Date :

Deep Learning

Evaluation Policy:

Tutorials - 25
Mid Semester - 30
End Semester - 40
Assignment/Project - 30

Book: "Deep Learning" by Ian Goodfellow, Yoshua Bengio

No. of features = No. of nodes in input layer
No. of output nodes = No. of classes

$O_j = f(w_{ij} . x_j)$

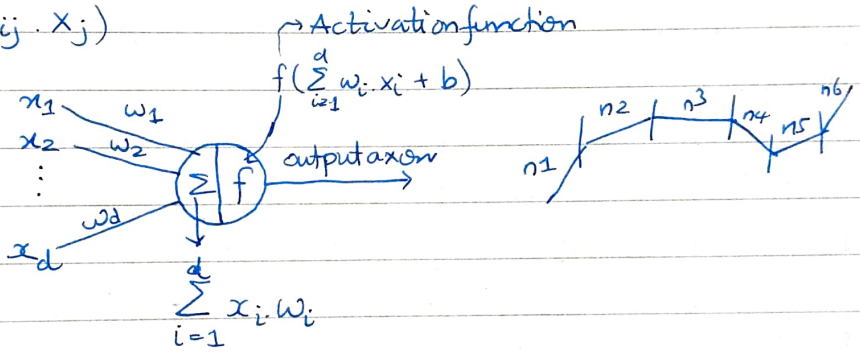


Fig.: Simple Mathematical Depiction of Neuron

$g_i(x) = w^t x + w_0$

$g_i(x) = a^t y$

$g_i(x) = [w_1\ w_2 \cdots w_d\ w_0]^t\ [x_1\ x_2 \cdots x_d\ 1]$

Neural Networks:

1) Neural Networks is a classifier
2) It is a supervised learning / training
3) (x_i, y_i), x_i - i^{th} Input feature, y_i - i^{th} class label
   ↳ target label/ expected output

Two phases - train & test - good generalization

Date :

LDF:

1) The classes are linearly separable.
2) No probability distribution information about the class is required.
3) Intercept and orientation are two important parameters. $g(x) = g_1(x) - g_2(x)$
4) The weight vector w is orthogonal to the decision boundary
5) One of the conditions to update the weight vector is called perceptron.
6) $g_i(x) = \sum w_i x_i + w_0 \Rightarrow g_i(x) > 0$ for correct classification
   - w_i: random initially
   - x_i: known
   - $g_i(x) > 0$: Perceptron criterion function / Uniform criterion

Neural Networks:

1) Changeable parameters : w
2) x_i are given as fixed inputs.
3) y_i is also fixed

$$g_i(x) = \sum_{i=1}^{d} w^t x_i + w_0 = W^t X$$

$$\boxed{g_i(x) = a^t y > 0.}$$ (Append 1 to X, w_0 to w)

↳ uniform criterion function.

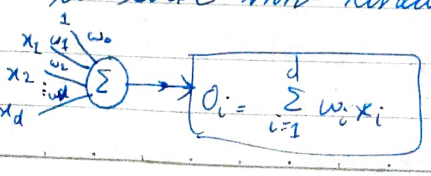
4) Perceptron criterion function

$g_i(x) > 0$

5) Relaxation criterion function

$g_i(x) > b$ → bias.

6) To depict/ get a complex non-linear classes, you need to solve non- linearly separable classes.



Date : 25/01/2024

Compare actual and expected output

$E = \sum_{i=1}^{N} (A.O. - E.O.)^2$ Min. E by changing w.

↳ cost function

Min. cost function E → correct class predicted.

7) $a[0] \Rightarrow$ randomly chosen weight vector.

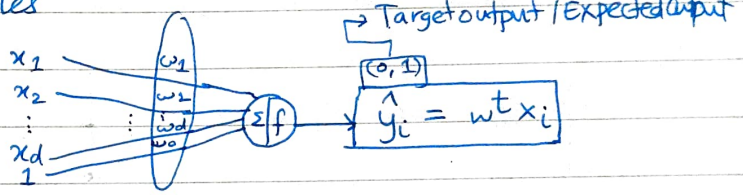
8) $a_{k+1} = a_k + \eta \sum_{\forall y \in \text{Misclassified}} y$

25/01/2024

Neural Networks (cont'd)

$i = 1, 2, \cdots, N$ ↳ no. of samples

It is supervised learning



(x_i, y_i) ↳ label

y_i : Expected output (or) target output

$\hat{y}_i$: Actual output

x_i : i^{th} Sample Feature Vector

N : Total number of samples

$E = \frac{1}{2} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$, where E is the sum of squared Error

$\frac{\partial E}{\partial w} = \frac{2}{2} \sum_{i=1}^{N} (\hat{y}_i - y_i) \cdot \frac{\partial}{\partial w} (\hat{y}_i - y_i)$ ↳ target output

$\frac{\partial E}{\partial w} = \sum_{i=1}^{N} (\hat{y}_i - y_i) \cdot \frac{\partial}{\partial w} (w^t x_i)$

$\frac{\partial E}{\partial w} = \sum_{i=1}^{N} (\hat{y}_i - y_i) \cdot x_i$

Date :

Weight updation Rule:

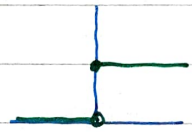
$$W_{new} \leftarrow W - \eta \cdot \frac{\partial E}{\partial w} \quad (\nabla E)$$

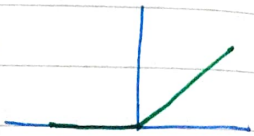
$$\boxed{W_{new} \leftarrow W_{old} - \eta \sum_{i=1}^{N} (\hat{y}_i - y_i) \cdot x_i}$$

Repeat until all samples are correctly classified.

$a^t y > 0$: Uniform criterion function.

SLSO- Single Layer Single Output is used for simple perceptron

Step function: 

ReLU function: 

Training in Neural Network (Fill in the Blanks):

1) In this, the changeable parameter is weight vector(w)
2) In neural network, we should have some weight updation rule
3) The weight updation has to be done using training sample
4) Keep on updating weight, until all samples are correctly classified
5) Since it is supervised learning, we know expected / target output
6) The actual output ~~predi~~ computed by NN is called $\hat{y}$ (predicted output)
7) The difference between A/P value and T/E value is error.
8) The convergence is until all samples are correctly classified at once
9) Single output NN takes care of two-class problem
10) But if multiple class problem is involved, then we should have neural network having a single layer multiple output units
11) O/P N.L. function is called activation function.
12) Non-linear function used is step function
13) The step function is used usually only in output layer.

Date : 30/1/2024

14) The same function cannot be used in hidden layers as it is not differentiable because there is an abrupt change at T=0

15) otherwise, sigmoid function is used at hidden layer.

30/1/2024

Weight updation rule

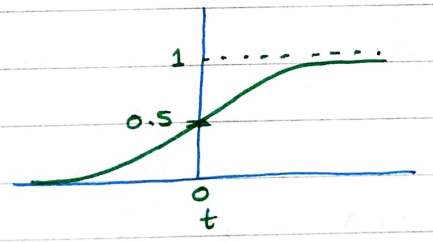
$$W_{new} \leftarrow W_{old} - \eta \frac{\partial E}{\partial W}$$

$$W_{new} \leftarrow W_{old} - \eta \sum_{i=1}^{N} (\hat{y}_i - y_i) \cdot x_i$$ ← Batch optimization

$$W_{new} \leftarrow W_{old} - \eta (\hat{y}_i - y_i) \cdot x_i$$ ← Stochastic optimization

The step function is not differentiable, so we can use the sigmoid activation function.

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



$$\sigma(s) = \frac{1}{1+e^{-s}}, \quad s = w^t \cdot x$$

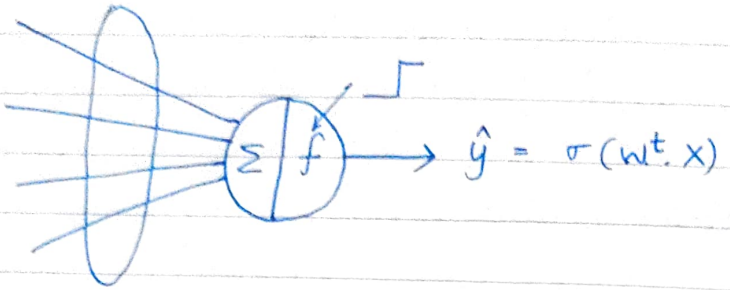
when $s = 0$; $\frac{1}{1+e^{-0}} = 0.5$

when $s \to \infty$; $\frac{1}{1+e^{-\infty}} \approx 1$

when $s \to -\infty$; $\frac{1}{1+e^{\infty}} \approx 0$

Date :

SLSO with sigmoid as non-linear function



The derivative of sigmoidal function is $\sigma(s)(1-\sigma(s))$

$$\boxed{\sigma'(s) = \sigma(s)\,(1-\sigma(s))}$$

(x_i, y_i) ← target value

$$E = \frac{1}{2}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

$$\frac{\partial E}{\partial w} = \frac{2}{2}\sum_{i=1}^{N}(\hat{y}_i - y_i)\cdot\frac{\partial}{\partial w}(\hat{y}_i - y_i)$$

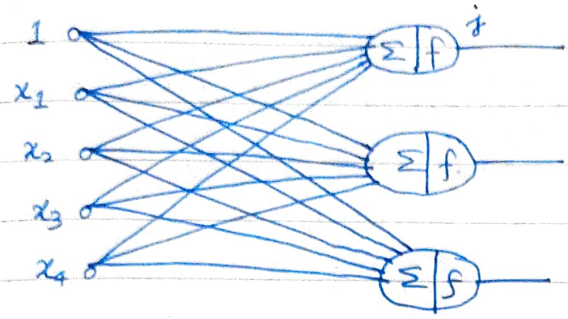
$$\frac{\partial E}{\partial w} = \sum_{i=1}^{N}(\hat{y}_i - y_i)\cdot\frac{\partial}{\partial w}(\hat{y}_i)$$

$$\frac{\partial E}{\partial w} = \sum_{i=1}^{N}(\hat{y}_i - y_i)\cdot\frac{\partial}{\partial w}(\overbrace{\sigma(w^t.x_i)}^{\sigma(\hat{y}_i)})$$

$$\frac{\partial E}{\partial w} = \sum_{i=1}^{N}(\hat{y}_i - y_i)\cdot x_i\,(\hat{y}_i\,(1-\hat{y}_i))$$

Date : 31/1/2024

SLMO (Single Layer Multiple Output)



$$o_j = f\left(\sum_{i=1}^{d} w_{ij} \cdot x_i\right)$$ ← Sigmoidal function

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial \theta_j} \cdot \frac{\partial \theta_j}{\partial w_{ij}}$$

$$o_j = \frac{1}{1+e^{-\theta_j}}, \quad \theta_j = \sum_{i=1}^{d} w_{ij} \cdot x_i$$

o_j - Predicted value / Actual Value

(x_i, y_i) t_j - target output

x_i - Feature vector

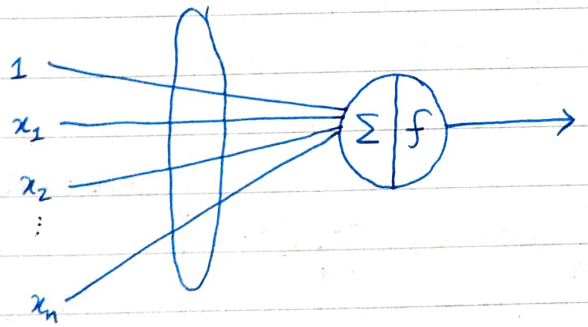
f - Sigmoid function

w_{ij} - Weight from i^{th} neuron (input layer) to j^{th} neuron

$$E = \sum_{j=1}^{c} (o_j - t_j)^2, \text{ where } c \text{ is the number of classes.}$$

31/1/2024

Recap (SLSO)



$$\hat{y}_i = \sigma(w^t \cdot x_i)$$

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

$$\sigma(w^t \cdot x) = \frac{1}{1+e^{-w^t \cdot x}} \; ; \; \frac{\partial}{\partial s}(\sigma(s)) = \sigma(s)(1-\sigma(s))$$

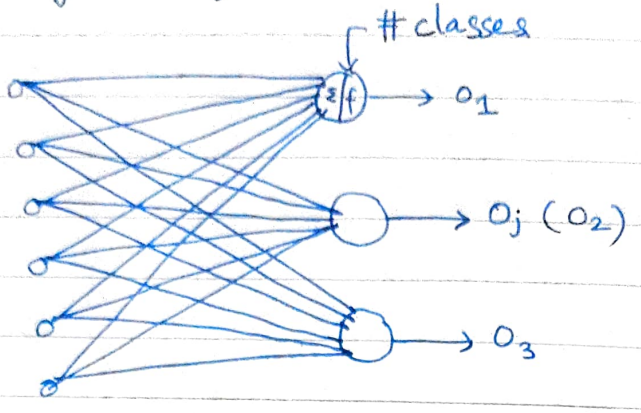
$$\sigma'(w^t \cdot x) = \sigma(w^t \cdot x) \cdot (1 - \sigma(w^t \cdot x)) \cdot x$$

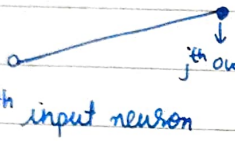
Chain rule:

$$y = f(g(x))$$
$$y' = f'(g(x))\, g'(x)$$

Date :

SLMO (Single Layer Multiple Output)





$$O_j = \sigma\left(\sum_{i=1}^{d} w_{ij} x_i\right), \quad \theta_j = \sum_{i=1}^{d} \omega_{ij} \cdot x_i, \quad O_j = \frac{1}{1+e^{-\theta_j}}$$

Non-linear function - activation function.

Hidden layer - No restriction on number of nodes.

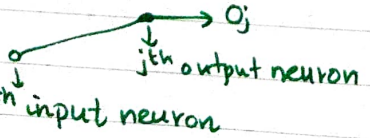
$i = 1, 2, \ldots, d$; $j = 1, 2, \ldots, C$, where d: dimension of F.V.

C : Number of classes

$$E = \frac{1}{2} \sum_{j=1}^{C} (O_j - t_j)^2, \text{ where } t_j \text{ or } y_i \text{ is called target output.}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{2}{2} \sum_{j=1}^{C} (O_j - t_j) \cdot \frac{\partial}{\partial w} (O_j - t_j) \quad \text{(constant)}$$

$$\frac{\partial E}{\partial w_{ij}} = \sum_{j=1}^{C} (O_j - t_j) \cdot \frac{\partial}{\partial w} (O_j)$$



$$O_j = \sigma\left(\sum_{i=1}^{d} \omega_{ij} \cdot x_i\right)$$

$$O_j = \frac{1}{1+e^{-\theta_j}}, \text{ where } \theta_j = \sum_{i=1}^{d} \omega_{ij} \cdot x_i$$

Date :

$$\frac{\partial E}{\partial w_{ij}} = \sum_{j=1}^{c} (o_j - t_j) \cdot \frac{\partial}{\partial w} (o_j)$$

$$\frac{\partial E}{\partial w_{ij}} = \sum_{j=1}^{c} (o_j - t_j) \cdot \frac{\partial o_j}{\partial \theta_j} \cdot \frac{\partial \theta_j}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = \sum_{j=1}^{c} (o_j - t_j) \cdot o_j(1-o_j) \cdot x_i \quad (\because \sigma'(s) = \sigma(s)(1-\sigma(s)))$$

Weight updation rule :

$$w_{new} \leftarrow w_{old} - \eta \frac{\partial E}{\partial w_{ij}}$$

$$\boxed{w_{new} \leftarrow w_{old} - \eta \sum_{j=1}^{c} (o_j - t_j)\, o_j(1-o_j)\, x_i}$$

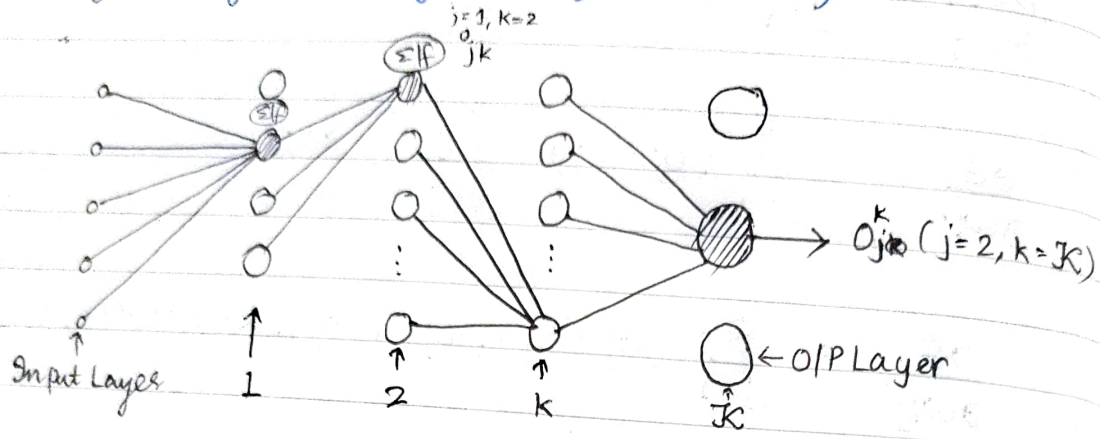
Repeat for 'n' iterations.

Shape of o_j = Shape of $(w_{ij} \cdot x_i)$ $\quad (1 \times (d+1))$

$(1 \times d)$ $(1 \times (d+1))$

Date :

Multiple Layer Multiple Output (MLMO)

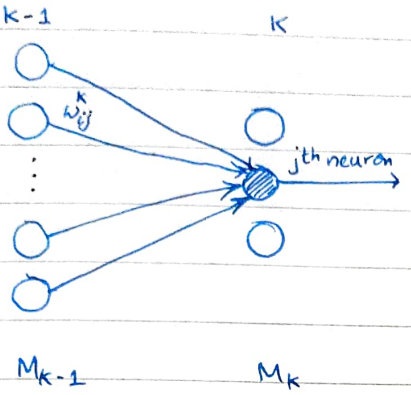


O_j^k - Output of j^{th} neuron at k^{th} layer.

Date : 1/2/2024

MLMO (Multiple Layer Multiple output)
Neural Network (or) Multilayer Perceptron

weight updation rule at 'output layer' k to k-1.



$$O_j^k = f\left(\sum_{i=0}^{M_{k-1}} w_{ij}^k \cdot x_i^{k-1}\right) \quad \sigma$$

$$O_j = \sigma\left(\sum_{i=1}^{M_{k-1}} w_{ij}^k \cdot O_i^{k-1}\right)$$

(C) no. of classes.

M_k : No. of nodes at k^{th} layer
M_{k-1} : No. of nodes at $(k-1)^{th}$ layer
O_j^k : Output at j^{th} neuron at k^{th} layer
w_{ij}^k : Weight between i^{th} and j^{th} neuron to k^{th} layer
t_j^k : target value.
O_i^{k-1} : O/P obtained at i^{th} neuron in $(k-1)^{th}$ layer.

$$E = O_j^k - t_j^k$$

$$O_j^k = \frac{1}{1+e^{-\theta_j}} \; ; \; \theta_j^k = \sum_{i=1}^{M_{k-1}} w_{ij}^k \cdot O_i^{k-1}$$

$$E = \frac{1}{2} \sum_{j=1}^{M_k} (O_j^k - t_j^k)^2$$

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{2}{2} \sum_{j=1}^{M_k} (O_j^k - t_j^k) \cdot \frac{\partial}{\partial w_{ij}^k} (O_j^k - \boxed{t_j^k}) \quad \text{fixed}$$

$$\Rightarrow \frac{\partial E}{\partial w_{ij}^k} = \sum_{j=1}^{M_k} (O_j^k - t_j^k) \cdot \frac{\partial O_j^k}{\partial \theta_j^k} \cdot \frac{\partial \theta_j^k}{\partial w_{ij}^k}$$

Date : 1/2/2024

$$\Rightarrow \frac{\partial E}{\partial w_{ij}^k} = \sum_{j=1}^{M_k} \underbrace{(o_j^k - t_j^k) \cdot o_j^k (1 - o_j^k)}_{\delta_j^k} \; o_i^{k-1}$$

$$\because \frac{\partial o_j^k}{\partial \theta_j^k} = \frac{\partial}{\partial \theta_j^k} \left(\frac{1}{1 + e^{-\theta_j^k}} \right) = o_j^k (1 - o_j^k)$$

$$\because \sigma'(s) = \sigma(s)(1 - \sigma(s)), \; s = o_j^k$$

$$\frac{\partial \theta_i^k}{\partial w_{ij}^k} = \frac{\partial}{\partial w_{ij}^k} \left(\sum_{i=1}^{M_{k-1}} w_{ij}^k \cdot o_i^{k-1} \right) = o_i^{k-1}$$

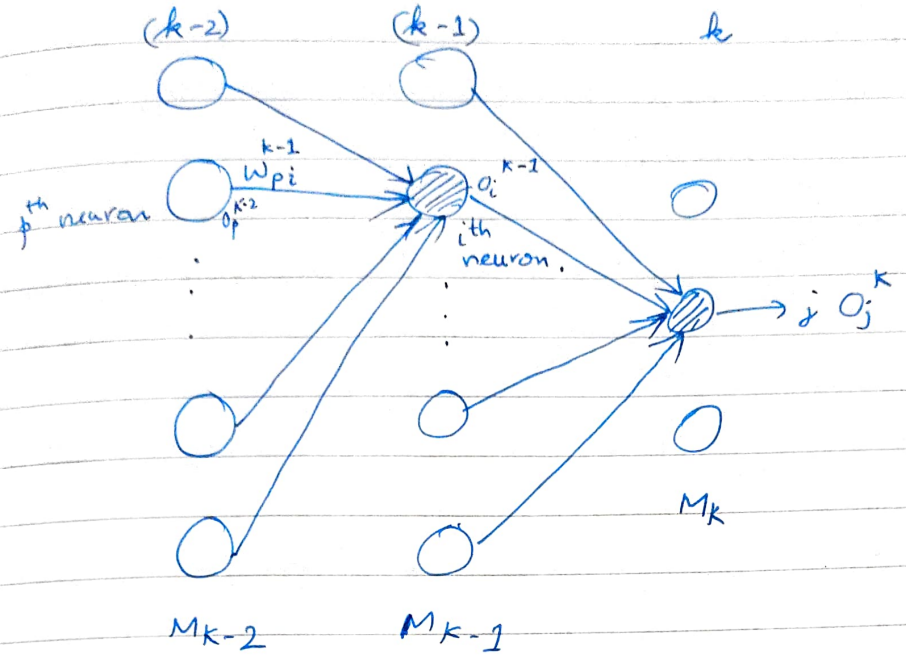
$$\frac{\partial E}{\partial w_{ij}^k} = \sum_{i=1}^{M_{k-1}} \sum_{j=1}^{M_k} \underbrace{\delta_j^k}_{\text{Backprop. intuition.}} \cdot o_i^{k-1}$$

The error obtained in j^{th} neuron of k^{th} layer is backpropagated to all i neurons in $(k-1)^{th}$ layer connected to j^{th} neuron of k^{th} layer.

$$w_{ij}^k{}_{new} \leftarrow w_{ij}^k{}_{old} - \eta \, \delta_j^k \cdot o_i^{k-1}$$

Date :

Weight updation rule between hidden layer (k-1) to (k-2)



$$o_i^{k-1} = \frac{1}{1+e^{-\theta_i^{k-1}}} \; ; \; \theta_i^{k-1} = \sum_{p=1}^{M_{k-2}} w_{pi}^{k-1} \cdot o_p^{k-2}$$

~~$E = \frac{1}{2}\left(\sum_{i=1}^{M_{k-1}} (o_i^{k-1} - t_j^{k-1})^2\right)$~~

$$E = \frac{1}{2} \sum_{j=1}^{M_k} (o_j^k - t_j^k)^2$$

$$\frac{\partial E}{\partial w_{pi}^{k-1}} = \frac{2}{2} \sum_{j=1}^{M_k} (o_j^k - t_j^k) \cdot \frac{\partial}{\partial w_{pi}^{k-1}} (o_j^k - \boxed{t_j^k})$$ (t_j^k: fixed)

$$= \sum_{j=1}^{M_k} (o_j^k - t_j^k) \cdot \frac{\partial o_j^k}{\partial \theta_j^k} \cdot \frac{\partial \theta_j^k}{\partial o_i^{k-1}} \cdot \frac{\partial o_i^{k-1}}{\partial \theta_i^{k-1}} \cdot \frac{\partial \theta_i^{k-1}}{\partial w_{pi}^{k-1}}$$

$$\frac{\partial E}{\partial w_{pi}^{k-1}} = \sum_{j=1}^{M_k} \underbrace{(o_j^k - t_j^k) \cdot o_j^k (1-o_j^k) \cdot w_{ji}^k \cdot o_i^{k-1}(1-o_i^{k-1})}_{\delta_j^k} \cdot o_p^{k-2}$$

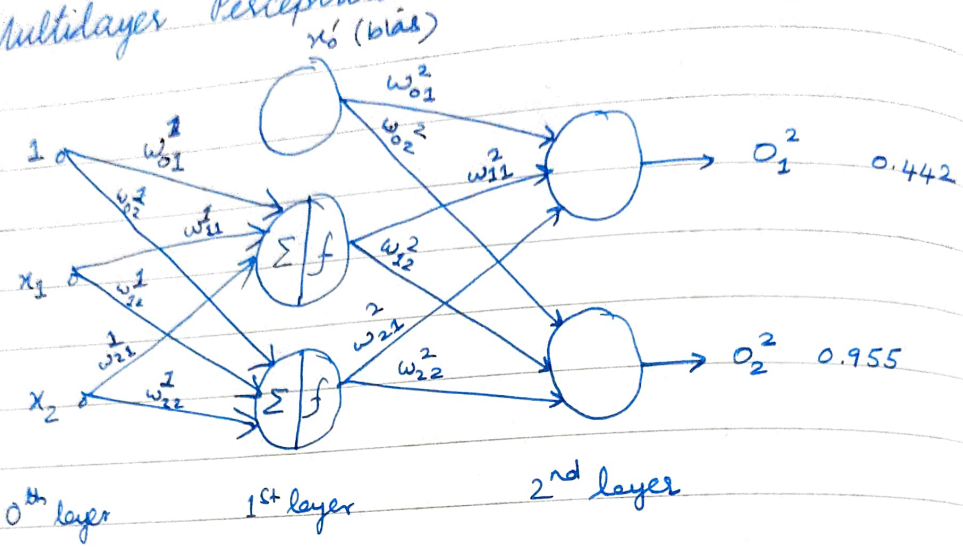
$$\delta_i^{k-1} \cdot o_p^{k-2}$$

Error from i^{th} neuron of $k-1^{th}$ layer is passed to p^{th} neuron of $(k-2)^{th}$ layer

$$w_{pi}^{k-1} = w_{pi}^{k-1} - \eta \, \delta_i^{k-1} \cdot o_p^{k-2}$$

2/2/2024

Multilayer Perceptron - Problem Solving



do
1. Feed forward pass
2. Back propagate pass (derivative of error)
3. Update the weight

until 'E' is minimized.

[Similar qn. for midsem]

$$\begin{bmatrix} w_{01}^1 & w_{11}^1 & w_{21}^1 \\ w_{02}^1 & w_{12}^1 & w_{22}^1 \end{bmatrix} \begin{bmatrix} w_{01}^2 & w_{11}^2 & w_{21}^2 \\ w_{02}^2 & w_{12}^2 & w_{22}^2 \end{bmatrix} \quad X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \quad t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0.5 & 1.5 & 0.8 \\ 0.8 & 0.2 & -1.6 \end{bmatrix} \begin{bmatrix} 0.9 & -1.7 & 1.6 \\ 1.2 & 2.1 & -0.2 \end{bmatrix} \quad X = \begin{bmatrix} 1 \\ 0.7 \\ 1.2 \end{bmatrix} \quad t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

1) Feed forward pass [layer '0' to layer '1']

$$\begin{bmatrix} 0.5 & 1.5 & 0.8 \\ 0.8 & 0.2 & -1.6 \end{bmatrix} \begin{bmatrix} 1 \\ 0.7 \\ 1.2 \end{bmatrix} = \begin{bmatrix} 2.51 \\ -0.98 \end{bmatrix}$$

Date :

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

$$\Rightarrow \begin{bmatrix} \frac{1}{1+e^{-2.51}} \\ \frac{1}{1+e^{-(0.98)}} \end{bmatrix} = \begin{bmatrix} 0.92 \rightarrow x_1^1 / o_1^1 \\ 0.27 \rightarrow x_2^1 / o_2^1 \end{bmatrix}$$

Feed forward from layer 1 to 2.

$$\begin{bmatrix} 0.9 & -1.7 & 1.6 \\ 1.2 & 2.1 & -0.2 \end{bmatrix} \begin{bmatrix} 1 \\ 0.92 \\ 0.27 \end{bmatrix} = \begin{bmatrix} -0.232 \\ 3.057 \end{bmatrix}$$

$$\text{sigmoid}\left(\begin{bmatrix} -0.232 \\ 3.057 \end{bmatrix}\right) = \begin{bmatrix} 0.442 \leftarrow o_1^2 \\ 0.955 \leftarrow o_2^2 \end{bmatrix}$$

Backpropagation from output layer(k) to previous layer(k-1)

$$w_{new} \leftarrow w_{old} - \eta \cdot \frac{\partial E}{\partial w}$$

$$w_{new} \leftarrow w_{old} - \eta \cdot \delta_j^k \cdot o_i^{k-1}$$

$$\delta_j^k = o_j^k (1-o_j^k) \cdot (o_j^k - t_j^k)$$

Find $\delta_1^2 - o_1^1$ and $\delta_2^2 - o_2^2$

$$\delta_1^2 = o_1^2 (1-o_1^2) \cdot (o_1^2 - t_1^2)$$

$$= 0.442(1-0.442)(0.442-1) = -0.138.$$

$$\boxed{\delta_1^2 = -0.14}$$

$$\delta_2^2 = O_2^2(1-O_2^2)(O_2^2 - t_2^2)$$

$$\delta_2^2 = 0.955\,(1-0.955)\,(0.955-0)$$

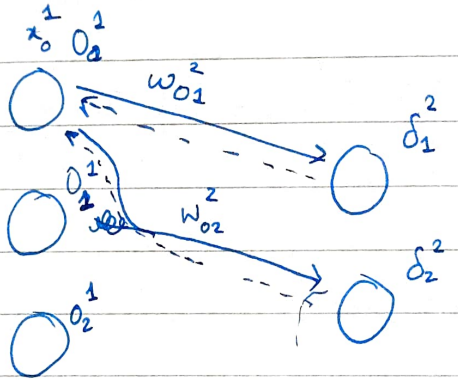
$$\boxed{\delta_2^2 = 0.045}$$

$$\boxed{\delta_j^k = O_j^k(1-O_j^k)(O_j^k - t_j^k)}$$

$$\boxed{\frac{\partial E}{\partial w} = \delta_j^k \cdot O_i^{k-1}}$$

Formulae.

$$\frac{\partial E}{\partial w_{01}^2} = \delta_1^2 \cdot O_0^1 = -0.138 \times 1 = -0.138$$



Reference Diagram

$$\frac{\partial E}{\partial w_{02}^2} = \delta_2^2 \cdot O_0^1 = 0.045 \times 1 = 0.045$$

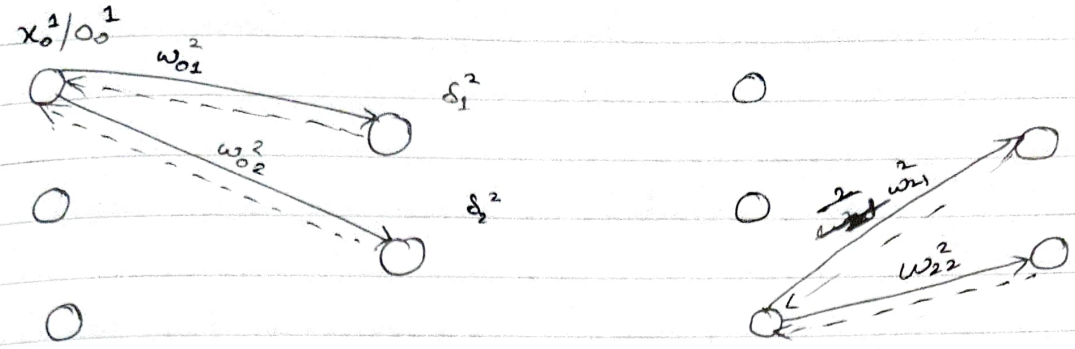
$$\frac{\partial E}{\partial w_{11}^2} = \delta_1^2 \cdot O_1^1 = -0.138 \times 0.92 \simeq -0.127$$

$$\frac{\partial E}{\partial w_{12}^2} = \delta_2^2 \cdot O_1^1 = 0.045 \times 0.92 = 0.0414$$

$$\frac{\partial E}{\partial w_{21}^2} = \delta_1^2 \cdot O_2^1 = -0.138 \times 0.27 = -0.037$$

$$\frac{\partial E}{\partial w_{22}^2} = \delta_2^2 \cdot O_2^1 = 0.045 \times 0.27 = 0.012$$

Date :



Update the respective weights :-

$$W_{new} \leftarrow W_{old} - \eta \cdot \frac{\partial E}{\partial w}, \quad \boxed{\eta = 0.5}$$

$$w_{01}^2 \leftarrow w_{01}^2 - \eta \cdot \frac{\partial E}{\partial w_{01}^2} = 0.9 - 0.5 \times -0.138 = 0.969.$$

$$w_{11}^2 \leftarrow w_{11}^2 - \eta \cdot \frac{\partial E}{\partial w_{11}^2} = -1.7 - 0.5 \times -0.126 = -1.637.$$

$$w_{02}^2 \leftarrow w_{02}^2 - \eta \cdot \frac{\partial E}{\partial w_{02}^2} = 1.2 - 0.5 \times 0.045 = 1.777$$

$$w_{12}^2 \leftarrow w_{12}^2 - \eta \cdot \frac{\partial E}{\partial w_{12}^2} = 2.1 - 0.5 \times 0.04 = 2.07$$

$$w_{21}^2 \leftarrow w_{21}^2 - \eta \cdot \frac{\partial E}{\partial w_{21}^2} = 1.6 - 0.5 \times -0.037 = 1.618$$

$$w_{22}^2 \leftarrow w_{22}^2 - \eta \cdot \frac{\partial E}{\partial w_{22}^2} = -0.2 - 0.5 \times 0.012 = -0.206$$

Next, we have to update the weights in layer 1.

$\frac{\partial E}{\partial w_{01}^2}, \frac{\partial E}{\partial w_{11}^2}, \frac{\partial E}{\partial w_{21}^2}, \frac{\partial E}{\partial w_{02}^2}, \frac{\partial E}{\partial w_{12}^2}, \frac{\partial E}{\partial w_{22}^2}$

-0.138 — w_{01}
-0.126 — w_{11}
-0.037 — w_{21}
0.03989 —
0.0369 —
0.01088 —

Date :

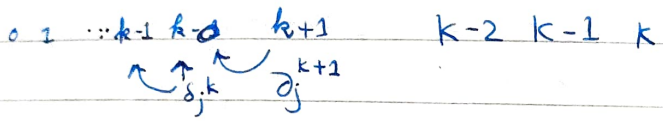
Algorithm for back propagation at any hidden layer 'k':

(1) For any hidden layer weight w_{ij}^{k},

$$\frac{\delta E}{\delta w_{ij}^{k}} = \delta_j^{k} \cdot O_i^{k-1}$$

Error is back propagated to (k-1) layer.

(2) $$\delta_i^{k} = \left(\sum_{j=1}^{M_{k+1}} \delta_j^{k+1} \cdot w_{ij}^{k+1} \right) O_i^{k} \left(1 - O_i^{k}\right)$$

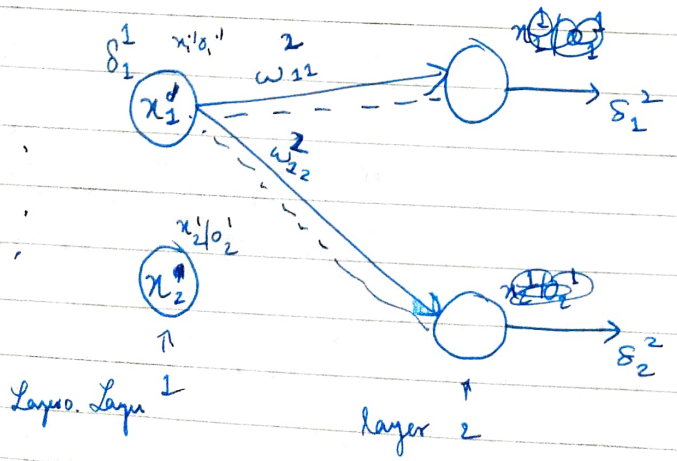


(3) Weight updation rule for any hidden layer

$$w_{ij}^{k} \leftarrow w_{ij}^{k} - \eta \frac{\partial E}{\partial w_{ij}^{k}}$$

$$\frac{\partial E}{\partial w_{ij}^{k}} = \delta_j^{k} \cdot O_i^{k-1}$$

Back propagation from k-1 to k-2 (1 to 0)



$\delta_1^2 = -0.137$
$O_1^1 = 0.92$

$\delta_2^2 = 0.045$
$O_2^1 = 0.27$

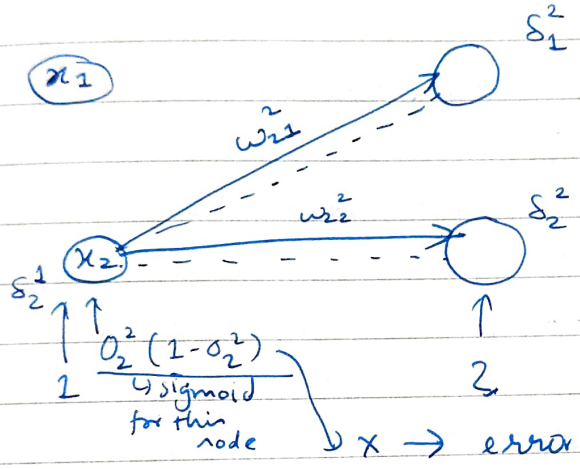
Date :

$$\delta_1^1 = O_1^1(1-O_1^1)\cdot\left[\delta_1^2 \cdot w_{11}^2 + \delta_2^2 \cdot w_{12}^2\right]$$

First find error at 1st layer!.

$$\delta_1^1 = 0.92(1-0.92)\left[-0.137 \times -1.7 + 0.045 \times 2.1\right]$$

$$\boxed{\delta_1^1 = 0.0241.}$$



$$\delta_2^1 = 0.27(1-0.27)\left[\underbrace{-0.137}_{\delta_1^2} \times \underbrace{1.6}_{w_{21}^2} + \underbrace{0.045}_{\delta_2^2} \times \underbrace{-0.2}_{w_{22}^2}\right]$$

$$\boxed{\delta_2^1 = -0.044}$$

$$\frac{\partial E}{\partial w_{01}^1} = \delta_1^1 \cdot x_0^0 \quad (O_0^0) = 0.024 \times 1 = 0.024$$

$$\frac{\partial E}{\partial w_{02}^1} = \delta_2^1 \cdot x_0^0 = -0.04 \times 1 = -0.04$$

$$\frac{\partial E}{\partial w_{11}^1} = \delta_1^1 \cdot x_1^0 = 0.024 \times 0.7 = 0.017$$

Date :

$$\frac{\partial E}{\partial w_{12}^1} = \delta_2^1 \cdot O_1^0 = \delta_2^1 \cdot x_1^0 = -0.04 \times 0.7 = -0.028$$

$$\frac{\partial E}{\partial w_{21}^1} = \delta_1^1 \cdot x_2^0 = 0.024 \times 1.2 = 0.0288$$

$$\frac{\partial E}{\partial w_{22}^1} = \delta_2^1 \cdot x_2^0 = -0.04 \times 1.2 = -0.048$$

Unit 5- PRML

$$w_{01}^1 \leftarrow w_{01}^1 - \eta \cdot \frac{\partial E}{\partial w_{01}^1}$$

$$w_{01}^1 \leftarrow 0.5 - 0.5 \times 0.024 = 0.488$$

$$w_{02}^1 = w_{02}^1 - \eta \cdot \frac{\partial E}{\partial w_{02}^1} = 0.8 - 0.5 \times -0.04 = 0.82$$

$$w_{11}^1 = 1.5 - 0.5 \times 0.017 = +1.4915$$

$$w_{12}^1 = 0.2 - 0.5 \times -0.028 = 0.214$$

$$w_{21}^1 = 0.8 - 0.5 \times 0.0288 = -0.786$$

$$w_{22}^1 = 1.6 - 0.5 \times -0.048 = -1.594 \;/\; -1.576.$$

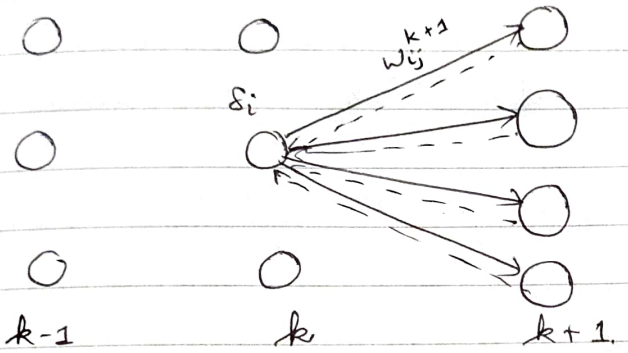
Date: 6/2/2024

Back propagation algorithm for any hidden layer 'k':

w_{ij}^k is the weight connecting for any hidden layer 'k'.

① Calculate δ_i^k as follows:

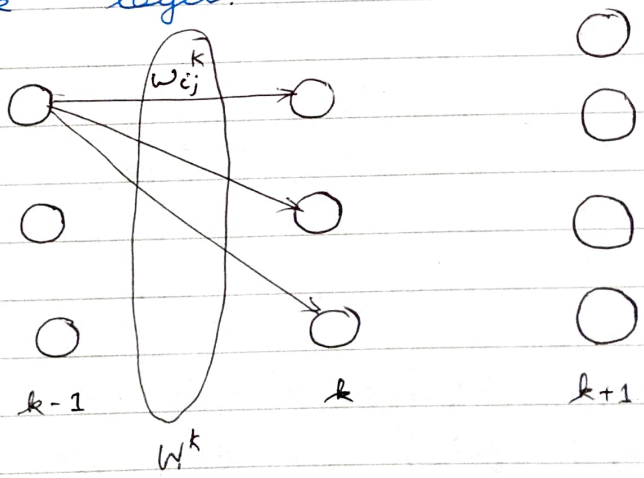
$$\delta_i^k = o_i^k (1 - o_i^k) \sum_{j=1}^{M_{k+1}} \delta_j \cdot w_{ij}^{k+1}$$



② Update the weight w_{ij}^k as follows:

$$w_{ij}^k \leftarrow w_{ij}^k - \eta \cdot \frac{\partial E}{\partial w_{ij}^k}$$

③ $k-1^{th}$ to k^{th} layer.



$$w_{ij}^k = w_{ij}^k - \eta \cdot \delta_j^k \cdot o_i^{k-1}$$

Date :

① Feed forward pass

$$f\left(\sum_{i=1}^{d} x_i \cdot w_i\right)$$

② Back propagation at output layer

$$\delta_i^k = (o_i^k - t_i^k)\, o_i^k (1 - o_i^k)$$

③ Back propagation at hidden layer 'k'

$$\delta_i^k = o_i^k (1 - o_i^k) \cdot \sum_{j=1}^{M_{k+1}} \delta_j^{k+1} w_{ij}^{k+1}$$

$$w_{ij}^k \leftarrow w_{ij}^k - \eta \cdot \frac{\partial E}{\partial w_{ij}^k}$$

$$\Rightarrow w_{ij}^k \leftarrow w_{ij}^k - \eta \cdot \delta_j^k \cdot o_i^{k-1}$$

i → left, j → right layer (notation).

Simple NN for linear separable problem:

F.V. x_1	F.V. x_2	Target output y
0	0	0
0	1	0
1	0	0
1	1	1

(# samples)

$d = 2$

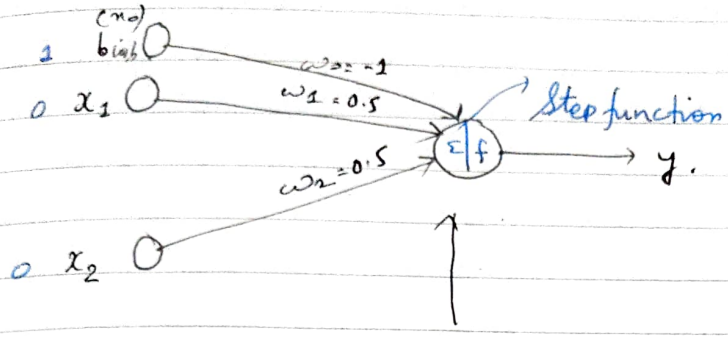
$N = 4$, # samples.

① # Neurons in I/P layer depends on no. of F.V. = 2 (here)
② # Neurons in O/P layer depends on no. of classes = 1 (bi-class)
③ Decide the number of layers required.

Date :

Initialize weight vector

Decide non-linear function / activation function.

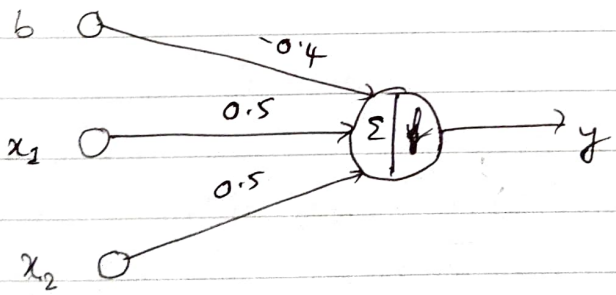


Input Layer (0^{th} Layer) Output layer.

Bias = 1 (initially)

Draw perceptron for OR gate

x	y	x OR y
0	0	0
0	1	1
1	0	1
1	1	1



X-OR gate

$\bar{x}_1 x_2 + x_1 \bar{x}_2$ $\quad \bar{x}_1 \bar{x}_2 + x_1 x_2$

~~$x_1 \bar{x}_2 + \bar{x}_1 x_2$~~

(Non-linearly separable)

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

$h1 = x_1 + x_2$ OR $\quad h2 = \overline{x_1 \cdot x_2}$ NAND $\quad h1 \cdot h2$ AND

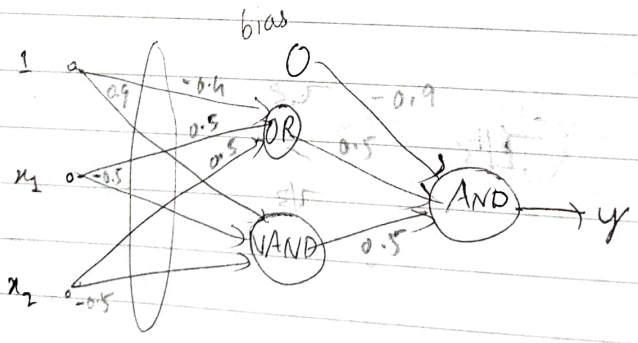
0	1	0
1	1	1
1	1	1
1	0	0

$w \cdot x \qquad \sum_{i=1}^{d} w_i \cdot \phi(x)$

$w \cdot \phi(x)$

If samples are not linearly separable in low-dim. space is transformed to higher dim. space to make it linearly separable.

Draw neural network for the same. (H/W)



Find the weight vectors

Date :

Data Collection

Pre-processing

Feature Extraction } "Iris" dataset | Dimensionality Reduction / SIFT / SURF

LOD Inter Discriminative features

hand crafted features

Training Testing → Classification → To build a classifier → ML

Types

- S.L.
- U.S.L.
- R.L

Bayes' Classifier

KNN | N.N.(S.L.)

LOF. SVM

DT

RF

Data collection → I/P

Types of learning algo.

S.L
U.S.L
R.L.

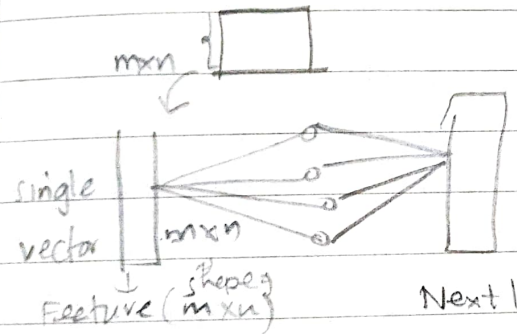
F.E. + Classification → DL Model.

→ O/P

→ Neural Networks. (Only this for DL)

→ autoEncoders.

Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN)



GPLUS

All pixel intensity values - feature (Curse of Dimensionality).

ANN - Just extracts from given features. (Nothing new).

CNN does extraction of features.

ANN & CNN - Feed Forword N.N.

Recurrent Neural Network