17-7-24

Page No.
Date

II) BASICS OF NEURAL NETWORK

1. LOGISTIC REGRESSION → ALGORITHM FOR BINARY CLASSIFICATION

- IMAGE READING: an image has 3 channels RBG. an $m \times n$ PIXEL IMAGE will be read as 3 matrices each of size $m \times n$.
- ∴ INPUT ~~vect~~ FEATURE VECTOR will have a DIMENSION OF $m \times n \times 3 = n_x$.

FEATURE VECTOR

NOTATION:

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

$m \rightarrow$ NO. OF TRAINING examples

m_{train} = training examples m_{test} = testing examples

$$X = \begin{bmatrix} | & | & | & & | \\ x^1 & x^2 & x^3 & \cdots & x^m \\ | & | & | & & | \end{bmatrix} \uparrow n_x \quad \leftarrow m \rightarrow$$

$$Y = \begin{bmatrix} y^1, y^2, y^3 \cdots y^m \end{bmatrix}$$

$X.shape = (n_x, m)$

$Y.shape = (1, m)$

- LOGISTIC REGRESSION ⇒ USES SIGMOID FUNCTION
  ↳ aims TO HAVE a PROBABILITY OUTPUT. (B/W 0 & 1)

∴ $\text{OUTPUT}(\hat{y}) = \sigma(w^T x + b)$

WHERE σ (SIGMOID) is $\sigma(z) = \frac{1}{1 + e^{-z}}$

⇒ HERE, w is an n_x DIMENSIONAL VECTOR
& b is a real number

$P(y|x) = \hat{y}^{y}(1-\hat{y})^{(1-y)}$

If $y=1$: $P(y|x) = \hat{y}$ & If $y=0$ $P(y|x) = 1-\hat{y}$

16/7/24 $\therefore \log[P(y|x)] = y(\log \hat{y}) + (1-y)\log(1-\hat{y})$

Page No. Date

2. LOGISTIC REGRESSION COST-FUNCTION.

⇒ LOSS FUNCTION: $L(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$

where $\hat{y}$ ⇒ generated output (predicted) & y = true label.

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

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

want $\hat{y}$ very small ←

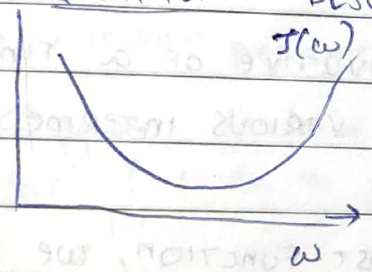
- SO LOSS FUNCTION CHECKS DIFFERENCE FOR 1 TRAINING EXAMPLE

⇒ COST FUNCTION: → mean of LOSS FUNCTIONS OF all TRAINING EXAMPLES

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

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log \hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right)$$

3. GRADIENT DESCENT ⇒ AIMS TO MINIMIZE COST FUNCTION



Repeat {

$$w := w - \alpha\left(\frac{dJ(w)}{dw}\right)$$

learning rate

If cost function depends on 2 variables $J(w,b)$: then

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

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

* COST FUNCTION ⇒ SHOULD BE A CONVEX FUNCTION ⇒ TO ENSURE ONLY 1 OPTIMAL SOLN.

FOR WHICH

Date

4. COMPUTATION GRAPH → DIRECTED GRAPH THAT IS USED FOR EXPRESSING & EVALUATING MATHEMATICAL EXPRESSIONS

USED FOR

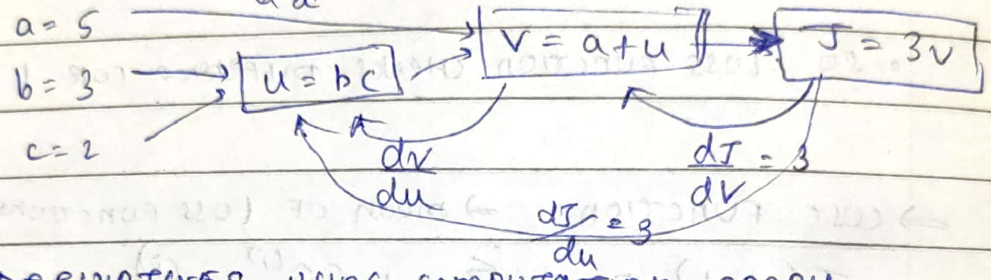
FORWARD COMPUTATION

BACKWARD COMPUTATION.

example: $J(a,b,c) = 3(a+bc) = 3(5+3\times2) = 33$

$u = bc$ $\quad v = a+u$ $\quad J = 3v$

$\frac{dJ}{da} = 3$



⇒ DERIVATIVES USING COMPUTATION GRAPH.

* COMPUTATION GRAPH → FORWARD COMPUTATION → YOU CAN CALCULATE THE COST FUNCTION

→ BACKWARD COMPUTATION → TO CALCULATE THE DERIVATIVES TO FIND GRAD

⇒ d_{var} ⇒ REPRESENTS DERIVATIVE OF A FINAL OUTPUT VARIABLE WRT VARIOUS INTERMEDIATE QTY.

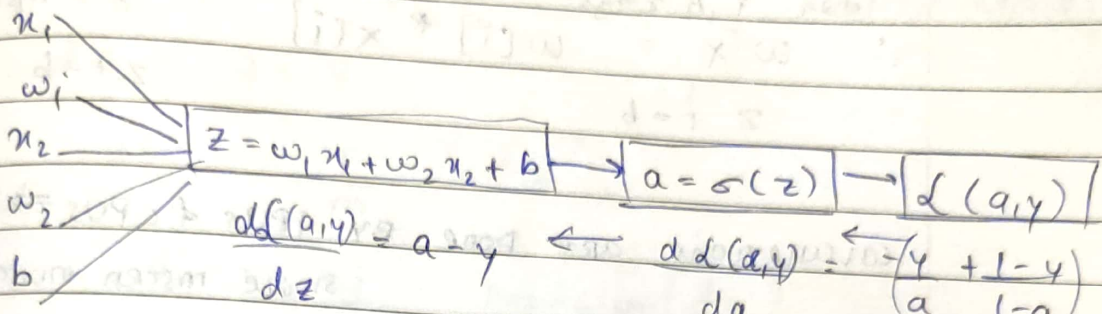
* PREV PAGE: TO MINIMIZE COST FUNCTION, WE USE MAXIMUM LIKELIHOOD ESTIMATION TO FIND THE RIGHT PARAMETERS.

Page No.
Date

5. GRADIENT DESCENT IN LOGISTIC REGRESSION.

$z = w^T x + b \qquad \hat{y} = a = \sigma(z)$

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



$\frac{d\mathcal{L}}{dw_1} = dw_1 = x_1 dz \qquad \frac{d\mathcal{L}}{dw_2} = x_2 dz$

$\frac{d\mathcal{L}}{db} = dz.$

∴ CHANGE PARAMETERS TO IMPLEMENT 1 STEP OF GRADIENT DESCENT.

$w_1 := w_1 - \alpha dw_1$

$w_2 := w_2 - \alpha dw_2$

$b := b - \alpha db.$

⇒ GRADIENT DESCENT FOR m TRAINING EXAMPLES IN LR

Suppose features: $dw_1 = 0$, $dw_2 = 0$ & $db = 0$, $J = 0$

for $i = 1$ to m

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

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

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

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

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

$dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}$ } If only 2 features are there.

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

$J /= m, \quad dw_1 /= m \quad dw_2 /= m \quad db /= m;$

UPDATE PARAMETERS

19/7/24

Page No.
Date

5. VECTORIZATION ⇒ eliminates FOR-LOOP FROM CODE
↳ makes code FASTER

1. DOT PRODUCT (MATRIX) IN FOR LOOP ⇒ np.dot(a,b)

for i in range (n-x)

$w^T x = w[i] * x[i]$ ⇒ z = np.dot(w,x)

z += b z += b

CALCULATIONS ARE DONE BY GPUs & CPUs ⇒ SIMD
↓
SINGLE INSTRN MULTIPLE DATA

2. FOR APPLYING EXPONENTIAL OPERATN TO EVERY ELEMENT IN THE MATRIX/VECTOR.

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \Rightarrow u = \begin{bmatrix} e^{v_1} \\ e^{v_2} \\ \vdots \\ e^{v_n} \end{bmatrix}$$

NON-VECTORISED

```
u = np.zeroes((n,1))
for i in range(n):
    u[i] = math.exp(v[i])
```

VECTORISED

```
import numpy as np
u = np.exp(v)
```

⇒ np.log(v) np.abs(v) np.maximum(v,0)

⇒ VECTORIZING LOGISTIC REGRESSION:

$$z = [z^{(1)}\ z^{(2)} \dots z^{(m)}] = w^T X + [b\ b\ b \dots b]$$

z = np.dot(w.T, X) + b (1×m broadcasting)

Page No.
Date

7.8. CALCULATING GRADIENT IN LR BY VECTORIZATION:

in section 5, ★. VECTORIZED code:

$z = w^T X + b$ = np.dot(w.T, X) + b.

$A = \sigma(z)$ Y = labelled output

$dz = A - Y$

$dw = \frac{1}{m} X dz^T$

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

∴ IST STEP FOR GRADIENT DESCENT DONE : $w := w - \alpha dw$

$b := b - \alpha db$.

8. BROADCASTING

FEATURE IN PYTHON ⇒ EXPANDS a VECTOR TO FORM a MATRIX TO FACILITATE +−*/

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + [100, 10, 1] \Rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 100 & 10 & 1 \\ 100 & 10 & 1 \\ 100 & 10 & 1 \end{bmatrix}$$