Week 2

Lec-§1

Binary Classification

Logistic regression (Algo for Binary classification)

Image ⟶ Yes/No (for cat pic)
(I/P) (y)
(x)

Image split into RGB channels
~~turning~~ the intensity matrix & unrolling its into a single feature vector (1D)

I/P O/P

take feature vector ⟶ cat or not?
'x' 'y'

Single training ex. (x, y) $x \in \mathbb{R}^{n_x}$, $y = \{0, 1\}$

Training ex: $\{x^1, y^1\}, \{x^2, y^2\} \dots \{x^m, y^m\}$

$m_{tr.}$ = training ex. m_{test} = testing ex

$n_x \times m$ dim matrix

$$X = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x^1 & x^2 & \cdots & x^m \\ \vdots & \vdots & & \vdots \end{bmatrix} \} n_x$$

$\underbrace{\qquad\qquad}_{m}$

X.shape = $n_x \times m$

$$Y = [y^1 \; y^2 \cdots y^m]$$

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

Lec-7

Computation graph

forward pass → o/p

backward pass → compute derivatives

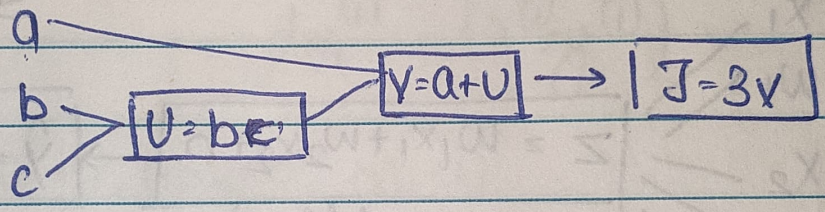
$J(a,b,c) = 3(a+bc)$

$u = bc$

$v = a+u$

$J = 3v$

} steps



Lec 8

Derivatives using a computation graph

$v+0.01 \longrightarrow J+0.03 \quad \therefore \frac{dJ}{dv} = 3$

$a+0.01 \longrightarrow v+0.01 \quad \frac{dv}{da} = 1$

$\therefore \frac{dJ}{da} = \frac{dv}{da} \times \frac{dJ}{dv}$ } chain rule

$= 3$

$\frac{dJ}{db} = \frac{dJ}{du} \cdot \frac{du}{db} = 3 \times c = 3c$

Lec-9 Logistic regression gradient degeent

Logistic regression

$z = w^T x + b$

$\hat{y} = a = \sigma(z)$ {sigmoid fn.}

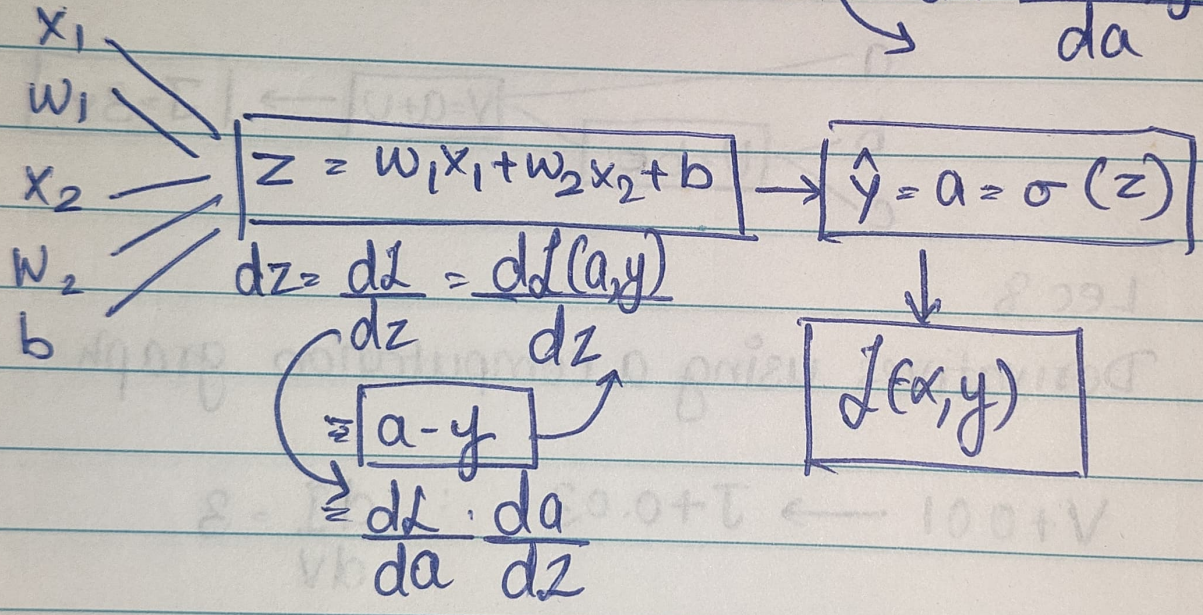
$\mathcal{L}(a,y) = -(y \log(a) + (1-y)\log(1-a))$

↳ loss fn.

(y → true val, a → o/p)

$= -\frac{y}{a} + \frac{1-y}{1-a}$

$da = \frac{d\mathcal{L}(a,y)}{da}$



Backpropogation (how much we nudge to change)

$\frac{d\mathcal{L}}{dw_1} \cdot dw_1 = x_1 \cdot dz$

$dw_2 = x_2 \cdot dz$

$dw_2 = x_2$

$db = dz$

Final:

$w_1 = w_1 - \alpha dw_1$

$w_2 = w_2 - \alpha dw_2$

$b = b - \alpha db$ ④

(Gradient descent)

Lec-10 Examples for m examples

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

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

$$\frac{\partial}{\partial w_1} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\frac{\partial}{\partial w_1} L(a^{(i)}, y^{(i)})}_{dw_1^{(i)} - (x^{(i)}, y^{(i)})}$$

$J=0, dw_1=0, dw_2=0, db=0$

for $i=1$ to m

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

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

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

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

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

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

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

$n=2$ tak

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

~~algo~~

removes explicit for loops which slows down.

Lec-11 Vectorization

vectorized

$v = \underbrace{np.dot(w, x)}_{w^T x} + b$

Lec-12 Examples of vectorization

$u = np.exp(v) \longrightarrow v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \quad u = \begin{bmatrix} e^{v_1} \\ \vdots \\ e^{v_n} \end{bmatrix}$

np.abs(v)

np.log(v)

np.maximum(v, 0)

Lec-13. vectorizing logistic regression

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

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

$(1 \times m)$

$$Z = \begin{bmatrix} | & | & & | \\ w^T x^{(1)} + b & w^T x^{(2)} + b & \cdots & w^T x^{(m)} + b \\ | & | & & | \end{bmatrix}_{1 \times m}$$

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

(broadcasting)

Lec-14 Vectorizing gradient computations

$dz^{(1)} = a^{(1)} - y^{(1)}$ $dz^{(2)} = a^{(2)} - y^{(2)} \dots$

$dz = [dz^{(1)}\ dz^{(2)} \dots\ dz^{(m)}]$

$A = [a^{(1)}\ a^{(2)} \dots\ a^{(m)}]$

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

$\boxed{dz = A - Y}$

$dw /= m$

$dw = 0$ $dw += x^{(1)} dz^{(1)}$ $dw += dz^{(2)} x^{(2)} \dots$

$db = 0$ $db += dz^{(1)}$ $db += dz^{(2)} \dots$

$db /= m$

$db = np.sum(dz)/m$

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

$\begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(m)} \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$

$[x^{(1)} dz^{(1)}\ x^{(2)} dz^{(2)} \dots]$

$n \times 1$

Lec-15 Broadcasting in python.

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

expands

Lec-16 python/numpy tips & tricks

Lec-18 Logistic regression cost function

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

interpret, $\hat{y} = P(y=1|x)$

if $y=1$: $P(y|x) = \hat{y}$

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