Proceedings of 4th International Symposium on
Intelligent Manufacturing Systems, September 6-8, 2004: 1196-1206

ARTIFICIAL NEURAL NETWORKS EDUCATION SIMULATOR

Mehmet Tektaş Vedat Topuz Necla Tektas

[1,2,]Marmara Üniversitesi Teknik Bilimler Meslek Yüksekokulu 81400 Göztepe İstanbul

Tel: (0216) 336 57 70 / 624 Fax: (0216) 418 25 05

ABSTRACT:

ANN process information in asimilar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in paralel to solve a spesific problem. Over the last decades ANN is the most popular artificial intelligence technics which used from industry to education, from military to medicine. Because of the ANN is a interdicipliner branch, we have to understand its arthitecture and structure detailed.

Especially, learning process in technology education is effective,if it supports simulation tools. Education which based on simulator is the most factor effecting learning and understanding.Thus, ANN simulator which realized in this study is increased ANN education visually. According to this aim, in this study, by the way learning process, we considered ANN and realized ANN simulator which includes some components. These components are inputs, weights, activation functions, neurons of layer,learning types and network types according to help learning ANN visually.

Keywords: ANN, education, simulator.

I. INTRODUCTION

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements(neurones) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

There are multitudes of different types of ANNs (Artificial Neural Network). Some of the more popular include the multilayer perceptron which is generally trained with the backpropagation of error algorithm. Another way of classifying ANN types is by their method of learning (or training), as some ANNs employ supervised training while others are referred to as unsupervised or self-organizing. Supervised training is analogous to a student guided by an instructor. Unsupervised algorithms essentially perform clustering of the data into similar groups based on the measured attributes or features serving as inputs to the algorithms. This is analogous to a student who derives the lesson totally on his or her own. ANNs can be implemented in software or in specialized hardware.

In section II, we explained the ANN in detail which includes Modelling a neuron, Activation function, Perceptron, Learning rules (Hebb, Hopfield, Delta), Layers of Neurons (Backpropagation Algorithms), Teaching an Artificial Neural Network (Supervised learning., Unsupervised learning). In section III, we explained realizing ANN simulator which has three main part (Perceptron, having diffrent learning rules one layers ANN, for supervised learning three layer backpropagation algorithms).

II. ARTIFICIAL NEURAL NETWORKS

An ANN is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. This is true of ANNs as well. ANN is the type of information processing system whose architecture is inspired by the structure of biological neural systems based on the following assumptions:

Characteristics of Neural Networks

- Each neuron is connected to other neurons by means of interconnections or links with an associated weight.
- Memories are stored or represented in a neural network in the pattern of interconnection strengths among the neurons.
- Information is processed by changing the strengths of interconnections and/or changing the state of each neurons.
- A neural network is trained rather than programmed.

Strengths of Neural Networks

- Generalization
- Self-organization
- Can recall information based on incomplete or noisy or partially incorrect inputs.
- Inadequate or volatile knowledge base
- Performs well in data-intensive applications
    - Data is intrinsically noisy and error-prone

2.1. ANN history

The branch of artificial intelligence called neural networks dates back to the 1940s, when McCulloch and Pitts [1943] developed the first neural model. This was followed in 1962 by the perceptron model, devised by Rosenblatt, which generated much interest because of its ability to solve some simple pattern classification problems. This interest started to fade in 1969 when Minsky and Papert [1969] provided mathematical proofs of the limitations of the perceptron and pointed out its weakness in computation.

The last decade, however, has seen renewed interest in neural netivorks, both among researchers and in areas of application. The development of more-powerful networks, better training algorithms, and improved hardware have all contributed to the revival of the field. Neural-network paradigms in recent years include the Boltzmann machine, Hopfield's network, Kohonen's network, Rumelhart's competitive learning model, Fukushima's model, and Carpenter and Grossberg's Adaptive Resonance Theory model [Wasserman 1989; Freeman and Skapura 1991]. The field has generated interest from researchers in such diverse areas as engineering, computer science, psychology, neuroscience, physics, and mathematics.

2.2. Modelling of a Neuron

To model the brain we need to model a neuron. Each neuron performs a simple computation. It receives signals from its input links and it uses these values to compute the activation level (or output) for the neuron. This value is passed to other neurons via its output links.The input value received of a neuron is calculated by summing the weighted input values from its input links. That is ; $n = W_k.p_k$ olur.(1)

An activation function takes the neuron input value and produces a value which becomes the output value of the neuron. This value is passed to other neurons in the network.This is summarised in this diagram and the notes in figure 1.

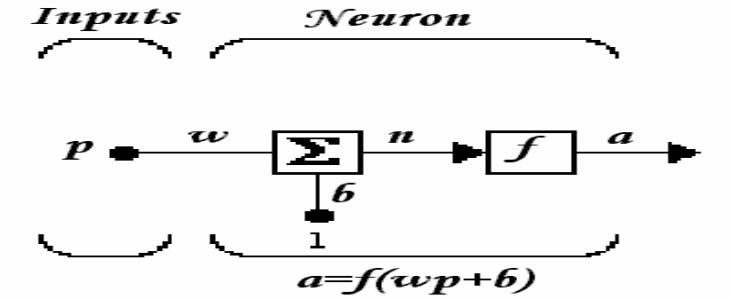

Figure 1. Modelling of an Artificial Neuron

Where;

p:Input value; **w**:Weight ; **n:**Weighted sum of inputs; **f:**Aktivation function; **a:**Output value

2.3. Activation Function

The behaviour of an ANN depends on both the weights and the input-output function (activation function) that is specified for the units. This function typically falls into one of three categories: linear (or ramp), threshold, sigmoid. For linear units, the output activity is proportional to the total weighted output. For threshold units, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value. For sigmoid units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurones than do linear or threshold units, but all three must be considered rough approximations.Some common activation functions are shown in figure 2.

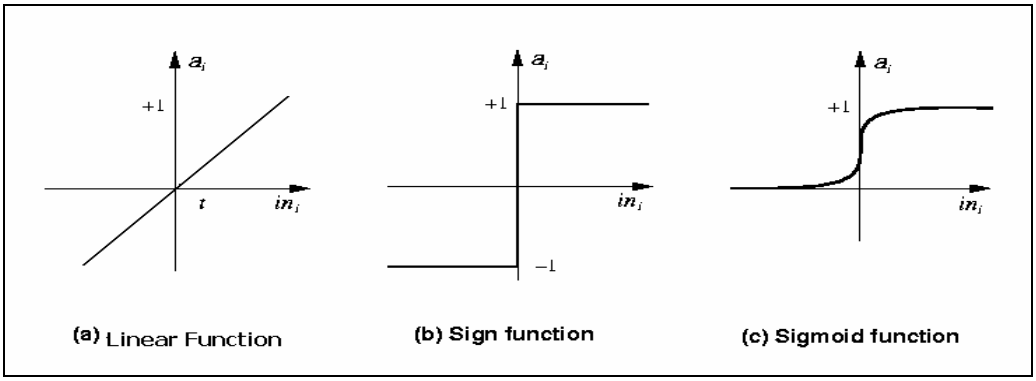

Figure 2 ANN Activation Functions

These functions can be defined as follows.

Lin(x)= x; Sign (x)=+1 if x >= 0, else –1 Sigmoid(x) $=1/(1+e^{-x})$

2.4. Perceptron

Early ANNs, usually consisting of a single layer, and using simple threshold functions, were called Perceptrons. A typical perceptron type network is shown in figure 3. The architecture of a Perceptron consists of a single input layer of many neurodes, and a single output layer of many neurodes. The network must also implement the Perceptron learning rule for weight adjustment. This learning rule compares the actual network output to the desired network output to determine the new weights.

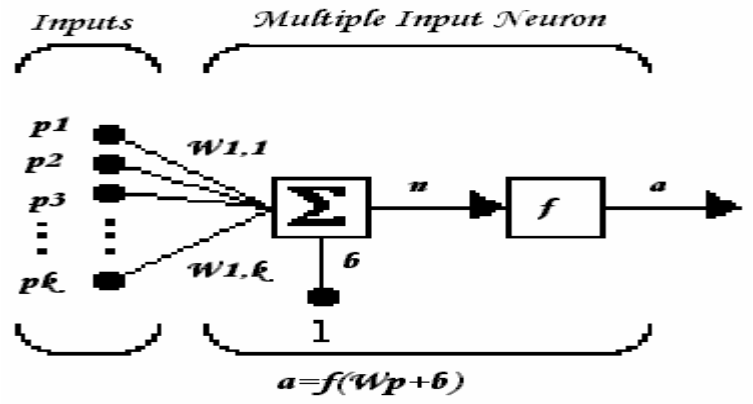

Figure 3 Modelling of Perceptron

Before a network such as the one presented above can be used, it must be taught This is accomplished by altering the weights in such a way that the output from the network moves toward the value specified by the designer for a particular input. This algorithms is shown in figure 4.

Inıtialize $W \leftarrow [0...0]$

Until a complete pass through the training set ;Weight updates do: $\{$

1→ Select an example $\varepsilon_{(k)} = (P_k, c_k)$ and compute $W * P_k$,

The output of the neuron $a_k = 1$ İf $W * P_k > 0$

$= 0$ otherwise

2→ $W \leftarrow W + \eta (c_k - a_k) X_k$

$\}$ $W^* \leftarrow W$

Where $\eta = LearningRate > 0$; $c_k = Desired\ Output$

Figure 4 Perceptron Learning Algorithms

2.5. Learning Rules

There are a variety of learning laws which are in common use. These laws are mathematical algorithms used to update the connection weights. Most of these laws are some sort of variation of the best known and oldest learning law, Hebb's Rule. Man's understanding of how neural processing actually works is very limited. Learning is certainly more complex than the simplification represented by the learning laws currently developed. Research into different learning functions continues as new ideas routinely show up in trade publications etc. A few of the major laws are given as an example below.

- **Hebb's Rule**

The first and the best known learning rule was introduced by Donald Hebb. The description appeared in his book *The organization of Behavior* in 1949. This basic rule is: If a neuron receives an input from another neuron, and if both are highly active (mathematically have the same sign), the weight between the neurons should be strengthened.

- **Hopfield Law**

This law is similar to Hebb's Rule with the exception that it specifies the magnitude of the strengthening or weakening. It states, "if the desired output and the input are both active or both inactive, increment the connection weight by the learning rate, otherwise decrement the weight by the learning rate." (Most learning functions have some provision for a learning rate, or a learning constant. Usually this term is positive and between zero and one.)

- **The Delta Rule**

The Delta Rule is a further variation of Hebb's Rule, and it is one of the most commonly used. This rule is based on the idea of continuously modifying the strengths of the input connections to reduce the difference (the delta) between the desired output value and the actual output of a neuron. This rule changes the connection weights in the way that minimizes the mean squared error of the network.

2.6. Layers of Neurons

In the section above, simple networks of one layer were discussed; these networks are a subset of a larger class of network called the Feedforward or Associative network. A generalised feedforward network can have any number of layers, its distinguishing feature is that it has no signal paths which feed data from the outputs back towards the inputs. In other words, data only flows in one direction; from the inputs to the outputs. This is in contrast to the Feedback or Auto-Associative network in which data can flow back from output to input. is shown in figure 5.

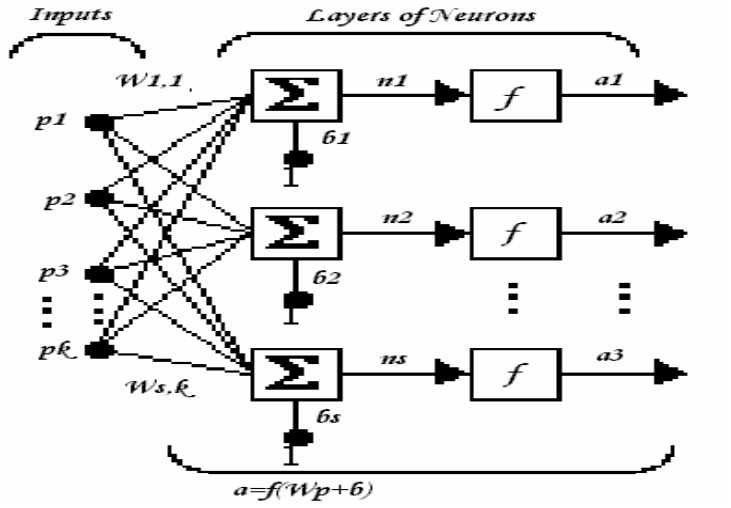

Figure 5 Layers Of Neurons

2.7. Back Propagation

Back Propagation (BP) is a method for training multilayer feedforward networks is shown in figure 6. It works by training the output layer in the same way as was shown for the perceptron, and then propagating the error calculated for these output neurons, back though the weights of the net, to train the neurons in the inner (hidden) layers.

Its basis is that the state of the network always changes in such a way that the output follows the error curve of the network downwards: that is, the error always decreases . This idea is called *Gradient descent.*

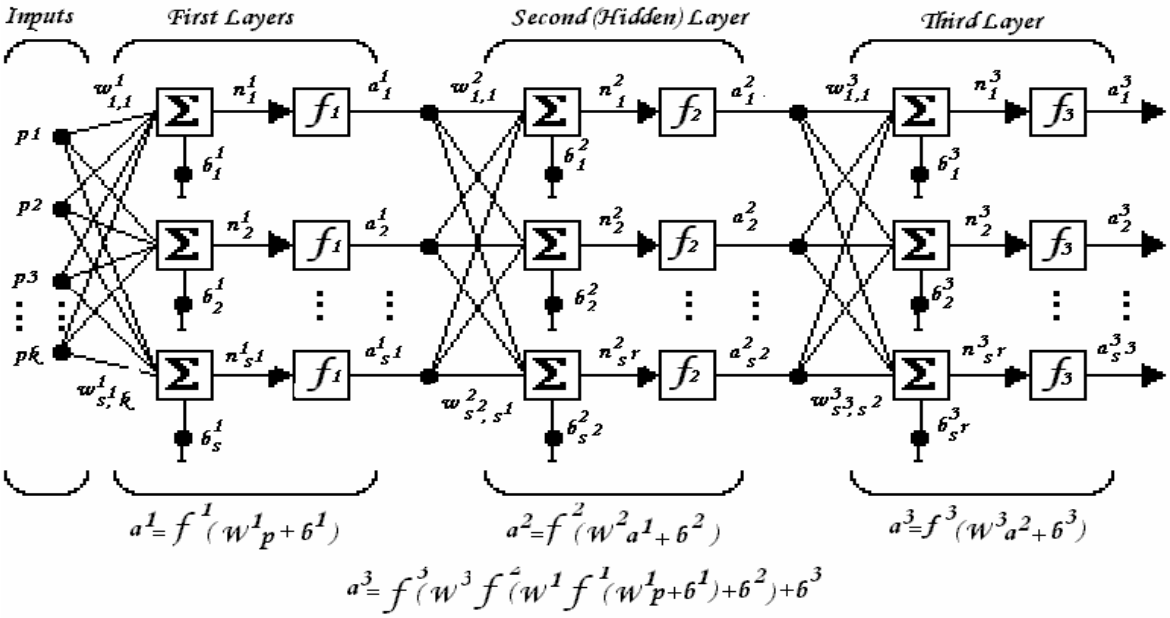

Figure 6 Three Layer ANN

"Back-Propagation" is a mathematical procedure that starts with the error at the output of a neural network and propagates this error backwards through the network to yield output error values for all neurons in the network. A common form of learning is "trial and error". A "trial" is the output of a system in response to particular stimuli. An "error" is the external reaction to the output of the system that is supplied to the system as some other kind of stimulus. A system capable of "trial and error" learning relies on receiving feedback that describes the nature and severity of mistakes. The system can use the error information to make corrections in the way it responds to that particular combination of stimuli in the future.

Back-Propagation yields neuron error values throughout a neural network. Learning occurs when neuron input weights and bias values are adjusted in an attempt to reduce the output error for the same stimuli .It should be noted that defining a mechanism for learning implicitly defines the nature of phenomena that will frustrate learning.

First, an input is applied to the network and the output is calculated. The output is then compared with the target and an error is calculated, as with the perceptron. In the perceptron case the neurons were considered to be simple threshold units; however, if they have a more complex activation function, then the error must be multiplied by the derivative of the activation function (for example the sigmoid function). Where BP differs from simple perceptron learning is in the training of the hidden layers. The idea is to propagate the value of error calculated for the output neurons, back through the weights, to the neurons of the hidden layer, and hence calculate a value of error for these. This is done by multiplying the value of error from each output layer by the weight connecting that output layer to the hidden layer neuron, and adding all the contributions together. Again, if we have an activation function, we must multiply by its derivative.The process is then repeated for all the hidden layer neurons.

2.8 Teaching an Artificial Neural Network

- **Supervised Learning.**

The vast majority of artificial neural network solutions have been trained with supervision. In this mode, the actual output of a neural network is compared to the desired output. Weights, which are usually randomly set to begin with, are then adjusted by the network so that the next iteration, or cycle, will produce a closer match between the desired and the actual output. The learning method tries to minimize the current errors of all processing elements. This global error reduction is created over time by continuously modifying the input weights until an acceptable network accuracy is reached.

With supervised learning, the artificial neural network must be trained before it becomes useful. Training consists of presenting input and output data to the network. This data is often referred to as the training set. That is, for each input set provided to the system, the corresponding desired output set is provided as well. In most applications, actual data must be used. This training phase can consume a lot of time. In prototype systems, with inadequate processing power, learning can take weeks. This training is considered complete when the neural network reaches an user defined performance level. This level signifies that the network has achieved the desired statistical accuracy as it produces the required outputs for a given sequence of inputs. When no further learning is necessary, the weights are typically frozen for the application. Some network types allow continual training, at a much slower rate, while in operation. This helps a network to adapt to gradually changing conditions.

Training sets need to be fairly large to contain all the needed information if the network is to learn the features and relationships that are important. Not only do the sets have to be large but the training sessions must include a wide variety of data. If the network is trained just one example at a time, all the weights set so meticulously for one fact could be drastically altered in learning the next fact. The previous facts could be forgotten in learning something new. As a result, the system has to learn everything together, finding the best weight settings for the total set of facts. For example, in teaching a system to recognize pixel patterns for the ten digits, if there were twenty examples of each digit, all the examples of the digit seven should not be presented at the same time.

How the input and output data is represented, or encoded, is a major component to successfully instructing a network. Artificial networks only deal with numeric input data. Therefore, the raw data must often be converted from the external environment. Additionally, it is usually necessary to scale the data, or normalize it to the network's paradigm. This pre-processing of real-world stimuli, be they cameras or sensors, into machine readable format is already common for standard computers. Many conditioning techniques which directly apply to artificial neural network implementations are readily available. It is then up to the network designer to find the best data format and matching network architecture for a given application. After a supervised network performs well on the training data, then it is important to see what it can do with data it has not seen before. If a system does not give reasonable outputs for this test set, the training period is not over. Indeed, this testing is critical to insure that the network has not simply memorized a given set of data but has learned the general patterns involved within an application.

- **Unsupervised Learning.**

Unsupervised learning is the great promise of the future. It shouts that computers could someday learn on their own in a true robotic sense. Currently, this learning method is limited to networks known as self-organizing maps.This promising field of unsupervised learning is sometimes called self-supervised learning. These networks use no external influences to adjust their weights. Instead, they internally monitor their performance. These networks look for regularities or trends in the input signals, and makes adaptations according

to the function of the network. Even without being told whether it's right or wrong, the network still must have some information about how to organize itself. This information is built into the network topology and learning rules.

An unsupervised learning algorithm might emphasize cooperation among clusters of processing elements. In such a scheme, the clusters would work together. If some external input activated any node in the cluster, the cluster's activity as a whole could be increased. Likewise, if external input to nodes in the cluster was decreased, that could have an inhibitory effect on the entire cluster. Competition between processing elements could also form a basis for learning. Training of competitive clusters could amplify the responses of specific groups to specific stimuli. As such, it would associate those groups with each other and with a specific appropriate response. Normally, when competition for learning is in effect, only the weights belonging to the winning processing element will be updated.

2.9 Network Selection

Because all artificial neural networks are based on the concept of neurons, connections and transfer functions, there is a similarity between the different structures or architectures or neural networks. The majority of the variations stems from the various learning rules and how those rules modify a network's typical topology. The following sections outline some of the most common artificial neural networks. They are organized in very rough categories of application. these categories are not meant to be exclusive, they are merely meant to seperate out some of the confusion over networks architectures and their best matches to specific applications. Basically, most applications of neural networks fall into the follwing categories:

- prediction , classification, data association

Table I ANN Selection Table

Network Type	**Networks**	**Use for Network**
Prediction	• Back-propagation • Delta Bar Delta • Directed Random Search • Higher Order Neural Networks	Use input values to predict some output (e.g. pick the best stocks in the market, predict weather, identify people with cancer risks etc.)
Classification	• Learning Vector Quantization	Use input values to determine the classification (e.g. is the input the letter A, is the blob of video data a plane and what kind of plane is it
Data Association	• Hopfield • Boltzmann Machine • Hamming Network	Like Classification but it also recognizes data that contains errors (e.g. not only identify the characters that were scanned but identify when the scanner isn't working properly)

Table I shows the differences between these network categories and shows which of the more common network topologies belong to which primary category. this chart is intended as a guide and is not meant to be all inclusive. There are many other network derivations, this chart only includes the some of them. Some of these networks, which have been grouped by application, have been used to solve more than one type of problem. Feedforward back-propagation in particular has been used to solve almost all types of problems and indeed is the most popular one.

2.10 Applications of Artificial Neural Networks

Unlike traditional expert systems where a knowledge base and necessary rules have to be defined explicitly, neural networks do not need rules instead they generate rules by learning from shown examples. This makes ANNs general purpose classification tools to be used in pattern recognition and classification systems. Neural networks provide a closer approach to human perception and recognition than traditional computing. When inputs are noisy or incomplete neural networks can still produce reasonable results. Neural networks are used successfully in the following areas.Language Processing (Text-to-speech and Speech-to-text applications),Data compression, Security, Image Recognition, Optical Character Recognition, Texture Detection and Segmentation, Handwriting recognition, Target classification, Industrial inspection, Optimization problems such as travelling salesman problem, Signal processing (prediction, system modeling, noise filtering etc.), Financial and Economic Modeling ,Control Systems ,Servo Control.

There are other areas in which neural networks might be applied successfully. They might include intelligent e-commerce applications in which customer buying intentions are recognized from various interactions of the user with the web site.

III. REALIZED ARTIFICIAL NEURAL NETWORK SIMULATOR

Realized ANN simulator in this study is composed of three structure. Firstly, perceptron which includes one and two inputs are realized. Here,input values and activation functions (hardlim, hardlims) of perceptron can changeable and due to this situations output values are examined. Such a perceptron is shown in figure 7.

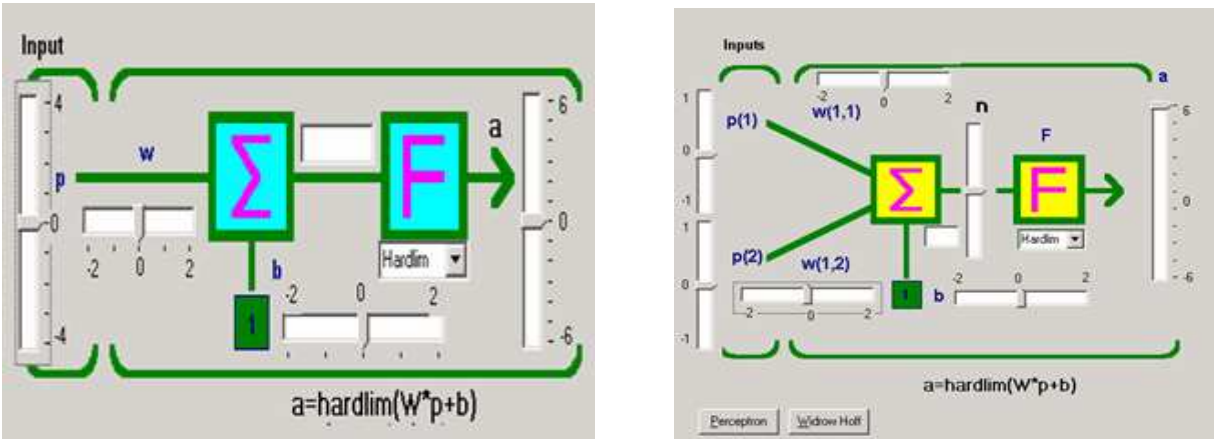

Figure 7 Simulation of One, Two Inputs Perceptron User Screen

Secondly, an ANN model which has one layer and multi-neron is considered. In this model, a simulatör is designed to realize Hebbian, Delta, Perception and Widrow Half Learning algorithms. In this simulator, learning algorithms, neuron numbers, neuron inputs numbers and activation functions can selected. Such a structure is shown in Figure 8.İnitial weight and bias values can be set randomly. When user create input and destination values, weight and output matrix modification are seen in each step. Diferantial error between destination and output matrix can be examined graphically.

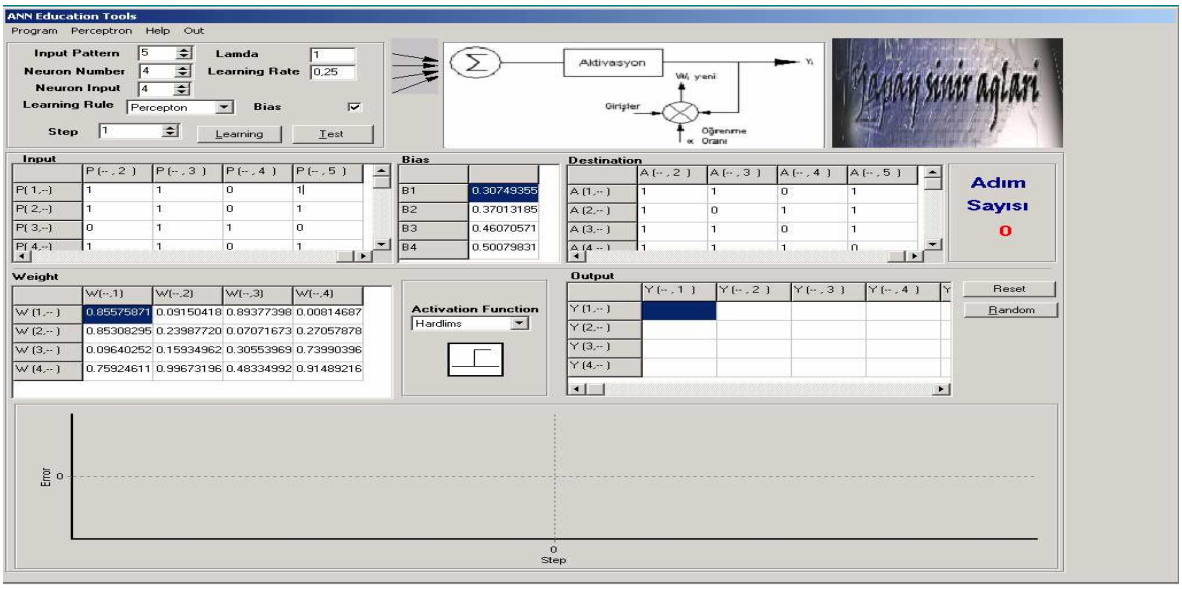

Figure 8 Simulation Of Different Learning Rules User Screen.

Thirdly, an ANN which has three layer is realized. In this structure, input numbers, neuron numbers of each layer, pattern numbers and activation functions (Purelin, Tansig, logsig),Learning rate can be selected. Such a structure is shown in Figure 9.Initialize values of weights can be assigned randomly. Input matrices-destination matrices value can be read from text file. Result weight matrices, output matrice also can be saved to the text file. By way of learning ANN, realized software supported with graphics interface. In addition to this, ANN learning phases are examined graphically and its has help file for ANN .

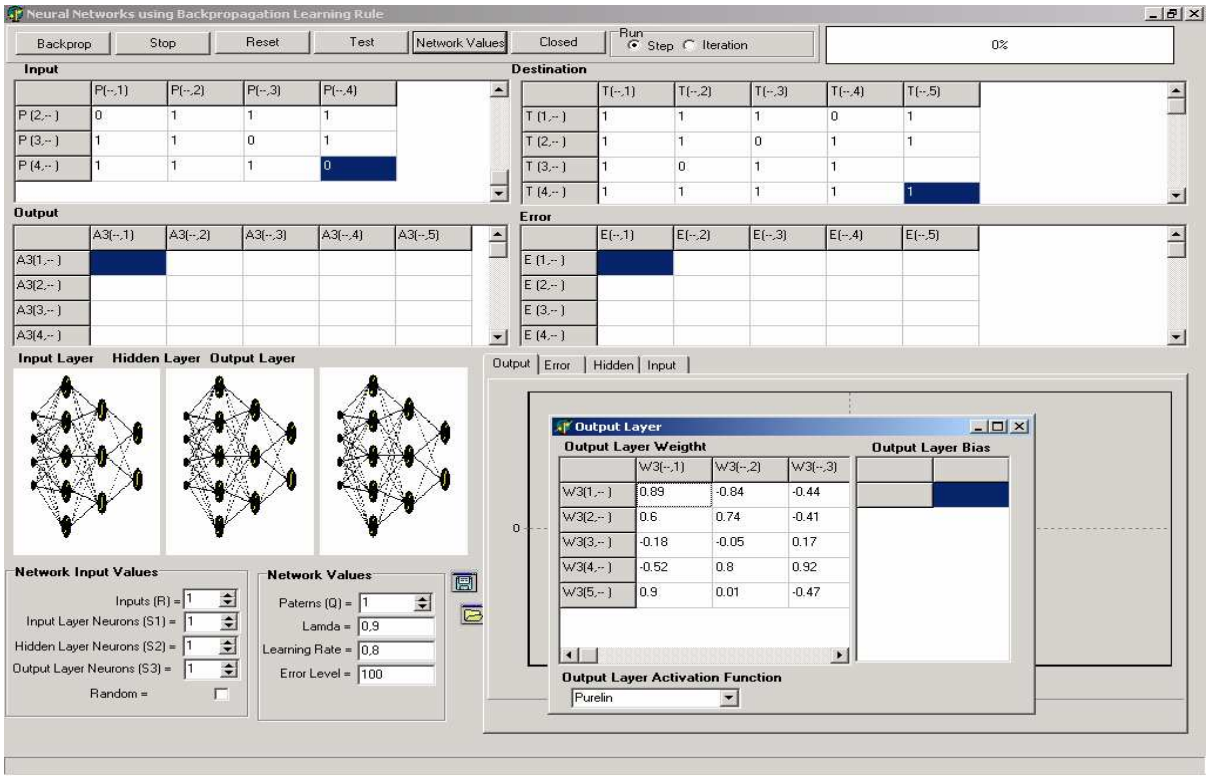

Figure 9 Simulation of Backpropagation Algorithms User Screen

In this study, we considered ANN and realized ANN simulator which help learning process and explained this process according to the inputs, weights, activation functions, learning types and outputs.

IV.CONCLUSION

As mentioned in this study, ANN is the most applicability AI technics in real world. Especially, learning process in technology education is effective,if it supports simulation tools. Education which based on simulator is the most factor effecting learning and understanding.Thus, ANN simulator which realized in this study is increased ANN education visually. That's way, ANN has to be considered educationally. Aim of the this, ANN Simulator used in graduate level class in university. When we compared previous lesson via to success, perception and learn, we observed that using that ANN simulator is useful educationally.

REFERENCES

1. Haykin Simon, "Neural Networks", 1994 Macmillan College Publishing Company Inc. ISBN 0-02-352761-7

2. http://www.dacs.dtic.mil/techs/neural/neural.title.html

3. Data & Analysis Center for Software, "Artificial Neural Networks Technology", 1992 (http://www.dacs.dtic.mil/techs/neural/neural.title.html, printed November 1998)

4. Neural and Adaptive Systems: Fundamentals Through Simulations, Jose Principe, Neil R. Euliano, and W. Curt Lefebre, John Wiley 2000.

5. Neural Network Design, Hagan, demuth, and Beale, PWS Publishing Company, 1996.

6. Mohammed H. Hassoun,*Fundamentals of Artificial Neural Networks,* MIT Press (1995).

7.Block, H.D. 1962. *The Perceptron: A Model for Brain Functioning, I*. Reviews of Modern Physics, Vol 34, pp 123-135. Reprinted in Anderson and Rosenfeld,1988, pp 138-150

12. Callan, R. (1999) *The Essence of Neural Networks*, Prentice Hall, ISBN 0-13-908732-X

13. Davalo, E., Naim, P. (1991). *Neural Networks*, The Macmillan Press Ltd, ISBN 0-333-54996-1

14. Fausett, L. 1994. *Fundamentals of Neural Networks : Architectures, Algorithms and Applications*. Prentice-Hall, ISBN 0-13-103805-2

15. Hebb, D.O. 1949. *The Organization of Behavior*. New York: John Wiley & Sons. Introduction and Chapter 4 reprinted in Anderson & Rosenfeld, 1988, pp 45-56

16. Le Cun, Y. 1986. *Learning Processes in an Asymmetric Theshold Network*. Disordered Systems and Biological Organization (Bienenstock, E., Fogelman-Smith, F., Weisbuch, G. (eds)), NATO ASI Series, F20, Berlin: Springer-Verlag

17. McCulloch, W.S., Pitts, W. 1943. *A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics*, Vol 5, pp 115-133. Reprinted in Anderson & Rosenfeld, 1988, pp 18-28.

18. Minsky, M.L., Papert, S.A. 1988. *Perceptrons, Expanded Edition*. Cambridge, MA: MIT Press. Original Edition, 1969.

23.http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

24.http://www.dacs.dtic.mil/techs/neural/neural10.html

25.http://www.generation5.org/nnintro.shtml