Genetics Algorithm → focus on optimization
→ use when population is rgada
→ best fit will survive.

→ Population

→ Calculate fitness

→ Selection

→ Cross Over

→ Mutation

→ Stoping Criteria.

TSP: Salesman needs to visit all the cities only once and return to the starting one while minimizing the total distance travelled.

- Representation: population is represented as a sequence of cities
- Calculate fitness: each route is evaluated to find fitness by calculating total distance travelled for that route
- Selection: solution are selected based on fitness, greater fitness (less distance travelled) is selected.
- Genetic Operator:
  - → Crossover: Pair of selected routes are combined to create a new route.
  - → Mutation: Random changes are introduced. means swapping or altering cities in Routes
- Replacement: new populated is replaced by the prev pop
- Iteration: 2-5 steps are repeated until termination condition satisfied
- Termination: Terminate when optimal distance is found.

Knapsack: you have set of items, each with a weight and value, and a knapsack with max weight capacity. Goal is to determine the most valuable combination of items that can be placed in the knapsack without exceeding the limit.

- Representation: population represents a combination of items that could be placed in knapsack
  - → Represented as binary string
  - 1 (item is included), (0) represents not included.
- Initialization: Creating an initial population (Randomly / by using heuristic)
- Evaluation: evaluate every solution.
- Selection: Select parent based on fitness (total value) higher total value higher chance to be selected.

• Genetic Algorithms / Operator:

to create new offspring

→ Crossover: Combined parts of two selected parents to generate a new solution

eg parent 1 → 10 10 | 10 10 1 1

parent 2 → 1 1 0 | 1 0 1 0 0 1 0 .

new parent 1 → 1 0 1 0 1 0 0 1 0

new parent 2 → 1 1 0 1 1 0 1 0 1 1

→ Mutation: Involves randomly changing some bits in the solution.

eg 1 0 1 0 0 1 0 0 1 0

flip the 8 bit from 0 → 1 1 0 1 0 0 1 0 1 1 0

• Replacement: New population replaces prev once

• Iteration: Repeat steps 3-6 (evaluation - Replacement

• Termination: when optimal solution found (terminate

Naive Bayes Classification:

→ Bayes Theorem: m estimate = ?

Person	Covid (Y/N)	Flue (Y/N)	Fever (F/N)
1	Y	N	Y
2	N	Y	Y
3	Y	Y	Y
4	N	N	N
5	Y	N	Y
6	N	N	Y
7	Y	N	Y
8	Y	N	N
9	N	Y	Y
10	N	Y	N

P(fever | Yes) = 7/10

P(fever | No) = 3/10

	Yes	No
Flue	3/7	2/3
Covid	4/7	2/3

Given Person (flu, Covid): 3/7 * 4/7 * 7/10 = 0.17

$P(Yes \mid flu, Covid) = P(flu \mid Yes) * P(Covid \mid Yes) * P(Yes)$

$P(No \mid flu, Covid) = P(flu \mid No) * P(Covid \mid No) * P(No)$

$= 2/3 * 2/3 * 3/10 = 0.13$

As the Yes fever class value is greater we predict it to be in Yes data.

Id3.

Decision Tree.

entropy / info gain → high → root node.

outcomes → on leaf node.

→ ML algo used for classification / regression task

→ ~~Decision~~ Tree Structure

→ Decision nodes

→ Leaf Nodes

→ Splitting

→ Entropy & information gain

→ Pruning.

Day	Weather	Temp	Humidity	Wind	Play F/B
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Cloudy	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Cloudy	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Cloudy	Mild	High	Strong	Yes
13	Cloudy	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

3 4 7 8 6

we have to find entropy & info gain to make decision Tree.

→ To find root node we have to find ent & info of every data set.

Calculate IG of weather.

Step 1, entropy of entire dataset → play f/b.

$$S\{+9,-5\} = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

Step 2 : entropy of all attribute of weather.

entropy of sunny: $S\{+2,-3\} = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.97$

entropy of Cloudy: $S\{+4,-0\} = \frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0$

entropy of Rainy: $S\{+3,-2\} = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$

Step 3 Information Gain = entropy of complete data set $- \frac{5}{14}$ Ent(S) $- \frac{4}{14}$ Ent(C) $- \frac{5}{14}$ Ent(R).

$$IG = 0.94 - \frac{5}{14}(0.97) - \frac{4}{14}(0) - \frac{5}{14}(0.97) = 0.247$$

Calculate Info Gain of Temp.

Step1, Entropy of entire.

$$S\{+9,-5\} = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94.$$

Step2. Entropy of all attributes.

→ entropy of Hot $\{+2,-2\} = \frac{-2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1.$

→ entropy of Cold $\{+3,-1\} = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.81$

→ entropy of Mild $\{+4,-2\} = -\frac{4}{6}\log\frac{4}{6} - \frac{2}{6}\log\frac{2}{6} = 0.91.$

Step 3, Information Gain:

$$= 0.94 - \frac{4}{14}(1) - \frac{4}{14}(0.81) - \frac{6}{14}(0.91) = 0.03$$

Calculate Information Gain of Humidity:

Step 1, entropy of entire dataset → 0.94.

Step 2: entropy of all attributes.

→ entropy of High [+3, -4] = $\frac{-3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7}$ = 0.

→ entropy of Normal [+6, -1] = $\frac{-6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7}$ = 0.59

Step 3: Information Gain

$$= 0.94 - \frac{7}{14}(0.98) - \frac{7}{14}(0.59) = 0.15$$

Calculate Information Gain of Wind.

Step 1: 0.94.

Step 2: all attributes

entropy of Weak [+6, -2] = $\frac{-6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8}$ = 0.8

entropy of Strong [+3, -3] = $\frac{-3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6}$ = 1

Step 3: $0.94 - \frac{8}{14}(0.81) - \frac{6}{14}(1) = 0.04$

Gain (Weather) = 0.24	
Gain (Temp) = 0.03	→ This gain will tell us about the root node which is weather (highest) gain.
Gain (Humidity) = 0.15	
Gain (Wind) = 0.04	

Sunny Cloudy Rain.

↓

Yes.
(because all are Yes).

For Sunny

Day	Weather		Temp	Humidity	Wind	Play P/E
1	Sunny	No	Hot	High	Weak	No
2	Sunny	No	Hot	High	Strong	No
8	Sunny	No	Mild	High	Weak	No
9	Sunny	Yes	Cold	Normal	Weak	Yes.
11	Sunny	Yes.	Mild	Normal	Strong	Yes.

Step 1: Entropy of entire data set → Temp

$S[+2, -3] = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.97$

Step 2. all attributes,

→ entropy of Hot {0, -2} = $\frac{-0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$

→ entropy of mild {+1, -1} = $\frac{-1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$

→ entropy of cold {+1, 0} = $\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1} = 0$.

Step 3 : Info gain

$= 0.97 - \frac{2}{5}(0) - \frac{2}{5}(1) - \frac{1}{5}(0)$ **= 0.57**.

↳

Info gain of Humidity :

Step 1 : 0.97

Step 2 :

entropy of High {0, -3} = 0

entropy of Normal {+2, 0} = 0.

Step 3 : **0.97**

↳

Info gain of Wind.

Step 1. 0.97

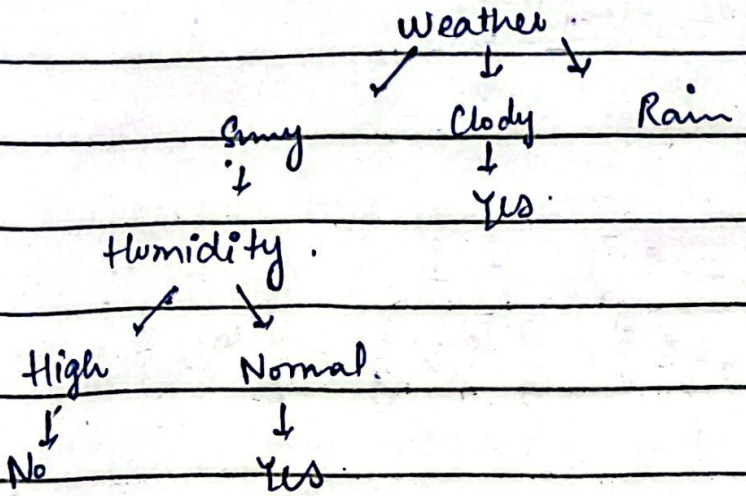
Step 2 :

E(S) = {+1, -1}₂ 1

E(w) {+1, -2} = $\frac{-1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}$ **= 0.91**

↳

Info Gain (Temp) = 0.57 } Humidity
Info gain (Humidity) = 0.97 } ment
Info gain (Wind) = 0.91. } root node.



Now Solve for Rain:

Day	Weather	Temp	Humidity	Wind	Play F/B)
4	Rain.	Mild	High	Weak	Yes
5	Rain	Cool.	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No.
10	Rain	Mild.	Normal	Weak	Yes.
14	Rain	Mild	High	Strog	No.

Info gain of Temp.

Step1: $4+3,-2\xi = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$

Step2:

$S(\text{mild}) = 4+2,-1\xi = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.91$

$S(\text{Cool}) = 4+1,-1\xi = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$

Step3:

$$I.G = 0.97 - \frac{3}{5}(0.91) - \frac{2}{5}(1) = 0.024$$

Info of Humidity

Step 1: 0.97

Step 2:

$S(\text{high}) = \{+1, -1\} = 1$

$S(\text{Normal}) = \{+2, -1\} = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.91$

Step 3:

$I.G = 0.97 - \frac{2}{5}(1) - \frac{3}{5}(0.91) = 0.024$

Info of Wind

Step 1: 0.97

Step 2:

$S(\text{Weak}) = \{+3, 0\} = 0$

$S(\text{Strong}) = \{0, -3\} = 0$

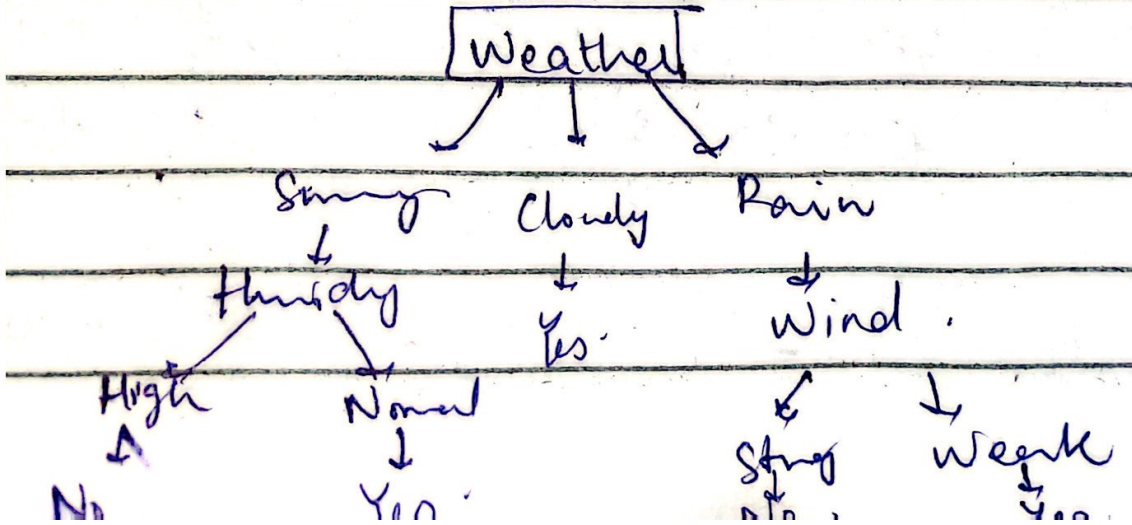
Info gain $= 0.97$

$S(\text{Temp}) = 0.024$

$S(\text{Humidity}) = 0.024$

$S(\text{Wind}) = 0.97.$

} Wind (highest)



Reduced error pruning post rule pruning.

Issues in Decision Tree.

1. Overfitting the Data. (it will work good for training data but not for testing data)
2. Incorporating Continuous valued attributes (convert continous values to discrete value)
3. Handling training examples with missing attribute values.
4. Handling attributes with different costs.
5. Alternative measures for selecting atributes.
6. Determining depth of tree for correct classification

Avoid Overfitting in Decision Trees.

→ Why overfitting occures?

Building data that "(adapt/rely) to much on the" toraing data may lead to overfitting

2. Continuous Values →

eg.

age	bought product
25	Yes
30	No
35	Yes
40	Yes
45	No
50	Yes.

when we have continous data determine a threshold to split the ages into groups to build the decision tree.

threshold = 37.5

grp 1 ← if age <= 37.5

grp 2 ← if age > 37.5.

3. Missing value:

1) Completely ignor (but not prefearable)

2) fill the missing value with estimated or calculated value.

Study Hrs	Test Score	Grade
4	85	A
5	90	B
6	?	e
7	80	A
?	75	B
3	88	A

Mean Test Score:

$= \frac{85+90+80+88}{4} = 85.7$

↓

replace it at 3rd Test score.

Random Forest.

→ extension of decision Tree

→ Decision Tree → 1 tree Random Forest → Multiple Tree

→ ensemble (group) learning.

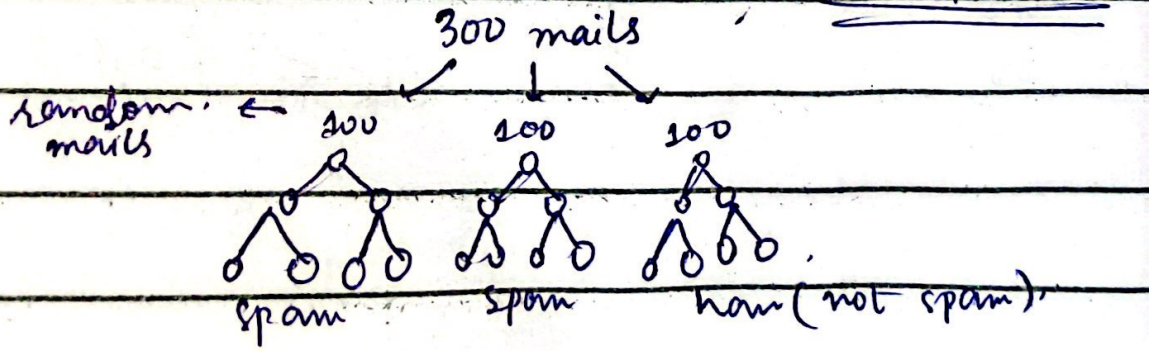
→ no overfitting (traing data works perfect but when we put testing data it did work → overfitting).

→ use in classification & regression

↓

more accurate

→ classification



majority wins → spam

→ Regression → neumeric data.

10 20 15

mean value is the output := $\frac{10+20+15}{3} = 15$.

↔

Neural Networks — google Translate etc.

neuron — connection (sharing of data)

Artificial : Brain → Machine

eg: human brain → recognize tony stark even if she has beard or shaved bcz our minds are trained but machine might took some time to recognize. on the other hand 8^8 → machine can give output much quicker than human brain.

learning algorithm :

Artificial:

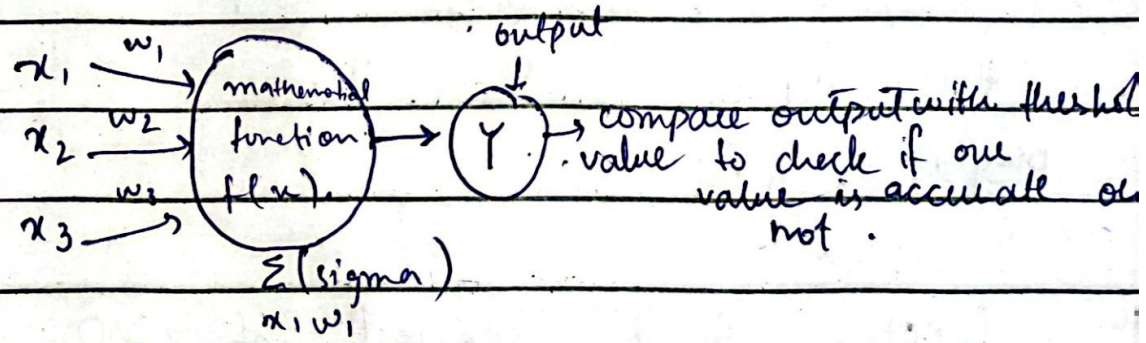
Input data → Meaningful info → learning by Neural improvement Networks

Biological:

Dendrites (take inputs) → Cell body (process the data) → axons (pass our data through them)

→ Terminal axons (connected with next dentrites)

Artificial.



MCP.

Mcculoch Pits model.

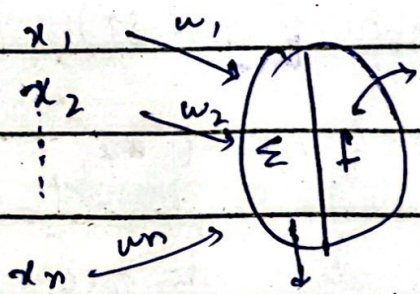
→ 1st biological neuron model (1943 by Warren Mccholeb & Walter Pitts)

→ aka (linear threshold Gate model).

→ Directed weight graph is used for connecting neurons.

→ Basic building block of neural network

→ two possible state → Active (1)
→ Silent (0).



it produces the output using threshold (T)

output = { 1 if x ≥ T
0 if x < T }

aggregates the weighted input into single numeric value.

$\Sigma x_1 w_1 + x_2 w_2 \cdots x_n w_n \cdot] X$.

Bias / Threshold → min value of weighted active input for a neuron to fire.

→ if weighted i/p is larger than T then o/p → 1 else 0.

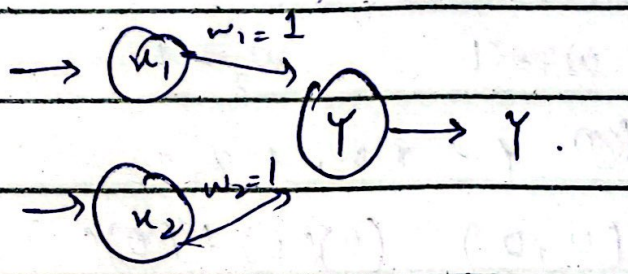
output: $f(a)$ ↳ $\Sigma \cdot w_i x_i - T$.

this is a function that is { $\theta(x) = 1$ if $x \geq 0$.
$\theta(x) = 0$ if $x < 0$

Implement of MCP. (in AND gate).

x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0

Threshold → neuron will fire if both inputs are high in this case.



$w_1 = 1 \quad w_2 = 1$

$(1,1)$; $y_{in} = x_1w_1 + x_2w_2 = (1)(1) + (1)(1) = 1+1 = 2$.

$(1,0)$; $y_{in} = x_1w_1 + x_2w_2 = (1)(1) + (0)(1) = 1+0 = 1$.

$(0,1)$; $y_{in} = (0)(1) + (1)(1) = 0+1 = 1$.

$(0,0) = (0)(1) + (0)(1) = 0+0 = 0$

if θ (threshold) value is ≥ 2, (neuron will fire).

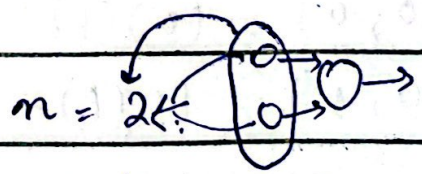
can also be obtained by

$$\theta \geq nw - p$$

$n \rightarrow$ no of neurons.

$w \rightarrow$ +ve weight

$p \rightarrow$ neg. weight



$w = 1$.

$p = 0$ (no inhibitory weights)

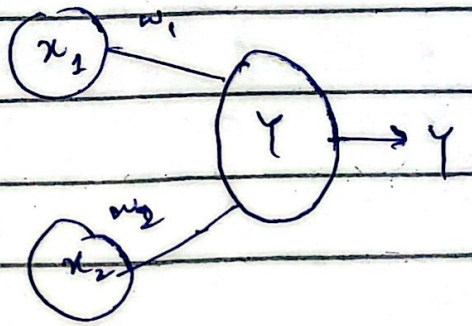
$\theta \geq (2)(1) - 0$

$\theta \geq 2$.

output: $\begin{cases} \text{if } \theta \geq 2 & 1 \\ \theta < 2 & 0 \end{cases}$

Implementation of MCP (ANDNOT).

x_1	x_2	Y
0	0	0
0	1	0
1	0	1
1	1	0



∴ $w_1 = 1$ $w_2 = 1$

$Y = x_1 w_1 + x_2 w_2$.

$(0,0) = (0)(1) + (0)(1) = 0$

$(0,1) = (0)(1) + (1)(1) = 1$

$(1,0) = (1)(1) + (0)(1) = 1$

$(1,1) = (1)(1) + (1)(1) = 2$.

Change weight values:

$w_1 = 1$ $w_2 = -1$

$(0,0) = (0)(1) + (0)(-1) = 0$

$(0,1) = (0)(1) + (1)(-1) = -1$

$(1,0) = (1)(1) + (0)(-1) = 1$

$(1,1) = (1)(1) + (1)(-1) = 0$

if $\theta = 1$ then neuron will fire.

$= x_1 x_2 - \sim_1 - \sim_2$

$\longleftrightarrow$

Artificial Neural Network.

" An ANN is an information-based model that is inspired by the way a biological nervous system works, such as brain ".

→ This model tries to replicate the most basic function of the brain.

→ ANN is composed of highly interconnected processing units (neurons) to solve specific problem.

→ Like human being, ANN learn by example.

→ configued for a specific application.

→ each neuron is conned by a "link"

→ each link is associated with weight which contain info about input signal.

→ This info is used by neural network to solve a prob

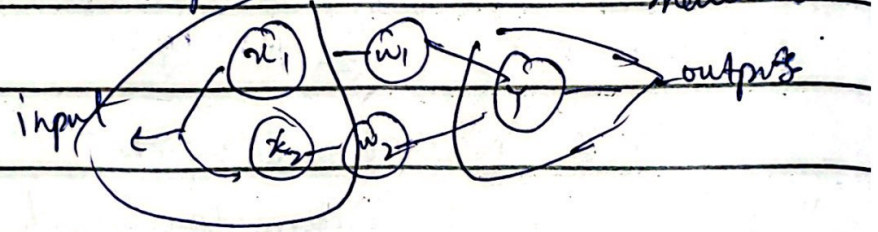
→ they (ANN) have the ability to learn, recall ay train pattern or data

Neuron

→ each neuron has an internal state of its own

→ This internal state is called activation level of neuron.

→ the activation signal is transmitted to other neurons



3 Basic Components of ANN:

→ model synapti interconnections

→ training or learning rules adopting for updating ay adjusting the connection weights

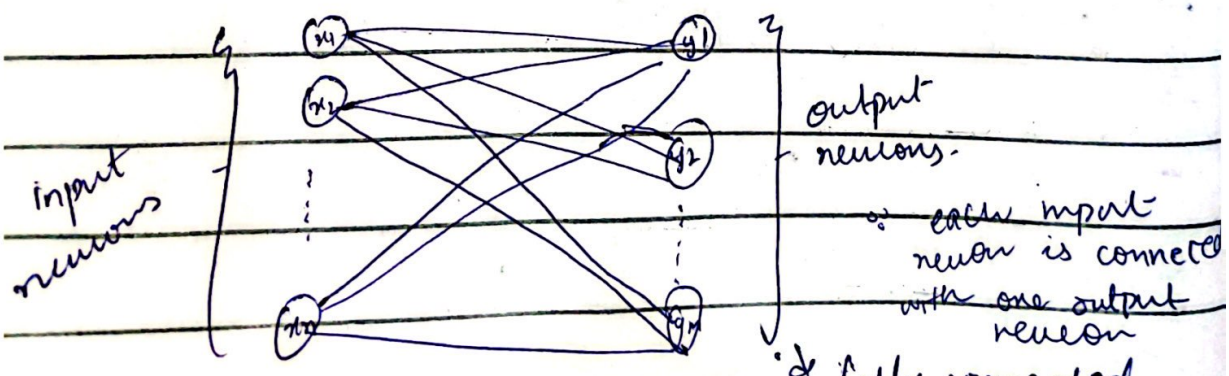
→ the activation function.

single layer feed forward
multilayer "
single node with its own fe
single layer recurrent net
multilayer "

1 point

2. Connection

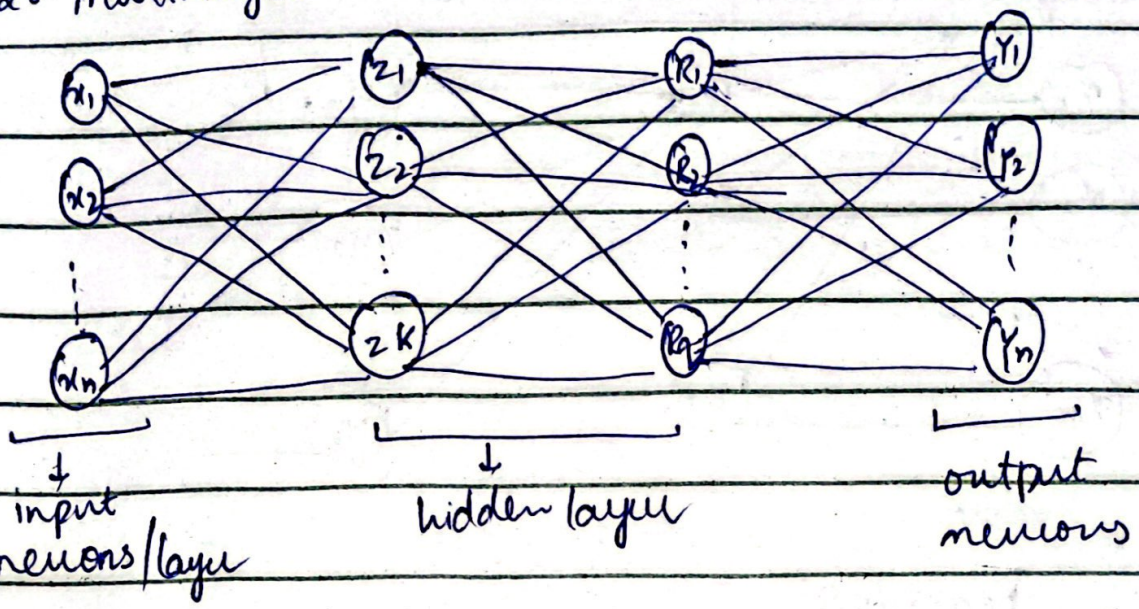
① Single layer feed forward network.



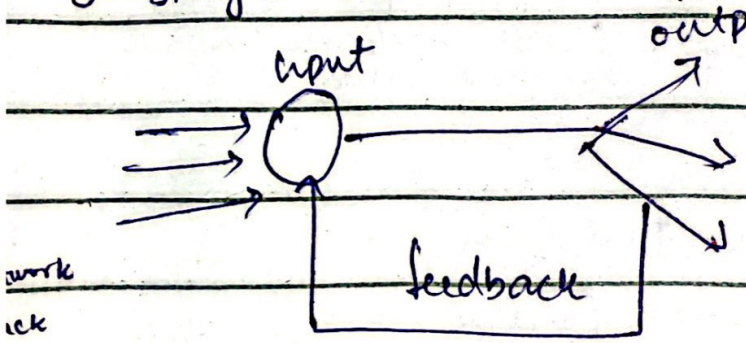
∴ each input neuron is connected with one output neuron

& fully connected.

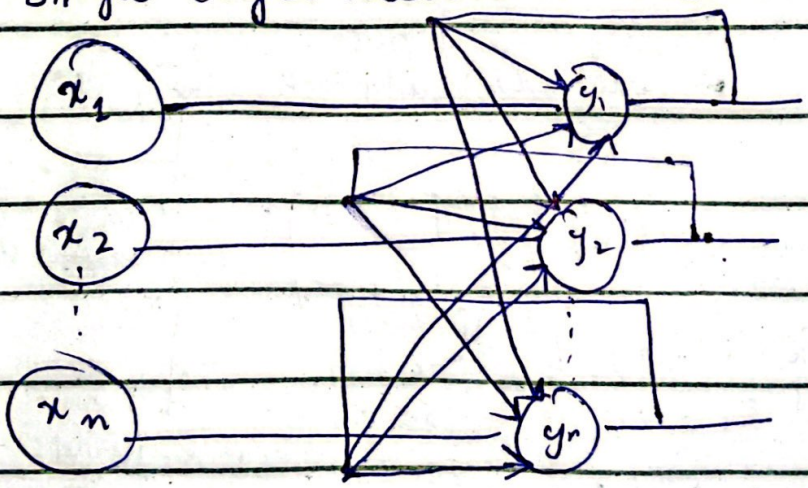
2. Multilayer



3. Single-Node with feedback.

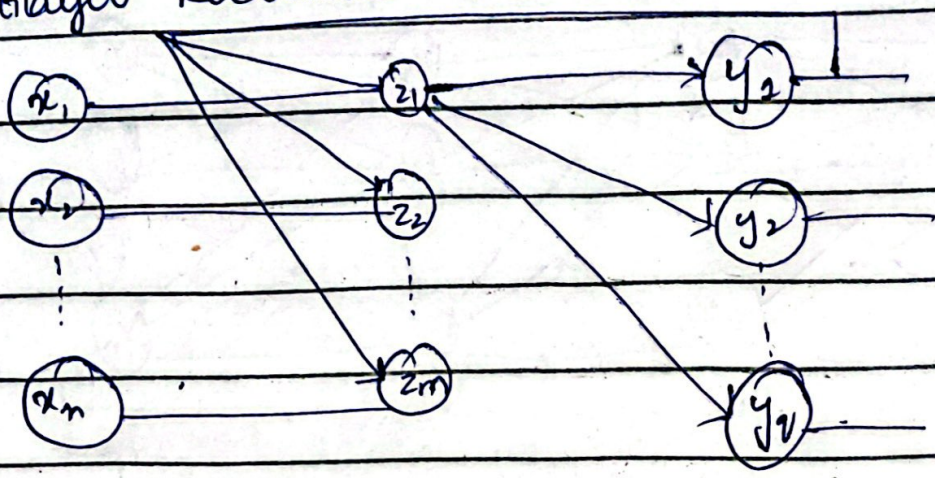


4. Single layer recurrent network.



* output will become inputs

6. Multilayer Recurrent Network



(2) point learning

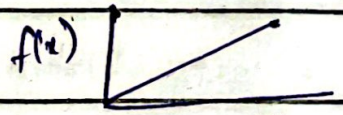
(1) Parameter learning: updates the connecting weight.

(2) Stuctural learning: focuses on change in network stucture.

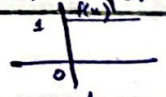
3rd point Activation Function.

There are several activation funcs.

1) Identity function: linear → $f(x) = x$ for all x.

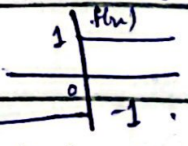


∵ output = input
→ input uses identity activation function.



2) Binary step function: output → binary (0,1). $f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ 0 & \text{if } x < \theta \end{cases}$ ∵ used in single layer nets

3) Bipolar step function: output → bipolar (+1, -1). $f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ -1 & \text{if } x < \theta \end{cases}$



∵ used in single layer nets.

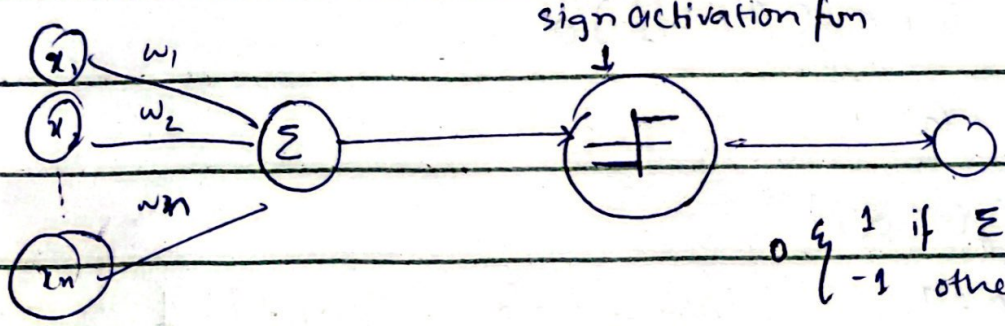
4) Ramp function: $f(x) = \begin{cases} 1 & \text{if } x > 1 \\ x & \text{if } 0 \leq x \leq 1 \\ 0 & \text{if } x < 0 \end{cases}$

Perceptron

→ used to build ANN

→ takes real valued inputs, calculates a linear combination, then output 1 if result > threshold and -1 if < threshold

$$\theta(x_1 \cdots x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$



$$o \begin{cases} 1 & \text{if } \Sigma > 0. \\ -1 & \text{otherwise} \end{cases}$$

AND GATE PRECEPTRON TRAINING RULE

→ ANN.

→ $w1 = 1.2$

$w2 = 0.6$

Threshold = 1

learning Rate = 0.5

A	B	A^B
0	0	0
0	1	0
1	0	0
1	1	1

(i) $wi * xi$

$y = x_1 w_1 + x_2 w_2$.

• $(0,0) = (0)(1.2) + (0)(0.6) = 0$.

Y < threshold As, targeted value (A^B) and output is same so no weight updation.

$(0,1) = (0)(1.2) + (1)(0.6) = 0.6$ " " ".

$(1,0) = (1)(1.2) + (0)(0.6) = 1.2$

As, Y > ~~targeted~~ threshold value. output = 1, As target (A^B) & output are diff there would be some weight updation.

→ Through Perceptron Training Rule:

$w_i' = w_i + n(t - o)x_i$

(o → actual output; t → targeted value)

for 1st case: $x_1 = 1, B = 0$ $t = 0$. $o = 1$. $n = 0.5$.

$w_1 = w_1 + n(t-o)x_1$

$w1 = 1.2 + 0.5(0 - 1)1$

$= 0.7$

$w_2 = w_2 + n(t-o)x_2$

$w2 = 0.6 + 0.5(0-1)0$

$= 0.6$

new weights:

$w_1 = 0.7$ $w_2 = 0.6$. Threshold = 1 $n = 0.5$

(i) $Y = x_1w_1 + x_2w_2$.

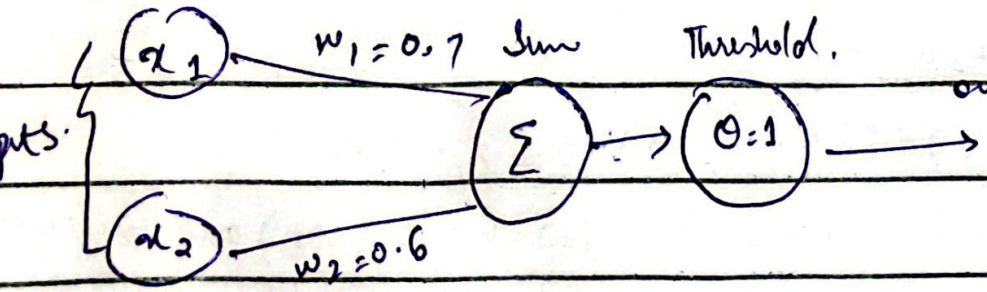
$(0,0) = (0)(0.7) + (0)(0.6) = 0$.

$(0,1) = (0)(0.7) + (1)(0.6) = 0.6 = 0$

$(1,0) = (1)(0.7) + (0)(0.6) = 0.7 = 0$

$(1,1) = (1)(0.7) + (1)(0.6) = 1.3$.

Y > threshold output = 1 ∴ output = (A^B) fals
no weight updation.



OR GATE PERCEPTRON TRAINING RULE

→ ANN

$w_1 = 0.6$

$w_2 = 0.6$

Threshold = 1

learning rate = $n = 0.5$

A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

(i) $y = x_1 w_1 + x_2 w_2$.

$(0,0) = (0)(0.6) + (0)(0.6) = 0$

$(0,1) = (0)(0.6) + (1)(0.6) = 0.6 = 0$.

$(1,0) = (1)(0.6) + (0)(0.6) = 0.6 = 0$

→ targeted value = 1. output = 0.

weight updation:

$w_i = w_i + n(t-o)x_i$

$w_1 = w_1 + n(t-o)x_1$

$w_1 = 0.6 + 0.5(1-0)0$

$w_1 = 0.6$

$w_2 = 0.6 + 0.5(1-0)1$

$w_2 = 1.1$

∵ $w_1 = 0.6$ $w_2 = 0.6$

$x_1 = 0$ $x_2 = 1$

$n = 0.5$ $t = 1$.

$o = 0$.

new weights =

$w_1 = 0.6$ $w_2 = 1.1$.

$Y = x_1 w_1 + x_2 w_2$. $w_1 = 0.6$, $w_2 = 1.1$

$(0,0) = (0)(0.6) + (0)(1.1) = 0$.

$(0,1) = (0)(0.6) + (1)(1.1) = 1.1$

$Y >$ threshold $0 = 1$ $0 == t$ (no weight updations).

$(1,0) = (1)(0.6) + (0)(1.1) = 0.6$

$Y <$ threshold $0 = 0$ $0 != t$ (weigut updati

∴ $w_1 = 0.6$, $w_2 = 1.1$, $x_1 = 1$, $x_2 = 0$, $\eta = 0.5$, $0 = 0$, $t = 1$.

$w_i = w_i + \eta(t - 0) x_1$

$w_1 = w_1 + \eta(t - 0) x_1$

$w_1 = 0.6 + 0.5(1 - 0)1 =$

$= 1.1$

$w_2 = 1.1 + 0.5(1 - 0)0$

$= 1.1$

new weigets:

$w_1 = 1.1$ $w_2 = 1.1$.

$Y = x_1 w_1 + x_2 w_2$

$(0,0) = (0)(1.1) + (0)(1.1) = 0$

$0 <$ threshold output = 0 $0 == t$ (no weigut updatio

$(0,1) = (0)(1.1) + (1)(1.1) = 1.1$

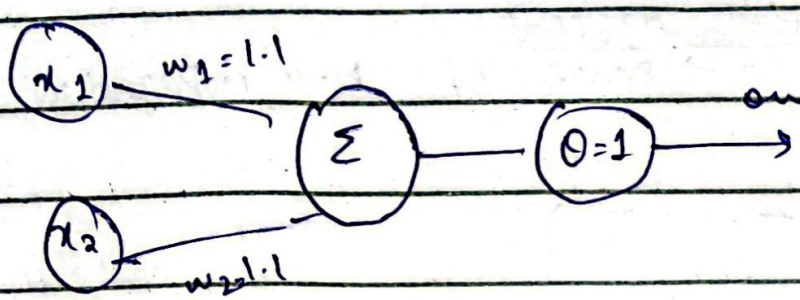
$Y >$ threshold output = 1 $0 == t$ (n.w.u).

$(1,0) = (1)(1.1) + (0)(1.1) = 1.1$

$Y >$ threshold output = 1 $0 == t$ (n.w.u).

$(1, 1) = (1)(1.1) + (1)(1.1) = 2.2$.

$y >$ threshold output = 1 $\quad o == t$ (n.w.u



XOR PRFCEPTRON TRAINING RULE:

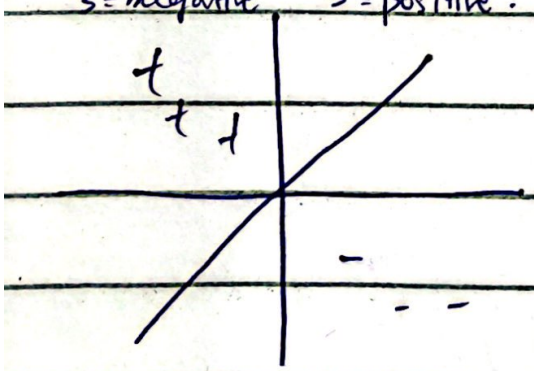
$Y = x_1 \bar{x}_2 + \bar{x}_1 + x_2$.

x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

GRADIENT DESENT & DELTA RULE.

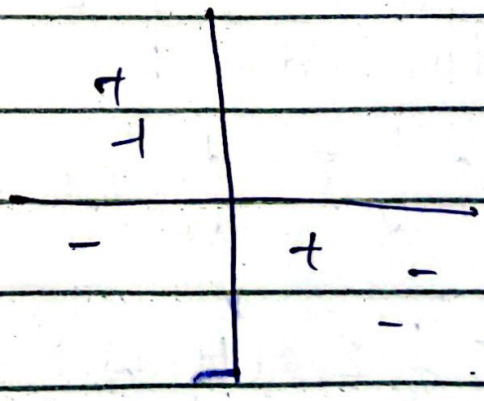
linearly seperable

total = 6 data points.
3 = negative 3 = positive.



given data is seperable with a straight line

non linearly seperable.



or no straight line to seperate data.

→ Perceptron Rule search successful weight if (linearly seperable).

→ Delta Rule → If not linearly seperable.
↳ use gradient descent.
↳ basis of back propagation.

$$\Delta w = -\eta \nabla E(\vec{w})$$

η = Learning rate (determines step size in gradient decent)

-ve sign → shows we want to move the weight vector in Direction that "decreases" E

Component form: A.k.A

$$w_i \rightarrow w_i + \Delta w_i$$

$$\therefore \Delta w_i = -\eta \frac{\partial E}{\partial w_i} \qquad \Delta E(\vec{w}) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2} \right]$$

for all training eg

$$\text{value of } E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$\frac{dE}{dw_i} = \frac{d}{dw_i} \frac{1}{2} \sum (t_d - o_d)^2$$

$$= \frac{1}{2} \sum \frac{d}{dw_i} (t_d - o_d)^2$$

$$= \frac{1}{2} \sum 2(t_d - o_d) \frac{d}{dw_i} (t_d - o_d)$$

constant (as it is)

$$= \sum (t_d - o_d) \frac{d}{dw_i} (t_d - \vec{w} \cdot \vec{x}_d)$$

derivation is 0 in all cases except when vector w become w_i, the derivation would be 1 in that case.

$$\because \; o(\vec{x}) = \vec{w} \cdot \vec{x}$$

$$\frac{dE}{dw_i} = \sum_{d \in D} (t_d - o_d)(-x_{id})$$

$$\Delta w_i = -\eta \frac{dE}{dw_i}$$

$$= +\eta \sum (t_d - o_d)(x_{id})$$

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id}$$

$$w_i \leftarrow w_i + \Delta w_i$$

GRADIENT DESCENT ALGO.

→ used to seperate the non linearly seperable data

→ weights are updated by:

$$w_i \leftarrow w_i + \Delta w_i$$

where, $\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id}$.

Backpropagation In ANN

→ Neural network based technique

disadvantage of neural network

→ it takes long time for training

adv:

→ tolerate noisy data

→ good for unrecognize (untrained) patterns

* Backpropagation is an iterative algo.

First layer → Hidden layer → Output layer

dataset, target attribute, weight, bias → already known.

Two layer → (F) → [(H) → (O)]

do not count

Feed Forward → (F) → (H) → (H) → O

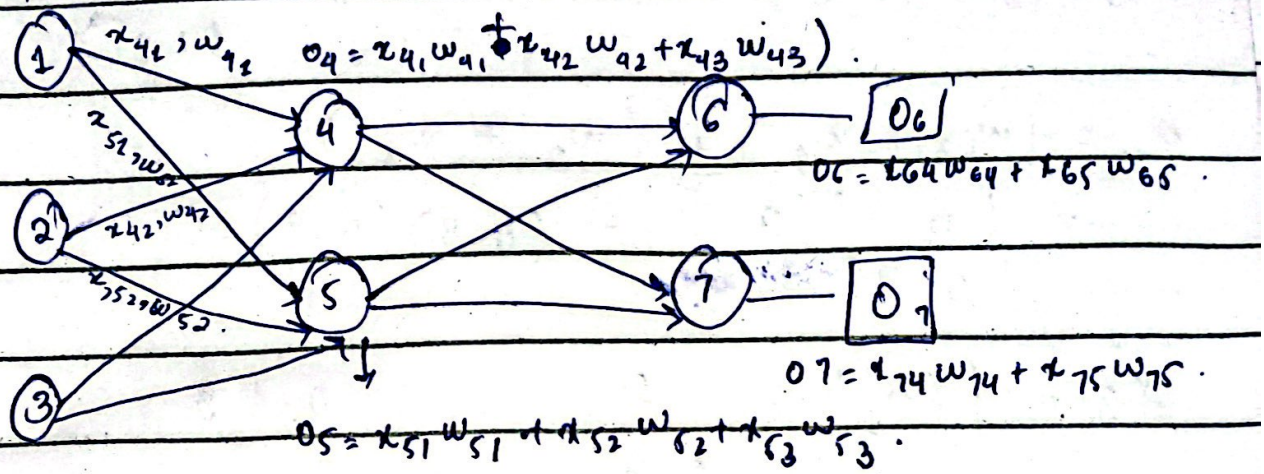
cross inputs

multiple hidden layers.

Fully - connected → straight forward.

(F) → (H) → (H) → (O)

→ compute output O_n of every unit.



→ each network k, calculate error. δ_k ↩ for output layer. δ_6, δ_7.

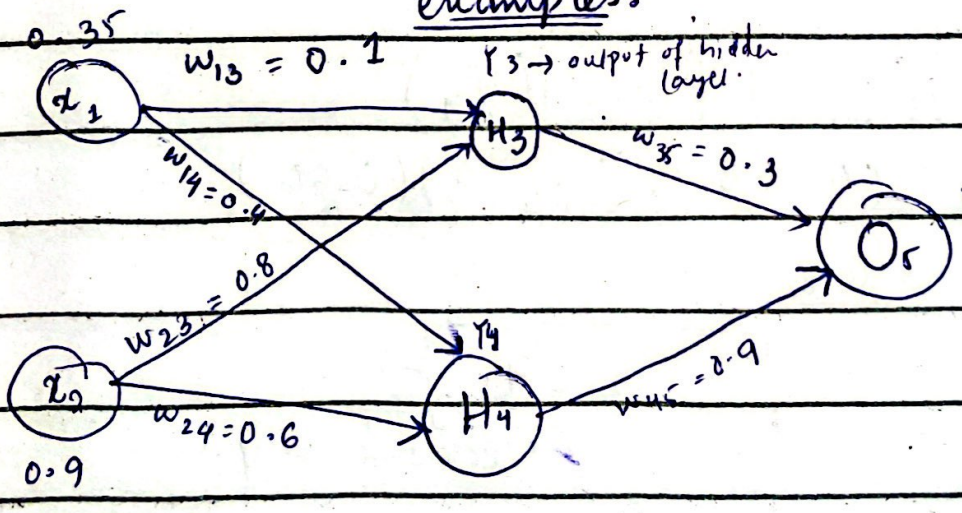
$$\delta_k \leftarrow O_k(1 - O_k)(t_k - O_k)$$

O_k: output, t_k: target

→ each network h, calculate its error term δ_h ↩ for hidden layer δ_4, δ_5.

$$\delta_h \leftarrow O_h(1 - O_h) \sum_{k \in \text{output}_s} w_{h,k}\,\delta_k$$

examples:



Forward pass = Compute output for Y_3, Y_4 & Y_5.

$a_j = \sum_j (w_{ij} * x_i)$ $\quad Y_j = F(a_j) = \frac{1}{1+e^{-a_j}}$

$a_1 = (w_{13} * x_1) + (w_{23} * x_2)$

$= (0.1)(0.35) + (0.8)(0.9)$

$= 0.75$

$Y_3 = f(a_1) = \frac{1}{1+e^{-0.75}} = 0.68$

$a_2 = (w_{14})(x_1) + (w_{24})(x_2)$

$= (0.4)(0.35) + (0.6)(0.9)$

$= 0.68$

$Y_4 = \frac{1}{1+e^{-0.68}} = 0.66$

$a_3 = (w_{35})(y_3) + (w_{45})(y_4)$

$= (0.3)(0.68) + ((0.9)(0.66)$

$= 0.801$

$Y_5 = \frac{1}{1+e^{-0.801}} = 0.69 \rightarrow$ Network output

error $= Y_{target} - Y_5$

$= 0.5 - 0.69 = -0.19$

learning rate constant

weight updation: $w_{ji} = \eta \delta_j o_j$

if j is output unit:

$\delta_j = o_j (1 - o_j)(t_j - o_j)$

$\delta_j = o_j (1 - o_j) \sum_k \delta_k w_{kj}$ → if j is a hidden unit

Backward pass:

$\delta_5 = Y(1 - Y)(y_{target} - Y)$.

$= (0.69)(1 - 0.69)(0.5 - 0.69)$

$= -0.0406$

for hidden:

$\delta_3 = Y_3 (1 - Y_3) w_{35} * \delta_5$

$= (0.68)(1 - 0.68)(0.3)(-0.046)$

$= -0.0026$.

$\delta_4 = Y_4 (1 - Y_4)(w_{45})(\delta_5)$.

$= (0.66)(1 - 0.66)(0.9)(-0.0406)$

$= -0.0082$

$\Delta w_{ji} = \eta \delta_j o_i$.

$\Delta w_{45} = \eta \delta_5 Y_4 = (1)(-0.0406)(0.66) = -0.02$

now we will add this to the old weight value to find the new weight updation.

$w_{45}(new) = \Delta w_{45} + w_{45}(old) = -0.0269 + 0.9$

$\Delta w_{45}(new) = 0.03911$

$\Delta w_{14} = \eta \delta_4 x_1 = (1)(-0.0082)(0.35$

$= -0.00287.$

$\Delta w_{14}(new) = \Delta w_{14} + w_{14}(old).$

$\Delta w_{14} = -0.00287 + 0.4 = \boxed{0.3971}$

∴ we will find updated weight for all weights

again using forward pass

now find Y_3, Y_4, Y_5 values by using updated weights

$a_1 = (w_{13} * x_1) + (w_{23} * x_2)$

$= (0.0991)(0.35) + (0.79)(0.9)$

$a_1 = 0.7525$

$Y_3 = 1/1 + e^{-0.7525} = 0.6797.$

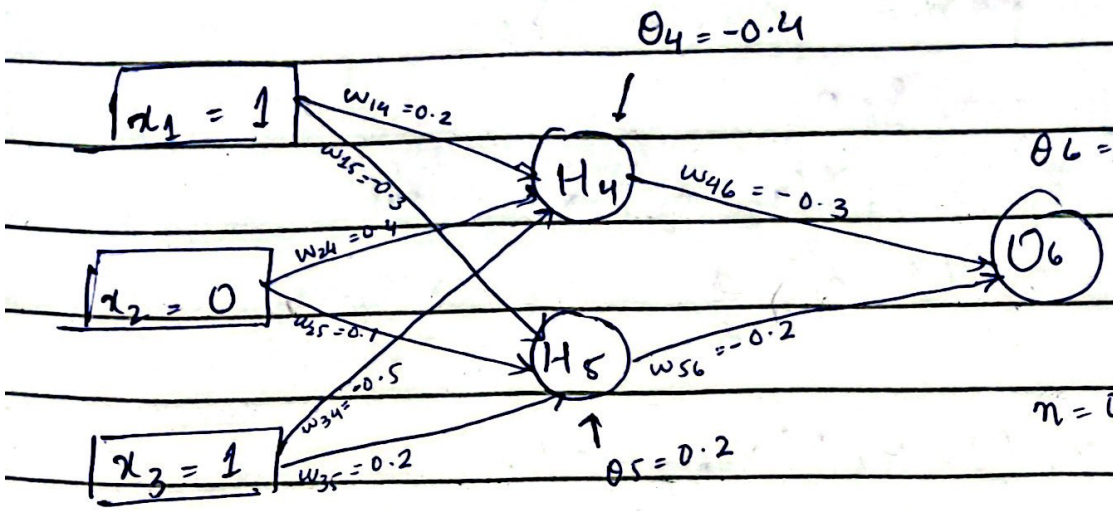
⋮

$Y_5 = 0.68 \rightarrow$ Network output

Error = $Y_{target} - Y_5$

$0.5 - 0.68 = -0.182$

MLP (backpropagation)



Forwaud pass = y_4, y_5, y_6.

$a_j = \sum_j (w_{ij} * x_i)$ $\quad Y_j = F(a_j) = 1/1+e^{-a_j}$.

$a_4 = (w_{14})(x_1) + (w_{24})(x_2) + (x_{34} + x_3) + \theta_4$

$a_4 = -0.7$

$Y_4 = 1/1+e^{0.7} = 0.332$.

$a_5 = (w_{15})(x_1) + (w_{25})(x_2) + (w_{35})(x_3) + \theta_5$.

$a_5 = 0.1$

$Y_5 = 1/1+e^{-0.1} = 0.525$.

$a_6 = (w_{46})(y_4) + (w_{56})(y_5) + \theta_6$.

$a_6 = -0.105$

$Y_6 = 1/1+e^{0.105} = 0.474$

Error = $Y_{target} - Y_6 = 0.526$ → update weight to get rid of this error

Backward pass = $\delta_4, \delta_5, \delta_6$.

$\Delta w_{ji} = \eta \delta_j o_i$

$\delta_j = o_j (1 - o_j)(t_j - o_j) \rightarrow$ output

$\delta_j = o_j (1 - o_j) \sum_k \delta_k w_{kj} \rightarrow$ hidden

$\delta_6 = Y_6 (1 - Y_6)(Y_{target} - Y_6)$

$= (0.474)(1 - 0.474)(1 - 0.474)$

$= 0.1311$

for hidden:

$\delta_5 = Y_5 (1 - Y_5)(\delta_6)(w_{56})$

$= -0.0065$

$\delta_4 = Y_4 (1 - Y_4)(\delta_6)(w_{46})$

$= -0.0087.$

$\Delta w_{ij} = \eta \delta_j o_i$.

$\Delta w_{46} = \eta \delta_6 Y_4 = 0.039$

$\Delta w_{46}(new) = \Delta w_{46} + \Delta w_{46}(old) = -0.261$

:

ind. all.

again use forward pass.

to find Y_4, Y_5, Y_6. with updated weights

$y_4 = 0.32$

$y_5 = 0.52$.

$y_6 = 0.51$ → network output.

Error = $y_{target} - y_6$.

$= 1 - 0.51$ $\boxed{= 0.48}$.