

Binary Classification

- The Red, Blue, Green :

$$\begin{bmatrix} 225 \\ 231 \\ \vdots \\ 255 \\ 134 \\ \vdots \\ 225 \end{bmatrix}$$

- The Binary :
- given the x, y :

$$(x, y), x \in \vec{R}^{n_x}, y \in \{0, 1\}$$

- The ruled the m is training example :

$$\{(x^1, y^1), (x^2, y^2), \dots, (x^m, y^m)\}$$

- ruled the $m_{test} = m$ test example
- The ruled a matrix X :

$$X = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ x^1 & x^2 & \dots & x^m \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}, x \in \vec{R}^{n_x}, X \in \vec{R}^{n_x \times m}$$

- The ruled a matrix Y :

$$Y = [y^1 \quad y^2 \quad \dots \quad y^m], y \in \{0, 1\}, y \in \vec{R}^{1 \times m}$$

Remember the X can't do the X^T

In Python:

- `X.shape(n_x , m), Y.shape(1, m)`

- The X row is n_x , the col is m

- The Y row is 1, the col is m

Logistic Regression

- Given 'x', The probability is :

$$\hat{y} = P \times (y = 1|x), x \in \vec{R}^{n_x}$$

- The logistic parameter is :

$$w \in \vec{R}^{n_x}, b \in R$$

- The Output is :

$$\hat{y} = w^T \times x + b, 0 \leq y \leq 1$$

- In this way, we can get the sigmoid function :

$$\hat{y} = \sigma(w^T \times x + b), z = w^T + b \Rightarrow \sigma(z) = \frac{1}{1 + e^{-z}}$$

- If the $z \rightarrow +\infty$:

$$\sigma(z) = \lim \frac{1}{1 + 0} = 1$$

- If the $z \rightarrow 0$ or $-\infty$:

$$\sigma(z) = \lim_{z \rightarrow -\infty} \frac{1}{1 + e^z} = 0$$

- But if someone designed the $x_0 = 1$:

$$x \in \mathbb{R}^{n_x+1}, \hat{y} = \sigma(\theta^T \times x) \quad \text{with } \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \quad \text{where } b = \theta_0, w = \{\theta_1, \theta_2, \dots, \theta_{n_x}\}$$

Logistic Regression Lost Function

train the logistic regression's parameters : w and b

- Given the m (m_{test}) training examples:

$$\{(x^1, y^1), (x^2, y^2), \dots, (x^m, y^m)\}$$

- From them to gain the w and b, and to gain the $\hat{y}^{(i)} \rightarrow y^{(i)}$

The Lost(Error) Function

To gain or maybe to say suit for the single training

$$\text{def } L(\hat{y}, y) = \frac{1}{2} (\hat{y} - y)^2$$

But the result leads to the error of the real, the image is uneven

- To deal with the bug :

$$\text{def } L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

- If $y = 1$, $L(\hat{y}, y) = -\log \hat{y}$:

$$\Rightarrow \log \hat{y} \rightarrow +\infty, 0 \leq \hat{y} \leq 1 \Rightarrow \lim \hat{y} = 1$$

- If $y = 0$, $L(\hat{y}, y) = -\log(1 - \hat{y})$:

$$\Rightarrow \log(1 - \hat{y}) \rightarrow +\infty, 0 \leq \hat{y} \leq 1 \Rightarrow \lim \hat{y} = 0$$

Cost Function

For the whole training examples

$$\text{def } J(w, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m} \sum_{i=1}^m [-y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

Gradient Descent

We already have known that:

$$\hat{y} = \sigma(w^T x + b), \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

For logistic regression, almost any initialization method will work

- gradient descent

Start at the initial point and move towards the steepest downhill, then keep moving towards the steepest downhill

- Having this **trend**

Repeat : Learn Rate : α // The $J(w, b)$ use as $J(w)$ // $w := w - \alpha \frac{dJ(w)}{dw} \Rightarrow w := w -$

Gradient Descent moves towards the global minimum

- So We can gain the method of the Gradient Descent

$$w := w - \alpha \frac{\partial(w, b)}{\partial w}, b := b - \alpha \frac{\partial(w, b)}{\partial b}$$

In Python Code :

The math function $\frac{\partial(w, b)}{\partial w}$ was credited as dw

Alternatively, all integrals are denoted as dx

The Logistic Regression Gradient Descent

- given the m_{test} :

$$\begin{bmatrix} w_1 & w_2 \\ x_1 & x_2 \end{bmatrix}, b$$

- then have this function list :

$$z = w_1x_1 + w_2x_2 + b \rightarrow \alpha = \sigma(z) \rightarrow L(a, y)$$

- for the L to α :

$$d\alpha = \frac{dL(a, y)}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$$

- for the α to z :

$$\frac{d\alpha}{dz} = \alpha(1-\alpha)$$

- for the L to z :

$$dz = \frac{\partial L(a, y)}{\partial z} = \frac{\partial L(a, y)}{\partial \alpha} \cdot \frac{d\alpha}{dz} = a - y$$

- for the L to w :

$$dw_m = \frac{\partial L(a, y)}{\partial w_m} = x_m \cdot dz = x_m(a - y)$$

- for the L to b :

$$db = \frac{\partial L(a, y)}{\partial b} = dz = a - y$$

Then we can use the :

$$w := w - \alpha dw$$

$$b := b - \alpha db$$

to update the params w and b

