Face Features Recognition Using Soft Computing Method

Dr. Vijay Pathak[1], Mr.Deepak[2]
[1]Institute of Computer Science&Technology, Varanasi
[2]Research Scholar, Uttarakhand Technical University, Dehradun, U.K.
[1]vijayshepa@gmail.com
[2]dpky85@gmail.com

Abstract—**Evolutionary computing method isa computing paradigm that originated in the biological word. There various techniques used by various researchers to recognize the face. In this paper we recognized the face using Soft computing method.**

1 NEURAL COMPUTING

Artificial Neural Networks (ANN) is a computing paradigm that originated in the biological world. Neural Computation does not have to be the computation carried out by nerve cells. An artificial system can emulate a simplified version of a neural computational system. ANN is an example of such an artificial neural system. Even though the name ANN has been the most common but other names have been used synonymously as well. Examples of these names are Neural Computing, Connectionism, Parallel Distributed Processing, and Connection Science [1].

The multidisciplinary nature of the field of neural networks and its origin in biological science makes it difficult to state a rigorous definition for the field and what it addresses. This is the same problem with Evolutional and Genetic Computing. However, few references have attempted such a definition. A definition given by Igor Aleksander and Helen Morton is given as follows. *"Neural computing is the study of networks of adaptable nodes which, through a process of learning from task examples, store experiential knowledge and make it available for use"* [1].

ANNs have often been used as an alternative to the techniques of standard nonlinear regression and cluster analysis to carry out statistical analysis and data modeling. In addition, computer scientists and engineers have seen ANNs, as providing a new experimental paradigm for Parallel

Distributed Processing, rather than the algorithmic paradigm that dominated the field of machine intelligence prior to the ANN revolution [2].

Although scientists from various fields worked on the study of understanding and modeling of neuro-sciences, ANNs were actually realized in the 1940s. Warren McCulloch and Walter Pitts designed the first ANNs [3]. The first learning rule for ANNs was designed by Donald Hebb in McGill University [4]. In the 1950s and 1960s, ANNs entered their first flowering era. The most remarkable implementations of that era were the development of the Perceptronsand the ADALINE algorithm. After that, there was a rather quiet period in the 1970s, regardless of the works of Kohonen, Anderson, Grossberg, and Carpenter. The 1980s witnessed the second revival of ANNs. Back-Propagation, Hopfield Nets, Neocognitron, and Boltzmann Machinewere the most remarkable developments of that era [5].

An ANN is a computational structure designed to mimic biological neural networks. The ANN consists of

computational units called *neurons*, which are connected by means of weighted interconnections. The weight of an interconnection is a number that expresses the strength of the associated interconnection.

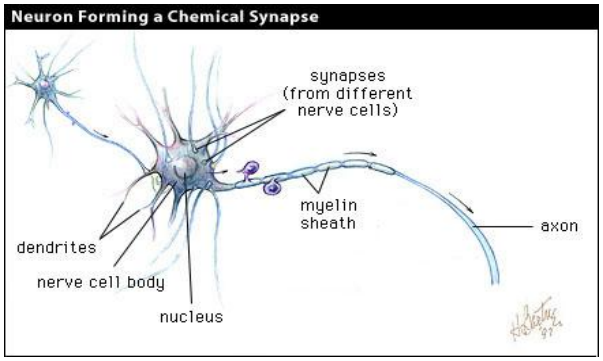
The main characteristic of ANNs is their ability to learn. The learning process is achieved by adjusting the weights of the interconnections according to some applied learning algorithms. Therefore, the basic attributes of ANNs can be classified into Architectural attributes and Neurodynamic attributes [6]. The architectural attributes define the network structure, i.e., number and topology of neurons and their interconnectivity. The neurodynamc attributes define the functionality of the ANN.

ANNs were developed in the 1960s after a series of developments, proposals, and implementations. The most remarkable foundational achievements are the work on Spontaneous Learning by Rosenbaltt in 1959 [7], Competitive Learning by Stark, Okajima, and Whipple in 1962 [8], and ADALINE/MADALINE algorithms by Widrow and Hoff in 1960 [9, 10]. However, it is important to note that modeling a neuron mathematically has been a research problem for over a hundred years [6].

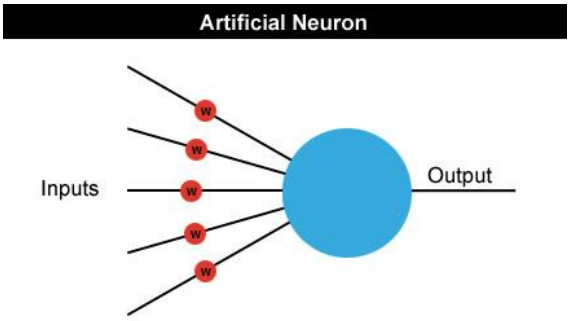
2. SO, WHAT EXACTLY A NEURAL NETWORK IS?

A neural network is mans crude means of making an attempt to simulate the brain electronically. therefore to grasp however a neural web works we tend to initial have a glance at however the previous grey substance will its business.

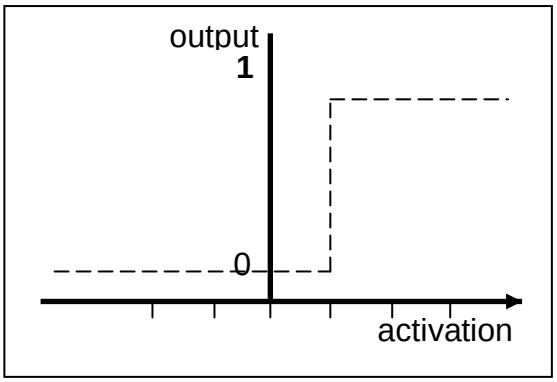
Our brains square measure created of regarding a hundred billion small units known as neurons. every vegetative cell is connected to thousands of different neurons and communicates with them via chemistry signals. Signals returning into the vegetative cell square measure received via junctions known as synapses, these successively square measure situated at the top of branches of the vegetative cell cell known as dendrites. The vegetative cell endlessly receives signals from these inputs so performs to a small degree little bit of magic. What the vegetative cell will (this is over simplified i would add) is total up the inputs to itself in how so, if the top result's larger than some threshold price, the vegetative cell fires. It generates a voltage and outputs a signal along something called an *axon*. Just have a good look at the illustration and try to picture what is happening within this simple little cell.



Neural networks are made up of many artificial neurons. An artificial nerve cell is solely associate electronically modelled biological nerve cell. what number neurons area unit used depends on the task at hand. It may well be as few as 3 or as several as many thousand. One optimistic man of science has even arduous wired a pair of million neurons along within the hope he will come back up with one thing as intelligent as a cat though the majority within the AI community doubt he are going to be thriving. There area unit many alternative ways that} of connecting artificial neurons along to make a neural network however I shall be concentrating on the foremost common which is termed a feed forward network.



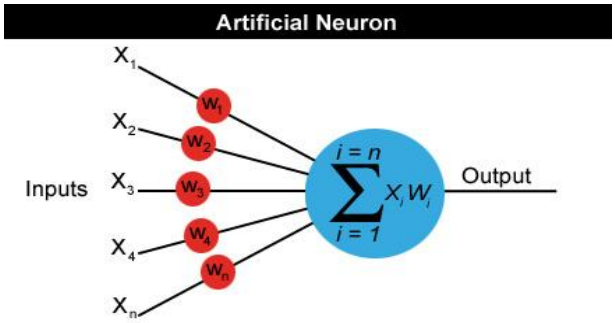
Each input into the somatic cell has its own weight related to it illustrated by the red circle. A we have a tendency toight is just a floating purpose range and it's these we alter once we eventually return to coach the network. The weights in most neural nets is each negative and positive, so providing repressive influences to every input. As every input enters the nucleus (blue circle) it's increased by its weight. The nucleus then sums of these new input values which supplies United States of America the activation (again a floating purpose range which might be negative or positive). If the activation is bigger than a threshold worth - lets use the quantity one as Associate in Nursing example - the somatic cell outputs a sign. If the activation is a smaller amount than one the somatic cell outputs zero. This is typically called a *step* function as shown in figure below:



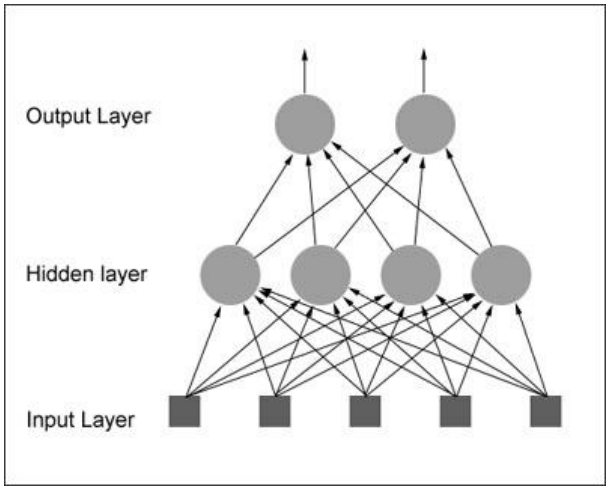
A neuron can have any number of inputs from one to n, where n is the total number of inputs. The inputs may be represented therefore as x_1, x_2, x_3… x_n. And the corresponding weights for the inputs as w_1, w_2, w_3… w_n. Now, the summation of the weights multiplied by the inputs we talked about above can be written as $x_1w_1 + x_2w_2 + x_3w_3 \ldots + x_nw_n$, . So, the activation value is

$$a = x_1w_1 + x_2w_2 + x_3w_3 \ldots + x_nw_n .$$

Fortunately there is a quick way of writing this down which uses the Greek capital letter sigma S, which is the symbol used by mathematicians to represent summation.



Well, we have to link several of these neurons up in some way. One way of doing this is by organising the neurons into a design called a *feed forward network*. It gets its name from the way the neurons in each layer feed their output forward to the next layer until we get the final output from the neural network. This is what a very simple feed forward network looks like:

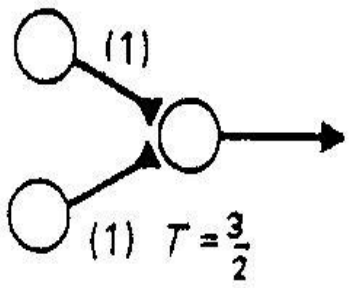


Each input is sent to every vegetative cell within the hidden layer then each hidden

layer's vegetative cell's output is connected to each neuron within the next layer. There will be any variety of hidden layers inside a feed forward network however one is typically enough to live up to for many issues you may tackle. conjointly the quantity of neurons I've chosen for the on top of diagram was utterly absolute. There will be any variety of neurons in every layer, it all depends on the matter.

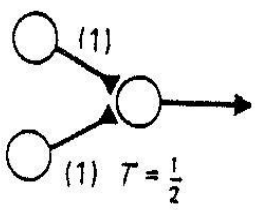
*3.*MODEL NEURONS: NEURODES

The building-block of computer-model neural networks may be a process unit known as a neurode, that captures several essential options of biological neurons. In the diagram, 3 neurodes square measure shown, which may perform the operation "AND",



ie, the output neurode can fireplace provided that the 2 input neurodes square measure each firing. The output neurode includes a "threshold" worth (T) of 3/2 (ie, 1.5). If neither or only 1 input neurode is firing, the overall input to the output neurode are going to be but one.5, and therefore the output neurode won't fireplace. However, if each input neurodes square measure firing, the overall input of one+1=2 are going to be bigger than the brink worth of 1.5, and therefore the output neurode can fireplace. Similarly, Associate in Nursing "OR" operation are often enforced exploitation an equivalent design, however dynamical the brink worth to zero.5. during this case, the output neurode fires provided that it receives input from either or each neurodes.

The values in parenthesis (1) on the connections between the neurodes square measure weights of the connections, adore the conjugation strength of vegetative cell connections. In biological neural networks the firing of a vegetative cell may result in varied amount of that vegetative cell. Imagine, for example, a neuron neurotransmitter released at the synapses of

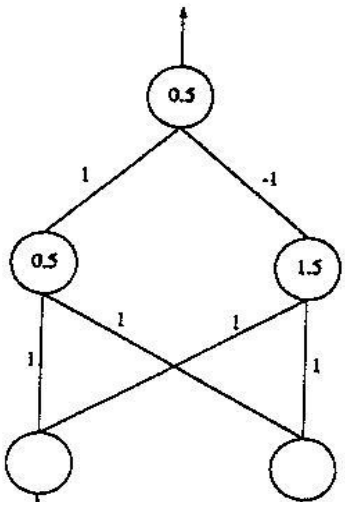


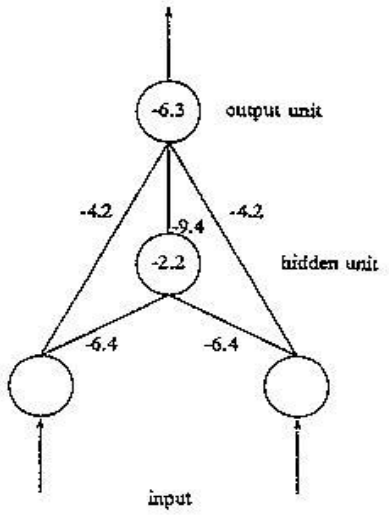
with 3 axons leading to 3 pre-synaptic terminals. One terminal releases neurotransmitter from 20 vesicles, another from 100 vesicles and the third from 900 vesicles. The synaptic strength (the **weight**) of the second terminal is 5 times as great as the first, everything else being equal. In the neurodes of computer models, weights tend to be values between -1 and +1. Notice that in the examples shown, the weights could have been (0.8) rather than (1) and the results would be the same.

Now consider a more complex network, one designed to do the logical operation "EXCLUSIVE-OR" (XOR). The threshold values are shown inside the neurode circles and the weights are shown alongside theconnections. Note the addition of a neurode (the **hidden neurode**) between the input and output neurodes. In an XOR operation, the output neurode only fires if one (but not both) of the input neurodes fire. In this case, the hidden neurode will not fire if only one input neurode fires. This will cause the output neurode to fire, since +1 is greater than the 0.5 threshold. But if both input neurodes fire, the result is a total input

of 1+1-2=0 to the output neurode. Since 0 is less than the 0.5 threshold of the output neurode, the output neurode will not fire.

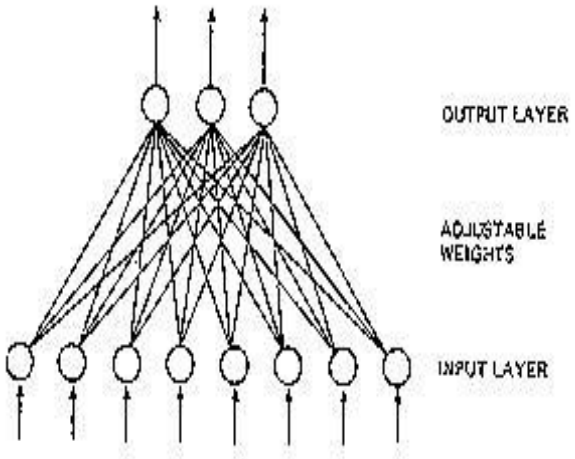
The





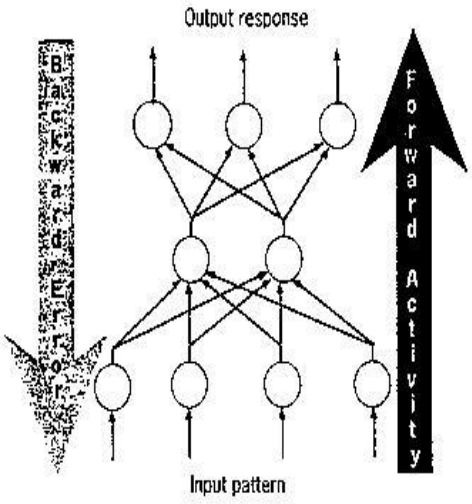
solution shown is not the only possible solution to the XOR problem in a simple neurode network. There are, in fact, infinitely many possible solutions. Two more example solutions are shown. Negative connection weights represent inhibitory rather than excitatory weights (synapses). Note that threshold values can also be less than zero.

In these examples the relationships between the thresholds, weights, inputs and outputs can be analyzed in detail. But in neural networks (both computer and biological) with large numbers of inputs, outputs and hidden neurodes (neurons), the task of determining weights and threshold values required to achieve desired outputs from given inputs becomes practically impossible. Computer models therefore attempt to **train** networks to adjust their weights to give desired outputs from given inputs. If biological memory and learning are the result of synapse strengths -- and modifications of synapse strengths -- then the computer models can be very instructive. Computer neural network models are described in terms of their architecture (patterns of connection) and in terms of the way they are trained (rules for modifying weights). I will therefore classify my descriptions into four categories: (1) Perceptrons& Backpropagation, (2) Competitive Learning, (3) Attractor Networks and (4) Other Neural Network Models.



4 PERCEPTRONS& BACKPROPAGATION

The architecture of a Perceptron consists of a single input layer of many neurodes, and a single output layer of many neurodes. The simple "networks" illustrated at the beginning, to produce logical "AND" and "OR" operations have a Perceptron architecture. But to be called a Perceptron, 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. For



A backpropagation network trains with a two-step procedure. The activity from the input pattern flows forward through the network, and the error signal flows backward to adjust the weights.

example, if the network illustrated gives a "0 1 0" output when "0 1 1" is the desired output for some input, all of the weights leading to the third neurode would be adjusted by some factor.

The Adaline is a modification of the Perceptron, which substitutes bipolar

(-1/+1) for binary (0/1) inputs, and adds "bias". But the most important modification is the use of a **delta learning rule**. As with the Perceptron, the delta rule compares desired output to actual output to computer weight adjustment. But the delta rule squares the errors and averages them to avoid negative errors cancelling-out positive ones. Adalines have been used to eliminate echoes in phone lines for nearly 30 years.

Neural network research went through many years of stagnation after Marvin Minsky and his colleague showed that Perceptrons could not solve problems such as the EXCLUSIVE-OR problem. Several modifications of the Perceptron model, however, produced the **Backpropagation** model -- a model which can solve XOR and many more difficult problems. Backpropagation has proven to be so powerful that it currently accounts for 80% of all neural network applications. In Backprop, a third neurode layer is added (the **hidden layer**) and the discrete thresholding function is replaced with a continuous (sigmoid) one. But the most important modification for Backprop is the **generalized delta rule**, which allows for adjustment of weights leading to the hidden layer neurodes in addition to the usual adjustments to the weights leading to the output layer neurodes. Using the generalize delta rule to adjust the weights leading to the hidden units is **backpropagating** the error-adjustment.

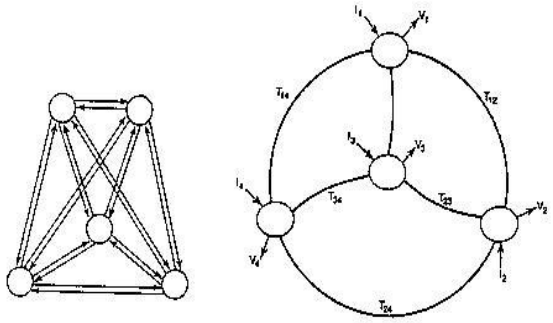
5 COMPETITIVE LEARNING

The prototypic competitive learning ("self-organizing") model is the Kohonen network (named after the Finnish researcher who pioneered the research). A Kohonen network is a two-layered network, much like the Perceptron. But the output layer for a two-neurode input layer can be represented as a two-dimensional grid, also known as the "competitive layer". The input values are continuous, typically normalized to any value between -1 and +1. Training of the Kohonen network does not involve comparing the actual output with a desired

output. Instead, the input vector is compared with the weight vectors leading to the competitive layer. The neurode with a weight vector most closely matching the input vector is called the **winning neurode**.

For example, if the input vector is (0.35, 0.8), the winning neurode might have weight vector (0.4, 0.78). The learning rule would adjust the weight vector to make it even closer to the input vector. Only the winning neurode produces output, and only the winning neurode gets its weights adjusted. In more sophisticated models, only the weights of the winning neurode and its immediate neighbors are updated.

After training, a limited number of input vectors will map to activation of distinct output neurodes. Because the weights are modified in response to the inputs, rather than in response to desired outputs, competitive learning is called **unsupervised** learning, to distinguish it from the **supervised** learning of Perceptrons, Adalines and Backpropagation. In supervised learning, comparison is made between actual outputs and desired outputs supplied by an external supervisor. There is no external supervisor in competitive learning.



Simplified 4-neuron Hopfield network

6 ATTRACTOR NETWORKS

The most notable attractor networks are the **Hopfield Network [12]**, the **Boltzman Machine [13]** and the **Bidirectional Associative Memory** (BAM). The Hopfield Network can be represented in a number of ways, all of which are somewhat equivalent.

The diagram on the left indicates that every neurode has a connection with every other neurode in two directions, but it omits the detail that each neurode is also an input neurode and an output neurode, as is shown in the middle diagram. The diagram on the right is called a **Crossbar Network** representation of a Hopfield Network, and it is a convenient tool when analyzing connection weights as a matrix of numbers

The Hopfield Network is presented with an input vector, and the input vector remains active as the neurodes update their weights one-by-one in sequence (usually more than once for each neurode) until the output is constant. Weights are updated on the basis of the difference between input and output for each individual neurode. This process of arriving at the output is called **relaxation** or **annealing**, and can be expressed as an energy equation -- which is exactly what was done by physicist John Hopfield who conceived of this network.

The lower energy states are the "attractors" of the network. The settling of the network into its lowest energy state can be compared to a ball rolling to the bottom of a hill. If the hill has a hump, however, the ball may not fall to its lowest energy state, but be caught in a **local minimum**. The **Boltzman Machine** is a modified Hopfield Network that adds a "Boltzman temperature term" ("noise") to jostle the ball out of the local minimum.

REFERENCES

[1] Aleksander, Igor and H. Morton: An Introduction to Neural Computing. Chapman and Hall, London, UK 1990.

[2] Gurney, Kevin: An Introduction to Neural Networks. UCL Press, London, UK 1999.

[3] W. S. Mculloch and W. Pitts: A Logical calculus of the ideas immanent in nervous activity. Bull. Math. Bio. Phys. 5 p115-133 1943.

[4] D. O. Hebb: The organization of Behaviour: A Neuropsychological Theory. New York: Wiley, 1949.

[5] Fausett, Laurene: Fundamentals of Neural Networks: Architectures, Algorithms, and Applications. Prentice Hall, NJ, USA 1994.

[6] Kartalopoulos, Stamatios: Understanding Neural Networks and Fuzzy Logic: Basic Concepts and Applications. IEEE Press, NJ, USA 1996.

[7] Rosenblatt, F. (1958), "The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain", Psychological Review, Vol. 65, pp. 386–408.

[8] Stark, L., Okajima, M., Whipple, G. H., Computer Pattern Identification Techniques: Electrocardiographics Diagnosis, Comm. of the ACM, 5, 527-532, 1962.

[9] Widrow, B., and M. Hoff (1960), "Adaptive Switching Circuits", WESCON Convention Record, New York, 1960.

[10] Widrow, B., and M. Lehr (1990), "30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation", Proceedings of the IEEE, Vol. 78, No. 9, pp. 1415–1442.

[11] Bonissone, Piero: "Soft Computing: The Convergence of Emerging Reasoning Technologies" Soft Computing. Springer-Verlag, Germany/USA 1997.

[12] J. J. Hopfield: Neural networks and physical systems with emergent collective computational capabilities. Proceedings National Academy of Sciences (USA) 79,p2554-2558, 1982.

[13] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski: A learning algorithm for Boltzmann machines. Cognitive Sci., 9 p147-169, 1985.