SATHYABAMA UNIVERSITY

SECX1048 FUNDAMENTALS OF FUZZY LOGIC AND ARTIFICIAL NETWORK

UNIT III FUNDAMENTALS OF ANN

Fundamentals of ANN:

Neural networks can learn and are actually taught instead of being programmed. Artificial Intelligence techniques such as neural networks, genetic algorithms and fuzzy logic are among the most powerful tools available for detecting and describing subtle relationships in massive amounts of seemingly unrelated data.

Teaching mode can be supervised or unsupervised. Neural Networks learn in the presence of noise. Neural computing is an information processing paradigm, inspired by biological system, composed of a large number of highly interconnected processing elements(neurons) working in unison to solve specific problems.

Artificial neural networks (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.

Neural networks have performed successfully where other methods have not, predicting system behavior, recognizing and matching complicated, vague, or incomplete data patterns. Apply ANNs to pattern recognition, interpretation, prediction, diagnosis, planning, monitoring, debugging, repair, instruction and control.

Some of the applications are :

- Biomedical Signal Processing
- Biometric Identification
- System Reliability
- Business
- Spiral Inductor Modeling
- Target Tracking

Biological Neuron:

Animals are able to react adaptively to changes in their external and internal environment, and they use their nervous system to perform these behaviours. An appropriate model/simulation of the nervous system should be able to produce similar responses and behaviours in artificial systems. The nervous system is build by relatively simple units, the neurons, so copying their behavior and functionality should be the solution.

The human brain consists of a large number; more than a billion of neural cells that process information. Each cell works like a simple processor. The massive interaction between all cells and their parallel processing only makes the brain’s abilities possible. The typical biological neuron is as shown in the below figure 1.

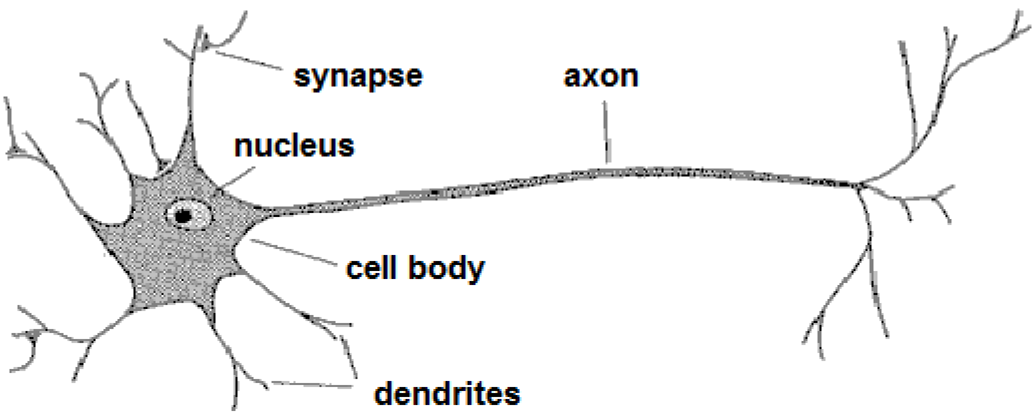


Fig.1. Biological Neuron

Dendrites are branching fibres that extend from the cell body or soma. Soma or cell body of a neuron contains the nucleus and other structures, support chemical processing and production of neurotransmitters. The spikes travelling along the axon of the pre-synaptic neuron trigger the release of neurotransmitter substances at the synapse.

The neurotransmitters cause excitation or inhibition in the dendrite of the post-synaptic neuron. The integration of the excitatory and inhibitory signals may produce spikes in the post-synaptic neuron. The contribution of the signals depends on the strength of the synaptic connection.

Axon is a singular fiber carries information away from the soma to the synaptic sites of other neurons (dendrites ans somas), muscels, or glands. Axon hillock is the site of summation for incoming information. At any moment, the collective influence of all neurons that conduct impulses to a given neuron will determine whether or n ot an action potential will be initiated at the axon hillock and propagated along the axon. Myelin sheath consists of fat-containing cells that insulate the axon from electrical activity. This insulation acts to increase the rate of transmission of signals.

A gap exists between each myelin sheath cell along the axon. Since fat inhibits the propagation of electricity, the signals jump from one gap to the next. Nodes of Ranvier are the gaps (about 1 μm) between myelin sheath cells. Since fat serves as a good insulator, the myelin sheaths speed the rate of transmission of an electrical impulse along the axon. Synapse is the point of connection between two neurons or a neuron and a muscle or a gland. Electrochemical communication between neurons take place at these junctions. Terminal buttons of a neuron are the small knobs at the end of an axon that release chemicals called neurotransmitters.

Some numbers related to human brain:

The human brain contains about 10 billion nerve cells (neurons)

Each neuron is connected to the others through 10000 synapses

Properties of the brain:

It can learn, reorganize itself from experience

It adapts to the environment

It is robust and fault tolerant

Artificial Models of Neuron:

An artificial neuron is a mathematical function conceived as a simple model of a real (biological) neuron. This is a simplified model of real neurons, known as a Threshold Logic Unit. There are two popular models of ANN, namely, MP Model and Perceptron model.

The McCulloch-Pitts Neuron:

- The first mathematical model of a neuron
- Binary activation: fires (1) or not fires (0)
- Excitatory inputs: the a's, and Inhibitory inputs: the b's
- Unit weights and fixed threshold θ
- Absolute inhibition

Basic Elements:

Neuron consists of three basic components: weights, thresholds and a single activation function.

An output line transmits the result to other neurons. In other words, The input to a neuron arrives in the form of signals. The signals build up in the cell. Finally the cell discharges (cell fires) through the output. The cell can start building up signals again. The equation for the output of a McCulloch-Pitts neuron as a function of 1 to n inputs is written as Output = sgn (n i=1 Input i – φ) , where φ is the neuron's activation threshold. If n i=1 Input i ≥ φ then Output = 1 If n i=1 Input i < φ then Output = 0.

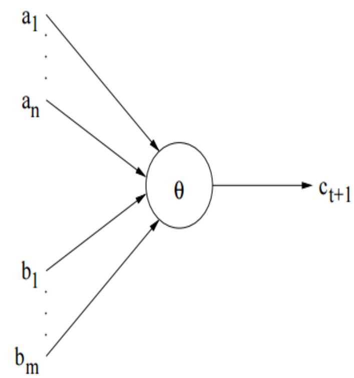


Fig.2. MP Model of Artificial Neuron

The typical logical operations using MP model are:

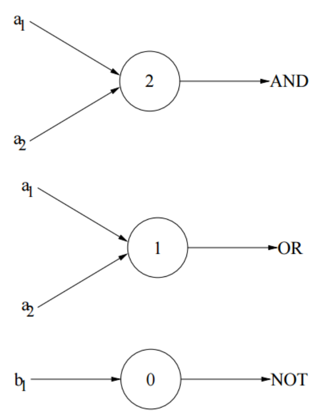


Fig.3. Logical operations using MP model

Any task or phenomenon that can be represented as a logic function can be modelled by a network of MP-neurons. {OR, AND, NOT} is functionally complete . Any Boolean function can be implemented using OR, AND and NOT . Canonical forms: CSOP or CPOS forms.

In this McCulloch-Pitts neuron model, the missing features are:

Non-binary input and output

Non-linear summation

Smooth thresholding

Stochastic

Temporal information processing.

Problems with MP-neurons are, weights and thresholds are analytically determined. Cannot learn, very difficult to minimize size of a network.

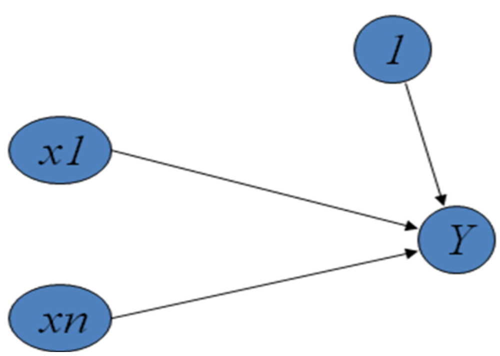
Types of ANN- Properties – Different learning rules:

Artificial neural network types vary from those with only one or two layers of single direction logic, to complicated multi–input many directional feedback loops and layers. On the whole, these systems use algorithms in their programming to determine control and organization of their functions. Most systems use "weights" to change the parameters of the throughput and the varying connections to the neurons. Artificial neural networks can be autonomous and learn by input from outside "teachers" or even self-teaching from written-in rules.

Two types of networks exist; namely, feed forward neural networks and Recurrent Neural Networks.

Single Layer Feedforward NN

A single layer network has one layer of connection weights. Often the units can be distinguished as input units which receive signals from the outside world and output units from which the response of the net can be read. In a typical single layer net the input units are fully connected to output units but are not connected to other units and the output are not connected to other output units.



$$net = b + \sum_{i=1}^{n} x_i w_i$$

Fig.4. Single layer model

Bias *b* is treated as the weight from a special unit with constant output 1.

Multilayer feedforward NN

A multilayer net is a net with one or more layers of nodes between the input units and the output units. Typically there is a layer of weights between two adjacent levels of units. Multilayer nets can solve more complicated problems that can single layer nets, but training may be more difficult.

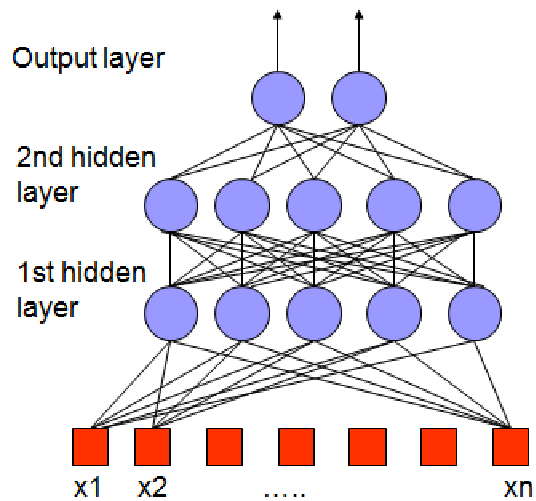


Fig.5. Multilayer model

Competitive layer: - A competitive layer forms a part of a large number of neural networks. The competitive interconnections have weights of -ε.

Tasks to be solved by artificial neural networks:

- Controlling the movements of a robot based on self-perception and other information (e.g., visual information);
- Deciding the category of potential food items (e.g., edible or non-edible) in an artificial world;
- Recognizing a visual object (e.g., a familiar face);
- Predicting where a moving object goes, when a robot wants to catch it.

- A mathematical model to solve engineering problems
- Group of highly connected neurons to realize compositions of non linear functions

Tasks

- Classification
- Discrimination
- Estimation

Different Learning Rules

- Supervised learning
- Unsupervised learning
- Reinforced learning
- Hebbian learning
- Gradient descent learning
- Competitive learning
- Stochastic learning

Supervised learning: Every input pattern that is used to train the network is associated with an output pattern which is the target or the desired pattern. A teacher is assumed to be present during the training process, when a comparison is made between the network's computed output and the correct expected output, to determine the error. The error can then be used to change network parameters, which result in an improvement in performance. Supervised networks are universal approximators (Non recurrent networks).

Type of Approximators :

- Linear approximators : for a given precision, the number of parameters grows exponentially with the number of variables (polynomials)
- Non-linear approximators (NN), the number of parameters grows linearly with the number of variables

Unsupervised learning: In this learning method the target output is not presented to the network.It is as if there is no teacher to present the desired patterns and hence the system learns of its own by discovering and adapting to structural features in the input patterns.

Reinforced learning: In this method, a teacher though available, doesnot present the expected answer but only indicates if the computed output correct or incorrect. The information provided helps the network in the learning process.

Hebbian learning: This rule was proposed by Hebb and is based on correlative weight adjustment. This is the oldest learning mechanism inspired by biology. In this, the input-output pattern pairs (xi,yi) are associated by the weight matrix W, known as the correlation matrix.

Gradient descent learning: This is based on the minimization of error E defined in terms of weights and activation function of the network. Also it is required that the activation function employed by the network is differentiable, as the weight update is dependent on the gradient of the error E.

Competitive learning: In this method, those neurons which respond strongly to input stimuli have their weights updated. When an input pattern is presented, all neurons in the layer compete and the winning neurons undergo weight adjustment. Hence it is a winner-takes-all strategy. Stochastic learning: In this method, weights are adjusted in a probablistic fashion. An example is evident in simulated annealing the learning mechanism employed by Boltzmann and Cauchy machines, which are a kind of NN systems.

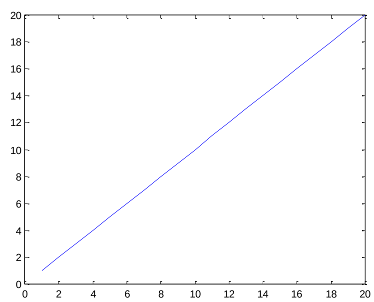
Types of Activation Functions:

A set of input connections brings in activations from other neuron. A processing unit sums the inputs, and then applies a non-linear activation function (i.e. squashing/transfer/threshold function). Sigmoidal neuron and Guassian neuron functions are :

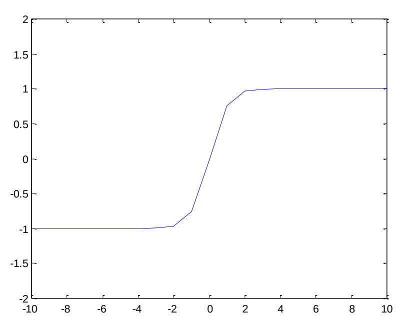
$$y = \frac{1}{1 + e^{-w^T x - a}}$$

$$y = e^{-\frac{||x-w||^2}{2a^2}}$$

Linear : y=x



Hyperbolic tangent



$$y = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Training ANN:

Training is nothing but learning. Training reduces the error and makes the ANN efficient. Whether our neural network is a simple Perceptron, or a much more complicated multilayer network with special activation functions, we need to develop a systematic procedure for determining appropriate connection weights. The general procedure is to have the network learn the appropriate weights from a representative set of training data. In all but the simplest cases, however, direct computation of the weights is intractable. Instead, we usually start off with random initial weights and adjust them in small steps until the required outputs are produced .

Perceptron Model

Perceptron model was proposed by Rosenblatt in 1958. In this model, Weights and thresholds can be determined analytically or by a learning algorithm. Continuous, bipolar and multiple-valued versions .Efficient minimization heuristics exist in this type.

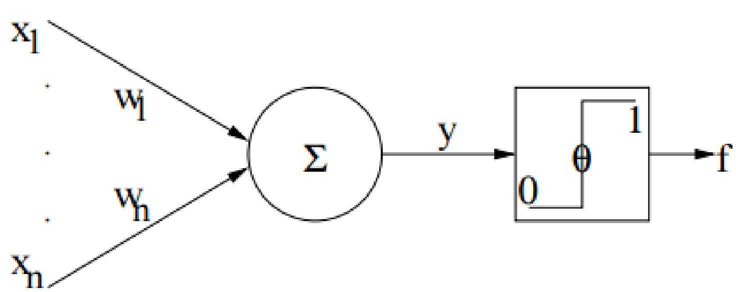


Fig.6. Perceptron Model of Artificial Neuron

Input: $\vec{x} = (x_0 = 1, x_1, \ldots, x_n)$

Weight: $\vec{w} = (w_0 = -\theta, w_1, \ldots, w_n)$, $\theta =$ bias

Net input: $y = \vec{w}\vec{x} = \sum_{i=0}^{n} w_i x_i$

We consider here a NN, known as the Perceptron, which is capable of performing pattern classification into two or more categories. The perceptron is trained using the perceptron learning rule. We will first consider classification into two categories and then the general multiclass classification later. For classification into only two categories, all we need is a single output neuron. Here we will use bipolar neurons. The simplest architecture that could do the job consists of a layer of N input neurons, an output layer with a single output neuron, and no hidden layers.

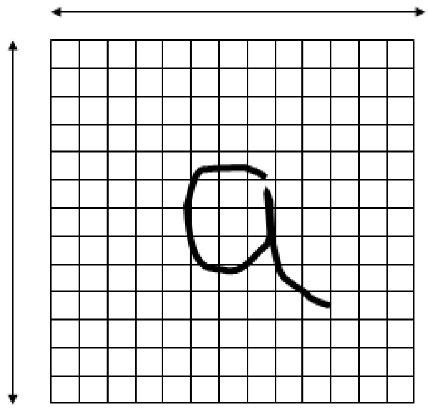
Training Algorithms:

Most of the algorithms used in training artificial neural networks employ some form of gradient descent, using backpropagation to compute the actual gradients. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction. The backpropagation training algorithms are usually classified into three categories: steepest descent (with variable learning rate, with variable learning rate and momentum, resilient backpropagation), quasi-Newton (Broyden-Fletcher-Goldfarb-Shanno, one step secant), Levenberg-Marquardt and conjugate gradient (Fletcher-Reeves update, Polak-Ribiére update, Powell-Beale restart, scaled conjugate gradient).

Training a neural network model essentially means selecting one model from the set of allowed models (or, in a Bayesian framework, determining a distribution over the set of allowed models) that minimizes the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and statistical estimation.

Problem Solving Using Learning Rules and Algorithms:

- Pattern recognition
  - Patterns: images, personal records, driving habits, etc.
  - Represented as a vector of features (encoded as integers or real numbers in NN)
  - Pattern classification:
    - Classify a pattern to one of the given classes
    - Form pattern classes
  - Pattern associative recall
    - Using a pattern to recall a related pattern
    - Pattern completion: using a partial pattern to recall the whole pattern
    - Pattern recovery: deals with noise, distortion, missing information



$2^{256\times256\times8} \approx 10^{158000}$ different images

Fig. 7. Image 256x256 pixels

This image consists of:

8 bits pixels values (grey level)

Necessary to extract features

Normalization

Inputs of the neural net are often of different types with different orders of magnitude (E.g. Pressure, Temperature, etc.)

It is necessary to normalize the data so that they have the same impact on the model

Center and reduce the variables

$$\overline{x_i} = \frac{1}{N}\sum_{n=1}^{N} x_i^n \longleftarrow \text{Average on all points}$$

$$\sigma_i^2 = \frac{1}{N-1}\sum_{n=1}^{N}\left(x_i^n - \overline{x_i}\right)^2 \longleftarrow \text{Variance calculation}$$

$$x_i^n = \frac{x_i^n - \overline{x_i}}{\sigma_i} \longleftarrow \text{Variables transposition}$$

Sometimes, the number of inputs is too large to be exploited

The reduction of the input number simplifies the construction of the model

Goal : Better representation of the data in order to get a more synthetic view without losing relevant information

Reduction methods (PCA, CCA, etc.)

A common criticism of neural networks, particularly in robotics, is that they require a large diversity of training for real-world operation. Any learning machine needs sufficient representative examples in order to capture the underlying structure that allows it to generalize to new cases. Dean A. Pomerleau, in his research presented in the paper "Knowledge-based

Training of Artificial Neural Networks for Autonomous Robot Driving," uses a neural network to train a robotic vehicle to drive on multiple types of roads (single lane, multi-lane, dirt, etc.). A large amount of his research is devoted to (1) extrapolating multiple training scenarios from a single training experience, and (2) preserving past training diversity so that the system does not become overtrained (if, for example, it is presented with a series of right turns – it should not learn to always turn right). These issues are common in neural networks that must decide from amongst a wide variety of responses, but can be dealt with in several ways, for example by randomly shuffling the training examples, by using a numerical optimization algorithm that does not take too large steps when changing the network connections following an example, or by grouping examples in so-called mini-batches.

Linear Separability Limitation and Its Over Comings

In Euclidean geometry, linear separability is a geometric property of a pair of sets of points. This is most easily visualized in two dimensions by thinking of one set of points as being colored blue and the other set of points as being colored red. These two sets are *linearly separable* if there exists at least one line in the plane with all of the blue points on one side of the line and all the red points on the other side. This idea immediately generalizes to higher-dimensional Euclidean spaces if line is replaced by hyper plane.

The problem of determining if a pair of sets is linearly separable and finding a separating hyperplane if they are arises in several areas. In statistics and machine learning, classifying certain types of data is a problem for which good algorithms exist that are based on this concept.

- If two classes of patterns can be separated by a decision boundary, represented by the linear equation then they are said to be linearly separable. The simple network can correctly classify any patterns.
- Decision boundary (i.e., *W, b* or *q*) of linearly separable classes can be determined either by some learning procedures or by solving linear equation systems based on representative patterns of each classes
- If such a decision boundary does not exist, then the two classes are said to be linearly inseparable.
- Linearly inseparable problems cannot be solved by the simple network , more sophisticated architecture is needed.

- Examples of linearly separable classes

Logical AND function: Patterns (bipolar) decision boundary

x1	x2	y	
-1	-1	-1	w1 = 1
-1	1	-1	w2 = 1
1	-1	-1	b = -1
1	1	1	q = 0
			-1 + x1 + x2 = 0

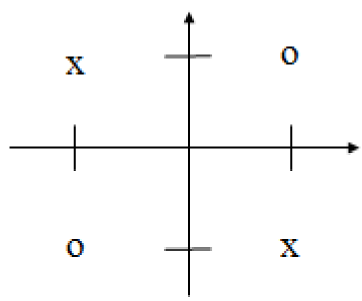
Logical OR function: Patterns (bipolar) decision boundary

x1	x2	y	
-1	-1	-1	w1 = 1
-1	1	1	w2 = 1
1	-1	1	b = 1
1	1	1	q = 0
			1 + x1 + x2 = 0

- Examples of linearly inseparable classes

Logical XOR (exclusive OR) function: Patterns (bipolar) decision boundary

x1	x2	y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1



No line can separate these two classes, as can be seen from the fact that the following linear inequality system has no solution because we have b < 0 from (1) + (4), and b >= 0 from (2) + (3), which is a contradiction. XOR can be solved by a more complex network with hidden units.

QUESTIONS FOR PRACTICE

PART A

1. What is a biological neuron?
2. What is learning?
3. Define artificial neuron?
4. What are the different types of ANN?
5. Brief about activation functions?
6. List out training algorithms?
7. What is training of a neural network?
8. Brief about XOR problem?

PART B

1. Explain about biologically inspired neuron in detail with neat diagram?
2. Discuss about different types of ANN models in detail?
3. Elaborate on the different model of artificial neuron?
4. Discuss about MP model of neuron?
5. Explain Perceptron model of neuron with neat diagrams?
6. Explain an example of using learning rules and algorithms?
7. Discuss about linear separability and elaborate about limitations and solution?