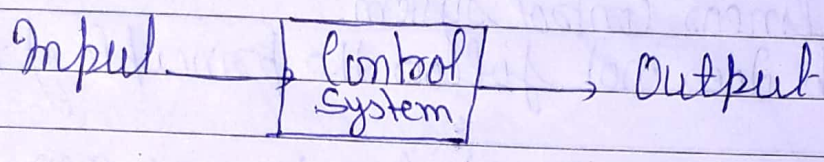
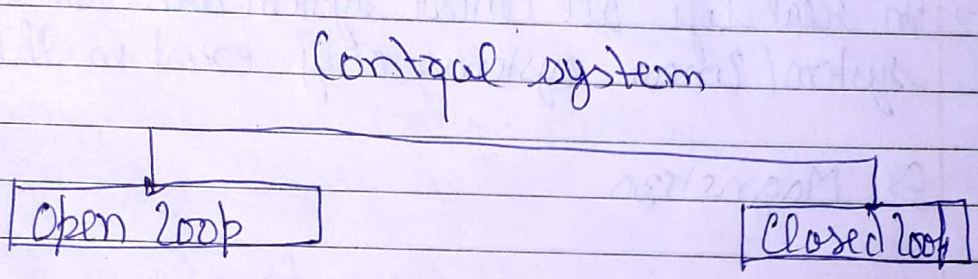
Date
Page No.

Control System

A control system is a system, which provide the desired response by controlling the output





There are various types of control system. Which can be broadly categorised as

1. Linear control system
2. Non linear control system.

(1) Linear control system:-

→ Linear control system obeys the principle of superposition.

→ Superposition theorem having two important properties.

(a) Homogeneity:-

→ A system is said to be homogeneous, if we multiply input with some constant (A) then the output will be also be multiplied by the same value of constant (A).

(b) Additivity

Input	output
a_1	b_1
a_2 a_1+a_2	b_2 b_1+b_2

Ex. Linear control system
→ A purely resistance network with a constant DC source

Non linear control system:
→ Which does not follow the principle of homogeneity

→ In real life all control system are non-linear system (Linear systems only exist in theory)

Ex Magnetizo

Linear controller:-

→ PID controller
→ Frequency Domain
- Time Domain approach.

→ State feedback controller
- Pole Placement
→ Linear Quadratic Regulator

→ Robust controller:
When plant is associated with parametric or unstructured uncertainties.

Non linear control

When the plant is non linear, there are certain popular control methodologies.

Non linear control Strategies:-

→ feedback linearization
→ Back Stepping
→ Sliding Mode Control.
→ Singular Perturbation Method.
→ Passivity Based Control.

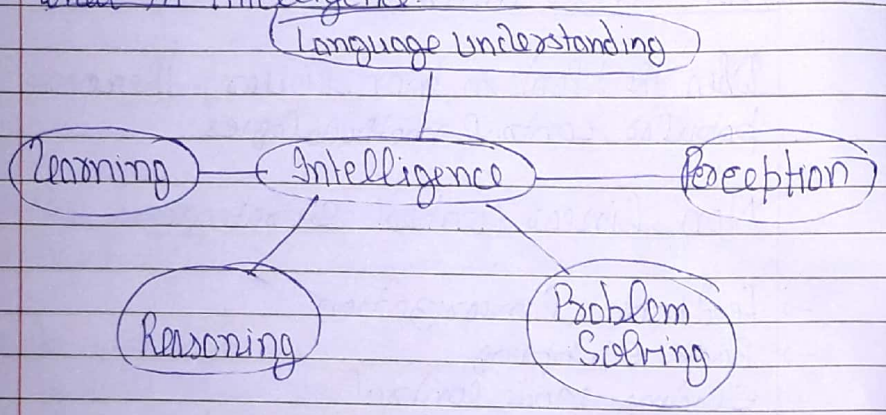
→ ~~We will not used these~~
We will not discussing in detail about these methodologies. but we will adapt these principles while designing intelligent control

~~Limit~~ Limitations of Nonlinear control strategies

→ Controller design in system specific.
→ No generalized tools for Analysis

→ fails
→ Unstructured Dynamics
→ friction Flexibility.
→ Parameter Uncertainties
→ Unknown dynamics.

What is intelligence:-



Intelligence:-

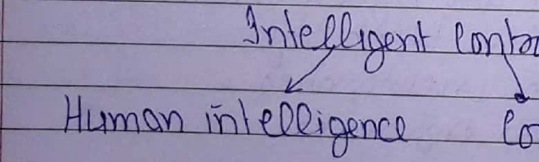
→ The ability to learn or understand or to deal with new or trying situation

→ the ability to acquire and apply knowledge and skills

Intelligent machine:-

An intelligent machine has following three functions

① Intelligently observed - Stochastic fields
② Intelligent prediction - System identification
③ Intelligent interaction → Adaptive control



Intelligent control describes the discipline where control methods are developed that attempt to emulate important characteristics of human intelligence.

→ These characteristics include adaptation and learning, planning under large uncertainty and coping with large amounts of data.

Why Intelligent control:

→ Should take care of model uncertainties
→ Should be adaptive to change in environment
→ Distributed in nature

Levels of intelligence:

→ Lower level
Ability to sense the environment

→ Higher level
Ability to recognize objects and events
To represent knowledge in a world model

→ Advanced:-
Capacity to perceive and understand, to choose wisely, and to act successfully under a large variety of circumstances so as survive and prosper in a complex and often hostile environment

Intelligent Control Strategies

Automata
Hybrid System
Petri nets
Neural nets
Fuzzy Logic
Evolution Algorithms.

Basic Concepts of Neural Networks

Neural Networks, which are simplified models of the biological neuron system, is a massively parallel distributed processing system made up of highly interconnected neural computing elements that have the ability to learn and thereby acquire knowledge and make it available for use.

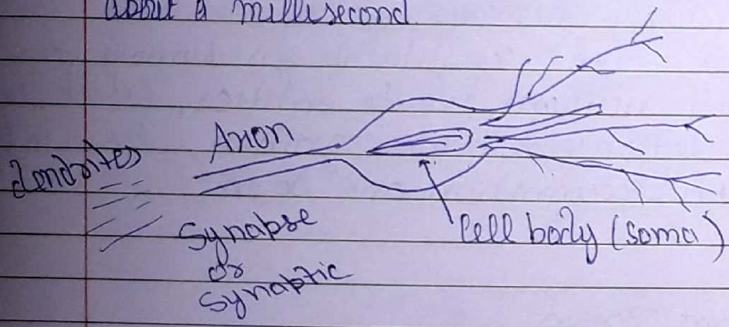
→ Neural Network are simplified imitations of the central nervous system.
The structural constituents of a human brain termed neurons are the entities, which perform computations such as cognition, logic inference pattern recognition and so on.

Humain Brain

→ Brain contains about 10^{10} basic units called neurons
→ Each neurons is connected to about 10^4 other neurons.
→ A neuron is a small cell that receives electro-chemical signals from its various sources and in turn responds by transmitting electrical impulses to other neurons.
→ An average brain weight about 1.5 kg and an average neuron has a weight of 1.5×10^{-9} grams.

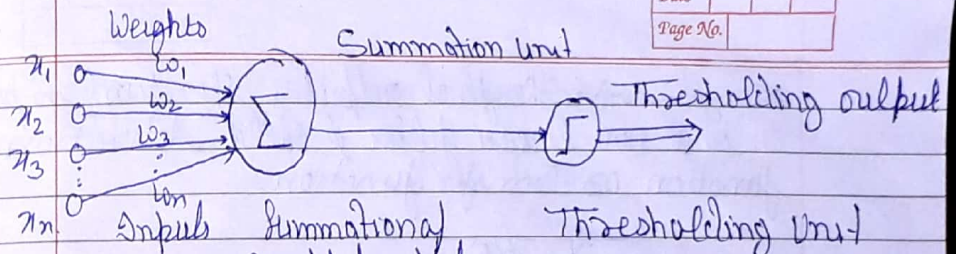
→ A neuron is composed of a nucleus - a cell body known as soma.
→ The soma are long irregularly shaped filaments

and attached to dendrites.
→ The dendrites behave as input channels
→ The dendrites received input
→ The another type of link attached to the soma is the Axon.
→ The Axon is electrically active and serves as an output channel.
→ Axon, which mainly appear on output cells are non-linear threshold devices which produce a voltage pulse called Action potential or spike that last for about a millisecond



~~Model of an Artificial Neuron:~~

→ The human brain is a highly complex structure as a massive, highly interconnected network of simple processing element called neurons
→ ~~Eve~~ However the behaviour of a neuron can be captured by a simple model as shown in figure.
→ Every component of the model bears a direct analogy to the actual constituent of a biological neuron and hence is termed as artificial neuron



Simple model of an artificial neuron

Here, $x_1, x_2, x_3 \dots x_n$ are the n inputs to the artificial neurons.
$w_1, w_2, w_3 \dots w_n$ are the weights attached to the input links

A biological neuron receives all inputs through the dendrites, sum them and produced an output if the sum is greater than a threshold value.
The input signals are passed on to the cell body through the synapse which may accelerate or retard an arriving signal.

→ An effective synapse which transmits a stronger signal will have a correspondingly larger ~~wei~~ weight while a weak synapse will have smaller weights. Thus weights are here are multiplicative factors of the inputs to account for the strength of the synapse. Hence, the total input I received by the soma of the artificial neuron is

$$I = w_1x_1 + w_2x_2 + \dots + w_nx_n$$
$$= \sum_{i=1}^{n} w_ix_i$$

To generate the final output y, the sum is passed on to a non-linear filter ϕ called Activation function, or Transfer function.

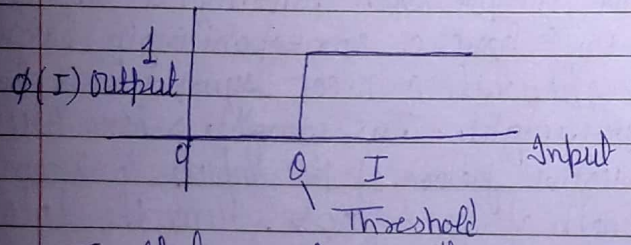
$$Y = \phi(I)$$

A very commonly used Activation function is the Thresholding function. In this, the sum is compared with a threshold value θ. If the value of I is greater than θ, then the output is 1 else it is zero.

$$Y = \phi\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

Where, ϕ is the step function known as Heaviside function and is such that

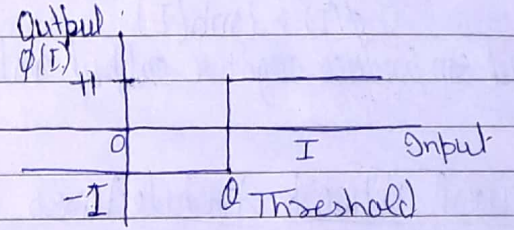
$$\phi(I) = \begin{cases} 1 & I > 0 \\ 0 & I \leq 0 \end{cases}$$



The output signal is either 1 or 0 resulting in the neuron being on or off.

Signum function :- / Quantizer function.

$$\phi(I) = \begin{cases} +1 & I > 0 \\ -1 & I \leq 0 \end{cases}$$



Signum function.

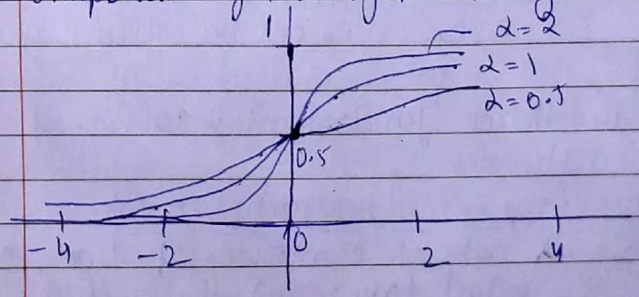
Sigmoidal function :-

This function is a continuous function that varies gradually between the asymptotic value 0 and 1 or -1 and +1 and is given by

$$\phi I = \frac{1}{1 + e^{-\alpha I}}$$

Where, α is the slope parameter, which adjust the abruptness of the function as it changes between the two asymptotic values.

→ Sigmoidal functions are differentiable, which is an important feature of NN theory.



Sigmoidal function

Hyperbolic tangent function

The function is given by

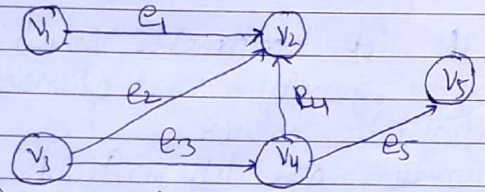
$$\phi(I) = \tanh(I)$$

and can produce negative output values

Neural Network Architectures

An Artificial Neural Network is defined as a data processing system consisting of a large number of simple highly interconnected processing elements (artificial neurons) in an architecture inspired by the structure of the cerebral cortex of the brain.

Example of directed graph or a digraph



Where $V = \{v_1, v_2, v_3, v_4, v_5\}$ are called vertices
$E = \{e_1, e_2, e_3, e_4, e_5\}$ are called Edges

There are three fundamental calles of neural networks

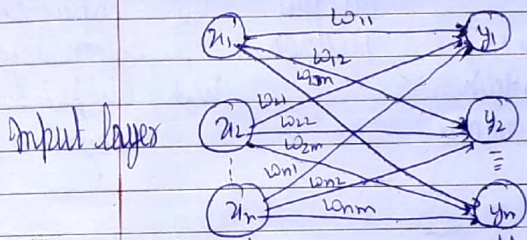
① Single layer feedforward Network

This type of network comprises of two layers namely the input layer and the output layer

→ The input layer neurons receive the input signal and the output layer neuron receive the output signals.

→ The synaptic links carrying the weights connected every input neuron to the output neuron but not vice-versa. Such a network is said to be feedforward type.

→ The input layer merely transmits the signals to the output layer. Hence the name single layer feedforward network.



Input layer — Output layer

Input neurons — Output neurons

$x_i \rightarrow$ Input neurons
$y_i \rightarrow$ Output neurons
$w_{ij} =$ Weights

Single layer feedforward network.

Multilayer feedforward Network

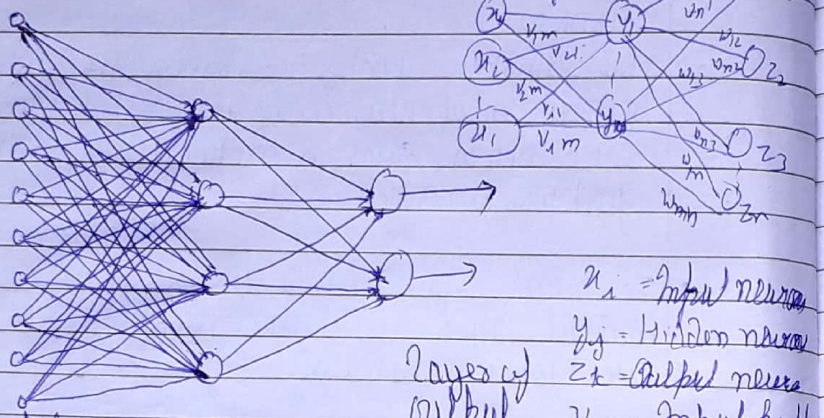
→ This network is made up of multiple layers.

→ In between the input and output ~~layer~~ or more ~~but then two~~ hidden layers are present.

→ By adding one or more hidden layers, the network is enabled to extract higher order statistics

→ The ability of hidden neurons to extract higher order statistics is particularly valuable when

the size of the input layer is large.

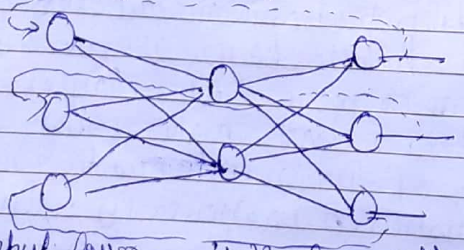
→ 

x_i = Input neuron
y_j = Hidden neuron
z_k = Output neuron
v_{ij} = Input-hidden layer weight
w_{jk} = Output-hidden weights

Input layer of source node | Layer of hidden neuron | Layer of output neuron

Recurrent Networks:

These networks differs from feedforward network architectures in the sense that there is atleast one feedback loop. Thus in these networks, for example, there could exist one layer with feedback connections.
There could also be neurons with self feedback links. i.e the output of a neuron is fed back into itself as input



Input layer | Hidden layer | output layer

Characteristics of Neural Network

→ The neural network exhibit mapping capabilities that is, they can map input patterns to their associated output pattern

→ The neural network learn by examples thus, NN architecture can be trained with known example of a problem before they are tested for their inference capability on unknown instance of the problem. They can, therefore, identify new objects previously untrained

→ The neural network posses the capability to generalize thus, they can predict new outcomes from past trend

→ The neural network are robust system and are fault tolerant. They can, therefore, recall full patterns from incomplete, partial or noisy pattern

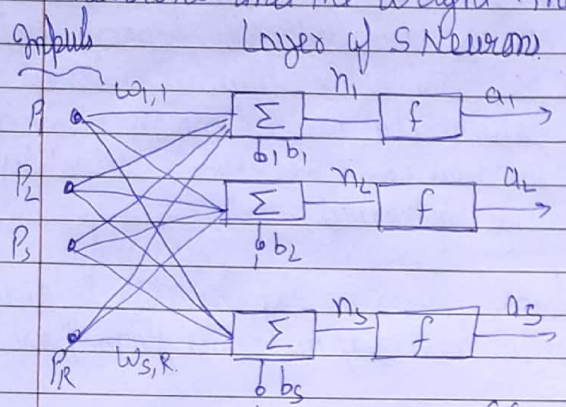
→ The neural network can process information in parallel at high speed, and in a distributed manner.

Date
Page No.

Network Architectures

A layer of Neurons:-

A single layer network of S neurons and each of the R input is connected to each of the neurons and the weight matrix now has S rows.



$$a = f(Wp + b)$$

The layer includes the weight matrix, the summer the bias vector b, the transfer function boxes and the output vector a.

Each element of the input vector P is connected to each neurons through the weight matrix W. Each neurons has a bias b_i, a summer, a transfer function f and an output a_i. Taken together, the output form the output vector a.

It is common for the number of inputs to a layer to be different from the number of neurons (i.e $R \neq S$)

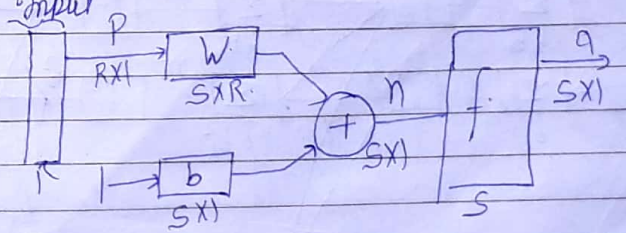
?

You might ask if all the neurons in a layer must have the same transfer function. The answer is no; you can define a single layer of neurons having different transfer functions by combining two of the network shown above in parallel. Both networks would have the same inputs and each network would create some of the output

The input vector elements enter the network through the weight matrix W:

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & & & \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

Fortunately, the S-neuron, R-input, one layer network also can be drawn in abbreviated notation



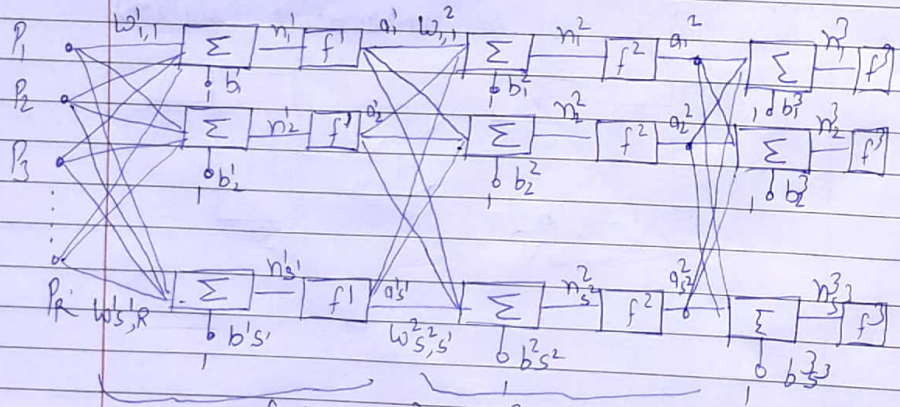
Multiple layers of Neurons:

Now consider a network with several layers. Each layer has its own weight matrix W, its own bias vector b, a net input vector n and an output vector a. We need to introduce some additional notation to distinguish between these layers

→ The weight matrix for the first layer is written as W^1, and the weight matrix for the second layer is written as W^2, W^3 is for third layer

There are R input, S^1 neurons in the first layer S^2 neurons in the second layer etc. The different layers can have different numbers of neurons

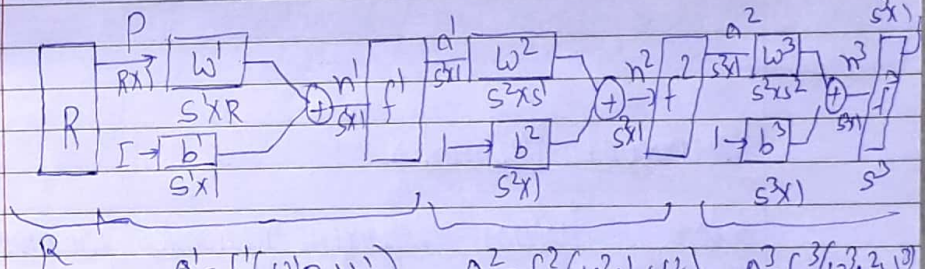
The output of the layer one & two are the input for layer two and three



$a^3 = f^3(W^3f^2(W^2f^1(W^1p + b^1) + b^2) + b^3)$

A layer whose output is the network output is called an output layer. The other layers are called hidden layers.

The same three layer network discussed previously also can be drawn using our abbreviated notation.



$a^1 = f^1(W^1p + b^1)$ $a^2 = f^2(W^2a^1 + b^2)$ $a^3 = f^3(W^3a^2 + b^3)$

$a^3 = f^3(W^2f^2(W^2f^1(W^1p + b^1) + b^2) + b^3)$

Three layer Network, Abbreviated Notation

The multilayer networks are more powerful than single-layer network. For instant, a two layer network having a sigmoid first layer and a linear second layer can be trained to approximate most functions arbitrarily well. Single layer networks cannot do this.

→ Most practical neural networks have just two or three layers. four or more layers are used rarely.

Biases:-

One can choose neurons with or without biases

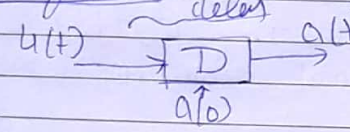
→ The biases gives the network an extra variable. and so you might expect that networks with biases would be more powerful than those without

→ A neuron without a bias will always have a net input n of zero when the network input P are zero. This may not be desirable and can be avoided by use of a bias

Recurrent Networks

Before we discuss recurrent networks, we need to introduce some simple building blocks

① Delay Block

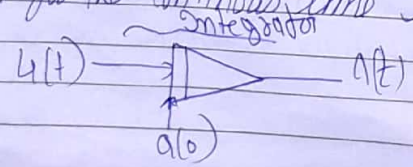


the delay output $a(t) = u(t-1)$

→ the output is the input delayed by one time step

② Integrator:

Another related building block which we will used for the continuous time recurrent network

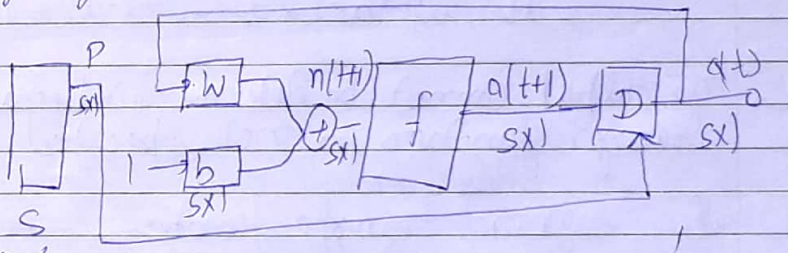


$$a(t) = \int_0^t u(t)\,dz + a(0)$$

$a(t)$ = integral output
$u(t)$ → integral input
$a(0)$ → initial condition

Recurrent Network

A recurrent network is a network with feedback some of its ~~input~~ output are connected to its input. This is quite different from the network that we have studied thus far, which were strictly feed forward with no backward connection



Initial condition

$a(0) = P \quad a(t+1) = \text{satlins}(Wa(t)+b)$

satlins is transfer function that saturates in the range $[-1, 1]$

In this particular network the vector P supplies the initial conditions (i.e $a(0) = P$) Then future outputs of the network are computed from previous output

$a(1) = \text{satlins}(Wa(0)+b)$, $a(2) = \text{satlins}(Wa(1)+b)$

Recurrent networks are potentially more powerful than feedback networks.

Date
Page No.

Q1. The input to a single-input neuron is 2.0, its weight is 2.3 and its bias is −3

(i) What is the net input to the transfer function
(ii) What is the neuron output
(iii) What will be the neuron output if it has the following transfer function

a. (a) Sigmoid
(b) Tangent

Solution:-

(i) The net input is given by

$$n = wp + b = (2.3)(2) + (-3) = 1.6$$

(ii) The output cannot be determined because the transfer function is not specified

(iii)(a) for sigmoid transfer function

$$a = \frac{1}{1+e^{-1.6}} = 0.8320$$

(b) for tangent transfer fun

$$a = \tanh(1.6) =$$

Date
Page No.

2. Given a two input neuron with the following parameters $b = 1.2$, $w = [3, 2]$ and $p = [-5\ 6]^T$. Calculate the neuron output for the following transfer function

1. A symmetrical hard limit transfer function (Signum)
2. A Sigmoid

$$n = wp + b = [3\ 2]\begin{bmatrix} -5 \\ 6 \end{bmatrix} + 1.2$$

$$= -1.8$$

(1) $a = \text{hardlims}(-1.8) = -1$

(2) $a = \text{sigmoid}(-1.8) = \tanh(-1.8) = \frac{1}{(1+e^{1.8})} = 0.$

(3) A single layer neural network is to have six inputs and two outputs. The output are to be limited to and continuous over the range 0 to 1. What can you tell about the network architecture specifically

(1) How many neurons are required
(2) What are the dimensions of the weight matrix?
(3) What kind of transfer function could be used
(4) Its bias required

Sol (i) Two neurons, one for each output, are required
(ii) The weigh matrix has two rows corresponding to the two neurons and six columns corresponding to six input

(iii) The logsig transfer function would be most appropriate.

(iv) Not enough information is given to determine a bias is required

Learning rules

Learning Rule:-

→ Learning rules mean a procedure for modifying the weights and biases of a network

→ The purpose of the learning rule is to train the network to perform some tasks. There are many types of neural network. learning rules. They fall into three broad categories:- ~~Supervised lea~~

Learning rules

- Supervised learning
- Reinforcement learning
- Unsupervised learning

Supervised learning:

→ In this 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 learning process, when a comparison is made between the network's computed output and the passed expected output, to determine the errors. The error can then be used to change network parameters. which result in an improvement in performance

Ex The learning rule is provided:
$\{P_1, t_1\}, \{P_2, t_2\}$ ——— $\{P_Q, t_Q\}$,

Where P_0 is an input to the network and t_0 is the corresponding correct output. As the input are applied to the network, the network output are compared to the targets. The learning rule is then used to adjust the weight and biases of the network in order to move the network output closer to the targets.

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.

In unsupervised learning, the weight and biases are modified in response to network input only. There are no target output available.

Reinforcement learning:

In this learning, a teacher though available does not present the expected answer but only indicates the computed output is correct or incorrect. The information provided helps the network in its learning processes. A reward is given for a correct answer computed and a penalty for a wrong answer. But reinforced learning is not one of the popular forms of learning.

Supervised and unsupervised learning methods which are most popular forms of learning have found expression through various rules.

Some of the widely used rules have been presented below

Hebbian learning:

This rule was proposed by Hebb (1949) and is based on correlative weight adjustment. This is the oldest learning mechanism inspired by biology.

In this the input-output pattern pairs (X_i, Y_i) are associated by the weight matrix W, known as the correlation matrix. It is computed as

$$W = \sum_{i=1}^{n} X_i Y_i^T$$

Here, Y_i^T is the transpose of the associated output vector Y_i.

Gradient descent learning:

This is based on the minimization of error E defined in terms of weight and the 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

Thus, if Δw_{ij} is the weight update of the link connecting the ith and jth neuron of the two neighbouring layers, then Δw_{ij} is defined as

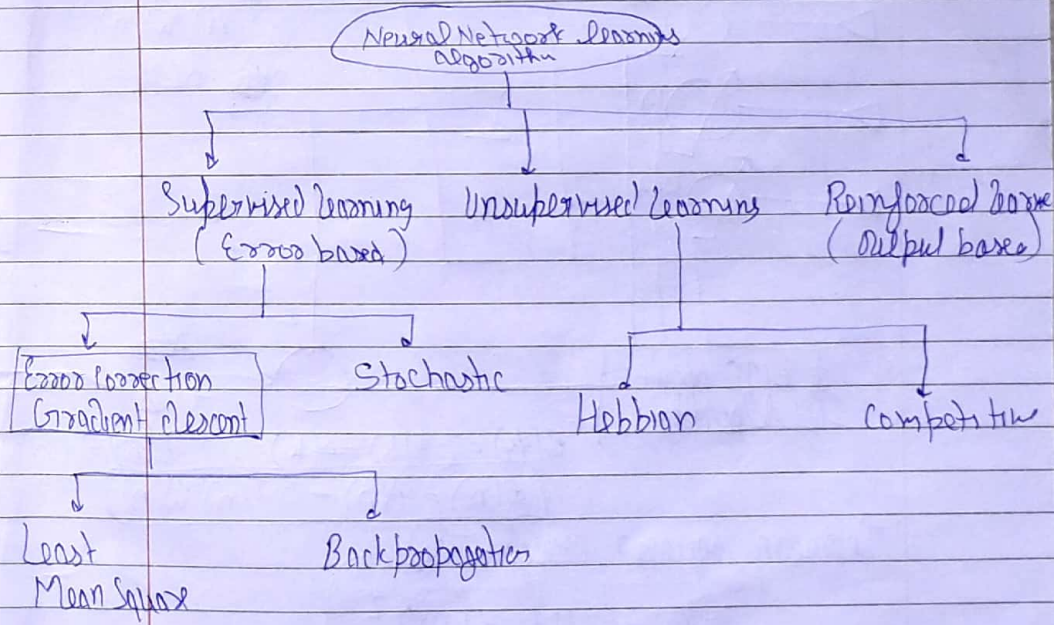
$$\Delta w_{ij} = \eta \frac{\partial E}{\partial w_{ij}}$$

where η is the learning rate parameter and $\frac{\partial E}{\partial w_{ij}}$ is the error gradient with reference to the weight w_{ij}

The Widrow and Hoff's Delta rule and Back propagation learning rule are all examples of this type of learning mechanism

Competitive learning:
In this method those neurons which respond strongly to input stimuli have their weight updated.
When an input pattern is presented, all neuron in the layer compete and the winning neuron undergoes weight adjustment. Hence, it is a winner takes-all strategy

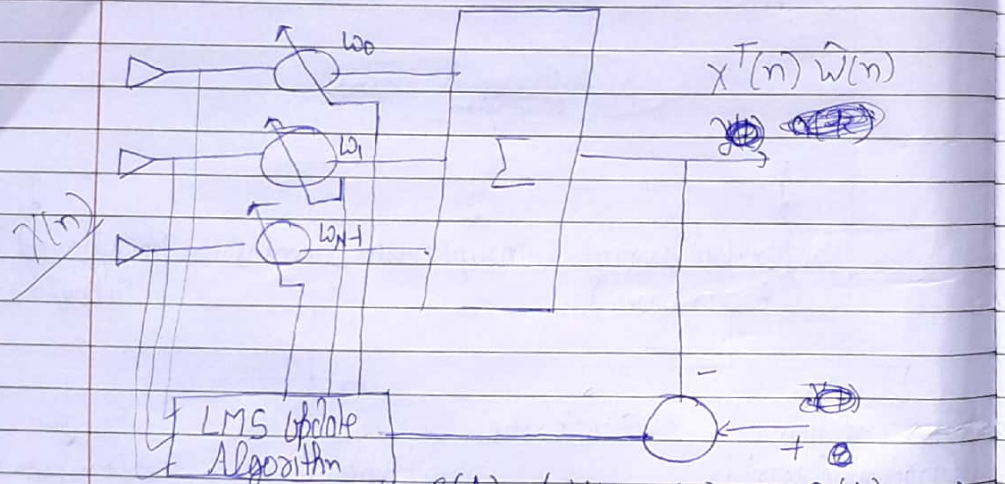
Stochastic learning:
In this method, weights are adjusted in a probabilistic fashion
An example is evident in simulated annealing the learning mechanism employed by Boltzmann and Cauchy machines, which are a kind of NN system



Date
Page No.

Widrow-Hoff Learning Rule
OR
Least Mean square learning (LMS)

- The widrow Hoff learning rule is applicable for the supervised training of neural Network
- It is independent of the activation function of neurons used.
- Since it minimizes the squared error between the desired output



$e(n) = d(n) - x^T(n)\,\hat{w}(n)$

The mean square error $F(x)$

$$F(x) = (t(k) - a(k))^2$$
$$= e^2(k)$$

Where the expectation of the squared error has been replaced by the squared error at iteration k. Then, at each iteration we have a gradient estimate of the form

$$\hat{\nabla} F(x) = \nabla e^2(k)$$

The first R elements of $\nabla e^2(k)$ are derivatives with respect to the network weights, while the $(R+1)$st element is the derivative with respect to the bias. Thus we have

The least mean-square (LMS) algorithm is configured to minimize the instantaneous value of the cost function

$$\xi(\hat{w}) = \frac{1}{2} e^2(n)$$

Where $e(n)$ is the error signal measured at time n. Differentiating $\xi(\hat{w})$ with respect to the weight vector $\hat{w}$ yields

$$\frac{\partial \xi(\hat{w})}{\partial \hat{w}} = e(n) \frac{\partial e(n)}{\partial w}$$

As the
As with the least squares filter, the LMS algorithm operates with a linear neuron, so we may express the error signal as

$$e(n) = d(n) - x^T(n)\,\hat{w}(n)$$

Hence

$$\frac{\partial e(n)}{\partial \hat{w}(n)} = -x(n)$$

ε Epsilon

η - Eta

ξ → x_i

$$\frac{\partial \mathcal{E}_a(\hat{w})}{\partial \hat{w}(n)} = -x(n)e(n)$$

using this latter result as the instantaneous estimate of the gradient vector, we may write

$$\hat{g}(n) = -x(n)e(n)$$

According to, the ~~left~~ steepest descent algorithm ~~is basically described by~~

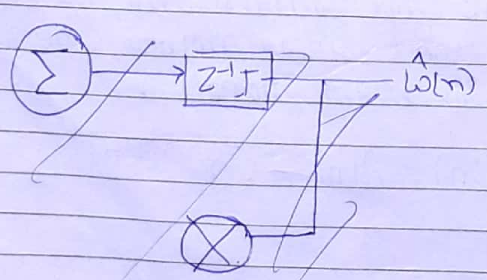
$$w(n+1) = w(n) - \eta \hat{g}(n)$$

Putting the value of $\hat{g}(n)$ in to above equation

$$\hat{w}(n+1) = w(n) - \eta[-x(n)e(n)]$$

$$= \hat{w}(n) + \eta x(n)e(n)$$

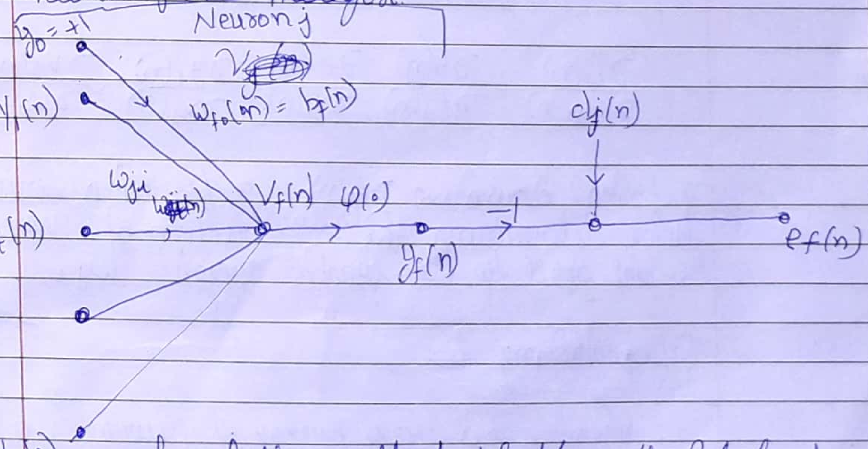
Where η is a positive constant called learning rate, parameter



The back Propagation Algorithm :-

The popularity of on-line learning for the supervised training of multilayer perceptrons has been further enhanced by the development of the back propagation algorithm

To describe this algorithm, consider in fig. The defects neurons j being fed by a set of function signals produced by a layer of neurons to its left. The induced local field $v_j(n)$ produced at the input of the activation function associated with neuron j is therefore



Signal flow graph highlighting the details of output neuron j.

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \qquad \text{(1)}$$

Where m is the total no of inputs (excluding the bias) applied to neuron j. The synaptic weight w_{j0} (corresponding to the fixed input $y_0 = +1$) equals to the bias b_j

applied to the neuron j.
Hence the function signal $y_j(n)$ appearing at the output of the neuron j at iteration n is

$$y_j(n) = \phi_j(v_j(n)) \quad \text{——} (2)$$

In the similar manner to the LMS algorithm, the back propagation applies a correction $\Delta w_{ji}(n)$ to the synaptic weight $w_{ji}(n)$, which is proportional to the partial derivative $\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)}$.

According to the chain rule of calculus, we may express this gradient as

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)} \quad (3)$$

The partial derivative $\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)}$ represent a sensitivity factor, determining the direction of search in weight space for the synaptic weight w_{ji}

~~Differentiating to~~
~~According to the~~
the instantaneous error energy of neuron j is defined by

$$\mathcal{E}_j(n) = \frac{1}{2} e_j^2(n) \quad \text{——} (4)$$

Summing the error energy contribution all the neurons in the output layer, we express the total instantaneous error energy of the whole network as

$$\mathcal{E}(n) = \sum_{j \in C} \mathcal{E}_j(n) \quad \text{——} (5)$$

$$\mathcal{E}(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n) \quad \text{——} (6)$$

Differentiating above eq. with respect to $e_j(n)$, we get

$$\frac{\partial \mathcal{E}(n)}{\partial e_j(n)} = e_j(n) \quad \text{——} (7)$$

The error signal produced at the output of neuron j is defined by

$$e_j(n) = d_j(n) - y_j(n) \quad \text{——} (8)$$

Where $d_j(n)$ is the ith element of the desired response vector $d(n)$.
Differentiating both side with respect to $y_j(n)$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \quad \text{——} (9)$$

differentiating Eq (2) with respect to $v_j(n)$ we get

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \phi_j'(v_j(n)) \quad \text{——} (10)$$

Where the use of prime (on right hand side) signifies differentiation with respect to argument

Differentiating Eq (1) with respect to $w_{ji}(n)$ yield

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \quad \text{——} (11)$$

The use of equation 7, 9, 10 & 11 in Eq. (3) we get

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = -e_j(n)\, \phi_j'(v_j(n))\, y_i(n) \quad \text{——} (12)$$

The correction $\Delta w_{ji}(n)$ applied to $w_{ji}(n)$ is defined by the delta rule

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} \quad \text{---(13)}$$

where η is the learning rate parameter of the back propagation algorithm.
The use of minus sign accounts for gradient descent in weight space (i.e., seeking a direction for weight change that reduces the value of $\mathcal{E}(n)$)

From the equation of 12 & 13, we get

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \quad \text{---(14)}$$

where $\delta_j(n)$ is local gradient $\delta_j(n)$.

$$\delta_j(n) = \frac{\partial \mathcal{E}(n)}{\partial v_j(n)}$$

$$= \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)}$$

$$= e_j(n) \cdot \phi_j'(v_j(n)) \quad \text{---(15)}$$

The local gradient point to required changes in synaptic weights.
The local ~~gradient~~ gradient $\delta_j(n)$ for output neuron j is equal to the product of the ~~error~~ corresponding error signal $e_j(n)$ from that neuron and the derivative $\phi_j'(v_j(n))$ of associated activation function

From Eqs (4.15) and (4)
from the Eq 14 & 15, we note that a key factor involved in the calculation of the weight adjustment $\Delta w_{ji}(n)$ is the error signal $e_j(n)$ at the output of the neuron j.
In this context, we may identify two distinct cases, depending on where in the network neuron j is located.

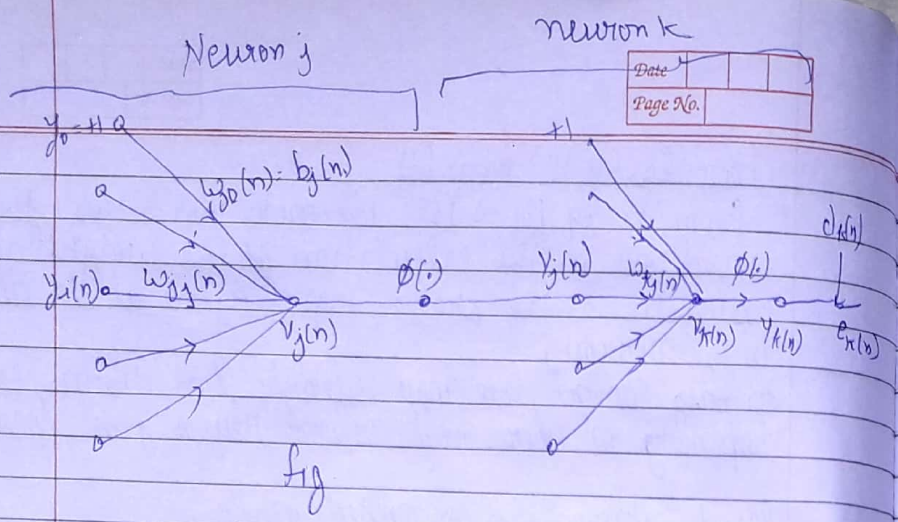
Case 1 Neuron j is an output node

This case is simple to handle because each output node of the network is supplied with a desired response of its own, making it ~~not~~ straightforward matter to calculate the associated error signal
The local gradient $\delta_j(n)$ is calculated with the help of equation 15.

Case 2 Neuron j is a Hidden Node

When neuron j is located in a hidden layer of the network, there is no specified desired response of the neuron. Accordingly, the error signal for a ~~neuron~~ hidden neuron would have to be determined recursively and working backwards in terms of the error signals of all the neurons to which that hidden neuron is directly connected, this is where the development of the back-propagation ~~@~~ algorithm gets complicated.

Consider the situation in fig., which ~~depicts~~ depicts neuron j as hidden node of the network. According to Eq (15), we may redefine the local gradient $\delta_j(n)$ for hidden neuron j as



fig

Consider

$$\delta_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)}$$

$$= -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \cdot \phi_j'(v_j(n)), \quad \text{— (16)}$$

neuron j in hidden

To calculate the partial derivative $\frac{\partial \mathcal{E}(n)}{\partial y_j(n)}$, we may proceed as follows

$$\mathcal{E}(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n) \quad \text{— (17)}$$

where k is an output node

Diff. Eq (17) with respect to the function signal $y_j(n)$, we get

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \quad \text{— (18)}$$

Next, we use the chain rule for the partial derivative $\frac{\partial e_k(n)}{\partial y_j(n)}$ and rewrite eq (18) in equivalent form

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)} \cdot \frac{\partial v_k(n)}{\partial y_j(n)} \quad \text{— (19)}$$

However from the figure, we note that

$$e_k(n) = d_k(n) - y_k(n)$$
$$= d_k(n) - \phi_k(v_k(n)) \quad \text{— (20)}$$

neuron k is an output node

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\phi_k'(v_k(n)) \quad \text{— (21)}$$

We also note from the figure that for neuron k, the induced local field is

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n) \quad \text{— (22)}$$

where m is the total number of input (excluding the bias) applied to neuron k and the corresponding input is fixed at the value +1.

Differentiating Eq (22) with respect to $y_j(n)$ yields

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \quad \text{— (23)}$$

Putting the value of equation 21 & 23 in equation (19) we gets

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = -\sum e_k(n) \phi_k'(v_k(n)) w_{kj}(n)$$
$$= -\sum_k \delta_k(n) w_{kj}(n) \quad \text{— (24)}$$

Finally using Equation (24) in Eq (16), we get the back propagation formula for the local gradient $\delta_j(k)$

$$\delta_j(n) = \phi_j'(v_j(n)) \sum_{k} \delta_k(n) w_{kj}(n) \quad (25)$$

neuron j in hidden

The outside factor $\phi_j'(v_j(n))$ involved in the computation of the local gradient $\delta_j(n)$ in Eq 25 depends solely on the activation function associated with the hidden neuron j. The remaining factor involved in this computation namely the summation over k - depends on two sets of terms