Week-2

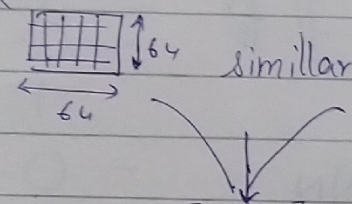
Logistic Regression is a type of binary classification

Binary Classification

example: Cat or Non-Cat

→ The image ~~vectors~~ pixel matrices are combined for input vector

Consider 64p image



input vector $X = \begin{bmatrix} \} R \\ \} G \\ \} B \end{bmatrix}$ here, $n = n_x = 12288$ $(64 \times 64 \times 3)$

Single traing example is represented as (x, y) $x \in \mathbb{R}^{n_x}$, $y \in \{0, 1\}$
(feature vector) (label)

m # training examples

$(x^{(i)}, y^{(i)})$ → i^{th} training example $(i < m)$

M_{train}, M_{test}

input matrix $X = \begin{bmatrix} | & | & | & & | \\ x^{(1)} & x^{(2)} & x^{(3)} & \cdots & x^{(m)} \\ | & | & | & & | \end{bmatrix}$ n_x

training set

m

$X \in \mathbb{R}^{n_x \times m}$

$X.shape = (n_x, m)$

$Y = [y^{(1)} y^{(2)} \cdots y^{(m)}]$

$Y \in \mathbb{R}^{1 \times m}$

$y.shape = (1, m)$

for this problem, given x we want $\hat{y}$.

where, $\hat{y} = P(y = 1 | x)$ [probability of a cat image for input x]

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

Output $\hat{y} = w^T x + b$ what used to be in linear Regression

But, the output may be +ve or -ve.

So, instead, we use

$\hat{y} = \sigma(w^T x + b)$ for contraining the output

$\sigma(z) = \frac{1}{1+e^{-z}}$ if z is large $\sigma(z) \approx 1$

if z is small $\sigma(z) \approx 0$

My Note:

we take $w^T x$ to get a scalar ~~product~~ value where in we can define a line (x, y). Though, we will use individual values of each vector to generate individual vectors. Next, we combine these vectors in a matrix. In order to get $\vec{x}$ & $\vec{y}$ of similar size

Loss (error) function:

$L(\hat{y}, y) = -(y \log \hat{y} + (1-y) \log(1-\hat{y}))$ [why this?]

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

So, in order to minimize error, we need to have large $\log \hat{y}$.

In order to get $\log \hat{y}$ ↑, we need to have large $\hat{y}$ [graph of log]

sigmoid's largest & smallest values

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

here, $\log(1-\hat{y})$ needs to be large

So, $(1-\hat{y})$ needs to be large

Thus, $\hat{y}$ needs to be really small (very -ve)

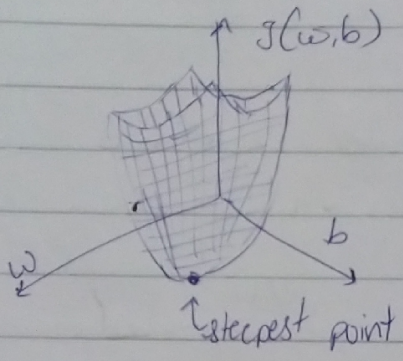
Cost function:

It indicates how good your training is doing.

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

So, we try to adjust the parameters in a way that cost function minimize

Gradient Descent



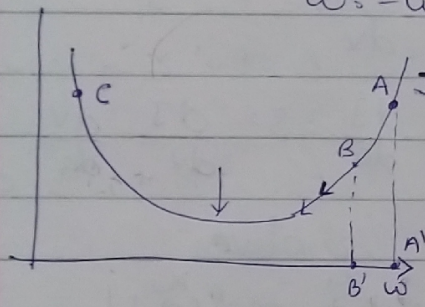
Cost function is a convex function
The gradient descent tries to minimize the cost function.
← Assume w as a real number here, for understanding

What does gradient descent do?

Repeat {

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

(α: learning Rate; $\frac{d}{dw}J(w)$: derivative is actually slope of the function)



Here, only 2-D graph is used for simplicity.

Initially value of w is A' an J(w) is A. The derivative at A is positive, so, weights are updated, and we reach B.
At point C, the slope would be -ve and hence weight will be added.
Finally global minimum is achieved.

Similarly, in real scenario, gradient descent will change w and b as:

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

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

since J is a function of w and b, instead of d, we use ∂ to denote partial derivation

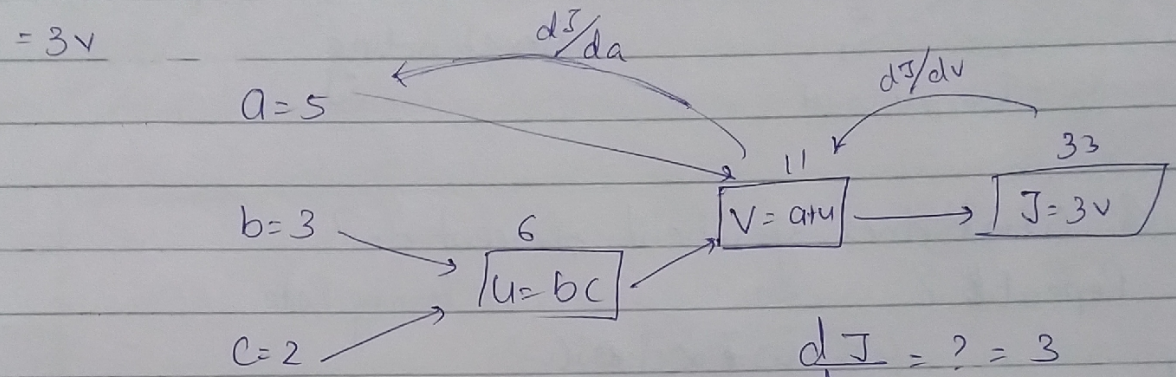
Computat^n Graph

$J(a,b,c) = 3(a+bc)$

$u = bc$

$v = a+u$

$J = 3v$



forward → get output

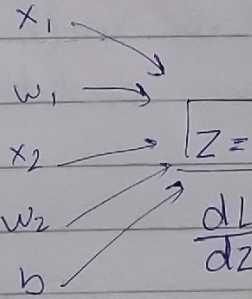
backward → get derivatives

$\frac{dJ}{dv} = ? = 3$

$\frac{dJ}{da} = 3 = \frac{dJ}{dv}\frac{dv}{da}$ chain rule

Now, we take Logistic regression derivatives.

Assuming we have two variables ~~x1~~ and ~~x2~~ x_1 x_2



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

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

$\frac{dL}{dz} = \frac{dL}{da} \times \frac{da}{dz}$ —(ii)

$\frac{dL(a,y)}{da} = -\left[\frac{y}{a} - \frac{(1-y)}{1-a}\right]$ —(i)

$= \left[\frac{-y + (1-y)}{a \quad (1-a)}\right]\left[\frac{1}{z}\right]$

$\frac{da}{dz} = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) = \frac{d}{dz}\left(1+e^{-z}\right)^{-1} = -\frac{d/dz(1+e^{-z})}{(1+e^{-z})^2}$

$= -\left[\frac{-(0+e^{-z})}{(1+e^{-z})^2}\right] = \frac{e^{-z}}{(1+e^{-z})^2}$

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

$= \frac{1}{(1+e^{-z})} \cdot \left[\frac{(1+e^{-z})}{(1+e^{-z})} - \frac{1}{(1+e^{-z})}\right] = \sigma(z)\left[1 - \frac{1}{\sigma(z)}\right]$

$= a[1-a]$ —(iii)

$$= \frac{dL}{da} \times \frac{da}{dz}$$

$$= \left[\frac{-y}{a} + \frac{(1-y)}{1-a}\right] a(1-a) \quad \text{from I, II \& III}$$

$$= -y(1-a) + (1-y)a$$

$$= -y + ay + a - ay$$

$$= a - y \quad -(IV)$$

$$\frac{dL}{dw_1} = \frac{dL}{dz} \times \frac{dz}{dw_1} \quad -(V) \quad \text{\& similar for } w_2$$

$$\frac{dz}{dw_1} = \frac{d}{dw_1}[w_1 x_1 + w_2 x_2 + b] \qquad \frac{dz}{db} = 1 \quad -(VII)$$

$$= x_1 \quad -(VI) \qquad \text{similar for } w_2$$

$$\therefore \frac{dL}{dw_1} = x_1(a-y) \quad (VII) \quad \& \quad \frac{dL}{dw_2} = x_2(a-y) \quad \text{from IV, V, VI}$$

$$\frac{dL}{db} = (a-y) = dz \quad \text{from V}$$

Logistic Regression on m examples

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

$$= \frac{1}{m} \sum_{i=1}^{m} a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b)$$

Here, derivative of cost function will be,

$$\frac{d}{dw} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \frac{dL}{dw}(a^{(i)}, y^{(i)})$$

$dw^{(i)}$ is a function of $x, y \& \hat{y}$ (from VII)

Vectorization

It is used to reduce the for loops in a code

$z = np.dot(w, x) + b$

eliminating 1st for loop for weights

$J=0, dw_1=0, dw_2=0, db=0$ → $dw = np.zeros((n_x, 1))$

for i = 1 to m:

$z^{(i)} = w^T x^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J += -[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)})]$

$dz^{(i)} = a^{(i)}(1-a^{(i)})$

$dw_1 += x_1^{(i)} dz^{(i)}$

$dw_2 += x_2^{(i)} dz^{(i)}$

$dw += x^{(i)} dz^{(i)}$

$db += dz^{(i)}$

$J = J/m$ $dw_1 = dw_1/m$, $dw_2 = dw_2/m$, $db = db/m$

$dw /= m$

eliminating 2nd for loop

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \qquad W^T = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

X.shape = $n_x \times m$ $\qquad$ w^T.shape = $n_x \times m$

$$Z = [z^{(1)} \; z^{(2)} \cdots z^{(m)}] = w^T X + [b \; b \cdots b] = [w^T x^{(1)} + b \;\; w^T x^{(2)} + b \;\; \cdots \;\; w^T x^{(m)} + b]$$

$1 \times m$

$a^{(1)} = \sigma(z^{(1)}) \quad a^{(2)} = \sigma(z^{(2)})$

$A = [a^{(1)} \; a^{(2)} \cdots a^{(m)}] = \sigma(Z)$

$$db = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)} = \frac{1}{m} np.sum(dz)$$

$$dw = \frac{1}{m} X dz^T = \frac{1}{m} \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ dz^{(2)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$$

$$= \frac{1}{m} np.dot(X, dz)$$

Z = np.zeros(1, m)

Z = np.dot(w.T, X) + b

A = σ(Z)

~~X / Y / m /~~ ← Not written cost funct^n

dZ = A - Y

dW = $\frac{1}{m}$ np.Sum(~~x~~ dZ. T)

$dW = \frac{1}{m}$ ~~np/sum~~ (np.dot(X, dZ.T))

$db = \frac{1}{m}$ np.sum(dZ)

$W := W - \alpha \cdot dW$ ~~dot np.dot~~

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

Broadcasting

$\begin{bmatrix}1\\2\\3\end{bmatrix} + 100 \Rightarrow \begin{bmatrix}1\\2\\3\end{bmatrix} + \begin{bmatrix}100\\100\\100\end{bmatrix} = \begin{bmatrix}101\\102\\103\end{bmatrix}$

[1 2 3] + 100 => [1 2 3] + [100 100 100] = [101 102 103]

Tip: use np.random.randn(5,1) instead of (5) ← rank 1 array should not be used

used assertions assert(a.shape == (5,1))

~~Loss function Need~~

~~if y=1, we need~~

Programming Assignment

db = sum(dZ, axis=1) $dW = \underset{2\times3}{X} \cdot \underset{3\times1}{dZ^T} = 2\times1$

Propagate function

w	(2, 1)
X	(2, 3)
Y	(1, 3)

$w = \begin{bmatrix}\\\end{bmatrix}_{2\times1}$ $X = \begin{bmatrix}\\\end{bmatrix}_{2\times3}$ features $Y = \begin{bmatrix}\\\end{bmatrix}_{1\times3}$ o/p

(1,3) · (1,3)

$Z = \underset{1\times2}{w^T} \times \underset{2\times3}{X} + b = 1\times3 + b$

$\begin{bmatrix}\\\\\end{bmatrix}^T$ $A = \begin{bmatrix}\\\\\end{bmatrix}^T$ $dZ = \begin{bmatrix}\end{bmatrix}^T - \begin{bmatrix}\end{bmatrix}^T$

σ(Z) 1×3