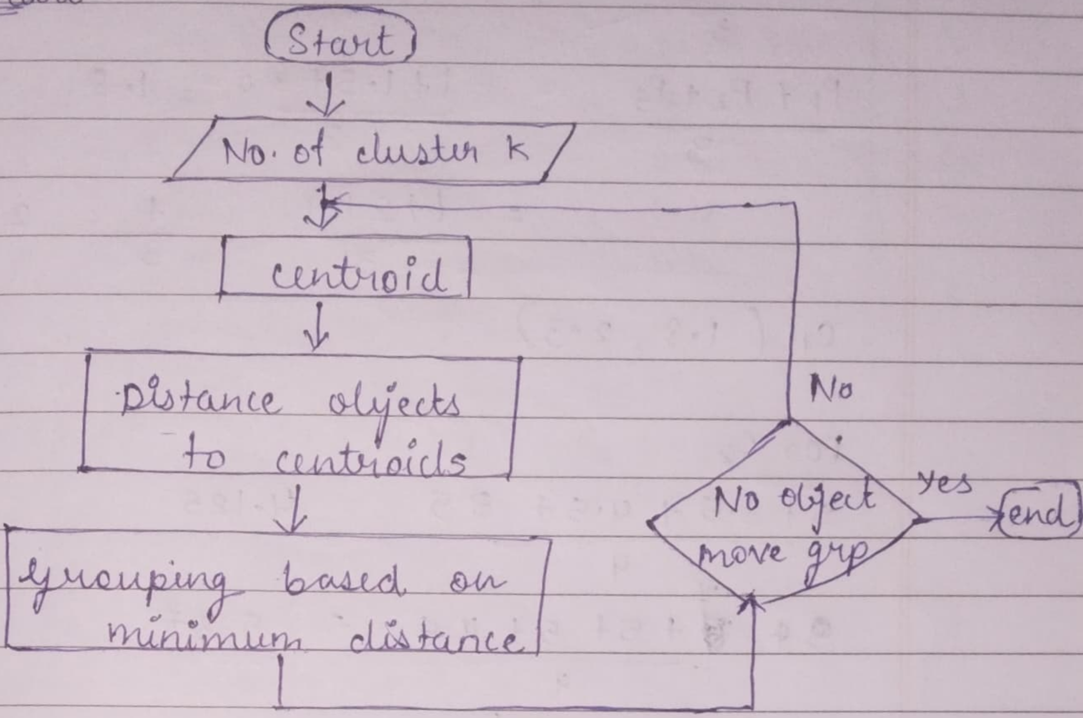
Unit :- 5

Unsupervised Learning

5/3

- Flowchart


Start
No. of cluster K
centroid
Distance objects to centroids
grouping based on minimum distance
No object move grp
No
yes
end

Points	Variable 1	Variable 2
C1 ← 1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
C2 ← 4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Points	$C_1(1.0, 1.0)$	$C_2(5.0, 7.0)$	Allocation
$P_1(1.0, 1.0)$	0	$\sqrt{52} = 7.21$	C_1
$P_2(1.5, 2.0)$	$\sqrt{1.25} = 1.11$	$\sqrt{37.25} = 6.10$	C_1
$P_3(3.0, 4.0)$	$\sqrt{13} = 3.6$	$\sqrt{13} = 3.6$	C_1
$P_4(5.0, 7.0)$	$\sqrt{52} = 7.21$	0	C_2
$P_5(3.5, 5.0)$	$\sqrt{22.25} = 4.71$	$\sqrt{3.20}$ 2.5	C_2
$P_6(4.5, 5.0)$	5.3	2.0	C_2
$P_7(3.5, 4.5)$	2.06	3~~.24~~ 2.9	C_2

For Eg:2

C_1, C_2, C_3

$\cancel{\frac{P_1 + P_2 + P_3}{3} = 0+x}$

$\frac{P_1 + P_2 + P_3}{3} = \frac{1+1.5+3.0}{3} = 1.8$

$= \frac{1+2+4}{3} = \frac{7}{3} = 2.33$

$C_1\ (1.8, 2.3)$

For C_2

$\frac{5 + 3.5 + 4.5 + 3.5}{4} = 4.125$

$\cancel{3+}\ \frac{\overset{7}{\cancel{3}} + 5 + 5 + 4.5}{4} = 5.37$

$C_2\ (4.12, 5.37)$

For 2nd Iteration

Points	$C_1(1.8, 2.3)$	$C_2(4.12, 5.37)$
P_1	1.31	
P_2	0.42	
P_3	1.447	
P_4		
P_5		
P_6		
P_7		

	$C_1(1,1)$	$C_2(1.5,2)$	$C_3(3,4)$	
P_1	0	1.12	3.61	C_1
P_2	1.12	0	2.50	C_2
P_3	3.61	2.5	0.	C_3
P_4	7.21	6.4	3.61	C_3
P_5	5.15	4.3	1.12	C_3
P_6	5.70	4.92	1.80	C_3
P_7	4.60	3.82	0.71	C_3

$C_1 = (1,1)$ C_2 $(1.5, 2.0)$ $C_3 = (3,4)$

for C_1 $C_1 = (1,1)$

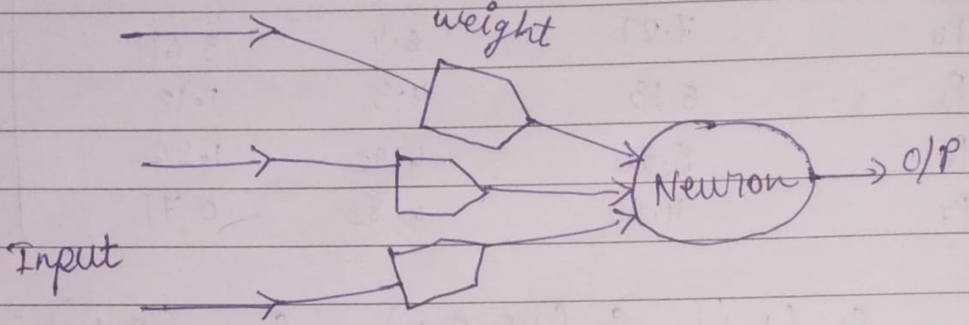
$C_2 = (1.5, 2.)$

$C_3 =$ (~~5.2~~, $(3.9, 4.3)$

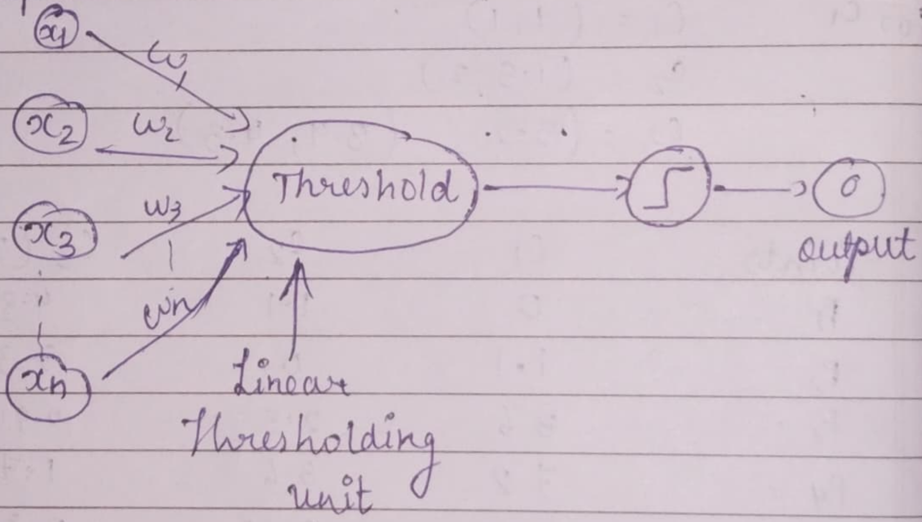
Points	C_1	C_2	C_3 $(3.9, 4.3)$	
P_1	0	1.1	4.3	C_1
P_2	1.1	0	3.3	C_2
P_3	3.6	2.5	0.94	C_3
P_4	7.2	3.6	1.7	C_3
P_5	4.7	3.60	0.8	C_3
P_6	5.3	4.2	0.9	C_3
P_7	4.3	3.2	0.44	C_3

* Neural Networks

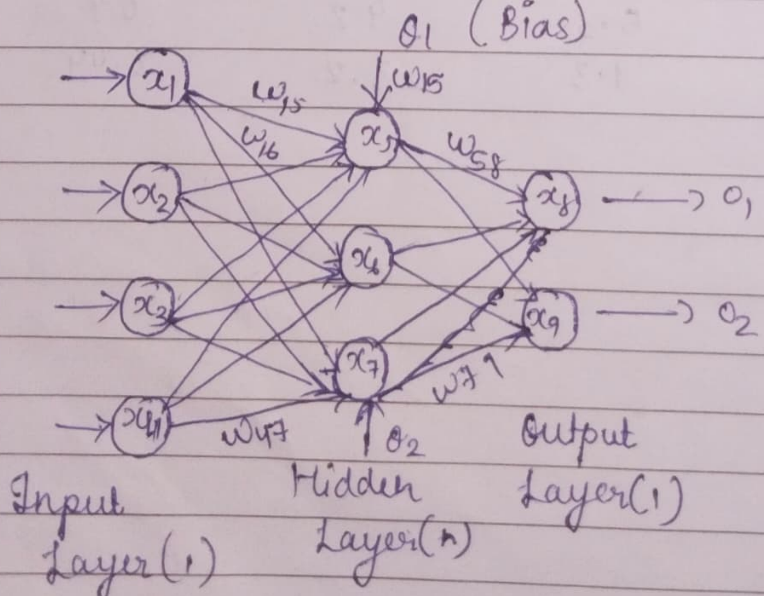
Artificial Neural Network



Perceptron



Neural Network



Activating fn : To solve non-linearity

* Learning Algorithm : Back Propagation

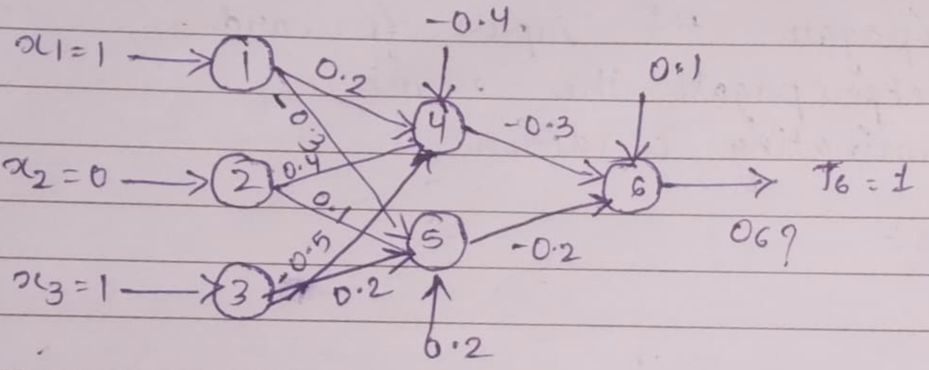
Method :

1) Initialize weights
2) Propagate the inputs forward
3) Backpropagate the error
4) Terminating condition.

Steps :-

```
    for each training tuple X in D {
        for each input layer unit j {
            $O_j = I_j$
        for each hidden or output layer unit j {
            $I_j = \Sigma_i w_i O_i + \theta_j$
            $O_j = \frac{1}{1 + e^{-I_j}}$
```

Eg:- Consider multilayer feed forward network $l = 0.9$, initial weights & biases are given. 1st Training tuple $x = (1,0,1)$ and class label 1.



⇒ learning rate $\eta = 0.9$

For hidden layer activations

$$I_j = \Sigma_i w_i O_i + \theta_j$$

$$I_4 = (1 \times 0.2) + (0 \times 0.4) + 1 \times (-0.5) + (-0.4)$$
$$= -0.7$$

$$I_5 = (1 \times -0.3) + (0 \times 0.1) + (1 \times 0.2) + 0.2$$
$$= 0.1$$

Applying sigmoid activation fn :-

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

$$a_4 = \sigma(-0.7) = \frac{1}{1 + e^{+0.7}}$$
$$\approx 0.331$$

$$a_5 = \sigma(0.1) = \frac{1}{1 + e^{-0.1}} \approx 0.525$$

The output neuron receives

$$I_6 = (a_4 \times -0.3) + (a_5 \times -0.2) + 0.1$$
$$= (0.331 \times -0.3) + (0.525 \times -0.2) + 0.1$$
$$= -0.1043$$

Applying sigmoid on output neuron

$a_6 = \sigma(-0.1043) = \frac{1}{1+e^{0.1043}} = 0.474$

Compute Error

$E = \cancel{\frac{1}{2}(T-a_6)^2 = \frac{1}{2}(1-0.474)^2}$

$= 1/2\ (0.276) = \cancel{0.138}$

$E_6 = (0.474)(1-0.474)(1-0.474)$

$= 0.131$

$E_4 = (0.331)(0.669)(0.131 \times -0.3)$

$= -0.0087$

$E_5 = (0.525)(1-0.525)(0.131 \times -0.2)$

$= -0.0065$

Computing weight updates

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

$\Delta w_{4,6} = (l)\, E_j\, O_i$

$= 0.9 \times 0.131 \times 0.331 = 0.039$

$w_{4,6} = -0.3 + 0.039 = -0.261$

$\Delta w_{56} = 0.9 \times 0.131 \times 0.525 = 0.0619$

$w_{5,6} = -0.2 + 0.0619 = -0.138$

$\Delta w_{1,4} = 0.9 \times -0.0087 \times 1 = -0.0078$

$w_{1,4} = 0.2 + (-0.0078) = 0.1922$

$\Delta w_{2,4} = 0.9 \times -0.0087 \times 0 = 0$

$w_{2,4} = 0.4 + 0 = 0.4$

$\Delta w_{3,4} = 0.9 \times -0.0087 \times 1 = -0.0078$

$w_{3,4} = -0.5 + (-0.0078) = -0.5078$

$\Delta w_{1,5} = 0.9 \times -0.0065 \times 1 = -0.0058$

$w_{1,5} = \cancel{0.5 + (-0.}$

$-0.3 + (-0.0058) = -0.3058$

$\Delta W_{2,5} = 0.9 \times -0.0065 \times 0 = 0$

$W_{2,5} = 0.1 + 0 = 0.1$

$\Delta W_{3,5} = 0.9 \times -0.0065 \times 1 = -0.0058$

$W_{3,5} = 0.2 + -0.0058 = 0.1942$

Updating Biases

$\Delta \theta j = 0.9 \times Ej$

$\theta j = \theta j + \Delta \theta j$

$\Delta \theta_4 = 0.9 \times -0.0087 = -0.0078$

$\theta_4 = -0.4 + -0.0078 = -0.4078$

$\Delta \theta_5 = 0.9 \times -0.0065 = -0.0058$

$\theta_5 = 0.2 + -0.0058 = 0.1942$

$\Delta \theta_6 = 0.9 \times 0.131 = 0.118$

$\theta_6 = 0.1 + 0.118 = 0.218$ //

★ Types of Activation Functions

1> Identity function
2> Threshold / step function
3> ReLU (Rectified Linear Unit function)
4> Sigmoid function
5> Hyperbolic tangent function

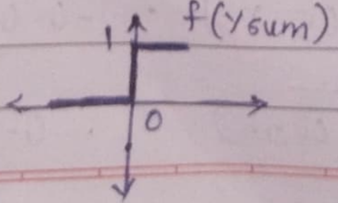
1> Identity :-

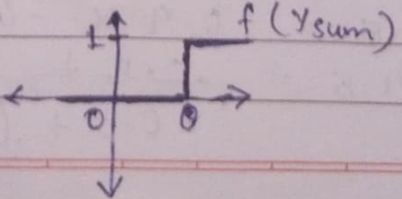
$Y_{out} = f(x) = x$, for all x (input = o/p)

2> Step fn :-

$$Y_{out} = f(Y_{sum}) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

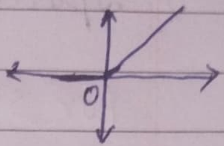
↓



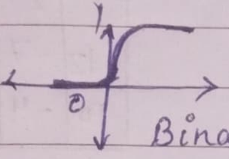


$$Y_{out} = f(Y_{sum}) = \begin{cases} 1, & x \geq \theta \\ 0 & x < \theta \end{cases}$$

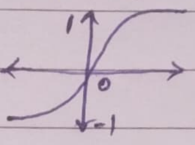
3.) ReLU => $f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$



4.) Sigmoid fn => $f(x) = \dfrac{1}{1+e^{-x}}$



Binary



(Bipolar)

5.) Hyperbolic fn :- $f(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$

$$Y_{out} = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ 1, & x < 0. \end{cases}$$

★ Deep Learning

=> Convolutional Neural Networks (CNN) :-

Basic convolution Operation

step:

Input

4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

*

filter

1	0	-1
1	0	-1
1	0	-1

=

Result

2	6	-4	5
0	4	0	-1
-5	0	3	6
-2	5	0	3

stride : 1

Padding = 1

For HW

Stride = 2

Padding = 1

Info. Loss ?

generate the Result.

Application :-

1> Parameter sharing

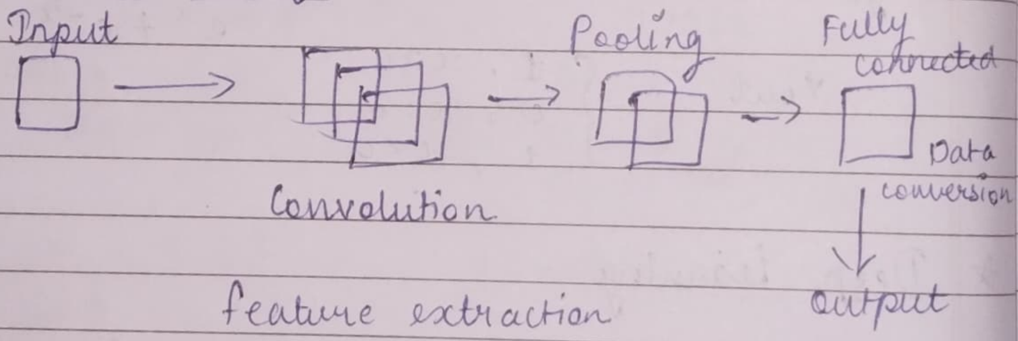
2> Sparcity of connections.

For stride = 2

Result

-4	2	-6	8
-11	2	-3	12
-12	-7	7	12
-10	-4	5	9

* Architecture of CNN :-



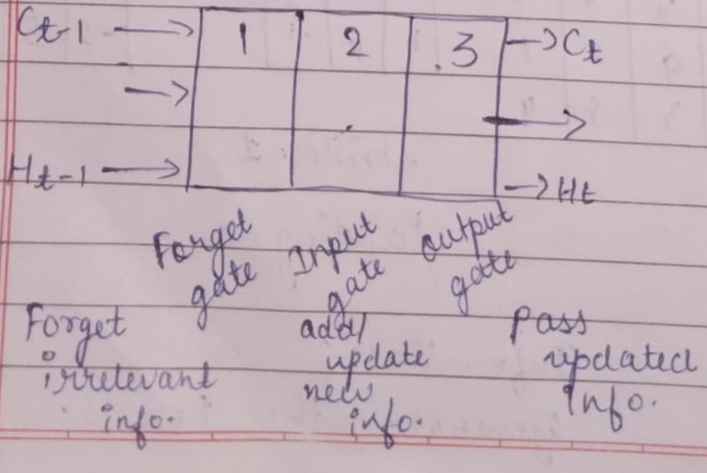
=> Flattering

=> Dropout

=> Activation

★ RNN (Recurrent Neural Network)

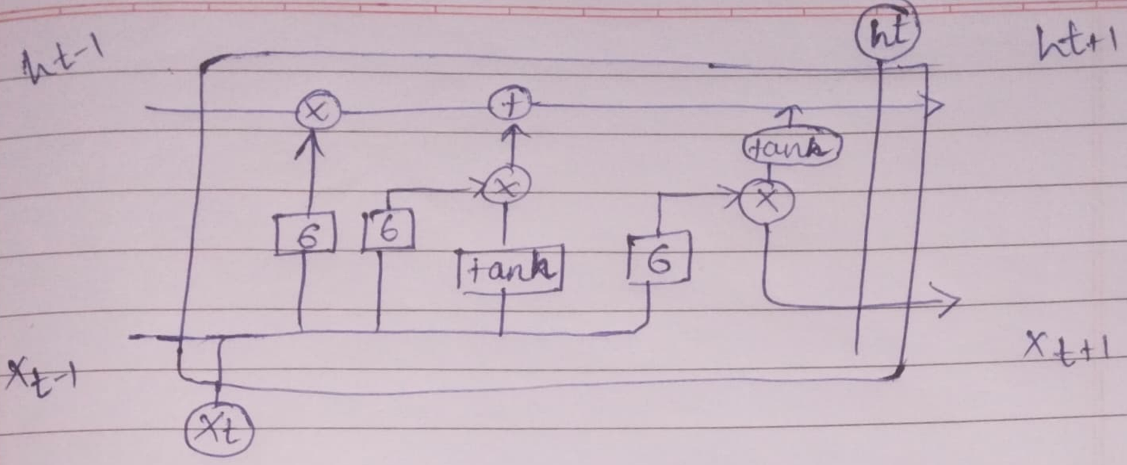
↳ LSTM (Long Short Term Memory)



H = Hidden State

C - cell state

t = time stamp


ht-1
ht
ht+1
×
+
tank
×
×
6
6
tank
6
xt-1
xt+1
xt