Journal of Theoretical and Applied Information Technology

www.jatit.org

HAND WRITTEN CHARACTER RECOGNITION USING BACK PROPAGATION NETWORK

SRINIVASA KUMAR DEVIREDDY, SETTIPALLI APPA RAO

Nalanda Institute of Engineering & Technology, Kantepudi , Sattenapalli (M), Guntur (Dt.), A.P., India.
E-Mail: srinivaskumar_d@yahoo.com , apparao_settipalli@yahoo.com

ABSTRACT

A Neural network is a machine that is designed to model the way in which the brain performs a particular task or function of interest: The network is usually implemented by using electronic components or is simulated in software on a digital computer. *" A neural network is a massively parallel distributed processor made up of simple processing units which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:*
1)Knowledge is required by the network from its environment through a learning process.
2)Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge".

In this paper, we proposed a system capable of recognizing handwritten characters or symbols, inputted by the means of a mouse. The system provides means for training the input characters first, then there is a classification option where the patterns or symbols that have already been trained should be fed, in order to recognize it. There are full options for the users, like (1) To load a default set of patterns which can either be trained or the default training file can be loaded. After which the recognition takes place. (2) To classify a line of text and (3) Options to either load a pattern file, trained or untrained & finally the option to create new patterns by the user to train & classify.

Key words *: Neural Network, ANN, Neuron, Knowledge, BPN, Supervised, Pattern Recognition.*

INTRODUCTION:

An Artificial Neural Network (ANN) is information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

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. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest. Other advantages include:

1**. Adaptive learning**: An ability to learn how to do tasks based on the data given for training or initial experience.
2.**Self-Organization**: An ANN can create its own organization or representation of the information it receives during learning time.
3. **Real Time Operation**: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4**. Fault Tolerance via Redundant Information Coding**: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That

restricts the problem solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don't exactly know how to do. Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

Neural networks are a form of multiprocessor computer system, with i)simple processing elements, ii)a high degree of interconnection, iii)simple scalar messages, & iv)adaptive interaction between elements. Conventional computers use a cognitive approach to problem solving; the way the problem is to solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault. Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

Historical background

Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several years. Many important advances have been boosted by the use of inexpensive computer emulations. Following an initial period of enthusiasm, the field survived a period of frustration and disrepute. During this period when funding and professional support was minimal, important advances were made by relatively few researchers. These pioneers were able to develop convincing technology which surpassed the limitations identified by Minsky and Papert. Minsky and Papert, published a book (in 1969) in which they summed up a general feeling of frustration (against neural networks) among researchers, and was thus accepted by most without further analysis. Currently, the neural network field enjoys a resurgence of interest and a corresponding increase in funding.

The first artificial neuron was produced in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pits. But the technology available at that time did not allow them to do too much. Neural networks have seen an explosion of interest over the last few years, and are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics. Indeed, anywhere that there are problems of prediction, classification or control, neural networks are being introduced. This sweeping success can be attributed to a few key factors:

Power. Neural networks are very sophisticated modeling techniques capable of modeling extremely complex functions. In particular, neural networks are *nonlinear* (a term which is discussed in more detail later in this section). Neural networks also keep in check the *curse of dimensionality* problem that bedevils attempts to model nonlinear functions with large numbers of variables.

Ease of use. Neural networks *learn by example*. The neural network user gathers representative data, and then invokes *training algorithms* to automatically learn the structure of the data. Although the user does need to have some heuristic knowledge of how to select and prepare data, how to select an appropriate neural network, and how to interpret the results, the level of user knowledge needed to successfully apply neural networks is much lower than would be the case using (for example) some more traditional nonlinear statistical methods.

Basic Study :The Biological Model -How the Human Brain Learns?: The brain is principally composed of a very large number (circa 10,000,000,000) of *neurons*, massively interconnected (with an average of several thousand interconnects per neuron, although this varies enormously). Each neuron is a specialized cell which can propagate an electrochemical signal. The neuron has a branching input structure, a cell body, and a branching output structure (the axon). The axons of one cell connect to the dendrites of another via a synapse. When a neuron is activated, it *fires* an

electrochemical signal along the axon. This signal crosses the synapses to other neurons, which may in turn fire. A neuron fires only if the total signal received at the cell body from the dendrites exceeds a certain level (the firing threshold).

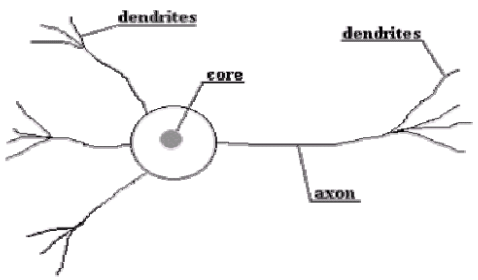
Recent research in cognitive science, in particular in the area of no conscious information processing, have further demonstrated the enormous capacity of the human mind to infer ("learn") simple input-output co variations from extremely complex stimuli. Thus, from a very large number of extremely simple processing units (each performing a weighted sum of its inputs, and then firing a binary signal if the total input exceeds a certain level) the brain manages to perform extremely complex tasks. Of course, there is a great deal of complexity in the brain is interesting that artificial neural networks can achieve some remarkable results using a model not much more complex than this. Much is still unknown about how the brain trains itself to process information, so theories abound. In the human brain, a typical neuron collects signals from others through a host of fine structures called dendrites. The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. At the end of each branch, a structure called a synapse converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.

From Human Neurons to Artificial Neurons

We conduct these neural networks by first trying to deduce the essential features of neurons and their interconnections. We then typically program a computer to simulate these features.

Figure 1) Biological Neuron

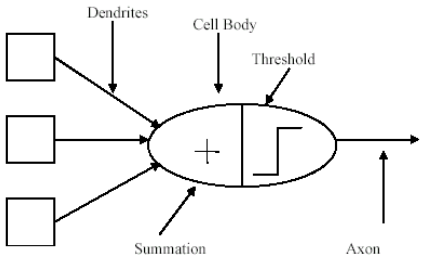
However because our knowledge of neurons is



incomplete and our computing power is limited, our models are necessarily gross idealizations of real networks of neurons. To capture the essence of biological neural systems, an artificial *neuron* is defined as follows:

It receives a number of inputs. Each input comes via a connection that has a strength (or *weight*); these weights correspond to synaptic efficacy in a biological neuron. Each neuron also has a single threshold value. The weighted sum of the inputs is formed, and the threshold subtracted, to compose the *activation* of the neuron. The activation signal is passed through an activation function to produce the output of the neuron. If the step activation function is used (i.e., the neuron's output is 0 if the input is less than zero, and 1 if the input is greater than or equal to 0).

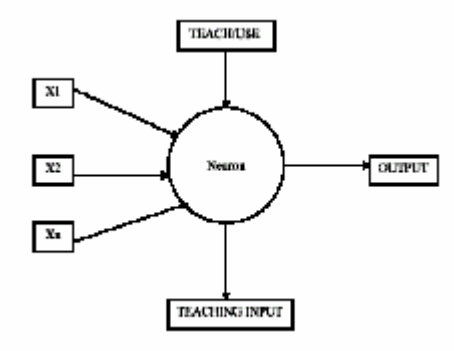
A neural network has its neurons divided into subgroups of fields and elements in each subgroup are placed in a row or a column. Each subgroup is then called to as layer of neurons in the network. A neural network may have input layer that supply the input signals for the neurons in the next layer, output layer where output is generated



and in between them hidden layer(s) that process information between input and output layers. Two neurons are connected with a weight which may be

Figure 2) The neuron model

positive, negative, or zero. Basically, the internal activation or raw output of a neuron in a neural network is a weighted sum of its inputs multiplied by weights connected to it. In mathematical form $Y = W_i X_i$ where, Y is the activation (output); W_i, the ith weight of i^{th} neuron; and X_i; is the i^{th} input to i^{th} neuron.



Figurue 3) A simple Neuron

Artificial Neural Networks : A simple neuron

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.

This describes an individual neuron. The next question is: how should neurons are connected together? If a network is to be of any use, there must be inputs and outputs. Inputs and outputs correspond to sensory and motor nerves such as those coming from the eyes and leading to the hands. However, there also can be hidden neurons that play an internal role in the network. The input, hidden and output neurons need to be connected together. A simple network has a *feed forward* structure: signals flow from inputs, forwards through any hidden units, eventually reaching the output units. Such a structure has stable behavior. However, if the network is *recurrent* it can be unstable, and has very complex dynamics. Recurrent networks are very interesting to researchers in neural networks, but so far it is the feed forward structures that have proved most useful in solving real problems.

When the network is executed, the input variable values are placed in the input units, and then the hidden and output layer units are progressively executed. Each of them calculates its activation value by taking the weighted sum of the outputs of the units in the preceding layer, and subtracting the threshold. The activation value is passed through the activation function to produce the output of the neuron. When the entire network has been executed, the outputs of the output layer act as the output of the entire network.

X1:		**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**
X2:		**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**
X3:		**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
OUT:		**0**	**0**	**0/1**	**0/1**	**0/1**	**1**	**0/1**	**1**

Firing rules

The firing rule is an important concept in neural networks and accounts for their high flexibility. A firing rule determines how one calculates whether a neuron should fire for any input pattern. It relates to all the input patterns, not only the ones on which the node was trained. A simple firing rule can be implemented by using Hamming distance technique. The rule goes as follows:

Take a collection of training patterns for a node, some of which cause it to fire (the 1-taught set of patterns) and others which prevent it from doing so (the 0-taught set). Then the patterns not in the collection cause the node to fire if, on comparison, they have more input elements in common with the 'nearest' pattern in the 1-taught set than with the 'nearest' pattern

in the 0-taught set. If there is a tie, then the pattern remains in the undefined state.

For example, a 3-input neuron is taught to output 1 when the input (X1,X2 and X3) is 111 or 101 and to output 0 when the input is 000 or 001. Then, before applying the firing rule, the truth table is;

As an example of the way the firing rule is applied, take the pattern 010. It differs from 000 in 1 element, from 001 in 2 elements, from 101 in 3 elements and from 111 in 2 elements. Therefore, the ‘nearest’ pattern is 000 which belongs in the 0-taught set. Thus the firing rule requires that the neuron should not fire when the input is 001. On the other hand, 011 is equally distant from two taught patterns that have different outputs and thus the output stays undefined (0/1).By applying the firing in every column the following truth table is obtained:

X1:		**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**
X2:		**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**
X3:		**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
OUT:		**0**	**0**	**0**	**0/1**	**0/1**	**1**	**1**	**1**

The difference between the two truth tables is called the generalization of the neuron. Therefore the firing rule gives the neuron a sense of similarity and enables it to respond 'sensibly' to patterns not seen during

training. The single artificial neurons can now be interconnected in many different ways leading to a variety of neural networks with different architectures, learning rules and abilities. The most important ones are: Feed forward networks, Adaptive Resonance Theory (ART), Hopfield nets, Kohonen's self-organizing feature maps, Radial Basis Functions (RBF), Boltzmann-machines, and Cascade-correlation. A very simple way is to organize the neurons in several layers as shown in Figure. This architecture is called a feed forward net, since neurons of one layer are only connected with neurons of the succeeding layer, without any recurrent connections. Normally these nets consist of one input layer, one or two hidden layers and one output layer. With such a net, input data are mapped from the n-dimensional input space to an m-dimensional output space.

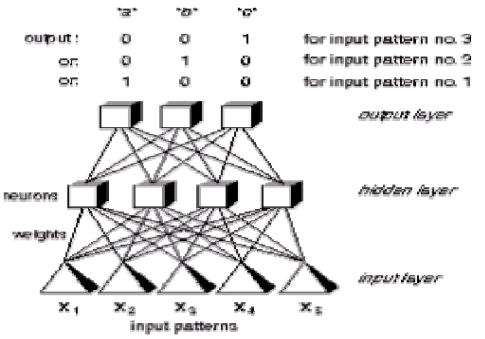


Figure4) Feed Forward Neural Network

The set of Fourier coefficients of the spoken character "a" is presented to the input layer. The desired output is that output neuron no. 1, representing the character "a", should be **1** and all other output neurons should be zero. When presenting the spectrum of the character "b" as input to the net, output neuron no. 2, representing the character "b", should be **1** and all the others, zero and so on. In order to be able to accomplish this task, the net has to be trained by a supervisor (supervised learning) just as a child learns to read and to write. This is done by presenting each pattern (in this example the Fourier coefficients of every character of the alphabet) many times to the net and at the same time furnishing the desired output (in this case the corresponding symbol associated with one output neuron), until it correctly classifies the presented patterns. In order to assess the performance of the trained neural net, it has to be tested with patterns not contained in the training data set (e.g. the Fourier spectrum of characters spoken by another person and/or superposed with various amounts of noise). However, the net should not produce an output for which it has not been trained.

The detailed training procedure is as follows:

1. Split the data set into a training set and a test set. Normally the training set is larger than the test set. Often the desired outputs have to be normalized to the range [0 : 1] since the sigmoid function only returns values in this range. The input patterns do not have to be normalized.
2. Initialize all weights, including all biases, to small random values normally in the range of [-1 : +1]. This determines the starting point on the error surface for the gradient descent method, whose position can be essential for the convergence of the network.
3. Forward propagation of the first input pattern of the training set from the input layer over the hidden layer(s) to the output layer, where each neuron sums the weighted inputs, passes them through the nonlinearity and passes this weighted sum to the neurons in the next layer.
4. Calculation of the difference between the actual output of each output neuron and its corresponding desired output. This is the error associated with each output neuron.
5. Back propagating this error through each connection by using the Back propagation learning rule and thus determining the amount each weight has to be changed in order to decrease the error at the output layer.
6. Correcting each weight by its individual weight update.
7. Presenting and forward propagating the next input pattern. Repeat steps 3-7 until a certain stopping criterion is reached, for example that the error falls below a predefined value.

The one-time presentation of the entire set of training patterns to the net constitutes a training epoch. After terminating the training phase the trained net is tested with new, unseen patterns from the test data set. The patterns are forward propagated, using the weights now available from training, and the error at the output layer is determined (no weight-update is performed!). If performance is sufficiently

good, the net is ready for- use. If not, it has to be retrained with the same patterns and parameters or something has to be changed (e.g. number of hidden neurons, additional input patterns, different kinds of information contained in the input patterns, ...).

It is also important that the net not be "over trained": if it is trained for too many epochs it starts "memorizing" the training patterns and can no longer recognize patterns other than those it was explicitly trained for. This effect can be compared to the over fitting of a discretely sampled function.

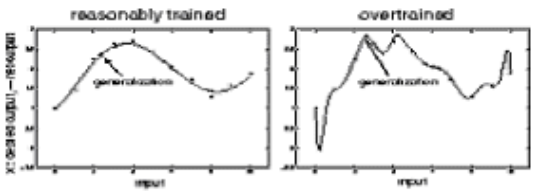


Figure 5) Example of a neural net trying to fit a discretely sampled function

In summary the goal of training a neural network is that it becomes able to generalize. This means that it only extracts some important features of the data and thus also classifies slightly differing patterns to belong to the same class. This also provides robustness in the presence of noise.

In general, we don't know the exact nature of the relationship between inputs and outputs - if you knew the relationship, you would model it directly. The other key feature of neural networks is that they learn the input/output relationship through training. There are two types of training used in neural networks, with different types of networks using different types of training. These are supervised and unsupervised training, of which supervised is the most. The functionality of a neural network is determined by the combination of the topology and the weights of the connections within the network. The topology is usually held fixed, and the weights are determined by a certain training algorithm. The process of adjusting the weights to make the network learn the relationship between the inputs and targets is called learning, or training. Many learning algorithm have been invented to help find an optimum set of weights that results in the solution of the problems. They can roughly be divided into two main groups.

1) Supervised Learning

The network is trained by providing it with inputs and desired outputs (target values). These input-output pairs are provided by an external teacher, or by the system containing the network. The difference between the real outputs and the desired outputs is used by the algorithm to adapt the weights in the network. It is often posed as function approximation problem given training data consisting or pairs of input patterns x, and corresponding target t, the goal is to find a function f(x) that matches the desired response for each training input.

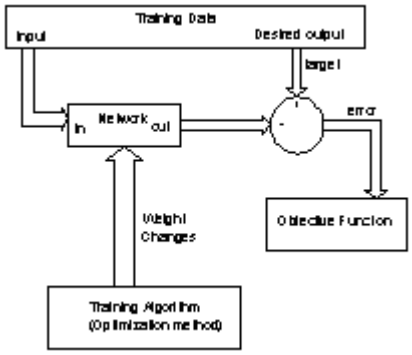


Figure 6) Supervised Learning Model

2) Unsupervised Learning

With unsupervised learning, there is no feedback from the environment to indicate if the outputs of the network are correct. The network must discover features, regulations, correlations, or categories in the input data automatically. In fact, for most varieties of unsupervised learning, the targets are the same as inputs. In other words, unsupervised learning usually performs the some task as an auto-associative network, compressing the information from the inputs. In this mode of learning, the network must discover for itself any possibly existing patterns, regularities, separating properties etc. While discovering, these, the network undergoes changes of its parameters, which is called *self – organization.* The technique is unsupervised learning is often used to perform clustering as the unsupervised classification of objects without providing information about the actual classes. Finally, learning is often not possible in an unsupervised environment, as would probably be true in the case illustrated pattern classes not easily discernible even for a human.

To illustrate some problems that often arise when we are attempting to automate complex pattern- recognition applications let us consider the design of a computer program that must translate a 5*7 matrix of

binary numbers representing the bit-mapped pixel image of an alphanumeric character to its equivalent eight-bit ASCII code. This basic problem, pictured below, appears to be relatively trivial at first glance. Since there is no obvious mathematical function that will perform the desired translation, and because it would undoubtedly take too much time (both human and computer time) to perform a pixel by pixel correlation, the best algorithmic solution would be to use a lookup table

The lookup table needed to solve this problem would be a non dimensional linear array of ordered pairs, each taking the form:

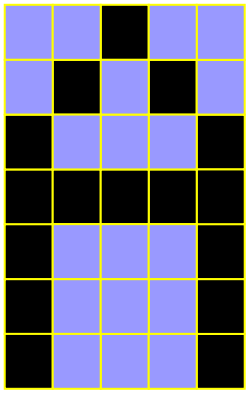


Figure 7) Mapping of "A"Character image

Each character image is mapped to its corresponding code. The first is the numeric equivalent of the bit-pattern code, which we generate by moving the seven rows of the matrix to a single row and considering the result to be a 35-bit binary number. The second is the ASCII code associated with the character. The array would contain exactly the same number of the pairs as there were characters to convert.

Although the lookup-table approach is reasonably faster and easy to maintain, there are many situations that occur in real systems that cannot be handled by this method. For example, consider the same pixel-image –to –ASCII conversion process in a more realistic environment. Let suppose that our character image scanner alters a random pixel in the input image matrix due to noise when the image was read. This single pixel error would cause the lookup algorithm to return either a null or the wrong ASCII Code, since the match between the input pattern and the target pattern must be exact. Now consider the amount of additional software that must be added to the lookup table algorithm to improve the ability of the computer to guess at which character the noisy image should have been. Single-bit errors are fairly easy to find and correct. Multibit errors become increasing difficult as the number of bit errors grows.

To complicate matters even further, how could our software compensate for noise on the image if that noise happened to make an Output look like a Q or an E look like an accurate output all the time, an inordinate amount of CPU time would be spent eliminating noise from the input-pattern period to attempting to translate it to ASCII.

One solution to this dilemma is to take advantage of the parallel nature of the neural network to reduce the time required by the sequential processor to perform the mapping. In addition, system- development time can be reduced because the network can learn proper algorithm without having some one deduce that algorithm in advance. Our paper deals with the recognition of the hand written characters which are predefined by the user. It maps to the actual patterns but not the ASCII Character set

To simplify our project, we will restrict the number of characters the neural network recognize to the ten decimal digits,0,1,….9,rather than using the full **ASCII** character set .We adopt this constraint only to the clarify the example: there is no reason why an **ANS** could not be recognize all characters, regardless of case or style. Since our objective is to have the neural network determine which of ten digits a particular hand –drawn character is ,we can create a network structure that has ten discrete output units, one for each character to be identified .The strategy simplifies the character –discrimination function of the network ,as it allows us to use a network that contains binary units on the **output** layer. Further more ,if we insist that the output units behave according to a simple on-off strategy, the process of converting an input signal to an output signal becomes a simple majority function .

Based on these considerations, we now know that our network should contain ten binary units as its output structure. Similarly, we must determine how we will model the character input for the network .Keeping in mind that we have already indicated a preference for binary output units, we can again simplify our task if we

model the input data as a vector containing binary elements, which will allow us to use a network with only one type of processing unit. To create this type of input, we borrow an idea from the video world and pixels the character .We will arbitrarily size the pixels image as a 10*8 matrix ,using a 1 to represent a pixel that is on and a 0 to represent a pixel that is off.

Furthermore, we can dissect the matrix into a set of row vectors, which can then be concatenated into a single row vector of dimension 80.Thus; we have defined the dimension and characteristics of the input pattern for our network. At this point ,all that remains is to the size the number of processing units (called hidden units)that must be used internally , to connect them to the input and output units already defined using **weighted connections** and to train the network with example data pairs, This concept of learning by example is externally important .As we shall see a significant advantage of an **ANS** approach to solving a problem is that we need not have a well –defined process for algorithmically converting an input to an output .Rather, all that we need for most networks is a collection of representative examples of the desired translation. The **ANS** than adapts itself to reproduce the desired outputs when presented with the example inputs.

In addition, as our example network illustrates, an **ANS** is robust in the sense that it will respond with an output even when presented with input that it has never seen before, such as patterns containing noise. If the input noise has not obliterated the image of the character, the network will produce a good guess using those portions of the image that where not obscured and the information that it has stored about how the character are supposed to look. the inherent ability to deal with noisy or obscured patterns is a significant advantage of an **ANS** approach over a traditional algorithmic solution .It also illustrates a neural network maxim: the power of an **ANS** approach lies not necessarily in the elegancy of the particular solution, but rather in the generality of the network to find its own solution to particular problems given only examples of the desired behavior

Once our network is trained adequately, we can show images of numerals written by people whose writing was not used to train the network .If the training as been adequate the information propagating through the network will result in a single element at the output having binary one value and that unit will be one that corresponds to the neural network that has written

As an example of how we might benefit from this separation considers a system that utilizes software simulation of a neural network has part of its programming. In this case, the network would be modeled in the host computer system as set of data structures that represents the current state of network.

The process of training the network is simply a matter of altering the connection weights systematically to encode the desired input –output relationship. If we code the network simulator such that the data structures used by the network are allocated dynamically, and are initialized by reading of connection-weight the data from the disc five, we can also create a network simulator with a similar structure in another offline computer system. When the on-line system must change to satisfy new operational requirements, we can develop the new connection weights off-line by training the network simulator in the remote system. Later; we can update the operational system by simply changing the connection-weight initialization file from the previous version produced by the off-line system.

These examples hint at the ability of the neural network to deal with complex pattern recognition problem, but they are by no means indicative of the limits of the technology. Finally the distinction made between the natural and artificial system is intentional. We cannot overemphasize the fact that the **ANS** models we will examine bear only a perfunctory resemblance to their biological counterparts. What is important about these models is that they all exhibit the useful behaviors of learning, recognizing, and applying relationships between objects and patterns of object in the real world. In this regard they provide us with a whole new set of tools that we can use to solve difficult problems.

The back propagation approach

Problems such as noisy image-to –ASCII example are difficult to solve by the computer due to the incompatibility between the machine and the problem. Mathematical process is not what is needed to recognize complex patterns in noisy environments. In fact, an algorithmic search

of even a relatively small input space can prove to be time consuming. The problem is the sequential nature of the computer itself. In most cases, the time required by the computer to perform each instruction is so short that the aggregate time required for even a large program is insignificant to the human users. However, for applications that must search through a large input space, or attempt to correlate all possible

Backpropagation Algorithm:

1)Let A be the number of units in the input layer, as determined by the length of the training input vectors. Let C be the number of units in the output layer. Now choose B, the number of units in the hidden layer. The input and hidden layers each have an extra unit used for thresholding; therefore, the units in these layers will sometimes be indexed by the ranges (0,…..,A) and (0,…..,B). We denote the activation levels of the units in the input layer by xj , in the hidden layer by hj , and in the output layer by oj . Weights connecting the input layer to the hidden layer are denoted by w1ij, where the subscript i indexes the input units and j indexes the hidden units. Likewise, weights connecting the hidden layer to the output layer are denoted by w2ij, with i indexing to hidden units and j indexing output units.

2)Initialize the weights in the network. Each should be set randomly to a number between -0.1 and 0.1. W_{ij} = random(-0.1,0.1) for all i =0,…..A, j = 1,……B W_{ij} = random(-0.1,0.1) for all i =0,…..B, j = 1,……C

3)Initialize the activations of the network. The values of these thresholding units should never change. a)X_0= 1.0 b)h_0 = 1.0

4)Choose an input-output pair. Suppose the input vector is X_i and the target output vector is Y_i Assign activation levels to the input units.

5)Propagate the activations from the units in the input layer to the units in the hidden layer using the activation functions

$$\Delta h_j = \frac{1}{1+e^{-\sum_{i=0}^{B} W1_{ij} h_i}} \quad for\ j = 1,...,C$$

Note that i ranges from 0 to A. W_{10j} is the thresholding weight for hidden unit j (its propensity to fire irrespective of it inputs).X_0 is always 1.0.

6)Propagates the activations from the units in the hidden layer to the units in the output layer

$$h_j = \frac{1}{1+e^{-\sum_{i=0}^{B} W2_{ij} h_i}} \quad for\ j = 1,...,C$$

Again, the thresholding weight w_{20j} for output unit j plays a role in the weighted summation. h_0 is always 1.0.

7)Compute the errors of the units in the output layer denoted $\delta 2_j$. Error is based on the network's actual output (O_j) and the target output (Y^i)

$$\delta 2_j = o_j(1-o_j)(y_j - o_j) \quad for\ all\ j = 1,...B$$

8)Compute the errors in the units in the hidden layer, denoted $\delta 1_j$.

$$\Delta 1_j = h_j(1-h_j)\sum_{i}^{c} \delta 2_i \times w2_{ji} \quad for\ all\ j = 1,...B$$

$$\Delta w1_{ij} = \eta \bullet \delta 1_j \bullet h_i \quad for\ all\ i = 0,...,A,\ j = 1,...,B$$

9)Adjust the weights between the hidden layer and the output layer. The learning rate denoted is denoted by $\eta\ \eta$; its functions is in the same as in perception learning. A reasonable value of η is 0.35.

$$\Delta w2_{ij} = \eta \bullet \delta 2_j \bullet h_i \quad for\ all\ i = 0,...,B,\ j = 1,...,C$$

10)Adjust the weights between the input layer and the hidden layer.

$$\Delta w1_{ij} = \eta \bullet \delta 1_j \bullet h_i \quad for\ all\ i = 0,...,A,\ j = 1,...,B$$

11)Go to step 4 and repeat. When all the inputs-output pairs have been presented to the network, one epoch has been completed.

Repeat steps 4 to 10 for as many epochs as desired.

BPN Training Algorithm

The following description tends to assume a pattern classification problem, since that is where the BP network has its greatest strength. However, you can use back-propagation for many other problems as well, including compression, prediction and digital signal processing. When you present your network with data and find that the output is not as desired, what will you do? The answer is obvious: we will modify some connection weights. Since the network weights are initially random, it is likely that the initial output value will be very far from the desired output.

We wish to improve the behavior of the network. Which connection weights must be modified, and by how much, to achieve this

objective? To put it another way, how do you know which connection is responsible for the greatest contribution to the error in the output? Clearly, we must use an algorithm which efficiently modifies the different connection weights to minimize the errors at the output. This is a common problem in engineering; it is known as optimization. The famous LMS algorithm was developed to solve a similar problem; however the neural network is a more generic system and requires a more complex algorithm to adjust the many network parameters. One algorithm which has hugely contributed to neural network fame is the back-propagation algorithm. The principal advantages of back-propagation are simplicity and reasonable speed (though there are several modifications which can make it work faster). Back-propagation is well suited to pattern recognition Problems. The training algorithm for a BPN consists of the following steps:(i) Selection and Preparation of Training Data (ii) Modification of the neuron connection weights (iii) Repetition (iv) Running (v) Hazards.

Selection and Preparation of Training Data

A neural network is useless if it only sees one example of a matching input/output pair. It cannot infer the characteristics of the input data for which you are looking for from only one example; rather, many examples are required. This is analogous to a child learning the difference between different types of animals -the child will need to see several examples of each to be able to classify an arbitrary animal. If they are to successfully classify birds they will need to see examples of sparrows, ducks, pelicans and others so that he or she can work out the common characteristics which distinguish a bird From other animals (such as feathers, beaks and so forth). It is also unlikely that a child would remember these differences after seeing them only once - many repetitions may be required until the information `sinks in'. It is the same with neural networks. The best training procedure is to compile a wide range of examples which exhibit all the different characteristics you are interested in. It is important to select examples which do not have major dominant features which are of no interest to you, but are common to your input data anyway. One famous example is of the US Army `Artificial Intelligence' tank classifier. It was shown examples of Soviet tanks from many different distances and angles on a bright sunny day, and examples of US tanks on a Cloudy day. Needless to say it was great at classifying weather, but not so good at picking out enemy tanks. If possible, prior to training, add some noise or other randomness to your example (such as a random scaling factor). This helps to account for noise and natural variability in real data, and tends to produce a more reliable network. If you are using a standard unscaled sigmoid node transfer function, please note that the desired output must never be set to exactly 0 or 1. The reason is simple: whatever the inputs, the outputs of the nodes in the hidden layer are restricted to between 0 and 1 these values are the asymptotes of the function. To approach these values would require enormous weights and/or input values, and most importantly, they cannot be exceeded. By contrast, setting a desired output of 0.9 allows the network to approach and ultimately reac*h* this value from either side, or indeed to overshoot. This allows the network to converge relatively quickly. It is unlikely to ever converge if the desired outputs are set too high or too low. Once again, it cannot be overemphasized: a neural network is only as good as the training data! Poor training data inevitably leads to an unreliable and unpredictable network.

Repetition : Since we have only moved a small step towards the desired state of a minimized error, the above procedure must be repeated many times until the MSE drops below a specified value. When this happens, the network is performing satisfactorily, and this training session for this particular example has been completed. Once this occurs, randomly select another example, and repeat the procedure. Continue until you have used all of your examples many times.

Running: Finally, the network should be ready for testing. While it is possible to test it with the data you have used for training, this isn't really telling you very much. Instead, get some real data which the network has never seen and present it at the input. Hopefully it should correctly classify, compress, or otherwise process the data in a satisfactory way.

Hazards:A consequence of the back-propagation algorithm is that there are situations where it can get `stuck'. Think of it as a marble dropped onto a steep road full of potholes. The potholes are `local minima' - they can trap the algorithm and prevent it from descending further. In the event that this happens, you can resize the network or try a different starting point. Some enhancements to the BP algorithm have been developed to get around this - for example one approach adds a *momentum* term, which essentially makes the marble

heavier - so it can escape from small potholes. Other approaches may use alternatives to the Mean Squared Error as a measure of how well the network is performing.

Existing System: Before the age of the computer, there were many mathematical problems that humans could not easily solve, or more precisely humans were too slow in solving. Computers enabled these often simple but slow and tedious tasks to be performed quickly and accurately. The first problems solved with computers were calculating equations to resolve important physical problems, and later displaying a nice GUI, making word processors and so on. However, there are many common tasks which are trivial for humans to perform yet which are extremely difficult to formulate in a way that a computer may easily solve. These include: (i)Signal processing such as pattern recognition, voice recognition, image processing etc., (ii)Compression, (iii)Data reconstruction (e.g. classification where part of the data is missing), (iv)Data mining, (v)Data simplification

Template Matching

Earlier techniques for pattern recognitions, include the technique of Template Matching. In this technique the patterns are just matched together as a human compare two structures with their exact features & characteristics matching. Template Matching are oversensitive to shift in position and distortions in shape of the stimulus patterns, and it is necessary to normalize the position and the shape of stimulus pattern beforehand. A good method for normalization has not been developed yet. Therefore, the finding of an algorithm for character recognition which can cope with shift in position and distortion has long been desired. In this project, we implement an algorithm which gives an important solution to this problem. The algorithm used here can be realized with a multilayered network consisting of neuron like cells. It is organized by supervised learning and acquires the ability for correct character recognition. So, naturally, scientists, engineers and mathematicians tried to make an intellectual abstraction which would enable a computer work in a similar way to that in which the human brain works – a neural network. This paper explains the principles of the neural network, and will show the relationship with a biological neural network. After that, when the basics are well understood, you will be shown a famous neural network topology called a multi-layer perception, to be trained with the Feed-forward Back-Propagation Network (BPN) algorithm. Finally, we will explain you how to use a simple and lightweight java library which implements an arbitrary 3-layer BPN and so can be used to solve some of the complicated problems described above.

Specification : To develop a system capable of recognizing handwritten characters or symbols, inputted by the means of a mouse. The system provides means for training the input characters first, then there is a classification option where the patterns or symbols that have already been trained should be fed, in order to recognize it.

There are full options for the users, like

1. To load a default set of patterns which is already present in the system, which can either be trained or the default training file can be loaded. After which the recognition takes place.
2. To classify a line of text.
3. Options to either load a pattern file, trained or untrained & finally the option to create new patterns by the user to train & classify.

Weight initialization

Set all weights and node threshold to small random numbers. Note that the node threshold is negative of the weight from the bias unit (whose activation level is fixed at 1)

Calculation of activation

1.The activation level of an input unit is determined by the instance presented to the network.

2.The activation level O_j of a hidden and output unit id determined by .is a sigmoid function

$$O_j=F(\Sigma W_{ji}Oi-\theta j)$$

Weight training

1. Start at the output units and work backward to the hidden layers recursively.Adjust weights by

$$W_{ji}(t+1)=W_{ji}(t)+W_{ji}$$

Where $W_{ji}(t)$ is the weight from unit to unit at time t and Wji is the weight adjustment.

2. The weight change is computed by

$$\Delta W_{ji}= \eta\delta_j O_i$$

Where η is a trial-independent learning rate and Dj is the error gradient at unit j. Convergence is sometimes faster by adding a momentum term:

$$W_{ji}(t+1)=W_{ji}(t) + \eta\ \delta_j O_i+[W_{ji}(t)-W_{ji}(t-1)]$$

Where $0<\alpha<1$

3. The error gradient is given by :For the output units:

$$\delta_j=O_j(1-O_j)(T_j-O_j)$$

Where T_j is the desired (target) output activation and Oj is the actual activation at output unit j.For the hidden units:

$$\delta_j = O_j(1-O_j)\ \Sigma\delta kWkj$$

where δ_k is the error gradient at unit k to which a connection points from hidden unit j.

4. Repeat iterations until convergence in terms of the selected error criteria. An iteration includes presenting an instance, calculating activations, and modifying weight's

CONCLUSION:

The BPN network designed proposed has the ability to recognize stimulus patterns without affecting by shift in position not by a small distortion in shape of input pattern. It also has a function of organization, which processes by means of Supervised Learning. If a set of input patterns are repeatedly presented to it, it gradually acquires the ability to recognize these patterns. It is not necessary to give any instructions about the categories to which the stimulus patterns should belong. The performance of the network has been demonstrated by simulating on a computer. We do not advocate that the network is a complete model for the mechanism of character recognition in the brain, but we propose it as a working design for some neural mechanisms of visual pattern recognition. It is conjectured that, in the Human Brain the process of recognizing familiar patterns such as alphabets of our native Language differs from that of recognizing unfamiliar patterns such as foreign alphabets, which we have just begun to learn. The design of information processing proposed in this project is of great use not only as an inference upon the mechanism of brain but also to the field of Engineering. One of the largest and longstanding difficulties is in designing a pattern recognizing machine has been the problem how to cope with the shift in position and the distortion in the shape of the input patterns. The network proposed in this paper gives a partial solution to this difficulty. This principle can also be applied to auditory information processing such as Speech Recognition.

REFERENCES:

[1]Aha, D. W., Kibler, D. & Albert, M. K. (1991) Instance-based learning algorithms. *Machine Learning* **6**(1), 37–66.

[2]Angluin, D. & Valiant, L. G. (1979) Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and System Sciences* **18**, 155–193.

[3]Anthony, M. & Shawe-Taylor, J. (1993) A result of Vapnik with applications. *Discrete Applied Mathematics* **47**, 207–217.

[4]Atkeson, C. G. (1991) Using locally weighted regression for robot learning. In *Proceedings of the IEEE Conference on Robotics and Automation (Sacremento, CA, 1991)*, pp. 958– 963. IEEE Press.

[5]Brodley, C. E. & Utgoff, P. E. (1995) Multivariate decision trees. *Machine Learning* **19**, 45–77.

[6]Buntine, W. & Niblett, T. (1992) A further comparison of splitting rules for decision-tree induction. *Machine Learning* **8**, 75–86.

[7]Burnell, L. & Horvitz, E. (1995) Structure and chance: Melding logic and probability for software debugging. *Communications of the ACM* **38**(3), 31–41, 57.

[8]Catlett, J. (1991) On changing continuous attributes into ordered discrete attributes. In *Proc of the European Working Session on Learning – EWSL-91*, ed. Y. Kodratoff, pp. 164–178. Berlin: Springer.

[9]Cesa-Bianchi, M., Freund, Y., Helmbold, D. P., Haussler, D., Schapire, R. E. & Warmuth, M. K. (1993) How to use expert advice. In *Proc. of the Twenth-Fifth ACM Symposium on the Theory of Computing (San Diego, CA, 1993)*, pp. 382–391. New York: ACM Press.

[10]Cesa-Bianchi, M., Freund, Y., Helmbold, D. P., Haussler, D., Schapire, R. E. & Warmuth, M. K. (1996) How to use expert advice. *Journal of the ACM* .

[11]Cost, S. & Salzberg, S. (1993) A weighted nearest neighbor algorithm for learning with symbolic features.*Machine Learning* **10**, 57–78.

[12]Cover, T. M. (1968) Capacity problems for linear machines. In *Pattern Recognition*, ed. L. Kanal, pp. 283–289. Thompson.

[13]Craven, M. W. & Shavlik, J. W. (1996) Extracting tree-structured representations of trained networks. In Touretzky *et al.* (1996), pp. 24–30. ISBN 0-262-20107-0.

[14]Drucker, H. & Cortes, C. (1996) Boosting decision trees. In Touretzky *et al.* (1996), pp. 479–485. ISBN 0-262-20107-0.

[15]Drucker, H., Cortes, C., Jaeckel, L. D., LeCun, Y. & Vapnik, V. (1994) Boosting and other ensemble methods. *Neural Computation* **6**(6), 1289–1301.

[16]Elomaa, T. & Rousu, J. (1996) Finding optimal multi-splits for numerical

attributes in decision tree learning. NeuroCOLT Technical Report Series NC-TR-96-041, Department of Computer Science, University of Helsinki.

[17]Fayyad, U. M. & Irani, K. B. (1993) Multi-interval discretization of continuous-valued attributes in decision tree generation. In *Proceedings of the the Thirteenth International Joint Conference on Artificial Intelligence (Chambery, France, 1993)*, pp. 1022–1027. San Francisco: Morgan Kaufmann.

[18]Freund, Y. (1990) Boosting a weak learning algorithm by majority. In *Proceedings of the Third Workshop on Computational Learning Theory*, pp. 202–216. Morgan Kaufmann.

[19]Freund, Y. (1995) Boosting a weak learning algorithm by majority. *Info and Computation* **121**(2), 256–285.

[20]Freund, Y. & Schapire, R. E. (1995) A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pp. 23–37. Springer.

[21]Freund, Y. & Schapire, R. E. (1996b) Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*.

[22]Fukunaga, K. & Flick, T. E. (1984) An optimal global nearest neighbor metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**, 314–318.

Author's Biography

Srinivasa Kumar Devireddy received the B.E. degree in Computer Science & Engineering from Karnataka University, Dharwad in 1992 and M.S. degree in Software Systems from Birla Institute of Technology and Science, Pilani in 1995. He is currently working as a faculty member in the department of Computer Science & Engineering, Nalanda Institute of Engineering & Technology, Guntur. He is a member of IEEE. He has guided many projects in the area of Computer Science, image processing and content based Image Retrieval. His research interests are in the areas of Biometrics, Image Processing and Content Based Image Retrieval.

AppaRao Settipalli received the Diploma in ECE from ETI AF, Jalahalli, Bangalore in the year 1984 and he received the B.E. (Electronics & Tele communication Engg.), ETI AF, Jalahalli, Banglore in the year 2001. Awarded MBA degree with the specialisation HR & Marketing from Acharya Nagarjuna University, Guntur in the year 2007.He is having 26 Years of Combined experience in Industry, Teaching, and Administration. He is member of various professional bodies like ISTE, IETE, IEEE & ACM and Currently he is working as Administrative Officer in NIET, Sattenapalli.