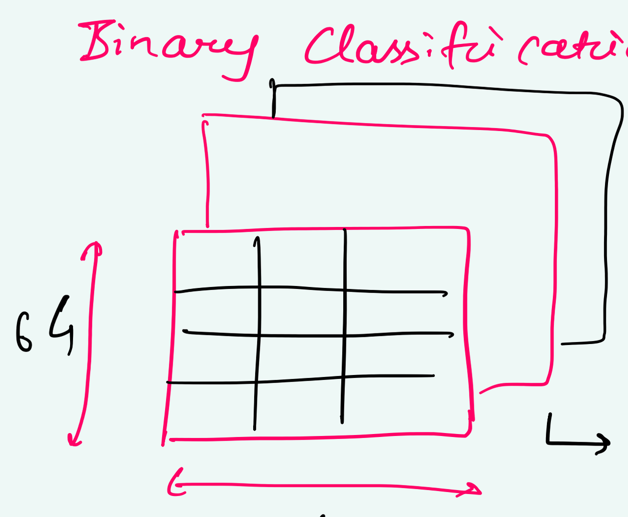
Week2

Monday, April 20, 2020 6:22 PM

Binary Classification



$x = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$ $64 * 64 * 3$

↳ depth is 3

↳ a picture of cat $64 * 64$ pixels. → cat yes/no

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

* m training examples → result is there with every x ex.

↳ $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots (x^{(m)}, y^{(m)})\}$ Pair of x & y.

Mtrain & Mtest → notations.

RGB. depth {in case of image}

$* \; x^{(1)} =$ [ ] $64*64$, [ ] $64*64$, [ ] $64*64$

Now

$X = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ \vdots & \vdots & & \vdots \end{bmatrix}$ $\updownarrow n_x$ → $64 \times 64 * \boxed{3}$

$\longleftarrow m \longrightarrow$ - - → no. of images.

↳ IMPORTANT

X.shape = (n_x, m) $n_x = 64 * 64 * 3$.

$Y = [y^{(1)} \; y^{(2)} \; y^{(3)} \dots, y^{(m)}]$

$y \in \mathbb{R}^{1*m}$ Y.shape = $(1, m)$

Logistic Regression

↳ $\hat{y} = P(y = 1 / x)$ $\quad 0 \le \hat{y} \le 1\}$ → must be b/w 0 & 1.

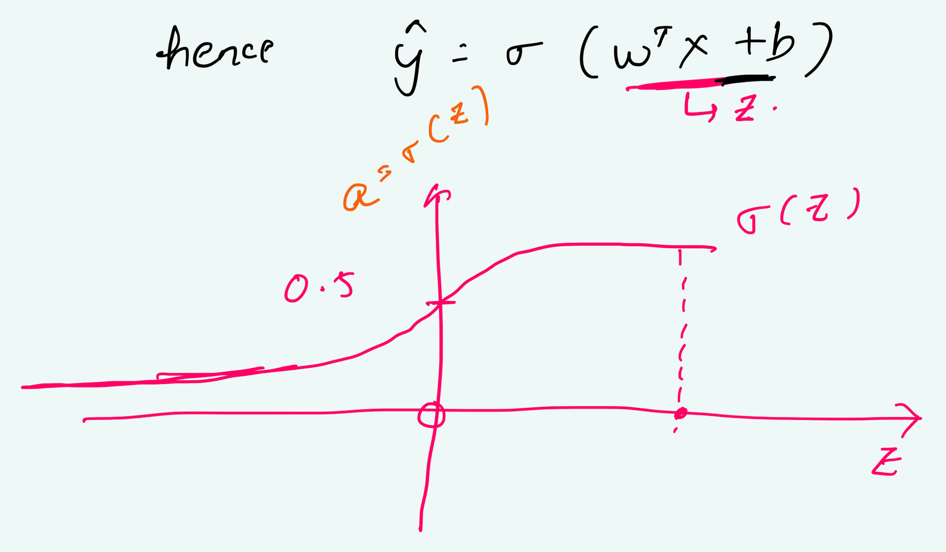
↳ Result of logistic regression.

param → $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

output → $\hat{y} = w^T x + b$. {range can vary} ∴ squeeze it b/w 0 & 1

hence $\hat{y} = \sigma(\underbrace{w^T x + b}_{z})$

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



Aim → to learn w and b. to compute $\hat{y}^{(i)}$

→ $z^{(i)} = w^T x^{(i)} + b$ $\left.\begin{matrix} x^{(i)} \\ y^{(i)} \\ z^{(i)} \end{matrix}\right\}$ i^{th} training example

Important.

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

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

$\hat{y}$ as big as possible ↳ $L(\hat{y}, y)$ → small.

If $y = 0$; $L(\hat{y}, y) = -\log(1 - \hat{y})$ small / large ∴ $\hat{y}$ as small.

avg of all loss f^n across entire training set.

Cost function:- Cost of parameters. {find w & b such that it reduces}

$J(w, b) = \dfrac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$

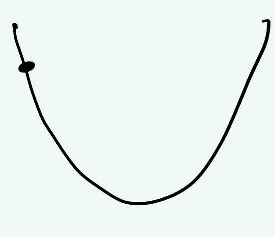
$= -\dfrac{1}{m} \sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$ {Avg across all examples}

Gradient Descent

$J(w, b)$ → convex f^n

↓

reason to use this



Global minima

Repeat {

$w := w - \alpha \dfrac{d(J(w))}{d(w)}$

learning rate works on ↑ & ↓ of slope.

rate of change. {slope}

} until it converges

dw → variable for derivative

$b := b - \alpha \dfrac{d(J(w))}{d(b)}$ } Important.

Computation Graphs:-

main aim → find $\boxed{\dfrac{d(\text{Final output})}{d\,\text{var}}}$ → dvar

a = 5, b = 3, c = 2

$u = bc$ → $v = a + u$ → $J = 3v$

a	u	v	J
			6 +3
1	1	2	9 +3
1	2	3	12
1	3	4	

$\dfrac{dJ}{du} = 3$ → $\dfrac{dJ}{dv} \cdot \dfrac{dv}{du}$ = 3

$\quad\quad 3 \quad\; 1$

In case of logistic regression!

x_1, w_1, x_2, w_2, b → $\boxed{z = w_1 x_1 + w_2 x_2 + b}$ → $\boxed{a = \sigma(z)}$ → $\boxed{L(a, y)}$

$'dz' = \dfrac{dL}{dz}$ $\quad 'da' = \dfrac{dL}{da}$

∵ we have 'da'.

$\dfrac{dL}{da} \cdot \dfrac{da}{dz}$ } chain rule

$\dfrac{dL}{dz} = \dfrac{dL}{da} \cdot \dfrac{da}{dz}$

$-(y \log a + (1-y) \log(1-a))$

$= \dfrac{1}{a} + \dfrac{1}{1-a}$

$\dfrac{\partial L}{\partial w_1} =$ "dw_1" $= x_1 . dz$. $\quad dw_2 = x_2 . dz$ $\quad db = dz$

learning rate.

$w_1 := w_1 - \alpha \, dw_1$

$w_2 = w_2 - \alpha \, dw_2$

$b = b - \alpha \, db$

} update rule.

coding tips :- np.random.randn (5)

→ rank 1 array

↳ Do not use.

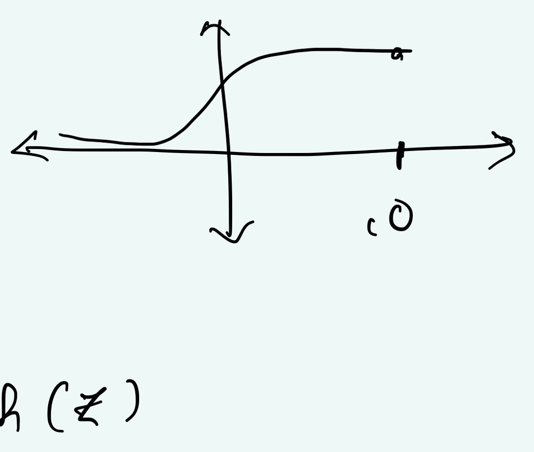
Derivatives of Activation function

$g(z) = \dfrac{1}{1 + e^{-z}}$. $\quad \dfrac{d(g(z))}{dz} = g(z) * (1 - g(z))$

↳ $g'(z)$ $\quad$ ↳ $a * (1 - a)$

$z > 10$

$g'(z) \approx 0$



$g(z) = \tanh(z)$

$g'(z) = (1 - \tanh^2(z))$