Chapter-1
Introduction

Biological neuron of human brain VS the machine:

Human brain	Machine
1) It is more intelligent	1) It is less intelligent than brain
2) The processing speed is slower	2) The processing speed is faster
3) The capacity of parallel processing is higher	3) The capacity of parallel processing is lower
4) Capacity of fault tolerance is higher	4) Fault tolerance capacity is lower
5) It can learn things from its environment itself	5) It can't learn things from its environment.

Human brain	Machine
6) Provide result between [0-1]	6) Provide result 0 or 1
7) Distributed processing is occurred automatically	7) It is not possible here to occur distributed processing automatically.

Graceful degradation ☆☆

It is a state where the performance of the system slowly falls from high level to a reduced level but without dropping catastrophically to zero.

occurs in case of continuing damage of nurone

Fault tolerance:

→ It is a vital feature ~~of the~~

→ of the operation of the human brain.

→ Everyday a few neurons die

→ as part of the natural course of events.

→ more are lost

→ if brain gets bumped about.

→ But it continuous working

→ as if nothing has happened.

Structure of the Brain:

→ basic unit of the brain: neuron

→ brain contains ten thousand million (10^{10})
↳ basic unit

→ each neuron is connected to about ten thousand (10^4) others.

→ neurons form two main types.

↳ local processing interneuron cells — have their input and output connections over about 100 microns. (100μ)

↳ output cells.
→ connect brain to muscle
→ " from sensory organs into the brain.

→ neuron accepts many inputs.
↳ all added up in some fashion.

→ if enough active inputs (1) are received at once

neuron will be activated or ~~file~~ fire.

→ if not neuron will remain in its inactive.

~~Soma:~~

→ Soma:

Body of the neuron.

Dendrite:

Attached to the soma are long irregularly shaped filaments called dendrite.

→ act as the connections through which all the input to the neuron arrive.

Axon:

→ electrically active.

→ Serves as the output channel of the neurone.

→ it is a non-linear threshold device

→ action potential:

Producing a voltage pulse, called an action potential. It lasts about 1 millisecond (10^{-3}s). when the resting potential within the soma rises above a certain critical threshold.

→ it is in fact a series of rapid voltage spike.

Synapse:

Axon terminates in a specialised contact, called a synapse,
↳ couples the axon with the dendrite of another cell.

→ no direct linkage across the junction

Neurotransmitters:

Synapse releases chemicals called neurotransmitter when its potential is raised sufficient by the action potential.

→ neurotransmitter diffuse across the gap.

→ chemically activate gates on the dendrites.

→ when open, allow charged ions to flow.

→ this flow of ions

↳ alters the dendritic potential.

→ provides a voltage pulse on the dendrite

Synaptic junction:

At the synaptic junction the number of gates opened on the dendrite depends on number of neurotransmitters released.

Synaptic cleft:

Space between the cell membrane of an axon terminal and that of the neighboring cell

→ with which it forms a synapse across which a nerve impulse is transmitted.

Also known as synaptic gap

Learning in Biological Systems:

→ Learning occurs when modifications are made to the effective coupling between one cell to another

→ at the synaptic junction.

→ Neurotransmitters released from the synaptic vesicles

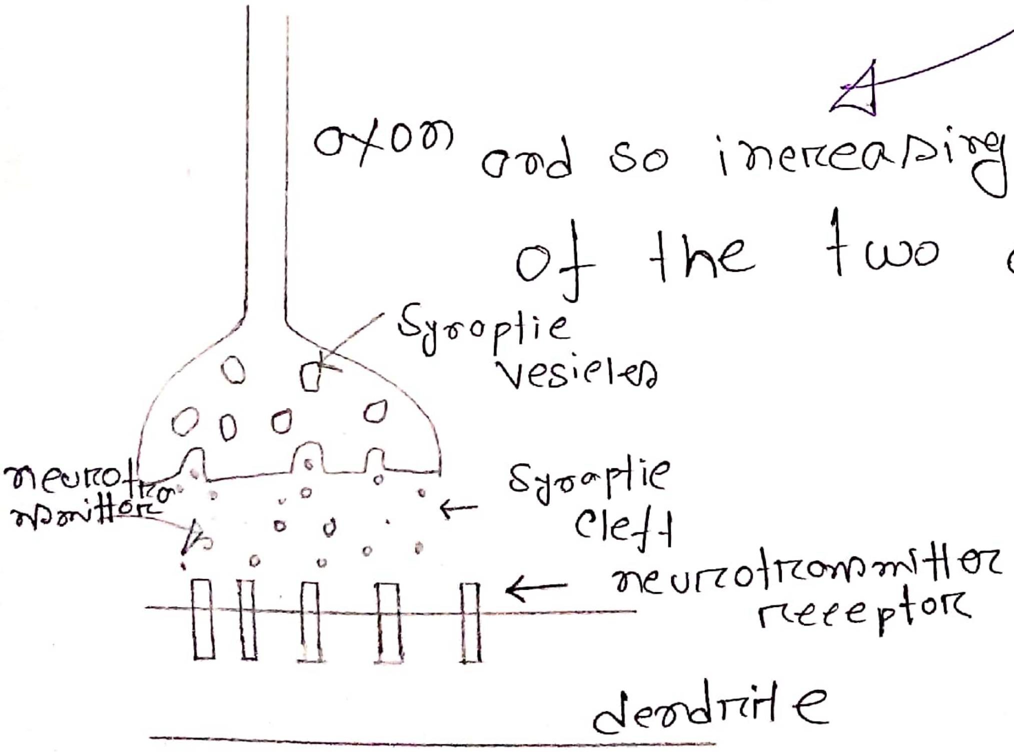
↳ diffuse across the synaptic cleft

trigger the receivers on the dendrite.

→ Neurotransmitter has the effect of opening more gates

↳ on the dendrite on the post synaptic side of the junction.

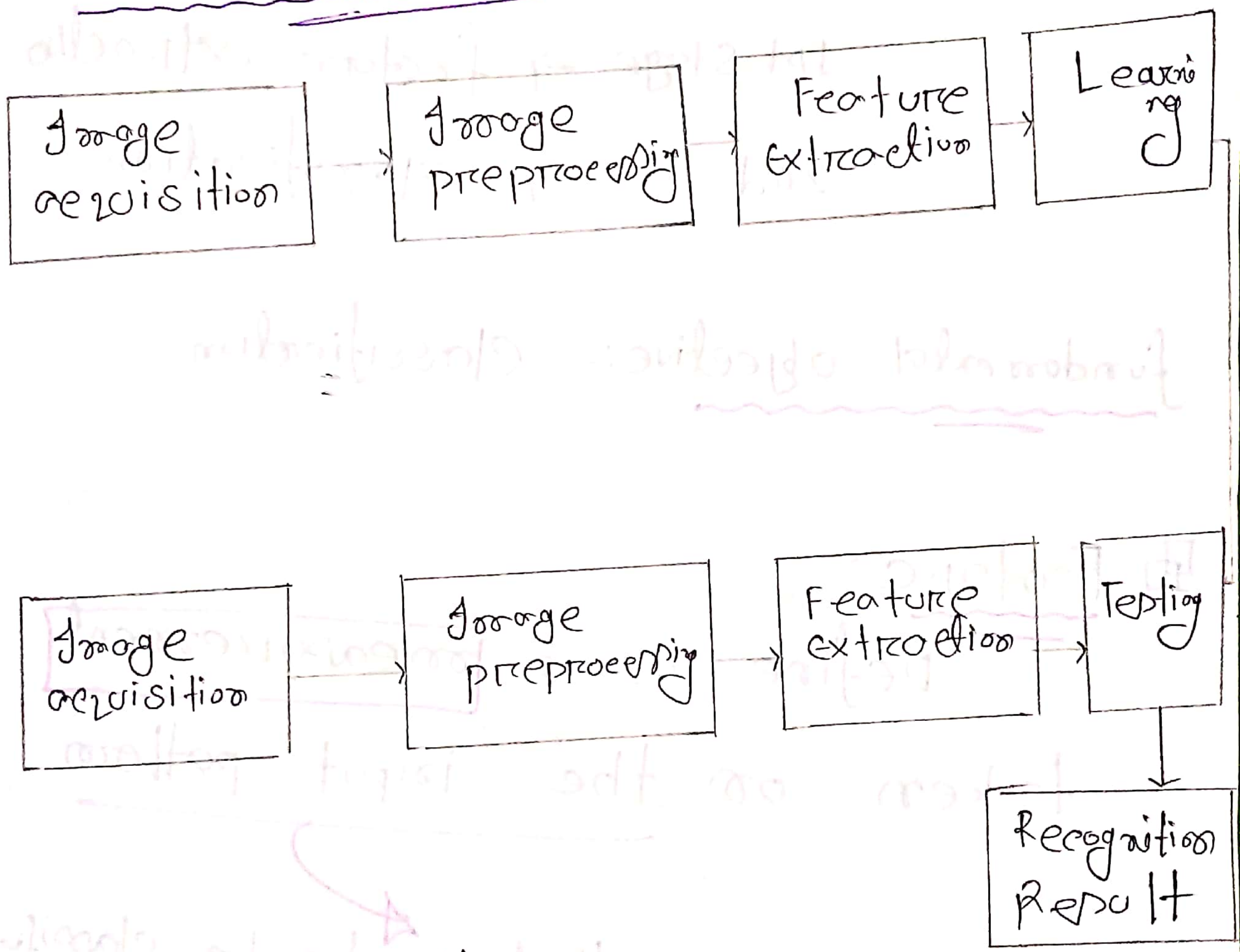
and so increasing coupling effect of the two cells.



Chapter-2

Pattern Recognition:

Pattern recognition (block diagram)



Pattern recognition:

It can be defined as act of taking new data and taking an action based on category of data.

A pattern recognition system can be considered as

→ 2 stage devices.

1st stage → feature extraction

2nd " → classification

fundamental objective: classification

Feature:

Define as a measurement token on the input pattern

that is to be classified

→ provide a definite characteristic of that input type.

Example:

If we wish to distinguish the letter 'F' from the letter 'E'

we need to compare the number of vertical and horizontal strokes

in the character.

Feature vector:

If we make n measurements on our input pattern,

each of which is unique feature

we can use algebraic notation to create a set of these features call it a feature vector.

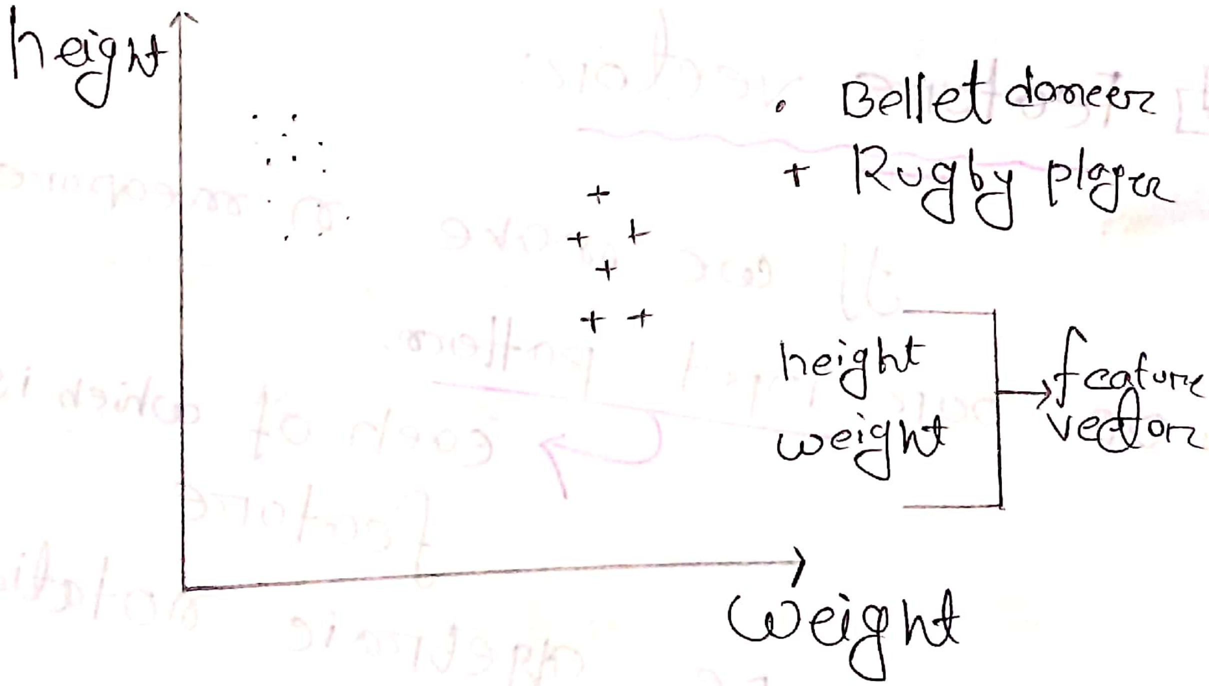
Feature Space:

The dimensionality of the feature vector,

↳ number of element in it

creates an n dimensional feature space.

Example:



Discriminant function:

The mathematical definition of a decision boundary is a "discriminating function"

It is a function that maps our input features onto a classification space.

It is tried to make this function as simple as possible.

Classification techniques:

Fall into two broad categories

→ Numeric technique

→ Non Numeric "

Numeric:

→ include deterministic and Statistical measures

→ Considered as measures made on the geometric pattern Space.

Non-numeric:

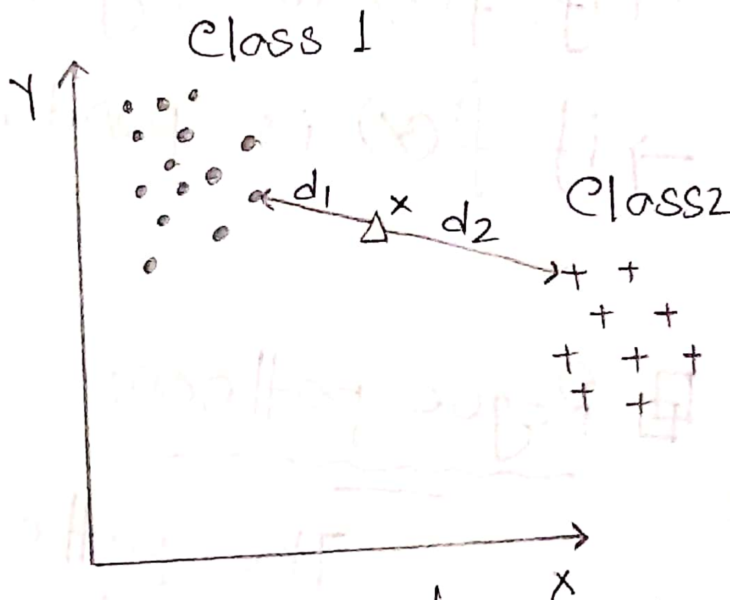
Those which take us into the domain of symbolic processing that is dealt with by such methods as fuzzy sets.

Nearest Neighbour Classification:

Two classes represented

- Class I
- Class 2

in pattern space.



we wish to decide to which of the two the unclassified pattern, X, belongs.

Δx unclassified pattern

d_1 shortest distance to class1

d_2 Shortest distance to class2

→ make a decision based on the shortest distance

↳ to the neighbouring class samples.

→ Formally that defines a discriminant function $f(x)$ by

$f(x)$ = closest (class1) - closest (class2)

~~→ if~~

→ if $f(x)$ is negative then class1 membership

→ if $f(x)$ is positive then class2 "

Rogue pattern:

The pattern that has class membership of one class but does in fact lie closer to another class.

If our unclassified input is measured against the rogue sample.

↳ invariably result in misclassification.

Soln

→ take several distance measures against many class samples.

↳ effect of any rogue measurement is likely to be averaged out

Distance metries

~~we need~~

Severcal methods are used.

- Hamming distance measure
- Euclidean distance "
- City block distance (manhattan)
- Square distance

Haming distance measure:

For two vectors

$$X = (x_1, x_2, \ldots)$$

$$Y = (y_1, y_2, \ldots)$$

haming distance, $\boxed{H = \Sigma (|x_i - y_i|)}$

- used to compare binary vectors
- provides number of ~~two~~ different bits between two vectors.

simple to calculate

Euclidean distance measure:

In a rectangular coordinate system for two vectors (X and Y)

$$d(X,Y)_{euc} = \sqrt{\sum_{i=1}^{n} (X_i - Y_i)^2}$$

dimensionality of the vector.

For two dimensional,

$$d(X,Y)_{euc} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

NB:

City block distance

The method is defined by

$$\boxed{D_{cb} = \sum |x_{ij} - y_{ij}|}$$

→ faster to compute than the Euclidean

Square distance

Defined by the maximum of the distance between two vectors

$$\boxed{D_{sq} = MAX\,|x_i - Y_i|}$$

Linear classifier:

Consider Simple two dimensional, two-class discrimination problem.

→ Wish to classify an input into one of two possible classes A or B.

→ Add a weight vector to pattern space

it's orientation used to define a linear decision boundary.

→ decision boundary defines a discriminating function $f(x)$

$$f(x) = \sum_{i=1}^{n} w_i x_i$$

where, x_i = i-th component of an input vector
w_i = i-th " " weight "
N = dimensionality of input vector

Class definition:

if $f(x) > 0$ = Class A
if $f(x) < 0$ = class B.

→ problem lies in actually finding a suitable weight vector

→ If we expand the discriminant function using matrix algebra
↳ we can visualise the dependence of the output on the value of the weight vector.

$f(x) = \sum w_i x_i - \theta$

This expands to,
$f(x) = (|w| \cdot |x| \cos\phi) - \theta$

where,
ϕ is = angle between the vector x and w.

2 parameters control the position of the decision boundary
in the pattern space.

→ Slope of the line
→ y-axis intercept.

determined by the value of the weight vector
controlled by bias value θ

we have, $\sum w_i x_i - \theta = 0$

$$x_1 \times w_1 + x_2 \times w_2 - \theta = 0$$

Rearranging this gives us:

$$x_2 = -w_1/w_2 \times x_1 + \theta/w_2$$

→ comparing this with $y = mx + c$

$$m = -w_1/w_2$$

$$c = \theta/w_2$$

~~tion~~

☆ linear classifiers can also be used

to separate more than two classes.

- → by arranging many decision boundaries
- → performing several tests to satisfy the condition for each class.

4-class problem using linear classifier:

→ $\boxed{\text{decision boundaries}}$ can be selected to test between A or BCD

→ if the result is not A, then test for B or CD

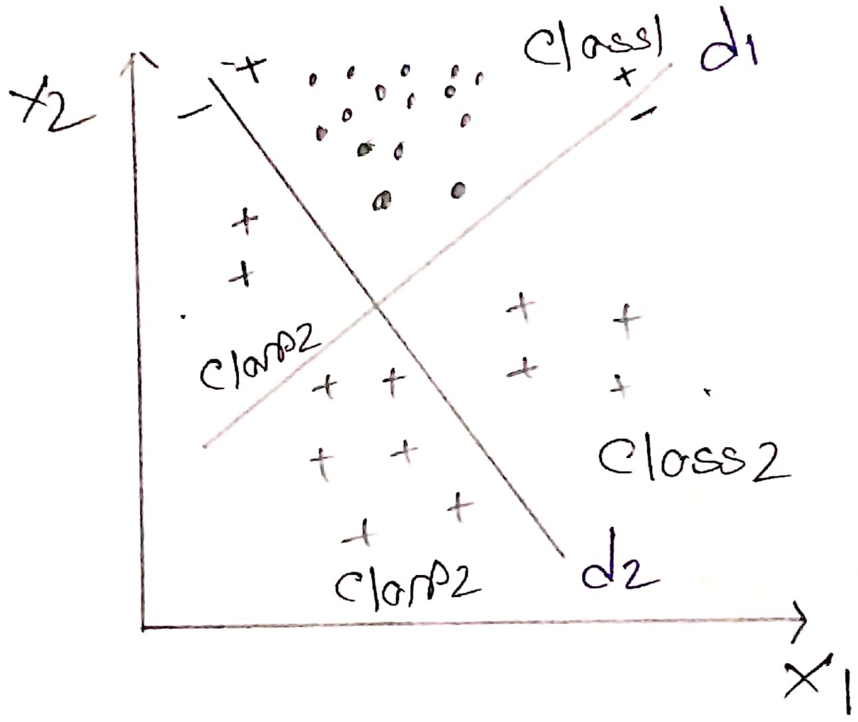
→ if not B then test for C or D.

→ for different class boundary conditions the decision surface can be split up in a piecewise fashion.

Classification	Sign of decision line d_1	d_2
Class 1	+	+
Class2	+	+
	+	−
	−	−

→ if not B then test for C or D.

→ for different class boundary conditions the decision surface
↳ can be split up in a piecewise fashion.



classification	Sign of decision line d_1	d_2
Class 1	+	+
Class2	-	+
	+	-
	-	-

Chapter-3

The Basic Neuron

Basic feature of a biological neuron:

Basic function:

→ to add up its inputs

→ to produce an output

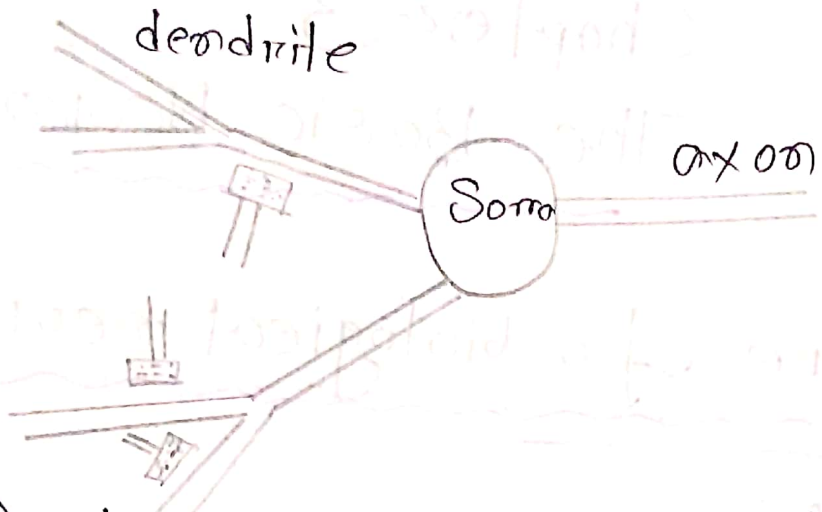
if this sum is greater than some value ~~kn~~

→ Known as threshold.

→ inputs to the neuron arrive along the dendrites.

→ connected to the output from other neurones by specialised junctions

→ called synapses



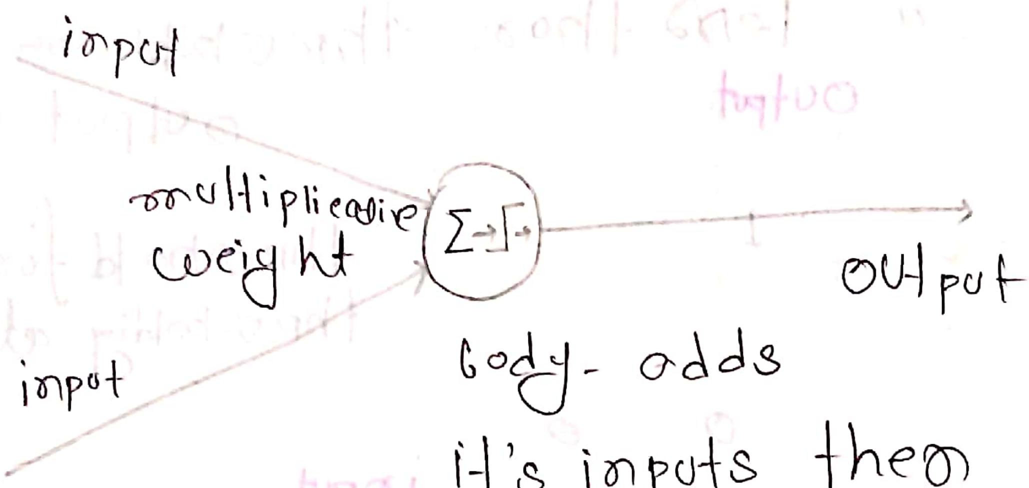
The features are summarise as follows

- Output from a neuron is either on or off
- output depends only on the inputs.
- A certain number must be on at any one time in order to make the neuron fire.

Artificial Neuron

One node of the neuron with capturing the above with important features is called artificial neuron.

Basic model of the neuron :



body - adds it's inputs then thresholds.

→ performs a weighted sum of its inputs.

→ Compares this to some
internal threshold level

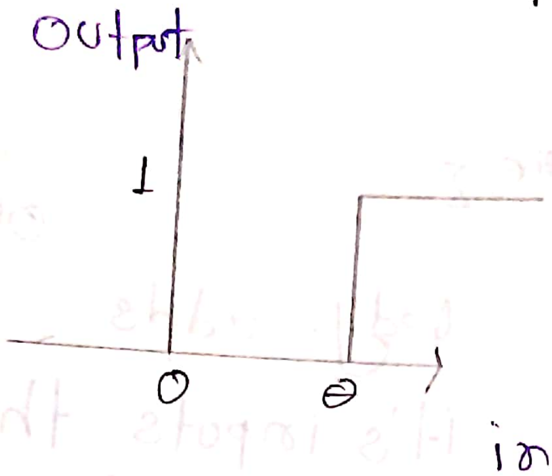
→ turns on only if this level is exceeded.

$$\text{total input} = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \sum_{i=1}^{n} w_i x_i$$

→ Sum has to be compared to a certain value in the neuron.
threshold value

→ Sum is greater than threshold
output a 1

→ " " less than threshold ~~ot~~
output a 0

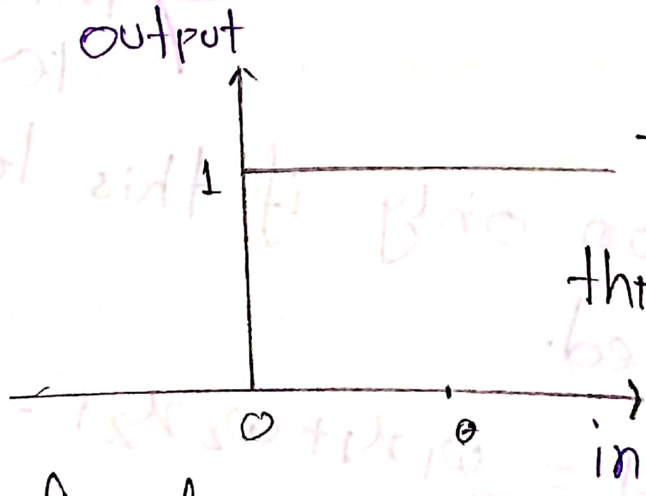


threshold function
thresholding at θ

→ threshold value can be subtracted from the weighted sum.

→ if result is positive, then output a 1

→ if " " negative " " a 0



Threshold function.
thresholding at zero

threshold function is also called
→ Step function
→ Heaviside "

Biasing the neuron:

- take the threshold out of the body of the model neuron.
- connect it to an extra input.
- extra input is fixed to be "on" all time.
- extra input is multiplied by a weight $= -\theta$
  $=$ minus of the threshold value.
- $-\theta$ is known as <u>neuron's bias or offset</u>

Calling the output y we can write

$$y = f_h \left[\sum_{i=1}^{n} w_i x_i - \theta \right]$$

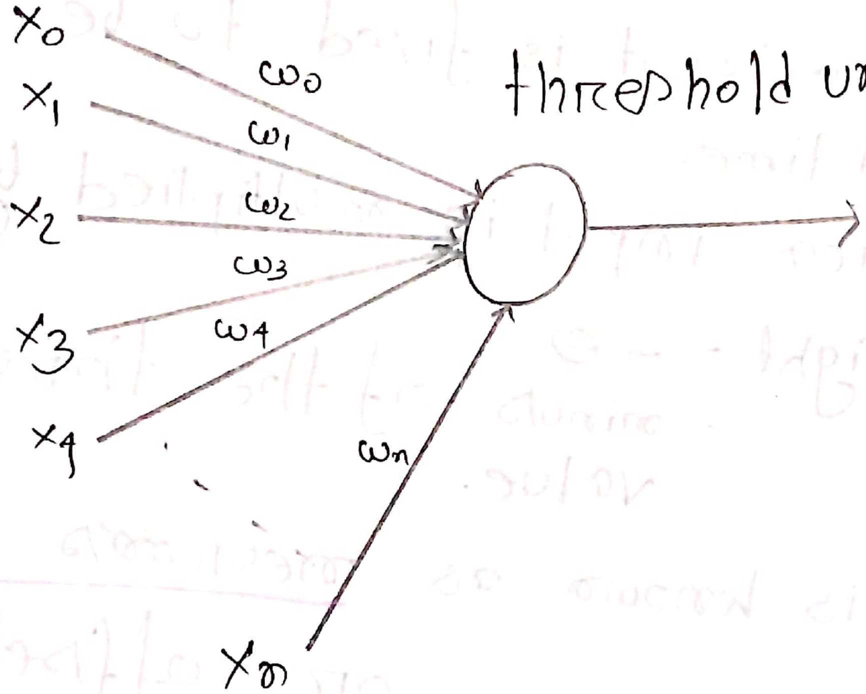
↳ step function
→ actually Heaviside function

$f_h(x) = 1 \quad x > 0$
$f_h(x) = 0 \quad x \leq 0$

in case of biasing the neuron,

$$y = f_n\left[\sum_{i=0}^{n} w_i x_i\right]$$

Model for biasing the neuron:



→ proposed by Mc Culloch and Pitts
↳ in 1943

Chapter-4
The Multilayer Perceptron

Credit assignment problem:

This is a type of problem happens when the network is unable to determine which of the input weights should be increased and which should not and so is unable to work out what changes should be made ~~to pr~~ to produce a better solution next time.

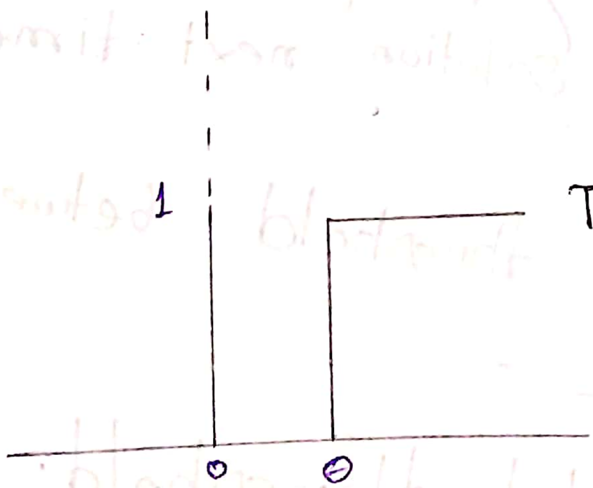
Soln:
- Linear threshold between limits.
- Sigmoidal threshold.

Hard limiting thresholding function:

It makes the inputs from the outputs and adjusting the model so that difficulty by the tracing the root of the problem can be solved.

→ related with single layer perceptron.

→ Removes the information that is needed if the network is to successfully learn.



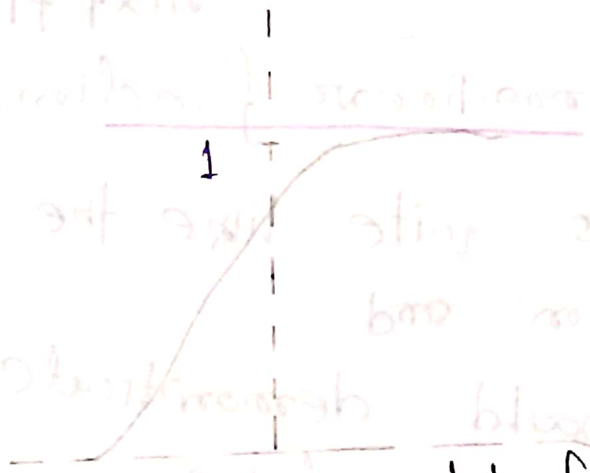
Threshold function.

Sigmoidal function:

→ mathematical function with a sigmoid curve

→ non-linear thresholding function.

layers of perceptron units using linear functions are no more powerful than a suitably chosen single layer.



Sigmoidal function

It is defined as, $f(net) = 1/(1+e^{-k\,net})$

where, $0 < f(net) < 1$

Advantages of sigmoidal function.

→ enough informations about the output is available

↳ to units in earlier layer.

so that it can adjust weights with these enough information.

→ decrease the error next time.

→ it is a non-linear function.

So it is quite like the step function and so should demonstrate behavior of similar nature.

□ Multilayer perceptron:

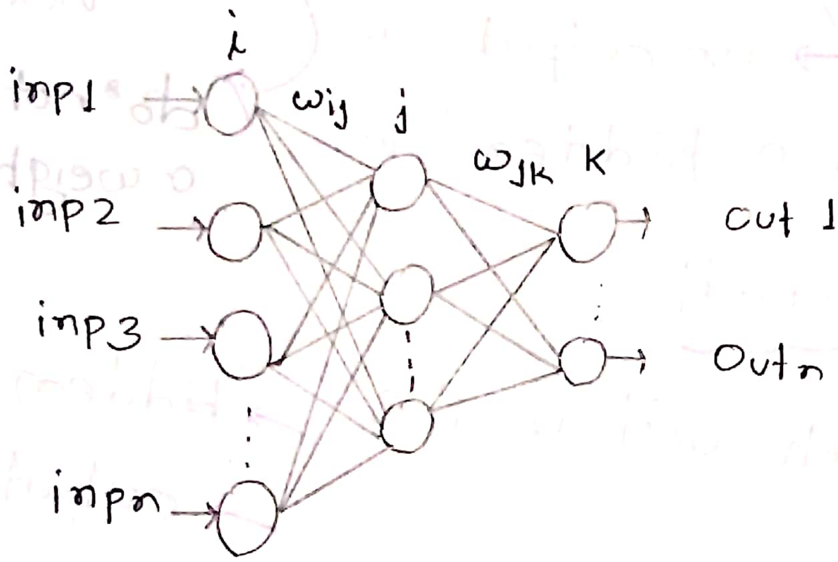
→ 3 layers
- an input layer → receive input → serve to distribute the value to the next layer → do not perform a weighted sum or thresh.
- an output "
- a hidden "

perceptron unit:

1 Each unit in the → hidden layer → output layer

except that the thresholding function.

Mathematics of multilayer perceptron, or building a multilayer perceptron:



Step-1: initialise w_{ij}, w_{jk} and threshold values for each PE

Step-2: network provides the input patterns & desired respective output patterns

Step-3: input patterns are connected to the hidden PEs through the weights w_{ij}

Forward propogation

For i to j

$net_{oj} = \sum \omega_{ij} O_{oi}$

$activ_j = net_j + \omega u_{hj}$

$O_{oj} = 1/(a + e^{-K_1 * activ_j})$

For j to k

$\star K_1, K_2$ is spread factor

$net_{ok} = \sum \omega_{jk} O_{oj}$

$activo_k = net_{ok} + uo_k$

$O_{ok} = 1/(a + e^{-K_2 * ativo_k})$

error $\delta_{ok} = t_{ok} - O_{ok}$

Backword propogation

For k to j

$\Delta \omega_{jk} = \eta_2 K_2 \delta_{ok} O_{oj} O_{ok} (1 - O_{ok})$

$\omega_{jk} = \omega_{jk} + \Delta \omega_{jk}$

$\Delta uo_k = \eta_2 K_2 \delta_{ok} O_{ok} (1 - O_{ok})$

Forward propogation

For i to j

$$net_{aj} = \sum \omega_{ij} O_{ai}$$

$$activ_j = net_j + \omega u_{hj}$$

$$O_{aj} = 1/(a + e^{-K_1 * activ_j})$$

For j to k

*K_1, K_2 is spread factor

$$net_{ak} = \sum \omega_{jk} O_{aj}$$

$$activo_k = net_{ak} + uo_k$$

$$O_{ak} = 1/(a + e^{-K_2 * ativo_k})$$

error $\delta_{ak} = t_{ak} - O_{ak}$

Backward propogation

For k to j

$$\Delta \omega_{jk} = \eta_2 K_2 \delta_{ak} O_{ak} O_{aj}(1 - O_{ak})$$

$$\omega_{jk} = \omega_{jk} + \Delta \omega_{jk}$$

$$\Delta uo_k = \eta_2 K_2 \delta_{ak} O_{ak} (1 - O_{ak})$$

$$U_{ok} = U_{ok} + \Delta U_{ok}$$

For j to i

$$\Delta \omega_{ij} = \eta_1 K_1 O_{ai} O_{aj} (1 - O_{aj}) \sum \delta_{ak} \omega_{jk}$$

$$\omega_{ij} = \omega_{ij} + \Delta \omega_{ij}$$

$$\Delta U h_j = \eta_1 K_1 O_{aj} (1 - O_{aj}) \sum \delta_{ak} \omega_{jk}$$

$$U h_j = U h_j + \Delta U h_j$$

Calculating Error:

Error can be calculated ~~as~~, by using

$$Error_{a} = \frac{1}{2} \sum (t_{ok} - O_{ak})^2$$

makes the math a bit simpler

~~Calcu~~

Calculation of output unit O_{pj}:

$$O_{pj} = f(net) = 1/(1 + e^{-k\,net})$$

$$f'(net) = \frac{1}{(1+e^{-k\,net})^2} e^{-k\,net} K$$

$$= K f(net) (1 - f(net))$$

Multilayer perceptron learning algorithm:

**

4 steps

1. Initialise weights and threshold

2. Present input and desired output

$X_p = x_0, x_1, x_2, \ldots x_{n-1}$

$T_p = t_0, t_1, \ldots t_{m-1}$

number of input nodes → n

" of output " → m

$w_0 = -\theta$

x_0 is always 1

3. Calculate actual output

$$y_{pj} = f\left[\sum_{i=0}^{n-1} w_i x_i\right]$$

→ passes it as input to the next layer.

→ final layer output values = O_{pj}

4. Adopt weights

Start from the output layer, and work backward.

$$\boxed{\omega_{ij}(t+1) = \omega_{ij}(t) + \eta \delta_{pj} O_{pj}}$$

η = gain term

δ_{pj} = error term

For output units

$$\delta_{pj} = k O_{pj}(1 - O_{pj})(t_{pj} - O_{pj})$$

For hidden units

$$\delta_{pj} = k O_{pj}(1 - O_{pj}) \sum_{k} \delta_{pk} \omega_{jk}$$

Where, sum is over k nodes in ~~sum is~~ the layer above node j.

~~Feature deetor~~

~~Feature deetor~~

Feature detector:

Feature deetor are individual neurons or group of neurons in the brain that respond to specific attributes of stimulus, movement, orientation etc.

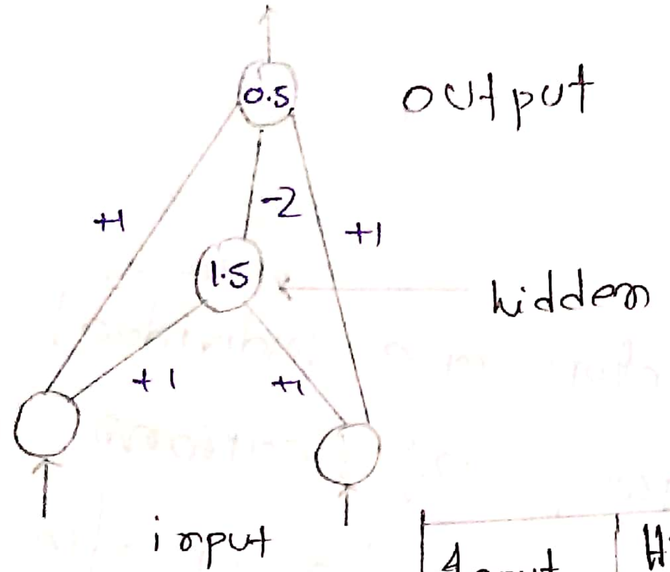
☆ In MLP hidden unit acts as a feature detector.

How to detect feature?

→ Let consider a XOR problem

→ in this case hidden unit ~~as~~ act as a feature detector. → when both inputs are on.

→ vewed as recoding the basic inputs network can learn the required mapping of input patterns to the output ones.



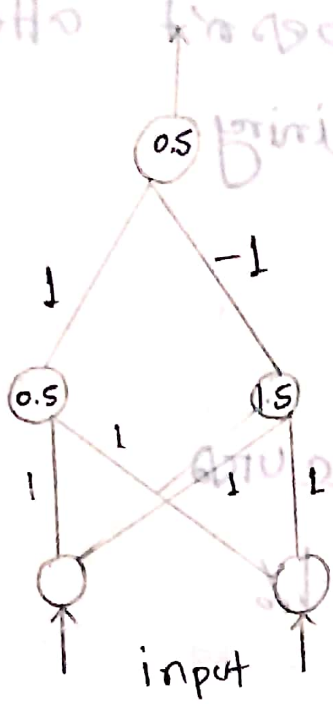
Input	Hidden	Output
00	0	0
01	0	1
10	0	1
11	1	0

☆ → it is possible to produce different network topologies

→ to solve the same problem.

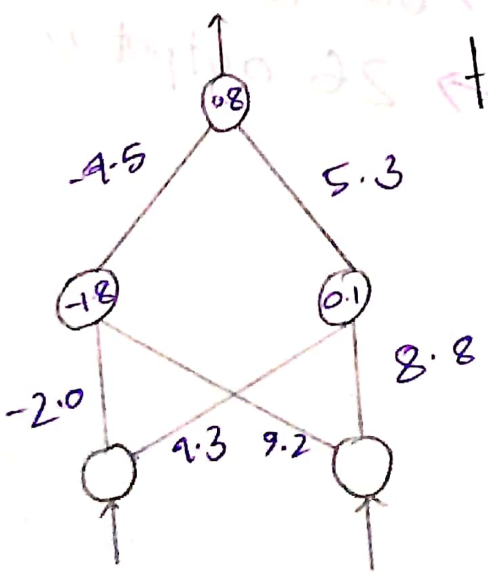
→ direct connection from input to output

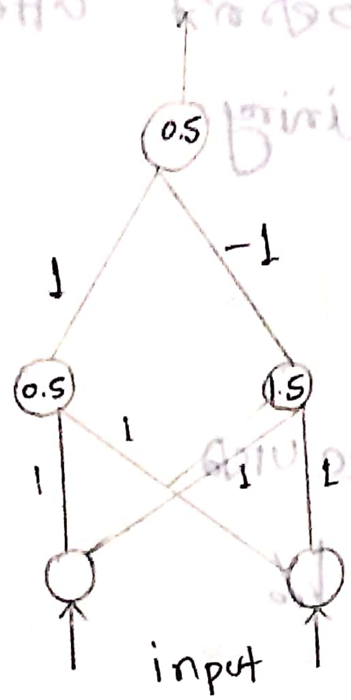
→ no direct " " " to output



→ learning rule is not guaranteed to produce convergence

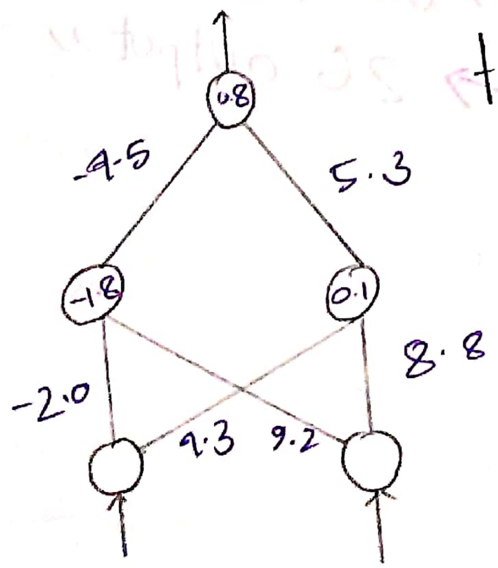
→ it is possible for the network to fall into a situation → it is unable to learn the correct output.





→ learning rule is not guaranteed to produce convergence

→ it is possible for the network to fall into a situation → it is unable to learn the correct output.



→ Stable situation & doesn't alter with further training

local minimum in XOR:

local minimum occurs infrequently - about ·1%

Application of MLP:

i) NETtalk:

→ learns to pronounce English text

→ it consists of

→ 203 input unit

→ 80 hidden "

→ 26 output "

ii) Airline Marketing Tactician (AMT):

2 Stage procedure.

→ MLP that produces forecasts of seat demand.

→ allocate airline resources

to meet these projected demands using

→ standard optimisation technique

2 NN network

→ predicts demand for the seats

→ predicts no-show rate for each class

iii) ~~EG~~ ECG Noise Filtering:

→ Shows the heartbeat of patient.

→ the net has

- → 50 input units
- → 12 hidden "
- → 1 output "

input: 50 time samples of the noisy signal.

output: magnitude of the output

→ noise free value at center of the time frame.

Financial Application:

→ predict the Stock market.

→ trader's "assistant" uses a network

to extract the significant features from past examples

Convex region / Convex hulls:

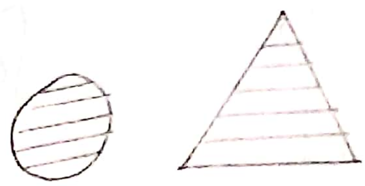
★ It is a region in which any point can be connected to any other by a straight line does n't cross the boundary of the region

Types of convex hulls

2 types
→ open
→ closed

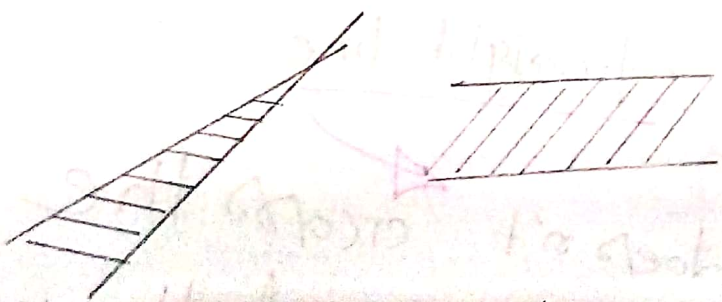
Closed:
has a boundary all around it

Ex: Circle, triangle



Open:
doesn't have a boundary all around it.

Ex:
two parallel line



NB: total number of sides in region = number of units in the first layer.

Arbitrary re shape:

The combination of convex regions may intersect, overlap or be separate from each other, producing arbitary shapes

Kolmogorov theorem:

The arbitary complexity of shapes that we can create, means that we never need more than three layers in a network.

NB

→ 1st layer of MLP produce → line

→ 2nd " of MLP " → convex regions

→ 3rd " of MLP " → arbitary region

~~Generaliq~~

Generalization property:

→ Neural network is good at interpolation

→ not so good at extrapolation

→ it can solve any problem is real world.

→ learning time is large

→ An unseen pattern is classified with others that share the same distinguishing features

→ A unseen pattern is an intermediate mixture of two previously taught patterns

Chance of misclassification

local minima problem

In every direction in which the network could move in energy landscape, the energy is higher that than at the current position

- → network settles into a stable situation
- → does not provide the correct output.

Solution of local minima problem:

→ Lowering the gaining term

→ Addition of internal nodes

→ occur when two or more disjoint classes are categorised as the same.

→ adding more units to allow a better recoding of the input

→ Momentum term

Adding momentum term is fairly successful to reduce the occurrence of the minima. $\delta_p \omega_{ij}(t+1) = \omega_{ij}(t) + \eta \delta_{pj} O_{pi} + \alpha(\omega_{ij}(t) - \omega_{ij}(t-1))$

→ Addition of noise

momentum term $\boxed{0 < \alpha < 1}$

If random noise is added,

→ this perturbs the GD algorithm from the line of steepest descent

prevents

→ enough to knock the system out of local minimum.

Chapter-5
Kohonen- Self-Organising Networks

~~About~~

Self Organisation map(SOM)

→ brain uses spatial mapping to model complex data structure internally

→ allow to perform data compression

on the vectors to be Stored in the network.

→ allows the network to store data in such a way

→ that spatial or topological relationship

→ in the training data are maintained

Structure of Kohonen model:

- neurons aren't arranged in layer as in multi layer perceptron but rather on a flat grid

- all inputs connect to every node in the network.

- no separate output layer

→ organises the nodes in the grid into local neighbourhoods.

→ topological map:

autonomously organised by a cyclic process of comparing input pattern to "vectors"

Kohonen Algorithm

1) Initialise network:

→ Define $w_{ij}(t)$ $[0 \le i \le n-1]$

→ weight from input i to j at time t

→ set the initial radius of the neighbourhood

→ around node j, $N_j(0)$

2) Present input:

input $x_0(t), x_1(t), x_2(t), \ldots x_{n-1}(t)$

where, $x_i(t)$ is the input to node i at time t.

3. Calculate distances

distance between input to each output node j,

$$d_j = \sum_{i=0}^{n-1} [x_i(t) - w_{ij}(t)]^2$$

4. Select minimum distance:

Designate the output node with minimum d_j to be j^*

i.e $j^* = \text{minimum}(d_j)$

5. Update weights

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)(x_i(t) - w_{ij}(t))$$

For j in $N_{j^*}(t)$, $0 \leq i \leq n-1$

Here η is gain term $(0 < \eta(t) < 1)$

Also update the radius.

$$N_j^{*}(t) = N_j(t) - \eta\, N_j(t)$$

6. Repeat by going to 3

termination condition:

Vector quantisation:

A technique to perform data compression on the vectors stored in the ~~mia~~ neighbourhoods

Operation steps:

→ All inputs nods are connected to every node in the network

→ All the node in the grid is itself an output node

- Feedback is restricted to lateral connection → to immediate neighbouring nood.

* multi-dimensional data can be represented in a much lower dimensional space

lossy data compression technique

Vector quantizer:

It maps k-dimensional vectors in the vector space R^k into a finite set of vectors

$Y = \{y_i : i = 0, 1, ----N\}$

Codeword:

Each vector y_i is called a code vector or codeword

Codebook:

Set of all the codewords is called codebook.

Voronoi region:

Associated with each codeword y_i, is a nearest neighbor region called voronoi region.

$$V_i = \{x \in R^k : \|x - y_i\| \leq \|x - y_j\|, \text{ for all } j \neq i\}$$

→ the representative codeword is determined to be the closest in Euclidean distance from the input vector

$$d(x, y_i) = \sqrt{\sum_{j=1}^{k} (x_j - y_{ij})^2}$$

jth component of the input vector

jth component of the codeword y_i

How does VQ work in compression:

- composed of two operations
  - encoder
  - decoder

encoder:

- takes an input vector
- outputs the index of the codeword that offers the lowest distortion.
  - found by evaluating the Euclidean distance
- index of the codeword is sent through a channel.

decoder:

→ receives the index of the codeword

→ replaces the index with the associated codeword.

Neighbourhoods:

topological neighbourhoods:

This is dynamically changing boundary that defines how many nodes surrounding the wining node will be affected with weight modification during training process.

Initially each node in the network will be assigned a large neighbourhood

⊞ Necessity of the reduction of neighbourhood size

→ The effect of shrinking the neighbourhood is to localise areas of similar activity

→

Process of reducing neighbourhood size:

→ When a node is selected as the closest match to an input it will have its weights adopted to tune it to the input signal

Step-1:

→ ~~force~~

→ Initially with random weight vectors and large neighbourhoods around each node.

→ for each training input the best match node is found.

→ the weight change is calculated

→ all the nodes in the neighbourhood are adjusted

8

Step-2

The neighbourhood size a have also shrunk so that weight modification now have smaller field of influences

Step 3:

→ The neighbourhoods have shrunk to a pre defined limit of nodes.

→ Nodes within the region have all been adopted to to represent an average spread of values about the training data for that class.

Q How quickly do we reduce neighbourhood size and to what final size?

Soln.

→ Adoption rate be a linearly decreasing function with the number of passes through the training set.

→ This ensures that cluster form accurate internal representation of the training data as well as causing the network to converge to a solution within a predefined time limit.

E

* Training is effected by

→ adoptation rate

→ rate at which the neighbourhod is reduced

→ Shape of the neighbourhood boundary

→ square

→ circular

→ hexagonaj

may provide optional results in some case

E

* Training is effected by

→ adoptation rate

→ rate at which the neighbourhood is reduced

→ shape of the neighbourhood boundary

→ square

→ circular

→ hexagonaj

may provide optional results in some case

⊞ Self-organizing map (SOM)

A self-organizing map (SOM) or self-organizing feature map (SOFM) is a type of artificial neural network (ANN) that is trained using unsupervised learning → to produce a low-dimensional, discretized representation of the input space of the training sample.

It is a ~~ma~~ method to do dimensionality reduction.

→ apply competitive learning

Supervised learning vs unsupervised learning

Supervised	unsupervised
1. The learning is guided by the knowledge. ~~what~~	1. There are no previous knowledge for learning
2. It is computationally very complex	2. Less computational complexity
3 Number of classes or labeles are known.	3. Number of classes are not known
4. Example: → classification → regression	4. Example → clustering → Anomaly detection → dimensionality reduction
4. Used to classify feature observation	4. Used to understand (explore) data

Application of supervised learning:

→ Text categorization
→ Face Detection
→ Signature recognition
→ Customer discovery
→ Spam detection
→ Weather forecasting
→ Predicting housing prices based on the prevailing market price

Application of unsupervised learning:

→ Identification of human errors during data entry
→ Conducting accurate basket analysis
→ Malware detection

Advantages & disadvantages of supervised and unsupervised learning:

Supervised learning

Advantages:

→ classes represents the feature of the ground

→ Training data is reusable unless feature changes

Disadvantages:

→ Cost and time involve in selecting training data

→ depend heavily on training set.

Unsupervised learning

Advantages:

- No previous knowledge required
- The opportunity for human error is minimised
- produces unique classes
- Less computationally complex
- Fairly quick and easy to run

Disadvantages:

- spectral classes do not always correspond to informational classes
- Spectral properties of classes can also change time

Chapter-6
Hopfield Network

History

- invented by John Hopfield in the early 1980's
- it has some similarities with perceptrons
- has some difference " perceptron
- it uses an energy function
- related the network to other physical systems

Hopfield network:

- consists of a number of nodes
- each connected to every other node (fully-connected network)
- ★ symmetrically-weighted network
  - as, $\boxed{w_{ij} = w_{ji}}$
- each node has
  - a threshold
  - a step function
- weighted sum of their inputs minus threshold passing through step function
- 2-states inputs
  - binary (0-1)
  - bipolar (-1, +1)

Hopfield Network Algorithm:

☆☆☆

1. Assign connection weights

$$w_{ij} = \begin{cases} \sum_{s=0}^{m-1} x_i^s x_j^s & i \neq j \\ 0 & i = j, \quad 0 \leq i, j \leq m-1 \end{cases}$$

m = number of pattern

x_i^s = element i of the exemplar pattern for class s

2. Initialise with unknown pattern

$$\mu_i(0) = x_i \qquad 0 \leq i \leq N-1$$

$\mu_i(t)$ is output of node i at time t.

3. iterate until convergence

$$\mu_i(t+1) = f_h \left[\sum_{j=0}^{N-1} w_{ij}\, \mu_j(t) \right]$$

$$0 \leq j \leq N-1$$

where,

f_h is the hard limiting non-linearity

Repeat the iteration until the outputs from the nodes remain unchanged

· ~~teoe~~

□ teaching stage:

The weights between the neurons are set → using the equation of algorithm

⊞ Recognition stage

output of the net is forced to match that on imposed unknown pattern at time zero.

NB

→ net is allowed to iterate freely in discrete time steps

Stable situation

when the output remains unchanged

⊞ Operation of Hopfield network:

★★

→ initialise the network

→ input unknown pattern

→ iterate to convergence

Energy function for Hopfield net:

$$E = -\frac{1}{2}\sum_{i}\sum_{j\neq i} w_{ij} x_i x_j + \sum_{i} x_i T_i$$

where,

w_{ij} = weight between node i to node j

x_i = output from node i

x_j = " " " j

T_i = threshold value of node i

NB

$w_{ii} = 0$; i.e. i=j

$w_{ij} = w_{ji}$, as connections are symmetric

Relation between Number of node and Number of pattern

Chapter-3
A Look at Fuzzy Logic

Proposition:

A meaningful statement.

It can be true in one occasion and false on another.

→ truth value of true proposition → 1

→ " " of false " → 0

Fuzzy logic

It is a form of many-valued logic in which truth values of variable may be any real number between 0 & 1.

both inclusive.

→ employed to handle the concept of partial truth.

→ deals with propositions that can be true to a certain degree

Fuzzy Logic Vs Crisp Logic

Fuzzy Logic	Crisp Logic
1) multivalued logic	1) boolean or classical valued
2) values of fuzzy logic are between 0 to 1	2) Values of crisp logic can be 0 or 1

Fuzzy Logic	Crisp Logic
3) It supports a flexible sense of membership of element to a set.	3) It doesn't support this

Fuzzy set:

Those sets whose elements have the degree of membership.

A fuzzy set is a pair (U, m) where

U is a set and

$m : U \to [0,1]$ a membership fun

→ grade of membership. a candidate for inclusion/ universe of dis rse

~~Fuzzy set operation~~

Application of fuzzy logic:

Applied in the fied of
→ artificial intelligence
→ engineering
→ Computer science
→ operations research
→ robotics
→ pattern recognition

Commercial application of FL

→ A Subway in Sendai, Japan uses a fuzzy controller

→ to control a subway car

→ Camera and cam corders use fuzzy logic

→ to adjust autofocus mechanism

→ to cancel the jitter

→ caused by a shaking hand

→ Some automobile use fuzzy logic

for different control applicat

★ Nissan has patent on fuzzy logic breaking

→ breaking system

→ transmission control

→ fuel injectors.

→ Software applications

↳ to search and match images for certain pixel regions of interest.

⑩ Full Pixel Search:

Avion System S/W package

→ SuperCharts:

A stock market charting and research tool → from Omega Repear[ch] to determine wheather the market is

- bullish
- bearish
- neutral

Operation of Fuzzy set

→ Union
→ intersection
→ Complement

fit vector:

- If a, b, c, d are such that their degree of membership in fuzzy set A are 0.9, 0.4, 0.5 and 0. the fuzzy set A is given by the fit vector (0.9, 0.4, 0.5, 0)

Fit values:

The components of the fit vector are called fit values

Union:

Union of two fuzzy sets, is the maximum value of its degree of membership within the two fuzzy sets forming a union

$$A \cup B = \mu_{A \cup B}$$

$$= \max(\mu_A(x), \mu_B(x))$$

Intersection:

Intersection of two fuzzy sets is the minimum or the smaller value of its degree of membership

→ individually in the two sets forming the intersection

$A \cap B = \mu_{A \cap B} = \min(\mu_A(x), \mu_B(x))$

Complement:

Complement of a fuzzy set A is ~~the~~ defined by $\boxed{\mu_{\neg A}(x) = 1 - \mu_A(x)}$

Product of two fuzzy set:

$$\boxed{A \cdot B \rightarrow \mu_{A \cdot B} = \mu_A(x) \cdot \mu_B(x)}$$

Difference of two fuzzy set:

$$\boxed{A - B = A \cap B^c}$$

Ex:

Given that,

$A = \{(x_1, 0.4), (x_2, 0.2)\}$

$B = \{(x_1, 0.3), (x_2, 0.5)\}$

$\therefore B^c = \{(x_1, 0.7), (x_2, 0.5)\}$

$\therefore A - B = A \cap B^c$

$= \{(x_1, 0.4), (x_2, 0.2)\}$

Ans

Disjunctive sum (Exclusive OR)

$$A \oplus B = (A^c \cap B) \cup (A \cap B^c)$$

$$\therefore \mu_{A \oplus B}(x) = (A^c \cap B) \cup (A \cap B^c)$$

Sum of two fuzzy sets:

$$\mu_{A+B}(x) = [\mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)]$$

Product with a crisp:

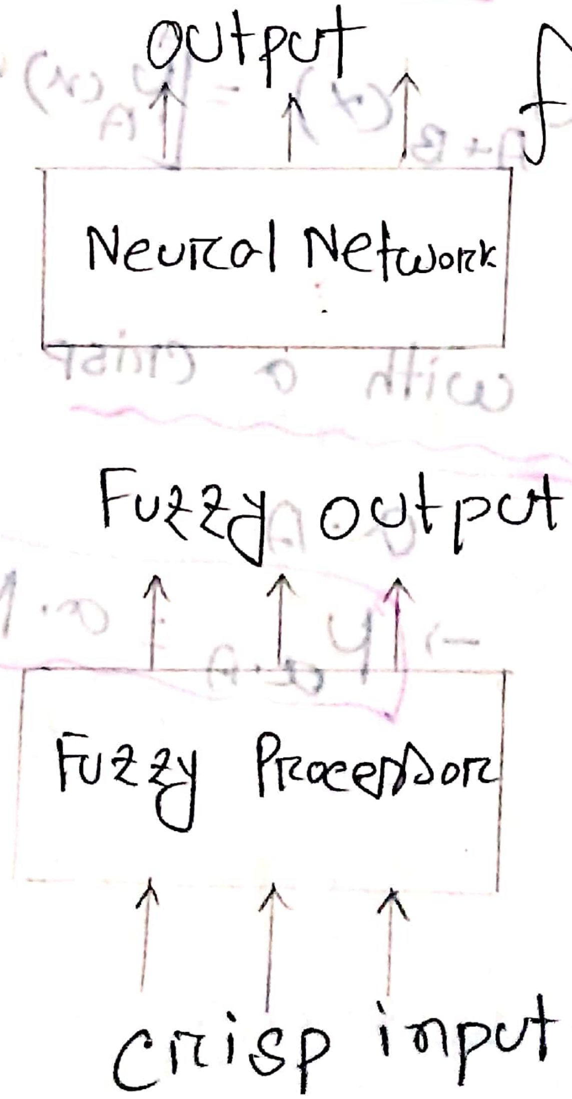
$$a \cdot A$$

$$\rightarrow \boxed{\mu_{a \cdot A} = a \cdot \mu_A(x)}$$

Fuzziness in Neural Network:

→ use fuzzifier (a) function to preprocess data

→ " a " " to post-process



for a neural network

Fuzzy control System:

→ it is a closed loop system

that uses the process of fuzzification

→ defuzzification is also used in FLC

to create crisp real values

machine under control

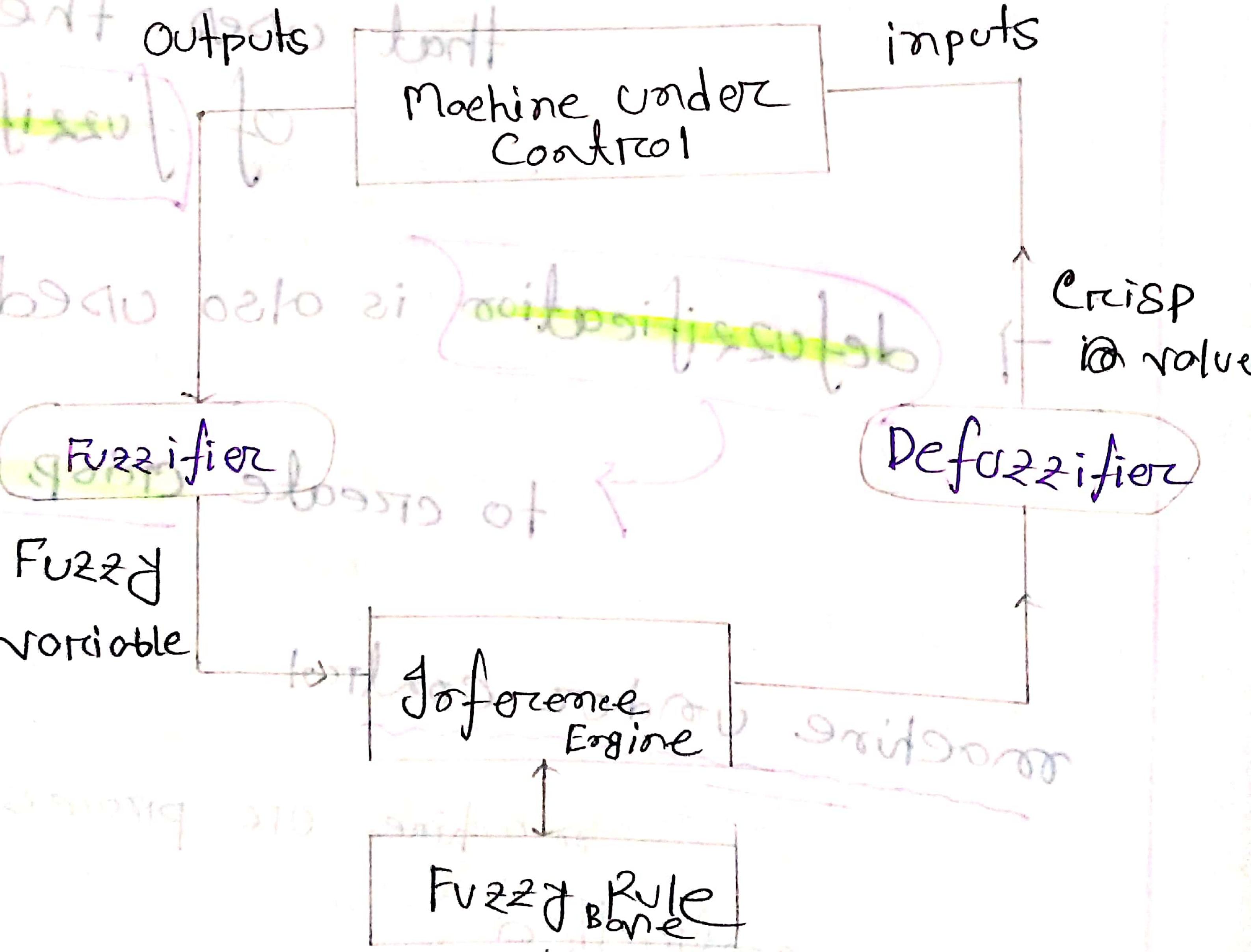
machine or process that is controlled

output:

measured response behaviors of user machine fuzzy outputs

some output pass et

through ~~the~~ a fuzzifier



interface engine / fuzzy rule base:

Converts fuzzy outputs to actions to take by accessing fuzzy rules in a fuzzy rule base

input

there are the (crisp) dials on the machine to controls its ~~bera~~ behavior

☆ key to development a fuzzy logic controller

→ iteratively construct a fuzzy rule base

~~Member~~

Fuzzy-membership function:

~~A mem~~

Membership function for a fuzzy set A on the universe of discourse X is defined as $\boxed{\mu_A : X \to [0,1]}$

where, each element of X is mapped to a value between 0 & 1

→ allows us to graphically represent a fuzzy set.

→ X axis represents → universe of discourse

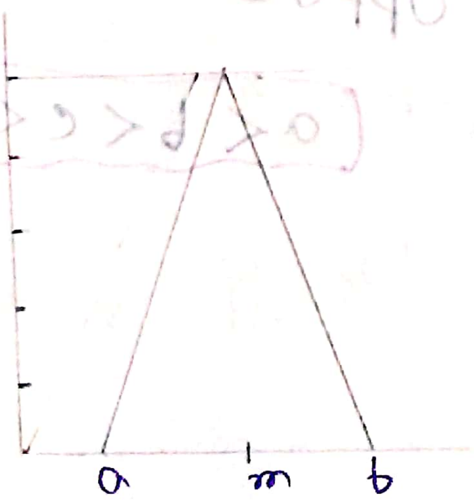
→ Y " " → degree of membership in the [0,1] inter.

List of membership function:

- → Tringular membership function
- → Trapezoidal " "
- → Gaussian " "
- → Piecewise linear " "
- → Singleton " "

Triangular function:

Defined by
a lower limit a
an upper " b
and a value m
where, $\boxed{a < m < b}$



$$\mu_A(x) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{m-a} & a < x \leq m \\ \frac{b-x}{b-m} & m < x < b \\ 0 & x \geq b \end{cases}$$

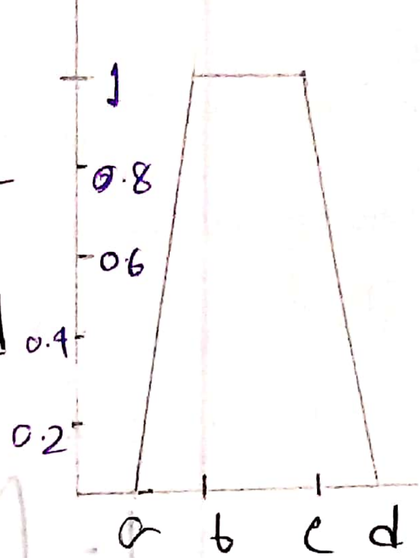
Trapezoidal function:

Defined by

- a lower limit a
- an upper " d
- a lower support limit b
- an upper " " c

where, $\boxed{a < b < c < d}$

$$\mu_A(x) = \begin{cases} 0 & (x<a) \text{ or } (x>d) \\ \dfrac{x-a}{b-a} & a \le x \le b \\ 1 & b \le x \le c \\ \dfrac{d-x}{d-c} & c \le x \le d \end{cases}$$



Two special case

→ R- function

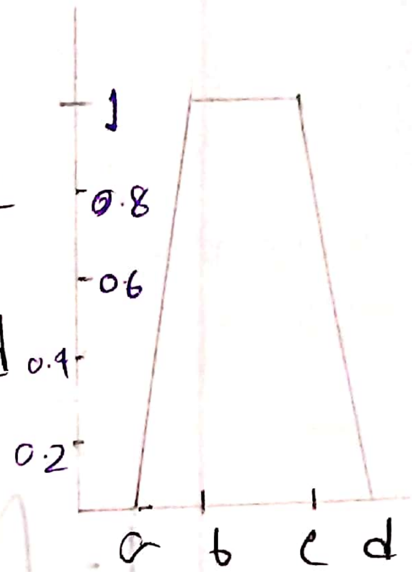
→ L- functions

R- functions:

with parameters $a = b = -\alpha$

$$\therefore \mu_A(x) = \begin{cases} 0 & x>d \\ \dfrac{d-x}{d-c} & c \le x \le d \\ 1 & x<c \end{cases}$$

$$\mu_A(x) = \begin{cases} 0 & (x<a) \text{ or } (x>d) \\ \dfrac{x-a}{b-a} & a \le x \le b \\ 1 & b \le x \le c \\ \dfrac{d-x}{d-c} & c \le x \le d \end{cases}$$



Two special case

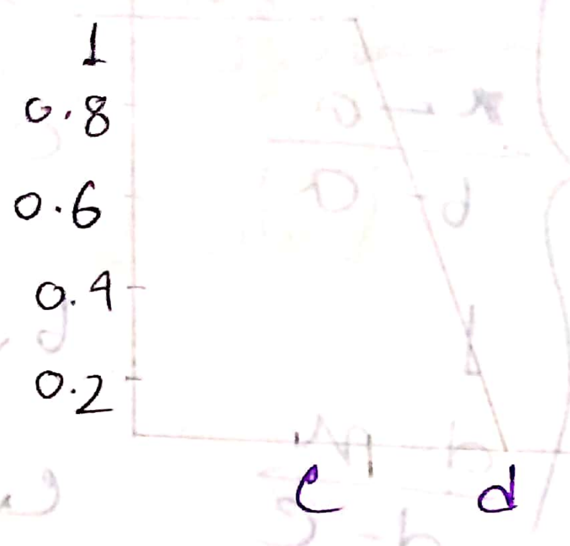
→ R-function

→ L-function

R-functions:

with parameters $a = b = -\alpha$

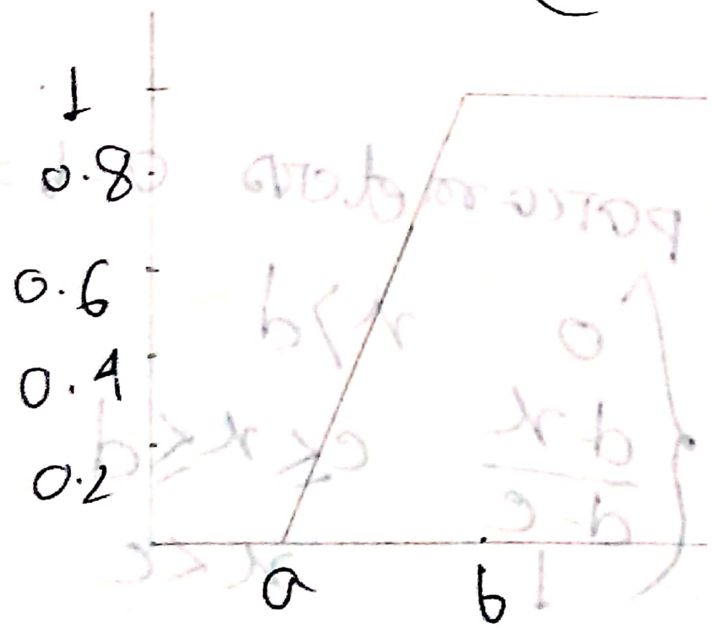
$$\therefore \mu_A(x) = \begin{cases} 0 & x > d \\ \dfrac{d-x}{d-c} & c \le x \le d \\ 1 & x < c \end{cases}$$



L- functions:

with parameters $c = d = +\alpha$

$$\mu_A(x) = \begin{cases} 0 & x < a \\ \dfrac{x-a}{b-a} & a \leq x \leq b \\ 1 & x \geq b \end{cases}$$

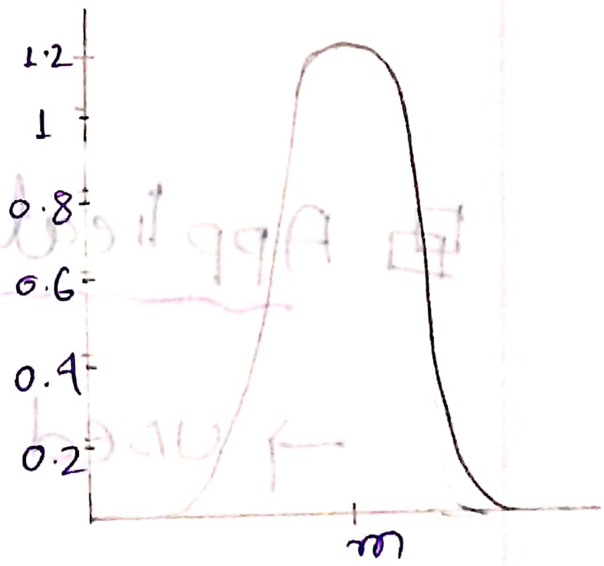


Gaussian function:

Defined by
→ a central value m
→ a standard deviation $k > 0$

The smaller k is, the narrower the "bell" is

$$\mu_A = e^{-\frac{(x-m)^2}{2k^2}}$$



Degree of membership:

The output of membership function.

→ value is always limited to between 0 and 1

Application of membership function

→ used in the fuzzification

→ " in " defuzzification

to map the non-fuzzy input values to fuzzy linguistic terms & vice versa

Chapter-16
Application of Fuzzy Logic

Section II: Fuzzy Control

□ Application of FLC function(s)

☆☆

→ Video camcoder Determine best focusing and lighting → when there is a movement in the picture

→ Washing machine adjust washing cycle by judging the dirt, size of load & type of fabric

→ Television

adjust brightness, color &
contrast of picture
↳ to please viewers

→ Vacuum cleaner

adjust the vacuum cleaner
motor power
↳ by judging the amount
of dust and dirt and
the floor characteristics

→ Hot water heater

adjust the heating element
power
↳ according to
the temperature &
the quality of water

→ Motor control Improve

the accuracy and range of motion control

→ under unexpected condition.

→ Subway train Increase

the stable drive and enhance the stop accuracy

→ by evaluating the passenger traffic conditions

→ Helicopter control Determine

the best operation actions

by judging

→ human instructions

→ flying control

Chapter-1 (GA)

A Gentle Introduction to Genetic Algorithm

Genetic Algorithm:

Genetic algorithm are search algorithm based on the mechanics of natural selection & natural genetics.

→ can solve both
- constrained &
- unconstrained

optimization problem → based on natural selection

→ developed by John Holland → at Michigan University goal of John Holland research

2 goals

1) to abstract and rigorously explain adaptive process of natural system

2) to design artificial systems s/w that retains the important mechanisms of natural system

NB central theme of research on GA → robustness

robustness:
Balance between efficiency and efficiency necessary for survival in many different environment

Conventional search methods:

- → calculus-based
- → enumerative
- → random
- → simulated annealing

NB: Conventional search methods are not robust

□ Genetic algorithm vs traditional methods:

1. GAs work with a coading of the parameter set.
   → not the parameters themselves

2. GAs search from a population of points
   → not a single point

3. GAs use payoff (objective function) information
   → not derivatives or other auxiliary knowledge

4. GAs use probabilistic transition rule

not deterministic rules

Genetic operators:

A simple genetic algorithm, generates good results, is composed of three operators:

- Reproduction
- Crossover
- mutation

☐ Reproduction: / Selection

It is a process in which individual strings are copied according to their objective function values f

It is also known as selection.

☐ Selection:

It is a stage of GA in which individual genomes are chosen from a population for later breeding.

Reproduction: / Selection

It is a process in which individual strings are copied according to their objective function values f

It is also known as selection.

Selection:

It is a stage of GA in which individual genomes are chosen from a population for later breeding.

Fitness function:

It is an objective function that determines how fit an individual is.

→ it gives a fitness score to each individual.

→ copying string according to their fitness value means that String with higher value → have a higher probability of contributing one or more offspring → in the next generation.

Importance of fitness function:

★★

i) It gives meaningful, measurable and comparable value from a given set of genes

ii) It finds a best solution from a set of solutions

iii) Evaluates individuals and reproductive success varies with fitness.

Crossover / recombination:

It is a genetic operator used to combine the genetic information of two parents to generate new offspring.

Types of crossover:

- → single point crossover
- → two ,, ,,
- → Uniform ,,
- → Arithmetic ,,
- → String

Single-point crossover:

→ A point on both parents chromosomes is picked randomely ~~and~~ designed as a cross over point

→ Bits to the right of that point are swapped.

1	2	3	4	5
3	1	5	4	2

⇒

1	2	3	4	2
1	2	3	4	5

Two-point crossover

→ two crossover points are picked randomly from the parent chromosomes

→ bits between two points are swapped

→ it is equivalent to performing

NB: two single-point cross over with different crossover points.

1	2	3	4	5	6	7	8
5	8	9	4	2	3	5	7

1	2	9	4	2	3	7	8
5	8	3	4	5	6	5	7

Uniform crossover:

→ Each (bit) gene is selected randomly from one of the corresponding genes of the parent chromosomes.

☐ Offspring:

Chromosome that is generated by the result of crossover.

☐ Mutation:

It is a genetic operator used to maintain genetic diversity from generation of a population of genetic algorithm chromosome to the next.

NB: It alters one or more gene values in a chromosome from its initial state.

Purpose of mutation operator:

★★

→ ~~ind~~

→ introduce diversity into the sampled population

→ to avoid the local minima by preventing the population of chromosomes from becoming too similar to each other.

→ to introduce new genetic structure.

→ to prevent premature convergence

→ to avoid only taking the fittest of the population → in generating the next generation.

→ to keep the gene pool well stocked

Types of mutation:

- → Bit string mutation
- → Flip Bit
- → Boundary
- → Non-uniform
- → Uniform
- → Gaussian
- → Shrink

→ for integer & float gene

Bit string mutation:

- Ensue through bit flips at random position

Ex:

1 0 1 0 0 1 0

↓

1 0 1 0 1 1 0

Flip Bit:

→ takes the chosen genome

→ inverts the bits

Boundary:

→ replaces the genome with either lower or upper bound randomly.

→ can be used for integer and float genes

Gaussian:

→ adds a unit Gaussian distributed random value → to the chosen gene

→ If it falls outside of the lower or upper bounds → for that gene.

new gene value is clipped.

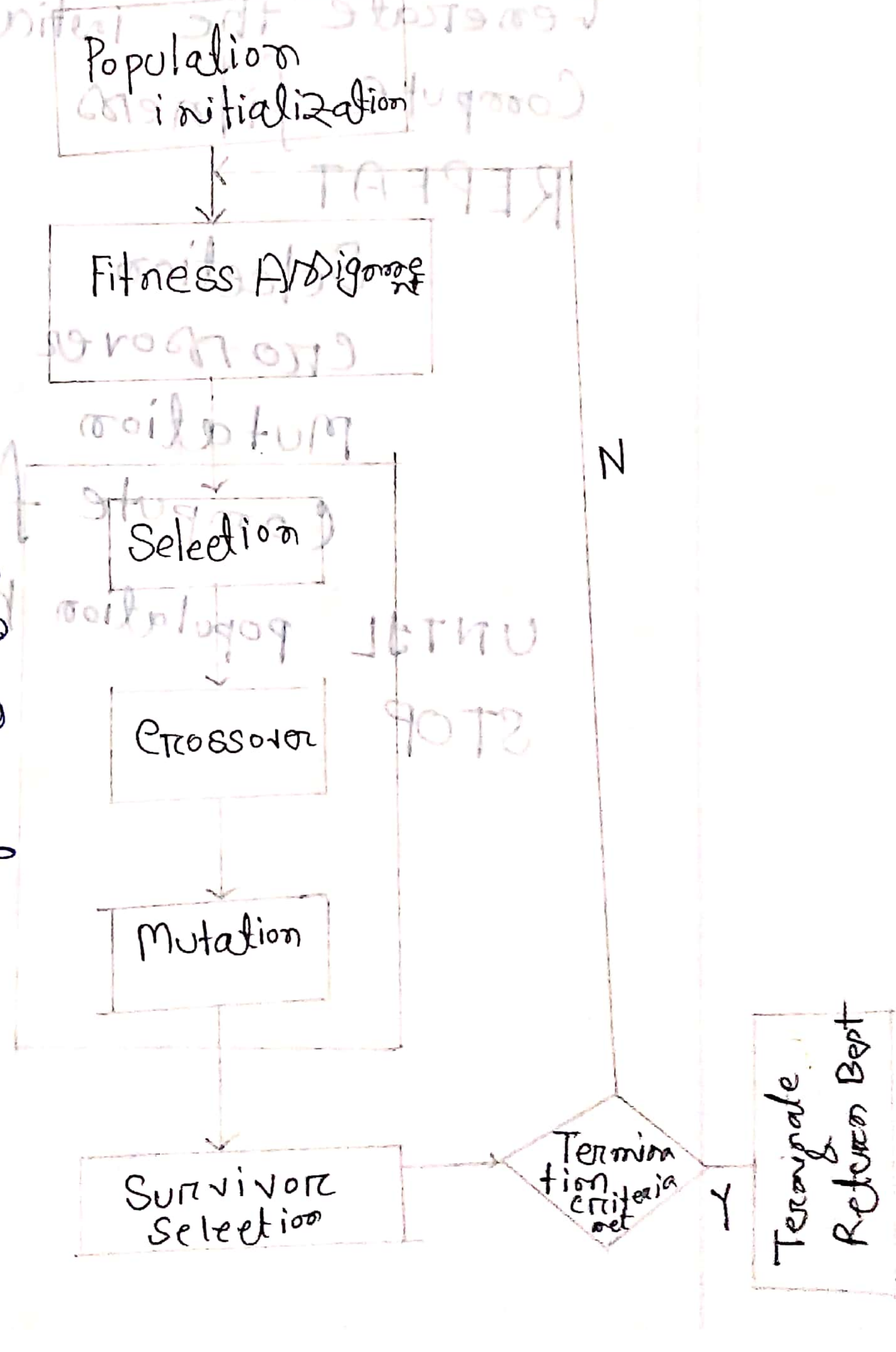
→ ~~this operator~~
→ can be used for integer and float gene

Uniform:

Replaces the value of the chosen gene with a uniform random values

between user specified upper and lower bound for that gene

⊞ Flow chart of GA:



Pseudocode for GA:

START
Generate the initial population
Compute fitness
REPEAT
    Selection
    Crossover
    Mutation
    Compute fitness
UNTIL population has converged
STOP

Method of selection in GA:

→ Roulette wheel Selection
→ Rank selection
→ Tournament selection
→ Random selection
→ Elitism "
→ Steady state selection

Tournament selection:

it can work with negative fitness values

- It is a method of choosing the individual from the set of individuals.

The winner of each tournament is selected to perform crossover.

→ Select K random chromosome
→ pick the best as parent.

Rank selection:

→ work with negative fitness values
→ used when each individuals have very close fitness.

→ First find out the fitness value
→ every population is ranked according to their fitness
→ the higher ranked individuals are preferred more than the lower ranked one.

Random selection:

Randomly select parents from existing population.

Elitism selection:

→ the best chromosome/or a few best chromosome are copied to the population in the next generation

→ can rapidly increase performance of GA.

as it prevents losing the best found solution.

Roulette wheel selection:

→ probability of choosing an individual is proportional to its fitness.

$\boxed{\text{Fitness} \propto \text{Chance to be selected}}$

→ probability of choosing individual i, $P_i = \dfrac{f_i}{\sum_{j=1}^{N} f_j}$

f_i → fitness of i

N → size of current generation.

→ if we are working on minimization problem, it is needed to transform it into maximization problem.

Roulette wheel selection:

→ probability of choosing an individual is proportional to its fitness.

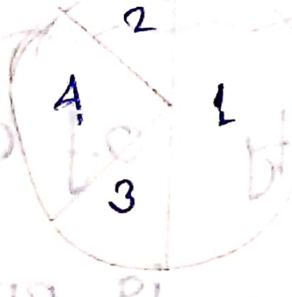
$\boxed{\text{Fitness} \propto \text{Chance to be ~~cho~~ selected}}$

→ probability of choosing individual i, $P_i = \dfrac{f_i}{\sum_{j=1}^{N} f_j}$

f_i → fitness of i

N → size of current generation.

→ if we are working on minimization problem, it is needed to transform it into maximization problem.



we use the following steps

→ compute the sum of all fitness, s

→ generate random number r with (0, s)

→ select chromosome where

$$\boxed{e_{sum} < r}$$