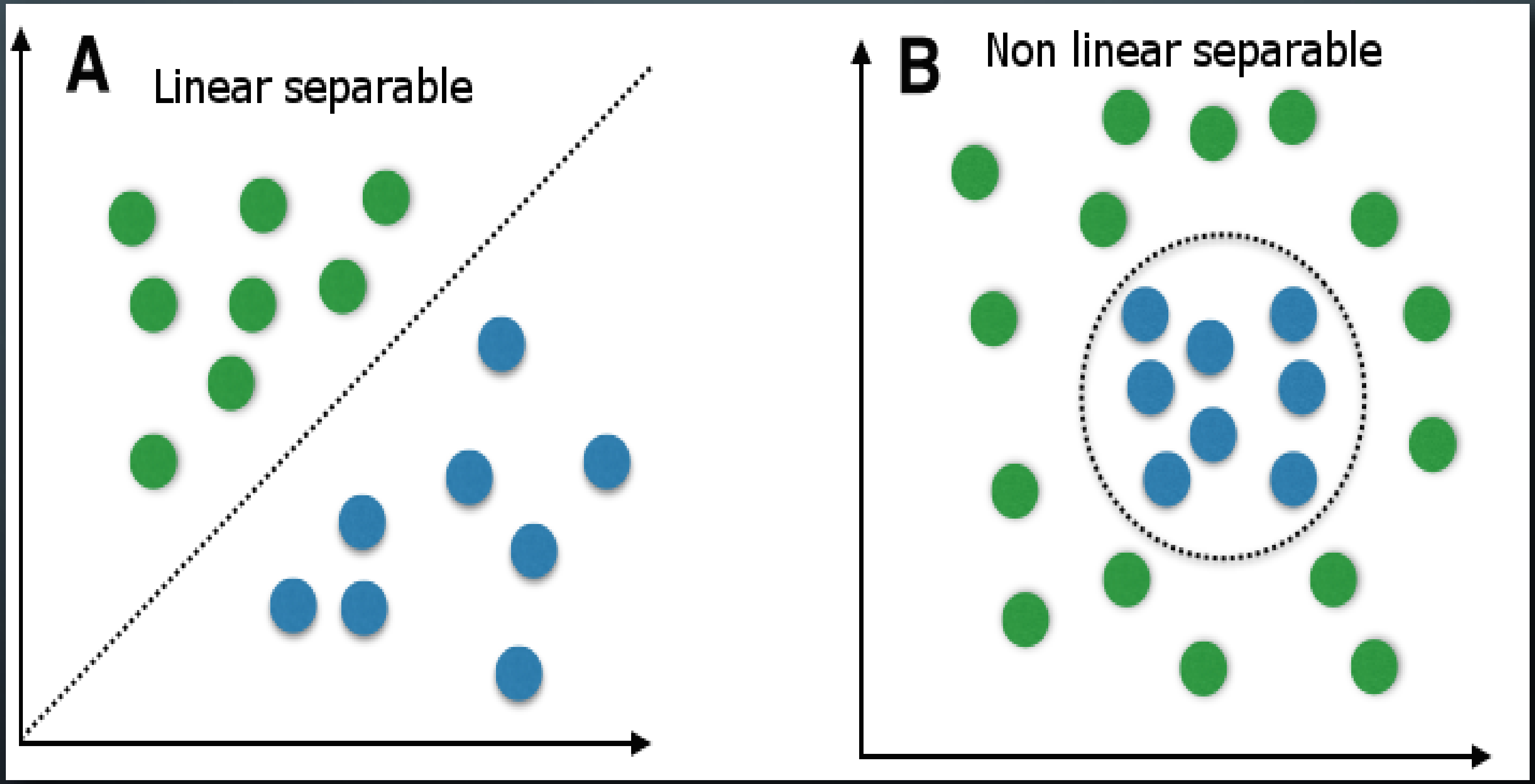
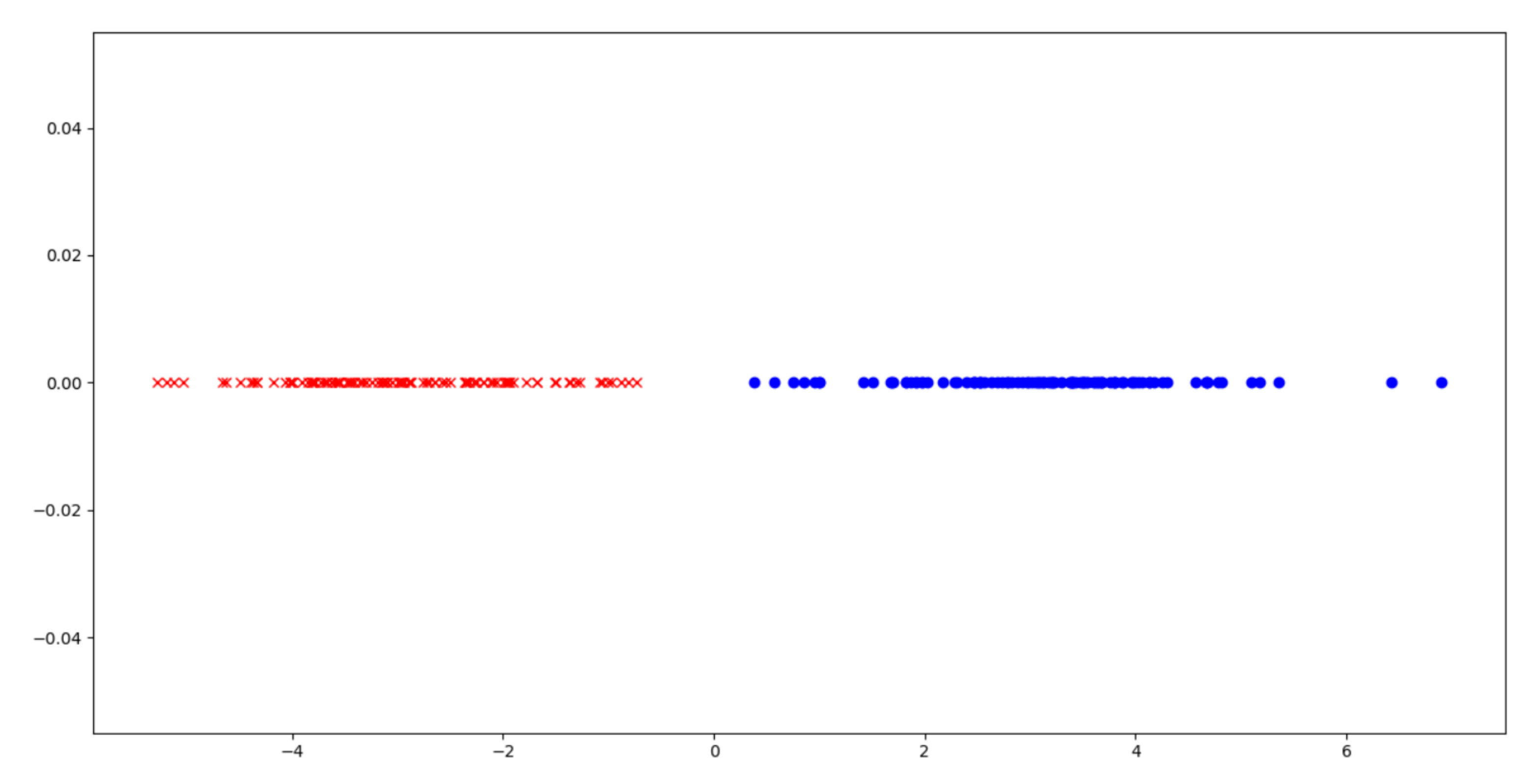
LOGISTIC REGRESSION

MOHAMMAD GHODDOSI

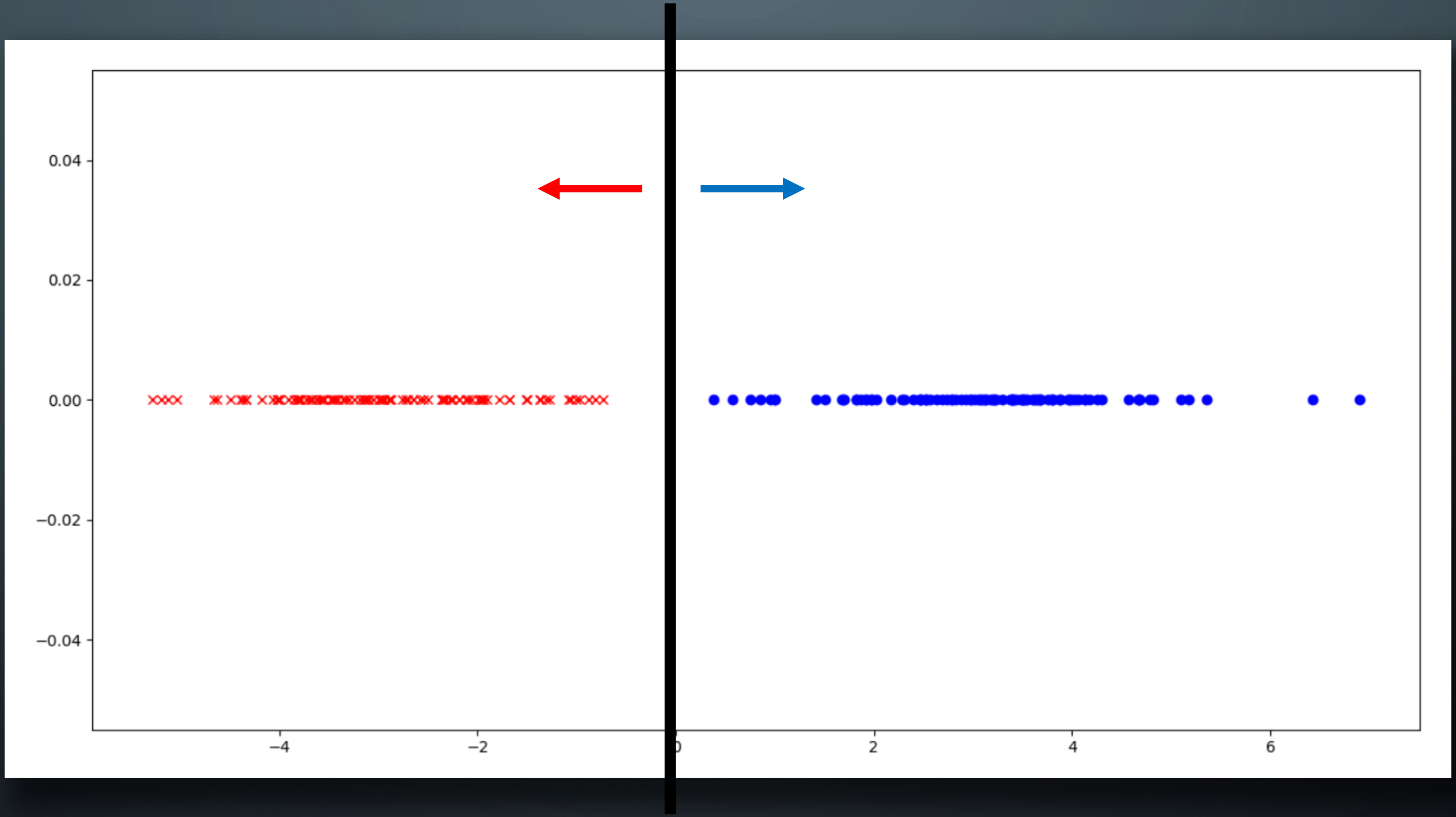
CLASSIFICATION



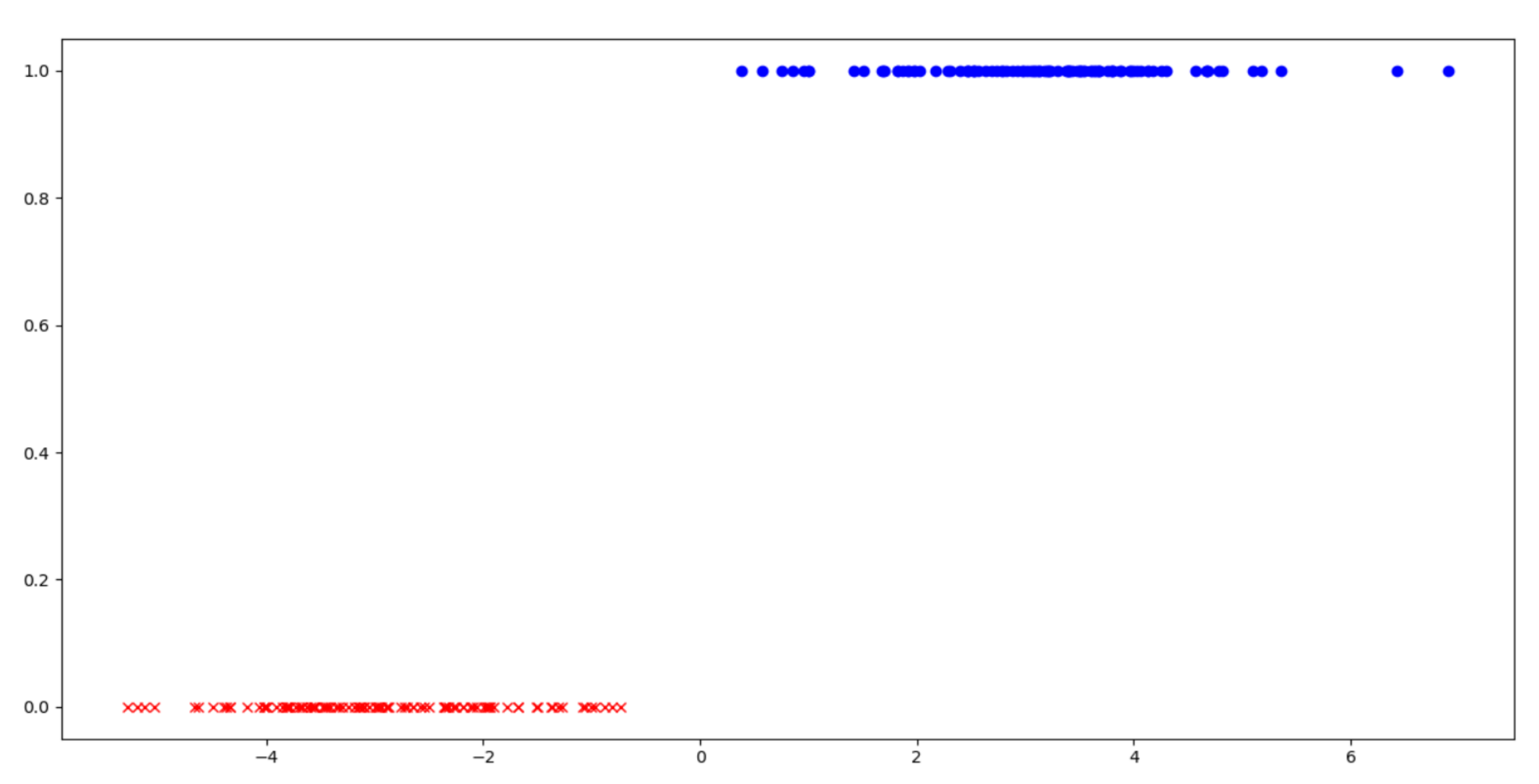
1D – CLASSIFICATION



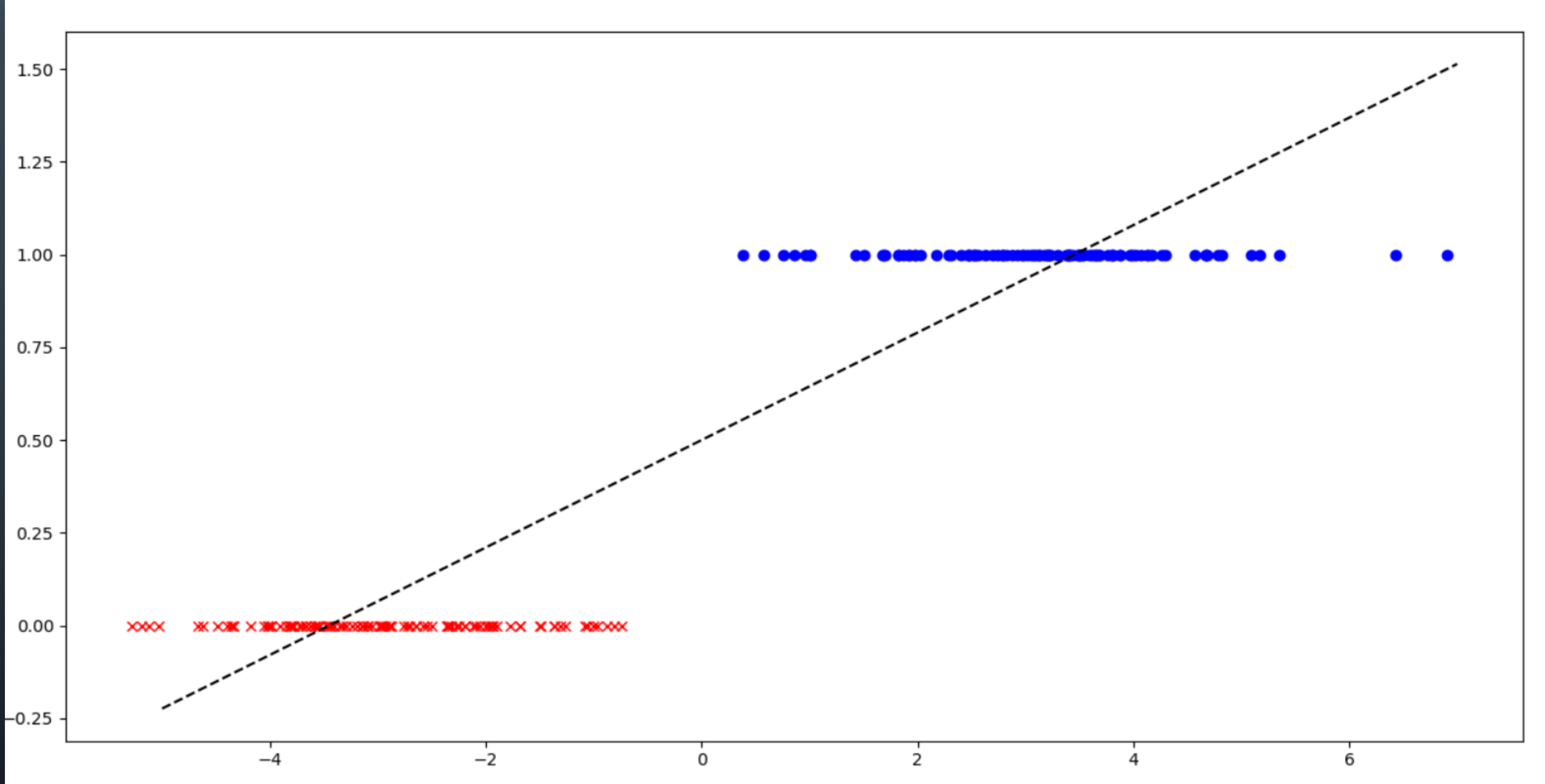
1D – CLASSIFICATION



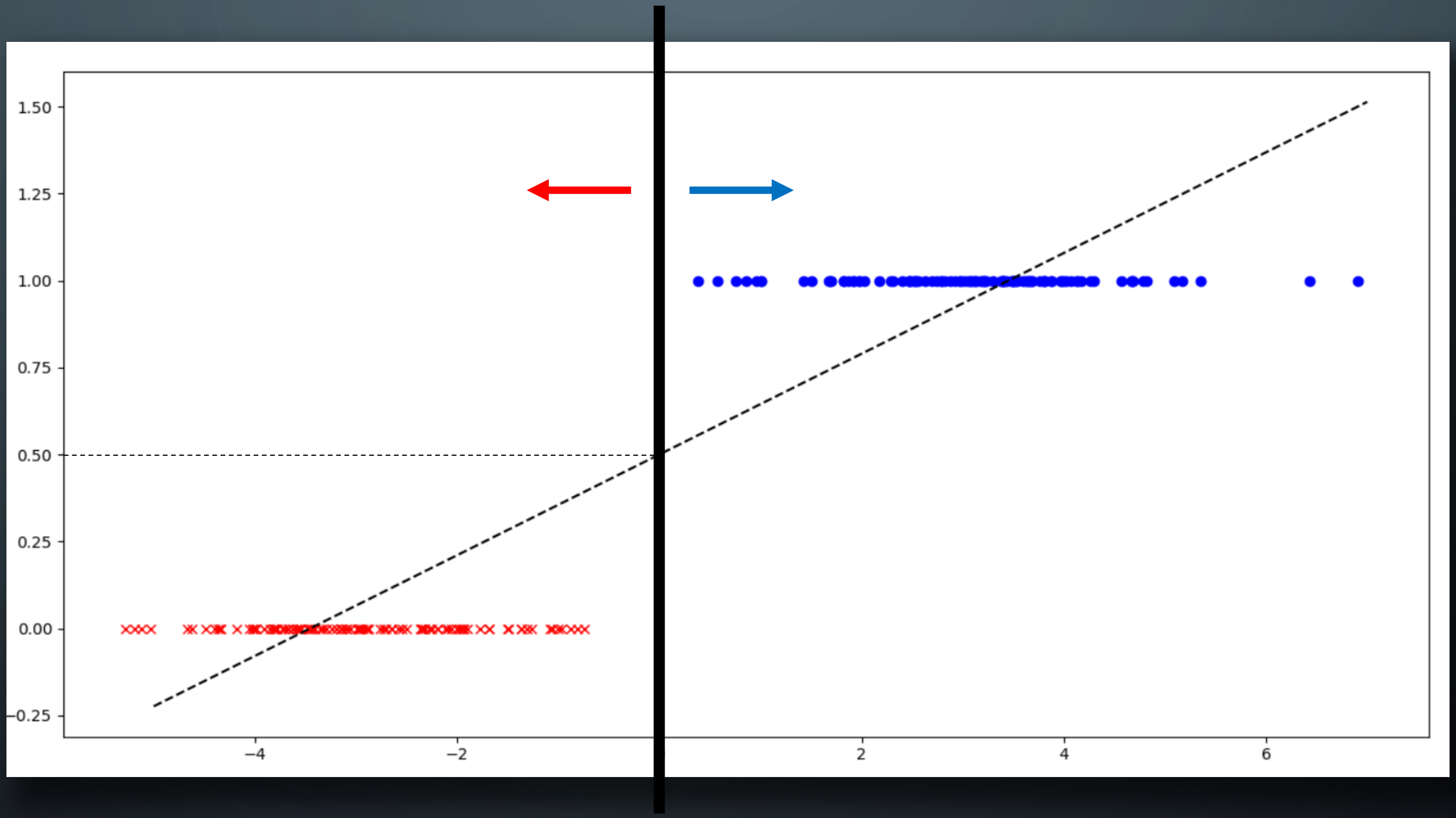
1D – CLASSIFICATION AS 2D REGRESSION



1D – CLASSIFICATION AS 2D REGRESSION



1D – CLASSIFICATION AS 2D REGRESSION

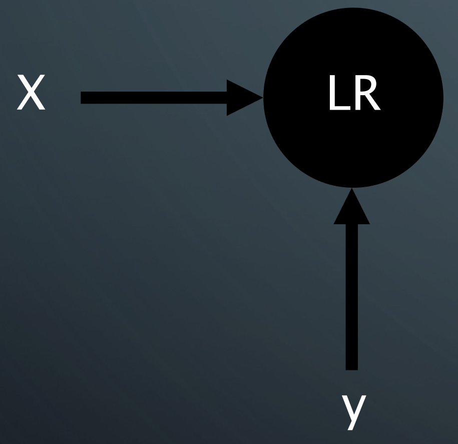


PROBLEMS

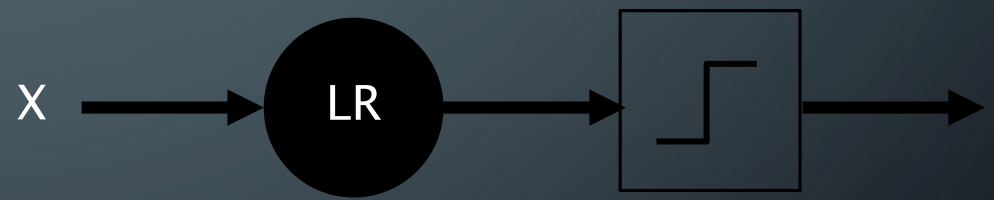
- It doesn't wok fine every where
  - One noisy point far away from data
- Error is not categorical
- What if we need probability

CLASSIFICATION AS REGRESSION

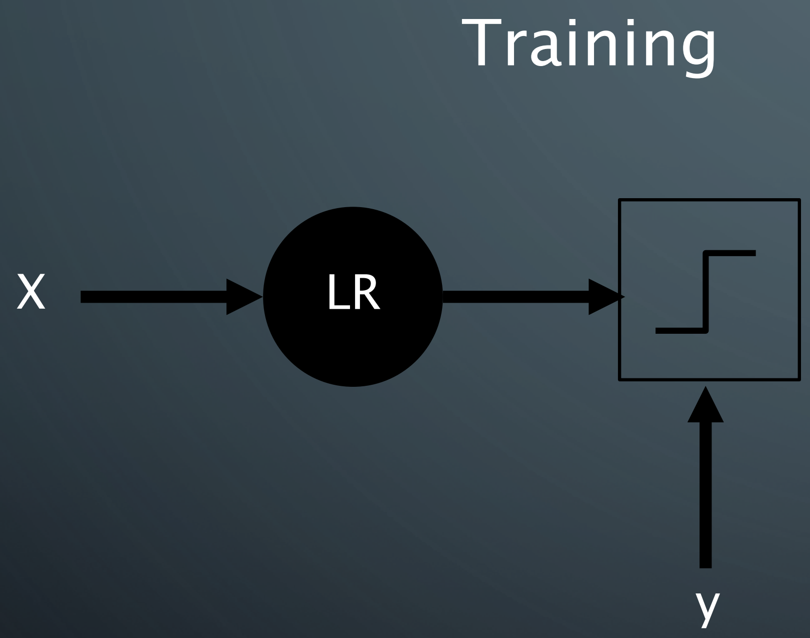
Training

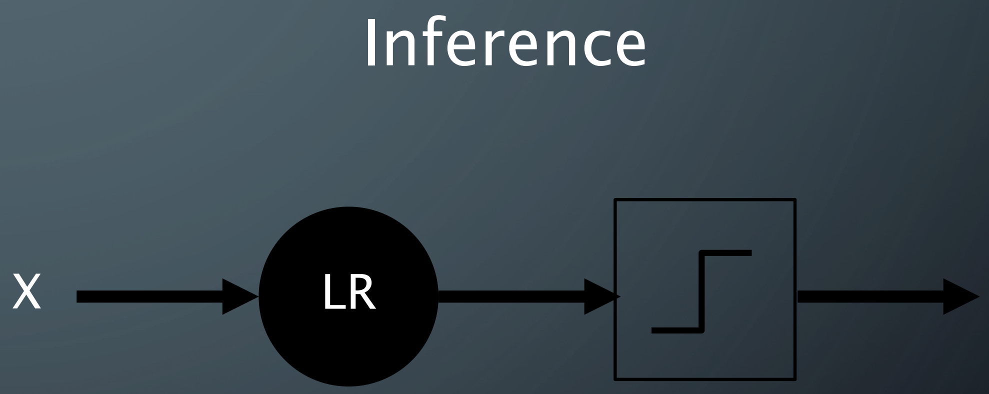


Inference

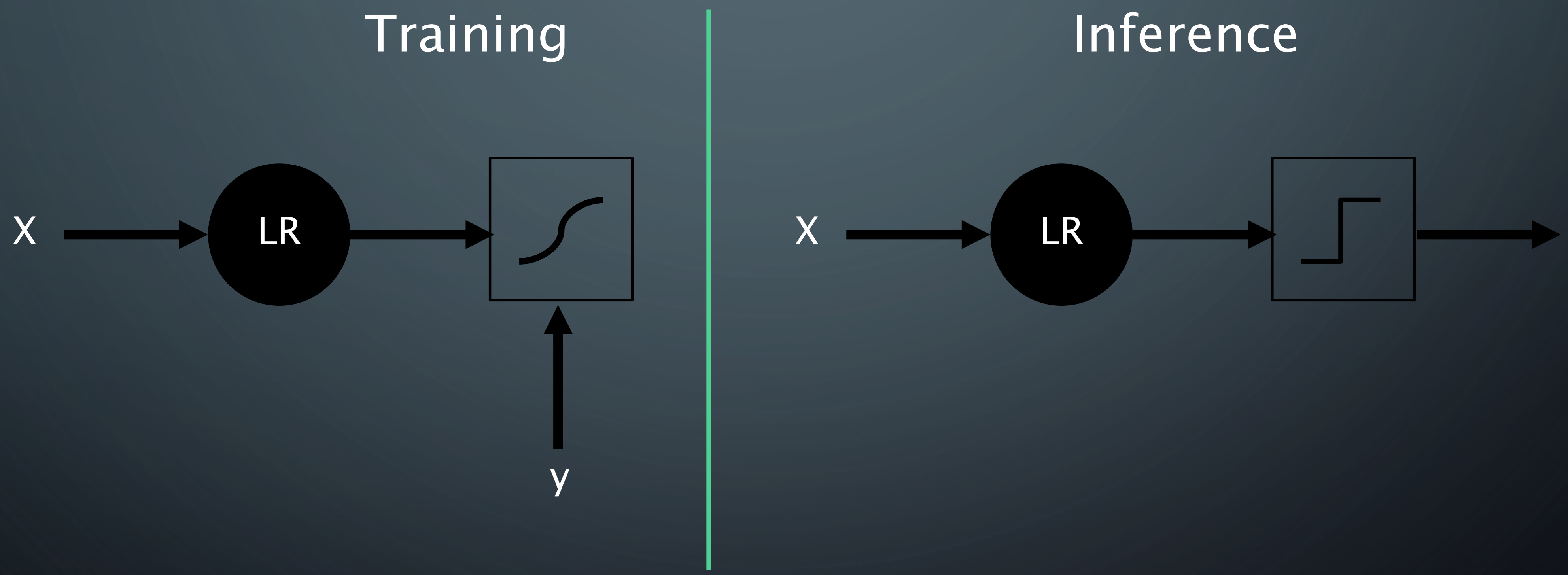


CLASSIFICATION



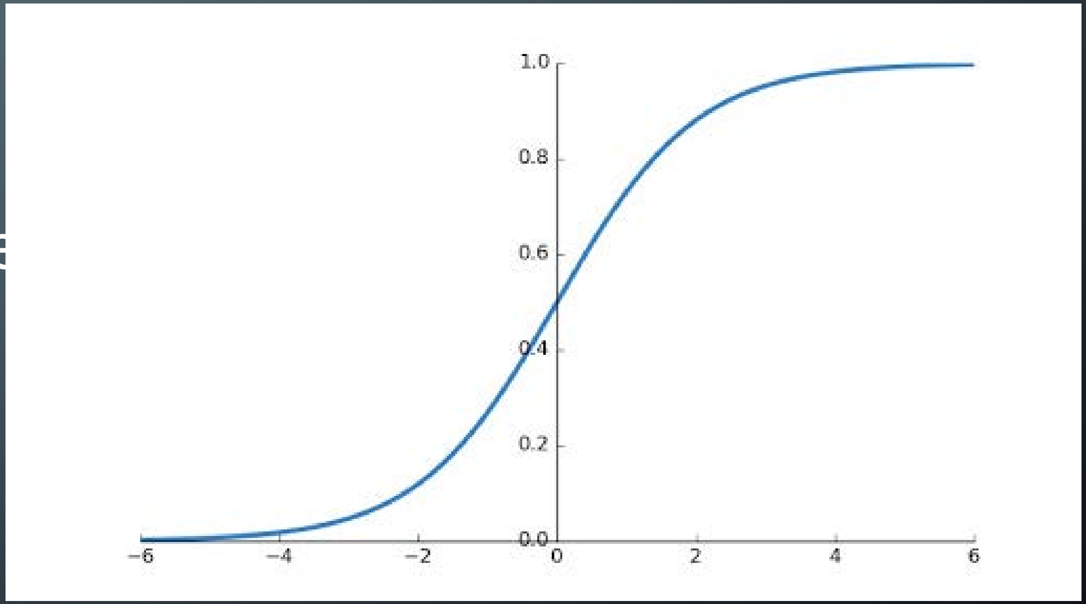


CLASSIFICATION



SIGMOID FUNCTION

- Output is a probability
- Output is between 0 and 1
- It has nice and smooth derivative



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

$$\sigma'^{(z)} = \sigma(z) * (1 - \sigma(z))$$

LOGISTIC REGRESSION (OUTPUT)

$$z = \sum_{i=1}^{N} w_i x_i$$

$$h = \frac{1}{1 + e^{-z}}$$

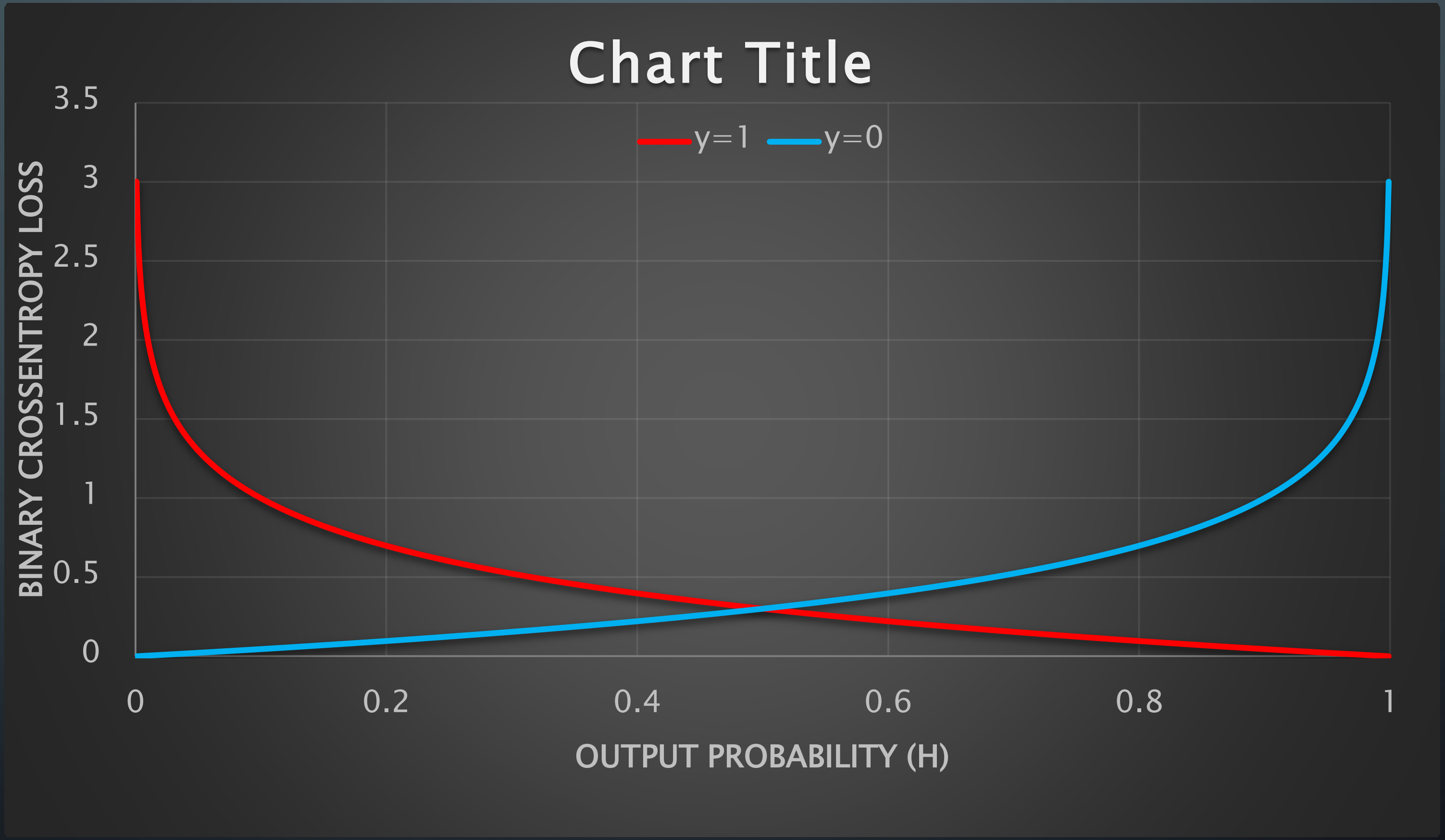
LOGISTIC REGRESSION (COST)

$$J = \frac{1}{N}\sum_{i=1}^{N} -y^{(i)} \;\; * \log\left(h^{(i)}\right) \; - \left(1 - y^{(i)}\right) * \log(1 - h^{(i)})$$

$$loss = \; -y \;\; * \log(h) \; - (1 - y) * \log(1 - h)$$

$$loss = \begin{cases} -\log(h) & y = 1 \\ -\log(1 - h) & y = 0 \end{cases}$$

LOGISTIC REGRESSION (LOSS)



LINEAR REGRESSION VS LOGISTIC REGRESSION

$$h = \sum_{i=1}^{N} w_i x_i$$

$$z = \sum_{i=1}^{N} w_i x_i \qquad h = \frac{1}{1 + e^{-z}}$$

LINEAR REGRESSION VS LOGISTIC REGRESSION

$$h = \sum_{i=1}^{N} w_i x_i$$

$$J = \frac{1}{2N} \sum_{i=1}^{N} \left(y^{(i)} - h^{(i)}\right)^2$$

$$z = \sum_{i=1}^{N} w_i x_i \qquad h = \frac{1}{1 + e^{-z}}$$

$$J = \frac{1}{N} \sum_{i=1}^{N} -y^{(i)} * \log\left(h^{(i)}\right) - \left(1 - y^{(i)}\right) * \log(1 - h^{(i)})$$

LINEAR REGRESSION VS LOGISTIC REGRESSION

$$h = \sum_{i=1}^{N} w_i x_i$$

$$J = \frac{1}{2N} \sum_{i=1}^{N} \left(y^{(i)} - h^{(i)} \right)^2$$

$$\frac{\partial J}{\partial w_j} = \frac{1}{N} \sum_{i=1}^{N} \left(h^{(i)} - y^{(i)} \right) x_j^{(i)}$$

$$z = \sum_{i=1}^{N} w_i x_i \qquad h = \frac{1}{1 + e^{-z}}$$

$$J = \frac{1}{N} \sum_{i=1}^{N} -y^{(i)} * \log\left(h^{(i)}\right) - \left(1 - y^{(i)}\right) * \log(1 - h^{(i)})$$

$$\frac{\partial J}{\partial w_j} = \frac{1}{N} \sum_{i=1}^{N} \left(h^{(i)} - y^{(i)} \right) x_j^{(i)}$$

DERIVATIVES

$$\frac{\partial z^{(i)}}{\partial w_j} = x_j^{(i)}$$

$$\frac{\partial h^{(i)}}{\partial z^{(i)}} = h^{(i)} \ * \left(1 \ - \ h^{(i)}\right)$$

$$\frac{\partial J}{\partial h^{(i)}} = \frac{-y^{(i)}}{h^{(i)}} + \frac{1 - y^{(i)}}{1 - h^{(i)}}$$

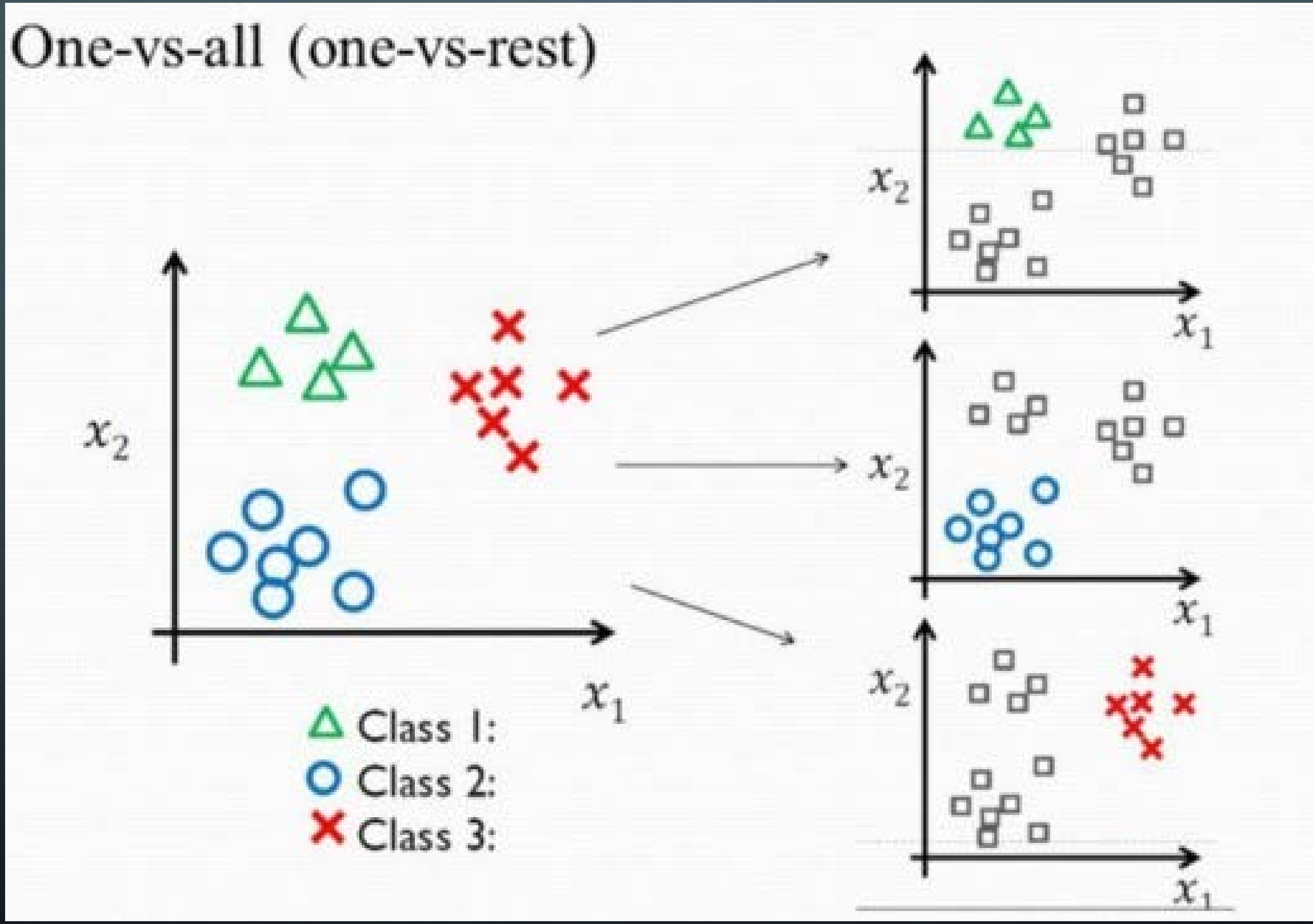
$$= \frac{h^{(i)} - y^{(i)}}{h^{(i)} \ * (1 - h^{(i)})}$$

$$\frac{\partial J}{\partial w_j} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial J}{\partial h^{(i)}} * \frac{\partial h^{(i)}}{\partial z^{(i)}} * \frac{\partial z^{(i)}}{\partial w_j}$$

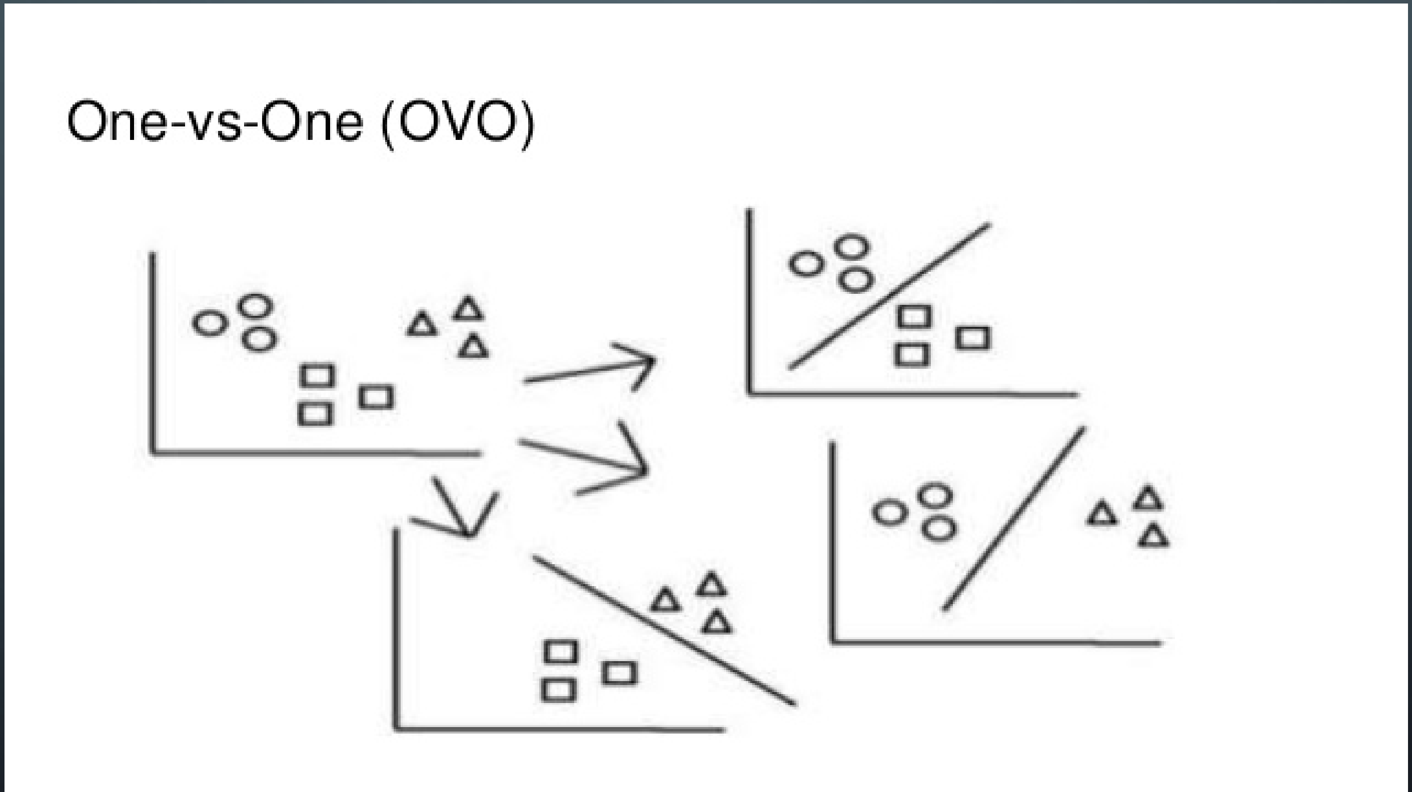
$$\frac{\partial J}{\partial w_j} = \frac{1}{N} \sum_{i=1}^{N} \left(h^{(i)} - y^{(i)}\right) * x_j^{(i)}$$

jupyter

MULTICLASS CLASSIFICATION(OVR)



MULTICLASS CLASSIFICATION(OVO)



BINARY LOGISTIC REGRESSION AS MATRIX MULTIPLICATION

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} = \sigma\left(\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_f^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_f^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \dots & x_f^{(N)} \end{bmatrix} . \begin{bmatrix} w_1 & w_2 & \dots & w_M & w_f \end{bmatrix}^T \right)$$

$$Y = \sigma(X.W^T)$$

BINARY LOGISTIC REGRESSION AS MATRIX MULTIPLICATION

$$\begin{bmatrix} y_1^{(1)} & y_2^{(1)} & \cdots & y_c^{(1)} \\ y_1^{(2)} & y_2^{(2)} & \cdots & y_c^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ y_1^{(N)} & y_2^{(N)} & \cdots & y_c^{(N)} \end{bmatrix} = \sigma\left(\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_f^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_f^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \cdots & x_f^{(N)} \end{bmatrix} . \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,f} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,f} \\ \vdots & \vdots & \ddots & \vdots \\ w_{c,1} & w_{c,2} & \cdots & w_{c,f} \end{bmatrix}^T \right)$$

$$Y = \sigma(X.W^T)$$

jupyter