properties in solving a large range of problems.

Data classification has often been tackled using ANN's - predominantly with the multi-layer perceptron network (MLP) that is trained using the backpropagation learning rule (Rumelhart *et al.*, 1986). Figure 2 shows a multi-layer perceptron network with six input nodes, three hidden nodes and one output node.

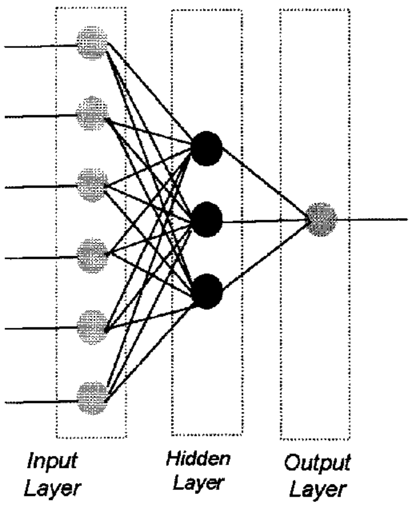


Figure 2: An example of a multi-layer perceptron neural network

The weights between the nodes in the various layers are modified by this learning rule. The MLP can deal with nonlinear classification problems because it is able to form complex decision regions (rather than just hyperplanes). Each node in the first hidden layer can create a hyperplane, while the nodes in the second hidden layer (the example shown here has only one hidden layer) combine these hyperplanes to create convex decision regions. In the third, or output layer, the nodes can combine these convex regions to form additional regions. Thus, in theory, it is possible to form any arbitrary region with two hidden layers and sufficient hidden units. However, supervised approaches such as backpropagation are usually considered too slow for many problems where the input dimensionality may be very high and the data set enormous.

A major disadvantage of such neural networks is that they are difficult to implement and miniaturise into an integrated circuit. It does not help too when there is a need for the storage of a large number of analogue weights as well as multipliers to process the incoming input signals. Moreover, such neural networks are commonly characterized by high connectivities, which would ultimately consume large areas of silicon when implemented as an integrated circuit. Consequently, this has motivated the development of random access memory (RAM) based neurons as a possible alternative.

3. WEIGHTLESS NEURAL NETWORKS

A weightless neural network is a type of neural network whose uniqueness lies in its lack of interconnection weights. The behaviour of the processing elements is not altered by changing the interconnection weights, but rather, by modifying the contents of memory units (Aleksander & Morton, 1990). Instead of the MCP node, such weightless neural networks use a conventional random access memory (RAM) as part of the processing node. Weightless Neural Networks (WNN's) have been around in various configurations since the 1960's. Ludermir *et al.* (1999) gives an interesting and comprehensive review of the topic.

The first useful weightless neural network is probably the WISARD, which stands for **WI**lkie, **S**tonham and **A**leksander's **R**ecognition **D**evice, - an adaptive pattern recognition machine that was completed in 1981(Aleksander & Morton, 1990; Ludermir *et al.*, 1999; Aleksander, 1984). This system was based upon the notion of *Discriminator Network.* The core of its operation is as follows. Consider a typical set of discrete binary data, which is split into training and test partitions. The training data is stored in the system's memory. Each test sample is then compared with every training sample and the class of the most similar training sample, using the Hamming distance measure, is taken to be the class of the test value. By collecting the results of all the test samples, a generalization assessment may be made.

Since WNN is using discrete binary data, it achieves the best possible level of generalization. Unfortunately, it uses large amounts of memory, in the form of RAM to store the training data. It also bases its generalisation measure upon all elements of the training sample, when the user may only wish to detect the presence of one feature within the sample. In addition, its implementation in hardware is problem dependent and it is not very suitable for on-line learning.

It may be seen that instead of adjusting weights, learning in weightless neural networks generally involves changing the contents of look-up tables. The generalized RAM based neuron model is shown in Figure 3 below. Notice that the neuron has a fixed number of inputs. Any input signals coming into each neuron causes one or more of the memory locations (L) to be addressed. The output of the neuron OP, is determined by the values contained in the addressed memory locations. The operation of such a neuron model is controlled by the **MODE** input.

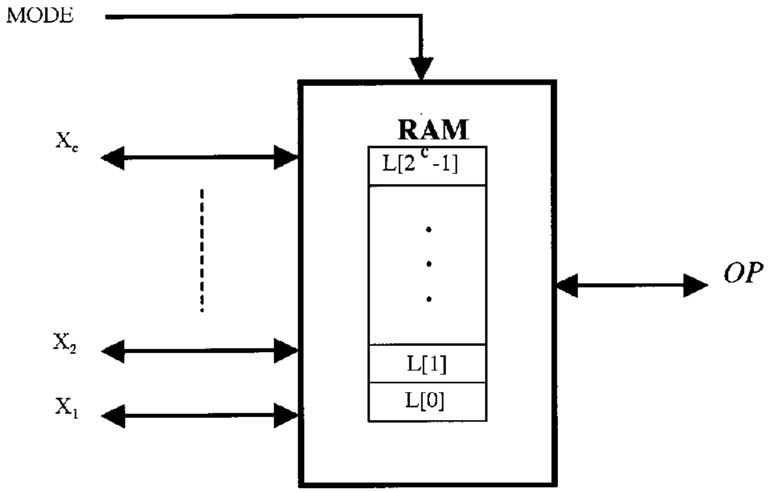
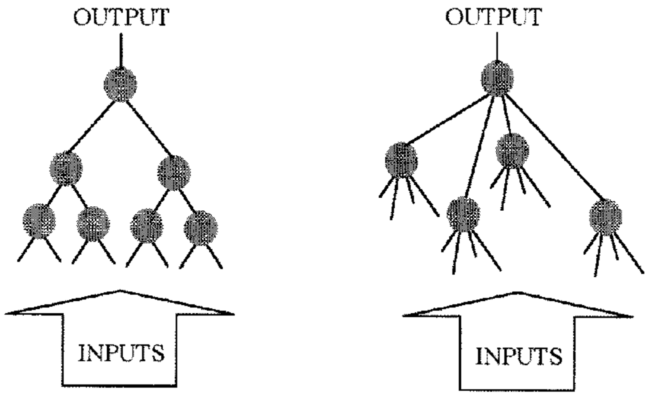


Figure 3: A generic RAM based neuron

The usual architecture for connecting RAM based neurons into a network is the pyramid. Figure 4 shows a pyramid structure for connectivities of two and four. The network is arranged into a number of layers. The top layer contains a single neuron. The output of a neuron is only connected to one input of a neuron on the next layer. Therefore the ith layer from the top of a pyramid network contains c^{i-1} neurons, and an N layer network will have c^N inputs and a single output.



(a) RAM neurons with connectivity = 2

(b) RAM Neurons with connectivity = 4

Figure 4: Pyramid Architecture of Weightless Neural Nets

However, many problems require more than one output from the network, and so the multi-pyramid architecture, as shown in Figure 5, is more commonly adopted. Several pyramids are thus used with their inputs connected in parallel. Therefore, the number of inputs to the entire network is equal to the number of inputs of a single pyramid, and the number of outputs is equal to the number of pyramids. In this example, there are five pyramids, feeding into five different outputs.

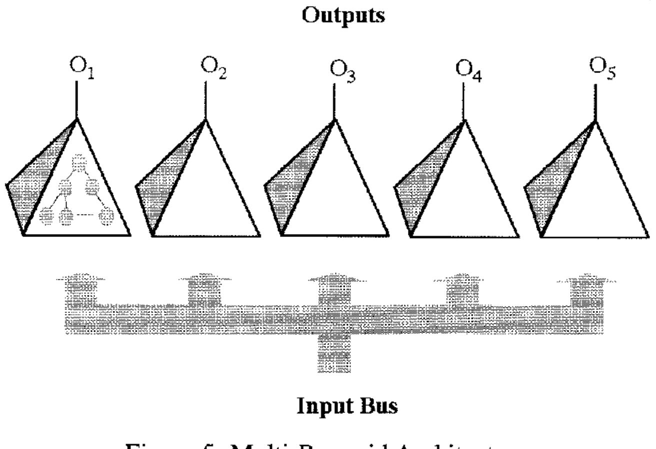


Figure 5: Multi-Pyramid Architecture

The weightless neural network described in this paper is derived from the Probabilistic Logic Neuron (PLN), a RAM based neural model that was proposed by Kan and Aleksander (1987). This is shown in Figure 6.

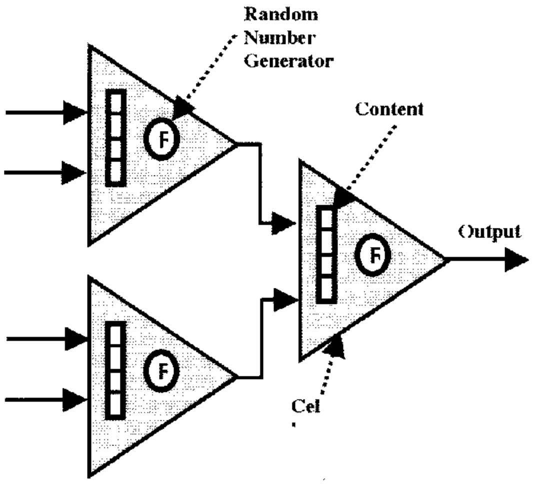


Figure 6: A weightless neural network based on the Probabilistic Logic Neuron

The PLN is a very simple model, consisting of little more than the RAM element itself. The RAM is addressed directly by the input lines, so that only one RAM location is addressed at any time. The *mode* input sets the RAM to either read or write mode. The RAM locations are each allowed to contain one of three values: True (**1**), False (**0**), and undefined (**U**). Initially (before training) all RAM locations are set to the undefined (**U**) value. When the neuron is in the read mode, it presents the value of the RAM location addressed by the inputs to the output. If the addressed location contains an undefined value, a random decision is made between a **0** or a **1**.

The network is trained to recognize a new pattern by repeatedly presenting the new pattern (with the neurons in read mode) until the pyramid outputs the desired output response. The neurons are then put into the write mode, so that the neuron output values which mapped the input pattern to the desired output are stored in each neuron's RAM.

To further illustrate the operation of the PLN, a simple example based on the architecture shown in Figure 6 above (in a realistic application, there may be many input layer cells with 4, or even 8 bit input address lines and several pyramids) was chosen. During the training phase, the input vector, which (in this case) is a 4 bit binary number, acts like an address generator. If the input pattern was, say, **0000** the first (top) *contents* location would be selected for both the input *cells*. When an undefined location is addressed like this, the random number generator is called. This outputs, from the cell, either a **1** or a **0** with equal probability. The outputs from the first layer cells become the address lines to the next layer. Again, at the start of the training phase, the second layer output cell's input vector will select a content value, which is undefined. Once more, a random output will be generated, resulting in the output of a **1**, or a **0**.

As this is a supervised learning network, there will be a desired output value available during the training phase. This is compared with the randomly generated value and if they are the same, a signal is sent back to each cell in the network. The effect of this signal is to cause the randomly generated values to be stored in the appropriately addressed locations. If the actual output disagrees with the desired value, a different signal is propagated back informing the cells to try again. The network will keep generating random outputs until the desired output is achieved, or the network fails to deliver after a set number of attempts. This procedure is repeated with each training pattern until the network is trained. Unfortunately, when the network is only partially trained, it often fails to deliver the desired output because some of the addressed locations have already been defined previously to a different value from that required by the latest training sample, - a phenomenon commonly known as *saturation*. Another problem with this network is that, after training, when a previously unseen sample was presented and if this addresses a location which has been missed during the training phase, the output will be random. Therefore, these networks have practically no generalization properties. It has been found that training them with random noise helps to improve generalization, but not well enough to make them very useful.

Filho *et al.* (1991) proposed a new model, which does not use random number generators. Instead, when an undefined location is selected by the input address to a cell, the output generates an undefined value and this is propagated forward to the next layer. For example, suppose the output from the first layer is the vector **1U**. This is interpreted by the output cell as having selected both addressed locations **10** and **11**, forming an *addressable set*, or {**10**, **11**}. The *Goal Seeking Neural Network* then has some rules on how to modify the contents, or perform recall.

The idea of having an addressable set together with some rules was developed further to produce the *Deterministic Adaptive RAM Network* (DARN) (Bowmaker & Coghill, 1992). The major innovation in this earlier version of the DARN was to make the desired output of every cell in the pyramid, the same as the desired output of the entire pyramid. Although the generalization performance improved over that of the Goal Seeking Neuron, the DARN's capabilities were still poorer than those of the MLP. Here a further enhancement to the DARN that has produced promising results is presented. The training and recall procedures are described in the following section.

4 DARN OPERATION

In the new DARN, each addressed content is now a signed register instead of a tri-state value. Initially, all of the registers in the network's cells are set to zero. The zero is interpreted as an undefined value. During learning, as will be seen later, the counter register may become positive or negative. The learning and recall behaviour is explained below.

4.1 Learning

The input address vector is presented to the network. The desired output of every cell in the pyramid is the same as the desired output of the pyramid. Beginning at the input layer, if the desired output is **1**, the addressed location is incremented by one. On the other hand, if the desired output is **0**, the location is then decremented by one. The method then involves calculating the address vectors for the next layer, moving towards the output. The input address vector to each cell in the next layer is constructed by invoking the *Recall* function (described below) on the previous layer. This address vector is then applied to the cell inputs on the next layer, and the contents of each cell are again modified as described above. This procedure is continued until the pyramid output is reached. The same method is applied to each pyramid. The entire training pattern set should be presented just once. The order in which the samples are presented is unimportant.

4.2 Recall

During the recall phase, again, the content of each addressed location, starting from the input layer, is interpreted as **U**, **0**, or **1** as the counter value is zero, negative, or positive respectively. Now, during recall, it may be seen that the output of each cell might propagate forward an undefined output. However, by adopting the Goal Seeking Network (GSN) approach of propagating the addressable set forward to the next layer, it will result in several locations being selected at the same time. When this happens, the following rules, as illustrated in Figure 7 below are adopted (they are essentially the same as for the GSN).

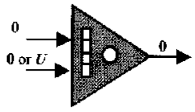
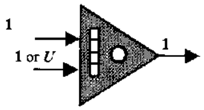
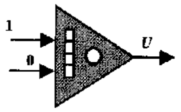
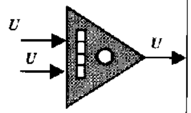
If at least one *zero* is addressed and no ones are addressed, the output is *zero.*		
If at least *one* one is addressed and no zeros are addressed, the output is *one.*		
Otherwise the output is *undefined.*		

Figure 7: Recall rules

Recall is propagated forward through the pyramid until the output is reached.

The following section will now describe efforts to assess and compare the performance of this new network's learning potential against the earlier DARN model on a few popular benchmarks.

5. EXPERIMENTAL RESULTS

5.1 Network Design Issues

For convenience, a 64-input network for the four benchmark datasets obtained from the UCI repository (UCI Repository of Machine Learning Databases, 2005) was used. There are essentially three practical options for the cell input size; that is, 2, 4, or 8 input address lines. A network configuration consisting of 4 input lines for each cell was chosen. As these data sets do not have the same number of features, the quantization levels for each feature had to be adjusted appropriately. Furthermore, the values of some of the features for several data sets had to be scaled up accordingly, as they are essentially unsuitable for representation in their original form for the WNN. In all cases, only values to the nearest integer were used. This is summarised in Table 1.

Table 1: Input data representation

Feature	Wine		B. Cancer		Diabetes		Iris	
	SF	Q	SF	Q	SF	Q	SF	Q
1	*1*	*4*	*1*	*7*	*1*	*5*	*1*	*16*
2	*1*	*3*	*1*	*7*	*1*	*9*	*1*	*16*
3	*4*	*4*	*1*	*7*	*1*	*8*	*1*	*16*
4	*1*	*5*	*1*	*7*	*1*	*8*	*1*	*16*
5	*1*	*8*	*1*	*7*	*1*	*11*		
6	*4*	*5*	*1*	*7*	*1*	*8*		
7	*5*	*5*	*1*	*7*	*25*	*7*		
8	*10*	*3*	*1*	*7*	*1*	*8*		
9	*1*	*3*	*1*	*8*				
10	*1*	*4*						
11	*5*	*4*						
12	*7*	*5*						
13	*1*	*11*						

Key :
SF : Scaling Factor
Q : Quantization (no. of bits)

A brief description of each data set along with the key features are given in tables 2 – 5.

Table 2: Iris data set

Title :	Iris
Description :	*This set contains 150 samples describing measurements on the Iris flower (two measurements each on the petal and sepal) making up the input attributes [16].*
Size of data set :	150
No. of features :	4
No. of classes:	3

Table 3: Wine data set

Title :	Wine
Description :	*The data are the results of a chemical analysis of wines grown in the same region in Italy but derived from 3 different cultivars[17]. The analysis determined the quantities of the 13 different constituents found in each of the three types of wines.*
Size of data set :	178
No. of features :	13
No. of classes:	3

Table 4: Breast cancer data set

Title :	Breast Cancer (*B. Cancer*)
Description :	*This database was obtained from the University of Wisconsin Hospital, Madison from William H. Wolberg[18], starting from 1989 until 1991. The features comprise the (tissue) clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses.*
Size of data set :	699
No. of features :	9
No. of classes:	2

Table 5: Pima Indians Diabetes data set

Title :	Diabetes
Description :	*This was taken from National Institute of Diabetes and Digestive and Kidney diseases[19]. The data was obtained from patients who are females of at least 21 years old and of Pima Indian heritage. The 8 features comprise the number of times pregnant, plasma glucose concentration, diastolic blood pressure, triceps skin fold thickness, 2-hour serum insulin, body mass index, diabetes pedigree function, and age.*
Size of data set :	768
No. of features :	8
No. of classes:	2

Two different validation techniques were examined for each of these four data sets. This is further described in the following sections.

5.2 90/10 Random Shuffling Validation Technique

In the first approach, which is labelled as 90/10 RSVT, the data was randomly shuffled a number of times to produce several sets of data partitioned approximately in a 90%-10% mix of training and test data sets respectively. The data sets, in each case, were further shuffled a hundred times and partitioned to produce additional data set pairs for training and testing purposes. The DARN simulator was then trained and tested for each data set. Following this, the average classification accuracy over the 100 runs for each data set was obtained.

5.3 The Leave-One-Out-Cross-Validation Technique

The second validation approach, which is commonly known as *Leave-One-Out-Cross-Validation* technique (L1OCV) (Stone, 1974), basically identifies one of the data samples in the collection as the test data. The remainder was used for training. This was repeated for each and every sample of data in the collection.

Fundamentally, for a collection of N data samples, there will be N simulations of one pair of (N-1) training sets and one test data set. Many believe that this approach provides a better picture of the performance of supervised classifiers.

A summary of the experimental results, expressed as the percentage correct, obtained for the two validation approaches on the benchmark data sets are shown in Tables 6 and 7.

Table 6: Experimental Results with *DARN*

	90/10 RSVT	**L1OCV**
Iris	78.73	81.33
Wine	31.47	73.60
B. Cancer	72.90	87.84
Diabetes	50.99	51.43

Table 7: Experimental Results with new *DARN*

	90/10 RSVT	**L1OCV**
Iris	81.13	94.67
Wine	31.67	76.40
B. Cancer	86.63	92.42
Diabetes	60.66	70.53

5.4 The multi-layer perceptron neural net as a classifier

To baseline the performance of the DARN, some additional experiments have been conducted with the multi-layer perceptron network that has a single hidden layer. This was trained with the standard generalized delta rule for 10 randomly shuffled data sets. Each of these data sets was partitioned in a 90%-10% mix of training and testing data respectively. The network characteristics for each of the four data sets are summarized along with the experimental results obtained in Table 8 below.

Table 8: Experimental set-up and results with the *MLP*

Features	**Iris**	**Wine**	**B. Cancer**	**Diabetes**
Input nodes	*64*	*64*	*64*	*64*
Hidden nodes	*10*	*10*	*10*	*10*
Output Nodes	*3*	*3*	*2*	*2*
Learning rate	*0.2*	*0.2*	*0.2*	*0.2*
Momentum	*0.9*	*0.9*	*0.9*	*0.9*
Max. training errors allowed	*0.01*	*0.01*	*0.01*	*0.01*
Average Classification Accuracy (%)	86.00	75.00	93.43	67.27

6. DISCUSSION & CONCLUSION

In this paper, a new weightless neural network, which has been used for data classification was described. The experimental results shown here were based on four sets of benchmark data belonging to various application areas. This ranged from studies of the four main features of the iris flower to the set of thirteen features from a database of information related to diabetes. A common network was used in classifying these four sets of data, and for validation, two different approaches were investigated. As the data sets come from a variety of sources, some of the features had to be pre-processed before being converted into the appropriate binary values. A comparison was also made between the two versions of DARN to justify the development of the proposed new model.

For both approaches, the *Leave-One-Out-Cross-Validation* technique has shown a higher level of classification accuracy than for the *90/10 Random Shuffling Validation* approach. This may not be totally unexpected, as the Leave-One-Out-Cross-Validation technique produces training sets with more samples than the *Random Shuffling Validation* approach. Comparing the results obtained, it is evident that the new DARN outperforms the old, in accurately classifying the data sets presented here. Even though the experimental results obtained from the multi-layer perceptron neural net surpasses those obtained with the *DARN*, it is believed that there is still a niche for the *DARN* to be used in appropriate applications, especially where the application is suitable to be implemented in hardware. One aspect of hardware implementation, which presents a major advantage over simulation, is that of the search time to find an addressable set within a RAM cell. Under simulation conditions this search is effectively depth first. When the cell size increases to, say, sixteen bits, the search time becomes impractical. However, when this is implemented in hardware form it may now be possible to conduct a breadth first search using parallel hardware. There is on-going work at the *Department of Electrical & Computer Engineering*, University of Auckland to implement a hardware realisation using Xilinx FPGA's.

7. ACKNOWLEDGEMENT

The authors would like to acknowledge the assistance of G.S. Poh, J.Y. Luke, Y.M. Aw and *K.M. Hoe* for their contributions in refining the efficiency of the DARN simulator. Finally, the authors are also grateful for C. C. Loy's help in running the additional series of experiments on the MLP.

8. REFERENCES

"*UCI Repository of Machine Learning Databases*", http://www.ics.uci.edu /~mlearn/MLRepository.html

Aeberhard, S., Coomans, D. and de Vel, O. (1992), "*The Classification Performance of RDA*", Tech. Report. No. 92-01, Department of Computer Science and Department of Mathematics and Statistics, James Cook University of North Queensland.

Aleksander, I., Thomas, W.V. and Bowden, P.A. (1984), "*WISARD: A radical step forward in image recognition*", Sensor Review, pp. 120-124.

Aleksander, I. and Morton, H. (1990), *Introduction to Neural Computing*, Chapman & Hull, pp. 1.

Aleksander, I. and Morton, H. (1990), *Introduction to Neural Computing*, Chapman & Hull, pp. 70.

Bowmaker, R.G. and Coghill G.G. (1992), *Improved Recognition Capabilities for Goal Seeking Neuron*, Electronics Letters, 28(3), pp. 220-221.

Deneubourg, J. L. (1990), Goss, S., Franks, N.R., Sendova-Franks, A., Detrain, C. and Chretien, L., "*The Dynamics of Collective Sorting: Robot-like Ants and Ant-like Robots*". In Meyer, J-A, & Wilson, S., Eds, *Simulation of Adaptive Behaviour: From Animals to Animats*, MIT Press, Massachusetts, pp. 356-65.

Duda, R.O., Hart, P.E., and Stork, D.G. (2001), Pattern Classification, *John Wiley & Sons*, pp. 16 – 17.

Filho, E.C.D.B.C., Fairhurst, M.C. and Bisset, D.L. (1991), *Adaptive Pattern Recognition using Goal Seeking Neurons*, Pattern Recognition Letters, 12, pp. 131-138.

Fisher, R. A. (1936), The Use of Multiple Measurements in Taxonomic Problems, *Annals of Eugenics* 7, 2, 179-188.

Hoe, K.M., Lai, W.K. and Tai, T.S.Y. (2002), Homogeneous Ants for Web Document Similarity Modeling and Categorization, *International Workshop on Ant Algorithms*, Brussels Belgium, pp. 256-261.

Kan, W.K. and Aleksander, I. (1987), A Probabilistic Logic Neuron Network for Associative Learning, *Proceedings of IEEE 1st Int. Conf. on Neural Networks*, San Diego, 2, pp. 541-548.

Lippmann, R. P. (1989), Pattern classification using neural networks, *IEEE Communication Magazine*, pp. 47 – 64.

Ludermir, T.B., de Carvalho, A., Braga, A.P., and de Souto, M.C.P. (1999), Weightless Neural Models: A Review of Current and Past Works, *Neural Computing Surveys*, 2, Also available at, http://www.icsi.berkeley.edu/~jagota/NCS, pp. 41 – 61, .

Robert Hecht-Nielsen (1989), *Neurocomputing*, Addison-Wesley Publishing Company, pp. 2.

Harvey, R.L. (1994), *Neural Network Principles*, Prentice Hall, pp. 3.

Rumelhart, D.E., Hinton, G.E. and William, R.J. (1986), Learning internal representation by error propagation in *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, 1*, MIT Press, Cambridge, MA.

Smith, J.W., Everhart, J.E., Knowler, W.C. and Johannes, R.S. (1998), Using the ADAP learning algorithm to forecast the onset of diabetes mellitus, *In Proceedings of the Symposium on Computer Applications and Medical Care*, IEEE Computer Society Press, pp. 261 – 265.

Stone, M. (1994), Cross-validatory Choice and Assessment of Statistical Predictions, *Journal of the Royal Statistical Society*, Series B, 36, pp. 111 – 147.

Wolberg, W.H. and Mangasarian, O.L. (1990), Multisurface method of pattern separation for medical diagnosis applied to breast cancer cytology, *Proceedings of the National Academy of Sciences*, U.S.A., 87, pp. 9193 – 9196.