SEM-2

Machine Learning for Big Data

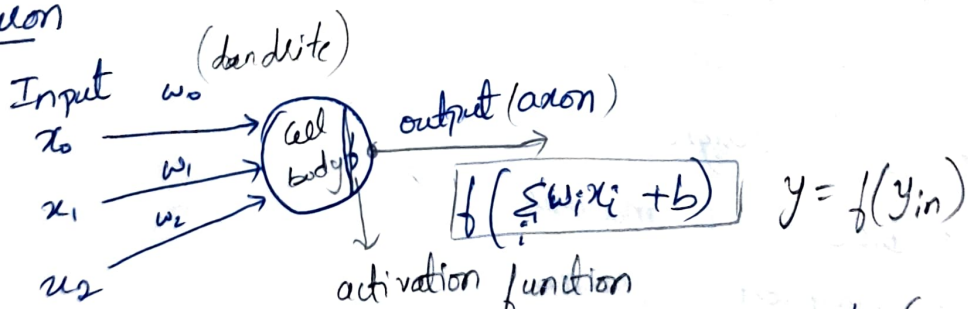
Topics

1. Artificial Neural Network
2. clustering
3. Support vector Machines & kernel methods
4. Deep Learning → ANN with extra hidden layer
5. Reinforcement Learning

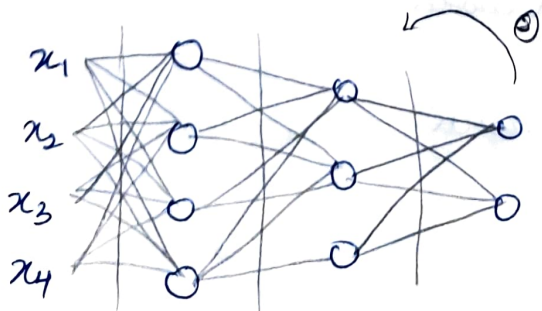
Linear → can be classified with single classifier

Non-linear → take the data to higher dimension and then classify

ANN → Neuron



$$y_{in} = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n + b \quad \text{(Net input)}$$



⊙ Recurrent Neural Network

→ Feed forward Neural network

(Backpropagetting) Neural Network is Feed Folward Neural Network

Back propogates the error.

What are all the different types of errors in ML algorithms?
How gradient descent approaches helps to reduce errors?
gradient descent is 'log-loss function'

Multiple classifiers for non linear data - Neural Network.

ANN applications - (previously matlab)

- pattern recognition - image analysis - shifted to neural network
- prediction
- optimization
- associative memory

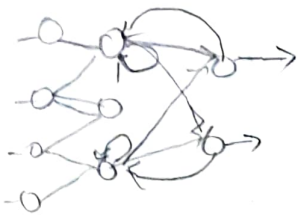
ANN in medicine -

- Image analysis
- Drug design
- Biochemical analysis
- Diagnostic systems

Network Architecture - 2 types

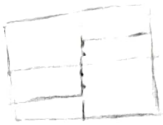
① Feed Folward Neural Network - Deep learning, Backpropagating Neural network

② Reurrent neural network



Weights - settings the values for weights - enable learning / training
weight is information to feed to neural network to train the model

Activation function - linear $f(x) = x$
non-linear (used in multilayer) $f(x) = \begin{cases} 1 & \text{if } f(x) \geq 0 \\ 0 & \text{if } f(x) < 0 \end{cases}$

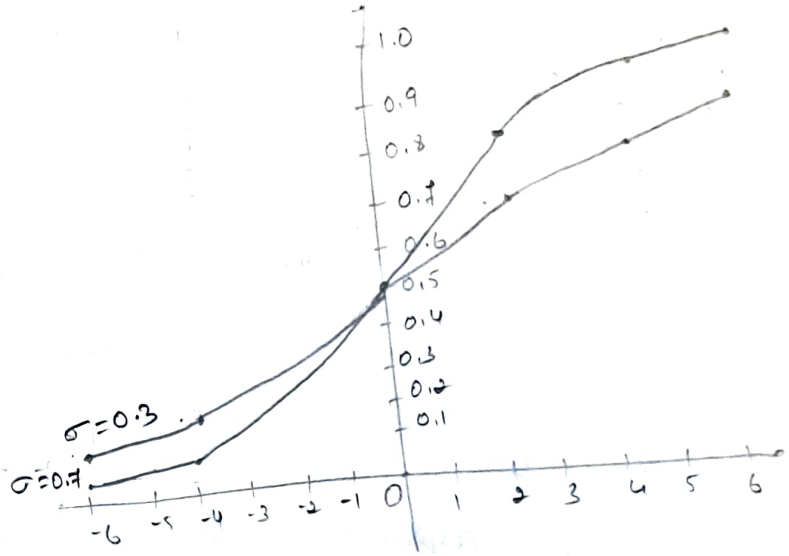


x	f(x)
-2	0
-1	0
0	1
2	1

* Sigmoid function

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

x	$f(x) = \frac{1}{1+e^{-\sigma x}}$	
	$\sigma = 0.3$	$\sigma = 0.7$
-6	0.14	0.014
-4	0.23	0.05
0	0.5	0.5
2	0.64	0.80
4	0.76	0.94
6	0.85	0.98



Bipolar sigmoid function

$$b(x) = 2\left(\frac{1}{1+e^{-\sigma x}}\right) - 1$$

$$b(x) = \frac{2}{1+e^{-\sigma x}} - 1$$

$$b(x) = \frac{2 - 1 - e^{-\sigma x}}{1 + e^{-\sigma x}}$$

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

$$f(x) = \begin{cases} 1 & \text{if } x \geq 1 \quad (0,1) \\ 0 & \text{if } x < 1 \end{cases} \qquad b(x) = 2f(x) - 1$$

vanishing gradient

sigmoid activation function has a problem of vanishing gradient after large no of inputs are sent to hidden layers.

tanh → non-linear & differentiable function similar to sigmoid function, but output values ranges from -1 to +1.

in deep nn, prefer relu for hidden layers as activation function.

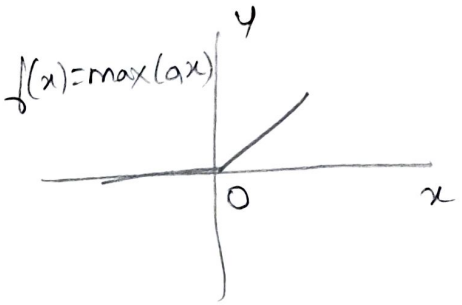
if multiple output then softmax

Relu (Rectified linear unit)

commonly used activation function in CNN. $f(x) = \max(0, x)$

x	$f(x) = \max(0, x)$
-100	0
-99	0
0	0
1	1
99	99
100	100

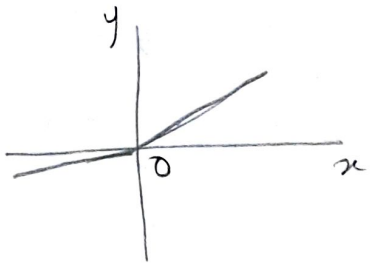
any value less than 0 then 0 and greater than 1 then x.



Leakey Relu

it is an improved version of relu function, instead of defining Relu function as 0 for x less than 0, we define it as a small linear component of x.

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



Softmax function

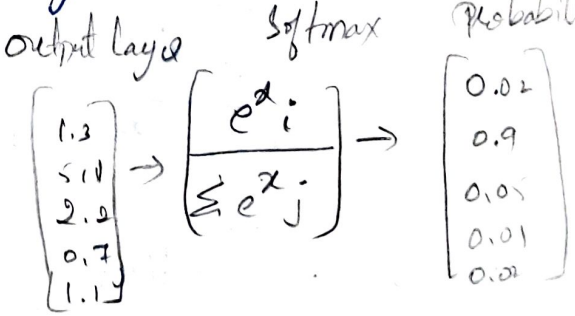
large no of outputs at the last layer to normalize the output.

softmax → probabilistic form

multiclass classification, probabilities sum is 1.

range is between 0 & 1.

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum e^{x_j}} \text{ for all } j$$



* Difference between pytorch & Tensorflow?

① McCulloch-pitts Neuron Model

The most fundamental unit of deep learning neural network is a perceptron.

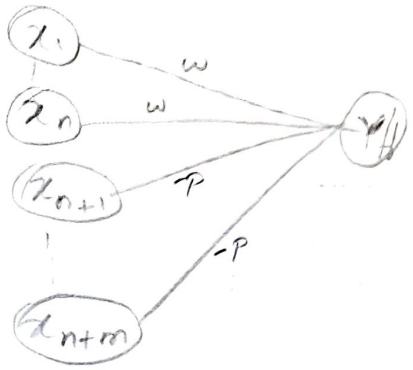
But the very first step towards the perceptron we use today was taken in 1943 by McCulloch and pitts, by mimicking the functionality of a biological neuron.

fixed weights = net input
⇒ activation function
⇒ output

M-P performs two tasks

① Task-1 : performs an aggregation of all inputs

② Task-2 : based on the aggregated values the M-P net makes the decision.



$$y_{in} = x_1 w_1 + x_2 w_2 + \dots + x_n w_n$$
$$= \sum_{i=1}^{n} x_i w_i$$

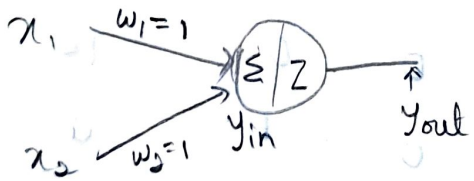
θ = threshold

$\theta = 1$

$$y_{out} = f(y_n) = \begin{cases} 1; & \text{if } f(y_{in}) \geq \theta \\ 0, & \text{if } f(y_{in}) < \theta \end{cases}$$

① Generate the output of logic AND function by McCulloch-pitts neuron model.

AND - 2 inputs



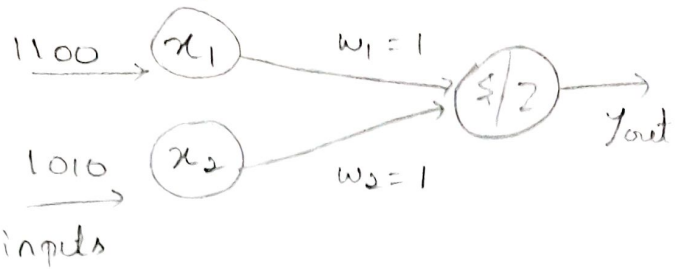
x_1	x_2	x_1 AND x_2	$y_{in} = x_1 + x_2$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	2

net input $y_{in} = x_1 w_1 + x_2 w_2 = x_1 + x_2$

Activation function $y_{out} = f(y_{in})$

activation $f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq 2 \\ 0 & \text{if } y_{in} < 2 \end{cases}$

OR function



x_1	x_2	x_1 or x_2	$y_{in} = x_1 + x_2$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	2

$$y_{in} = x_1 w_1 + x_2 w_2 = x_i w_i$$
$$= x_1 + x_2$$

Activation function $= y_{out} = f(y_{in}) = \begin{cases} 1 & \text{if } f(y_{in}) \geq 1 \\ 0 & \text{if } f(y_{in}) > 1 \end{cases}$

Assignment

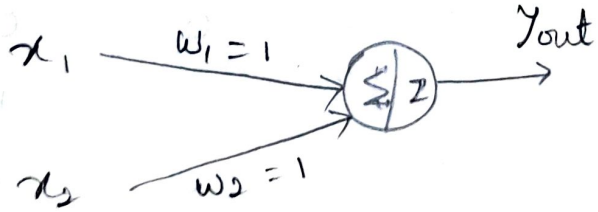
NOT function

x_1	x_1 NOT	$y_{in} = x w_1 + b$
0	1	0
1	0	1

$$y_{in} = x_1 w_1 + b$$

Activation function $= y_{out} = f(y_{in} = \begin{cases} 0 & \text{if } y_{in} \geq 1 \\ 1 & \text{if } y_{in} < 1 \end{cases}$

NAND function



x_1	x_2	NAND(x_1, x_2)	y_{in}
0	0	1	0
0	1	1	1
1	0	1	1
1	1	0	2

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

$w_1 = 1 \quad w_2 = 1 \quad b = 0 \quad \theta = 1$

Activation function $y_{out} = f(y_{in}) = \begin{cases} 0 & \text{if } y_{in} \geq 1 \\ 1 & \text{if } y_{in} < 1 \end{cases}$

NOR Function

x_1	x_2	$y_{in} = x_1w_1 + x_2w_2 + b$	$NOR(x_1, x_2)$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	2	0

$y_{in} = x_1w_1 + x_2w_2 + b$

$w_1 = 1 \quad w_2 = 1 \quad b = 0 \quad \theta = 1$

Activation function

$$y_{out} = f(y_{in}) = \begin{cases} 0 & \text{if } y_{in} \geq 1 \\ 1 & \text{if } y_{in} < 1 \end{cases}$$

Limitations of M-P Neuron

* Applies only to Boolean inputs
* we always need to hand code the threshold
* All inputs are holding equal priority, what if we want to assign more importance to single layer. & XOR doesn't work.

② Perceptron Networks

Frank Rosenblatt - 1962

Iterative weight adjustment (powerful technique) - learning rule

$\alpha t x_i$ → input

learning rate → target

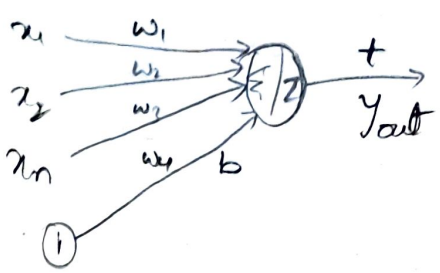
Purpose of training - to reduce error

Classification purpose

Types of perceptron Networks. - Single layer perceptron

Multilayer perceptron

(Back propagation neural network)



Bias is always to balance the network

- no error back propagating.

$$y_{in} = x_1w_1 + x_2w_2 + \cdots + x_nw_n + b$$

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

Steps

① Inputs

② initialize w_1, w_2, x_i, θ

activation $y_{out} = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq \theta \\ -1 & \text{if } y_{in} < \theta \end{cases}$

$y_{out} = t \rightarrow$ stop

update weights

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

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

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

update bias

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

$x_i = 1$

Example

- Develop a perception for AND function with bipolar inputs and targets. (initial weights are 0; learning rate = 1; and threshold = 1)

$w_1 = w_2 = b = 0$, $\alpha = 1$, $\theta = 1$

x_1	x_2	$t = x_1$ AND x_2	y_{out}
1	1	1	1
1	-1	-1	-1
-1	1	-1	-1
-1	-1	-1	-1

y_{in} = net input

$y_{in} = x_1w_1 + x_2w_2 + b$

Activation function

$$y_{out} = f(y_{in}) = \begin{cases} 1, & \text{if } y_{in} \geq 1 \\ -1, & \text{if } y_{in} < 1 \end{cases}$$

① $x_1 = 1$ $x_2 = 1$ $t = 1$

$w_1 = 0$ $w_2 = 0$ $b = 0$

$y_{in} = x_1w_1 + x_2w_2 + b$

$y_{in} = 0 + 0 + 0 = 0$

$y_{out} = f(y_{in}) = f(0) = -1$

target = 1 $\neq$ $y_{out} = -1$

update the weights

$w_1(n) = w_1(0) + \alpha t x_1$

$w_1(n) = 0 + 1 \times 1 \times 1 = 1$

$w_2(n) = w_2(0) + \alpha t x_2$

$w_2(n) = 0 + 1 \times 1 \times 1 = 1$

$b(n) = b(0) + \alpha t$

$= 0 + 1 \times 1 = 1$

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

$y_{in} = 1 \times 1 + 1 \times 1 + 1 = 3$

$y_{out} = f(y_{in}) = f(3) = 1$

(target = +1) == (y_{out} = 1)

Stop the learning

② Next pattern

$x_1 = 1 \quad x_2 = -1 \quad t = -1$

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

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

$y_{in} = 1 - 1 + 1 = 1$

$y_{out} = f(y_{in}) = f(1) = 1$

target = -1 ≠ y_{out} = 1

update weights

$w_1(n) = w_1(0) + \alpha t x_1$

$w_1(n) = 1 + 1 \times -1 \times 1 = 0$

$w_2(n) = w_2(0) + \alpha t x_2$

$w_2(n) = 1 + 1 \times -1 \times -1 = 2$

$b(n) = b(0) + \alpha t$

$b(n) = 1 + 1 \times -1 = 0$

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

$y_{in} = 0 + 2 + 0 = -2$

$y_{out} = f(y_{in}) = -1$

(target = -1) == (y_{out} = -1)

③ Next pattern

$x_1 = -1 \quad x_2 = 1 \quad t = -1$

$w_1 = 0 \quad w_2 = 2 \quad b = 0$

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

$y_{in} = 0 + 2 + 0 = 2$

$y_{out} = f(y_{in}) = 1$

target = -1 ≠ y_{out} = 1

update weights

$w_1(n) = w_1(0) + \alpha t x_1$

$= 0 + 1 \times -1 \times -1 = 1$

$w_2(n) = w_2(0) + \alpha t x_2$

$= 2 + 1 \times -1 \times 1 = 1$

$b(n) = b(0) + \alpha t$

$= 0 + 1 \times -1 = -1$

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

$$y_{in} = -1 + 1 - 1 = -1$$

check the activation $y_{out} = f(y_{in}) = -1$

$$(\text{target} = -1) == (y_{out} = -1)$$

— Stop

④ next pattern

$x_1 = -1 \quad x_2 = -1 \quad t = -1$
$w_1 = 1 \quad w_2 = 1 \quad b = -1$

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

$$y_{in} = -1 + 1 - 1 = -3$$

$$y_{out} = f(y_{in}) = -1$$

$$(\text{target} = -1) == (y_{out} = -1)$$

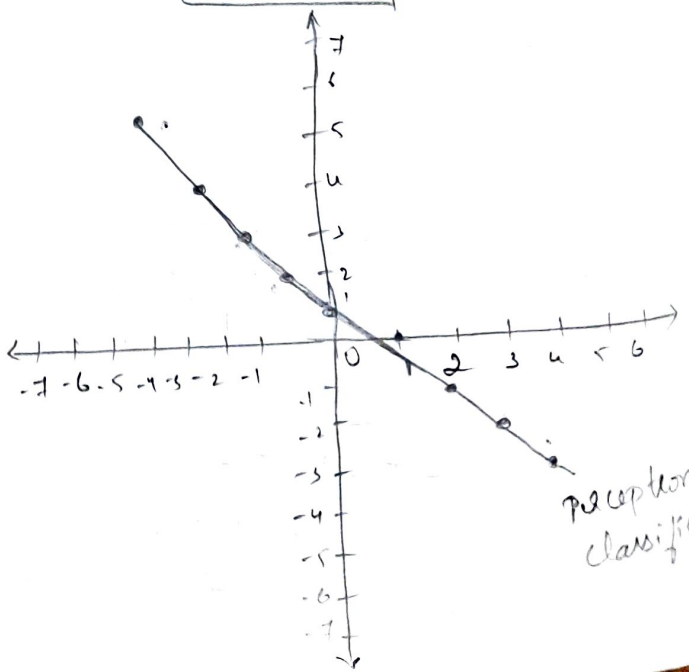
For the unseen class : $w_1 x_1 + w_2 x_2 + b = 0$

$w_1 = 1 \quad w_2 = 1 \quad b = -1$ final weights

$$x_1 + x_2 - 1 = 0$$

$$x_2 = 1 - x_1$$

x_1	$x_2 = 1 - x_1$
-4	$1-(-4) = 5$
-3	4
-2	3
-1	2
0	1
1	0
2	-1
3	-2
4	-3



- Develop a perceptron for OR function with bipolar inputs and targets. (initial weights are 0; learning rate = 1; and threshold = 1)

$w_1 = w_2 = b = 0$, $\alpha = 1$, $\theta = 1$

x_1	x_2	$t = x_1$ or x_2	y_{out}
1	1	1	1
1	-1	1	1
-1	1	1	1
-1	-1	-1	-1

Activation function

$$y_{out} = f(y_{in}) = \begin{cases} 1, & \text{if } y_{in} \geq 1 \\ -1, & \text{if } y_{in} < 1 \end{cases}$$

① First pattern

$x_1 = 1 \quad x_2 = 1 \quad t = 1$

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

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

$y_{in} = 0 + 0 + 0 = 0$

$y_{out} = f(y_{in}) = f(0) = -1$

target = 1 ≠ y_{out} = -1

update the weights

$w_1(n) = w_1(0) + \alpha t x_1$

$w_1(n) = 0 + 1 \times 1 \times 1 = 1$

$w_2(n) = w_2(0) + \alpha t x_2$

$w_2(n) = 0 + 1 \times 1 \times 1 = 1$

$b(n) = 0 + 1 \times 1 = 1$

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

$y_{in} = 1 \times 1 + 1 \times 1 + 0 = 2$

$\therefore y_{out} = f(y_{in}) = 1$

(target = 1) == (y_{out} = 1)

stop

② Next pattern

$x_1 = 1 \quad x_2 = -1 \quad t = 1$

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

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

$y_{in} = 1 + -1 \times 1 + 1 = 1$

$y_{out} = f(y_{in}) = f(1) = 1$

target = 1 == y_{out} = 1

③ next pattern

$x_1 = -1 \quad x_2 = 1 \quad t = 1$

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

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

$= -1 + 1 + 1$

$= 1$ ✓

(t = 1) == (y_{out} = 1)

④ $x_1 = -1 \quad x_2 = -1 \quad t = -1$

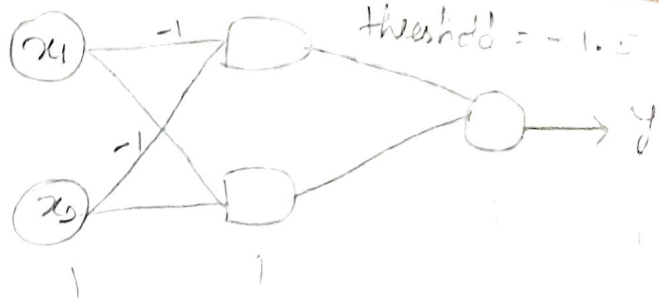
$w_1 = 1 \quad w_2 = 1 \quad b = 1$

$y_{in} = -1 - 1 + 1 = -1$

$f(-1) = -1$ ✓

(target = -1) == (y_{out} = -1)

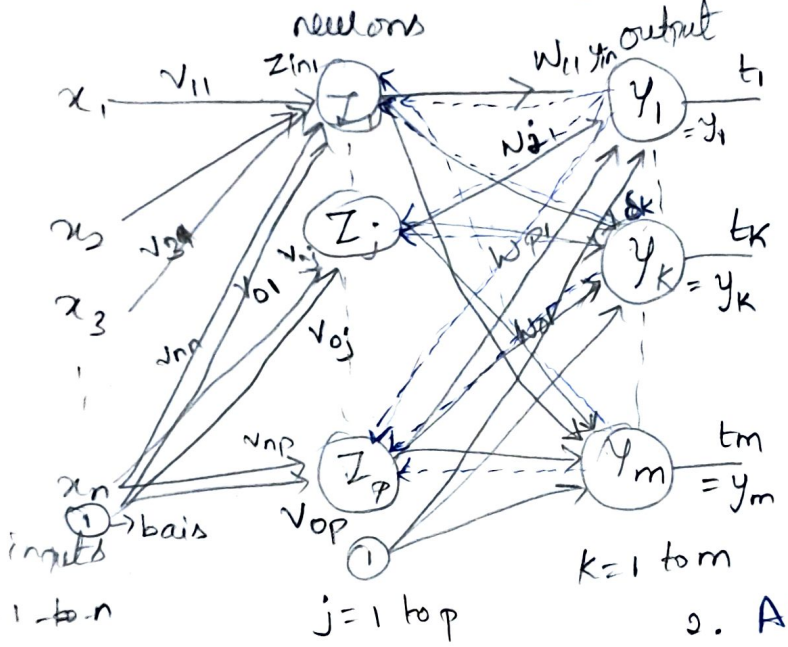
XOR Problem



Multilayer Networks

① Feed forward Networks
no feedbacks, output is calculated for every input to the net.
eg - Backpropagation nn
Extension of XOR, first is input - last is input, rest in between are hidden layer - deep nn

Back propagation Network (BPN)



① Feed Forward Step

1. net input at Z_1

$$Z_{in_1} = v_{01} + v_{11}x_1 + x_2 v_{21} + \cdots + x_n v_{n1}$$

$$Z_{in_j} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

2. Activation function at hidden layer

$$z_j = f(z_{inj}) = \frac{1}{1+e^{-z_{inj}}}$$

if sigmoidal

3. Feed forward

$$y_{in_1} = z_1 w_{11} + z_j w_{j1} + \cdots + z_p w_{p1} + w_{01}$$

$$y_{inK} = w_{0K} + \sum_{j=1}^{p} z_j w_{jK}$$

4. Activation

$$y_K = f(y_{inK}) = \frac{1}{1+e^{-y_{inK}}}$$

if sigmoidal

feed forward calculations is done by Keras. Sequential

Input is not neuron

Input = x_1, x_2, x_3

Input layer & hidden layer = Z_1, Z_j, Z_p

Output layer = $y_1, y_k \cdots y_m$

② Error back propogation step

$$\delta_1 = (t_1 - y_1) \cdot f'(y_{in_1}) \qquad \left[f(y_{in_1}) = f(y_{in_1})(1 - f(y_{in_1}))\right]$$

(t_1: Actual, y_1: predicted)

optimizer = adam

$$\delta_K = (t_K - y_K) \cdot f'(y_{in_K}) \qquad \left[f(y_{in_K})(1 - f(y_{in_K}))\right]$$

2. BP error to hidden layers Z

$$\delta_{inj} = \sum_{K=1}^{m} \delta_K \cdot w_{jK}$$

3. Error at hidden layer Z

$$\delta_j = \delta_{inj} \cdot f(Z_{inj})$$

Calculate error and send back to last layer

③ Update the weights

learning rate 0.3 to 0.5

$$v_{01(n)}^{new} = v_{01(0)}^{old} + \alpha \delta_j$$

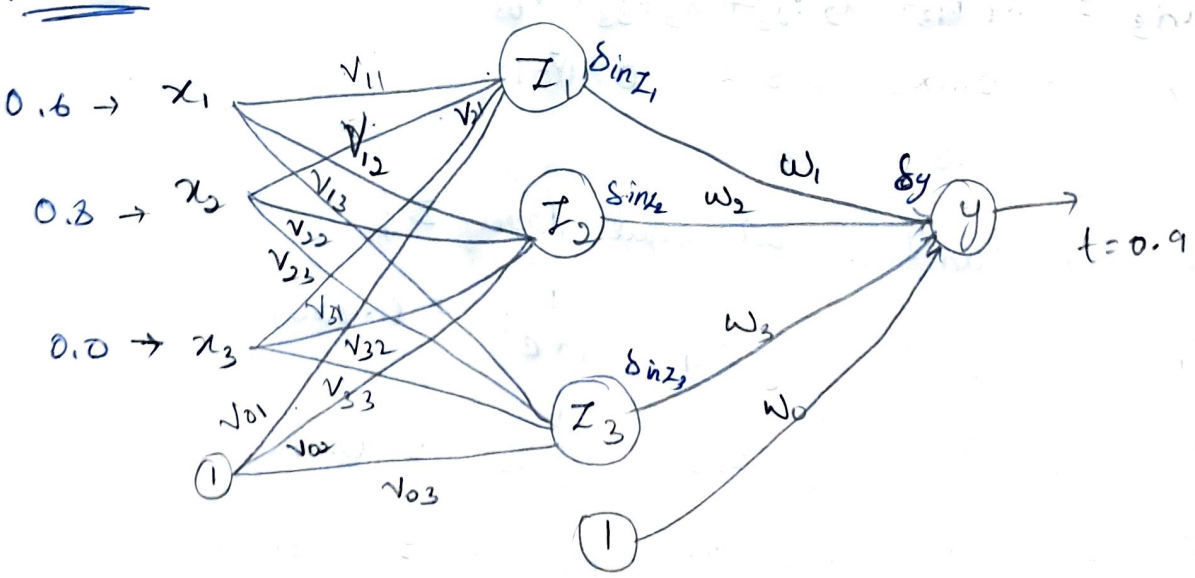
$$v_{0i(n)} = v_{0i(0)} + \alpha \delta_j x_1^{bais}$$

first layer $v_{ij(n)} = v_{ij(0)} + \alpha \delta_j x_i$

next layer $v_{0K(n)} = w_{0K(0)} + \alpha \delta_K$

$$v_{jK(n)} = w_{jK(0)} + \alpha \delta_K Z_j$$

Problem



$[w_1\ w_2\ w_3] = [-1\ \ 1\ \ 2]$

$w_0 = -1$ $\qquad \alpha = 0.3$

Activation = sigmoid

$$\begin{bmatrix} v_{01} & v_{02} & v_{03} \\ v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ 2 & 1 & 0 \\ 1 & 2 & 2 \\ 0 & 3 & 1 \end{bmatrix}$$

① feed forward network → net inputs / Activation - output

② Error - Back propagation

③ update weights

① FF N *@ net inputs at first layer / hidden layer / input layer

net input at $Z_1 \Rightarrow Z_{in_1}$

$Z_{in_1} = x_1 v_{11} + x_2 v_{21} + x_3 v_{31} + v_{01}$

$Z_{in_1} = (0.6 \times 2) + (0.8 \times 1) + (0 \times 0) + 0$

$Z_{in_1} = 2$

$Z_{in_2} = x_1 v_{12} + x_2 v_{22} + x_3 v_{32} + v_{02}$

$Z_{in_2} = (0.6 \times 1) + (0.8 \times 2) + (0 \times 3) + 0$

$Z_{in_2} = 2.2$

$Z_{in_3} = x_1 v_{13} + x_2 v_{23} + x_3 v_{33} + v_{03}$

$Z_{in_3} = (0.6 \times 0) + (0.8 \times 2) + (0 \times 1) + (-1)$

$Z_{in_3} = 0.6$

activation function at input layer Z;

$$Z_1 = f(Z_{in_1}) = \frac{1}{1+e^{-Z_{in_1}}} = \frac{1}{1+e^{-2}} = 0.8807$$

$$Z_2 = f(Z_{in_2}) = \frac{1}{1+e^{-Z_{in_2}}} = \frac{1}{1+e^{-2.2}} = 0.9002$$

$$Z_3 = f(Z_{in_3}) = \frac{1}{1+e^{-Z_{in_3}}} = \frac{1}{1+e^{-0.6}} = 0.6456$$

net input at y = $y_{in} = z_1 w_1 + z_2 w_2 + z_3 w_3 + w_0$

$y_{in} = 0.8803 \times -1 + 0.9002 \times 1 + 0.6596 \times 2 - 1$

$y_{in} = 0.3106$

$y_{out} = f(y_{in}) = \frac{1}{1+e^{-y_{in}}} = \frac{1}{1+e^{-0.3106}} = 0.577$

$t = 0.9 \neq y_{out} = 0.577$

② Error at output layer y

$\delta y = (t - y_{out}) \cdot f'(y_{in})$

$\delta y = (t - y_{out})\, f(y_{in})(1 - f(y_{in}))$

$\delta y = (0.9 - 0.577)(0.577)(1 - 0.577) = 0.078$

Error inputs to input layer z (back propagating)

$\delta in z_1 = \delta y \times w_1 = 0.078 \times -1 = -0.078$

$\delta in z_2 = \delta y \times w_2 = 0.078 \times 1 = 0.078$

$\delta in z_3 = \delta y \times w_3 = 0.078 \times 2 = 0.156$

Error at input layer after recieving error inputs

$\delta z_1 = \delta in z_1 f'(z_{in_1})$

$= \delta in z_1 \times f(z_{in_1}) \times (1 - f(z_{in_1}))$

$\delta z_1 = -0.078 \times 0.8807 \times (1 - 0.8807)$

$\delta z_1 = -0.00819$

$\delta z_2 = \delta in z_2 \times f'(z_{in_2})$

$= 0.078 \times 0.9002 \times (1 - 0.9002) = 0.007$

$\delta z_3 = \delta in z_3 \times f'(z_{in_3}) = 0.0356$

③ Update the weights → from last layer to first

$w_{0(n)} = w_{0(0)} + \alpha \delta y \quad (\text{bias} = 1) = -1 + 0.3 \times 0.078 = -0.9766$

$w_1(n) = w_0(n) + \alpha \delta y z_1 = -1 + 0.3 \times 0.078 \times 0.8808 = -0.9$

$w_2(n) = w_2(0) + \alpha \delta y z_2 = 1 + 0.3 \times 0.078 \times 0.9002 = 1.021$

$w_3(n) = w_3(0) + \alpha \delta y z_3 = 2 + 0.3 \times 0.078 \times 0.6456 = 2.015$

input weight updation

$v_{01}(n) = v_{01}(0) + \alpha \delta_{z1} = 0 + 0.3 \times (-0.00819) = 0.00245$

$v_{02}(n) = v_{02}(0) + \alpha \delta z_2 = 0 + 0.3 \times (0.007) = 0.0021$

$v_{03}(n) = v_{03}(0) + \alpha \delta z_3 = -1 + 0.3 \times (0.0356) = -0.9893$

$v_{11}(n) = v_{11}(0) + \alpha \delta z_1 x_1 = 2 + (0.3 \times (-0.00819) \times 0.6) = 1.998$

$v_{21}(n) = v_{21}(0) + \alpha \delta z_1 x_2 = 1 + (0.3 \times (-0.00819) \times 0.8) = 0.9980$

$v_{31}(n) = v_{31}(0) + \alpha \delta z_1 x_3 = 0 + (0.3 \times (-0.00819) \times 0) = 0$

$v_{12}(n) = v_{12}(n) + \alpha \delta z_2 x_1 = 1 + (0.3 \times (+0.007) \times 0.6) = 1.0012$

$v_{22}(n) = v_{22}(n) + \alpha \delta z_2 x_2 = 2 + (0.3 \times (0.007) \times 0.8) = 2.00168$

$v_{32}(n) = v_{32}(n) + \alpha \delta z_2 x_3 = 3 + (0.3 \times (0.007) \times 0) = 3$

$v_{13}(n) = v_{13}(0) + \alpha \delta z_3 x_1 = 0 + (0.3 \times (0.0356) \times 0.6) = 0.0064$

$v_{23}(n) = v_{23}(0) + \alpha \delta z_3 x_2 = 2 + (0.3 \times (0.0356) \times 0.8) = 2.008$

$v_{33}(n) = v_{33}(0) + \alpha \delta z_3 x_3 = 1 + (0.3 \times (0.0356) \times 0.0) = 1$

Clustering in machine learning

In real world, not every data we work upon has a target variable. popular under unsupervised learning is clustering Analysis. Goal is to group similar data points into a a group.

Clustering - Organization of unlabeled data into similarity groups called clusters. Data items within a cluster are 'similar'. Data items between clusters are 'dissimilar'

↓ Unsupervised learning

Clustering algorithms

dimension reduction algorithm → colums reducing

generative algorithms → data points (rows)

Example - Email dataset, when there is no lable for spam/ham then clustering.

applications - market segmentation, genetics, medical imaging, video recommendations

popular clustering algorithms

① K-means - fast (euclidean distance approach)

② Hierarchical

③ Density based spatial clustering (DBSCAN)

④ Gaussian mixture models - soft clustering

Types of clustering - ① Hard clustering → each data point belongs to a cluster completely or not.

eg - if 4 data points, must have 2 clusters

one data point = one cluster

② Soft clustering → instead of putting each data point into seperate cluster, probability or likelihood of data to be clustered is assigned for each data point

Distance measures used in clustering → find dissimilarity between two data points

1. Minkowski distance family
   - Euclidean distance (clustering) $= \sqrt{\sum (q_i - p_i)^2}$
   - Manhattan distance $= \sum_{i=1}^{d} |x_{i1} - x_{j1}|$
2. Cosine distance (text) → cosine of the angle between two vectors in multidimensional - high dimensional data
3. Mahalanobis distance

$$\cos\theta = \frac{\text{dot product}}{|a| \times |b|} = \frac{a.b}{\|a\| \times \|b\|}$$

K-Means Clustering

MacQueen, 1967

partitional clustering algorithm - dividing a dataset into distinct groups or clusters. each cluster has centre called centroid.

partitioning - based
nonoverlapping groups

K-means algorithm

Step-1 : choose the number of clusters (K) using elbow method

Step-2 : select at random K points, the centroids (not necessarily from your dataset)

Step-3: Assign each data point to the closest centroid → that forms K clusters

Step 4 : Compute and place the new centroid of each cluster.

Step 5 : Reassign each data points to the new closest centroid if any reassignment took place, go to step 4, otherwise stop.

Example

Determine which medicine belong to cluster 1 and which medicine to other cluster

Objects	Attribute-1 index weight	Attribute 2 PH
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4

① $K=2$ $C_1(1,1)$ $C_2(2,1)$

		$C_1(1,1)$	$C_2(2,1)$	smaller Compare C_1 & C_2 values
A	(1,1)	$\sqrt{(1-1)^2+(1-1)^2} = 0$	$\sqrt{(2-1)^2+(1-1)^2} = 1$	C_1
B	(2,1)	$\sqrt{(1-2)^2+(1-1)^2} = 1$	$\sqrt{(2-2)^2+(1-1)^2} = 0$	C_2
C	(4,3)	$\sqrt{(1-4)^2+(3-1)^2} = 3.6$	$\sqrt{(2-4)^2+(3-1)^2} = 2.82$	C_2
D	(5,4)	$\sqrt{(5-1)^2+(4-1)^2} = 5$	$\sqrt{(5-2)^2+(4-1)^2} = 4.24$	C_2

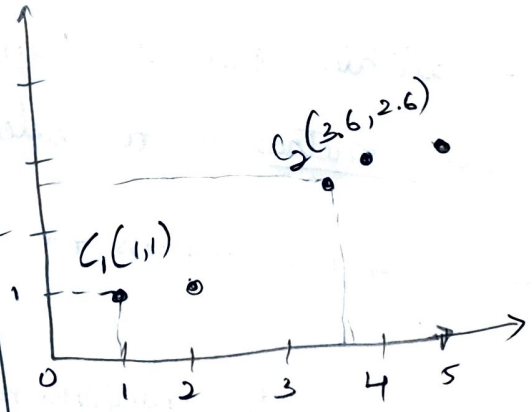
$C_1 = (1,1)$

$C_2 = (2,1), (4,3), (5,4)$

$C_2 = \left(\frac{2+4+5}{3}, \frac{1+3+4}{3}\right) = (3.6, 2.6)$

C_1 - remains same

②	$C_2(3.6, 2.6)$	smaller Compare
A	$\sqrt{(3.6-1)^2+(2.6-1)^2} = 3.0$	C_1
B	$\sqrt{(3.6-2)^2+(2.6-1)^2} = 2.26$	C_1
C	$\sqrt{(3.6-4)^2+(2.6-3)^2} = 0.56$	C_2
D	$\sqrt{(3.6-5)^2+(2.6-4)^2} = 1.97$	C_2



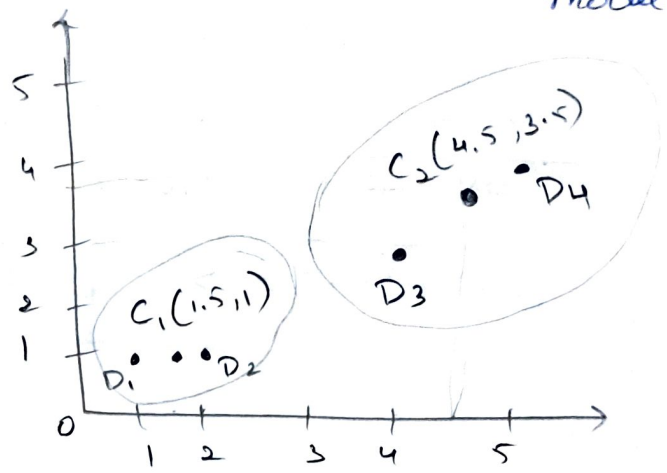
$C_1 = (1,1) + (2,1) = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1.5, 1)$

$C_2 = (4,3) + (5,4) = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4.5, 3.5)$

(3)

		$C_1(1.5, 1)$	$C_2(4.5, 3.5)$	Compute
A	(1,1)	$\sqrt{(1.5-1)^2+(1-1)^2} = 0.5$	$\sqrt{(4.5-1)^2+(3.5-1)^2} = 4.3$	C_1
B	(2,1)	$\sqrt{(2-1.5)^2+(1-1)^2} = 0.5$	$\sqrt{(4.5-2)^2+(3.5-1)^2} = 3.5$	C_1
C	(4,3)	$\sqrt{(4-1.5)^2+(3-1)^2} = 3.2$	$\sqrt{(4.5-4)^2+(3.5-3)^2} = 0.7$	C_2
D	(5,4)	$\sqrt{(5-1.5)^2+(4-1)^2} = 4.6$	$\sqrt{(5-4.5)^2+(4-3.5)^2} = 0.7$	C_2

Clusters & centroid's not change so stoping & model is ready.



K - means clustering :-

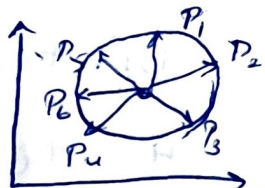
WCSS - within cluster sum of square - Inertia

Using elbow method - to select right number of clusters

K-means → K-means ++ . To avoid effect of bad centroid selection.

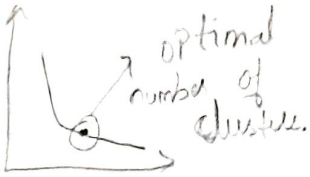
* WCSS → one with minimum value, when one placed to correct place. init = 10 - no. of locations
max_ita - for each point.

$\sum_{P=1}^{n=5} (P_i, C)^2$ = ? distance

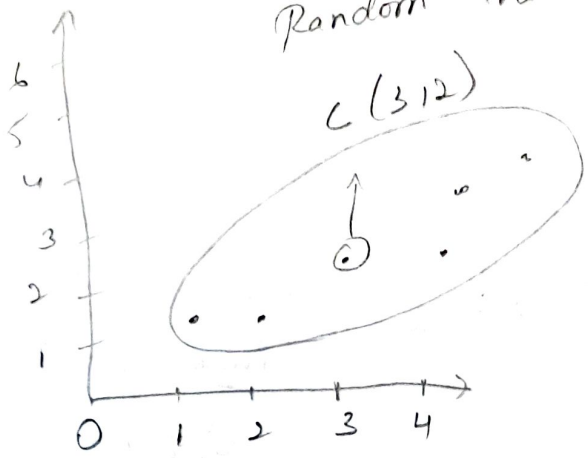


Kmeans = Kmeans (n_clusters = 1, init = 'K-means++', max_item=300) n_init=10, random=0)

Steps ① finding out right place / location of cluster
② finding right number of clusters.



Random initialization of centroids



Data points	C (3,2)	C (1,1)	C (5,3)
(1,1)	$(3-1)^2+(2-1)^2+$	$(1-1)^2+(1-1)^2$	$(5-1)^2+(1-1)^2+$
(2,1)	$(3-2)^2+(2-1)^2+$	$+(1-2)^2+(1-1)^2$	$(5-2)^2+(3-1)^2+$
(4,3)	$(3-4)^2+(2-3)^2+$	$+(1-4)^2+(1-3)^2$	$(5-4)^2+(3-3)^2+$
(5,4)	$(5-3)^2+(2-4)^2$	$+(1-5)^2+(1-4)^2$	$(5-5)^2+(3-4)^2=$
	$=17$	$=39$	$=35$

$dc_1 = (3-1)^2+(2-1)^2$

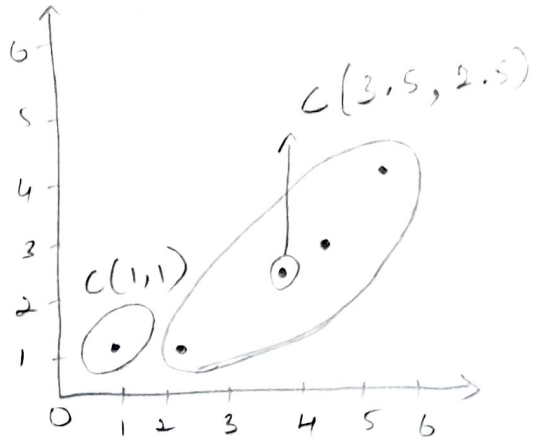
$dc_2 = 39$

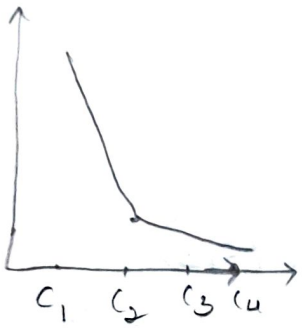
$dc_3 = 35.$

DPS	C_1 (1,1)	C_2 = (3.5, 2.5)
(1,1)	$(1-1)^2+(1-1)^2$	WCSS =
(2,1)	$=0$	$(3.5-2)^2+(2.5-1)^2$
(4,3)		$+(3.5-4)^2+(2.5-3)^2$
(5,4)		$+(3.5-5)^2+(2.5-4)^2$
		$=9.5$

Eg:

DPs	C(1.5,1)	C(4.5,3.5)
(1,1)		$(4.5-1)^2+$
(2,1)		$(3.5-1)^2+$
(4,3)		$(4.5-4)^2+$
(5,4)		$(3.5-3)^2+$
		$(4.5-5)^2+$
		$(3.5-4)^2 = 9.5$





→ C_2 has elbow

Strengths of k-means - easy to understand &

Time complexity : $O(tkn)$

n - no. of data points

k - no. of clusters

t - no. of iterations

k-means most popular clustering algorithm.

Since both k and t are small. k-means is considered a linear algorithm.

Problem faced

outliers - Handling with outliers

Remove some data points that are much farther away from

```
import matplotlib.pyplot as plt
from sklearn.cluster import Kmeans
```

Create arrays that resemble 2 variables in a data set.

```
x =
y =
```

turn the data into a set of points

```
data = list(zip(x, y))
```

Hierarchical Clustering

Find successive clusters using previously established clusters.

2 types → Agglomerative (bottom-up)
Divisive (top-down)

Agglomerative Algorithm - Begin with each data element as a seperate cluster and merge them into successively larger clusters

implement

Divisive Algorithm - Begin with the whole set and proceed to divide it into successive small clusters

applications same as K-means

Agglomerative algorithm

① Step 1 → Make each point a single point cluster → that forms N clusters.

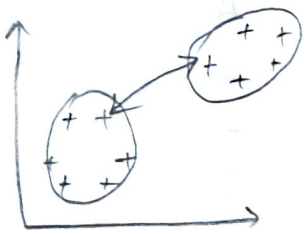
② Step 2 → take two closest data points

③ Step 3 → take closest clusters (Euclidean or manhattan) and make them one cluster

④ Step 4 → Repeat step 3 until there is only one cluster

⇓

Finish



Distance between clusters

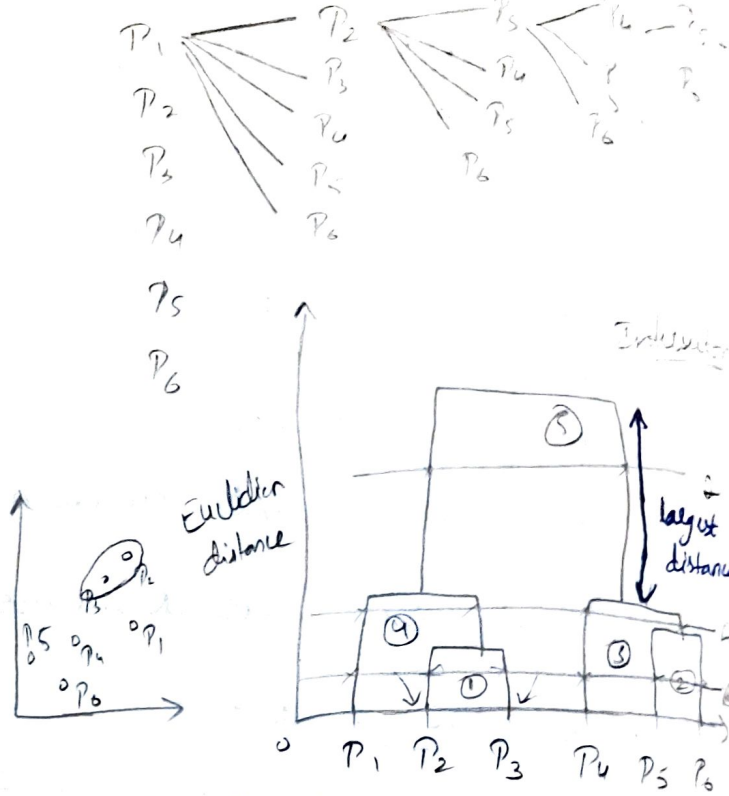
* option 1 : closest points
  Min (single) Linkage - WARD → within cluster variance
* option 2 : Furthest points
  Max (complete) Linkage
* option 3 : Average Distance
  Average Linkage
* option 4 : Distance b/w centroids of clusters
  centroid of WARD method

Dendrograms - How dendrograms work

	P_1	P_2	P_3	P_4	P_5	P_6
P_1	0					
P_2		0				
P_3	0.5		0			
P_4				0		
P_5					0	
P_6					0.5	0

(P_2 P_3) ①

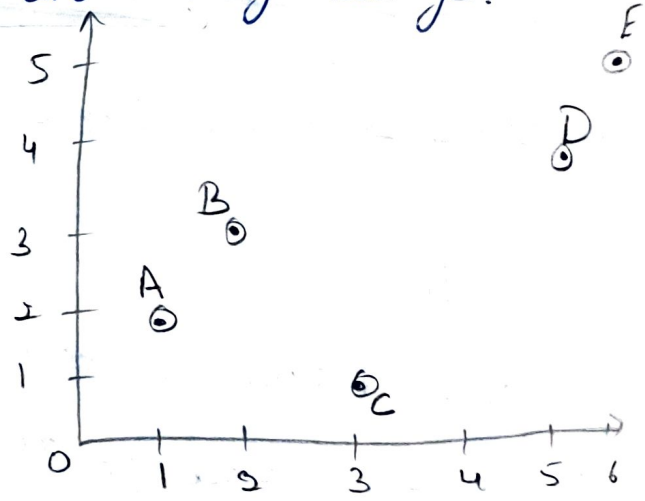
②



Example : Distance Between matrics used in hierarchical clustering.

- Single Linkage (min), complete Linkage (max) and Average Linkage.

Point	x	y
A	1	2
B	2	3
C	3	1
D	5	4
E	6	5



① Single / Minimum Linkage

Euclidean distance

$d(AB)$	$\sqrt{(2-1)^2+(3-2)^2} = \sqrt{2} = 1.41$
$d(AC)$	$\sqrt{(3-1)^2+(1-2)^2} = \sqrt{5} = 2.23$
$d(AD)$	$\sqrt{(5-1)^2+(4-2)^2} = \sqrt{20} = 4.47$
$d(AE)$	$\sqrt{(6-1)^2+(5-2)^2} = \sqrt{34} = 5.83$
$d(BC)$	$\sqrt{(3-2)^2+(3-1)^2} = \sqrt{5} = 2.23$
$d(BD)$	$\sqrt{(2-5)^2+(3-4)^2} = \sqrt{10} = 3.16$
$d(BE)$	$\sqrt{(2-6)^2+(3-5)^2} = \sqrt{20} = 4.47$
$d(CD)$	$\sqrt{(3-5)^2+(4-1)^2} = \sqrt{13} = 3.61$
$d(CE)$	$\sqrt{(3-6)^2+(5-1)^2} = \sqrt{25} = 5$
$d(DE)$	$\sqrt{(5-6)^2+(4-5)^2} = \sqrt{2} = 1.41$

	A	B	C	D	E
A	0				
B	1.41	0			
C	2.23	2.23	0		
D	4.47	3.16	3.61	0	
E	5.83	4.47	5	1.41	0

① Initial clusters = {A, B, C, D, E}

② min A and B (distance = 1.41)

③ merge A & B

{(AB), C, D, E} = new clusters

$d\{(AB), C\} = \min\{d(AC), d(BC)\}$

$= \min\{2.23, 2.23\}$

$d\{(AB), C\} = 2.23$

$d\{(AB), D\} = \min\{d(AD), d(BD)\}$

$= \min(4.47, 3.16)$

$d\{(AB), D\} = 3.16$

$d\{(AB), E\} = \min\{d(AE), d(BE)\}$

$= \min\{5.83, 4.47\}$

$d\{(AB), E\} = 4.47$

	AB	C	D	E
AB	2.23			
C	2.23	0		
D	3.16	3.61	0	
E	4.47	5	1.41	0

③ Minimum distance is between D and E

d is {1.41}

Merge {D and E} new clusters = {(AB), C, (DE)}

$d\{(AB)(DE)\}$

$= \min\{d(AD), d(AE), d(BD), d(BE)\}$

$= \min\{4.47, 5.83, 3.16, 4.47\}$

$= 3.16$

	AB	C	DE
AB	0		
C	2.23	0	
DE	3.16	3.61	0

$d\{(C)(DE)\}$

$= \min\{d(CD), d(CE)\}$

$= \min\{3.61, 5\}$

$= 3.61$

$d\{(ABC)(DE)\}$

$= \min(d(AD), d(AE), d(BD), d(BE), d(CD), d(CE)\}$

$= \min(4.47, 5.83, 3.16, 4.47, 3.61, 5\}$

$= \min(3.16)$

	ABC	DE
ABC	0	
DE	3.16	0

clusters

① Initial clusters = {A, B, C, D, E}

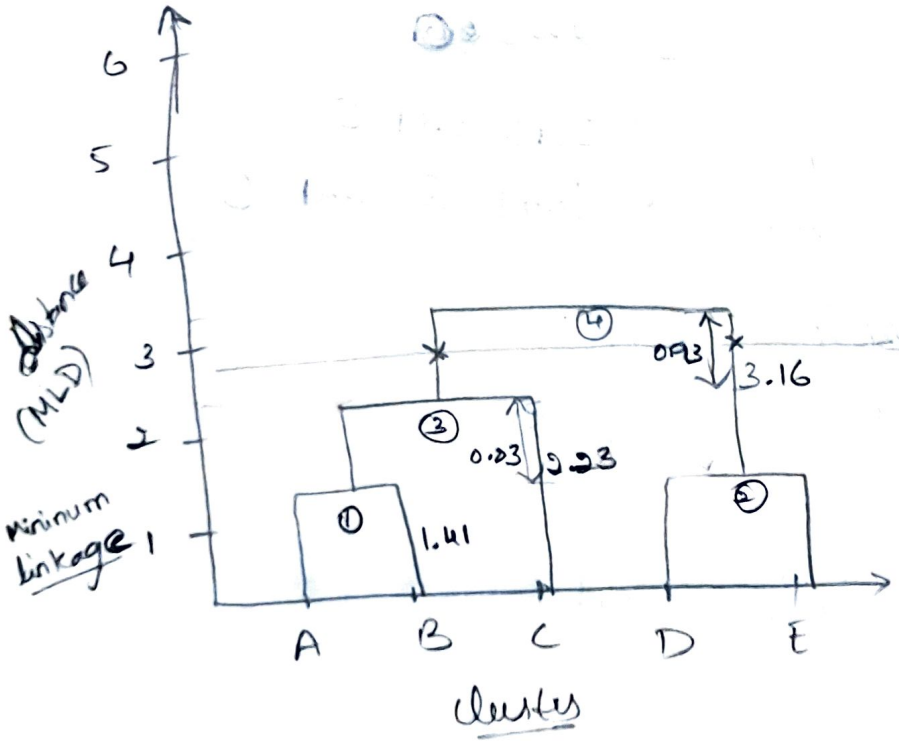
② A and B dist = 1.41

③ D & E dist = 1.41

④ (AB) & (C) dist = 2.23

⑤ (ABC) & (DE) dist = 3.16

Dendograms



$3.16 - 2.23 = 0.93$ ✓

$2.23 - 1.41 = 0.82$

= 2 clusters

② Complete linkage (maximum linkage)

(i) first take minimum $\min(AB) = 1.41$

initial clusters = {(AB), C, D, E}

(ii) $d(AB), C) = \max(d(AC), d(BC))$

$= \max(2.23, 2.83)$

$= 2.83$

$d(AB, D) = \max\{d(AD), d(BD)\}$

$= \max(4.47, 3.16) = 4.47$

$d(AB, E) = \max\{d(AE), d(BE)\}$

$= \max\{5.33, 4.47\} = 5.33$

	AB	C	D	E
AB	0			
C	2.23	0		
D	4.47	3.61	0	
E	5.33	5	1.41	0

$d(AB, DE) = \max\{d(AD), d(AE), d(BD), d(BE)\}$

$= 5.33$

$d(DE, C) = \max\{d(DC), d(EC)\} = 5$

	AB	C	DE
AB	0		
C	2.23	0	
DE	5.33	5	0

clusters

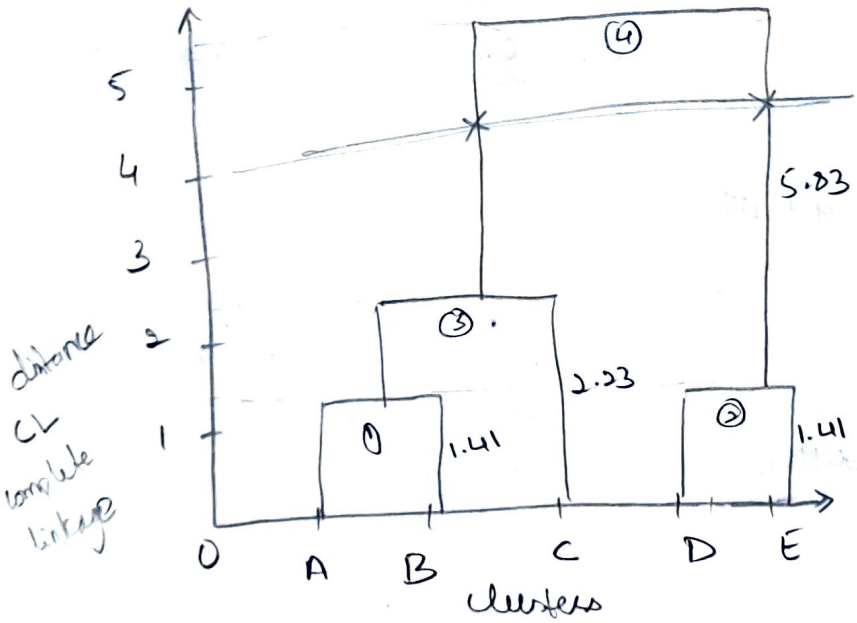
A & B = 1.41

D & E = 1.41

AB & C = 2.23

ABC & DE = 5.33

	ABC	DE
ABC	0	
DE	5.33	0



$5.83 - 2.23 = 3.59$ ✓

$2.23 - 1.41 = 0.82$

= 2 clusters

Hierarchical Clustering

Use case: targetting audience will form cluster.

if can belong to multiple clusters (soft clustering) and send advertising.

Expectation - Maximization Algorithm

Semi supervised learning - if features are observable then learning
if features are unobservable then predict.

↓ finds

local maximum likelihood of parameters, where latent parameters are observed (no labels)

Algorithm

① Expectation step (E - step) : algorithm computes the latent variable expectation of the log - likelihood using the current parameter estimates.

② maximization step (M - step) :

With Label

Example : C_1 & C_2 of getting head with C_1, θ_1 be probability θ_2 be head from C_2

$\theta_1 = \frac{\text{no of heads with } C_1}{\text{Total no of flips using } C_1}$ $\theta_2 = \frac{\text{no of heads with } C_2}{\text{Total no of flips using } C_2}$

					Coin A	Coin B
B	H	_	_	_		5H 5T
A	H	_	_	_	9H, 1T	
A	H	_	_	_	8H, 2T	
B	H	_	_	_		4H, 6T
A	T	_	_	_	7H, 3T	
					24H, 6T	9H, 11T

$\theta_1 = \frac{24}{24+6}$ $\theta_2 = \frac{9}{9+11}$

$\theta_1 = 0.8$ $\theta_2 = 0.45$

Without label

⇒ EM clustering algorithm

H	T	T	H	H	T	H	T
H	H	H	T	H	H	H	H
H	T	H	H	H	H	T	H
H	T	H	H	T	H	H	T
T	H	H	H	H	H	H	T

① Initial values of $\theta_A^{(0)} = 0.6$ $\theta_B^{(0)} = 0.5$

② Expectation - step $P(A) = \frac{L(A)}{L(A)+L(B)}$

K = no of heads

n = total no of flaker

$L(B) = \theta_B^K (1-\theta_B)^{n-K}$ $L(A) = \theta_A^K (1-\theta_A)^{n-K}$

$L(B) = (0.5)^6 (1-0.5)^{10-5}$ $L(A) = (0.6)^5 (1-0.6)^{10-5}$

$L(B) = 0.0009765$ $L(A) = 0.0007962$

$P(A) = \frac{0.0007962}{0.0007962+0.0009765}$ $P(B) = \frac{0.0009762}{(0.0007962)+0.0009765}$

$P(A) = 0.4491 = 0.45$ $P(B) = 0.55$

①

Coin A - H	T
P(A) X 5	P(A) X 5
0.45 X 5 = 2.2 L(H)	0.45 X 5 = 2.2 L(T)

Coin B - H	T
P(B) X 5	P(B) X 5
0.55 X 5 = 2.75 L(H)	0.55 X 5 = 2.75 L(T)

Usage of EM algorithm - It can be used to fill the missing data in a sample

- It can be used for the purpose of estimating the parameters of Hidden Markov Model (HMM)

- It can be used for discovering the values of latent variables.

Advantages - likelihood will increase with each iteration.
E-step and M-step are often easy to implementation

disadvantages - slow convergence.

MSE = min
MSE = Wt high

Clustering Metrics in Machine Learning

Linear regression - mean square error → performance measure
Root mean square error
square error

} supervised learning

Linear classification - confusion matrix
ROC - Receiver operating characteristics

Clustering (quality of the clusters)

↳ quantitative metrics

① Silhouette score
② Davis-Bouldin Index
③ Calinski-Harabasz Index

Silhouette analysis - study the seperation distance between the resulting clusters.

↳ how close each point in one cluster

+1 = sample is far away from the neighbouring clusters
0 = not properly grouped (overlapping), can be further divided.
-1 = assigned data points to array clusters

Calculating Silhouette coefficient

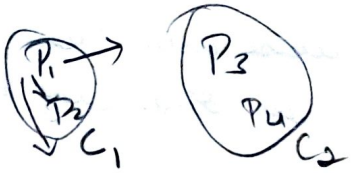
If no distance given - calculate using Euclidean distance

P_1 and P_2 are in cluster 1
P_3 and P_4 are in cluster 2

Point	cluster label
P_1	1
P_2	1
P_3	2
P_4	2

Distance matrix

Point	P_1	P_2	P_3	P_4
P_1	0	0.1	0.65	0.55
P_2	0.1	0	0.7	0.6
P_3	0.65	0.7	0	0.3
P_4	0.55	0.6	0.3	0



$$SC = 1 - \frac{a}{b}$$

① **Point P_1** - within cluster

$a = Avg[P_1 \to P_2]$

$a = 0.1$

from one point to all other clusters

$b = Avg[(P_1 \to P_3), (P_1 \to P_4)]$

$b = \frac{0.65 + 0.55}{2} = 0.6$

Silhouette score

SC for point $P_1 = 1 - \frac{a}{b}$

$= 1 - \frac{0.1}{0.6}$

$= 0.833$

② **Point P_2**

$a = Avg[(P_2 \to P_1)]$

$a = 0.1$

$b = Avg([P_2 \to P_3), (P_2 \to P_4)]$

$b = 0.65 \quad \left(\frac{0.7 + 0.6}{2}\right)$

SC for point $P_2 = 0.8461$

③ Point P_3

$a = Avg(P_3 \to P_4) = 0.3$

$b = Avg[(P_3 \to P_1)(P_3 \to P_2)] = \frac{0.65 + 0.7}{2} = 0.675$

$SC = 1 - (0.3 / 0.675) = 0.55$

highly coupled (0.83 & 0.84)

Avg SC for $C_1 = \frac{P_1 + P_2}{2} = 0.8395$

loosely coupled

Avg SC for $C_2 = \frac{P_3 + P_4}{2} = 0.5171$

Average of overall cluster $= \frac{0.8395 + 0.5171}{2} = 0.6783$

④ Point P_4

$a = Avg(P_4 \to P_3) = 0.3$

$b = Avg((P_4 \to P_1)(P_4 \to P_2)) = \frac{0.55 + 0.6}{2} = 0.575$

$SC = 1 - (0.3 / 0.575)$

$SC = 0.4782$

SVM (Support Vector Machine)

* Maximum margin linear separators
* Finding maximum margin separators
* Kernels methods for non-linear functions
* Varying length pattern classification using SVM.

SVM is a supervised learning model.

Approach 1 - create a label using keywords, for instance "urgent", "complaint", "help". → drawback mix, hard to implement rules.

Approach 2 - Use a supervised machine learning algorithm label classification. Logistic Regression

SVM is used for classification (DT, NBC, XGBoost, Random forest, ANN, ANN)

SVR (Support vector Regression) is used for regression.

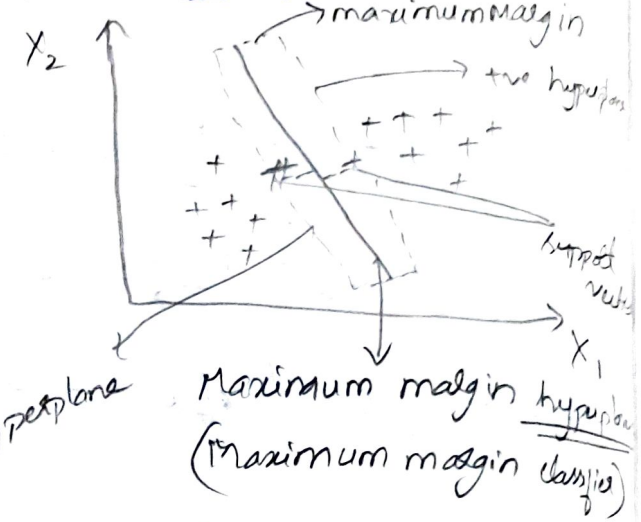
Vapnik and Chervonenkis → "Maximal margin classifier" → SVM was later also developed "kernel trick", to classify non-linear data.

Classification ⇒ 'Soft margin classifier' ⇒ which allows some kind of misclassification.

SVM → It is a supervised machine learning uses concept of maximum margin classification.

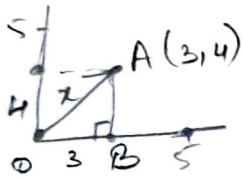
↓ linear seperator

Finding maximum margin between the hyperplanes that means maximum distances between the two classes. (more than 2 dimension)



Data points having close boundaries/appearance to other cluster of data points then we note them as vectors support vectors.

mathematical calculations

① 

vector = magnitude & direction

A = data point

$OA^2 = OB^2 + AB^2$

$OA^2 = 3^2 + 4^2$

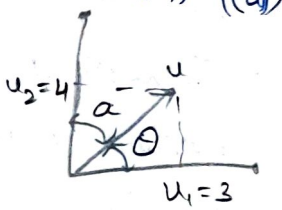
$OA = 5$

② The direction of a vector $u(u_1, u_2)$ is the vector $w\left(\frac{u_1}{\|u\|}, \frac{u_2}{\|u\|}\right)$

$u(u_1, u_2) = w\left(\frac{u_1}{\|u\|}, \frac{u_2}{\|u\|}\right)$

$w\left(\frac{3}{5}, \frac{4}{5}\right)$

direction of vector $= w(0.6, 0.3)$



③ sum of two vectors

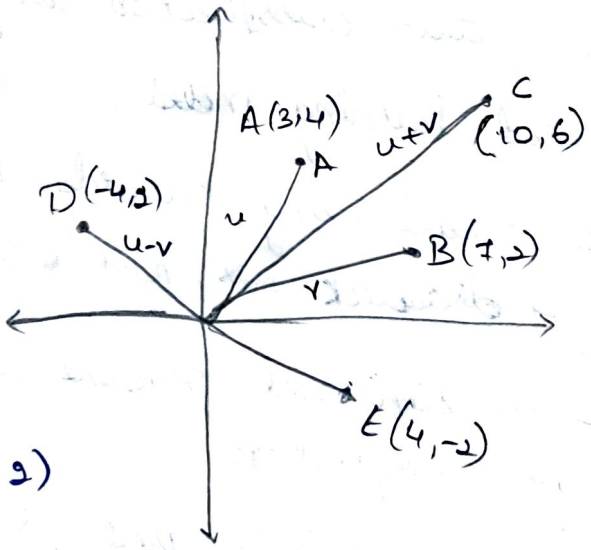
$u(u_1, u_2)$ and $v(v_1, v_2)$

$u+v = (u_1+v_1, u_2+v_2)$

$u_1=3 \quad u_2=4$
$v_1=7 \quad v_2=2 \quad = (10, 6)$

$u-v = (u_1-v_1, u_2-v_2)$
$= (3-7, 4-2) = (-4, 2)$

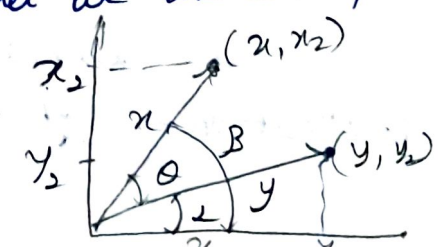
$v-u = (4, -2)$



How close the two points are to each other we use dot product.

$x \cdot y = \|x\| \|y\| \cos(\theta)$

$\cos(\theta) = \frac{\text{adjacent}}{\text{hypotenuse}}$



$\theta = \beta - \alpha$

$\cos(\beta - \alpha) = \cos(\beta)\cos(\alpha) + \sin(\beta)\sin(\alpha)$

$\cos(\beta) = \frac{x_1}{\|x\|}$ $\cos(\alpha) = \frac{y_1}{\|y\|}$

$\sin(\beta) = \frac{\text{opposite}}{\text{hypothenuse}}$ $\sin(\beta) = \frac{x_2}{\|x\|}$

$\sin(\alpha) = \frac{y_2}{\|y\|}$

$\cos(\beta - \alpha) = \frac{x_1}{\|x\|} \cdot \frac{y_1}{\|y\|} + \frac{x_2}{\|x\|} \cdot \frac{y_2}{\|y\|}$

$\cos\theta = \frac{x_1 y_1 + x_2 y_2}{\|x\|\|y\|}$

$\boxed{\|x\|\|y\|\cos\theta = x_1 y_1 + x_2 y_2}$

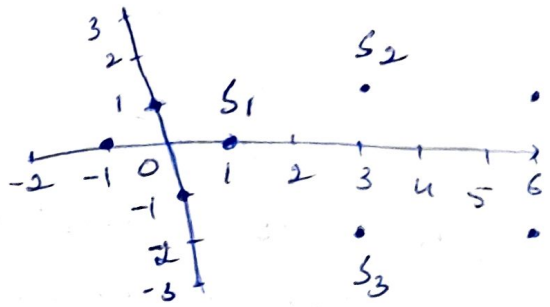
$x \cdot y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$

Example

Linear Equation Example

Suppose we are given the following positively labeled data points in $\mathbb{R}^2$

$\{\begin{pmatrix}3\\1\end{pmatrix}, \begin{pmatrix}3\\-1\end{pmatrix}, \begin{pmatrix}6\\1\end{pmatrix}, \begin{pmatrix}6\\-1\end{pmatrix}\}$ and the following negatively labeled data points in $\mathbb{R}^2$ $\{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\-1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}\}$



Support vectors $= s_1 = \begin{pmatrix}1\\0\end{pmatrix}$ $s_2 = \begin{pmatrix}3\\1\end{pmatrix}$ $s_3 = \begin{pmatrix}3\\-1\end{pmatrix}$

$\bar{s}_1 = \begin{pmatrix}1\\0\\1\end{pmatrix}$ $\bar{s}_2 = \begin{pmatrix}3\\1\\1\end{pmatrix}$ $\bar{s}_3 = \begin{pmatrix}3\\-1\\1\end{pmatrix}$

① Add bias

$\alpha_1 \bar{s}_1 \bar{s}_1 + \alpha_2 \bar{s}_2 \bar{s}_1 + \alpha_3 \bar{s}_3 \bar{s}_1 = -1 = \alpha_1 \begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix} + \alpha_2 \begin{pmatrix}3\\1\\1\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix} + \alpha_3 \begin{pmatrix}3\\-1\\1\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix}$

$2\alpha_1 + 4\alpha_2 + 4\alpha_3 = -1$ — (c1)

$\alpha_1 \bar{s}_1 \bar{s}_2 + \alpha_2 \bar{s}_2 \bar{s}_2 + \alpha_3 \bar{s}_3 \bar{s}_2 = +1$

$\alpha_1 \bar{s}_1 \bar{s}_3 + \alpha_2 \bar{s}_2 \bar{s}_3 + \alpha_3 \bar{s}_3 \bar{s}_3 = +1$

$\alpha_1 \begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}3\\1\\1\end{pmatrix} + \alpha_2 \begin{pmatrix}3\\1\\1\end{pmatrix}\begin{pmatrix}3\\1\\1\end{pmatrix} + \alpha_3 \begin{pmatrix}3\\-1\\1\end{pmatrix}\begin{pmatrix}3\\1\\1\end{pmatrix} = 1$

$4\alpha_1 + 11\alpha_2 + 9\alpha_3 = 1$ — (c2)

$\alpha_1 \begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}3\\-1\\1\end{pmatrix} + \alpha_2 \begin{pmatrix}3\\1\\1\end{pmatrix}\begin{pmatrix}3\\-1\\1\end{pmatrix} + \alpha_3 \begin{pmatrix}3\\-1\\1\end{pmatrix}\begin{pmatrix}3\\-1\\1\end{pmatrix}$

$4\alpha_1 + 9\alpha_2 + 11\alpha_3 = 1$ — (c3)

$2\alpha_1 + 4\alpha_2 + 4\alpha_3 = -1$

$-(4\alpha_1 + 11\alpha_2 + 9\alpha_3) = -1$

$-2\alpha_1 - 7\alpha_2 - 5\alpha_3 = -2$

$-2\alpha_2 + 2\alpha_3 = 0$

$-2\alpha_1 + 9\alpha_2 - 3\alpha_2 = -2$

$-2\alpha_1 - 6\alpha_2 = -2$

$\alpha_1 + 3\alpha_2 = 1$

$4\alpha_1 + 11\alpha_2 + 9\alpha_3 = 1$

$-(4\alpha_1 + 9\alpha_2 + 11\alpha_3) = -1$

$+\alpha_1 + 2\alpha_2 - 2\alpha_3 = 0$

$\alpha_2 = \alpha_3$

$3\alpha_2 = \alpha_2$

$\alpha_2 - 3\alpha_2 = 1$

$\alpha_2 = \frac{3}{4} = 0.75$

$\alpha_3 = 0.75$

$2\alpha_1 + 4\alpha_2 + 4\alpha_3 = -1$

$2\alpha_1 + 4(0.75) + 4(0.75) = -1$

$2\alpha_1 + 3 + 3 = -1$

$2\alpha_1 = -1 - 3 - 3 = -7$

$\alpha_1 = -\frac{7}{2} = -3.5$

② weight matrix

$\bar{w} = \sum_{i=1}^{n} \alpha_i \bar{s}_i$

$\bar{w} = \alpha_1 \bar{s}_1 + \alpha_2 s_2 + \alpha_3 s_3$

$\bar{w} = -3.5 \begin{pmatrix}1\\0\\1\end{pmatrix} + 0.75 \begin{pmatrix}3\\1\\1\end{pmatrix} + 0.75 \begin{pmatrix}3\\-1\\1\end{pmatrix}$

$\bar{w} = \begin{pmatrix}-3.5\\0\\-3.5\end{pmatrix} + \begin{pmatrix}3.75\\0.75\\0.75\end{pmatrix} + \begin{pmatrix}2.25\\-0.75\\0.75\end{pmatrix} = \begin{pmatrix}+1\\0\\-2\end{pmatrix}$

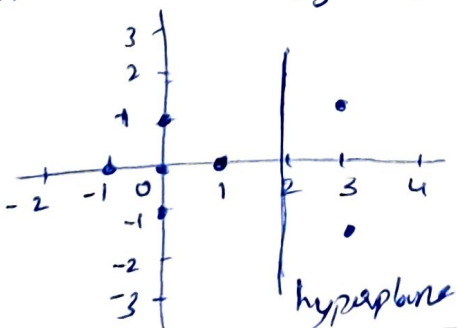
$\begin{pmatrix}1\\0\end{pmatrix}$ > parallel to y-axis

$\begin{pmatrix}0\\1\end{pmatrix}$ > parallel to x-axis

$\begin{pmatrix}0\\0\end{pmatrix}$ > origin

$|-2| = 2$ towards positive

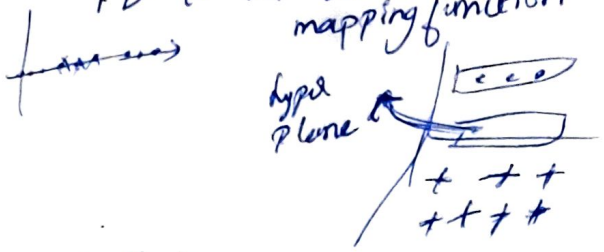
2 = negative direction



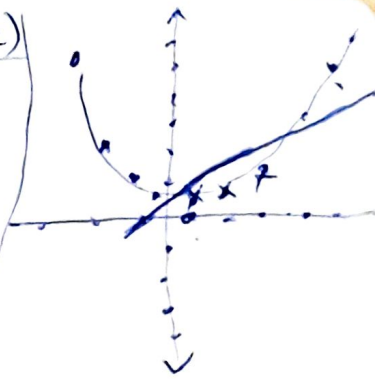
Non linear classifier

Mapping to a higher dimension

1D to 2D to 3D mapping function



X	X-5	$(X-5)^2 = f(x)$
2	-3	9
3	-2	4
4	-1	1
6	1	1
7	2	4
8	3	9
9	4	16
10	5	25
11		



$e^{-\infty} = 0$

$e^{0} = 1$

Kernel Trick

Gaussian RBF Kernel $K(\vec{x}, \vec{l}^{i}) = e^{\frac{-1\|\vec{x} - \vec{l}^{i}\|^2}{2\sigma^2}}$