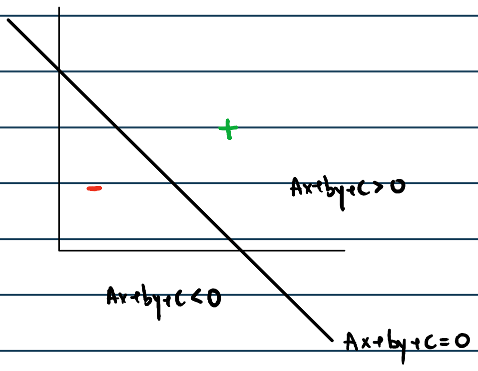
# Logistic Regression

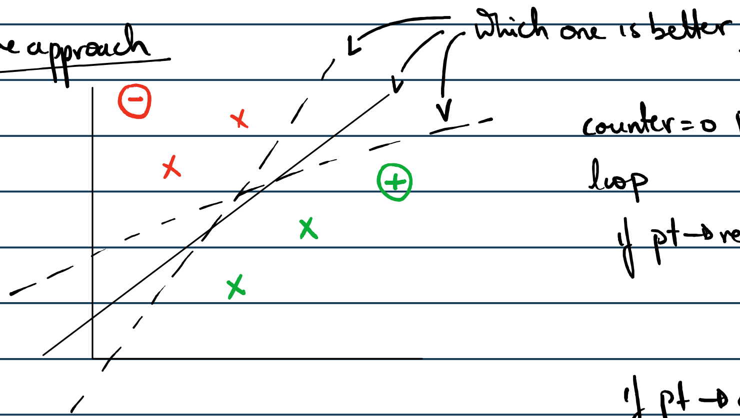
Monday 29 April 2024 2:15 PM

- → **Parametric ML Algo**
- → Although the name contains regression, it is a classification algorithm.
- → linear model




→ given two classes, draw a line such that it separates the classes.

naive approach




counter = 0

loop

  if pt → red and Ax+by+c>0

    misclassification → counter+1

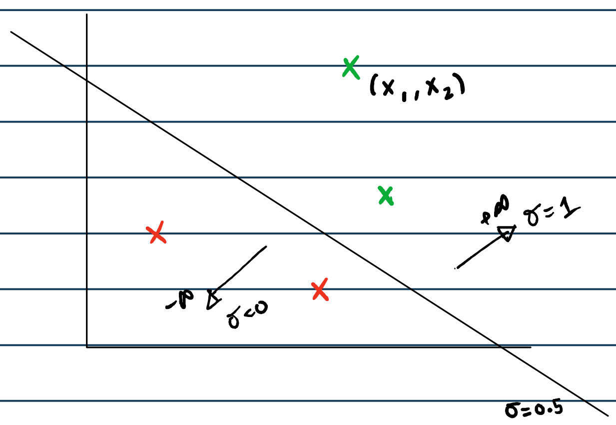
  if pt → green and Ax+by+c<0

    misclassification → counter+1

} → number of misclassifications

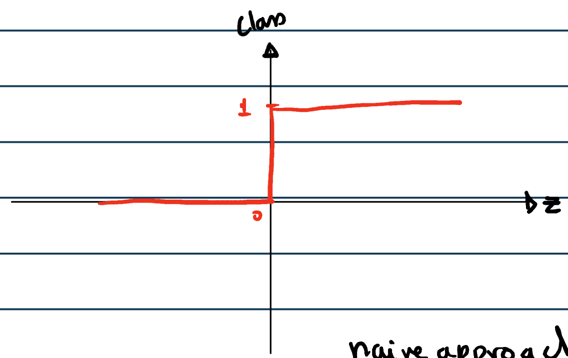
→ Instead of a black & white approach, is there a better way?

  → Based on the distance from the line, lets assign a number 0-1, and classify based on this number.



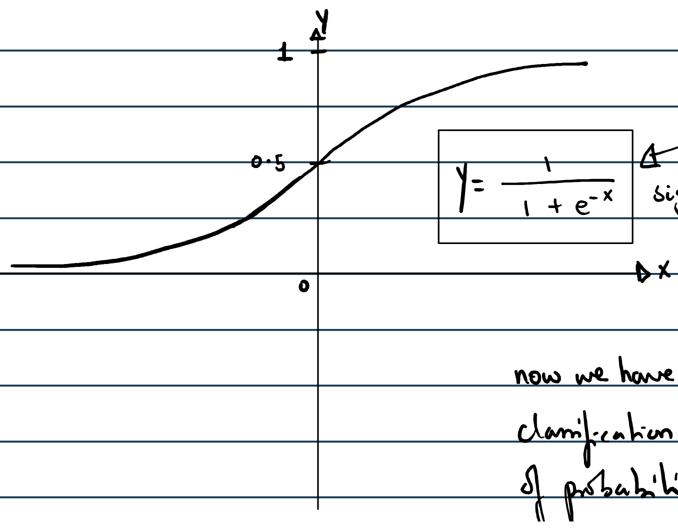

x - 1

x - 0




naive approach - Step function

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




$\boxed{y = \frac{1}{1+e^{-x}}}$ sigmoid → Probability of the point being +ve

$$\boxed{\frac{d\sigma}{dx} = \sigma(x)\left[1-\sigma(x)\right]}$$

now we have transformed our classification space into a 'gradient' of probabilities.

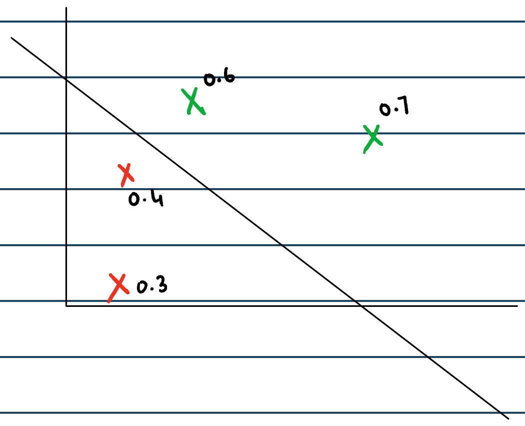
classification space into a gradient of probabilities.

→ How to figure out the best classification line? ⟶ min(loss function)

↳ Maximum likelihood

→ The likelihood function is the product of the predicted probabilities for the actual class of each observation

→ The model which has the greater likelihood is the better model.



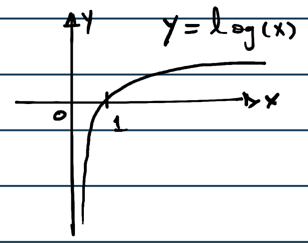

$$\text{likelihood} = 0.6 \times 0.7 \times (1-0.4) \times (1-0.3)$$
$$= 0.6 \times 0.7 \times 0.6 \times 0.7$$
$$= 0.1764$$

→ Find a line such that the likelihood is maximum.

→ This will be the best possible line to classify two classes.

→ When we are multiplying many probabilities we will encounter underflow.

$$\left[-\log(\text{maximum likelihood})\right]_{\text{minimize}}$$




$$\hat{y}_i = \sigma(\underbrace{z}_{\beta_0+\beta_1 x_1+\beta_2 x_2})$$

predicted +ve probability

$$\text{log loss error} = \frac{1}{n}\sum_{i=1}^{n} -\underbrace{y_i}_{\text{actual class } 1/0} \log(\hat{y}_i) - (1-y)\log(1-\hat{y}_i)$$

Minimize

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

where $\hat{y}_i = \sigma(z_i)$

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

→ Minimizing the log loss

$$\begin{array}{c|cccccc}
 & 1 & 2 & 3 & \dots & m & y \\
1 & X_{11} & X_{12} & X_{13} & \dots & X_{1m} & y_1 \\
2 & X_{21} & X_{22} & X_{23} & \dots & X_{2m} & y_2 \\
3 & X_{31} & X_{32} & X_{33} & \dots & X_{3m} & y_3 \\
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\
n & X_{n1} & X_{n2} & X_{n3} & \dots & X_{nm} & y_n
\end{array}$$

$$\hat{y} = \begin{bmatrix} \sigma(\beta_0 + \beta_1 X_{11} + \beta_2 X_{12} + \beta_3 X_{13} \dots + \beta_m X_{1m}) \\ \sigma(\beta_0 + \beta_1 X_{21} + \beta_2 X_{22} + \beta_3 X_{23} \dots + \beta_m X_{2m}) \\ \sigma(\beta_0 + \beta_1 X_{31} + \beta_2 X_{32} + \beta_3 X_{33} \dots + \beta_m X_{3m}) \\ \vdots \\ \sigma(\beta_0 + \beta_1 X_{n1} + \beta_2 X_{n2} + \beta_3 X_{n3} \dots + \beta_m X_{nm}) \end{bmatrix} = \sigma\left( \begin{bmatrix} 1 & X_{11} & X_{12} & X_{13} & \dots & X_{1m} \\ 1 & X_{21} & X_{22} & X_{23} & \dots & X_{2m} \\ 1 & X_{31} & X_{32} & X_{33} & \dots & X_{3m} \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & X_{n1} & X_{n2} & X_{n3} & \dots & X_{nm} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} \right)$$

$$\boxed{\hat{y} = \sigma(X\beta)}$$

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

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

$$L = -\frac{1}{n}\left[ Y\log(\hat{Y}) + (1-Y)\log(1-\hat{Y}) \right]$$ → matrix form

where $\hat{Y} = \sigma(X\beta)$

find $\beta$ matrix for which $L$ is minimum.

→ has no closed form solution

→ Approximation technique - Gradient descent

→ initialize $\beta$ with random values

for i in epochs:

$$\beta = \beta - \eta \frac{\Delta L}{\Delta \beta}$$

$\beta_{(m+1,1)}$

$Y_{n \times 1}$

$\hat{Y}_{n \times 1}$

$X_{(n,m+1)}$

$$\boxed{\frac{\Delta L}{\Delta \beta} = -\frac{1}{n}(Y-\hat{Y})X}$$

[practice deriving the gradient]

[practice deriving the gradient]

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(penalty='none', solver='sag')
```