Day 4:

MULTI-CLASS CROSS ENTROPY

j →

i	Animal	Door 1	Door 2	Door 3
1	Duck	0.7 P_{11}	0.3 P_{12}	0.1 P_{13}
2	Bear	0.2 P_{21}	0.4 P_{22}	0.5 P_{23}
3	walrus	0.1 P_{31}	0.3 P_{32}	0.4 P_{33}

Suppose assume.

Door 1	Door 2	Door 3
Duck	walrus	walrus
0.7	0.3	0.4

$P = 0.7 * 0.3 * 0.4$

$= 0.084$

$CE = -\ln(0.7) - \ln(0.3) - \ln(0.4)$

CE = 2.48

conclusion.

room_no

$y_1(j)$ = Duck = 1 else 0

$y_2(j)$ = Beaver = 1 else 0

$y_3(j)$ = walrus = 1 else 0

$$\text{cross Entropy} = -\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} \cdot \log(P_{ij})$$

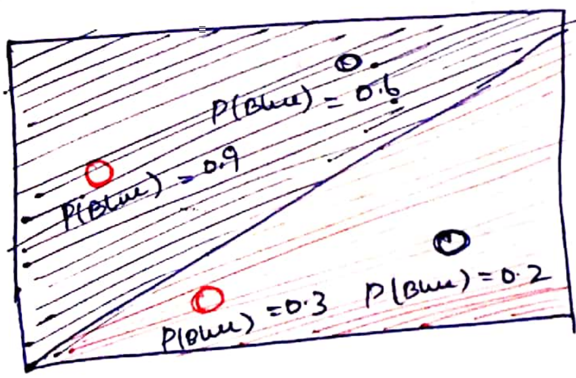
⇒ adding the log of prob of events that actually have occurred

[m → no of classes) animals

n → Door no.

LOGISTIC REGRESSION

Calculating the Error function



If y = 1 (our intention is to minimize the error function

$P(\text{Blue}) = \hat{y}$

$\text{Error} = -\log(\hat{y})$

$P(\text{red}) = 1 - P(\text{Blue})$

$= 1 - \hat{y}$

$\text{Error} \quad -\log(1-\hat{y})$

} combining those to produce a generalized one

$$\text{Error} = -\left((1-y)\log(1-\hat{y}) + y\log(\hat{y})\right)$$

If y = 0

If y = 1

$$\text{Error} = -\frac{1}{m}\sum_{i=1}^{m} (1-y_i)\log(1-\hat{y_i}) + y_i \log(\hat{y_i})$$

↑ generalized for all data points

If $\hat{y_i} \Rightarrow \sigma(wx+b)$

$$\text{Error} = \frac{-1}{m}\sum_{i=1}^{m} (1-y_i)\log(1-\sigma(wx_i+b)) + y_i \log(\sigma(wx_i+b))$$

Binary ↗

If we have multi class prob we have to go for multi-class Entropy

$$\Rightarrow \boxed{\frac{-1}{m}\sum_{i=1}^{m}\sum_{i=1}^{n} y_{ij}\log(\hat{y_{ij}})}$$

After minimizing the error function we are proceeding with Gradient Descent with smaller heights making it as w', b' new weights.

$\sigma(wx_i'+b')$

Gradient Descent.

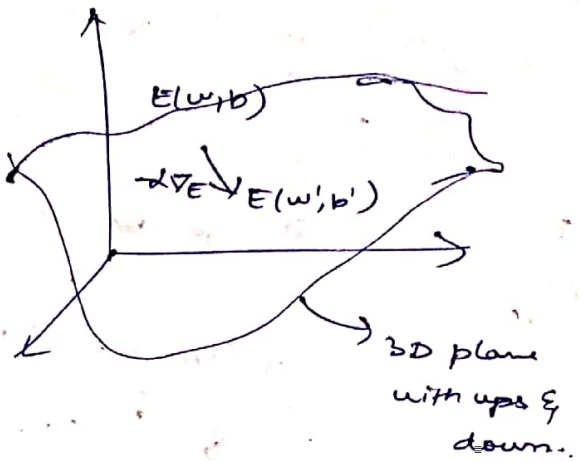
$\hat{y} = \sigma(wx+b)$ ← bad

$\hat{y} = \sigma(w_1x_1 + \dots + w_nx_n + b)$

$\nabla E = \left(\frac{\partial E}{\partial w_1}, \dots \frac{\partial E}{\partial w_n}, \frac{\partial E}{\partial b}\right)$

$\alpha = 0.1$

$$w_i' \leftarrow w_i - \alpha\left(\frac{\partial E}{\partial w_i}\right)$$
$$b_i' \leftarrow b - \alpha\left(\partial E/\partial b\right)$$
$$\hat{y} \leftarrow \sigma(w'x + b')$$



Gradient Calc for sigmoid function:

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

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

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

$$\sigma'(x) = \sigma(x) * (1 - \sigma(x))$$

If we have m points for binary Classification.

$$E = \frac{-1}{m}\sum_{i=1}^{m}\left(y_i \log(\hat{y_i})\right) + \left((1-y_i)\log(1-\hat{y_i})\right)$$

Error

[WKT: $\hat{y_i} = wx_i^{(i)} + b$]

Our goal is to calc

Error at a point x_i $[x_1, x_2, \dots x_n]$.

$$\nabla E = \left(\frac{\partial}{\partial w_i}E, \dots \frac{\partial}{\partial w_n}E, \frac{\partial}{\partial w_b}E\right)$$

→ Error each pt produce & calculate derivate of the error & avrge of error of all pts

$$E = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$ ←

Inorder to calculate the derivative of this error with respective to weights, we'll first calculate $\frac{\partial}{\partial w_j} \hat{y}$.

$$\hat{y} = \sigma(wx+b)$$

$$\frac{\partial}{\partial w_j} \hat{y} = \frac{\partial}{\partial w_j} \sigma(wx+b)$$

$$= \sigma(wx+b)(1-\sigma(wx+b)) * \frac{\partial}{\partial w_j}(wx+b)$$

$\sigma(x)(1-x)$ format

$$\Rightarrow \hat{y}(1-\hat{y}) * \frac{\partial}{\partial w_j}(w_1x_1 + w_2x_2 + \cdots + w_jx_j + \cdots + w_nx_n + b)$$

constant = 0

$$\Rightarrow \hat{y}(1-\hat{y}) \cdot x_j$$

now we can calc the Error derivation:

$$\frac{\partial}{\partial w_j} E = \frac{\partial}{\partial w_j}[-y \log(\hat{y}) - (1-y)\log(1-\hat{y})]$$

$$= -y \cdot \frac{\partial}{\partial w_j} \log(\hat{y}) - (1-y)\frac{\partial}{\partial w_j} * \log(1-\hat{y})]$$

$$= -y \cdot \frac{1}{\hat{y}} \frac{\partial}{\partial w_j}(\hat{y}) - (1-y) \cdot \frac{-1}{(1-\hat{y})} \cdot \frac{\partial}{\partial w_j}(1-\hat{y})$$

$$\Rightarrow -y \cdot \frac{1}{\hat{y}}[\hat{y}(1-\hat{y}) \cdot x_j] - (1-y) \cdot \frac{-1}{(1-\hat{y})} \hat{y}(1-\hat{y}) x_j$$

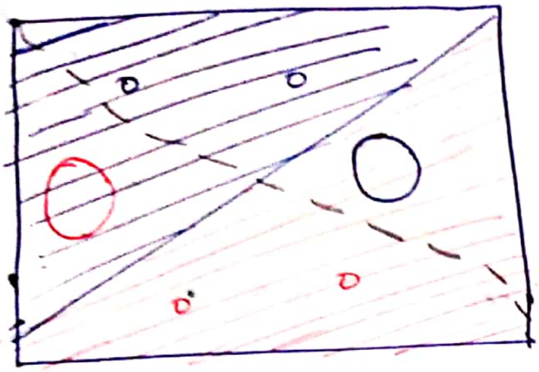
$$= -y[1-\hat{y}] \cdot x_j + (1-y)\hat{y}x_j$$

$$\Rightarrow -yx_j + y\hat{y} + \hat{y}x_j - y\hat{y}x_j$$

$$\Rightarrow -(y-\hat{y})x_j$$

$$\frac{\partial}{\partial b} E = -(y-\hat{y})$$

Gradient Descent :



1. Start with random weights.
   $w_1, \dots, w_n, b$

2. For every point $(x_i, \dots, x_n)$:

   2.1 for $i \leftarrow 1$ to n ;

   2.1.1 update $w_i' \leftarrow w_i - \alpha\left(\frac{\partial E}{\partial w_i}\right)$

   this is wat we done before

   $w_i \leftarrow w_i - \alpha(\hat{y} - y)x_i$

   2.1.2 update $b' \leftarrow b - \alpha\left(\frac{\partial E}{\partial b}\right)$

   $\leftarrow b - \alpha(\hat{y} - y)$

3. repeat until error is small.

Perceptron vs Gradient Descent;

Perceptron	Gradient Descent
If a point is misclassified. $w_i \rightarrow \begin{cases} w_i + \alpha x_i & y \text{ +ve} \\ w_i - \alpha x_i & y \text{ -ve} \end{cases}$	$w_i \leftarrow w_i + \alpha(y - \hat{y}_i)x_j$ for all points.

are those same?

labels: 1 & 0

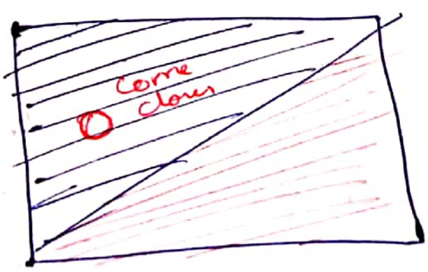
$\hat{y}$ → x

crct classified: $y - \hat{y} = 0 \Rightarrow \checkmark [y = \hat{y}]$

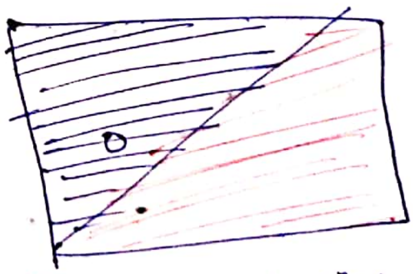
misclassified labelled as blue: $y - \hat{y} = 1$ if +ve

$y - \hat{y} = -1$ if -ve

Comparative study



(a) MISCLASSIFIED



(b) correctly classified

[well If it was a case in perceptron also it does nothing]

BUT

IN GD change weights the point is telling the line to move farther away

HYPOTHESIS :

(a) ⇒ come closer

(b) ⇒ move farther away

CONCLUSION

Atlast the line listens to all pts. & takes steps to a gud soln!

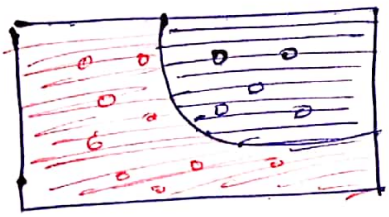
CONTINOUS PERCEPTRON:

Gradient Descent:

correct classified points	⇒ move farther away	⇒ more likely to be in my area
misclassified	⇒ come closer	⇒ So that I can also be in my area

It's all happening with straightening line.

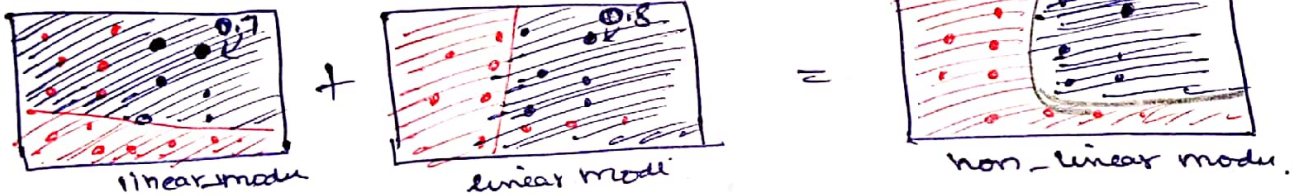
But it's be more comfort when dealing with curve ~~expecting~~ meeting the expected need.



non-linear

araise of Neural Network

Neural Network:



add the 1st blue point ⇒ 0.7 + 0.8 = 1.5

x

$0 < x < 1$

Squish → x → sigmoid fn

↓

0.82

σ (Probality of Point 1st linear mod + Prob of point in 2nd linear mode)

Weight the Sum I need to have 7 twins of model 1 than 5 twins of model 2

⇒ so, do this $7 * (0.7) + 5 * (0.8) - 6$

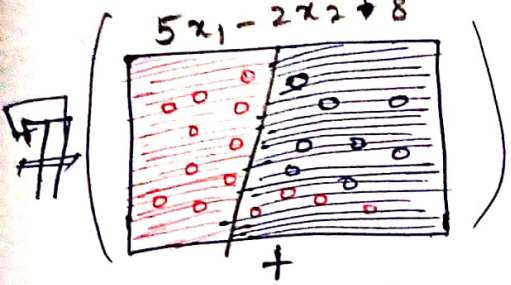
pt r

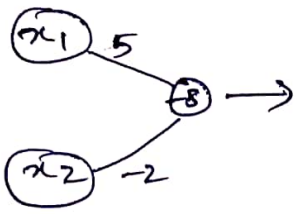
model 1

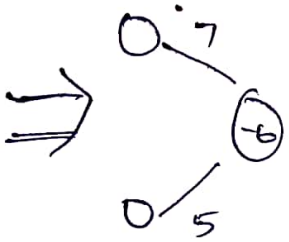
bias

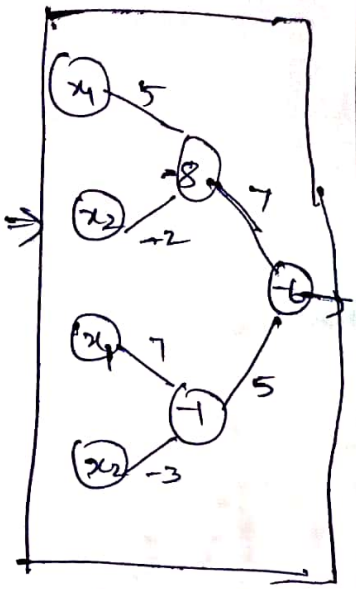
⇒ $\sigma(2.9) = 0.95$

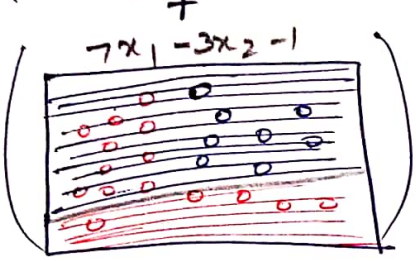
Neural Network

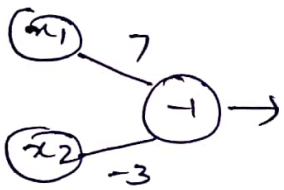




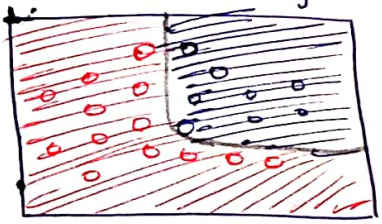


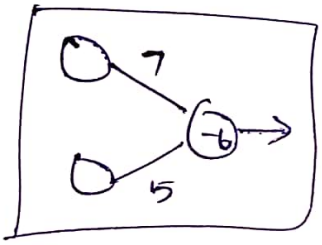




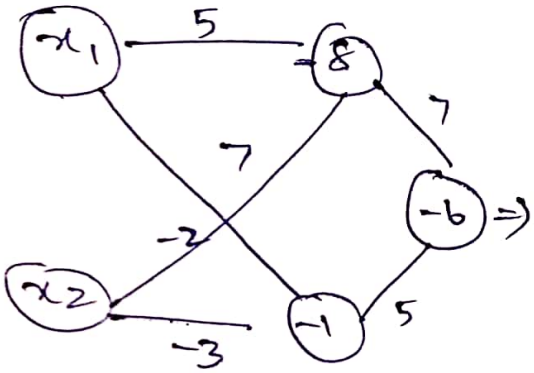


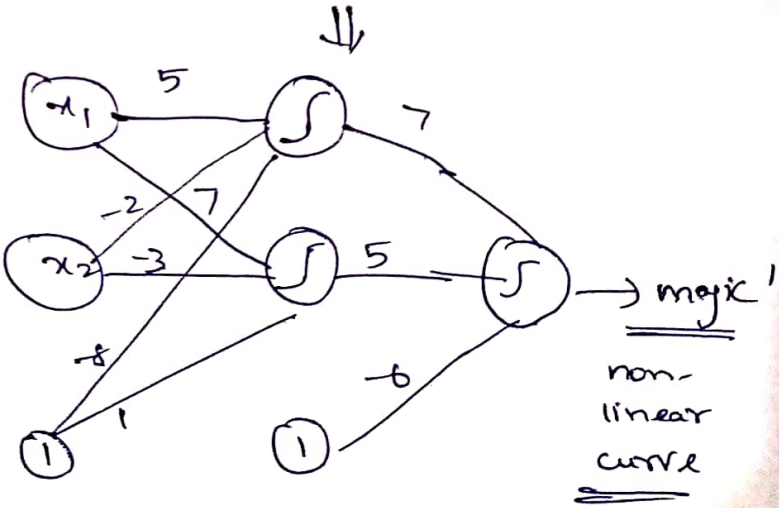
whenever $7x_1 + 5x_2 - 6$ we combin we get non-linear region





making compact





non-linear curve