N. I. MD. ASHAFUDDLA

এন. আই. মোঃ আশফুদ্দৌলা

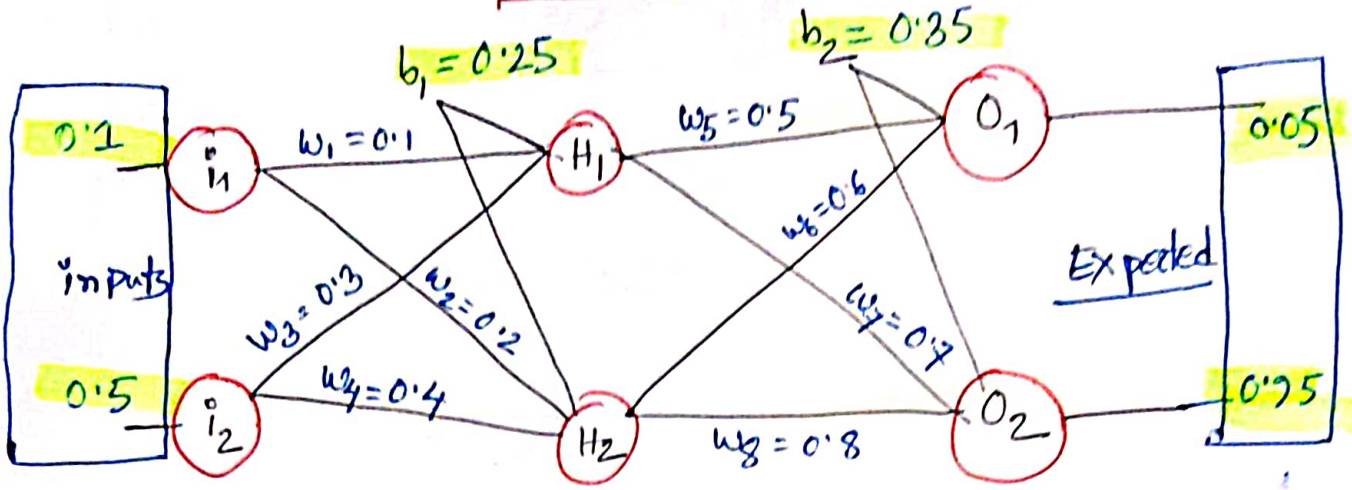
01932082303 /

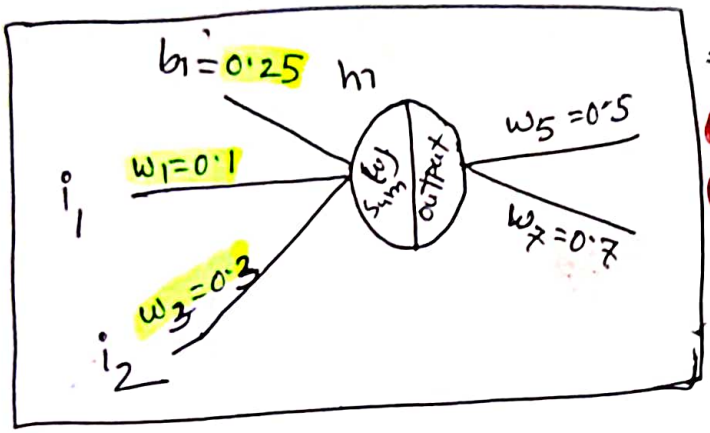
01701009987

Software dev,
IT Division, MBLBD.

✓ Logistic function = Sigmoid function.

BPNN





* inside a unit 2 operations happen
1. Sum (weight)
2. Squashing Sum (weight) using activation function.

* Activation function becomes the input to the next layer.

* Sigmoid Activation output is between 0 & +1

1. $Sum(H_1) = i_1 * w_1 + i_2 * w_3 + b_1$

$= 0.1 \times 0.1 + 0.5 \times 0.3 + 0.25 = 0.41$

$$Output(H_1) = \frac{1}{1 + e^{-Sum(H_1)}} = 0.601$$

2. $Sum(H_2) = i_1 \times w_2 + i_2 \times w_4 + b_1 = 0.47$

$$Output(H_2) = \frac{1}{1 + e^{-Sum(H_2)}} = 0.61538$$

3. $Sum(O_1) = Output(H_1) \times w_5 + Output(H_2) \times w_6 + b_2 = 1.01977$.

$$Output(O_1) = \frac{1}{1+e^{-Sum(O_1)}} = 0.73492.$$ Exp = 0.05

4. $Sum(O_2) = Output(H_1) \times w_7 + Output(H_2) \times w_8 + b_2 = 1.26306$

$$Output(O_2) = \frac{1}{1+e^{-Sum(O_2)}} = 0.779855.$$ Exp = 0.95.

Total Error $E_{Total} = \sum \frac{1}{2}(target - output)^2$

$$E_1 = \frac{1}{2}(T_1 - O_1)^2$$
$$= \frac{1}{2}(0.05 - 0.73492)^2 = 0.23456$$

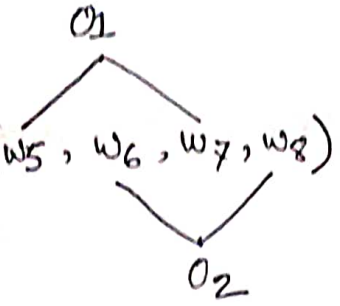
$$E_2 = \frac{1}{2}(T_2 - O_2)^2$$
$$= \frac{1}{2}(0.95 - 0.77955)^2 = 0.01452$$

$$E_{Total} = E_1 + E_2 = 0.24908.$$

Backpropagation

— aim is to distributed the total Error back to the netwo so as to update the weights in order to minimize the cost function.

weights in the output layer (w_5, w_6, w_7, w_8)



For w_5,

w_5 has contribution on E_1.

$E_1 \leftarrow O_1 \leftarrow \text{Sum}(O_1) \leftarrow w_5$

$$\frac{\delta E_{Total}}{\delta w_5} = \frac{\delta E_{Total}}{\delta \text{output}(O_1)} \times \frac{\delta \text{Output}(O_1)}{\delta \text{Sum}(O_1)} \times \frac{\delta \text{Sum}(O_1)}{\delta w_5} \quad \text{---(1)}$$

1. $$\frac{\delta E_{Total}}{\delta \text{Output}(O_1)} = \frac{\delta\left(\sum \frac{1}{2}(T-O)^2\right)}{\delta \text{Output}(O_1)}$$

$$= \frac{\delta\left[\frac{1}{2}(T_1-O_1)^2 + \frac{1}{2}(T_2-O_2)^2\right]}{\delta \text{Output}(O_1)}$$

$$= \frac{\delta \frac{1}{2}(T_1-O_1)^2}{\delta \text{Output}(O_1)}$$

$$= \frac{1}{2} \times 2(T_1 - O_1) * (-1)$$

$$= \boxed{O_1 - T_1}$$

2. $$\delta(x) = \frac{1}{1+e^{-x}} = (1+e^{-x})^{-1}$$

$$\frac{d\delta(x)}{dx} = \frac{d}{dx}(1+e^{-x})^{-1}$$

$$= (-1)(1+e^{-x})^{-2} e^{-x}(-1)$$

$$= e^{-x}\delta^2(x) \quad \delta(x)(1-\delta(x))$$

$$= \text{Output}(O_1)(1-\text{Output}(O_1))$$

3. $Sum(o_1) = Output(H_1) \times w_5 + Output(H_2) \times w_6 + b_2$

$$\frac{\delta Sum(o_1)}{\delta w_5} = Output(H_1)$$

using eq(2) we get —

$$\frac{\delta E_{Total}}{\delta w_5} = [Output(o_1) - target(2)] * [Output(o_1)(1 - Output(o_1)] * Output(H_1)$$

$$= 0.68492 * 0.19480 * 0.60108$$

$$= 0.08020$$

$$new\_w_5 = w_5 - \eta * \frac{\delta E_{Total}}{\delta w_5}$$

η = learning rate = 0.6

$$= 0.5 - 0.6 * 0.08020$$

$$= 0.45187$$

For w_6,

$$\frac{\delta E_{total}}{\delta w_6} = \frac{\delta E_{Total}}{\delta Output(o_1)} \times \frac{\delta Output(o_1)}{\delta Sum(o_1)} \times \frac{\delta Sum(o_1)}{\delta w_6}$$

$$= [Output(o_1) - target(1)] * [Output(o_1)(1 - Output(o_1)] * Output(H_2)$$

$$= 0.68492 * 0.19480 * 0.61538$$

$$= 0.08211$$

$$new\_w_6 = w_6 - \eta * \frac{\delta E_{Total}}{\delta w_6}$$

$$= 0.6 - 0.6 * 0.08211$$

$$= 0.55073.$$

For w_7,

$$\frac{\delta E_{Total}}{\delta w_7} = \frac{\delta E_{Total}}{\delta Output(O_2)} \times \frac{\delta Output(O_2)}{\delta sum(O_2)} \times \frac{\delta sum(O_2)}{\delta w_7}$$

$$= [O_2 - T_2] * [O_2 * (1 - O_2)] * O(H_1)$$

$$= (-0.17044) \times 0.17184 \times 0.60108$$

$$= -0.01760$$

$$new\_w_7 = w_7 - \eta * \frac{E_{Total}}{\delta w_7}$$

$$= 0.7 - 0.6 * (-0.01760)$$

$$= 0.71056.$$

$$new\_w_8 = w_8 - \eta * \frac{E_{Total}}{\delta w_8}.$$

$$= 0.81081$$

Weights in the hidden layer (w_1, w_2, w_3, w_4)

we need to calculate $\frac{\delta E_1}{\delta w_1}$ & $\frac{\partial E_2}{\delta w_1}$ separately.

$$\frac{\delta E_1}{\delta w_1} = \frac{\delta E_1}{\delta \text{Output}(O_1)} \times \frac{\delta \text{Output}(O_1)}{\delta \text{Sum}(O_1)} \times \frac{\delta \text{Sum}(O_1)}{\delta \text{Output}(H_1)} \times \frac{\delta \text{Output}(H_1)}{\delta \text{Sum}(H_1)} \times \frac{\delta \text{Sum}(H_1)}{\delta w_1}$$

$E_1 \leftarrow O_1 \leftarrow \text{Sum}(O_1) \leftarrow \text{Output}(H_1) \leftarrow \text{Sum}(H_1) \leftarrow$

1. $\frac{\delta E_1}{\delta \text{Output}(O_1)} = O_1 - T_1$.

2. $\frac{\delta \text{Output}(O_1)}{\delta \text{Sum}(O_1)} = O_1(1-O_1)$

3. $\text{Sum}(O_1) = \text{Output}(H_1) * w_5 + \text{Output}(H_2) * w_6 + b_2$

$\frac{\delta \text{Sum}(O_1)}{\delta \text{Output}(H_1)} = w_5$.

4. $\frac{\delta \text{Output}(H_1)}{\delta \text{Sum}(H_1)} = O(H_1)(1 - O(H_1))$

5. $\text{Sum}(H_1) = i_1 * w_1 + i_2 * w_3 + b_1$

$\frac{\delta \text{Sum}(H_1)}{\delta w_1} = i_1$

Now, $\frac{\delta E_1}{\delta w_1} = (O_1 - T_1) \times [O_1 - (1-O_1)] \times w_5 \times [O(H_1)(1-O(H_1))] \times i_1$

$= 0.68492 \times 0.19980 \times 0.5 \times 0.23978 \times 0.1 = 0.00159$.

$$\frac{\delta E_2}{\delta w_1} = \frac{\delta E_2}{\delta \text{Output}(O_2)} \times \frac{\delta \text{output}(O_2)}{\delta \text{sum}(O_2)} \times \frac{\delta \text{Sum}(O_2)}{\delta \text{output}(H_1)} \times \frac{\delta \text{Output}(H_1)}{\delta \text{Sum}(H_1)} \times \frac{\delta \text{Sum}(H_1)}{\delta w_1}$$

$$= [O_2 - T_2] \times [O_2(1-O_2)] \times w_7 \times [O(H_1)(1-O(H_1))] \times i_1$$

$$= (-0.17044) \times (0.17184) \times 0.7 \times 0.23978 \times 0.1$$

$$= 0.00049$$

$$\frac{\delta E_{Total}}{\delta w_1} = \frac{\delta E_1}{\delta w_1} + \frac{\delta w_2}{\delta w_1} = 0.00159 + (-0.00049) = 0.00110$$

$$new\_w_1 = w_1 - \eta * \frac{\delta E_{Total}}{\delta w_1}$$

$$= 0.1 - 0.6 * 0.00110$$

$$= 0.09933.$$

$new\_w_2 = 0.19919$

$new\_w_3 = 0.29667.$

$new\_w_4 = 0.39597.$

NN working principle

3 fundamental func —

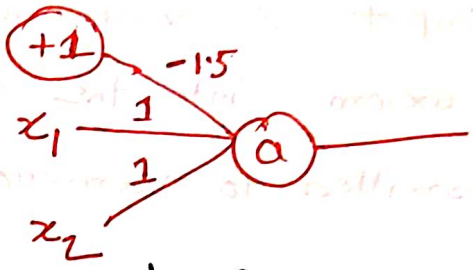
OR, AND, NOT, XOR

(-0.5) (-1.5) 1 (-2) 0.5 (-1 -1) $= \bar{A} \cdot \bar{B}$

Activation function. $f(x) = \begin{cases} 0, & \text{for } x < 0 \\ 1, & \text{for } x \geq 1. \end{cases}$

AND

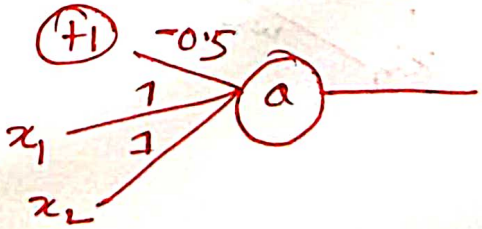
$a = f(-1.5 + x_1 + x_2)$



x_1	x_2	x_1 AND x_2	$f(a)$	a
0	0	0	-1.5	0
0	1	0	-0.5	0
1	0	0	-0.5	0
1	1	1	0.5	1

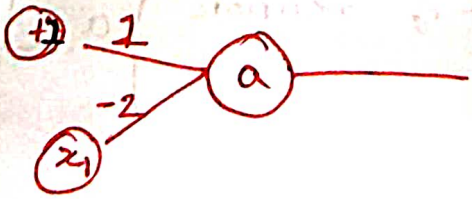
OR

$a = f(-0.5 + x_1 + x_2)$



NOT

$a = f(1 - 2x_1)$



MuHilayer

XNOR

x_1	x_2	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

This relationship can be modeled ~~as~~ using a single neuron.

✓ ~~Gen~~ Idea! complex Relations can be broken into simpler functions and combined.

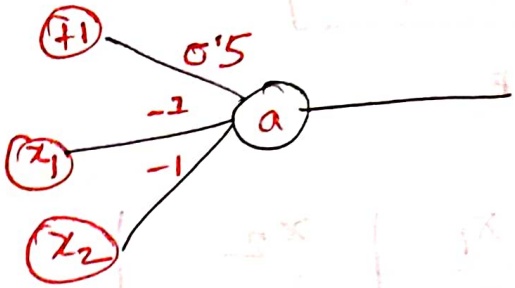
$$x_1 \text{ XNOR } x_2 = \text{NOT}(x_1 \text{ XOR } x_2)$$
$$= \text{NOT}\left[(A+B)\cdot(\bar{A}+\bar{B})\right]$$
$$= \overline{(A+B)} + \overline{(\bar{A}+\bar{B})}$$
$$= (\bar{A}\cdot\bar{B}) + (AB)$$

$x_1 \oplus x_2$
$= (A+B)\cdot(\bar{A}+\bar{B})$

Model For $(\bar{A}.\bar{B})$

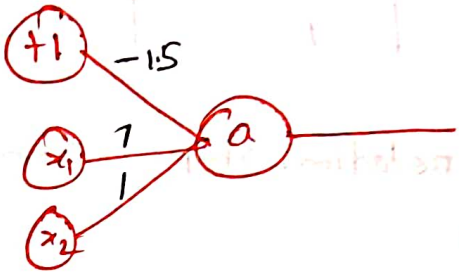
- output of this neuron

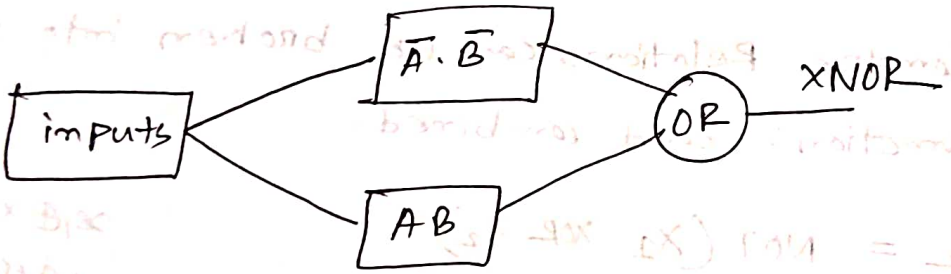
$$a = f(0.5 * -x_1 - x_2)$$

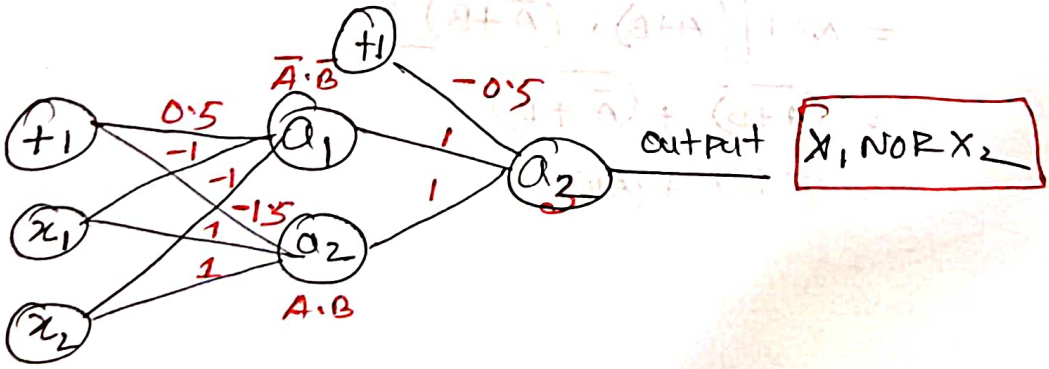


For $(A.B)$

$$a = f(-1.5 + x_1 + x_2)$$







x_1	x_2	a_1	a_2	a_3
0	0	1	0	1
0	1	0	0	0
1	0	0	0	0
1	1	0	1	1

Learning Rules

1. Hebbian Rule
2. Perceptron "
3. Delta " "

1. Hebbian ——

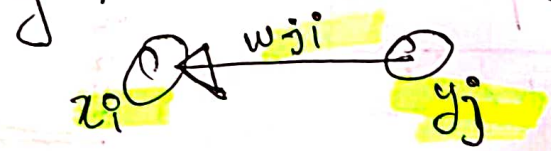
- unsup —
- Strengthen —
- increase the weight of conn —— at every

$$\Delta w_{ji}(t) = x_i(t) \cdot y_j(t)$$

at time (t)

input * output

(i) (j)



2. Perceptron.

- Error correcting
- supervised.
- Single layer (ff).
- Linear Activation function.

$$y = \begin{cases} 0 & \text{for } y_{in} < \theta \\ 1 & \text{for } y \geq \theta \end{cases}$$

$w_{new} = w_{old} + tx$

target * input

3. Delta Rule.

1. Widrow Hoff Rule.
2. LMS
3. minimize overall Error of training patterns.
4. Base Gradient descent approach.

This updates synaptic weights so as to min the error.

$$\Delta w_i = \eta x_i e \quad \text{where } e = (T - O)$$

Perceptron for Single output

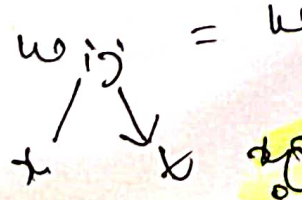
$$f(x) = \begin{cases} 0, & \text{for } y_{in} < \theta \\ 1 & \text{for } y_{in} \geq \theta \end{cases}$$

$w_{new} = w_{old} + \eta t x_i$

$b_{new} = b_{old} + \eta t$

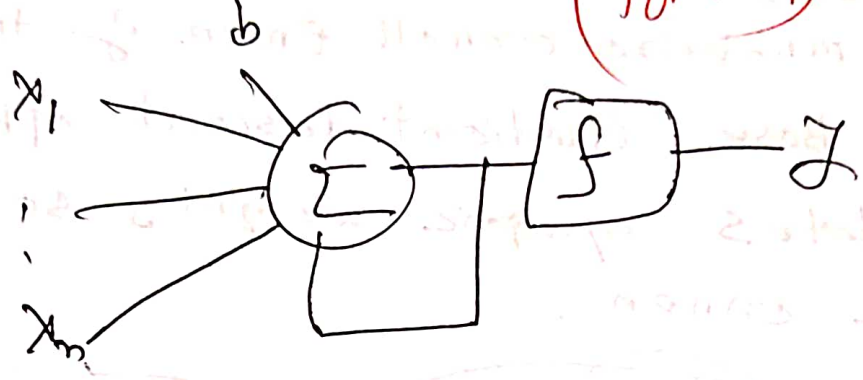
Multi

$w_{ij} = w_{ij} + \eta t_j x_j$



Adaptive Linear Neuron. (Adaline)

- Having single ~~li~~ linear Unit.
- Uses bipolar Activation function. (tan h) ~~tan~~



- uses delta Rule.

- Arch
  - similar to perceptron having Extra FB loop
  - ..

(tan h)

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

$$w_i(new) = w_i(old) + \eta(t-y)x_i$$

$$b = b + \alpha(t-y).$$

Paper

✓ CNN or ~~ConvNet~~

✓ we can identify, extract interesting & useful ~~info~~ patterns. from data.

✓ model consists of 3 consecutive 2D convolut
layer each having 2x2 kernel

✓ computer ← image = Array of pixel.
= height x width x dim img.

for [RGB (6x6x3)
Gray (4x4x1)].

To predict performance.

each input image is filtered by a series of layer.

Convolution
↓
pooling.
↓
Fully connected ⟶ used to classify.

Activation function.

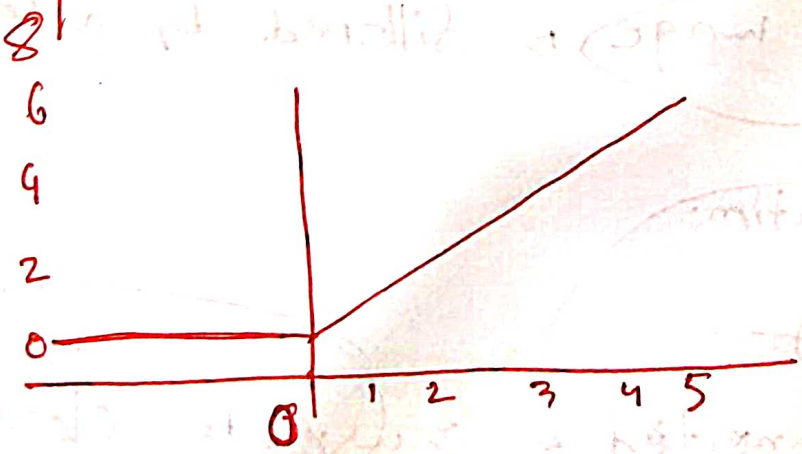
1. ReLu.
2. Tanh.
3. Sigmoid
4.

ReLu It is used in deep neural nets. Recently It has been shown to have 6 times improved convergence from the function of Tanh.

$$R(x) = \max(0, x)$$

if $x < 0$, $R(x) = 0$ and

$x >= 0$, $R(x) = x$.



Convo Neural Network.

Hyperparameters (knobs)

1. convolution
   - Number of features.
   - size of features.

2. pooling
   - window size — 2/3
   - window stride — 2

3. Fully connected.
   - Number of neurons.

Limitations

ConvNets only capture: local "spatial" patterns in data. If the data can't be made to look like an image, ConvNets are less useful.

Rule of Thumb

— If your data is just as useful after swapping any of your columns with each other, the you can't use CNN.

Hopefield Network algo

1. Assign connection weights.

$$w_{ij} = \begin{cases} \sum_{i=0}^{N-1} x_i^s x_j^s & i \neq j \\ 0 & i = j \end{cases} \qquad 0 \leq i, j \leq N-1$$

2. init with unknown pattern.

$$\mu_i(0) = x_i \qquad 0 \leq i \leq N-1$$

$\mu_i(t)$ = output of node (i) at time (t)

3. Iterate until convergence.

$$\mu_i(t+1) = f_n \left[\sum_{i=0}^{N-1} w_{ij} \mu_j(t) \right] \qquad 0 \leq j \leq N-1$$

Weight vector calculated from stored pattern in Hopfield Network.

$P_1 = \{1, -1, 1, -1\}$

$P_2 = \{-1, 1, -1, -1\}$

$P_3 = \{1, 1, -1, 1\}$

$P_4 = \{1, 1, 1, 1\}$

P_5

weight matrix =

$$w_{ij} = \begin{bmatrix} w_{00} & w_{01} & w_{02} & w_{03} \\ w_{10} & w_{11} & w_{12} & w_{13} \\ w_{20} & w_{21} & w_{22} & w_{23} \\ w_{30} & w_{31} & w_{32} & w_{33} \end{bmatrix}$$

Here, Every pattern has inputs x_0, x_1, x_2, x_3

When, $i = j$ weight = 0.

✓ Set these position = 0.

✓ Symmetric weight figure.

$$w_{ij} = \begin{cases} \sum_{s=0}^{N-1} x_i^s x_j^s & i \neq j \\ 0 & i = j, \quad 0 \leq i, j \leq N-1 \end{cases}$$

$w_{00} = 0$, $i = j$

$i=0, j=1$ $w_{01} = (-1 \times 1) + (-1 \times -1) + (1 \times 1) + (1 \times 1)$

$= -1 + 1 + 1 + 1 = 2$

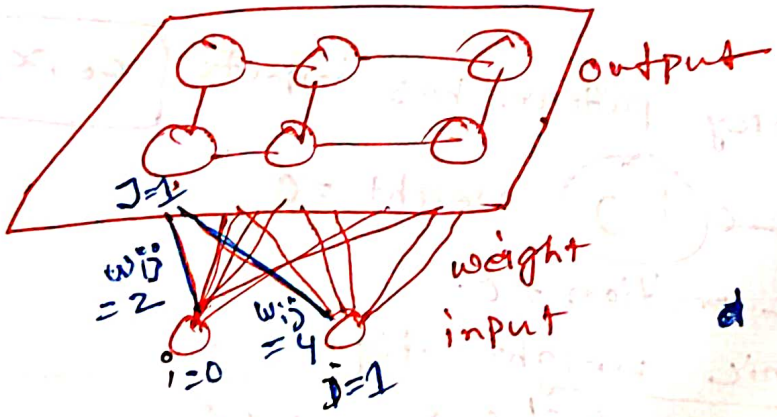
KSOM

(I) used to Reduce dim.

(II) beneficial if the network to form its own classifications of the training data.

(III) Note: The neurons are not arranged in in layers as in the MLP (input, hidden, out) but rather on a flat Grid (output)

(IV) All inputs connect to every node in the network.



Steps

1. init: Define $w_{ij}(t)$ $(0 \le i \le n-1)$ to be the weights from (i to node j) at (t) with small random value.

set init Radius around Node j, $N_j(0)$ to be large.

2. $x_i(t)$ = input to node i at time (t).

3. Calculate distance.

$$d_j = \sum_{i=0}^{n-1} (x_i(t) - w_{ij}(t))^2$$

$(2^2 + 4^2) = (20)$ example.

4. Select min (d)

5. Update weight for nearest node

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\left(x_i(t) - w_{ij}(t)\right)$$

$\eta(t)$ → gain term.

MLP app.

1. Airline ticketing.
2. ECG noise filtering.
3. Financial app.

Hamming dist

$$x = (x_1, x_2 \cdots x_n)$$

$$y = (y_1, y_2 \cdots y_n)$$

$$H = \sum (|x_i - y_i|)$$

Eucledian:

$$d(x, y) = \sqrt{\sum (x_i - y_i)^2}$$

KSOM Exmple

— calculate ksom cluster from.

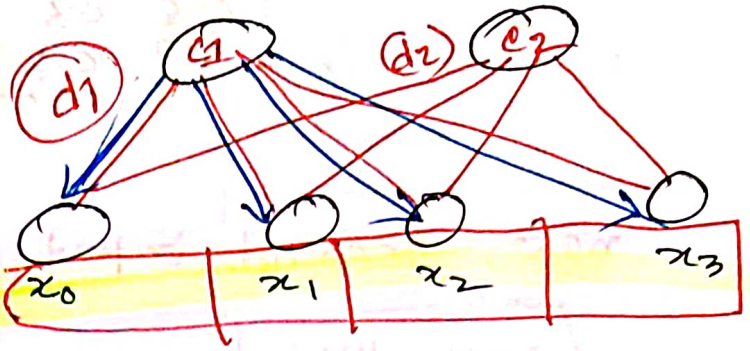
$x_0\ x_1\ x_2\ x_3$

$X_1 = [0\ 0\ 1\ 1]$

$X_2 = [1\ 0\ 0\ 0]$

$X_3 = [0\ 1\ 1\ 0]$

$X_4 = [0\ 0\ 0\ 1]$



[no. of cluster = 2
learning rate = 0.5.

init weight matrix =

$$w_{ij} = \begin{bmatrix} 0.2 & 0.9 \\ 0.4 & 0.7 \\ 0.6 & 0.5 \\ 0.8 & 0.3 \end{bmatrix}$$

winner

First vector $X_1 = [0\ 0\ 1\ 1]$

calculate distance.

$$d_j = \sum (w_{ij} - x_i)^2$$

$$D(1) = (0.2-0)^2 + (0.4-0)^2 + (0.6-1)^2 + (0.8-1)^2$$

$$= 0.4$$

$$D(2) = (0.9-0)^2 + (0.7-0)^2 + (0.5-1)^2 + (0.3-1)^2$$

$$= 2.04.$$

D1 will be the winner.

$$w_{ij} = w_{ij} + \eta(t)(x_i - w_{ij})$$

$$w_{11} = 0.2 + 0.5(0 - 0.2) = 0.1$$

$$w_{21} = w_{21} + 0.5(x_2 - w_{21})$$

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

$$w_{31} = 0.6 + 0.5(1 - 0.6) = 0.8$$

$$w_{41} = 0.8 + 0.5(1 - 0.8) = 0.9.$$

Healthcare

Neopenem® meropenem

updated matrix =

$$(new)w_{ij} = \begin{bmatrix} 0.1 & 0.9 \\ 0.2 & 0.7 \\ 0.8 & 0.5 \\ 0.7 & 0.3 \end{bmatrix}.$$

now, second input vector X_2 using,

new w_{ij}

Hopefield autoassociative

Bipolar input patterns
$w_{ij} = \sum s_i(p)\, T_j(p)$
$w_{ij} = 0$.
weights with no self conn.

1. Init weights.
2. ~~Test~~ Target $x = [1\ 1\ 1\ 0]$ is given

   Test with 2 missing

   $[-\ -\ 1\ 0]$.

● missing values will filled up with 0.

now. $[0\ 0\ 1\ 0]$.

3. $y_i = x_i \quad [i = 1 \text{ to } n]$.

$y = [0\ 0\ 1\ 0]$.

$y_1\ y_2\ y_3\ y_4$.

update করব step 4 এর y যাতে তা target $x = [1110]$ এর মতো হয়ে যায় | will complete testing

4. Calculate net input of Network.

$y_{in} = x_i + \sum y_i w_{ji}$ $\quad y = [0\ 0\ 1\ 0]$

5. Apply Activation function.

$$y_i = \begin{cases} 1, & \text{if } y_{in} > \theta_i \\ y_i, & \text{if } y_{in} = \theta_i \\ 0, & \text{if } y_{in} < \theta_i \end{cases}$$

$\theta = 0$

6. upday y vector $[0\ 0\ 1\ 0]$ & see if it is equal to the target

$$X = [1\ 1\ 1\ 0].$$

(*) Construct an autoassociative Discrete Hopefield network with input vector [1 1 1 -1]. Test discrete Hopefield network with missing values in the 1 & 2 position of stored vector.

Soln:

Input = $X = [1 \quad 1 \quad 1 \quad -1]$.

Weight matrix, $w_{ij} = \sum s^T(p)\, t(p)$

Here, $s = t$

$$= \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

no self conn. so, $w_{ij} = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$

✓ weights are initialized

Binary Representation of given input vector is [1 1 1 0]

For missing values (1 & 2) new vector is [0 0 10]

Now, $x = [0\ 0\ 1\ 0]$ (x_1)

$Y = [0\ 0\ 1\ 0]$.

(1) — Choosing unit (y_1) for updating activation.

$$y_{in_1} = x_1 + \sum_1 y_j w_{j1}$$

$$= 0 + [0\ 0\ 1\ 0] \begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

$$= 0 + 1 = 1$$

— Applying Activation $y_{in_1} = 1 > 0$,

so, $y_1 = 1$

Now, $y = [1\ 0\ 1\ 0]$.

(2) Choosing unit y_4

$$y_{in_4} = x_4 + \sum y_j w_{j4}.$$

$$= 0 + [0\ 0\ 1\ 0] \begin{bmatrix} -1 \\ -1 \\ -1 \\ 0 \end{bmatrix}$$

$$= 0 - 1 - 1 = -2 < 0.$$

y_{in_4} Activation $= 0$.

$$y = [1\ 0\ 1\ 0].$$

③ y_3 choosing.

$$y_{in_3} = x_3 + \sum y_j w_{j3}$$

$$= 1 + [1\ 0\ 1\ 0] \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$= 2. \quad > 0.$$

Activation: $y_{in_3} = 1.$

$$y = [1\ 0\ 1\ 0]$$

④ Choosing y_2.

$$y_{in_2} = x_2 + \sum y_j w_{j2}$$

$$= 0 + [1\ 0\ 1\ 0] \begin{bmatrix} 1 \\ 0 \\ 1 \\ -1 \end{bmatrix}$$

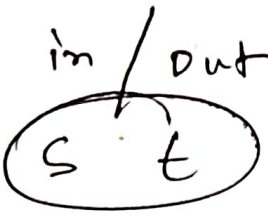
$$= 2 > 0.$$

Activations $y_{in_2} = 1.$

$$y = [1\ 1\ 1\ 0]$$

Association

- single layer net
- net can store a set of patterns.



Auto associative
$(s = t)$

Hetero

mapping 4 in | 2 out

in	output
1 0 0 0	1 0
1 1 0 0	1 0
0 0 0 1	0 1
0 0 1 1	0 1

If the input is $x = (0\,1\,0\,0)$ which is diff from train. the net still produce (10).

Adeno / Hebb

s_1	s_2	s_3	s_4
1	0	0	0
1	1	0	0
0	0	0	1
0	0	1	1

t_1	t_2
1	0
1	0
0	1
0	1

The input vectors are not ortho.

Train, set $\boxed{x_i = s_i}$.

$w_{ij} = w_{ij} + s_i t_j$ i.e. $\Delta w = s_i t_j$

Init weight $= 0$.

(1) $s : t = (1000) : (10)$.

$x_1 = 1$, $x_2 = x_3 = x_4 = 0$.

$y_1 = 1$ $y_2 = 0$

$w_{11} = w_{11} + x_1 y_1 = 1$

(2) $s : t = (11\ 00) : (10)$

$x_1 = 1$, $x_2 = 1$ $x_3 = x_4 = 0$.

$y_1 = 1$, $y_2 = 0$.

$w_{11} = w_{11} + x_1 y_1 = 2$.

1

$w_{21} = w_{21} + x_2 y_2$