UNIT-II

Artificial Neural N/w - 1

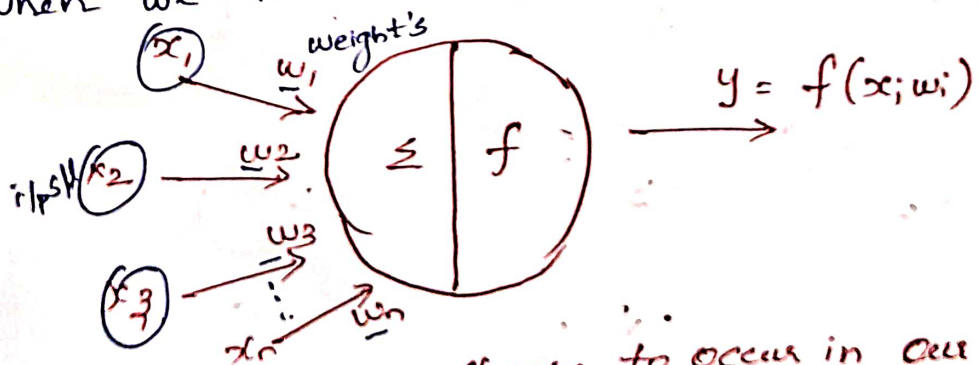
Introduction, Neural N/w Representation

1) Will learn Neural N/w 2) What is Artificial Neuron
3) How to represent that 4) What are the different parts are there

* Human brain - Million of neurons — Related to our Brain
Nephron → Related to Kidney

* Neurons Can't able to see by eye
* fire happen, Humans come back, How its stimulating means Neurons.
* Natural function are done artificially (i.e) called Artificial Neural N/w
  * Biological term is Neurons
  * Artificial / formal term is Node.

Ex: When we touch a hot vessel



* Redness, Bubbles may chances to occur in our Hand, At that moment, Neurons in our hand stimulated. Brain will Collect the info by using Neurons, and o/p will generate (i.e) Take out Hand "It is fraction of Seconds".

* Nodes also same, getting i/p, process, o/p.

* A Nodes has 2 Part ($\overset{①}{\Sigma}$ | $\overset{②}{f}$)
  - ① → Summation
  - ② → activation-fn

Summation:

Calculate the weighted Sum of all the i/p's.

$= x_1 w_1 + x_2 w_2 + x_3 w_3 + \ldots x_n w_n$ ($\Sigma x_i w_i$)

Once weighted Sum is Calculated, sent to activation fn.

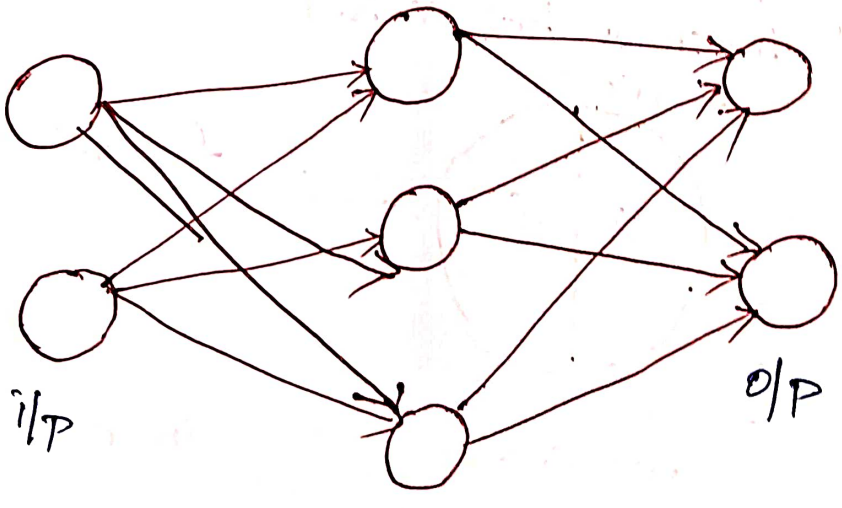
Activation function:

Generate the o/p based on i/p given
↓ $\Sigma x_i w_i$

Representation of ANN:-

ANN are divided in to 3 parts

1) Input layer
2) Hidden layer
3) output layer.



* We can take n number of Nodes each.

* i/p Connected Hidden Connected to o/p
* every Node should Connected to all the Nodes
* I/p layer will receive i/p signals / data.
* Hidden layer have neurons, it is used to extract the info the o/p will generate.
* o/p layer responsible for producing & Presenting the o/p to the user.

— x —

Appropriate Problems for Neural N/w learning:-

" Appropriate Problems in which Situation going to Apply a Neural N/w."

1. Instances have Many Attribute value pair's.

" What is meant by Attribute value pair's.

* Ex:- Age — old, mid, new, we in decision tree learning Cannot apply so many attribute pair's.
(Attribute) (Values)

* But in Neural N/w we Can apply Plenty of attribute pair's.

2. Target function has discrete values, Continous values or Combination of both. Independent to each Other Dependent to each Other.

* discrete values Only used in Decision tree, Whereas Continous also used in Neural Networks.

3. Training Examples with Errors or Missing values.
(We can apply in neural N/W)

4. Long training times are acceptable.
"ANN can also use for long training period of Time".
Due to more attributes calculation takes more Time.

5. Fast Evaluation of the target function learnt.
"learning (Training) some new subject from the Scratch will take time
But Revising (Evaluation) it won't take too much of time".

6. Ability for humans to understand the target function learnt by Machines is not important.
"Machine is learning that is Enough, it is not Mandatory to learn Human's at this case we using ANN".

— x —

Perceptions:-

- Basic Unit used to Build ANN.

What it will do?

- Takes Real value I/p, (Continuous I/p also), Calculates Linear Combination of these I/p's and generates O/p.

O/p = 1 if Result > threshold
-1 Otherwise
} This is how Perceptron works

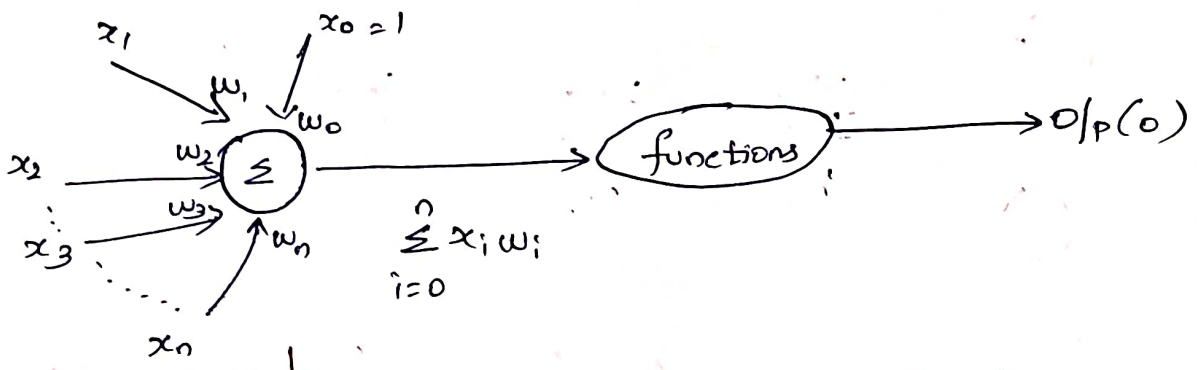
Initial Weight

O/P

$$o(x_1, x_2 \ldots x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n > 0 \\ -1 & \text{Otherwise.} \end{cases}$$

(Weight — linear combination of i/p)

How Perceptron Train Rule



* i/p with weight add up = Result (o/p) → given to fn.

o → actual o/p [function will generate actual o/p]

t → Target o/p. [expecting o/p]

If Actual = target ⇒ weights are fixed

Otherwise ⇒ Changed.

How to change weight?

* Initially have to pick Random weights.

* Later keep on applying iterations and Check if (o = t)

formula to change weight:-

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

$$\Delta w_i = n(t - o)x_i$$

x_i → i/p associated with w_i

$w_0 = x_0$, $w_1 = x_1$, $w_2 = x_2$

n → small constant value like (0.1, 0.2 ...)

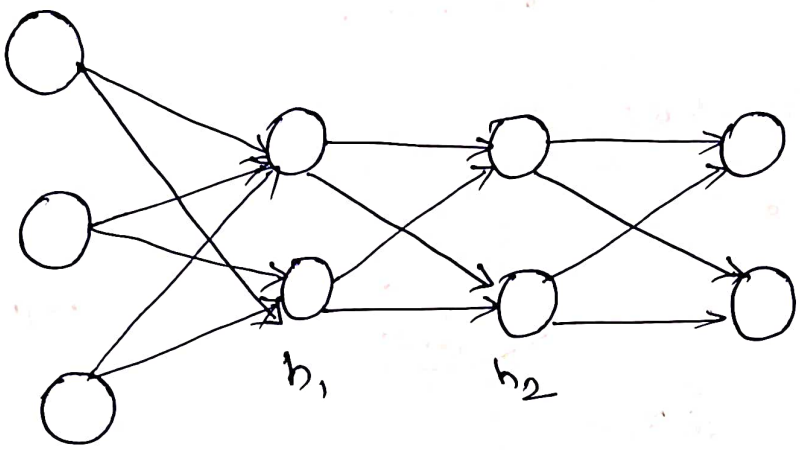
"Until Unless Keep on (o=t) iterations"

This is how perceptron training Rule goes on.

—x—

Multilayer - Neural N/w :-

- has 2 hidden



* A Single layer Only uses for Linearly Seperable data Not for non-linear
*
* In order to Support Non-linear, utilizing multi-layer Neural N/w.
* Working, Connection Same as Single layer. But multilayer Consist 2- hidden layer.
* Back propagation possible here ; (i.e) Going back

1
2
3
:
} o/p → actual o/p Obtain against Target

If (o = t) No problem

if (o ≠ t) will go back and change (i.e) (weights).

BACK PROPAGATION ALGORITHM:

* As we know we can go back whenever the error occurs in multilayer.

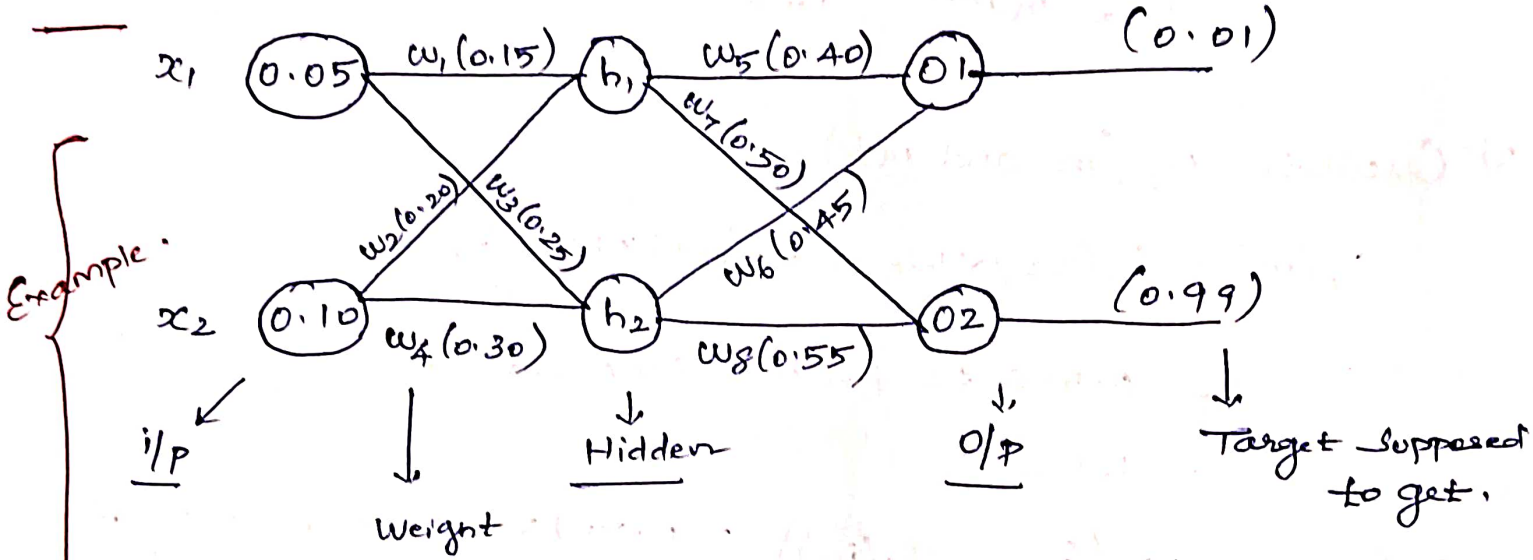
* When error occurs, we go in backward direction

(i.e) O/p → Hidden → i/p layer.

Part-1 : Calculate forward Propagation Error

i) Calculate h_1 (in and out).

Example.



i/p x_2 ← [b_1 = (0.35)] → Hidden h_2 ← [b_2 = (0.60)] → O/p. O2

b_1, b_2 are Bias factor's.

i) Calculate h_1 (in and out)

h_1(in) = Coming towards h_1

(i.e) $h_1(in) = w_1x_1 + w_2x_2 + b_1$

$= 0.15 \times 0.05 + 0.20 \times 0.10 + 0.35$

$= 0.377$

↳ formula.

$$h_1(out) = 1/\left(1+e^{-h_1(in)}\right) = \frac{1}{1+e^{-(0.377)}} = 0.5932$$

2) Calculate h_2 (in and out)

$$h_2(in) = x_1 w_3 + x_2 w_4 + b_1$$

$$= 0.05 \times 0.25 + 0.10 \times 0.3 + 0.35$$

$$= 0.0125 + 0.03 + 0.35 = 0.3925$$

$$h_2(out) = \frac{1}{\left(1+e^{-h_2(in)}\right)} = \frac{1}{1+e^{-0.3925}} = 0.5968$$

3) Calculate O_1 (in and out)

$$O_1(in) = h_1(out) \times w_5 + h_2(out) \times w_6 + b_2$$

$$= 0.593 \times 0.4 + 0.596 \times 0.45 + 0.6$$

$$= 1.105$$

$$O_1(out) = 1/\left(1+e^{-O_1(in)}\right) = 0.7513$$ (Slightly differ with Target value)

4) Calculate O_2 (in and out)

$$O_2(in) = h_1(out) \times w_7 + h_2(out) \times w_8 + b_2$$

$$= 0.5932 \times 0.50 + 0.5968 \times 0.55 + 0.6$$

$$= 1.22484$$

$$O_2(out) = 1/\left(1+e^{-O_2(in)}\right) = 0.7729$$ (Differ with Target Value)

* O_1 and O_2 o/p Not matching with target value, so that error occurs.

5) Calculate E_{Total} ; To calculate error.

$$E_{Total} = \sum \frac{1}{2}(target - o/p)^2$$

Remove Summation, mention error E

$$= E_{O1} + E_{O2}$$

↓ Error

$$= \frac{1}{2}(0.01 - 0.7513)^2 + \frac{1}{2}(0.99 - 0.7729)^2$$

$$= \frac{1}{2}(-0.7413)^2 + \frac{1}{2}(0.2171)^2$$

$$= 0.274 + 0.0235 = 0.29837 \text{ (Approx)}.$$

⇓

⇒ This is the amount of Error we got By using given weight's.

⇒ When the weight's Change we Can Reduce the Error Compared to Current error (0.29837) slight Change in Reduction of error.

Calculate Backward Propagation Error:-

(output Layer → Hidden Layer).

* In between o/p layer to Hidden Layer, what are the weights assigned such as,

w_5, w_6, w_7 and w_8 ← Only have to Change the identified weight's Value of

first let us adjust the weight's w_5

$$w_5^* = w_5 - \eta \frac{\partial E_{total}}{\partial w_5}$$

w_5^* ↑ adjusted weight

$\frac{\partial E_{total}}{\partial w_5}$ ↑ Partial differentiation.

η ↓ learning rate (Our wish fix value 0.6).

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5} \quad \text{[Chain Rule]}.$$

* $\frac{\partial E_{total}}{\partial out_{o1}} = Out_{o1} - target_{o1} = 0.751365 - 0.01$

$= 0.74135565$

* $\frac{\partial out_{o1}}{\partial net_{o1}} = Out_{o1}(1 - Out_{o1})$

$= 0.751365(1 - 0.751365) = 0.186815602$

* $\frac{\partial net_{o1}}{\partial w_5} = Out_{h_1} = 0.593269992$ (i.e) $h_1(out)$

(weight w_5 coming out of h_1)

$$\frac{\partial E_{total}}{\partial w_5} = 0.74135565 \times 0.186815602 \times 0.593269992$$

$$= 0.082167041$$

W.K.T $w_5^* = w_5 - \eta \frac{\partial E_{Total}}{\partial w_5}$

$= 0.4 - 0.6 \times 0.082167041$

$\boxed{w_5^* = 0.350699776.}$ New w_5 value

∴ Like this have to Calculate for w_6, w_7 & w_8.

Calculating Backward propagation of Error:-

[Hidden → i/p layer].

weights are (w_1, w_2, w_3, w_4).

first lets adjust w_1

$$w_1^* = w_1 - \eta \frac{\partial E_{Total}}{\partial w_1}$$

$$\frac{\partial E_{Total}}{\partial w_1} = \overset{A}{\frac{\partial E_{Total}}{\partial out_{h_1}}} \times \overset{B}{\frac{\partial out(h_1)}{\partial net_{h_1}}} \times \overset{C}{\frac{\partial net_{h_1}}{\partial w_1}}$$

$$\overset{A}{\frac{\partial E_{Total}}{\partial out_{h_1}}} = \frac{\partial E_{O1}}{\partial out_{h_1}} + \frac{\partial E_{O2}}{\partial out_{h_1}}$$

1st term:

$$\overset{1st}{\frac{\partial E_{O1}}{\partial net_{O1}}} \times \overset{2nd}{\frac{\partial net_{O1}}{\partial out_{h_1}}}$$

$$\Downarrow$$

$$\frac{\partial E_{O1}}{\partial out_{O1}} \times \frac{\partial out_{O1}}{\partial net_{O1}}$$

2nd term:

$$\overset{1st}{\frac{\partial E_{O2}}{\partial net_{O2}}} \times \overset{2nd.}{\frac{\partial net_{O2}}{\partial out_{h_1}}}$$

$$\Downarrow$$

$$\frac{\partial E_{O2}}{\partial out_{O2}} \times \frac{\partial out_{O2}}{\partial net_{O2}}$$

2nd term

$$\frac{\partial E_{O2}}{\partial out_{O2}} = (out_{O2} - \underbrace{target\ O_2}_{\text{given in Question}})$$

$$= 0.772928465 - 0.99$$

$$= -0.217071535$$

$$\frac{\partial out_{O2}}{\partial net_{O2}} = Out_{O2}(1 - Out_{O2})$$

$$= 0.7729(1 - 0.7729)$$

$$= 0.175510052$$

2nd term 1st

$$\frac{\partial E_{O2}}{\partial net_{O2}} = -0.217071535 \times 0.175510052$$

$$= -0.038098$$

Note ⊗

Like wise for 1st term also instead 2 Replace 1.

for $\frac{\partial E_{O1}}{\partial net_{O1}}$ = Ans.

0.138498 56.

1st term :- 2nd one.

$$\frac{\partial net_{o1}}{\partial out_{h_1}} = \text{on } o_1 \text{ from } h_1 \Rightarrow w_5$$

$$= 0.4$$

2nd term :- 2nd one

$$\frac{\partial net_{o_2}}{\partial out_{h_1}} = \text{on } o_2 \text{ from } h_1 \Rightarrow w_7$$

$$= 0.50$$

W.K.T formula

$$\frac{\partial E_{Total}}{\partial out_{h_1}} = \frac{\partial E_{o1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial out_{h_1}}$$

We calculated the values i.e

$$\frac{\partial E_{Total}}{\partial out_{h_1}} = 0.138498566 \times 0.4 + (-0.0380982 \times 0.50)$$

$$= 0.055399425 + (-0.019049119)$$

$$\boxed{\frac{\partial E_{Total}}{\partial out_{h_1}} = 0.036350306.} \rightarrow (A).$$

Next to find B :-

$$\frac{\partial out_{h_1}}{\partial net_{h_1}} = out_{h_1}(1 - out_{h_1})$$

(Known value.)

$$= 0.241300709.$$

To find C :-

$$\frac{\partial net_{h_1}}{\partial w_1} = \frac{\partial}{\partial w_1}(w_1 x_1 + w_2 x_2 + b_1)$$

(Differentiation)

differentiate with w_1

$w_2 x_2$: Constant, Differentiate becomes '0'.

b_1 : Constant

$$\frac{\partial net_{h1}}{\partial w_1} = x_1 = 0.05$$

x_1 value = 0.05 already known.

$$= 0.05$$

W.K.T formula (main formula).

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 \times 0.241300709 \times 0.05$$

$$= 0.00438568$$

$$w_1^* = w_1 - \eta \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.6 \times 0.00438568$$

$$w_1^* = 0.149736859 2.$$

New weight

Note:- Like wise for w_2, w_3, w_4.

//Set of updated weights, Again doing as a Part-1, until Unless do it up to the target value.//

— x — x — x — x — .

Artificial Neural Networks - 2:-

Remarks on the Back propagation Algorithm:-

1) Convergence and local Minima

* W.K.T, * Back propagation is multi layer Neural, N/W. We can go back and can change weight's.

Convergence:-

* All the neurons InterConnected to Each other

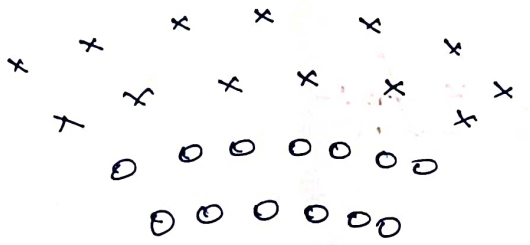
Local Minima:-

* Minimize the error only locally Not globally in Back propagation

* Radial Basis functions:-

* It is used in ANN
* It has only one hidden Nodes.

Ex:-



→ 2 classes (Star & Circle)

(iv) Not possible to Seperate this data using single line.

→ Data is not linearly seperable data.

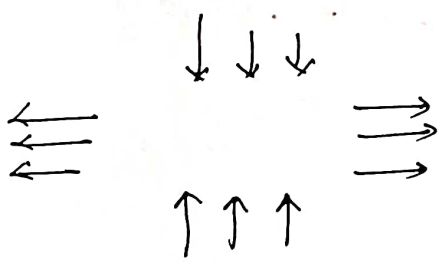
By using Radial we going to Convert linearly Seperable function. Then will eligible to seperate Star & Circle.

→ 2 steps.

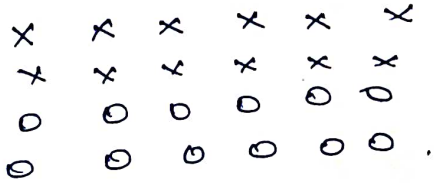
1) Increase the dimensionality (2D-3D)
(This Step is Not mandatory. Only Based on Requirement)

2) Expand the direction (Horizontal)

Compress the direction (vertical)



The o/p becomes:- Resultant Data Set is.



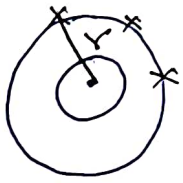
Non-linear to linear Convert.
"Data will come Closer and perfect to seperate"

→ We should follow the Rules to Compress & Express.

How RBF Does this?

* One data point as a Centre. (Consider one Centre Randomly).

* Draw Concentric Circles.



Diff Radius, Same Centre.

To Expand / Compress, We use 3 fn:-

1) Multi-Quadric:-

$$\phi(r) = (r^2 + c^2)^{1/2}$$

$c > 0 \Rightarrow$ Constant

2) Inverse multiquadric:-

$$\phi(r) = \frac{1}{(r^2 + c^2)^{1/2}}$$

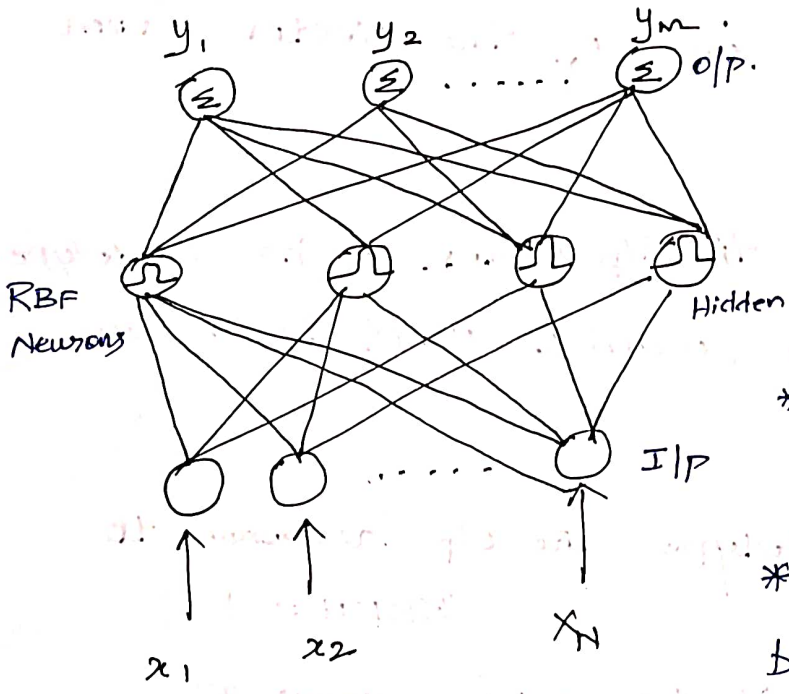
3) Gaussian fn:-

$$\phi(r) = \exp\left[\frac{-r^2}{2\sigma^2}\right]$$

$\sigma \Rightarrow$ Constant $(\sigma > 0)$.

— x —

Radial Basis fn Network :-



* Most important type of Neural N/w
* It is a special type of feed forward N/w.
* It is a variant of 3-layer feed forward N/w.
* RBFN perform Classification by measuring I/p Similarities

* Each RBFN Neurons is start up prototype which is one of the Ex from Training Set.
* When we want to classify the New Neuron (i/p) arrived. Each Neuron compute the Euclidean Distance, in the i/p and prototype.
* Then if the i/p more close to class A, then the Class B prototyp it is classified as class A.
* I/p vector arrived at $x_1, x_2 \ldots x_N$. and it consist of i/p layer, Hidden, O/p layer.
* I/p vector Basically. N- Dimension vector.

⇐ Entire i/p vector, shown to Each of the RBF Neurons.

* RBFN Neurons is hidden layer. Each RBFN stores the Prototype vector. Which is one of the vector from Training Set. $(x_1, x_2 \cdots x_N)$.

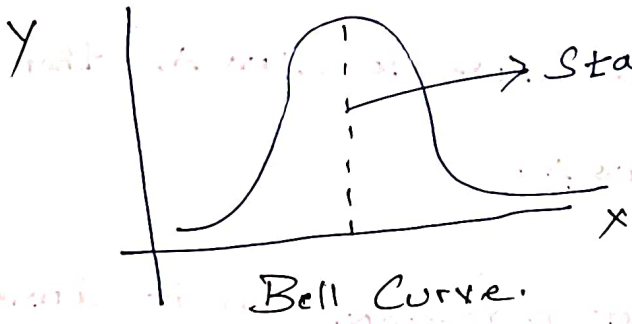
* Each RBF Neurons Compare the i/p vector to its Prototype and o/p of value between (0 and 1) which is a basic of Similarity, that means

if the i/p = Prototype the o/p Response to RBFN = 1

If the distance between i/p & prototype grow, Response fall RBFN = 0.

So that Shape of RBFN is in Bell Curve.

* Bell Curve is Normal distribution fn. is used to describe the graphical representation of Prob distribution



Standard deviation from that 'mean' Created Bell curve.

Bell Curve.

* Neuron Response value is called Activation value $(x_1, x_2 \cdots x_N)$

* The prototype vector also often called the Neurons Centred, Since the Value is Centre of Bell Curve

* O/p - layer - also known as Transfer fn.
  O/p Consist of Set of Nodes.
* Gives a Score for the Associated Category. like
  Category A, B, C.
* Classification made by Assign i/p to the Category with
  the highest Score.
* Score is Computed by taking Weighted Sum.
* That every An O/p Node Associated a Weight Value with each
  of the RBFN and Multiplied the Neurons Activation with
  the Rate. Before adding the Total Response.

Used for Regression

1) Logistic
2) Linear.
3) Different Method of Classification
4) Pattern Recognition.

—x—

Curse of Dimensionality:-

Why ML need large set of data? That Concept is more helpful to understand Curse of Dimensionality.

Training Data Set (House Prediction)

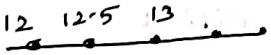
	1 House Size SFt	2 House location	3 No of bed rooms	4 No of bath	5 Cost of House
					13L
1	1200				12.5L
2					
10	1100				12L
10 more					
10	1150				

* How Can We Show this data set in Co-ordinate System.

* 5-Columns availabe So we need 5 Dimension Co-ordinate System.

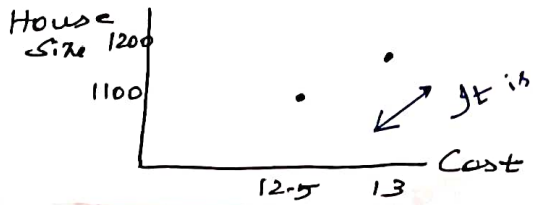
* If we need to Represent only "Cost of House", we required 1D Co-ordinate System.

How it looks?



"What we noted Here means, The data is very dense and Close to Each other".

* The Same Situation Not happend in 2dimension.
(If we take only Cost, House Size)



* What we understood here,?

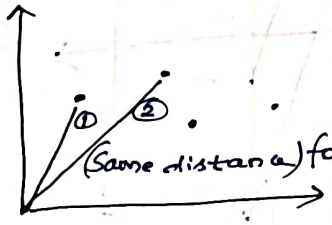
When dimension increases data become Sparse, So hard to generalize the data.

* Ex: In 2D, we want to need more Number of data points. in order to make Dense, ↑ (100), In 3D - (1000) points.

Disadvantage :-

* "So when dimension ↑, we need a large Number of data points" KNN.

* Every data points is equal distant from all other points.



(Same distance) for ① & ②, for all data points, Mathematically Proven.

* So that ML failed in High Dimension Co-ordinate System. (Not only KNN is related to ML).

—x—

Support vector Machine (SVM):-

Ranji Raj

Applications :

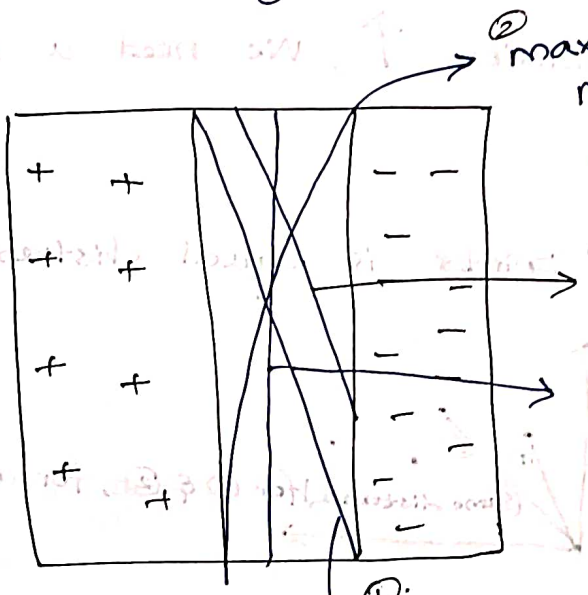
1) Handwritten Character
2) Digit Recognition
3) Text Categorization

* Also Svm performs very well in high dimensional data → it solves high dimensionality. (This is the Problem in many classifiers).

* (SVM)

Support Vector → * Subset of training data.
↓ * Used to Represent Decision Boundary.

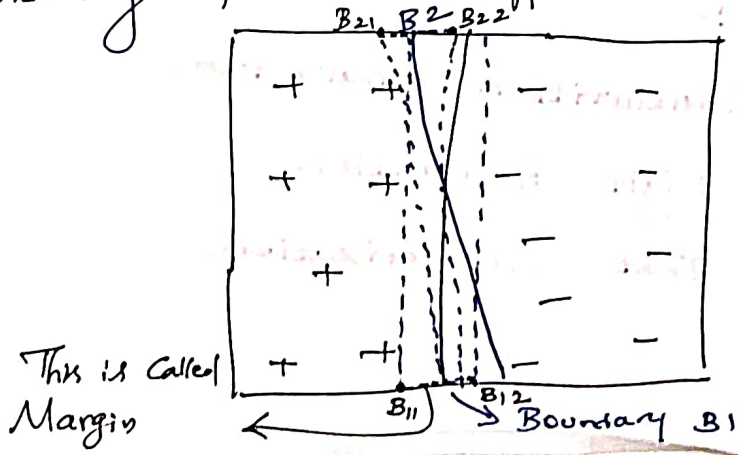
Training Examples
↑
[We can construct Machine / classifier]



* So choosing Hyperlane is more important.

⑤ (It is not equal perform well on any unseen example. When we Construct linear Svm.)

* How we choosing Hyperlane and affects linear SVM?



Ex

B_1 > B_2
margin margin

So that maximal Margin hyperplane Came to picture.

* When margin ↑ - Better Control over generalization Error

  Margin ↓ - Overfitting.

  (This problem discussed in ID_3)

* Similar Case happend to SVM as well.

* Whatever the Concept learned so far, In Statistics it is called SRM - Structural Risk Minimization.

* for Any Classifier

$$R \leq R_e + \varphi\left(\frac{h}{N}, \frac{\log(\eta)}{N}\right) \rightarrow \text{Prob.}$$

φ: monotone increasing fn. of capacity of Hyperplane; h: capacity; N: Total Number of Training Ex; R: Classifier; R_e: Training Error; $\leq$: inequality

* There is a Small trade off between Margin & Capacity.

$$h \propto \frac{1}{\text{margin}}.$$

1) If we need ↑ Higher capacity, margin ↓
2) ↓ Low " " ↑

* If we choose (1), margin ↓ So Overfitting, Poor generalization Error, model Becomes failure. To Overcome choose (2). Compromise in Capacity. In that Case we can go Other Classifiers. Specifically, Maximal Margin classifier its called as Linear SVM.

—x—.