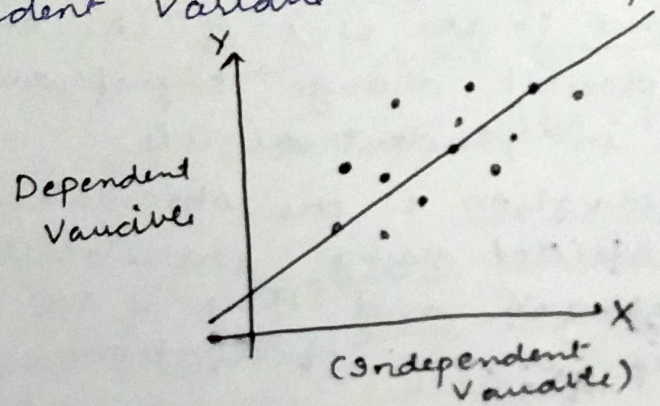
Unit -2.

Regression Analysis

It is a form of predictive modelling technique which investigate the relationship between a dependent (target) and independent variable (s) (predictor).

It is a powerful statistical method that allows you to examine the relationship between two or more variable of interest.

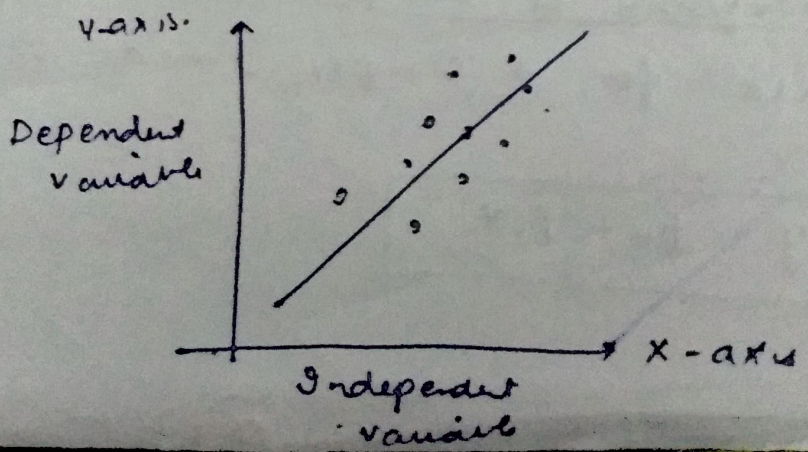
They exame the influence of one or more indep-endent variable on a dependent variable

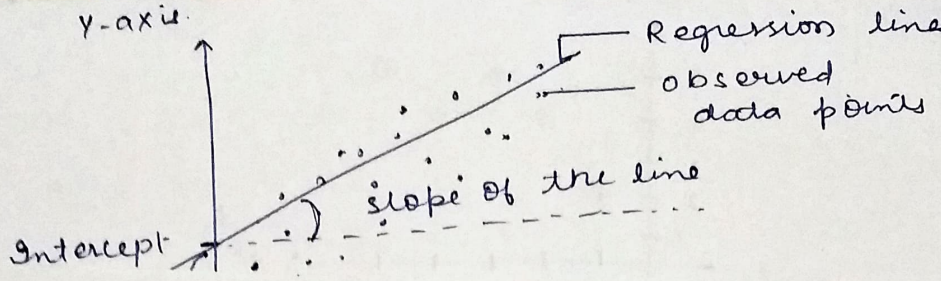


Types of Regression

1. Linear Regression

Linear Regression is a linear model, e.g. a model that assumes a linear relationship between dependent and independent variable. More specifically, that y (dependent variable) can be calculated from a linear combination of x (independent variable).





Simple linear Regression is used to estimate the relationship between two quantitative variables

You can use simple LR when you want to know:-

- How strong the relationship is between two variables.
- The value of dependent variable at a certain value of independent variable.

Assumptions of simple linear regression

Simple LR is a parametric test, meaning that it makes certain assumptions about the data. The assumption are :-

- Homogeneity of variance :- the size of the error in our prediction doesn't change significantly across the value of independent variable.
- Independence of observation :- the observations in the dataset were collected using statistically valid sampling methods, and there is no hidden relationship among observations
- Normality :- The data follows a normal distribution.

LR makes one additional assumption

- The relationship between the independent and dependent variable is linear: the line of best fit through the data point is a straight line (rather than a curve or some sort of grouping factor).

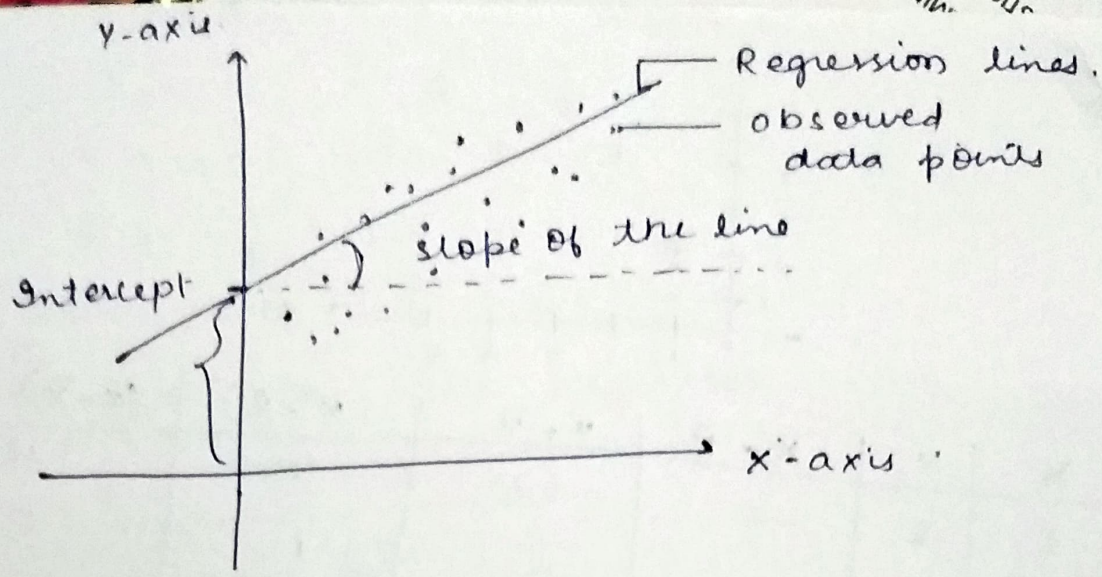
How to perform a simple linear Regression

The formula for a simple linear regression is :-

$$y = \beta_0 + \beta_1 X$$

,

Where.



- y is the predicted value of dependent variable (Y) for any given value of independent variable (x)
- β_0 is the intercept, the predicted value of y when x is 0.
- β_1 is the regression coefficient - (how much we expect y to change as x increase) or the slope of the linear relationship line.
- x is the independent variable (the variable we expect is influencing y)

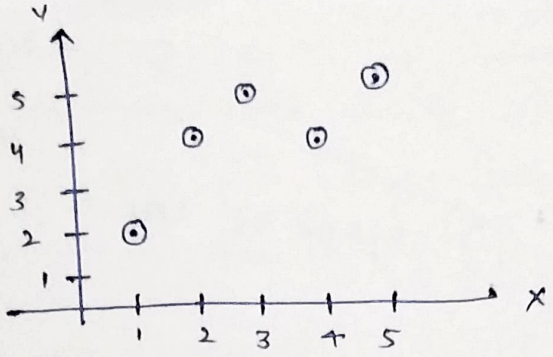
Example of linear Regression

let us find out the linear regression for the given example:-

X	1	2	3	4	5
y	2	4	5	4	5

Here x is the independent variable and y is the dependent variable

Let find the relation between x and y using least-square to fit a line to the data.



X	Y	$X - \bar{X}$	$Y - \bar{Y}$	$(X - \bar{X})^2$	$(X - \bar{X})(Y - \bar{Y})$
1	2	-2	-2	4	4
2	4	-1	0	1	0
3	5	0	1	0	0
4	4	1	0	1	0
5	5	2	1	4	2
$\bar{X} = 3$	$\bar{Y} = 5$			$\Sigma = 10$	$\Sigma = 6$.

$$\bar{X} = \frac{X_1 + X_2 + X_3 + X_4 + X_5}{5} = \frac{1+2+3+4+5}{5} = 3$$

$$\bar{Y} = \frac{Y_1 + Y_2 + Y_3 + Y_4 + Y_5}{5} = \frac{2+4+5+4+5}{5} = 4$$

$$\beta_1 = \frac{\Sigma (x - \bar{x})(y - \bar{y})}{\Sigma (x - \bar{x})^2} = \frac{6}{10} = 0.6.$$

The slope of the regression line i.e $\beta_1 = 0.6$
Now we need to find the y-intercept i.e. β_0

⇒ The mean points i.e. $(\bar{x}, \bar{y}) \Rightarrow (3, 4)$.
All the regression line must have to go through these point (3, 4)

So the eq. $\hat{y} = b_0 + b_1 x$ is true
for $X = 3$ and $Y = 4$.

So, $\hat{y} = b_0 + b_1 x$.

$4 = b_0 + (0.6)(3)$

$4 = b_0 + 1.8$

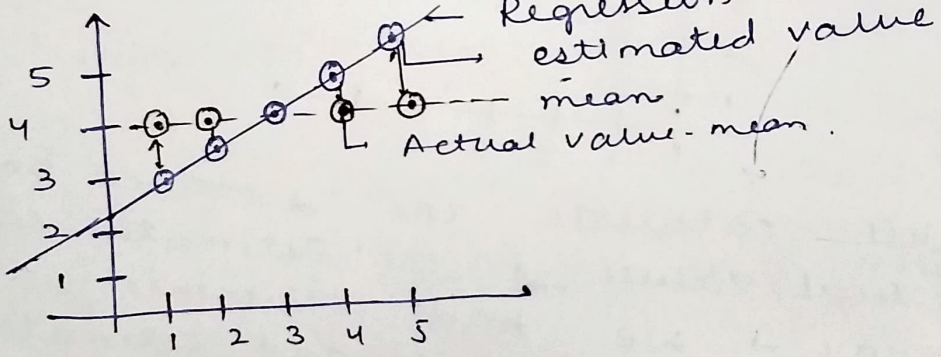
$b_0 = 4 - 1.8 = 2.2$

The equation for the regression line is.

$$\hat{y} = 2.2 + 0.6x$$

Now Calculate R^2.

R^2 determine X and Y are correlated or not. Large value imply a large effect. In this we compare actual values to the mean with the estimated values to the mean.



X	Y	$y - \bar{y}$	$(y-\bar{y})^2$	$\hat{y}$	$\hat{y} - \bar{y}$	$(\hat{y} - \bar{y})^2$
1	2	-2	4	2.8	-1.2	1.44
2	4	0	0	3.4	-0.6	0.36
3	5	1	1	4	0	0
4	4	0	0	4.6	0.6	0.36
5	5	1	1	5.2	1.2	1.44
$\bar{x} = 3$	$\bar{y} = 4$		6		$\Sigma = 0$ (Always 0)	$\Sigma = 3.6$

$$R^2 = \frac{\Sigma (\hat{y} - \bar{y})^2}{\Sigma (y - \bar{y})^2} = \frac{\text{estimated value - mean}}{\text{Actual value - mean.}}$$

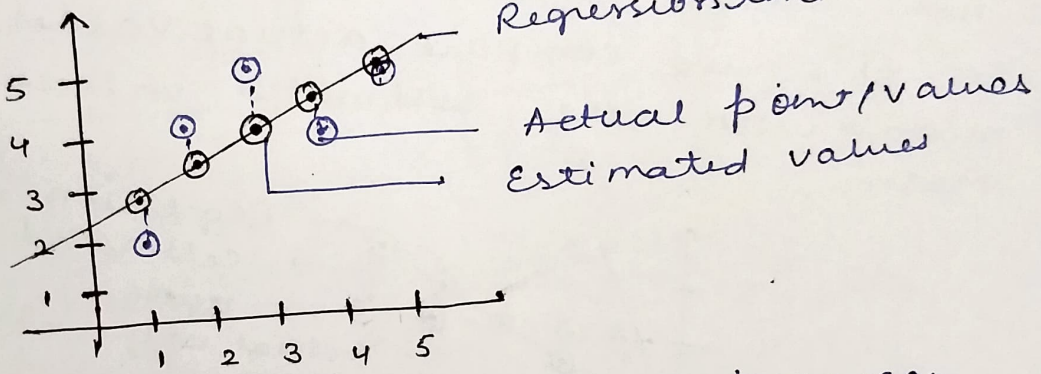
$$R^2 = \frac{3.6}{6} = 0.6.$$

R^2 lies b/w 0 to 1.

- $R^2 = 0$ (no relationship b/w X and Y).
- $R^2 = 1$ (perfect fit)
- $R^2 = 0.9$ (best case).

Another way to measure goodness of fit is Standard error of the estimate.

Standard error of the estimate used in Regression Analysis (p-value)



We will calculate the distance between the actual values & the estimated values. i.e. error → we have to minimize the error between actual and estimated values.

X	y	$\hat{y}$	$\hat{y} - y$	$(\hat{y} - y)^2$
1	2	2.8	0.8	0.64
2	4	3.4	-0.6	0.36
3	5	4	-1	1
4	4	4.6	0.6	0.36
5	5	5.2	0.2	0.04
				$\Sigma = 2.4$

$$\text{Standard error of the estimate} = \sqrt{\frac{\Sigma (\hat{y} - y)^2}{n-2}}$$

where n = number of observations

$$= \sqrt{\frac{2.4}{n-2}} = \sqrt{\frac{2.4}{5-2}}$$

$$= \sqrt{\frac{2.4}{3}} = \sqrt{0.8} = 0.89$$

Standard error of estimate = 0.89. = p-value.

P value determines if R^2 value is statistically significant or not.

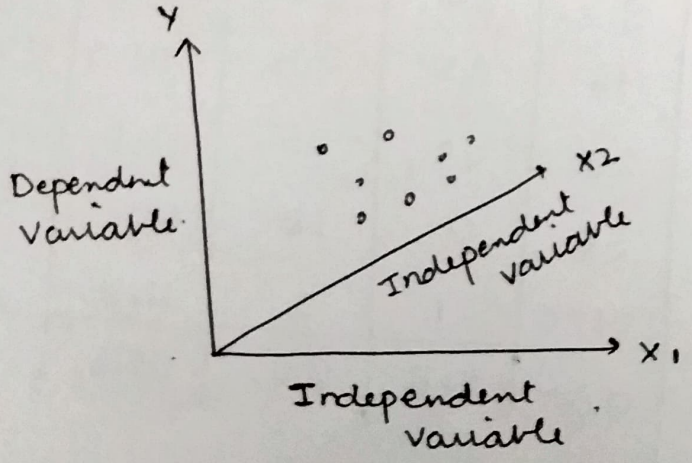
2. Multiple Regression

It is a model to find the relationship between one dependent variable with multiple independent variables.

Multiple regression equation can be written as :-

$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + \ldots + b_nx_n.$$

where n = number of independent variable.



As the number of independent variable increases to 2 our graph become 3D. The added third dimension represent other independent variable.

Let us take an example.

lets find out the regression line for the following data

Y	-3.7	3.5	2.5	11.5	5.7
X_1	3	4	5	6	2
X_2	8	5	7	3	1

The regression line equation will be :-

$$\hat{y} = b_0 + b_1 X_1 + b_2 X_2 \text{ , where}$$

$$b_1 = \frac{(\Sigma x_2^2)(\Sigma x_1 y) - (\Sigma x_1 x_2)(\Sigma x_2 y)}{(\Sigma x_1^2)(\Sigma x_2^2) - (\Sigma x_1 x_2)^2}$$

$$b_2 = \frac{(\Sigma x_1^2)(\Sigma x_2 y) - (\Sigma x_1 x_2)(\Sigma x_1 y)}{(\Sigma x_1^2)(\Sigma x_2^2) - (\Sigma x_1 x_2)^2}$$

Y	X_1	X_2	X_1X_1	X_2X_2	X_1X_2	X_1Y	X_2Y
-3.7	3	8	9	64	24	-11.1	-29.6
3.5	4	5	16	25	20	14	17.5
2.5	5	7	25	49	35	12.5	17.5
11.5	6	3	36	9	18	69	34.5
5.7	2	1	4	1	2	11.4	5.7
$\Sigma =$ 19.5	20	24	90	148	99	95.8	45.6

Now,

$$\Sigma x_1^2 = \Sigma X_1 X_1 - \frac{(\Sigma X_1)(\Sigma X_1)}{N}$$

$$= 90 - \frac{(20)(20)}{5} = 10.$$

$$\Sigma x_2^2 = \Sigma X_2 X_2 - \frac{(\Sigma X_2)(\Sigma X_2)}{N}$$

$$= 148 - \frac{(24)(24)}{5} = 32.8$$

$$\Sigma x_1 y = \Sigma X_1 Y - \frac{(\Sigma X_1)(\Sigma Y)}{N} = (95.8) - \frac{(20)(19.5)}{5}$$

$$= 17.8$$

$$\Sigma x_2 y = \Sigma X_2 Y - \frac{(\Sigma X_2)(\Sigma Y)}{N} = (45.6) - \frac{(24)(19.5)}{5.}$$

$$= -48$$

$$\Sigma x_1 x_2 = \Sigma X_1 X_2 - \frac{(\Sigma X_1)(\Sigma X_2)}{N}$$

$$= 99 - \frac{(20)(24)}{5} = 3$$

putting these values in previous equations of b_1 and b_2

$$b_1 = \frac{(\Sigma x_2^2)(\Sigma x_1 y) - (\Sigma x_1 x_2)(\Sigma x_2 y)}{(\Sigma x_1^2)(\Sigma x_2^2) - (\Sigma x_1 x_2)^2}$$

$$= \frac{(32.8)(17.8) - (3)(-48)}{(10)(32.8) - (3)(3)} = 2.28.$$

$$\boxed{b_1 = 2.28}$$

$$b_2 = \frac{(\Sigma x_1^2)(\Sigma x_2 y) - (\Sigma x_1 x_2)(\Sigma x_1 y)}{(\Sigma x_1^2)(\Sigma x_2^2) - (\Sigma x_1 x_2)^2}$$

$$= \frac{(10)(-48) - (3)(17.8)}{(10)(32.8) - (3)(3)} = -1.67$$

$$\boxed{b_2 = -1.67}$$

putting these value into regression line equation to calculate intercept.

$$\hat{y} = b_0 + b_1 X_1 + b_2 X_2 .$$

$$b_0 = \hat{y} - b_1 X_1 + b_2 X_2$$

The given equation is true for mean value of X_1, X_2, y i.e. $\left(\frac{20}{5}, \frac{24}{5}, \frac{19.5}{5}\right)$, so.

$$b_0 = \frac{19.5}{5} - (2.28)\cdot\frac{(20)}{5} + (-1.67)\frac{(24)}{5} = 2.796.$$

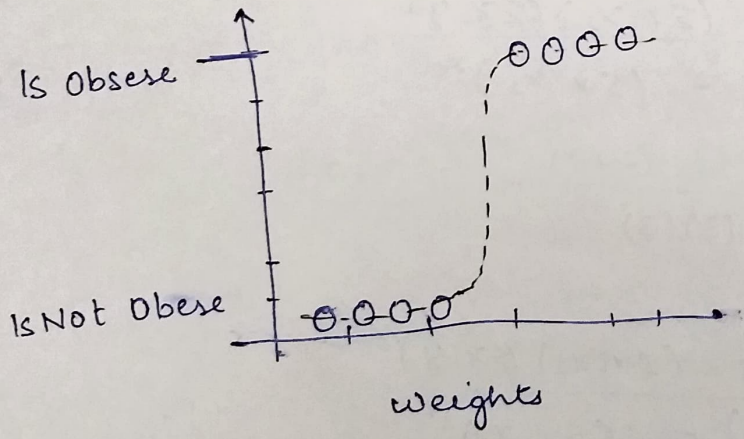
$$\boxed{b_0 = 2.796}$$

So, the equation of regression line is :-

$$\boxed{\hat{y} = 2.796 + (2.28) X_1 + (-1.67) X_2}$$

3.) Logistic Regression

Logistic Regression is similar to linear regression, except logistic regression predicts whether something is True or False, instead of predicting something continous like size.



So, instead of fitting a line to the data, logistic regression fits an 'S' shape 'logistic function'.

* It is mostly used for classification

logistic Regression's ability to provide probabilities and classify new samples using continous and discrete measurement. makes it popular machine learning method.

→ One big difference b/w linear regression & logistic regression is how the line is fit to the data.

- With linear Reg., we fit the line using "least mean squares". i.e. we find the line that minimizes the sum of squares of the residuals
- We use residuals to calculate R^2

* But logistic Regression does not have the same concept of a "residual". So its can't use least squares and it can't calculate R^2.

Artificial Neural Network

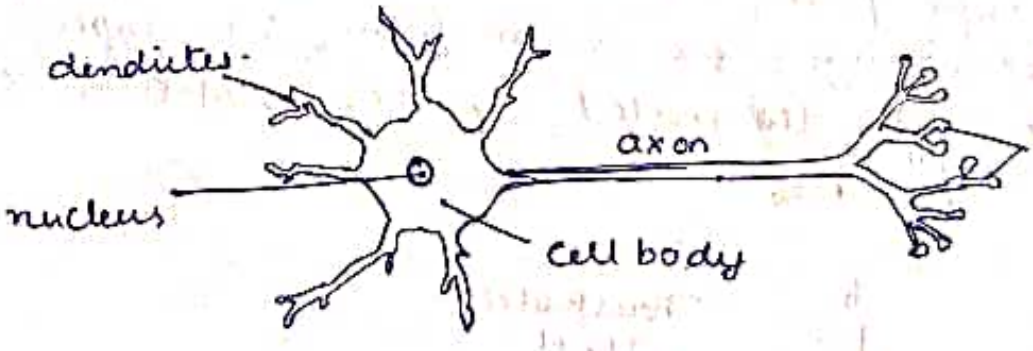
ANN are computational models that are inspired by Human brain. ANN are biologically inspired simulation performed on computer to perform certain specific tasks like - clustering, classification, pattern recognition.

Biologically inspired network of artificial Neuron configured to perform specific tasks.

What is Neural Network?

The term 'Neural' is derived from the Human Nervous system's basic functional unit 'neuron' or nerve cell that are present in the brain & other parts of human body.

Biological Neuron.



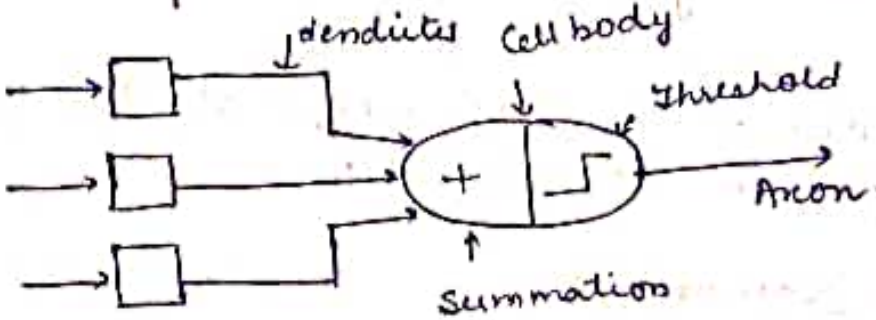
Dendrites : receives signal from other neurons.

Soma (cell body) : It sums all the incoming signals to generate output

Axon Structure : When the sum reaches a threshold value, neuron fires and the signal travels the axon to the other neuron.

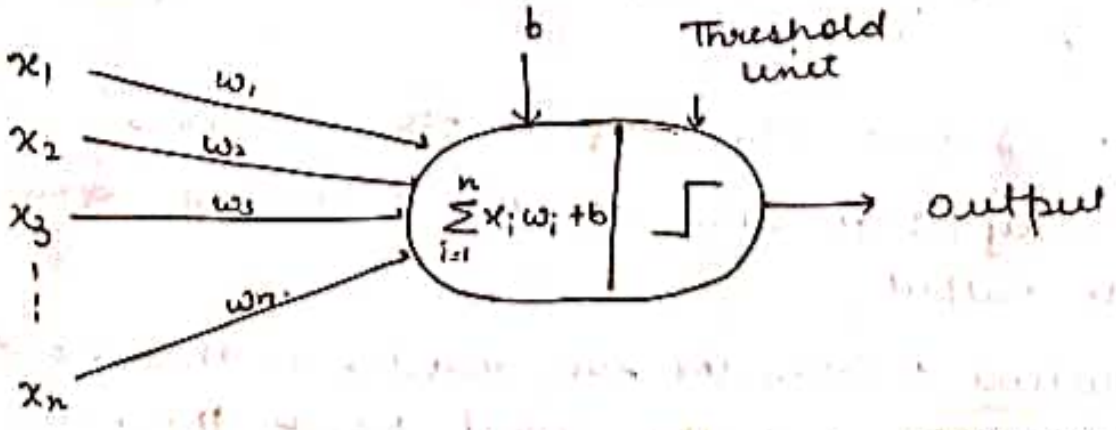
Synapses Working : The point of intersection of one neuron with the other neuron. The amount of signal transmitted depends upon the strengths (synaptic weights) of the connections.

→ A neuron network is a group of algorithms that identify the underlying relationship in a set of data similar to the human brain.
→ The NN helps to change the I/P so that the network gives the best best result without redesigning the O/P procedure.



How does ANN works?

ANN can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges with weights are connection b/w neuron O/P and neuron I/P.
ANN receives info. from the external world in the form of pattern and image in vector form. The inputs are mathematically designated by the notation $x(n)$ for n number of I/Ps.



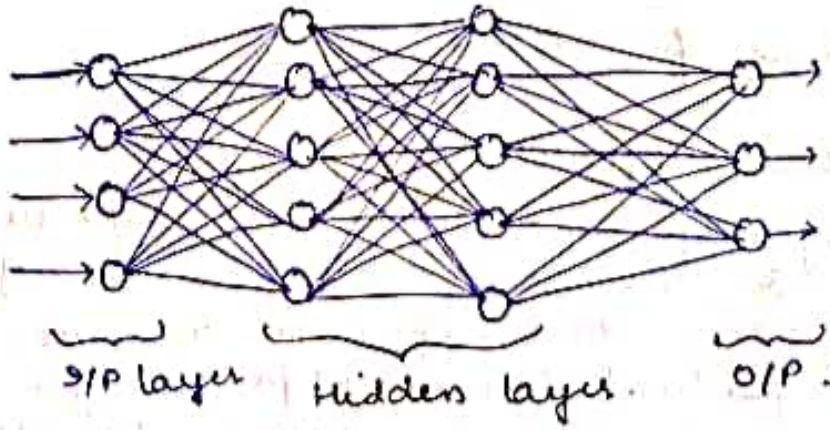
$w_1\ w_2\ w_3 \ldots w_n$ – weights of connection
$x_1\ x_2\ x_3 \ldots x_n$ – inputs
b – bias.

→ ~~ANN can be used~~
→ Each I/P is multiplied by its corresponding weights. Weights are info. used by the NN to solve a problem.
Typically weight represents the strength of the

interconnection between neurons inside the NN
→ The weighted i/p are all summed up inside the computing neuron (Artificial neuron). In case the weighted sum is zero, bias is added to make the o/p not zero or to scale up the system response. Bias has the weight and input always equal to '1'.
→ The sum corresponds to any numerical value ranging 0 to infinity. To limit the response to arrive at the desired value, the threshold value is set up. For this, the sum is passed through an activation function.
→ AF is set to the transfer function used to get the desired o/p.

ANN - basic Architecture

→ A typical NN contains a large no. of artificial neurons called units arranged in a series of layer.
→ ANN comprises of different layers -



Input layer. - It contains those neuron which receive i/p from the outside world on which the n/w will learn, recognize about or otherwise process.

Output layer. - It contains units that respond to the info about how its learned any task.

Hidden layer. - These neurons are in b/w i/p & o/p layers. The job of the hidden layer is to transform the i/p into something that the o/p unit can use in some ways.

Learning Techniques in Neural N...

1. **Supervised learning-**
In supervised learning, the training data is input to the network. and the desired output is known weights are adjusted until production yields desired value.

2. **Unsupervised learning**
The I/P data is used to train the n/w whose O/P is known. The n/w classifies the I/P data and adjust the weights by feature extraction in I/P data.

3. **Reinforcement learning**
Here the value of the output is unknown, but the network provides feedback on whether the O/P is right or wrong. It is semi-supervised learning.

Training Algorithms for ANN.

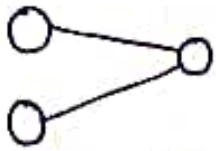
1. **Gradient Descent Algorithm.**
Simplest training algo used in case of supervised training model. In case, the actual O/P is different from the target O/P., the difference or error is find out. The gradient descent algo. changes the weights of the n/w in such a manner to minimize this mistake.

2. **Back Propagation Algorithm.**
It is an extension of the gradient-based delta learning rule. Here, after finding the error, the error is propagated backward from the I/P layer via hidden layer. It is used in the case of Multilayer NN.

Neural Network Architecture Types.

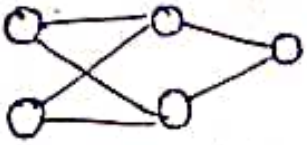
1. **Single layer Perceptron model.**

NN is having 2 I/P unit and one O/P unit with no hidden layers. These are also known as 'Single layer perceptrons".



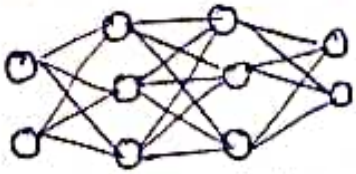
2. **Radial Basis Function Neural Network.**

Similar to feed-forward NN except radial basis function is used as AF of these neuron.



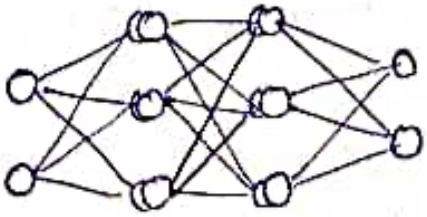
3. **Multi layer perceptron Neural Network.**

N/W uses more than one hidden layer of neurons. These are also called deep, feed-forward Neural Networks.



4. **Recurrent Neural Network.**

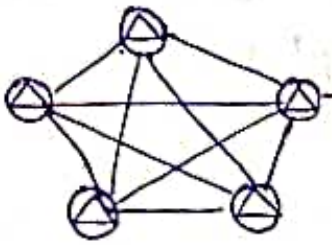
In this hidden layer neuron have self connection. Recurrent Neural Network poses memory. At any instance, Hidden layer Neuron receives activation from the lower layer as well as its previous activation value.



5. **Hopfield Network.**

A fully interconnected networks of neurons in which each neuron is connected to every other neurons the network is trained with I/P patterns by setting a value of neurons to the desired pattern. Then

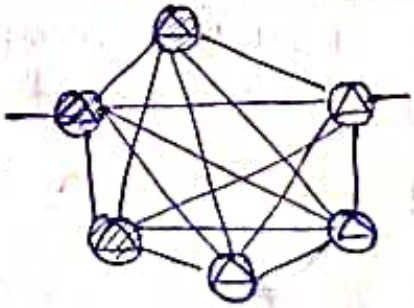
its weights are computed. The weights are not changed. Once trained for one or more patterns, the n/w will converge to the learned patterns. It is difficult from other NN.



backfed O/P unit.

5. Boltzman Machine NN

These n/w are similar to the Hopfield network except some neurons are input, while other are hidden in nature. The weights are initialized randomly & learn through the backpropagation algorithms.



backfed O/P layer.

Probabilistic Hidden layer.

Activation Function in Neural Network.

→ It defines the output of input or set of inputs or in other terms defines nodes of the O/P of node that is given as I/P.
→ They basically decides to activation or deactivate neurons to get the desired O/P.
→ It performs non-linear transformation on the I/P to get better results on a complex NN.
→ It helps to normalize the O/P of any input in the range between 1 to -1.
→ It reduces computation time if the Activation function is efficient.
→ The addition of AF to NN executes the non-linear transformation to input & make it capable to solve complex problems such as language translation and image classification.

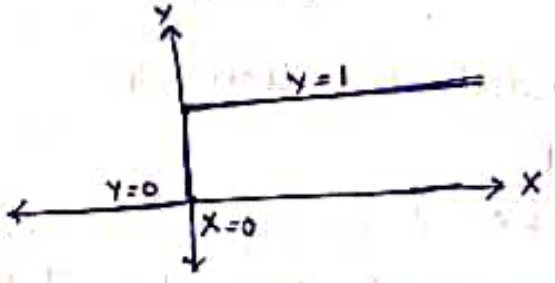
Types of Activation Function

Linear AF

1. Binary Step Activation Function :-

It is basically a threshold base classifier, in this, we decide some threshold value to decide O/P that neuron should be activated or deactivated.
It is useful to classify binary problem or classifier.

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$



2. Linear Activation Function :-

→ It is a simple Straight line AF where our function is directly proportional to the weighted sum of neuron or inputs.
→ Better in giving wide range of activations and a line of a +ve slope may increase the firing rate as the I/P rate increases

In binary, either a neuron is firing or not.

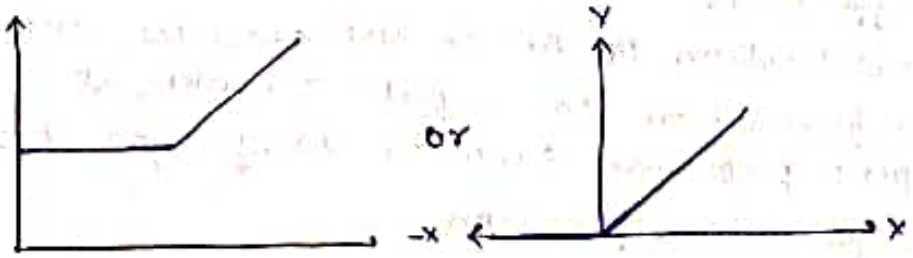
$$Y = mZ$$

where derivative with respect to Z is constant m.
The meaning gradient is also constant and it has nothing to do with Z.

Non-linear AF

3. ReLU (Rectified Linear Unit) AF

→ Most widely used AF ranges from 0 to infinity.
→ All the -ve values are converted into 0.
→ Although it looks like a linear function, ReLU has a derivative function & allows for back propagation.

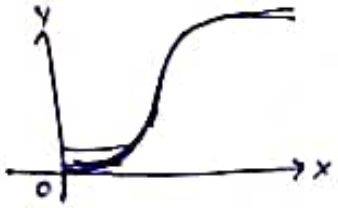


$$f(x) = \begin{cases} x, & x > 0 \\ 0, & \text{otherwise} \end{cases}$$

4. Sigmoidal AF

→ It is a probabilistic approach towards decision making and ranges in b/w 0 to 1.
→ As the range is minimum, prediction would be more accurate
The eqn for sigmoidal function is

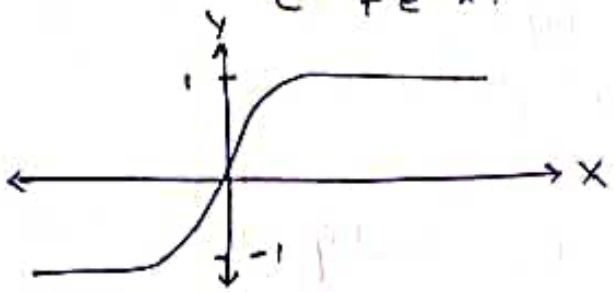
$$f(x) = \frac{1}{(1 + e^{(-x)})}$$



5. Hyperbolic Tangent Activation function (Tanh)

→ slightly better than sigmoidal function
→ It also predict btw two classes but it maps the -ve input into -ve quantity only and range in between -1 to 1.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



6. Softmax AF

→ It is mainly used at the last layer i.e. o/p layer for decision making.

→ It can be used for both binary as well as multi-class classification problem.

$$f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}$$

Problems of Linear AF :-

1. Not possible to use backpropagation to train the model. — the derivative of the function is a constant, and has no relation to the input X. So its not possible to go back & understand which weights in the i/p neurons can provide a better prediction.

2. All layers of the NN collapse into one — no matter how many layers in the NN, the last layer will be a linear function of the first layer. (because a linear combination of linear functions is still a linear function.
So it turns the NN into just one layer.
It is simply a linear regression model.

Problems with ReLU function.

1. The dying RELU problem - when I/P reaches zero, or are -ve, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot ~~function~~ learn.

Solution :- Leaky ReLU

Leaky ReLU

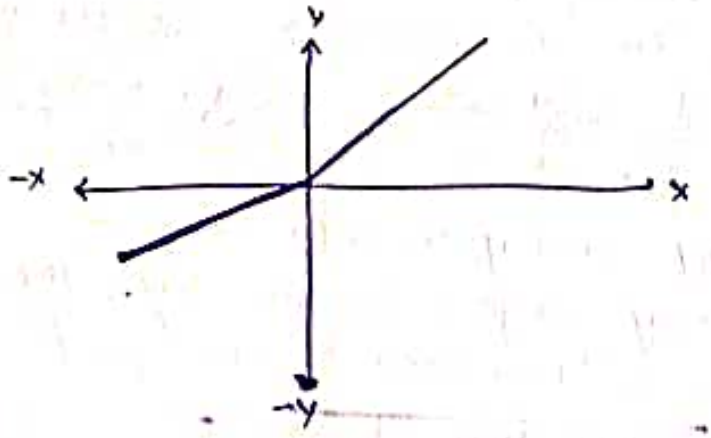
It has a small +ve slope in the -ve area, so it does enable backpropagation, even for negative input values.

2. Results not consistent - leaky ReLU does not provide consistent prediction for -ve i/p values.

Solution :- Parametric ReLU.

Parametric ReLU

Allows the -ve slope to be learned - unlike leaky ReLU, this function provides the slope of the -ve part of the function as an argument. Thus, it is possible to perform backpropagation and learn the most appropriate value of α.



Parametric

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & \text{otherwise} \end{cases}$$

Leaky

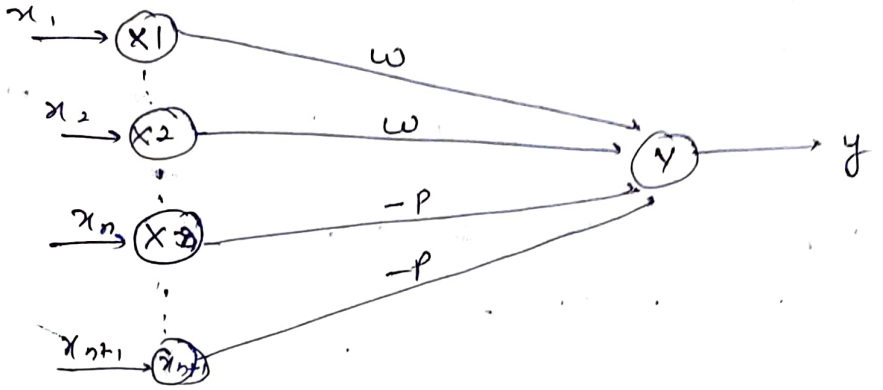
$$f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & \text{otherwise} \end{cases}$$

where $\alpha = 0.01$

McCulloch - Pitts Neuron Model (1943)

There is a fixed threshold for each neuron and if the net I/P to the neuron is greater than threshold then the neuron fires.

→ Most widely used in logic functions.



→ Has both excitory with weight ($w > 0$) & inhibitory with weight $-p$ ($p < 0$) connections.

→ Since the firing of O/P neuron is based on threshold, activation function is defined as :-

$$f(y_{in}) = \begin{cases} 1 & , y_{in} \geq \theta \\ 0 & , y_{in} \leq \theta \end{cases}$$

→ If inhibitory weights are used then threshold with AF should satisfy the following condition.

$$\theta > nw - p$$

→ No particular training algorithm.

→ Analysis has to be formed to determine value of weights and threshold.

Q. Implement AND NOT function using McCulloch Neuron. (use Binary Data).

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

Case-1 :- Assume both weights to be excitatory
i.e. $w_1 = w_2 = 1$

Calculate net i/p for the 4 i/p using.

$y_{in} = x_1 w_1 + x_2 w_2$.

for i/p.,

(1, 1) = $y_{in} = 1 \times 1 + 1 \times 1 = 2$.

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

(0, 1) = $y_{in} = 0 \times 1 + 1 \times 1 = 1$

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

→ It is not possible to fire neuron for i/p (1, 0) only. Hence the weights are not suitable.

Assume one weight is excitatory and other is inhibitory i.e.

$w_1 = 1$, $w_2 = -1$.

calculate net i/p.,

(1, 1) . $y_{in} = (1 \times 1) + (1 \times -1) = 0$.

(1, 0) $y_{in} = (1 \times 1) + (0 \times -1) = 1$

(0, 1) $y_{in} = (0 \times 1) + (1 \times -1) = -1$

(0, 0) $y_{in} = (0 \times 1) + (0 \times -1) = 0$.

→ Now it is possible to fire the neuron for i/p. only by fixing a threshold of 1. i.e. $\theta \geq 1$.

Thus $w_1 = 1$, $w_2 = -1$, $\theta \geq 1$

Value of θ can be calculated

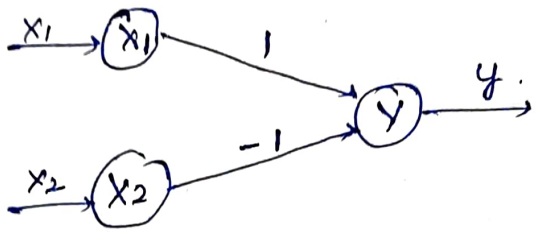
$\theta \geq n w - p$.

$\theta \geq 2 \times 1 - 1$

$\theta \geq 2 - 1$ $\quad \boxed{\theta \geq 1}$.

Thus, O/P of neuron Y can be written as

$$y = f(y_{in}) = \begin{cases} 1 & , y_{in} \geq 1 \\ 0 & , y_{in} < 1 \end{cases}$$



Rosenblatt's Perceptron (1957)

This model was designed to overcome most issues of the McCulloch-Pitts neuron.

- it can process non-boolean I/P.
- it can assign different weight each I/P automatically.
- the θ is computed automatically.

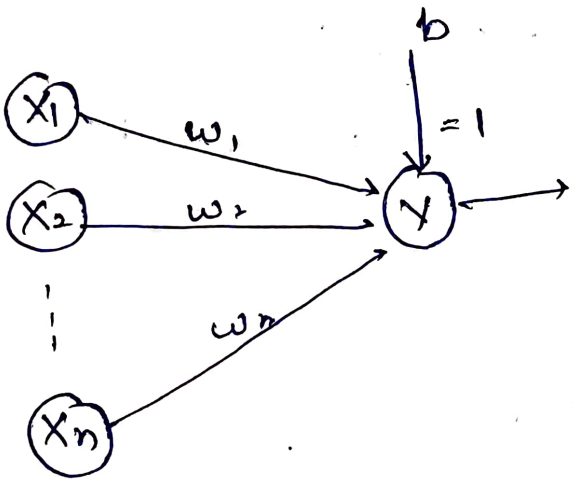
A perceptron is a single layer NN. A perceptron can simply be seen as a set of I/P, that are weighted and to which we apply an Activation function. This produces sort of a weighted sum of i/p, resulting in an O/P. This is typically used for classification problems, but can also be used for regression problems.

Two types of perceptron model.

1. Single layer Perceptron.
2. Multi- layer perceptron.

1. Single layer Perceptron.

- Used for classification of linear pattern
- consist of I/P, weight & bias.



$$Y_{in} = \sum w_i x_i + b$$

$$Y = f(y_{in})$$

- It consist of single neuron weight and bias.

Principal Component Analysis

→ converts high dimensionality into low dimensionality. (Dimension Reduction).

Perceptron Training Algorithm

Step 1:- Initialize weights & bias (for easy calculation, them to 0). Also initialize LR α ($0 < \alpha < 1$) for simplicity. set α to 1.

Step-2:- Calculate O/P of the n/w, to do so first calculate net I/P.

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

where 'n' is the no. of i/p neurons in the i/p layer, then apply AF over the net I/P. Calculated to obtain the O/P.

$$Y = f(y_{in}) = \begin{cases} 1 & ; y_{in} > \theta \\ 0 & ; -\theta \le y_{in} \le \theta \\ -1 & ; y_{in} < -\theta \end{cases}$$

Step-3. Weight & bias adjustment :- Compare the value to actual (calculated) and desired / target O/P.

if $y \neq t$, then.

$$w_i(new) = w_i(old) + \alpha x_i$$

$$b_i(new) = b(old) + \alpha t$$

else we have,

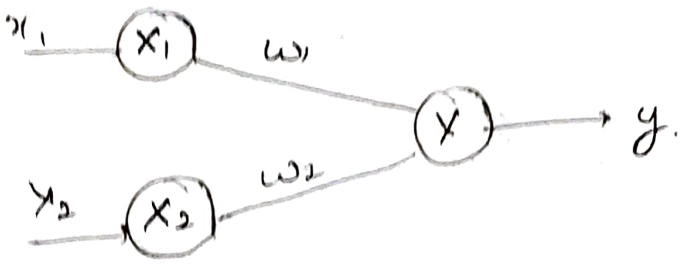
$$w_i(new) = w_i(old)$$

$$b(new) = b(old)$$

Example.

Q:. Implement AND function for (bipolar i/p & targets)

x_1	x_2	t
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1



Initialize $w_1 = w_2 = b = 0$ & $\theta = 0$, $\alpha = 1$.

For 1 s/p.

$x_1 = 1$ $x_2 = 1$ $t = 1$.

Calculate net i/p.

$$y_{in} = b + w_1 x_1 + w_2 x_2.$$
$$= 0 + 0 + 0 = 0.$$

O/P y computed by applying Activation over the net i/p calculated.

$$y = f(y_{in}) = \begin{cases} 1 & y_{in} > 0 \\ 0 & y_{in} = 0 \\ -1 & y_{in} < 0. \end{cases}$$

$f(0) = 0 \neq t$. Hence update weight.

w_1 (new) = w_1 (old) + $\alpha t x_1$,

$= 0 + 1 * 1 1 = 1.$

w_2 (new) = w_2 (old) + $\alpha t x_2$

$= 0 + 1 * 1 * 1 = 1 = 1$

b (new) = b(old) + αt.

$= 0 + 1 \times 1 = 1.$

$\Delta w_1 = \alpha t x_1 = 1$

$\Delta w_2 = \alpha t x_2 = 2$

$\Delta b = \alpha t = 1.$

for II I/P.

$x_1 = 1$, $x_2 = -1$, $t = -1$.

$y = b + w_1 x_1 + x_2 w_2 = 1 + 1 - 1 = 1$.

$y = f(y_{in}) = f(1) = 1 \neq t$. Hence update weight.

$w_1(new) = w_1(old) + (\alpha + x_1) = 1 + 1(-1)(1) = 0$

$w_2(new) = w_2(old) + (\alpha t x_2) = 1 + (1)(-1)(-1) = 2$.

$b(new) = b(old) + \alpha t. = 1 + (1)(-1) = 0$.

$\Delta w_1 = -1 = \Delta w_2 = 1$ $\Delta b = -1$

for III I/P.

$x_1 = -1$, $x_2 = 1$, $t = -1$.

$y(in) = b + w_1 x_1 + w_2 x_2 = 0 + (-1)(0) + (1)(2) = 2$.

$y(y_{in}) = f(2) = 1 \neq t$, Hence weight update.

$w_1(new) = 0 + (1)(-1)(-1) = 1$

$w_2(new) = 2 + 1(-1)(1) = 1$

$b(new) = 0 + (1)(-1) = -1$

$\Delta w_1 = 1$ $\Delta w_2 = -1$, $\Delta b = -1$

for IV I/P.

$x_1 = -1$ $x_2 = -1$ $t = -1$

$y_{in} = b + w_1 x_1 + w_2 x_2 = -1 + (1)(-1) + (1)(-1) = -3$.

$y = f(y_{in}) = f(-3) = -1 = t$. Hence don't update weight.

$\Delta w_1 = 0$ $\Delta w_2 = 0$ $\Delta b = 0$

$w_1 = 1$ $w_2 = 1$ $b = -1$

Epoch 1 completed.

Epoch 2.

→ for I I/P.

$x_1 = 1$, $x_2 = 1$, $t = 1$

$y_{in} = b + w_1 x_1 + w_2 x_2$.

$= -1 + (1)(1) + (1)(1) = 1$

$y = f(y_{in}) = f(1) = 1 = t$. Hence does not update

$\Delta w_1 = 0, \quad \Delta w_2 = 0, \quad \Delta b = 0$

$w_1 = 1, \quad w_2 = 1, \quad b = -1$

Backpropagation

The backpropagation algorithm performs learning on a multilayer feed-forward neural network.

Backpropagation learns by iteratively processing a set of training samples, comparing the network's prediction for each sample with the actual known class label.

For each training sample, the weights are modified so as to maximize the mean squared error between the network's prediction and the actual class.

These modifications are made in the "backwards" direction, i.e., from the output layer, through each hidden layer down to the first hidden layer (i.e. backpropagation).

The backpropagation algorithm is summarised as follows :-

Backpropagation Algorithm :-

Neural network learning for classification or prediction, using the backpropagation algorithm.

Input :

- D, a data set consisting of the training tuples and their associated target values.
- l, the learning rate.
- network, a multilayer feed-forward network.

Output :

A trained neural network.

Method :

1. Initialize all the weights & biases in the network;
2. while terminating condition is not satisfied
   {
3. for each training tuple X in D {
4. // Propagate the input forward :

```
5.  for each input layer unit j {
6.      Oj = Ij ; // O/P of input unit is its actual input value
7.  for each hidden or O/P layer unit j {.
8.      Ij = Σi wij Oi + θj ; // compute the net i/p of
        unit j w.r.t. the previous layer, i.
9.      Oj = 1/1+ e^-1j ; } // compute the O/P of each unit j
10.     // Backpropagate the errors:
11.     For each unit j in the O/P layer.
12.     Errj = Oj (1-Oj) (Tj -Oj); // compute the error.
13.     for each unit j in the hidden layers, from
        the last to the first hidden layer.
14.     Errj = Oj (1-Oj) Σk Errk wjk ; // compute the
        error w.r.t the next higher layer, k.
15.     for each weight wij in the network {.
16.     Δwij = (l) Errj Oi ; // weight increment.
17.     wij = wij + Δwij ; // weight update.
18.     for each bias θj in network. {
19.     Δθj = (l) Errj ; // bias increment
20.     θj = θj + Δθj ; // bias update.
21.     } }.
```

Initialize the weight

The weights in the n/w are initialize to small random numbers (eg., ranging from -1.0 to 1.0, or -0.5 to 0.5). Each unit has a bias associated with it. The biases are similarly initialize to small random numbers.

Each training tuple, X, is processed by the following steps.

Propagate the Inputs Forward.

First, the training tuple is fed to the i/p layer. of the n/w. The i/p pass through the i/p units unchanged. That is, for an i/p unit j, its

output, O_j, is equal to its i/p value, I_j.
Next, the net i/p and o/p of each unit in the hidden & output layers are computed. The net i/p of a unit in the hidden or o/p layer is computed to a linear combination of its i/p. The i/p of the unit are, in fact, the o/p of the units connected to it in the previous layer. To compute the net input to the unit, each i/p connected to the unit is multiplied by its corresponding weight, and that is summed. Given a unit j in the hidden or o/p layer, the net i/p, I_j, to unit j is

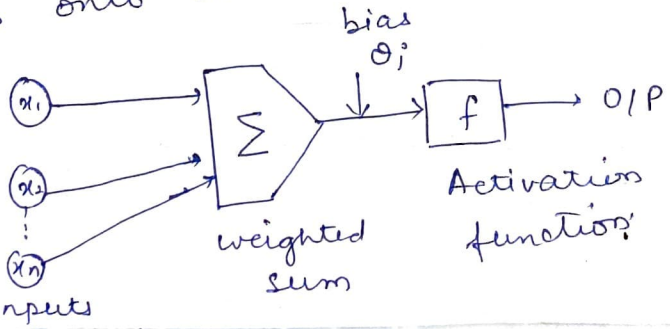
$$I_j = \sum_i w_{ij} O_j + \theta_j$$

where w_{ij} is the weight of the connection from unit i in the previous layer to unit j; O_i is the o/p of unit i from the previous layer; and θ_j is the bias of the unit. The bias acts as a threshold in that it serves to vary the activity of the unit.

Each unit in the hidden & o/p layers takes its net i/p and the applies an activation function to it, as shown. The function symbolizes the activation of the neuron of the neuron represented by the unit. The logistic; or sigmoid function is used. Given the net i/p I_j to unit j, then O_j, the o/p of unit j, is computed as:

$$O_j = \frac{1}{1 + e^{-I_j}}$$

This function is also referred to as a squashing function, because it maps a large input domain onto the small range of 0 to 1.



we compute the o/p values, O_j, for each hidden la up to and including the o/p layer, which gives th network's prediction.

Backpropagate the Error.

The error is propagated backward by updating the weights and biases to reflect the error of the network's prediction.

For a unit j in the o/p layer, the error Err_j is computed by :-

$$Err_j = O_j (1 - O_j)(T_j - O_j)$$

where O_j is the actual o/p of unit j, and T_j is the known target value of the given training tuple..

To compute the error of a hidden layer unit j, the weighted sum of errors of the units connected to unit j in the next layer is considered. The error of a hidden layer unit j is

$$Err_j = O_j (1 - O_j) \sum_K Err_K w_{jK}.$$

where w_{jK} is the weight of the connection from unit j to a unit K in the next higher layer, and Err_K is the error of unit K.

The weights and biases are updated to reflect the propagated errors. Weights are updated by the following equations, where Δw_{ij} is the change in weight w_{ij}

$$\Delta w_{ij} = (l)\, Err_j\, O_i$$

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

The variable l is the learning rate having a value between 0.0 to 1.0.

Biases are updated by the following equations below, where Δb_j is the change in the bias b_j.

$$\Delta b_j = (l)\, Err_j$$

$$b_j = b_j + \Delta b_j$$

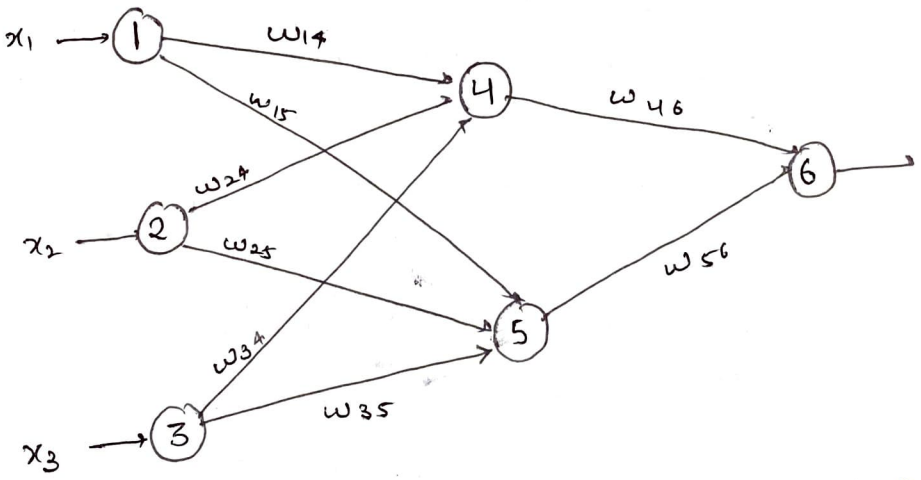
Note that here we are updating the weights and bias after the presentation of each tuple. This referred to as case updating.

Terminating condition

Training stops when.

- All Δw_{ij} in the previous epoch were so small as to be below some specified threshold, or.
- All percentage of tuples misclassified in the previous epoch is below some threshold, or.
- A prespecified no. of epochs have expired.

Example :-



where, $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $w_{14} = 0.2$, $w_{15} = -0.3$

$w_{24} = 0.4$ $w_{25} = 0.1$ $w_{34} = -0.5$, $w_{35} = 0.2$

$w_{46} = -0.3$ $w_{56} = -0.2$ $b_4 = -0.4$, $b_5 = 0.2$, $b_6 = 0.1$

Firstly, this sample is fed into the n/w, and the net i/p and o/p of each unit are computed.

$$O_4 = f(x_1 w_{14} + x_2 w_{24} + x_3 w_{34} + b_4)$$

$$O_4 = f(0.2 + 0 - 0.5 - 0.4)$$

$$O_4 = f(-0.7)$$

Using sigmoidal Activation function, where.

$$Y = f(y) = \frac{1}{1+e^{-y}}.$$

$$O_4 = \frac{1}{1+e^{-(-0.7)}} = 0.332.$$

Now we we calculate O/P of unit 5 same as unit 4.

$$O_5 = f(-0.37 0 + 0.2 + 0.2) = f(0.1) = \frac{1}{1+e^{-(0.1)}} = 0.525$$

$$O_6 = f[(-0.3)(0.332) - (0.2)(0.525) + 0.1)$$

$$= f(-0.105) = \frac{1}{1+e^{-(-0.105)}} = 0.474.$$

The error of each unit is computed and propagated backwards. The error values for each unit are:-

$$Err_6 = (0.474)(1-0.474)(1-0.474)$$

$$= 0.1311$$

$$Err_5 = (0.525)(1-0.525)(0.1311)(-0.2)$$

$$= -0.0065$$

$$Err_4 = (0.332)(1-0.332)(0.1311)(-0.3)$$

$$= -0.0087.$$

Now we will update the weights and biases.

$$w_{46} = -0.3 + [(0.9)(0.1311)(0.332)]$$

$$= -0.261$$

$$w_{56} = -0.2 + [(0.9)(0.1311)(0.525)]$$

$$= -0.138.$$

$$b_6 = 0.1 + [(0.9)(0.1311)] = 0.218$$

$w_{14} = 0.2 + [(0.9)(0.0087)(1)] = 0.192$

$w_{15} = -0.3 + [(0.9)(-0.0065)(1)] = -0.306$

$w_{24} = 0.4 + [(0.9)(-0.0087)(0)] = 0.4$

$w_{25} = 0.1 + [(0.9)(-0.0065)(0)] = 0.1$

$w_{34} = -0.5 + [(0.9)(-0.0087)(1)] = -0.508$

$w_{35} = 0.2 + [(0.9)(-0.0065)(1)] = 0.194.$

$b_5 = 0.2 + (0.9)(-0.0065) = 0.194.$

$b_4 = -0.4 + [(0.9)(-0.0087)] = -0.408.$

Time Series Analysis

Time series is a sequence of observations of categorical or numeric variable indexed by a date, or time stamp. A clear example of time series data is the time serie of a stock price.
In the following table, we can see the basic structure of time series data. In this case the observations are recorded every hour.

Time Stamp	Stock - Price
2021-10-11 09:00:00	100
2021-10-11 10:00:00	110
2021-10-11 11:00:00	105
2021-10-11 12:00:00	90
2021-10-11 13:00:00	120

Normally, the first step in time series is to plot the series, this is normally done with a line chart.
The most common application of time series analysis is forcasting future values of a numeric value using the temporal structure of the data. This means, the available observations are used to predict values from the future.

The temporal ordering of the data, implies the traditional regression methods are not useful. In order to build robust forecast, we need models that take into account the temporal ordering of the Data.

The most widely used model for Time Series Analysis is called Autoregressive moving Average (ARMA). The model consist of two parts, an autoregressive (AR) part and a moving average (MA) part. The model is usually then refered to as the ARMA (p, q) model where p is the autoregressive part and q is the order of the moving average part.

Autoregressive model.

The AR(p) is read as autoregressive model of order p. Mathematically, it is written as -

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \varepsilon_t$$

where $\{\phi_1, \ldots, \phi_p\}$ are parameters to be estimated, c is the constant, and the random variable ε_t represent the white noise. Some constraints are necessary on the values of the parameters so that the model remains remains stationary.

Moving Average.

The notation MA(q) refers to the moving average model of order q -

$$X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

where the $\theta_1, \ldots, \theta_q$ are the parameters of the model, μ is the expectation of X_t and the $\varepsilon_t, \varepsilon_{t-1}, \ldots$ are, the white noise error terms.

Auto Regressive moving Average (ARMA)

The ARMA (p, q) model combines p autoregressive terms and q moving-average terms. Mathematically the model is expressed with the following formula -

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \phi_i X_{t-1} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

We can see that the ARMA (p,q) model is combination of AR(p) and MA(q) models.

To give some intuition of the model consider that the AR part of the equation seeks to estimate parameters of X_t-i observations of in order to predict the value of the variable in X_t. It is in the end a weighted average of the past values. The MA section uses the same approach but with the error of previous observations, ε_t-i. So in the end, the result of the model is a weighted average.

Fuzzy logic

Fuzzy logic is an approach to computing based on "degree of truth" rather than the usual "true or false" (1 and 0) boolean value logic on which the modern computer is based.

- Fuzzy logic (FL) is a method of reasoning that resembles human reasoning. The approach of FL imitates the way of decision making in human that involves all intermediate possibilities between digital value YES or NO.
- The conventional logic block that a computer can understand takes precise input and produces a definite output as TRUE or FALSE, which is equivalent to human's YES or NO.
- The inventor of fuzzy logic, Lotfi Zadeh, observed that unlike computers, the human decision making includes a range of possibilities between YES or NO, such as.

Certainly Yes
Possibly Yes
Cannot Say
Possibly NO
Certainly No.

The fuzzy logic works on the levels of possibility of input to achieve the definite output.

Fuzzy logic System Architecture

It has four main parts as shown

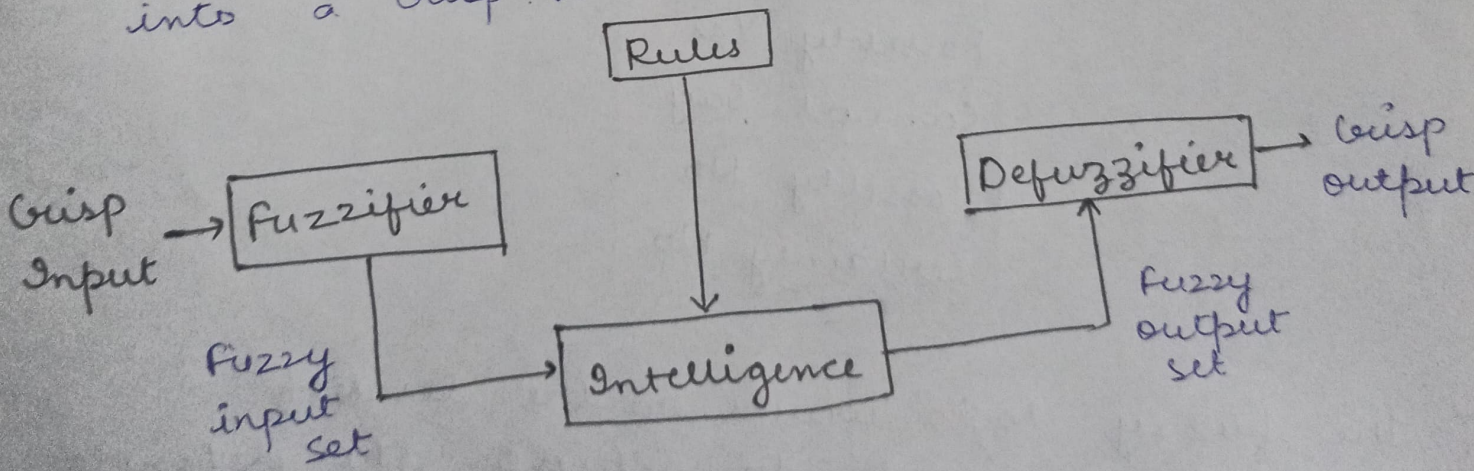
- **Fuzzification module** :- It transforms the system inputs, which are crisp numbers, into fuzzy sets. It splits the input signal into five steps such as :-

  LP - x is large positive
  MP - x is medium positive
  S - x is small
  MN - x is medium negative
  LN - x is large negative

- **Knowledge Base** :- It stores IF-THEN rules provided by the experts.

- **Inference Engine** :- It stimulates the human reasoning process by making fuzzy inference on the inputs and IF-THEN rules.

- **Defuzzification Module** :- It transforms the fuzzy set obtained by the inference engine into a crisp value.



Membership function

Membership function allow you to quantify linguistic term and represent a fuzzy set graphically. A member function for a fuzzy set A on the universe of discourse X is defined as

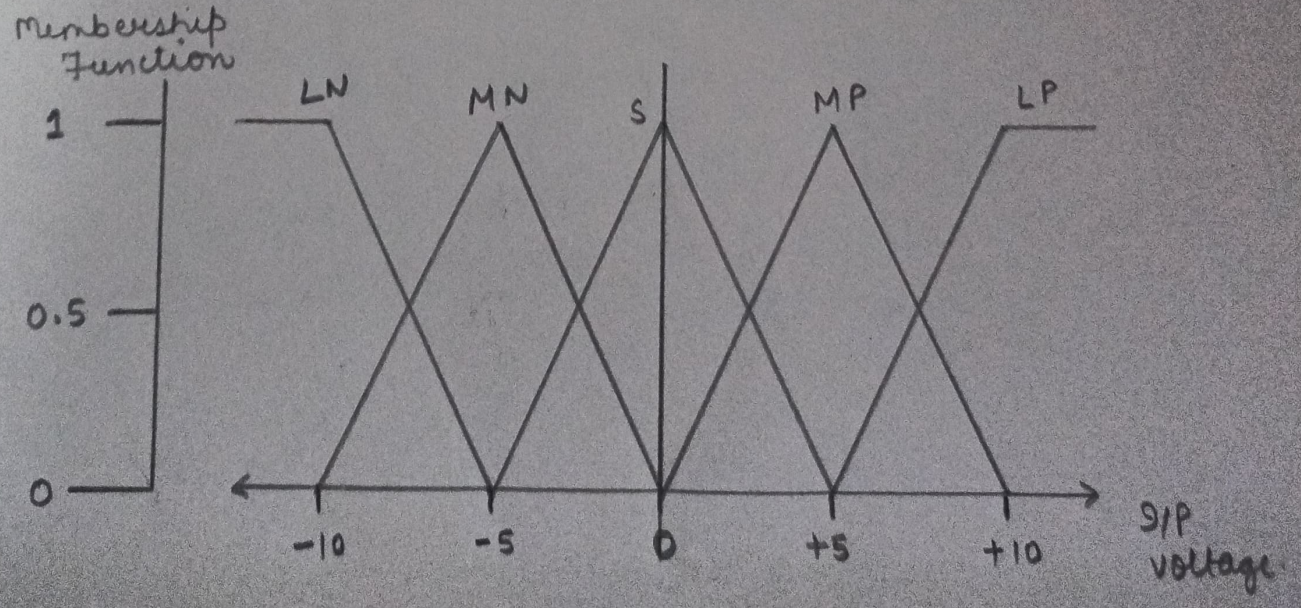
$$\mu_A : X \rightarrow [0, 1]$$

Here, each element of X is mapped to a value between 0 and 1. It is called membership value or degree of membership. It quantifies the degree of membership of the element in X to the fuzzy set A.

- x-axis represents the universe of discourse.
- y-axis represents the degree of membership in the [0, 1] interval.

There can be multiple membership functions applicable to fuzzify a numerical value. Simple membership functions are used as use of complex functions does not add more precision in the output.

→ All membership functions for LP, MP, S, MN and LN are shown as below.



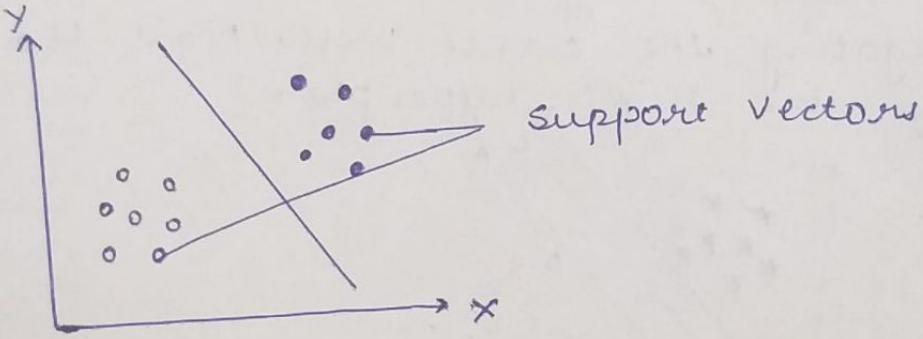
The triangular membership function shapes are most common among various other membership function shapes such as trapezoidal, sigmoidal and Gaussian.

Here, the input of 5-level fuzzifier varies from -10 volts to +10 volts. Hence the corresponding output also changes.

Support Vector Machine

"Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification and regression challenges. However, it is mostly used in classification problem. In the SVM algorithms, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate.

Then, we perform classification by finding the hyper-plane that differentiate the two classes very well.



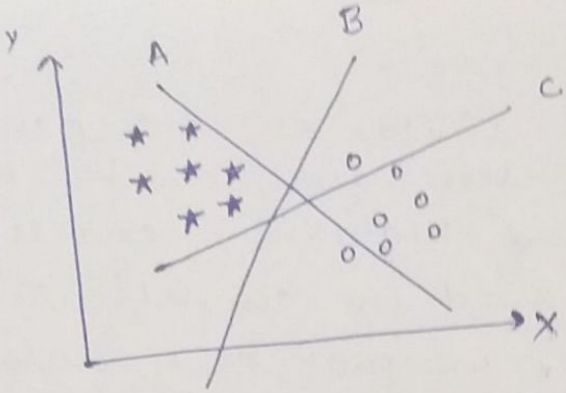
Support vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes. (hyperplane / line)

How does it work?

Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is "How can we identify the right hyperplane".

- **Identify the right hyperplane (Scenario-1):-**

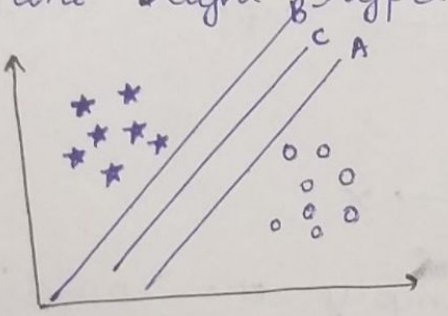
  Here, we have 3 hyperplanes (A, B, and C). Now, identify the right hyper-plane to classify stars and circles.



You need to remember a thumb rule to identify the right hyper-plane : "Select the hyper-plane which segregates the two classes better".
In this scenario, hyperplane "B" has excellently performed the job.

- Identify the right hyper-plane (Scenario-2)

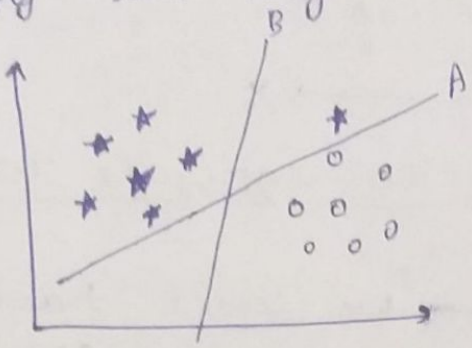
Here, we have 3 hyperplanes (A, B, and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?



Here, maximizing the distance b/w nearest data points and hyperplane will help us to decide the right hyper-plane. This distance is called as Margin.

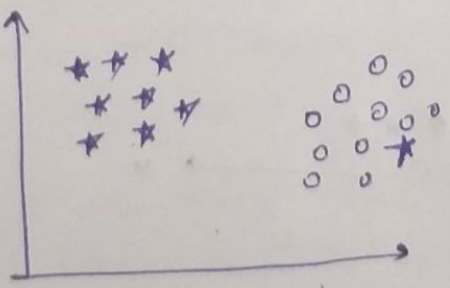
Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyperplane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chances of miss-classification.

- Identify the right hyper-plane (Scenario 3)
Use the rules as discussed in previous section to identify the right hyper-plane.

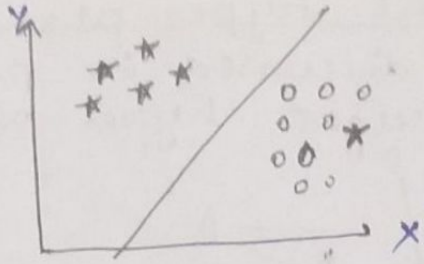


Some of you may have selected a hyper plane B as it has higher margin compared to A.
But, here is a catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin.
Here, hyper-plane B has a classification error and A has classified all correctly.
Therefore, right hyper-plane is A.

- Identify can we classify two class (Scenario-4)
Below, we are not able to segregate the two classes using a straight line, as one of the stars lies in the territory of other (circle) class as an outliner.

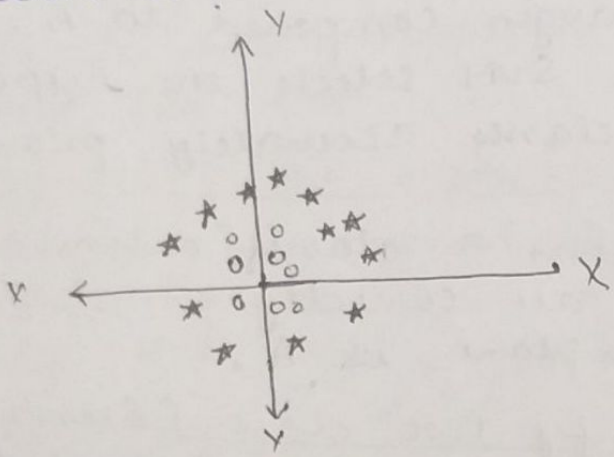


One star at other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.

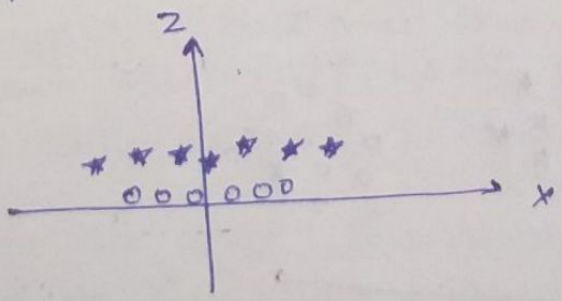


Find the hyper-plane to segregate to classes (Scenario 5)

In the scenario below, we can't have linear hyper plane between the two classes, so how to solve it?



SVM can solve this problem. Easily I. it solves by introducing another feature. Here, we add a new feature $z = x^2 + y^2$. Now, let's plot the data points on axis x and z.



On above plot, points to consider are :-

- All value of z would be positive always because z is squared sum of both x and y.
- In the original plot, circle appear close to the origin of x and y axis, leading to lower value of z and star relatively away from the origin result to higher value of z.

In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we add another this feature manually to have a hyper-plane

No, the SVM algorithm has a technique called the Kernel trick. The SVM Kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space. i.e. converts not separable problem to seperable problem. It is mostly used in non-linear seperation problem

Major Kernel Functions in SVM.

Kernel Function is a method used to take data as input and transform into the required form of processing data. "Kernel" is used to set of mathematical function used in SVM provides the window to manipulate the data.

So, Kernel function generally transforms the training set of data so that a non-linear decision surface is able to transformed to a linear equation in a higher number of dimension spaces.

Basically, it returns the inner product between two points in a standard feature dimension.

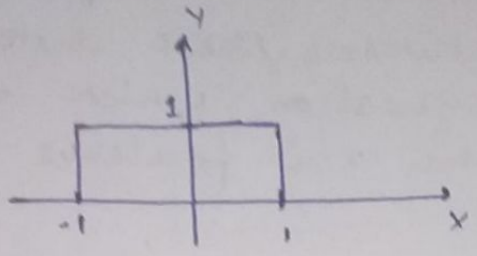
Kernel Rules

Define Kernel or a window function as follows.

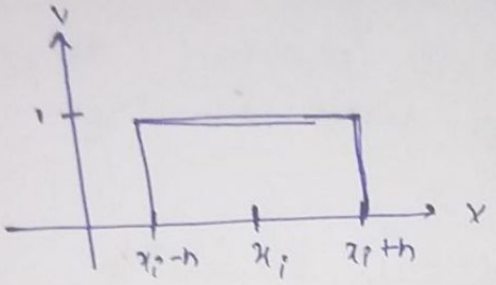
$$K(\bar{x}) = \begin{cases} 1 & \text{if } |\bar{x}| \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

This value of this function is 1 inside the closed ball of radius 1 centered at the origin, and 0 otherwise. As shown in the

figure below :-



For a fixed x_i, the function is $K(z - x_i)/h = 1$ inside the closed ball of radius h centered at the x_i, and 0 otherwise as shown in the figure.



So, by choosing the argument of $K(\cdot)$, you have moved the window to be centered at the point x_i and to be of radius h.

Examples of SVM Kernels.

(1.) Polynomial Kernel

It is popular in image processing. Equation is :-

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^d$$

where d is the degree of the polynomial

(2) Gaussian Kernel

It is a general-purpose kernel; used when there is no prior knowledge about the data. Equation is :-

$$K(x, y) = \exp\left(\frac{-|x - y|^2}{2\sigma^2}\right)$$

(3.) Gaussian Radial Basis function (RBF)

It is a general purpose kernel, used when there is no prior knowledge about the data.

Equation is :-

$$K(x_i, x_j) = \exp\left(-\gamma |x_i - x_j|^2\right)$$

for $\gamma = 0$

Sometimes parametrized using :-

$$\gamma = 1/2\sigma^2$$

(4) Laplace RBF Kernel

It is a general-purpose kernel, used when there is no prior knowledge about the data.

Equation is :-

$$K(x, y) = \exp\left(-\frac{|x - y|}{\sigma}\right)$$

(5.) Hyperbolic tangent Kernel

We can use it in neural networks.

Equation is

$$K(x_i, x_j) = \tanh(kx_i \cdot x_j + c)$$

for some (not every) $k > 0$ and $c < 0$.

(6) Sigmoid Kernel

We can use it as a proxy for neural networks. Equation is

$$K(x, y) = \tanh(\alpha x^T y + c)$$

Principal Component Analysis (PCA)

PCA is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contain most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller dataset are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithm without extraneous variables to process.

So to sum up, the idea of PCA is simple — reduce the number of variables of a data set, while preserving as much information as possible.

Example.

Q:- Given the following data, use PCA to reduce the dimension from 2 to 1.

Features	Ex-1	Ex-2	Ex-3	Ex-4
x	4	8	13	7
y	11	4	5	14

Solution :-

Step-1 :- Dataset

No. of features $n = 2$

No. of samples $N = 4$.

Step-2 :-

Computation of mean variable

$$\bar{x} = \frac{4+8+13+7}{4} = 8$$

$$\bar{y} = \frac{11+4+5+14}{4} = 8.5$$

Step-3

Computation of covariance matrix

Ordered pair of x and y are :-

(x, x) (x, y) (y, x) (y, y)

(i.) Covariance of all ordered pairs

$$cov(x, x) = \frac{1}{N-1} \sum_{K=1}^{N} (x_{iK} - \bar{x}_i)(x_{jK} - \bar{x}_j)$$

$$cov(x, x) = \frac{1}{4-1}\left[(4-8)^2 + (8-8)^2 + (13-8)^2 + (7-8)^2\right]$$

$$\boxed{cov(x, x) = 14.}$$

$$cov(x, y) = \frac{1}{4-1}\left[(4-8)(11-8.5) + (8-8)(4-8.5) + (13-8)(5-8.5) + (7-8)(14-8.5)\right]$$

$$\boxed{cov(x, y) = -11}$$

$$cov(y,x) = cov(x,y)$$

$$\boxed{cov(y,x) = -11}$$

$$cov(y,y) = \frac{1}{4-1}\left[(11-8.5)^2 + (4-8.5)^2 + (5-8.5)^2 + (4-8.5)^2\right]$$

$$\boxed{cov(y,y) = 23}$$

(ii) Covariance matrix.

matrix will be $n \times n$ i.e 2×2.

$$S = \begin{bmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{bmatrix} = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

Step-4.

Eigen value, eigen vector, and Normalized eigen vectors

(i) Eigen Value (λ)

$\det(S - \lambda I) = 0$,

where λ is eigen value and

I is the 2×2 identity matrix i.e. $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$$\lambda I = \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

$$\det \begin{bmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{bmatrix} = 0$$

$(14-\lambda)(23-\lambda) - (-11)(-11) = 0$

$\lambda^2 - 37\lambda + 201 = 0$

$\lambda = 30.3849, \quad 6.6151$

$\lambda_1 > \lambda_2$

So, $\lambda_1 = 30.3849 \rightarrow$ (First principal component)

$\lambda_2 = 6.6151$

(ii) Eigen vector of λ_1 (U_1) $\qquad U_1 = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$

$(S - \lambda_1 I) U_1 = 0$

where U_1 is the eigen vector of λ_1

$$\begin{bmatrix} 14-\lambda_1 & -11 \\ -11 & 23-\lambda_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = 0$$

$$\begin{bmatrix} (14-\lambda_1)u_1 - 11u_2 \\ -11u_1 + (23-\lambda_1)u_2 \end{bmatrix} = 0$$

Hence,

$(14-\lambda_1)u_1 - 11u_2 = 0 \qquad ---(1)$

$-11u_1 + (23-\lambda)u_2 = 0 \qquad ---(2)$

taking equation (1.)

$$\frac{u_1}{11} = \frac{u_2}{14-\lambda_1} = t$$

when $t = 1$

$$\boxed{u_1 = 11 \qquad u_2 = 14-\lambda_1}$$

Eigen vector of U_1 of $\lambda_1 = \begin{bmatrix} 11 \\ 14-\lambda_1 \end{bmatrix}$

$$= \begin{bmatrix} 11 \\ 14-30.3849 \end{bmatrix} = \begin{bmatrix} 11 \\ -16.3849 \end{bmatrix}$$

(iii) Normalise the eigen vector V_1

$$e_1 = \begin{bmatrix} \frac{11}{\sqrt{11^2 + (-16.38)^2}} \\ \frac{-16.38+9}{\sqrt{11^2 + (-16.38)^2}} \end{bmatrix} = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$$

Repeat the same steps for λ_2, we will get,

$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$$

Step-5:- Derive new data-set

	Ex1	Ex-2	Ex-3	Ex-4
First PC (PC1)	P_{11}	P_{12}	P_{13}	P_{14}

$$P_{11} = e_1^T \begin{bmatrix} 4-8 \\ 11-8.5 \end{bmatrix} \quad \text{:. T is transpose.}$$

$$= \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} -4 \\ 2.5 \end{bmatrix}$$

$$= -4.3052.$$

$$P_{12} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 8-8 \\ 4-8.5 \end{bmatrix}$$

$$= 3.7361$$

$$P_{13} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 13-8 \\ 5-8.5 \end{bmatrix}$$

$$= 5.6920$$

$$P_{14} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 7-8 \\ 14-8.5 \end{bmatrix}$$

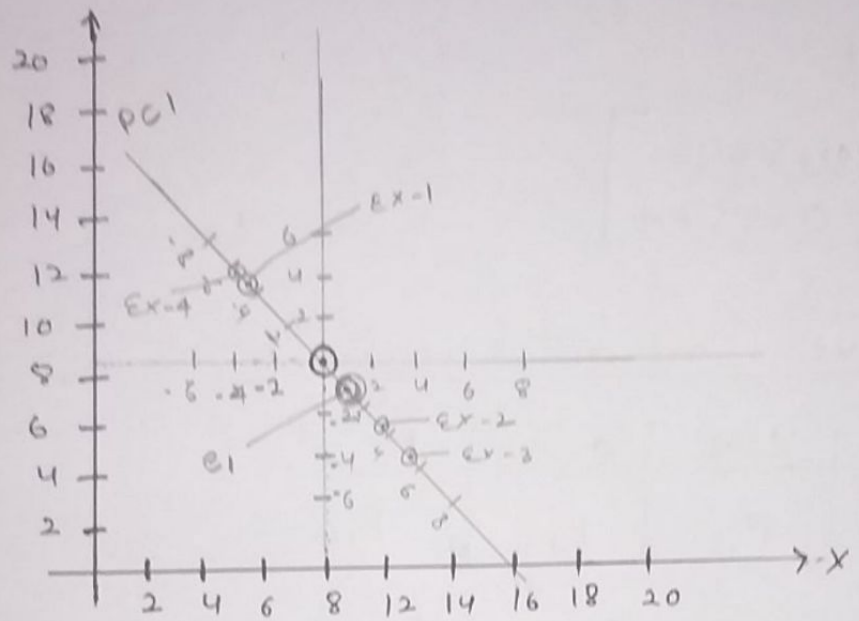
$$= -5.1238$$

So, new dataset will be :-

	Ex-1	Ex-2	Ex-3	Ex-4
PC1	-4.3052	3.7361	5.6928	-5.1238

Now, let's plat this dataset on a graph.

$(\bar{x}, \bar{y}) = (8, 8.5)$



plot mean value $(\bar{x}, \bar{y}) = (8, 8.5)$

There point will be new origin point

i.e (0, 0)

plot e1 (0.5574, -0.8303), Now join (0,0) and e1 to draw PC1,

Now plat the new dataset with dimension 1 on PC1.