ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

A Study on Basics of Neural Network

Shaiqua Jabeen[1], Shobhana D. Patil[2], Shubhangi V. Bhosale[3],

Bharati M. Chaudhari[4], Prafulla S. Patil[5]

Lecturer, Dept. of Computer Science, Dr. D. Y. Patil ACS College, Pune, Maharashtra, India[1,2,3,4,5]

ABSTRACT: Artificial neural networks commonly referred as the neural networks are the information or signal processing mathematical model that is based on the biological neuron. A neural network is a complex structure which consist a group of interconnected neurons which provides a very exciting alternatives for complex problem solving and other application which can play important role in today's computer science field so researchers from the different discipline are designing the artificial neural networks to solve the problems of pattern recognition, prediction, optimization, associative memory and control. In this paper we have presented the basic study of the artificial neural network, its characteristics and its applications.

KEYWORDS*:* Artificial Neural Network (ANN), Multi Layer Perceptron (MLP), Feedback Network, Feed-Forward Network, Artificial Neuron, Biological Paradigm, Radial Basis Function Network (RBFN).

I. INTRODUCTION

The study of brain is an interesting area since a long time. With advancement in the field of electronics and computer science, it was the assumed that we can use this natural way of this thinking process of brain to design some artificial intelligence system.

The first step toward artificial intelligence came into existence in 1943 when Warren McCulloch, a neurophysiologist, and a mathematician, Walter Pitts, wrote a paper on how neurons work. Mathematical analysis has solved some of the mysteries posed by the new models but has left many questions for future investigations. There is no need to say, the study of neurons, their interconnections, and their role as the brain's elementary building blocks is one of the most dynamic and important research fields in modern world of electronics and computer science. Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

1.1 Inspiration

The concept of neural network is basically introduced from the subject of biology where neural network plays an important and key role in human body. In human body work is done with the help of neural network. Neural Network is just a web of inter connected neurons which are millions and millions in number. With the help of these interconnected neurons all the parallel processing is done in human body and the human body is the best example of Parallel Processing.

A neuron is a special biological cell that process information from one neuron to another neuron with the help of some electrical and chemical change. It is composed of a cell body or soma and two types of out reaching tree like branches: the axon and the dendrites. The cell body has a nucleus that contains information about hereditary traits and plasma that holds the molecular equipments or producing material needed by the neurons as shown in Fig. 1.1.

The whole process of receiving and sending signals is done in particular manner like a neuron receives signals from other neuron through dendrites. The Neuron send signals at spikes of electrical activity through a long thin stand known as an axon and an axon splits this signals through synapse and send it to the other neurons as shown in Fig. 1.2.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

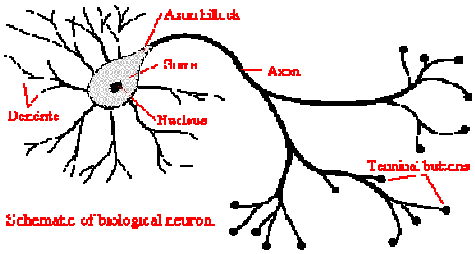

Fig. -1.1: Biological Neuron

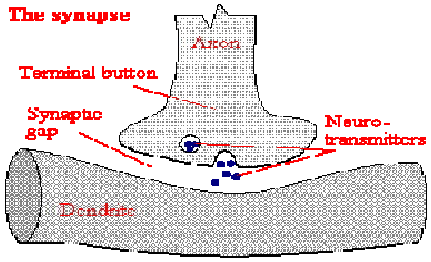

Fig. -1.2: Biological Neuron-Dendrite

The brain is a collection of about 10 billion interconnected neurons. Each neuron is a cell that uses biochemical reactions to receive process and transmit information. Each terminal button is connected to other neurons across a small gap called a synapse as shown in Fig. 1.2.

A neuron's dendritic tree is connected to a thousand neighbouring neurons. When one of those neurons fire, a positive or negative charge is received by one of the dendrites. The strengths of all the received charges are added together through the processes of spatial and temporal summation.

II.RELATED WORK

In [6] authors have explained the working of ANN in brief without any components and also explained the activation functions like threshold, sigmoid, step, linear etc. .In paper [9] authors have explained the artificial neural network model in detail with all the functions .In paper [11] author have explained training an artificial neural network in detail and they have also explained the simple neural model. In paper [12] author have explained the learning techniques in detail.

III.ARTIFICIAL NEURAL NETWORK

An Artificial Neuron is basically an engineering approach of biological neuron. It has device with many inputs and one output as shown in Fig. 2. ANN is consists of large number of simple processing elements that are interconnected with each other and layered also.

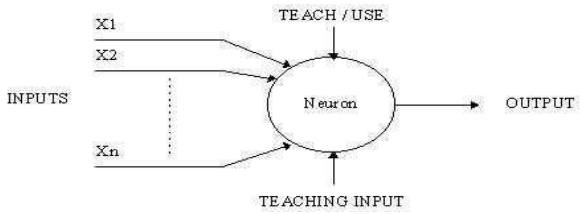

Fig. -2: ANN

Artificial Neural Networks are relatively crude electronic models based on the neural structure of the brain. The brain basically learns from experience. It is natural proof that some problems that are beyond the scope of current computers are indeed solvable by small energy efficient packages. This brain modelling also promises a less technical way to develop machine solutions. This new approach to computing also provides a more graceful degradation during system overload than its more traditional counterparts. These biologically inspired methods of computing are thought to be the next major advancement in the computing industry. Even simple animal brains are capable of functions that are currently impossible for computers. Computers do rote things well, like keeping ledgers or performing complex math. But computers have trouble recognizing even simple patterns much less generalizing those patterns of the past into

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

actions of the future. Now, advances in biological research promise an initial understanding of the natural thinking mechanism. This research shows that brains store information as patterns. Some of these patterns are very complicated and allow us the ability to recognize individual faces from many different angles. This process of storing information as patterns, utilizing those patterns, and then solving problems encompasses a new field in computing. This field, as mentioned before, does not utilize traditional programming but involves the creation of massively parallel networks and the training of those networks to solve specific problems. This field also utilizes words very different from traditional computing, words like behave, react, self-organize, learn, generalize, and forget.

Whenever we talk about a neural network, we should more popularly say ―Artificial Neural Network (ANN), ANN are computers whose architecture is modelled after the brain. They typically consist of hundreds of simple processing units which are wired together in a complex communication network. Each unit or node is a simplified model of real neuron which sends off a new signal or fires if it receives a sufficiently strong Input signal from the other nodes to which it is connected.

3.1 Working of ANN

Neural networks are the simple clustering of the primitive artificial neurons. This clustering occurs by creating layers which are then connected to one another. How these layers connect is the other part of the "art" of engineering networks to resolve real world problems.

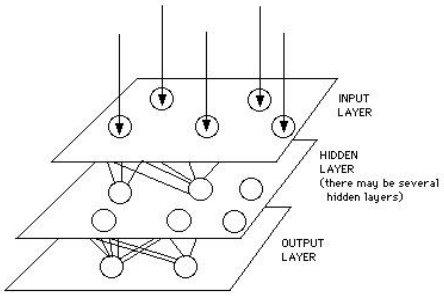

Fig. -3: Layers of ANN

Basically, all artificial neural networks have a similar structure or topology as shown in Fig. 3. In that structure some of the neurons interface to the real world to receive its inputs. Other neurons provide the real world with the network's outputs. This output might be the particular character that the network thinks that it has scanned or the particular image it thinks is being viewed. All the rest of the neurons are hidden from view

Training an Artificial Neural Network

Once a network has been structured for a particular application, that network is ready to be trained. To start this process the initial weights are chosen randomly. Then, the training, or learning, begins. There are two approaches to training - supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs. Unsupervised training is where the network has to make sense of the inputs without outside help. The vast bulk of networks utilize supervised training. Unsupervised training is used to perform some initial characterization on inputs. However, in the full blown sense of being truly self learning, it is still just a shining promise that is not fully understood, does not completely work, and thus is relegated to the lab.

1. Supervised Training:

In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

continually tweaked. The set of data which enables the training is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined. The current commercial network development packages provide tools to monitor how well an artificial neural network is converging on the ability to predict the right answer. These tools allow the training process to go on for days, stopping only when the system reaches some statistically desired point, or accuracy. However, some networks never learn. This could be because the input data does not contain the specific information from which the desired output is derived. Networks also don't converge if there is not enough data to enable complete learning. Ideally, there should be enough data so that part of the data can be held back as a test. Many layered networks with multiple nodes are capable of memorizing data. To monitor the network to determine if the system is simply memorizing its data in some non significant way, supervised training needs to hold back a set of data to be used to test the system after it has undergone its training.

If a network simply can't solve the problem, the designer then has to review the input and outputs, the number of layers, the number of elements per layer, the connections between the layers, the summation, transfer, and training functions, and even the initial weights themselves. Those changes required to create a successful network constitute a process wherein the "art" of neural networking occurs. Another part of the designer's creativity governs the rules of training. There are many laws (algorithms) used to implement the adaptive feedback required to adjust the weights during training. The most common technique is backward-error propagation, more commonly known as back-propagation

.

2. Unsupervised, or Adaptive Training

The other type of training is called unsupervised training. In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adaption. At the present time, unsupervised learning is not well understood. This adaption to the environment is the promise which would enable science fiction types of robots to continually learn on their own as they encounter new situations and new environments. Life is filled with situations where exact training sets do not exist. Some of these situations involve military action where new combat techniques and new weapons might be encountered. Because of this unexpected aspect to life and the human desire to be prepared, there continues to be research into, and hope for, this field. Yet, at the present time, the vast bulk of neural network work is in systems with supervised learning. Supervised learning is achieving results.

IV. NEURAL MODEL

A neuron has more than one input. A neuron with R inputs is shown in Fig. 4. The individual inputs p1,p2,...,pR are each weighted by corresponding elements w1, 1,w1, 2,...,w1,R of the weight matrix W.

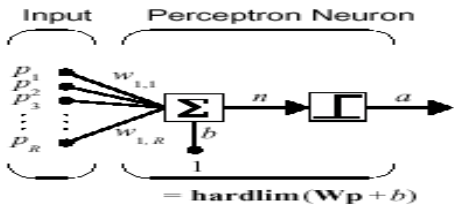

Fig. -4: Neural Model

4.1 Simple Neuron

A neuron with a single scalar input and no bias appears below in Fig.5.1.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

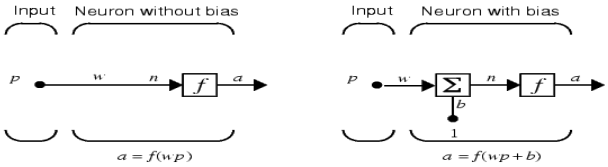

Fig. -5.1: Simple Neuron

The scalar input *p* is transmitted through a connection that multiplies its strength by the scalar weight *w*, to form the product w_p, again a scalar. Here the weighted input w_p is the only argument of the transfer function *f*, which produces the scalar output *a*. The neuron on the right has a scalar bias, *b*. You may view the bias as simply being added to the product w_p as shown by the summing junction or as shifting the function *f* to the left by an amount *b*. The bias is much like a weight, except that it has a constant input of 1.

The transfer function net input *n*, again a scalar, is the sum of the weighted input w_p and the bias *b*. This sum is the argument of the transfer function *f*. Here *f* is a transfer function, typically a step function or a sigmoid function, which takes the argument *n* and produces the output *a*. Examples of various transfer functions are given in the next section. Note that *w* and *b* are both *adjustable* scalar parameters of the neuron. The central idea of neural networks is that such parameters can be adjusted so that the network exhibits some desired or interesting behaviour. Thus, we can train the network to do a particular job by adjusting the weight or bias parameters, or perhaps the network itself will adjust these parameters to achieve some desired end.

Component 1: Weighting 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 element's summation function. Some inputs are made more important than others to have a greater effect on the processing element as they combine to produce a neural response. Weights are adaptive coefficients 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 its learning rules.

Component 2: Summation Function:

The inputs and corresponding weights are vectors which can be represented as (i_1, i_2. . . i_n) and (w_1, w_2. . . w_n). The total input signal is the dot product of these two vectors. The result; (i_1* w_1) + (i_2* w_2) +..... + (i_n* w_n); is a single number. The summation function can be more complex than just 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 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. Some summation functions have an additional 'activation function' applied to the result before it is passed on to the transfer function for the purpose of allowing the summation output to vary with respect to time. All is used in neural networks with this linear neuron. The bias term allows us to make affine transformations to the data.

Component 3: Transfer Function:

The result of the summation function is transformed to a working output through an algorithmic process known as the transfer function. In the transfer function the summation 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 and if it is less than the threshold, no signal (or some inhibitory signal) is generated. Both types of response are significant. The threshold, or transfer function, is generally non-linear. Linear functions are limited because the output is simply proportional to the input. The step type of transfer function would output zero and one, one and minus one, or other numeric combinations. Another type, the 'threshold' or ramping function, can mirror the input within a given range and still act as a step function outside that range. It is a linear function that is clipped to minimum and maximum values, making it non-linear.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

The transfer function for neural networks must be differential and therefore continuous to enable correcting error. The transfer function of a neuron is chosen to have a number of properties which either enhance or simplify the network containing the neuron. Derivative of the transfer function is required for computation of local gradient.
Crucially, for instance, any multilayer perceptron using a linear transfer function has an equivalent single-layer network, a non-linear function is therefore necessary to gain the advantages of a multi-layer network.
Below, u refers in all cases to the weighted sum of all the inputs to the neuron, i.e. for n inputs,

$$u = \sum_{i=1}^{n} w_i x_i$$

Where, w is a vector of synaptic weights and x is a vector of inputs.

Linear combination: In this case, the output unit is simply the weighted sum of its inputs plus a bias term. A number of such linear neurons perform a linear transformation of the input vector. This is usually more useful in the first layers of a network. A number of analysis tools exist based on linear models, such as harmonic analysis, and they can.

Step function: A step function is a function is likely used by the original Perceptron. The output is a certain value, A1, if the input sum is above a certain threshold and A0 if the input sum is below a certain threshold. The values used by the Perceptron were A1 = 1 and A0 = 0.

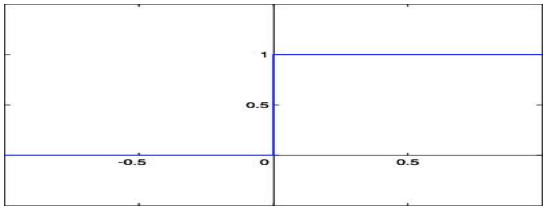

Chart -1.1: Step Function

As per the above, the output y of this transfer function is binary, depending on whether the input meets a specified threshold, θ. The "signal" is sent, i.e. the output is set to one, if the activation meets the threshold.

$$y = \begin{cases} 1 & \text{if } u \geq \theta \\ 0 & \text{if } u < \theta \end{cases}$$

This function is used in perceptrons and often shows up in many other models. It performs a division of the space of inputs by a hyper plane. It is especially useful in the last layer of a network intended to perform binary classification of the inputs. It can be approximated from other sigmoid functions by assigning large values to the weights.

Sigmoid: A fairly simple non-linear function, the sigmoid function such as the logistic function also has an easily calculated derivative, which can be important when calculating the weight updates in the network. It thus makes the network more easily manipulable mathematically, and was attractive to early computer scientists who needed to minimize the computational load of their simulations. It was previously commonly seen in multilayer perceptrons. However, recent work has shown sigmoid neurons to be less effective than rectified linear neurons. The reason is that the gradients computed by the back propagation algorithm tend to diminish towards zero as activations propagate through layers of sigmoidal neurons, making it difficult to optimize neural networks using multiple layers of sigmoidal neurons.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

The sigmoid function is an S-shaped graph. It is one of the most common forms of transfer function used in construction of ANNs. It is defined as a strictly increasing function. Mathematically its derivative is always positive. It exhibits a graceful balance between linear and nonlinear behaviour.

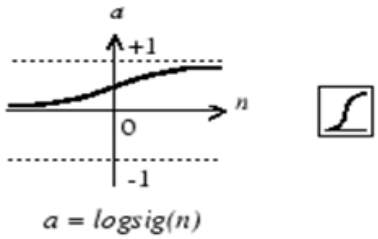

Chart -1.2: Log-Sigmoid Transfer Function

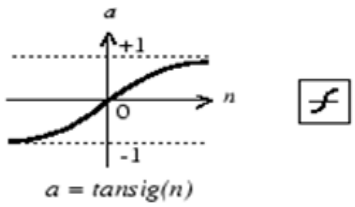

Chart -1.3: Tan-Sigmoid Transfer Function

This approaches a minimum and maximum value at the asymptotes. It is called a sigmoid when it ranges between 0 and 1, hyperbolic tangent when it ranges between -1 and 1. Both the function and its derivatives are continuous. Hyperbolic tangent function is an odd function

$$\tanh(-x) = -\tanh(x)$$

Using this property, only the absolute value of input is processed and the input sign is directly passed to the output. The Taylor series expansion of hyperbolic tangent is as follows:

$$\tanh(x) = x - x3/3 + 2x5/15 - 17x7/315 +$$

For small values of x, the higher order terms become small and can be ignored. Therefore, the hyperbolic tangent passes the small input values to output

$$\lim \tanh(x) = x \quad x \to 0$$

The output variation for large values of input is low. Considering the two last properties, input range is divided to three regions. Region I in which the output is approximately equal to input is named pass region while because of low variation of output in region III, it is named saturation region. Region II includes the rest of input range, named processing region.

Component 4: Scaling and Limiting:

After the 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 insures that the scaled result does not exceed an upper, or lower bound. This limiting is in addition to the hard limits that the original transfer function may have performed.

Component 5: Output Function (Competition):

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

Component 6: Error Function and Back-Propagated Value:

In most learning networks the difference between the current output and the desired output is calculated as an error which is then transformed by the error function to match particular network architecture. Most basic architectures use this error directly but some square the error while retaining its sign, some cube the error, other paradigms modify the error to fit their specific purposes. The error is propagated backwards to a previous layer. This back-propagated value

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

can be either the error, the error scaled in some manner (often by the derivative of the transfer function) or some other 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.

Component 7: Learning Function:
Its purpose is to modify the weights on the inputs of each processing element according to some neural based algorithm.

4.2 Network Architecture

The most commonly used structure is shown in Fig.6. This neural network is formed in three layers, called the input layer, hidden layer, and output layer. Each layer consists of one or more nodes, represented in this diagram by the small circles. The lines between the nodes indicate the flow of information from one node to the next. In this particular type of neural network, the information flows only from the input to the output (that is, from left-to-right). Other types of neural networks have more intricate connections, such as feedback paths.The nodes of the input layer are passive, meaning they do not modify the data. They receive a single value on their input, and duplicate the value to their multiple outputs.

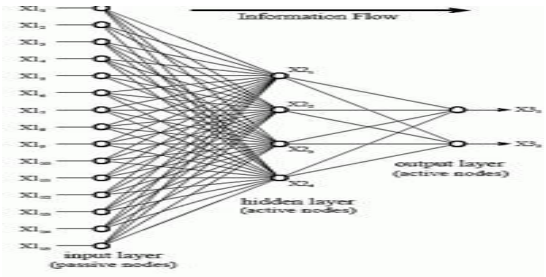

Fig. -6: Multiple Neurons

In comparison, the nodes of the hidden and output layer are active. This means they modify the data as shown in Fig.-6. The variables: $X1_1$, $X1_2$…$X1_{15}$ hold the data to be evaluated (see Fig.6). For example, they may be pixel values from an image, samples from an audio signal, stock market prices on successive days, etc. They may also be the output of some other algorithm, such as the classifiers in our cancer detection example: diameter, brightness, edge sharpness, etc.

Each value from the input layer is duplicated and sent to all of the hidden nodes. This is called a fully interconnected structure. As shown in Fig. 6, the values entering a hidden node are multiplied by weights, a set of predetermined numbers stored in the program. The weighted inputs are then added to produce a single number. This is shown in the diagram by the symbol, ∑. Before leaving the node, this number is passed through a nonlinear mathematical function called a sigmoid. This is an "s" shaped curve that limits the node's output. That is, the input to the sigmoid is a value between -∞ and +∞, while its output can only be between 0 and 1.

The outputs from the hidden layer are represented in the flow diagram (Fig. 6) by the variables: X21 , X22, X23 and $X2_4$. Just as before, each of these values is duplicated and applied to the next layer. The active nodes of the output layer combine and modify the data to produce the two output values of this network, X31 and X32.

Neural networks can have any number of layers, and any number of nodes per layer. Most applications use the three layer structure with a maximum of a few hundred input nodes. The hidden layer is usually about 10% the size of the input layer. In the case of target detection, the output layer only needs a single node. The output of this node is thresholded to provide a positive or negative indication of the target's presence or absence in the input data.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

V.CLASSIFICATION OF ANN

Neural networks can be hardware- (neurons are represented by physical components) or software-based (computer models), and can use a variety of topologies and learning algorithms.

5.1 Feedforward neural network

The feedforward neural network was the first and arguably most simple type of artificial neural network devised. In this network the information moves in only one direction—forward: From the input nodes data goes through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network. Feedforward networks can be constructed from different types of units, e.g. binary McCulloch-Pitts neurons, the simplest example being the perceptron. Continuous neurons, frequently with sigmoidal activation, are used in the context of back propagation of error.

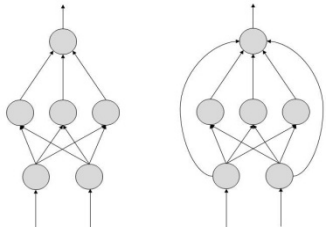

Fig. -7: Feedforward Neural Network

5.2 Recurrent neural network

Contrary to feedforward networks, recurrent neural networks (RNNs) are models with bi-directional data flow. While a feedforward network propagates data linearly from input to output, RNNs also propagate data from later processing stages to earlier stages. RNNs can be used as general sequence processors.

Fully recurrent network

This is the basic architecture developed in the 1980s: a network of neuron-like units, each with a directed connection to every other unit. Each unit has a time-varying real-valued (more than just zero or one) activation (output). Each connection has a modifiable real-valued weight. Some of the nodes are called input nodes, some output nodes, the rest hidden nodes.

For supervised learning in discrete time settings, training sequences of real-valued input vectors become sequences of activations of the input nodes, one input vector at a time. At any given time step, each non-input unit computes its current activation as a nonlinear function of the weighted sum of the activations of all units from which it receives connections. There may be teacher-given target activations for some of the output units at certain time steps. For example, if the input sequence is a speech signal corresponding to a spoken digit, the final target output at the end of the sequence may be a label classifying the digit. For each sequence, its error is the sum of the deviations of all activations computed by the network from the corresponding target signals. For a training set of numerous sequences, the total error is the sum of the errors of all individual sequences.

To minimize total error, gradient descent can be used to change each weight in proportion to its derivative with respect to the error, provided the non-linear activation functions are differentiable. Various methods for doing so were developed in the 1980s and early 1990s by Paul Werbos, Ronald J. Williams, Tony Robinson, Jürgen Schmidhuber, Barak Pearlmutter, and others. The standard method is called "back propagation" or BPTT, a generalization of back-propagation for feedforward networks. A more computationally expensive online variant is called "Real-Time Recurrent Learning" or RTRL. Unlike BPTT this algorithm is local in time but not local in space. There also is an online hybrid between BPTT and RTRL with intermediate complexity, and there are variants for continuous time. A major problem with gradient descent for standard RNN architectures is that error gradients vanish exponentially

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

quickly with the size of the time lag between important events, as first realized by Sepp Hochreiter in 1991. The Long short-term memory architecture overcomes these problems.

Hopfield network

The Hopfield network (like similar attractor-based networks) is of historic interest although it is not a general RNN, as it is not designed to process sequences of patterns. Instead it requires stationary inputs. It is an RNN in which all connections are symmetric. Invented by John Hopfield in 1982 it guarantees that its dynamics will converge. If the connections are trained using Hebbian learning then the Hopfield network can perform as robust content-addressable memory, resistant to connection alteration.

Self-organizing map

The self-organizing map (SOM) invented by Teuvo Kohonen performs a form of unsupervised learning. A set of artificial neurons learn to map points in an input space to coordinates in an output space. The input space can have different dimensions and topology from the output space and the SOM will attempt to preserve these.

Simple recurrent networks

A simple modification of the basic feedforward architecture above was employed by Jeff Elman and Michael I. Jordan. A three-layer network is used, with the addition of a set of "context units" in the input layer. There are connections from the hidden layer (Elman) or from the output layer (Jordan) to these context units fixed with a weight of one. At each time step, the input is propagated in a standard feedforward fashion, and then a simple back propagation-like learning rule is applied (this rule is not performing proper gradient descent, however).

The fixed back connections result in the context units always maintaining a copy of the previous values of the hidden units (since they propagate over the connections before the learning rule is applied).

5.3 Multilayer Perceptron

A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called back propagation for training the network. MLP is a modification of the standard linear perceptron and can distinguish data that are not linearly separable.

The multilayer perceptron consists of three or more layers (an input and an output layer with one or more hidden layers) of nonlinearly-activating nodes and is thus considered a deep neural network. Since an MLP is a Fully Connected Network, each node in one layer connects with a certain weight wij to every node in the following layer. Some people do not include the input layer when counting the number of layers and there is disagreement about wij should be interpreted as the weight from i to j or the other way around.

Multi Layer perceptron (MLP) is a feedforward neural network with one or more layers between input and output layer. Feedforward means that data flows in one direction from input to output layer (forward). This type of network is trained with the back propagation learning algorithm. MLPs are widely used for pattern classification, recognition, prediction and approximation. Multi Layer Perceptron can solve problems which are not linearly separable.

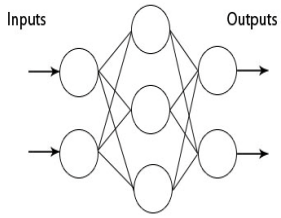

Fig -8.1: MLP

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

5.4 Radial Basis Function Network

An RBFN performs classification by measuring the input's similarity to examples from the training set. Each RBFN neuron stores a "prototype", which is just one of the examples from the training set. When we want to classify a new input, each neuron computes the Euclidean distance between the input and its prototype.

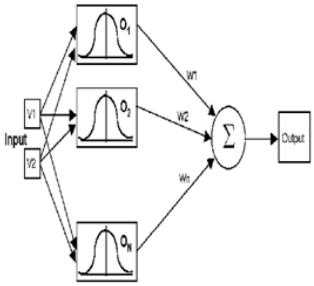

Fig -9: RBF

RBF networks have three layers:

1. Input layer – There is one neuron in the input layer for each predictor variable. In the case of categorical variables, N-1 neurons are used where N is the number of categories. The input neurons (or processing before the input layer) standardize the range of the values by subtracting the median and dividing by the interquartile range. The input neurons then feed the values to each of the neurons in the hidden layer.

2. Hidden layer – This layer has a variable number of neurons (the optimal number is determined by the training process). Each neuron consists of a radial basis function centered on a point with as many dimensions as there are predictor variables. The spread (radius) of the RBF function may be different for each dimension. The centers and spreads are determined by the training process. When presented with the x vector of input values from the input layer, a hidden neuron computes the Euclidean distance of the test case from the neuron's center point and then applies the RBF kernel function to this distance using the spread values. The resulting value is passed to the summation layer.

3. Summation layer – The value coming out of a neuron in the hidden layer is multiplied by a weight associated with the neuron (W1, W2... Wn in Fig. 9) and passed to the summation which adds up the weighted values and presents this sum as the output of the network. Not shown in this figure is a bias value of 1.0 that is multiplied by a weight W0 and fed into the summation layer. For classification problems, there is one output (and a separate set of weights and summation unit) for each target category. The value output for a category is the probability that the case being evaluated has that category.

VI. APPLICATIONS

Aerospace: High performance aircraft autopilots, flight path simulations, aircraft control systems, autopilot enhancements, aircraft component simulations, aircraft component fault detectors.

Automotive: Automobile automatic guidance systems, fuel injector control, automatic braking systems, misfire detection, virtual emission sensors, warranty activity analysers.

Banking: Check and other document readers, credit application evaluators, cash forecasting, firm classification, exchange rate forecasting, predicting loan recovery rates, measuring credit risk.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

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 layout, process control, chip failure analysis, machine vision, voice synthesis, nonlinear modelling.

Entertainment: Animation, special effects, market forecasting.

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

Insurance: Policy application evaluation, product optimization.

Manufacturing: Manufacturing process control, product design and analysis, process, machine diagnosis, real-time particle identification, visual quality inspection systems, beer testing, welding quality analysis, paper quality prediction, computer chip quality analysis, analysis of grinding operations, chemical product design analysis.

Medical: 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: Exploration, smart sensors, reservoir modelling, well treatment decisions, seismic interpretation.

Robotics: Trajectory control, forklift robot, manipulator controllers, vision systems, autonomous vehicles.

Speech: Speech recognition, speech compression, vowel classification, text to speech synthesis.

Securities: Market analysis, automatic bond rating, and stock trading advisory systems.

Telecommunication: Image and data compression, automated information services, real-time translation of spoken language, customer payment processing systems.

Transportation: Truck brake diagnosis systems, vehicle scheduling, routing systems.

VII. LIMITATIONS

The various Limitations of ANN are:-

 Neural networks are not magic hammers. Neural networks are still viewed by many as being magic hammers which can solve any machine learning problem, and as a result, people tend to apply them indiscriminately to problems for which they are not well suited. Although neural networks do have a proven track record of success for certain specific problem domains, as a consumer of machine learning technology, you're almost always better off using approaches that have stronger theoretical underpinnings, rather than just throwing a general-purpose neural network at your problem and hoping for the best.

 Neural networks are too much of a black box. This has several consequences. It makes them difficult to train: the training outcome can be nondeterministic and depend crucially on the choice of initial parameters, e.g. the starting point for gradient descent when training back propagation networks. It makes them hard to determine how they are solving a problem, because they are opaque. It makes them difficult to troubleshoot when they don't work as you

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

expect, and when they do work, you will never really feel confident that they will generalize well to data not included in your training set because, fundamentally, you don't understand how your network is solving the problem, or what idiosyncrasies of the training data the network has overfit to. For a classic example of this, see this article about the Pentagon's attempts in the 1980s to use neural networks to detect tanks in images: Neural Network Follies.

 Neural networks are not probabilistic. For the most part, NNs have few if any probabilistic underpinnings, unlike their more statistical or Bayesian counterparts. It's incredibly useful to know how confident your classifier is about its answers because that information allows you to better manage the cost of making errors by tuning your classifier. A neural network might give you a continuous number as its output (e.g. a score) but translating that into a probability is often difficult. Approaches with stronger theoretical foundations usually give you those probabilities directly.

 Neural networks are not a substitute for understanding the problem deeply. Instead of investing your time throwing a neural net at the problem, you're almost always better off investing a little extra time studying, analyzing and dissecting your data first, then using your better understanding to choose a technique with stronger theoretical foundations, a technique which you will now have more confidence will work well for your problem. For example, if you are building a classifier, rather than just throwing a back-prop neural network at your data and hoping for the best, spend time visualizing the data and selecting or creating the best input features using whatever domain-specific knowledge and expertise you have available to you.

VIII. CONCLUSION

In this paper we have discussed about the fundamentals of neural networks. By which we have concluded that the computing world has a lot to gain from neural networks. Their ability to learn by example makes them very flexible and powerful. Furthermore there is no need to devise an algorithm in order to perform a specific task; i.e. there is no need to understand the internal mechanisms of that task. They are also very well suited for real time systems because of their fast response and computational times which are due to their parallel architecture.

Neural networks also contribute to other areas of research such as neurology and psychology. They are regularly used to model parts of living organisms and to investigate the internal mechanisms of the brain.

Perhaps the most exciting aspect of neural networks is the possibility that some day 'conscious' networks might be produced. There are a number of scientists arguing that consciousness is a 'mechanical' property and that 'conscious' neural networks are a realistic possibility.

Finally, we would like to state that even though neural networks have a huge potential we will only get the best of them when they are integrated with computing, AI, fuzzy logic and related subjects.

REFERENCES

[1] Martin T. Hagen, Howard B. Demuth, Mark H. Beale and Orland De Jesus, "Neural Network Design" 2nd Edition.
[2] Kevin Gurney, "An Introduction To Neural Networks" UCL Press, 2004.
[3] David Kriesel, "A Brief Introduction To Neural Networks", 2005.
[4] Haykin S., "Neural Networks A Comprehensive Foundation", 2nd edition, Pearson Education, 1999.
[5] About Neural Network from website http://en.wikipedia.org / wiki/Neural_network .
[6] Vidushi Sharma,Sachin Rai & Anurag Dev," A Comprehensive Study of Artificial Neural Network"., International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10, pp.278-284,),2012.
[7] Christos Stergiou and Dimitrios Siganos, "Neural Networks".
[8] Carlos Gershenson, "Artificial Neural Networks for Beginners", United kingdom.
[9] N. Murata, S. Yoshizawa, and S. Amari, ―Learning curves, model selection and complexity of neural networks,‖ in Advances in Neural Information Processing Systems 5, S. Jose Hanson, J. D. Cowan,and C. Lee Giles, ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 607-614.
[10] Raul Rojas, "Neural Networks- A Systematic Introduction", Springer-Verlag, Berlin, 1996.
[11] Sonali B. Maind and Priyanka Wankar, "Research Paper on Basic of Artificial Neural Network" ,International Journal On Recent and Innovatiojn Trends in Computing and Communication, Vol. 2, Issue 1,pp.97-99,2014.
[12] Jurgen Schmidhuber, "Deep Learning in Neural Networks-An Overview", ELSEVIER,Vol. 61, pp85-117,2015.