Bridging Neural Networks and Fuzzy Logic in Order to Implement a Real World of Automation.

Pradeep.A.V*, Sharmili.S**
*(Department of Mechanical engineering, GITAM UNIVERSITY, Visakhapatnam-26)
** (Department of Computer Science, St.Joseph's College, Visakhapatnam-26)

ABSTRACT

This paper investigates the benefits of combining neural networks and fuzzy logic into neuro-fuzzy systems, especially for applications in character recognition tasks. In the field of artificial intelligence, **neuro-fuzzy** refers to combinations of artificial neural networks and fuzzy logic. Neuro-fuzzy hybridization results in a hybrid intelligent system that synergizes these two techniques by combining the human-like reasoning style of fuzzy systems with the learning and connectionist structure of neural networks. Neuro-fuzzy hybridization is widely termed as Fuzzy Neural Network (FNN) or Neuro-Fuzzy System (NFS) in the literature. Neuro-fuzzy system incorporates the human-like reasoning style of fuzzy systems through the use of fuzzy sets and a linguistic model consisting of a set of IF-THEN fuzzy rules. The main strength of neuro-fuzzy systems is that they are universal approximators with the ability to solicit interpretable IF-THEN rules.

Keywords - artificial intelligence, fuzzy logic, neural networks

1. INTRODUCTION

Many task which seem simple for us, such as reading a handwritten note or recognizing a face, are difficult task for even the most advanced computer. In an effort to increase the computer ability to perform such task, programmers began designing software to act more like the human brain, with its neurons and synaptic connections. Thus the field of "Artificial neural network" was born. Rather than employ the traditional method of one central processor to carry out many instructions one at a time, the Artificial neural network software analyzes data by passing it through several simulated processes which are interconnected with synaptic like "weight" Once we have collected several record of the data we wish to analyze, the network will run through them and "learn " the input of each record may be related to the result. After training on a few cases the network begin to organize and refines its on own architecture to feed the data to much the human brain; learns from example.

Why would anyone want a `new' sort of computer?

What are (everyday) computer systems good at... .....and not so good at?

Good at	**Not so good at**
Fast arithmetic	Interacting with noisy data or data from the environment
Doing precisely what the programmer programs them to do	Massive parallelism
	Massive parallelism
	Fault tolerance
	Adapting to circumstances

Where can neural network systems help?

- where we can't formulate an algorithmic solution.
- where we *can* get lots of examples of the behavior we require.
- where we need to pick out the structure from existing data.

What is a neural network?

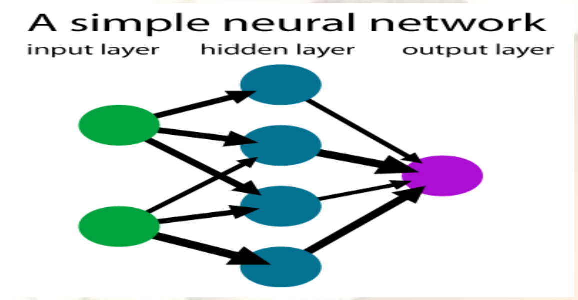
Neural networks are a form of multiprocessor computer system, with

- Simple processing elements
- A high degree of interconnection
- Simple scalar messages
- Adaptive interaction between elements

Artificial neural network (ANNs) are programs designed to solve any problem by trying to mimic structure and function of our nervous system. Neural network are based on simulated neurons which are joined together in a variety of ways to form networks. Neural network resembles the human brain in the following two ways: -

- A neural network acquires knowledge through learning
- A neural network's knowledge is stored with in the interconnection strengths known as synaptic weight.

Neural network are typically organized in layers. Layers are made up of a number of interconnected 'nodes', which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is output as shown in the graphic below.

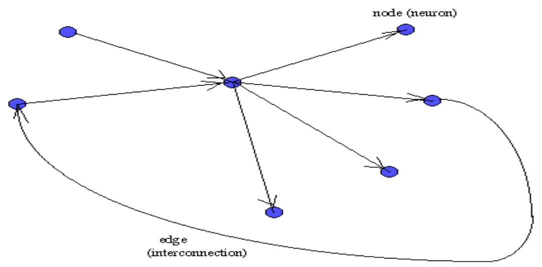


A neural network is a system that emulates the cognitive abilities of the brain by establishing recognition of particular inputs and producing the appropriate output. Neural networks are not "hard-wired" in particular way; they are trained using presented inputs to establish their own internal weights and relationships guided by feedback. Neural networks are free to form their own internal working and adapt on their own.

Commonly neural network are adjusted, or trained so that a particular input leads to a specific target output. There, the network is adjusted based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are used to train network. Once a neural network is 'trained' to a satisfactory level it may be used as an analytical tool on other data. To do this, the user no longer specifies any training runs and instead allows the network to work in forward propagation mode only. New inputs are presented to the input pattern where they filter into and are processed by the middle layers as though training were taking place, however, at this point the output is retained and no back propagation occurs.

2. THE STRUCTURE OF NERVOUS SYSTEM

Nervous system of a human brain consists of neurons, which are interconnected to each other in a rather complex way. Each neuron can be thought of as a node and interconnection between them are edge, which has a weight associates with it, which represents how mach the tow neuron which are connected by it can it interact.



The structure of Nervous system

2.1. Functioning of A Nervous System

The natures of interconnection between 2 neurons can be such that – one neuron can either stimulate or inhibit the other. An interaction can take place only if there is an edge between 2 neurons. If neuron A is connected to neuron B as below with a weight w, then if A is stimulated sufficiently, it sends a signal to B. The signal depends on the weight w, and the nature of the signal, whether it is stimulating or inhibiting.

$$\begin{tikzcd} \underset{A}{\bigcirc} \arrow[rr, "w"] & & \underset{B}{\bigcirc} \end{tikzcd}$$

This depends on whether w is positive or negative if its stimulation is more than its threshold. Also if it sends a signal, it will send it to all nodes to which it is connected. The threshold for different neurons may be different. If many neurons send signal to A, the combined stimulus may be more than the threshold. Next if B is stimulated sufficiently, it may trigger a signal to all neurons to which it is connected. Depending on the complexity of the structure, the overall functioning may be very complex but the functioning of individual neurons is as simple as this. Because of this we may dare to try to simulate this using software or even special purpose hardware.

3. Major components Of Artificial Neuron

This section describes the seven major components, which make up an artificial neuron. These components are valid whether the neuron is used for input, output, or is in one of the hidden layers.

3.1. Weighing factors

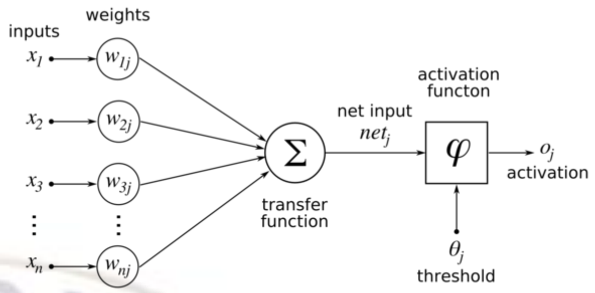
A neuron usually receives many simultaneous inputs. Each input has its own relative weight, which gives the input the impact that it needs on the processing elements summation function. These weights perform the same type of function, as do the varying synaptic strengths of biological neurons. In both cases, some input are made more important than others so that they have a greater effect on the processing element as they combine to produce a neuron response. Weights are adaptive coefficients within the network that determine the intensity of the input signal as registered by the artificial neuron. They are a measure of an input's connection strength. These strengths can be modified in response to various training sets and according to a network's specific topology or through its learning rules.

3.2. Summation Function

The first step in a processing element's operation is to compute the weighted sum of all of the inputs. Mathematically, the inputs and the corresponding weights are vectors which can be represented as (i1, i2,…………in) and (w1, w2,……………….wn). The total input signal is the dot, or inner, product of these two vectors. This simplistic summation function is found by multiplying each component of the ith vector by the corresponding component of the w vector and then adding up all the products. Input1 = i1*w1, input2=i2*w2, etc., are added as input1+input2+……….+input *n*. The result is a single number, not a multi-element vector. Geometrically, the inner product of two vectors can be considered a measure of their similarity. If the vector point in the same direction, the inner product is maximum; if the vectors points in opposite direction (180 degrees out of phase), their inner product is minimum. The summation function can be more complex than just the simple input and weight sum of products. The input and weighting coefficients can be combined in many different ways before passing on to the transfer function. In addition to a simple product summing, the summation function can select the minimum, maximum, majority, product, or several normalizing algorithms. The specific algorithm for combining neural inputs is determined by the chosen network architecture and paradigm.

3.3. Transfer Function

The result of the summation function, almost always the weighted sum, is transformed to a working out put through an algorithm process known as the transfer function. In transfer function the summation total can be compared with some threshold to determine the neural output. If the sum is greater than the threshold value, the processing element generates a signal. If the sum of the input and weight product is less than the threshold, no signal (no some inhibitory signal) is generated. Both types of response are significant.

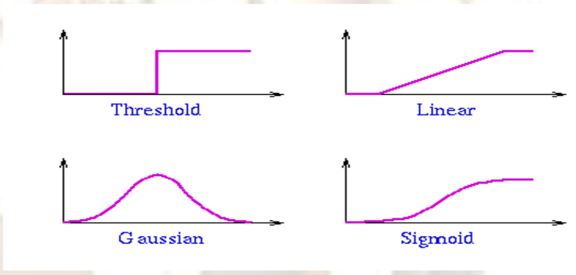


McCulloch-Pitts Neuron Model

Types of threshold functions

Sigmoid function	: f (x)=1/(1+exp(-x))
Step function	: f (x) =0 if x<T
	=k if x>=T
Ramp function	: f (x)=ax+b

The transfer function could be something as simple as depending upon whether the result of the summation function is positive or negative. The network could output zero and one, and minus one, or other numeric combinations. The transfer function would then be a "hard limiter" or step function.



Sample transfer functions

3.4. Scaling and Limiting

After the processing element's transfer function, the result can pass through additional processes which scale and limit. This scaling simply multiplies a scale factor times the transfer value, and then adds an offset. Limiting is the mechanism, which ensures that the scaled result does not exceed, and upper or lower bound. This limiting is in addition to the hard limits that the original transfer function may have performed.

3.5. Output function (competition)

At Each processing element is allowed one output signal, which it may output to hundreds of other neurons. This is the just like the biological neuron, where there are many inputs and only one output action. Normally, the output is directly equivalent to the transfer function's result. Some network topologies, however, modify the transfer result to incorporate competition among neighboring processing elements. Neurons are allowed to compete with each other, inhibiting processing elements unless they have great strength. Competition can occur one or both of two levels. First, competition determines which artificial neuron will be active, are provides an output. Second, competitive inputs help determine which processing elements will participate in the learning or adaptation process.

3.6. Error function and back-propagated value

In most learning networks the difference between the current output and the desired output is calculated. This raw error is then transformed by the error function to match particular network architecture. The most basic architectures use this error directly, but some square the error while retaining its sign, some cube the error, and other paradigms modify the raw error to fit their specific purposes. The artificial neuron's error is then typically propagated into the learning function of another processing element. This error term is sometimes called the current error. The current error is typically propagated backwards to a previous layer. Yet, this back-propagated value can be either the current error, the current error scaled in some manner (obtained by the derivative of the transfer function), or some desired output depending on the network type. Normally, this back-propagated value, after being scaled by the learning function, is multiplied against each of the incoming connection weights to modify them before the next learning cycle.

3.7. Learning function

The purpose of the learning function is to modify the variable connection weights on the inputs of each processing elements according to some neural base algorithm. This process of changing the weights of the inputs connections to achieve some desired results could also be called the adoption function, as well as the learning mode.

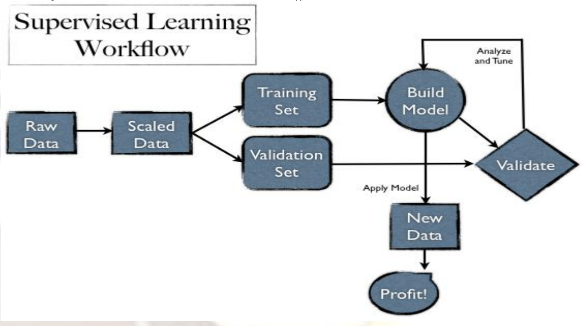
3.7.1. Paradigms of learning

There are three broad paradigms of learning:

- Supervised
- Unsupervised (or self-organized)
- Reinforcement learning (a special case of supervised learning)

3.7.1.1 Supervised learning

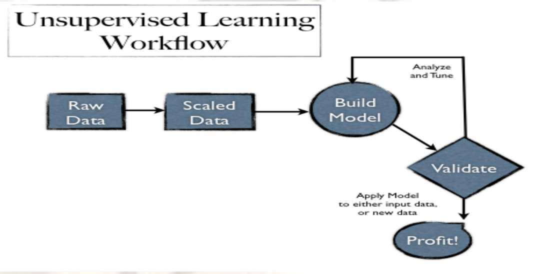
The vast majority of the artificial neural network solutions have been trained with supervision. In this mode, the actual output of a neural network is compared to the desired output. The network then adjusts weights, which are usually randomly set to begin with, so that the next iteration, or cycle, will produce a closer match between the desired and the actual output. This 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 acceptable network accuracy is reached.



Block dig. Of supervised learning

3.7.1.2. Unsupervised learning

In supervised learning the system directly compares the network output with a known correct or desired answer, whereas in unsupervised learning the output is not known. Unsupervised training allows the neurons to compete with each other until winner emerges. The resulting values of the winner neurons determine the class to which a particular data set belongs. Unsupervised learning is the great promise of the future. It shouts that computers could some day learn on their own in a true robotic sense.


Unsupervised Learning Workflow
Raw Data
Scaled Data
Build Model
Analyze and Tune
Validate
Apply Model to either input data, or new data
Profit!

Block dig. Of unsupervised learning

3.7.1.3 Reinforcement learning

Reinforcement learning is a form of supervised learning where adopted neuron receives feedback from the environment that directly influences learning.

3.7.2. Learning law

The following general learning rules is adopted in the neural network studies:

The weight vector w_i =[W_{i1}, W_{i2}……………………………….W_{in}] ^{t} increases proportion to the product of input x and learning signal r. The learning signal r is a function of w_i x, and sometimes of the teacher's signal d_i.

Hence we have, r =r (W_i, X, d_i) and increment in weight vector produced by the learning step at time t is ΔW_i (t) =cr [W_i (t), X (t), d_i (t)] X (t)

Where c is learning constant.

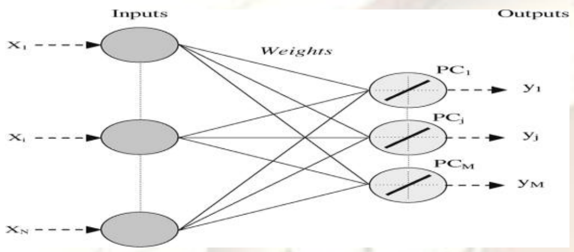
Thus W_i (t+1) =W_i (t)+ cr [W_i (t), X (t), d_i (t)] X (t)

Various learning rules are their assist the learning process.

They are :

3.7.2.1 Hebbian learning rule

This rule represents purely feed forward, unsupervised learning



According to this rule, we have

$r=f(W^t_i, X)$

and increment of weight becomes

$\Delta W_i=c\, f(W^t_i, X)\, X$

3.7.2.2 Perceptron learning rule

This learning is supervised and learning signal is equal to

$r = d_i - O_i$

Where $O_i=\mathrm{sgn}(W^t_i, X)$ and d_i is the desired response.

Weight adjustment in this method is

$\Delta W_i=c[d_i - \mathrm{sgn}(W^t_i, X)]\, X$

in this method of learning, initial weight can have any value and neuron much be binary bipolar are binary unipolar.

3.7.2.3. Delta learning rule

This rule is valid for continuous activation functions and in the supervised training mode. The learning signal is called as delta and is given as:

$r=[d_i - f(W^t_i X)]\, f'(W^t_i X)$

The adjustment for the single weight in this rule is given as:

$\Delta W_i = c\, (d_i - O_i)\, f'(\text{net } i)\, X$

In this method of learning, the initial weight can have any value and the neuron must be continuous.

3.7.2.4. Windrow-Hoff learning rule

This is applicable for the supervised training of neural networks and is independent of activation function.

Learning signal is given as

$r = d_i - W^t_i X$

The weight vector increment under this learning rule is :

$\Delta W_i=c\, (d_i - W^t_i X)\, X$

3.7.2.5. Correlation learning rule

Substituting r = d_i in general learning rule, we obtain correlation learning rule. The adjustment for the weight vector is given by:

$\Delta W_i = c\, d_i\, X$

3.7.2.6. Winner-take all learning rule

This rule is applicable for an ensemble of neurons, let's say being arranged in a layer of p units. This learning is base on the premise that one of the neurons in the layer, say the m^{th}, has the maximum response due to input x. This neuron is declared the winner. It's increment is computed as follows

$\Delta W_m = \alpha\, (X - W_m)$

3.7.2.7. outstar learning rule

This is an another learning rule that is best explained when the neurons are arranged in layers. This rule is designed to produce a desired response d of the layer of p neurons. This rule is concerned with the supervised learning and the weight adjustment is computed as:

$\Delta W_j = \beta\, (X - W_j)$

4. Training a neural network

Since the output of the neural network may not be what is expected, the network needs to be trained. Training involves altering the interconnection weights between the neurons. A criterion is needed to specify when to change the weights and how to change them. Training is an external process whiling learning is the process that takes place internal to the network. The following guideline will be of help as a step methodology for training a network.

4.1. Choosing the number of neurons

The number of hidden neurons affect how well the network is able to separate the data. A large number of hidden neurons will ensure correct learning and the network is able to correctly predict the data it has been trained on, but its performance on new data, its ability to generalize, is compromised. With too few hidden neurons, the network may be unable to learn the relationship amongst the data and the error will fail to fall below an acceptable level. Thus, selection of the number of hidden neurons is a crucial decision. Often a trial and error approach is taken starting with a modest number of hidden neurons and gradually increasing the number if the network fails to reduce its error. A much used approximation for the number of hidden neurons for a three layered network is N=1/2(j + k)+v P, where J and K are the number of input neurons and P is the number of patterns in the training set.

4.2. Choosing the initial weights

The learning algorithm uses steepest descent technique, which rolls straight downhill in weight space until the first valley is reached. This valley may not correspond to a zero error for the resulting network. This makes the choice of initial starting point in the multidimensional weight space critical. However, there are no recommended rules for this selection except trying several different weight values to see if the network results are improved.

4.3. Choosing the learning rate

Learning rate effectively controls the size of the step that is taken in multidimensional weight space when each weight is modified. If the selected learning rate is too large then the local minimum may be over stopped constantly, resulting in oscillations and slow convergence to lower error state. If the learning rate is too low, the number of iterations requires may be too large, resulting in slow performance. Usually the default value of most commercial neural network packages are in the range 0.1-0.3 providing a satisfactory balance between the two reducing the learning rate may help improve convergence to a local minimum of the error surface.

4.4. Choosing the activation function

The learning signal is a function of the error multiplied by the gradient of the activation function df/d (net). The larger the value of the gradient, the more weight the learning will receive. For training, the activation function must be monotonically increasing from left to right, differentiable and smooth.

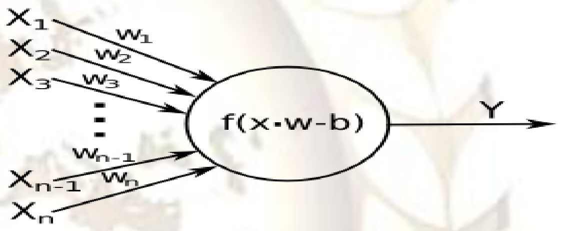
5. Models Of artificial Neural Networks

There are different kinds of neural network models that can be used. Some of the common ones are:

5.1. Perception model

This is a very simple model and consists of a single 'trainable' neuron. Trainable means that its threshold and input each input has a desired output (determined by us). If the neuron doesn't gives the desired output, then it has made a mistake. To rectify this, its threshold and/or input weights must be changed. How this change is to be calculated is determined by the learning algorithm.

The output of the perceptron is constrained to Boolean values – (true, false), (1, 0), (1, -1) or whatever. This is not a limitation because if the output of the perceptron were to be the input for something else, then the output edge could be made to have a weight. Then the output would be dependant on this weight.The perceptron looks like



X_1, X_2, …………., X_n are inputs. These could be real numbers or Boolean values depending on the problem.

y is the output and is Boolean.

w_1, w_2, …………, w_n are weights of the edges and are real valued.

T is the threshold and is a real valued.

The output y is 1 if the net input which is :

$w_1 x_1 + w_2 x_2 + \ldots\ldots + w_n x_n$

is greater than the threshold T. Otherwise the output is zero.

5.2. Feed – Forward Model

Elementary feed forward architecture of m neurons and receiving n inputs is shown in the figure. Its output and input vectors are respectfully.

$$O = [O] \quad O2 \ldots\ldots\ldots O_m]$$

$$X = [x] \quad x2 \ldots\ldots\ldots x_n]$$

Weight Wij connects the i^{th} neuron with the j^{th} input. Hence activation value net i for the i^{th} neuron is

$$\text{Net } i = {}_{j=1}\Sigma^{n} \text{ Wij Xj}$$

for i=1, 2, 3,……, n

Hence, the output is given as

$$Oi = f(W^{t}{}_{i}\, X) \qquad \text{for } i = 1, 2, 3, \ldots\ldots\ldots, m$$

Where Wi is the weight vector containing the weights leading towards
the ith output node and defined as

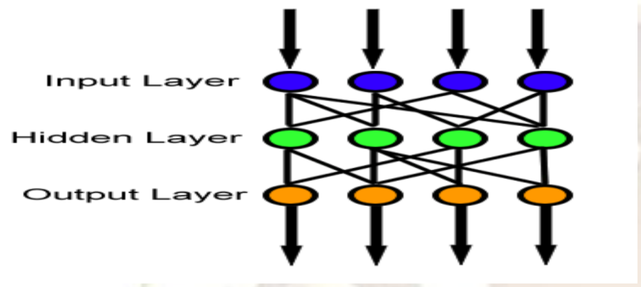
$$Wi = [Wi1 \;\; Wi2 \;\ldots\ldots\ldots\ldots\; Win]$$

If Γ is the nonlinear matrix operarte, the mapping of input space X to output space O implemented by the network can be expressed

$$O = \Gamma\,|W X|$$

Where W is the weight matrix, also called the connection matrix.

The generic feedforward network is characterized by the lack of feedback. This type network can be connected in cascade to create a multiplayer network.

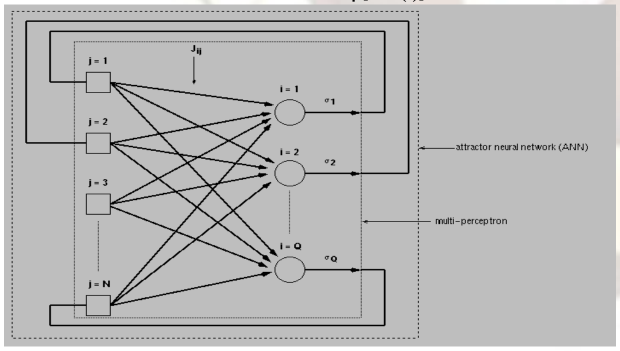


feed forward neural network

5.3. Feed-Back model

A feedback network can be obtained from the feedforawrd network by connecting the neuron's outputs to their inputs, as shown in the fig. The essence of closing the feedback loop is to hold control of output O_i through outputs O_j; for j =1, 2, ……, m. or controlling the output at instant t+Δ by the output at instant t. This delay Δ is introduced by the delay element in the feedback loop. The mapping of O(t) into O(t+ Δ) can now be written as

$$O(t+\Delta) = \Gamma[W\,o(t)]$$



5.4. Notations Used

M_1 is a 2-D matrix where $M_1[i]\,[j]$ represents the weight on the connection from the ith input neuron to the jth neuron in the hidden layer.

M_2 is a 2 –D matrix where $M_2[i][j]$ denotes the weight on the connection from the ith hidden layer neuron to the jth output layer neuron.

x, y and z are used to denote the outputs of neurons from the input layer, hidden layer and output layer respectively.

If there are m input data, then $(x_1, x_2, \ldots\ldots, x_m)$. P denotes the desired output pattern with components $(p_1, p_2, \ldots\ldots, p_r)$ for r outputs.

Let the number of hidden layer neurons be n.

β_o = learning parameter for the output layer.

β_h = learning parameter for the hidden layer.

α = momentum term

θ_j = threshold value (bias) for the jth hidden layer neuron.

τ_j = threshold value for the jth output layer neuron.

e_j = error at the jth output layer neuron.

t_j = error at the jth hidden layer neuron.

Threshold function = sigmoid function : F(x) = 1/(1 + exp(x)).

5.5 Mathematical expressions

Output of jth hidden layer neuron : $y_j = f((\Sigma_i x_i M_1[i][j]) + \theta_j)$

Output of jth output layer neuron : $z_j = f((\Sigma_i y_i M_2[i][j]) + \tau_j)$.

ith component of vector of output differences:

desired value – computed value = $P_j - z_j$

ith component of output error at the output layer:

$$e_j = p_j - z_j.$$

ith component of output error at the hidden layer:

$$t_i = y_i\,(1 - y_i)\,(\Sigma_j M_2[i][j]\,e_j)$$

Adjustment of weights between the ith neuron in the hidden layer and jth output neuron:

$$\Delta M_2[i][j]\,(t) = \beta_0\, y_i\, e_j + \alpha\,\Delta M_2\,[i][j]\,(t - 1)$$

Adjustment of weights between the ith input neuron and jth neuron in the hidden layer :

$$\Delta M_1[i][j] \;= \beta_h\, x_i\, t_j + \alpha\,\Delta M_1\,[i][j]\,(t - 1)$$

Adjustment of the threshold value for the jth output neuron:

$$\Delta\,\tau_j = \beta_0\, e_j$$

Adjustment of the threshold value for the jth hidden layer neuron:

$$\Delta\,\theta_j = \beta_h\, e_j$$

6. Neural network applications

Aerospace

- High performance aircraft autopilot, flight path simulation, aircraft control systems, autopilot enhancements, aircraft component simulation, aircraft component fault detection.

Automobile control

- Automobile automatic guidance system, warranty activity analysis.

Banking

- Check and other document reading credit application evaluation.

Credit card activity checking

- Neural networks are used to spot unusual credit card activity that might possibly be associated with loss of a credit card

Defense

- Weapon steering, target tracking, object discrimination, facial recognition, new kinds of sensors, sonar, radar and image signal processing including data compression, feature extraction and noise suppression, signal/ image identification.

Electronics

- Code sequence prediction, integrated circuit chip laying, process control, chip failure analysis, machine vision voice synthesis, nonlinear modeling.

Entertainment

- Animation, special effects, market forecasting.

Financial

- Real estate appraisals, loan advisor, mortgage screening, corporate bond rating, credit-line use analysis, and portfolio trading program, corporate financial analysis, and currency price prediction.

Industrial

- Neural networks are being trained to predict the output gasses of furnaces and other industrial process. They then replace complex and costly equipment used for this purpose in the past.

Insurance

- Neural networks are used in policy application evaluation, product optimization.

Manufacturing

- Neural networks are used in manufacturing process control, product design and analysis, process and machine diagnosis, real-time particle identification, visual quality analysis, paper quality prediction, computer – chip quality analysis, analysis of grinding operations, chemical product design analysis, machine maintenance analysis, project bidding, planning and management, dynamic modeling of chemical process system.

Medical

- Neural networks are used in breast cancer cell analysis, EEG and ECG analysis, prosthesis design, optimization of transplant times, hospital expense reduction, hospital quality improvement, and emergency room test advisement.

Oil and Gas

- Neural networks are used in exploration of oil and gas.

Robotics

- Neural networks are used in trajectory control, forklift robot, manipulator controllers, vision systems

Other application

- Artificial intelligence
- Character recognition
- Image understanding
- Logistics
- Optimization
- Quality Control
- Visualization

7. Advantages Of ANN

1. It involve human like thinking.
2. They handle noisy or missing data.
3. They create their own relationship amongst information – no equations!
4. They can work with large number of variables or parameters.
5. They provide general solutions with good predictive accuracy.
6. System has got property of continuous learning.
7. They deal with the non-linearity in the world in which we live.

8. Conclusion

In this present world of automation to automate systems neural network and fuzzy logic systems are required. Fuzzy logic deals with vagueness or uncertainty and neural network related to human like thinking. If we use only fuzzy systems or only NN complete automation is never possible. The combination is suitable because fuzzy logic has tolerance for impression of data, while neural networks have tolerance for noisy data. As there are different advantages and disadvantages usage of NN thus we use combination of neural networks and Fuzzy logic in order to implement a real world system without manual interference. Very promising results have been obtained in the sense that both an improvement in the recognition rate and a reduction in the complexity of the models are achieved.

Bibliography

References books:–

- "Introduction to Artificial Neural Network", by Jacek M. Zurada; Jaico Publishing House, 1999.
- "An Introduction to Neural Network", by James A. Anderson; PHI, 1999.
- "Elements of Artificial Neural Network", K. Mehrotra, C.K. Mohan and Sanjay ranka, MIT Press, 1997.
- "Neural Network nad Fuzzy System", by Bart Kosko; PHI, 1992.
- "Neural Network – A comprehensive foundation", Simon Haykin, Macmillan Publishing Co., Newyork, 1993.

Reference sites –

- http://www.cs.stir.ac.uk/~lss/NNInro/invSlides.htm
- http://www.bitstar.com/nnet.htm
- http://www.pmsi.fr/sxcxmpa.htm
- http://www.pmsi.fr/neurinia.htm