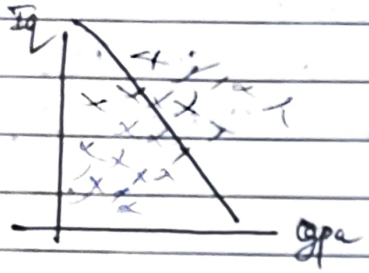
Date 23-May

Page No.
Date
YOUVA

LOGISTIC REGRESSION

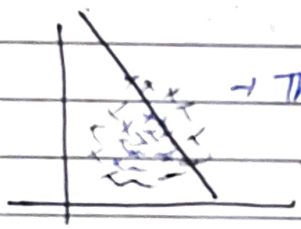
→ refer to Notepad for kypts

→ Works well on linear / sort of linear data
eg cgpa iq placement



Placement = 1 → ✗
Placement = 0 → X

→ works not well on non linear data
eg



→ This time line can't separate these 2 classes

→ multiple interpretations are there for LR

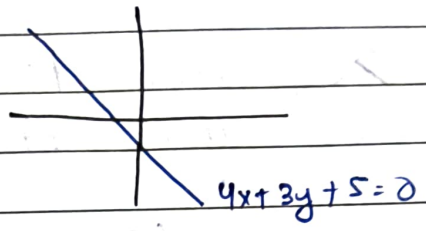
SOME BASIC GEOMETRY

- In linear regression we used $Y = mx + b$ as a eqⁿ of line
- In logistic we will use ⇒ $Ax + By + C = 0$

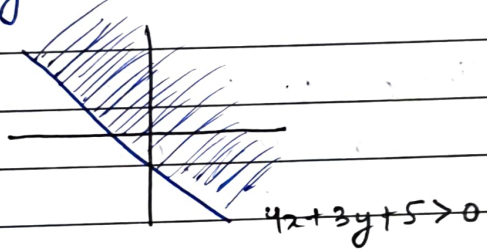
Concepts

① Every line has a +ve side and -ve side

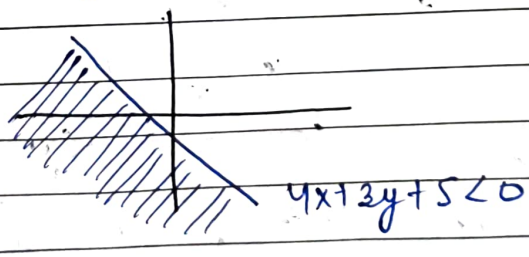
eg → We have line ⇒ $4x+3y+5=0$



- Positive side of line
- $4x+3y+5>0$



- Negative side / region of line
- $4x+3y+5<0$



② How to find out if a pt lies on line

- $4x+3y+5=0$ → line
- $(5,2)$ → pt
- Find if pt lies on that line
- $4x+3y+5=0$
- $4(5)+3(2)+5$

$20+6+5 = 31$, $31 \neq 0$. ∴ pt doesn't lie on line

(3) How to find if given pt is on positive side of line or the negative side

- if $Ax_1 + by_1 + c > 0$, then pt is in the region
- if $Ax_1 + by_1 + c < 0$, then pt is in -ve region

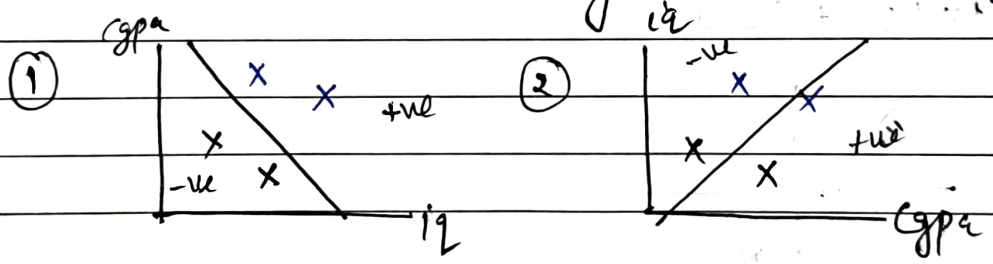
eg → $4x + 3y + 5$, pt ⇒ (2, 3)

$= 4(2) + 3(3) + 5$

$= 8 + 9 + 5$

$= 22$, since it is > 0, then pt is in the region of line

- **Ques** We have 2 models for classification which model you will select.



Ans
- model 1 performs better, as it was able to separate 2 blue pts and 2 black pts
- model 2 is doing misclassifications
- Model 1 is great model as it is correctly classified all the pts whereas model 2 is not good as it is doing certain misclassif-ications

- In classification, we have to make sure the true pts land in the region and -ve pts lands in -ve region

- How did we come to conclusion that model 1 is better?

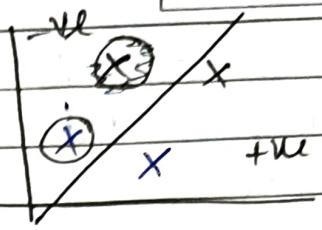
- because of the no of misclassification pts ie In model 1 there were no misclassified pts, but in model 2 there were 2 misclassified pts

- How to check through code if the pt is misclassified or not?

→ - loop on all pts.
- ~~considering~~ blue ~~green~~ pts ~~to~~ in data are positive and black pts in data to be negative
- we will loop through all pts
- and ask what is the color of the pt eg: → we have blue, and now will ask our eq^n of the i.e $Ax + By + c$ that the blue pt is true or not. i.e if $Ax + By + C > 0$ that means acc to model the green pt is in the region and in actually also it was a true pt

- It is correct classification

- Whereas :- if model is ⇒



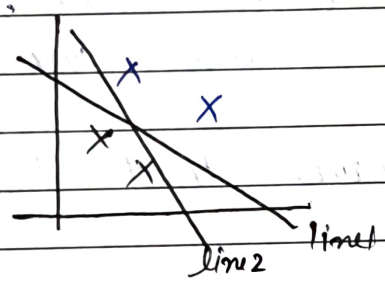
- let's say we picked up blue circle pt. Acc to data blue pt is +ve pt.
- let's ask the ~~model~~ line eqn ⇒ $Ax+By+c=0$
- say line says $Ax_1+By_1+c<0$, which means acc to line eqn it is -ve but our data is saying blue pt is +ve
- This is example of missclassification

// logic would be to count no of missclassifications pts for particular model.

```
count = 0
if ( pt → blue && Ax1+By1+c < 0){
    // missclassification
    count++
}
else if ( pt → black && Ax1+By1+c > 0)
    misslassification
    count++
}
```

- we will apply this logic on both models.
- In whichever model the count is more, it is not good model.

- This is not logistic Regression, It is very simplified version for classification
- And there is also a problem eg we have 2 models, where in both cases the model is have zero misclassified pts or you can say equal no of misclassified pts
- In this case, which model would you choose. Eg in digram

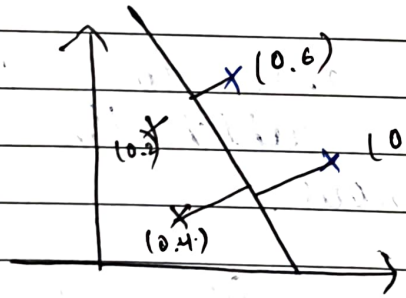


- Both lines were able to separate the 2 classes, now which line is the best we don't know.
- This is the problem with this approach
- Now, we will study, how Logistic regression solves this problem.
- Classification should not be so black and white it means it should not be so straightforward
- a better approach would instead of directly calling pts position above the line, we should assign some

some no and the pts that are more far away from line, the number should be that larger representing that this pt is more positive

- and if the pt is more close to line the number should be smaller representing that the pt is less far away from line
- same goes for -ve region, the pts that are close to line we should assign less no whereas pts that are far away from line should be assigned larger no.
- This is for just to show that distance matter

Eg



- Anology would be :- our basic approach was if our percentage is less than 50 we will not get job and if more than 50, we will get that job

But our new approach is saying, ~~that if out % is more than 50~~ placement to ho jayegi but agar apke percentage 95, 96 around hai

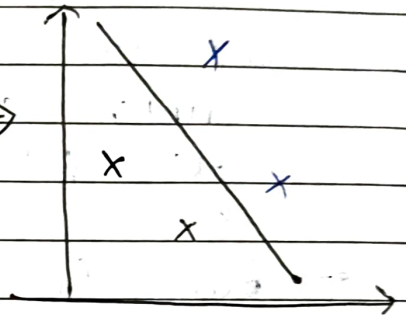
- But our new approach is saying 50 se jitna zada upar hai ex 96, 98 apki job or achi lagni chye. and same goes for jitna zada niche from 50 utni bekar job.

- How Normal approach is doing prediction?

data ⇒

iq	cgpa	Placement
60	6	0
40	4	0
80	8	1
90	9	1

Query ⇒



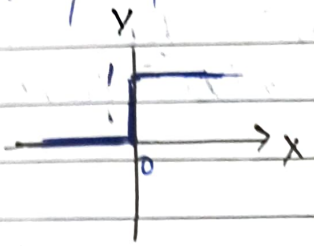
- Classification model ⇒ $Ax + By + C = 0$; Assume = $3x + 5y + 6 = 0$
- Query ⇒ {90, 9}
- $3(90) + 5(9) + 6 = 150$
- $150 > 0$, Hence output ⇒ 1 (that means blue)
- if < 0, then output would be 0 (that means black)

- In machine learning ex
- -classification model ⇒ $Ax + By + C$ se ⇒
  - $\beta_0 + \beta_1 x_1 + \beta_2 x_2$
  - $\beta_0, \beta_1, \beta_2$ are coeff, 90, 9 are ip's i.e x_1 and x_2

- Technically we are finding ⇒ $\beta_0 + \beta_1 x_1 + \beta_2 x_2$
- let's call it $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2$
- Then we are checking on z, i.e if $z > 0 \rightarrow 1$, and if $z < 0 \rightarrow 0$

$z < 0$	0
$z > 0$	1

- If you notice this is a mathematical fn → step function



- Technically, we are calculating $z = B_0 + B_1X_1 + B_2X_2$ and we are putting in ~~a~~ step fn ⇒ step(z) which gives 0, or 1 which is used for prediction

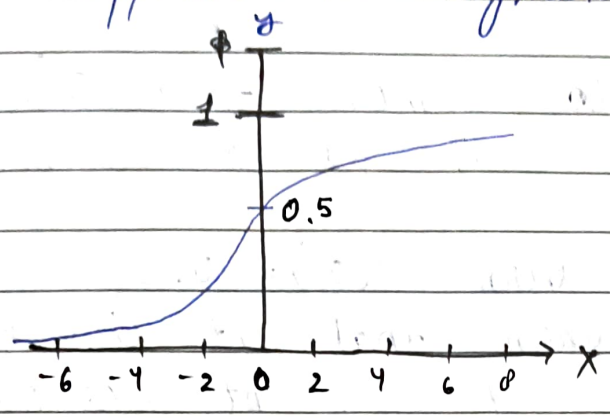
- But in new method, we don't want to use fn that outputs 0 or 1, we want a continuous output for each pt, that represents if a no assigned is more, the pt ~~this~~ is more far away and if no is less, then pt is less far away from line

- Since we don't want behaviour of 0 or 1 therefore we will stop using step fn

- We want something that if $z = B_0 + B_1X_1 + B_2X_2$ is more, then we want a large no in decimal

$z\uparrow = 0.9$ (z too much big, decimal also big)
$z\uparrow = 0.6$ (z big, decimal big)
$z\downarrow = 0.1$ (z too much small, decimal too much small)
$z\downarrow = 0.2$ (z small, decimal small)

- This is where a new function appears → Sigmoid fn

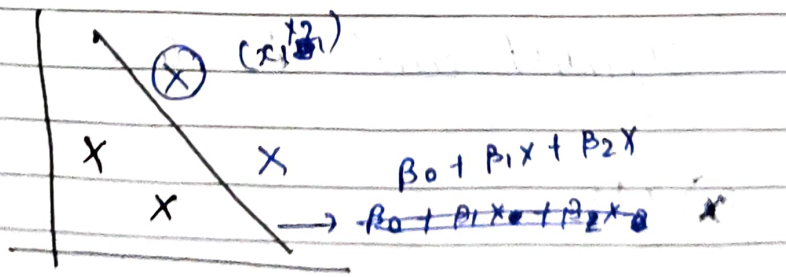


$$y = \frac{1}{1 + e^{-x}}$$

- If x value is large, then it tries to touch 1 (At infinite, y will be 1)
- If x value is too much small (-ve) then it tries to touch 0 (At infinte small value of x, y value will be 0)
- If x value is 0, then $y = 0.5$

- $x \to$ Infinty (∞), $y = 1$
- $x \to 0$, $y = 0.5$
- $x \to$ -ve Infinity $(-\infty)$, $y = 0$

Let's exchange it with step



- Let's calculate z for (x_1, x_2)

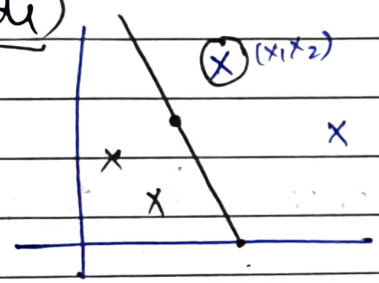
$z = \beta_0 + \beta_1 x + \beta_2 y$, $(x, y) \Rightarrow (x_1, x_2)$

$$\boxed{z = \beta_0 + \beta_1 x_1 + \beta_2 x_2}$$

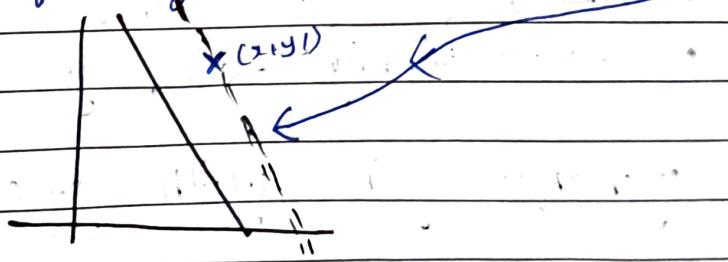
- give this z to sigmoid $\rightarrow \sigma(z)$
- from this you will get a no between 0 and 1
- Now if $z = 0$, then $\sigma(z) = 0.5$, and z is 0 when that pt lies on line, that means if any pt lies on line, then its sigmoid = 0.5

(A) → (+ve side)

I.

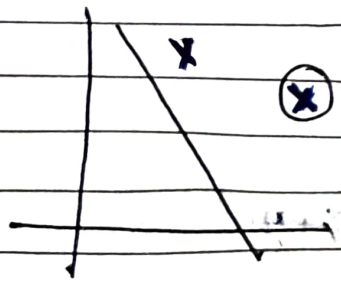


- Circled pt $\neq x_1, x_2$, for this calculate z, Obviously its $z > 0$, not too much greater than 2, ∴ sigmoid will be slight more than $\sigma = 0.5$, let's say $\Rightarrow \sigma(z) = 0.6$, that means if any pt comes on

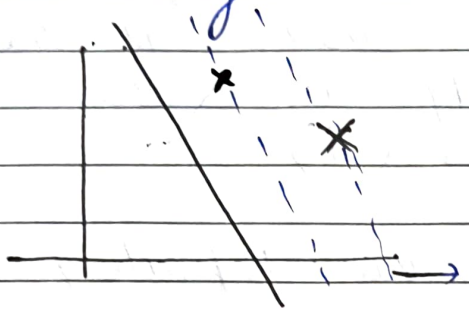


then it's sigmoid will be 0.6

II.



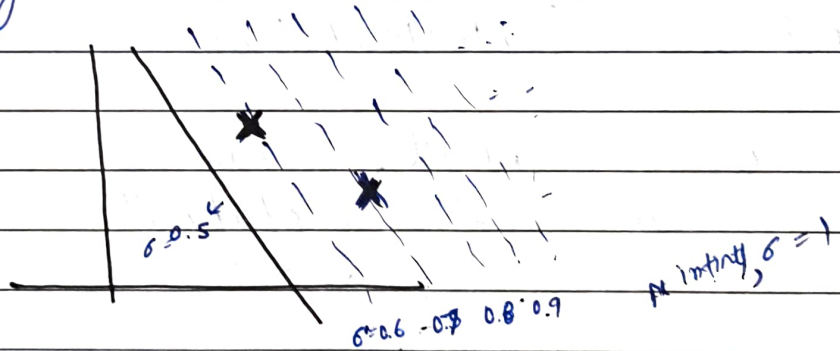
Now the circled pt 2 will more than the other therefore its sigmoid will be slight higher than before
Let's say $\sigma(z) = 0.7$



→ that means any pt on this line sigmoid will be 0.7

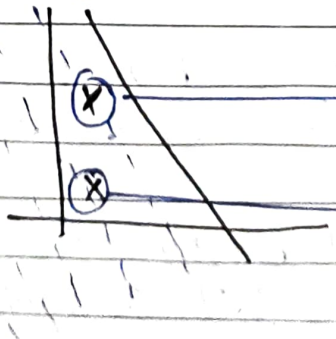
III. Similarly, thoda aur age jayenge sigmoid will be more and if infinite move then sigmoid will be 1

ie



(B) (-ve side)

- Similarly will happen to -ve side



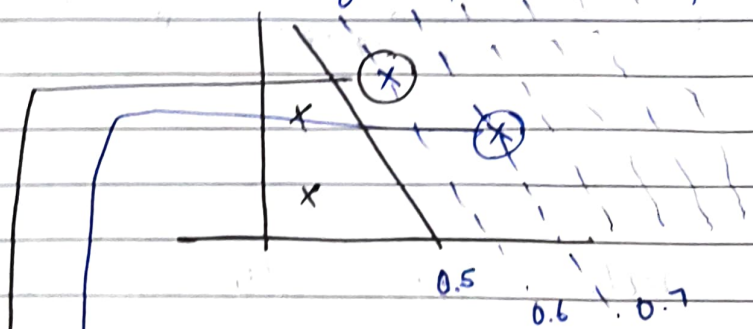
→ z < 0, σ(z) = 0.4 (not too much)

→ z < 0, smaller, σ(z) = 0.3

- Similarly, when z is - infinity σ(z) = 0

- That means, our classification region is converted into gradient
- If we move infintely in +ve, then sigmoid = 1, that means: if there is pt at infinity we are 100% sure it is positive (:1)
- Similarly, if we move infinitely in -ve and there is pt at -infinity then we are 100% sure that it is negative (0)
- In a way, notice there is nice probabilistic interpretation
- Now for a point, we can say in terms of probability

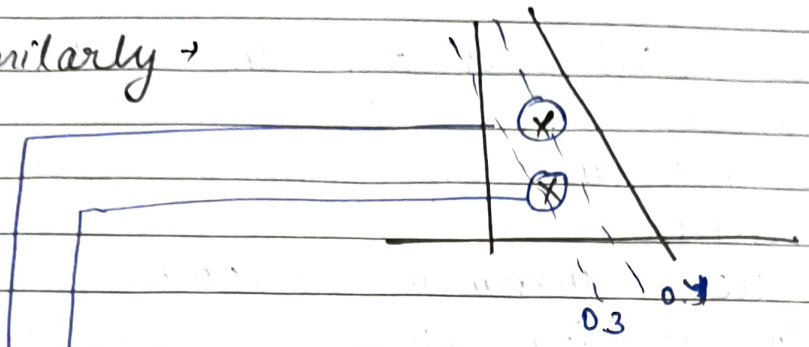
eg



for this pt, the probability of this pt to be +ve is 0.6 & -ve is 0.4

for this pt, probability of this pt to be +ve is 0.7 & being -ve is 0.3

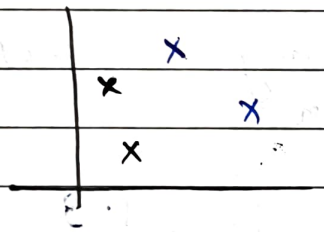
Similarly →



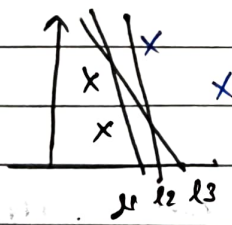
- Prob of this pt being +ve is 0.4, Hence p of this pt -ve is 0.6
- Prob of this pt +ve is 0.3, ∴ p of this pt -ve is 0.7

Maximum Likelihood

- Data →



- Ultimately we want a line that does best classification
- There could be multiple lines



- These all lines are doing classification properly
- But we want a best line.
- To find best line in Machine Learning, we use loss function
- We minimize the loss function using for given parameters and for which parameters the loss fn is minimum is the best line

- Therefore we will first try to find out best loss function.
- Loss fn is just the logic which tells which line is better
- We will use maximum Likelihood, which is the loss fn for Logistic regression

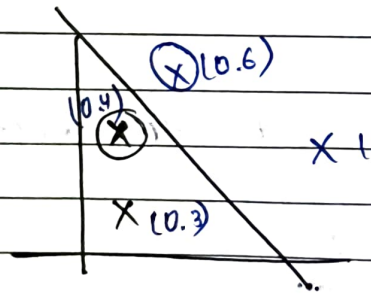
INTUITION OF: WHICH LINE IS BETTER

- We will use a : likelihood function : first we will assign (for now without calculation) each pt the probability of pt of 1 (or blue) w.r.t to our line.

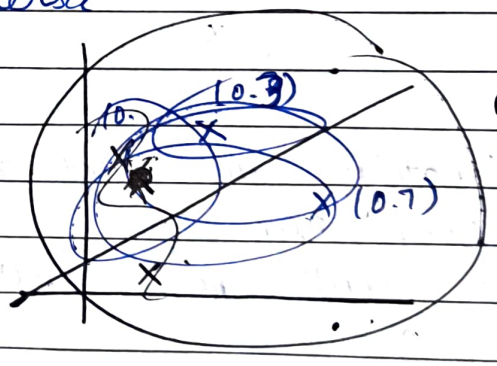
of pt being 1

- pts that are far away, ∴ prob will be more and vice versa

Line 1



line 1 / model 1



← ignore

- We will calculate likelihood : which basically means multiplying probabilities of each model for actual class.
- ~~ie means P of pt1 x i~~

ie means eg the circled pt is actualy blue in data, & we also calculate prob of that pt blue → 0.6

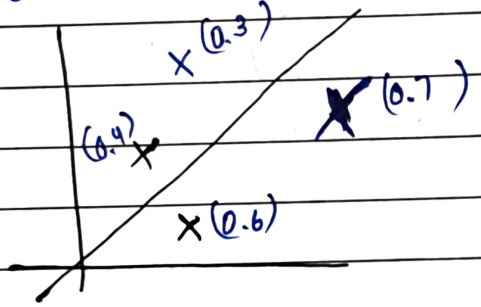
- eg 2: The second circle pt is actually black in data, but we finds its prob of blue is 0.4, but likelihood says p of actual class it's actual class is black ∴ P ⇒ 1 − 0.4 = 0.6

∴ Likelihood = 0.6 X 0.8 X (1−0.4) X (1−0.3)
= 0.6 X 0.8 X 0.6 X 0.7

Likelihood1 = 0.20

Similarly we have line 2

Line 2:



= 0.3 X 0.7 X (1−0.4) X (1−0.6)
= 0.3 X 0.7 X 0.6 X 0.4

Likelihood2 = 0.05

- Whosever likelihood is more, is better model
- It's actually logical, as if this number is big that means, har rang ka usi rang hone ka prob is high. i.e if prediction (prob) alligns with color, that means our prediction is right
- and if this no is less, that means aap blue ko black and black ko blue bula rahe ho.

- Maximum likelihood is a good parameter for judging because hum har pt se wo probability mang rahe hai jo wo hai hai
- Hena we have to find that like whose likelihood is maximum.
- Problem with using this approach
  - In real data, there are lakhs of pts which means lakhs of probability being multiplied which introduces the problem of underflow.
- Solution
  - We will use log of probabilities
  - i.e. ⇒ we had ⇒ $0.8 \times 0.6 \times 0.6 \times 0.7$
  - Applying log ⇒ $\log(0.8 \times 0.6 \times 0.6 \times 0.7)$
  - $= \log(0.8) + \log(0.6) + \log(0.6) + \log(0.7)$
  - We solved the underflow problem
  - But there is also a problem i.e. log of (0 to 1) is -ve no.
  - To prevent from this, we will take -ve of logs.

$= -\log 0.8 - \log 0.6 - \log 0.6 - \log 0.7$

by which we solved the -ve problem

- Also the product in likelihood we wanted was maximum

- but for logs, we want whose log is minimum. Ques is why?
- because log of small no is more -ve and we reverse it using -ve of log.

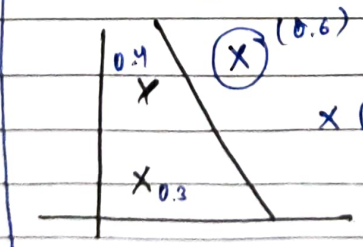
Eg
- $\log(0.1) = -1$ | $-\log(0.1) = +1$
- $\log(0.5) = -0.3$ | $-\log(0.5) = +0.3$

- till now, we have:
  $-\log 0.8 - \log 0.8 - \log 0.6 - \log 0.7$
- since there were 4 pts -
  $\Rightarrow -\log(\hat{y}_1) - \log(\hat{y}_2) - \log(\hat{y}_3) - \log(\hat{y}_4)$
  $\Rightarrow$ where $\hat{y}_1 = \sigma(z)$, & $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

But, we directly can't write it as this, as you can see $\hat{y} = \sigma(z)$, which means it is the probability of pt being blue(1) (it only considered P of blue, which means it is directly putting P of blue class without knowing actual class)

- Therefore it can be transformed as:-
  $-y_1 \log \hat{y}_1 - (1 - y_1) \log(1 - \hat{y}_1)$

Eg.



- actual pt = blue (1) = y_1
- $1 \times (\log(0.6)) - (1 - 1) \log(1 - \hat{y}_1)$
  $\log 0.6 - 0$
  $\log(0.6)$

Basically this eqn : $-y_1 \log \hat{y}_1 - (1-y_1) \log (1-\hat{y}_1)$

it is the logic that is finding prob of actual class, in this we know this pt is blue, which means p of blue should be assigned

- Similarly let's say actual data pt is black (0), i.e $y_2 = 0$
- It prevents (eq) from assigning directly the p of blue, but assigns the actual probability of class

$$= -y_2 \log \hat{y}_2 - (1-y_2) \log (1-\hat{y}_2)$$

$$= 0 \times (\log \hat{y}_2) - (1-0) \log (1-0.4)$$

$$= 0 - \log (0.6)$$

$$= -\log 0.6$$

That's how it prevents from assigning all prob of blue without knowing actual class

∴ ~~We had new eqn~~

$$= -y \log$$

∴ we transformed it into :-

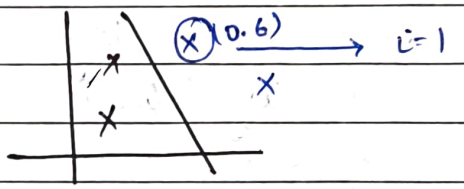
- The output we wanted was :-

$$-\log 0.8 - \log 0.6 - \log 0.6 - \log 0.7$$

which we can achieve it by using :-

$$-Y_i \log \hat{Y}_i - (1 - Y_i) \log (1 - \hat{Y}_i)$$

for $i = 1 \Rightarrow$



Y_i = Actual color = blue = 1

$\hat{Y}_1 = \sigma(z) = 0.6$

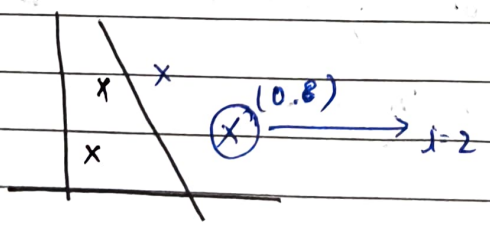
put these in eqn

$= -1 \times \log 0.6 - (1-1) \log (1 - 0.6)$

$= -1 \times \log 0.6 = -\log 0.6$

$= -\log 0.6$

for $i = 2$

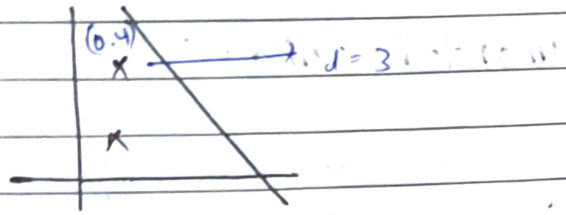


- $Y_i = 1$ (Actual ⇒ blue)
- $\hat{Y}_2 = \sigma(z_2) = 0.8$
- putting in eqn

$= -1 \log 0.8 - (1-1) \log (1 - 0.8)$

$= -\log 0.8$

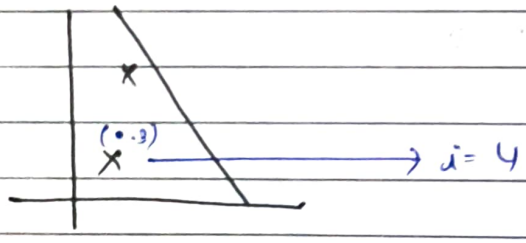
for (i=3).



$Y_3 = 0$ (Actual color = black)
$\hat{Y}_3 = \sigma(Z_3) = 0.4$ (Prob of blue)
put in eqn

$= -0 \times \log 0.4 - (1-0) \log (1-0.4)$
$= 0 - 1 \times \log (0.6)$
$= -\log 0.6$

for i=4



$Y_4 = 0$ (Actual color → black)
$\hat{Y}_4 = \sigma(Z_4) = 0.3$ (Prob of this pt being blue)

$= -0 \times \log 0.3 - (1-0) \log (1-0.3)$
$= -\log 0.7$

- That's how this eqn $\boxed{-y \log \hat{y} - (1-y_i) \log (1-\hat{y})}$ works for all points
- Since we have n pts

$$= -\sum_{i=1}^{n} y_i \log \hat{y}_i - (1-y_i) \log (1-\hat{y}_i)$$

If you want to find avg → divide by n

$$L \rightarrow -\frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)$$

- This is the loss function of Logistic regression also called as <u>log loss</u> / <u>Binary cross entropy</u>.
- for this, we have to find $\beta_0, \beta_1, \beta_2$ (coef for line) for which our loss function is minimize.
- Q → where are $\beta_0, \beta_1, \beta_2$ = ?

  Ans • $\hat{y}_i = \sigma(z_i)$

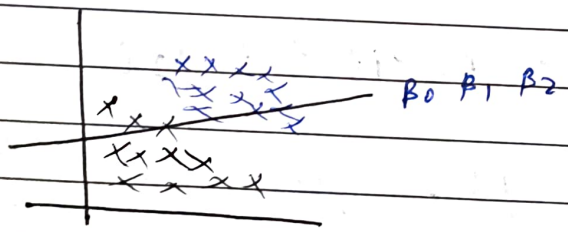
  • $z_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i}$
- We are at pt, when we have data, there can be multiple lines and each line have its set of coef ($\beta_0, \beta_1, \beta_2$)
- We want such coef (line) for which our loss function is minimum

Now we will apply Gradient Descent

GRADIENT DESCENT

$$L = \frac{-1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i) + -(1-y_i)\log(1-\hat{y}_i)$$

- Problem with this loss function
- There is no closed form solution
- We can't directly use some formula
- We can only approaximate, which can be done using Gradient Descent
- How? Let's see
- Data →



- We start with random line, it will have β_0 β_1 β_2
- We will update these values

$$\beta_0 = \beta_0 - \eta \frac{\partial L}{\partial \beta_0}$$ → Differentiation of LF w.r.t β_0

$$\beta_1 = \beta_1 - \eta \frac{\partial L}{\partial \beta_1}$$

$$\beta_2 = \beta_2 - \eta \frac{\partial L}{\partial \beta_2}$$

- finding $\frac{\partial L}{\partial \beta_0}, \frac{\partial L}{\partial \beta_1}, \frac{\partial L}{\partial \beta_2}$

- $$L.f = \frac{-1}{n} \sum_{i=1}^{n} y_i \log \hat{y}_i - (1-y_i) \log (1-\hat{y}_i)$$

- Let's assume there is only 1 pt

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

- We have to find $\frac{\partial L}{\partial \beta_1}$

$$\Rightarrow \frac{\partial L}{\partial \beta_1} = \frac{\partial}{\partial \beta_1}\left[-y \log \hat{y} - (1-y) \log (1-\hat{y})\right]$$

- that pt is →

x_1	x_2	y	$\hat{y}_i$
28	13	0	0.63

$$= \underbrace{\frac{-y}{\hat{y}} \frac{\partial \hat{y}}{\partial \beta_1}}_{\text{I}} \underbrace{-(-)\frac{\hat{y}(1-y)}{(1-\hat{y})} \frac{\partial (1-\hat{y})}{\partial \beta_1}}_{\text{II}} \left(\because \frac{d \log x}{dx} = \frac{1}{x}\right.$$

$$= \frac{-y}{\hat{y}} \left(\sigma(z)\cdot[1-\sigma(z)]\right) \frac{\partial z}{\partial \beta_1}$$

$$= \frac{-y}{\hat{y}} \hat{y}(1-\hat{y}) x_1$$

(Solving I term)

1) $\hat{y} = \sigma(z)$

$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

$\frac{\partial z}{\partial \beta_1} = x_1$

2) $\frac{\partial}{dx} \sigma(x) = \sigma(x)(1-\sigma(x))$

Solving 2nd term

$$= -\frac{(1-y_i)}{(1-\hat{y})} \frac{\partial (y - \hat{y})}{\partial \beta_1}$$

$$= -\frac{(1-y)}{(1-\hat{y})} \frac{\partial (1-\sigma(z))}{\partial \beta_1}$$

$$= -(1-$$

$$= \frac{(1-y)}{(1-\hat{y})} \hat{y}(1-\hat{y}) x_1$$

$$= \underline{(1-y)\hat{y} x_1}$$

Combining both underline of

$$\frac{\partial L}{\partial \beta_1} = -y(1-\hat{y}) x_1 + (1-y)\hat{y} x_1$$

$$\frac{\partial L}{\partial \beta_1} = (\hat{y} - y) x_1$$

for n pts

$$\boxed{\frac{\partial L}{\partial \beta_1} = (\hat{y}_i - y_i) x_{1,i}} \quad \text{---} (1)$$

Similarly

$$\boxed{\frac{\partial L}{\partial \beta_2} = (\hat{y}_i - y_i) x_2} \quad \text{---} (2)$$

$$\boxed{\frac{\partial L}{\partial \beta_0} = \hat{y} - y} \quad \text{---} (3)$$

Apply in code

Summary

Step 1 - Sigmoid

Step 2 - maximum likelihood

Step 3 - log → Binary Cross Entropy

Step 4: Gradient Descent → β_0 β_1 β_2 which minimizes L.