MODELING THE TOWER OF HANOI USING NEURAL NETWORK

MOHAMED OTHMAN
MOHD. HASSAN SELAMAT
ZAITON MUDA
LILI NORLIYA ABDULLAH
Department of Computer Science
University Pertanian Malaysia
43400 UPM Serdang Selangor

Abstract

This paper discusses the modeling of Tower of Hanoi using the concepts of neural network. The basis idea of backpropagation learning algorithm in Artificial Neural Systems is then described. While similar in some ways, Artificial Neural System learning deviates from tradition in its dependence on the modification of individual weights to bring about changes in a knowledge representation distributed across connection in a network. This unique form of learning is analyzed from two aspects: the selection of an appropriate network architecture for representing the problem, and the choice of a suitable learning rule capable of reproducing the desired function within the given network.

Key words: Tower of Hanoi; Backpropagation Algorithm; Knowledge Representation;

1. Introduction

"In the great temple of benares....... beneath the done which marks the centre of the world, rests a brass place in which are fixed three diamond needles, each a cubit high and as thick as the body of a bee. On one of these needles, at the creation, God placed sixty-four disks of pure gold, the largest disc resting on the brass plate and the others getting smaller and smaller up to the top one. This is the Towers of Brahma. Day and night unceasingly the priests transfer the discs from one diamond needle to another according to the fixed and immutable laws of Brahma, which require that the priest on duty must not move more than one disc at a time and that he must place this disc on a needle so that there is no smaller disc below it" by W.W.Rouse Ball, 1974.

This quotation from Mathematical Recreations and Essays hints at the origin of the classic puzzle, written by de Parville in La Nature, Paris, and known since 1883 as the "Towers of Hanoi".

In its classical form, the Towers of Hanoi puzzle consists of three vertical pegs mounted on a board and a set of disks or rings graded in size. Initially, all the rings are stacked on one peg in order by size, with the largest ring at the bottom and the smallest at the top. Solving the puzzle involves moving the rings from their original peg (source peg) to one of the other pegs (target peg) and stacking them in the same initial order by using two simple rules: move only one ring at a time, and never stack a ring on top of a smaller one.

Mathematicians, computer scientists, and artificial intelligence professionals keep a special place in their hearts for the Towers of Hanoi. Almost every computer-science textbook uses the Towers of Hanoi to teach recursion. Recursive Towers of Hanoi algorithms have also become one of the standard benchmark tests to evaluate the performance of computer hardware.

The minimum number of moves to complete the puzzle, if one makes all the right moves, is a function of the number of rings. For *n* rings, a minimum of $2^n - 1$ moves are necessary to complete the puzzle.

A recursive procedure to solve the Towers of Hanoi puzzle can be expressed:

```
Towers(n,source,target)
If n = 1 {\bf Then} Move (n,source,target)
Else Towers (n-1,source,spare)
Move (n,source,target)
Towers (n-1,spare,source).
```

Although recursive procedures are very powerful mechanisms that provide simple and elegant ways of thinking about these types of problems, they do not always use computer resources such as time and working memory in the most efficient way.

The human brain is the most complex computing device known to man. The brain's powerful thinking, remembering, and problem-solving capabilities have inspired many scientists to attempt computer modelling of its operation. One group of researchers has sought to create a computer model that matches the functionality of the brain in a very fundamental manner; the result has been neural computing. Neural networks are human attempts to simulate and understand what goes on in nervous system, with the hope of capturing some of the power of these biological systems. These models are composed of many nonlinear computational element operating in parallel and arranged in patterns reminiscent of biological neural nets.

Neural network, parallel distributed processing, or connectionist model provides a unique computing architecture whose potential has only begun to be tapped. Used to address problems that are intractable or cumbersome with traditional methods, these new computing architectures inspired by the structure of the brain are radically different from the computers that are widely used today. They represent some of the most active research areas in artificial intelligence and cognitive science today.

Neural networks provide an effective approach for a broad spectrum of applications. The recent excitement is due to the promising qualities artificial neural systems exhibit in addressing challenging questions of intelligence. For instance, characteristic of a neural system is its distribution of knowledge across a network of units. Neural systems have shown promising results for a number of these problems, including content addressable memory, pattern recognition and association, category formation, speech production, and global optimization, see Kohonen 1984; Rumelhart and McClelland 1986; Hopfield and Tank 1986.

Computational elements or nodes are connected via weights that are typically adapted during use to improve performance. Neural net models are specified by the net topology, node characteristics, and training or learning rules. These rules specify an initial set of weights and indicate how weights should be adapted during use to improve performance. Most neural net algorithm also adapt connection weights in time to improve performance based on current result. Adaptation or learning is a major focus of neural net research. Adaptation also provides a degree of robustness by compensating for minor variabilities in characteristics of processing elements. These nets are highly parallel building blocks that illustrate neural net components and design principles and can be used to construct more complex systems.

2. Input Representation

The task of modeling is performed in two ways. First, train a three-layer backpropagation network to solve the Towers of Hanoi puzzle by presenting it with all the steps necessary to solve the puzzle for a fixed number of rings. The net will have to learn the steps in the correct sequence. Second, train a composite network to solve the puzzle by learning its rules. By learning the rules, the net can solve the problem for any given number of rings.

To solve the Towers of Hanoi puzzle, the four-ring puzzle and a training set of 15 samples is used. Each sample represents one move and consists of two12-element vectors. Each vector is to be interpreted as three groups of four elements.

Each group represents one peg, and the position of each element within the group represents a ring size, increasing from left to right. For instance, the first input vector (1111 0000 0000) represents the initial condition of the puzzle when all four rings are on the first (source) peg. This is indicated by placing four ``1s" in the first group. The leftmost "1" on the first group represents the presence of the smallest ring in the first peg and so on. Presented with this input vector, the net must learn to generate the output vector (0111 1000 0000), indicating that the smallest ring must be moved from the first to the second (next clockwise) peg. For the second step the correct move will be (0011 1000 0100), indicating that the second ring has been placed on the third peg. Listing 1 shows the contents of the file holding the training set.

```
(1111 0000 0000)  (0111 1000 0000)
(0111 1000 0000)  (0011 1000 0100)
(0011 1000 0100)  (0011 0000 1100)
(0011 0000 1100)  (0001 0010 1100)
(0001 0010 1100)  (1001 0010 0100)
(1001 0010 0100)  (1001 0110 0000)
(1001 0110 0000)  (0001 1110 0000)
(0001 1110 0000)  (0000 1110 0001)
(0000 1110 0001)  (0000 0110 1001)
(0000 0110 1001)  (0100 0010 1001)
(0100 0010 1001)  (1100 0010 0001)
(1100 0010 0001)  (1100 0000 0011)
(1100 0000 0011)  (0100 1000 0011)
(0100 1000 0011)  (0000 1000 0111)
(0000 1000 0111)  (0000 0000 1111)
```

Listing 1: The 15 Samples of Trainning Set

A training set is a set of input or target pattern pairs. The patterns in the training set are presented to the network repeatedly. Each training iteration consists of presenting each input/output pattern pair once. When all patterns in the training set have been presented, the training iteration is completed, on the next training iteration is begun. This might entail hundreds or thousands of training iterations.

3. Methodology

To solve the Towers of Hanoi Puzzle, the backpropagation learning algorithm was chosen for its well known properties and its successes in solving similar tasks. The net built can be seen in Figure 1.

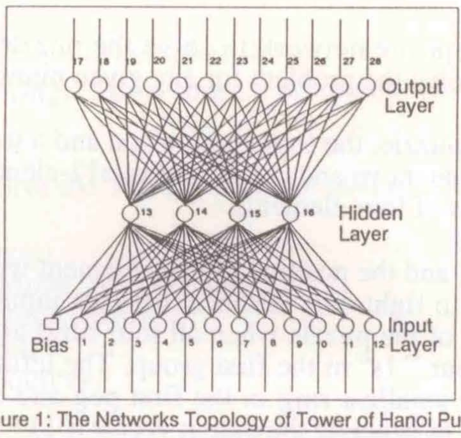


Figure 1: The Networks Topology of Tower of Hanoi Puzzle

Different networks were used to learn the minimum step sequence to solve thefour ring Towers of Hanoi Puzzle. The size (number of elements) of the inputvectors determined the number of neurodes in the input layer; the number of desired outputs (target vector's size) determined the number of neurodes in the output layer; and a different number of neurodes from three to twelve were used in the hidden layer.

The size of the hidden (middle) layer is really our choice. If the middle layer is too large, it will encourage the network to memorize the input patterns rather than generalized the input into features. This reduces the network's ability to handle unfamiliar inputs after training is complete. On the other hand, a middle layer that is too small will drastically extend the number of iterations required to train the network and will likely reduce the accuracy of recall.

There are three to twelve processing elements used. Figures 2(a), 2(b), 3(a), and 3(b), shows the result from each of the size tested.

Figure 2(a): Output for each Topology

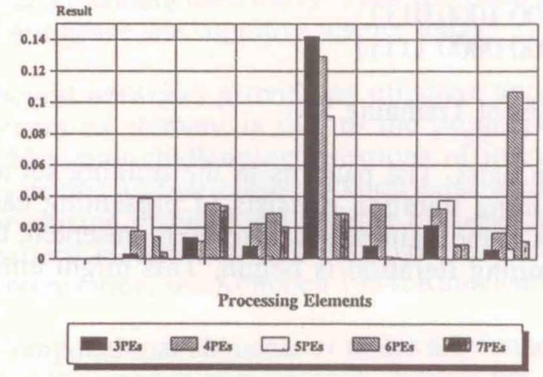


Figure 2(b): Output for each Topology

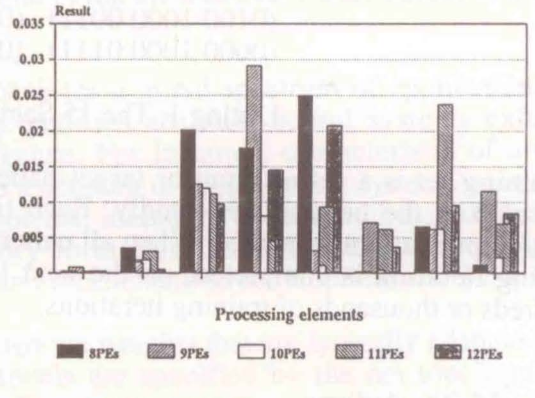


Figure 3(a): Output for each Topology

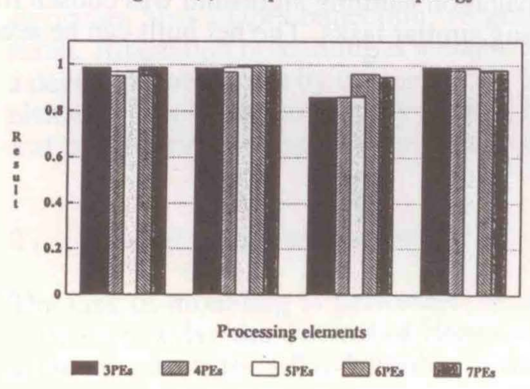
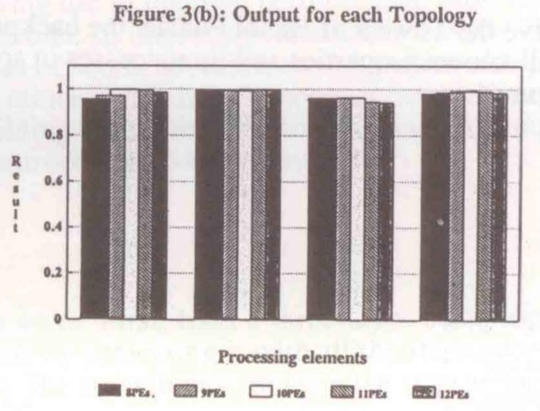


Figure 3(b): Output for each Topology



As we can see here, the ten neurodes in the middle layer can be considered as a compromise, and the most efficient network.

Typically an application of backpropagation requires both a training set and a test set. Both the training set and the test set contain input/output pattern pairs. While the training set is used to train the network, the test set used to assess the performance of the network after training is complete. In this application both sets are taken from real data.

This backpropagation networks are trained by a technique called supervised learning, where by the network is presented with a series of pattern pairs - each pair consisting of an input pattern and a target output pattern. Each pattern is a vector of real input pattern and is used to determine the error values in the network when the weights are adjusted. Upon each presentation, weights are adjusted to decrease the difference between the network's output and the target output.

The principal strength of backpropagation is:

1. Its relatively general pattern mapping capability; it can learn a tremendous variety of pattern mapping relationship.

2. It does not require any a priori knowledge of a mathematical function that maps the input patterns to the output patterns; backpropagation merely needs examples of the mapping to be learned.

3. The flexibility of the paradigm is enhanced by the large number of design choices available - choices for the number of layers, interconnections, processing units, learning constant and data representation.

4. Able to address a broad spectrum of applications.

The largest drawback with backpropagation appears to be its convergence time. Training sessions can require hundreds or thousands of iterations for relatively simple problems. Realistic applications may have thousands of examples in a training set, and it may takes days of computing time to complete training.

4. Output Representation

When a network is trained successfully, it produces correct answers more and more often as the training session progresses. It is important, to have a quantitative measure of learning. Table 1 shows the results obtained from training the different networks to learn the steps of the four ring puzzle.

Topology	Iteration	Time(sec)	Result
12-03-12	5000	476	Fail
12-04-12	274	31	Success
12-05-12	141	18	Success
12-06-12	66	12	Success
12-07-12	48	10	Success
12-08-12	52	10	Success
12-09-12	42	10	Success
12-10-12	42	9	Success
12-11-12	38	10	Success
12-12-12	37	10	Success

Table 1: Table of Results Obtained From Each Topology

All but the first net succeeded in learning the steps of the puzzle. It seems reasonable to attribute the failure of the 12-03-12 network to the small number of neurodes in its hidden layer. The net could not map the 12-dimensional input patterns properly into its 3-dimensional space. The training was stopped at the 5,000th iteration, because the value of the output error was oscillating rather than decreasing, as expected in the case of successful learning. Table 1 also shows how the number of learning iterations over the training set decreases while the size of the hidden layer increases. Meanwhile, the learning times seem to have an optimal point, a minima. When selecting from one of these networks, the trade-off between the number of learning iterations and times was taken in consideration. So the network with ten processing elements in the hidden layer was chosen, see Figure 4(a) and 4(b) for each of the criteria: iterations and time.

Figure 4(a): Iterations for each Topoloy

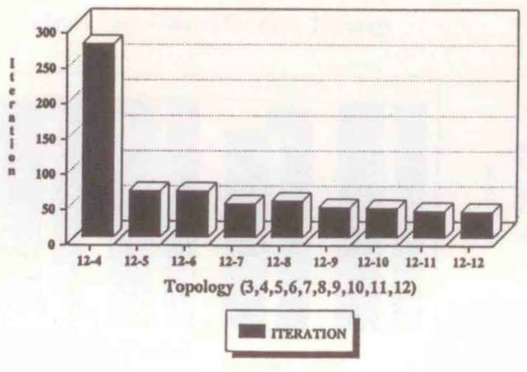
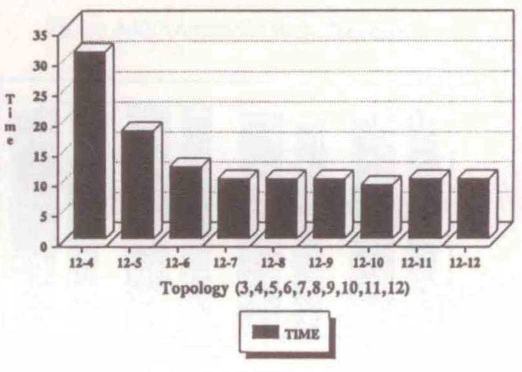


Figure 4(b): Time for each Topology



From the different number of processing elements in the hidden layer, it can be obtained a different set of summation for each topology as shown in Figure 5. Each topology too creates a different set of transfer function as in Figure 6.

Figure 5: Summation for each Topology

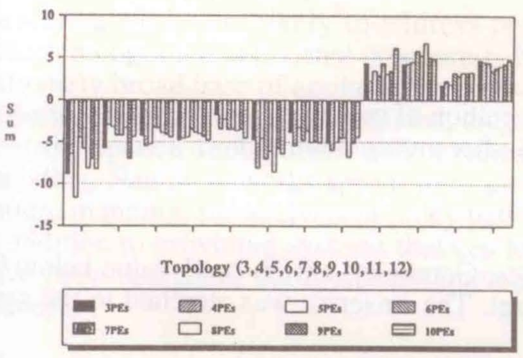
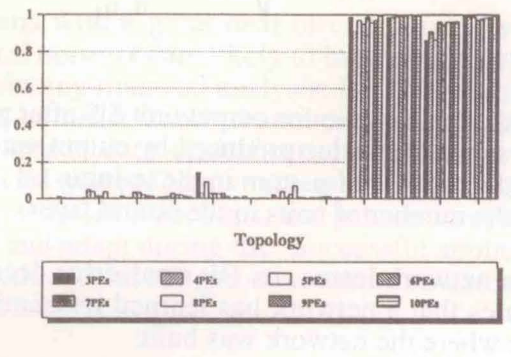


Figure 6: Transfer Function for each Topology



We too can use different set of transfer function on each network. Since the topology with 10 processing elements in the hidden layer is the best, a different set of transfer function: tanh, sine and sigmoid was trained. The sigmoid curve is the best solution because it activates the summation and transfer separately, where the linear and tanh function treat the summation and the transfer function equally. This is not true. From the equation given and it was discussed, see Aleksander, Igor and Morton Helen, 1990, Cybenko, George 1989 and Hecht-Nielsen, Robert, 1989, they are supposed to be different. The error occurred obtained in the sigmoid function is the minimum. That is why the sigmoid curve is the best transfer tool. See Figure 7. The errors are obtained from the difference between the desired output and the output produced. This can be seen in Figure 8 below.

Figure 7: Error for each Topology

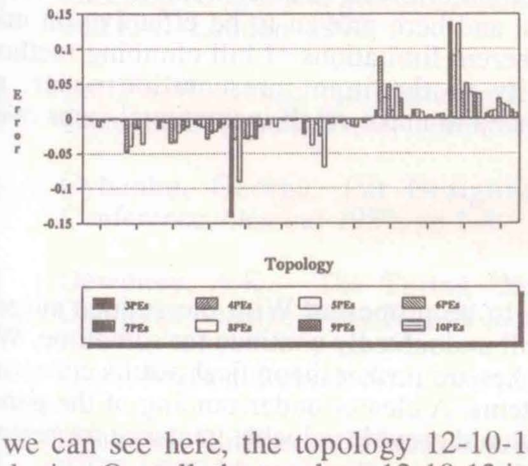
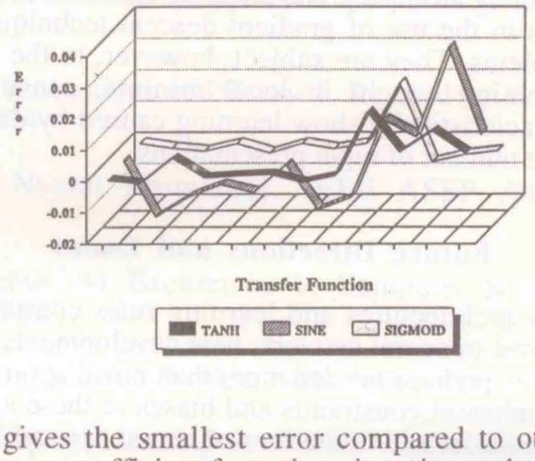


Figure 8: Error for Tanh, Sine, Sigmoid



As we can see here, the topology 12-10-12 gives the smallest error compared to other topologies. Overall, the topology 12-10-12 is the most efficient from time, iteration and error occurred.

The root-mean-squared (rms) error is usually calculated to reflect the degree to which learning was taken place in the network. This measurement reflects how close the network is to getting the correct answers.

$$\text{rms} = \sqrt{\frac{\Sigma p \Sigma j (t_{jp} - x_{jp})^2}{n_o n_j}} \qquad (1)$$

where

t_{jp} is the target value for output unit \$j\$ after presentation of pattern *p*
x_{jp} is the output value produced by output unit \$j\$ after presentation of *p*
n_p is the number of pattern in the training set
n_o is the number of units in the output layer

As the network learns, its {\it rms} error decreases. Generally, an {\it rms} value below 0.1 indicates that a network has learned its training set. The rmserror was attached to the same frame where the network was built.

5. Conclusion

Here, we can conclude that the ten neurodes in the middle layer i.e. 12-10-12 is the most efficient net topology for modeling the Tower of Hanoi. Its also shows that the learning rule success in solving the task.

Particular, two central concerns was examined: the nature of learnable representations within Artificial Neural System's designs, and the different forms of learning rules used to encode representation within the connective weights of a network. The choice of a network's architecture imposes certain inherent biases which bear directly on the kinds of functions that can ultimately be represented within the network.

A learning rule is an algorithm used to modify weights in an neural network for the purpose of acquiring an appropriate mapping function. The error correcting rules differ from correlational rules in the use of gradient descent techniques, and here proven to be effective on many problems. They are subject, however, to the inherent limitations of hill climbing methods: becoming caught in local minima, sensitivity to the input presentation order, and characteristically show learning caused by the need to make small incremental steps over a large number of input presentations.

6. Future Directions and Issues

New architectures and learning rules continue to be proposed. With the sudden surge of interest in neural network, new developments will undoubtedly continue for sometime. What is now perhaps needed more than novel approaches are further theoretical results concerning the inherent constraints and biases of these systems. A clearer understanding of the general capabilities and limitations of neural network's would provide valuable guidance for research in this field.

Computational models with neural network architectures have progressed tremendously since their inception in the 1950s. Advances over four decades have brought us from simple two-layered architectures that required cumbersome hardware implementation to the simulation of hundreds of thousands of processing units on a digital computer. Biological neural models have advanced from simplified models of binary-state neurons to the simulation of networks of neurons with many biological details included.

Representation and preprocessing can influence the performance level of a neural network and can help solve the problem of addressing the high degree of complexity in real-world problems.

Other issues that require further work include assessments of capabilities, accountability, and reliability in trained neural networks.

Neural networks are likely to address problems with a great deal of complexity, such as speech recognition and visual processing. Neural networks are likely to have an impact on an extremely broad base of applications areas, including financial analysis; image processing in defense, medical, industrial domains, diagnosis in medical and commercial domains; robotic control; speech recognition and synthesis; sensor data classification; and information encoding. Neural networks appear to be good at solving problems such as pattern recognition, pattern mapping, the analysis of noisy patterns, associative lookups, and pattern completions, in addition to providing systems that can learn and adapt during use. Successful applications have been designed, built, and commercialized, and investigators continue to extend this success.

Although evolution built the first neurobiological system, humankind's relentless search to emulate nature has motivated the neural network approach to computing. As the capabilities of neural network architectures become understood more completely, both their limits and abilities provide lessons for computer engineering and brain research. The continuing effort to evolve human-made systems that mimic parts of biological architectures has, with the advent of neural networks, taken a giant step forward. The success of biological neural systems is certain and self-evident. The performance and utility of artificial system is gradually unfolding as research progresses, and will continue to have an impact on our lives.

References

[1] Barr A. and Feigenbaum E., 1981a, The Handbook of Artificial Intelligence, vol: 1, Addison-Wesley Pub. Co., 1981.

[2] Aleksander, Igor and Morton, Helen: An Introduction to Neural Networks, Chapman and Hall, London, 1990.

[3] Ball, W.W. Rouse: {\it Mathematical Recreations and Essays}, Revised by H.S.M. Coxeter, University Press, Toronto, 1974.

[4] Cybenko, George: {\it Designing Neural Networks}, IEEE ASSP Annual Conference, October 1989, pp 1-3.

[5] Dewdney, A.K.: The Turing Omnibus: 61 Excursions in Computer Science, Computer Science Press, New York, 1989.

[6] Hecht-Nielsen, Robert: Neurocomputing, Addison-Wesley, New York, 1989.

[7] Hohensee, William and Mathews, Christopher: Learning in Artificial Neural Systems, IEEE ASSP Magazine, December 1987, pp 1-6.

[8] Hopfield, John, Tank, David W.: Neural Computation of Decisions in Optimization Problems, Biological Cybernatics, Volume 52, pp 141-152, 1985.

[9] Kosko, Bert: {\it Neural Network and Fuzzy System}, Prentice Hall, New Jersey, 1992.

[10] Sharkey, N.E: {\it Neural Computing Architecture}, MIT Press, Massachusetts, 1988.

[11] Anderson, J.A, Rosenfield E.: Neurocomputing: Foundations of Research, MIT Press, 1988.

[12] Desiero, D: Adding a Conscience to Competitive Learning, Proceedings International Conference on Neural Network, IEEE ASSP Press, New York, Volume I, pp 117-124, July 1985.

[13] Hebb, D: The Organization of Behaviour}, Wiley, New York, 1949.

[14] McCulloch, W.S, Pitts, Walter: A Logical Calculus of the Ideas Immanent in Nervous Activity, Bulletin of Mathematical-Biology, Volume V, pp 115-133, 1943.

[15] Rosenblatt,F: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Psychology Review, Volume 65, pp 386-408, 1958.