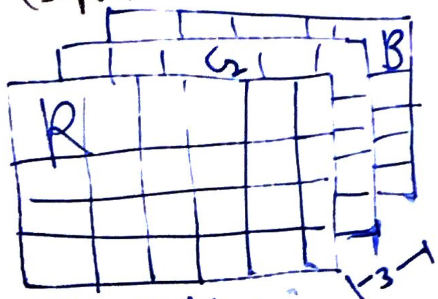
Deep learning

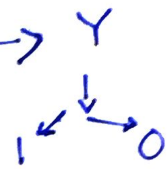
Binary Classification Problems:

eg Image (Input) ⟶ label : 0 or 1 (output label)



64 × 64

$\longrightarrow x = \begin{bmatrix} 255 \\ 231 \\ \vdots \\ 255 \\ 233 \\ \vdots \\ 255 \\ 127 \end{bmatrix} \longrightarrow$ Y



$64 \times 64 \times 3 = 12288$ 12288

notation:

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

m_{test} = # test examples = no. of test examples

$$X = \underbrace{\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & & | \end{bmatrix}}_{m} \; n_x \qquad \cancel{\begin{bmatrix} — x^{(1)T} — \\ — x^{(2)T} — \\ \vdots \\ — x^{(m)T} — \end{bmatrix}}$$

X.shape $= (n_x, m)$

makes neural network Coding HARDER

Similarly

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

Logistic Regression:

→ learning algorithm used when output labels (Y), in a supervised learning problem, are either 0 or 1 [ie for Bin. Class. Probs]

Given x, want $\hat{y} = P(y=1 \mid x)$

$x \in \mathbb{R}^{n_x}$

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

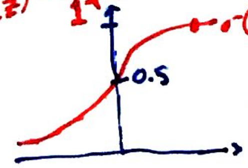
Output $\hat{y} = w^T x + b$ ← linear Regression!

∵ not to use here, it gives y = actual value

we want $P(y=1|x)$

we'd want $\hat{y} = \sigma(\underbrace{w^T x + b}_{z})$

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



Sigmoid f^n: gives values between 0 & 1

→ z large, $p \approx 1$

z small or -ve large $p \approx 0$

Big Oof:

alternate notation taught @drexel] ✱

$x_0 = 1$, $x \in R^{n_x+1}$

$\hat{y} = \sigma(\theta^T x)$

$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix}$ θ_0 } b → real no. or intercept; $\theta_1 \dots \theta_{n_x}$ } w

✱ I used this notation

✱ in log. regression project.

learn this notation from now on.

we're not gonna use this notation moving forward

QU: What's the parameters of logistic Regression?

Ans W, an n_x dimentional vector, & b, a real number

or $\theta_{1 \to n}$, θ_0

☺

✱ logistic regression can be viewed as a very small neural network

logistic Regression Cost function:

→ new stuff from ml course

→ to train the parameters w & b of logistic regression model, you need to define a cost function.

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

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

given $\underline{\{(x^{(1)}, y^{(1)}) \dots (x^{m}, y^{(m)})\}}$ we want $\hat{y}^{(i)} \approx y^{(i)}$

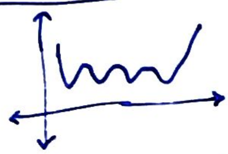
Loss (error function):

1) You could do $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ → half square error

→ this basically measures how well our algo is doing

→ If you do this however, the optimization problem (later) becomes non-convex → end up with optimization problem with lots of local optima

↓ gradient descent might not find local optima



2) another way $\boxed{L(\hat{y}, y) = -\left(\{y \log \hat{y}\} + \{(1-y) \log(1-\hat{y})\}\right)}$

rationale: if $y=1$ $L(\hat{y}, y) = -\log \hat{y}$ (← want as small as possible)

$\hat{y}$ = large → sigmoid can't be > 1

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

Cost function

→ loss f^n was defined wrt single training example

→ Cost f^n determines how well you're doing on entire training set

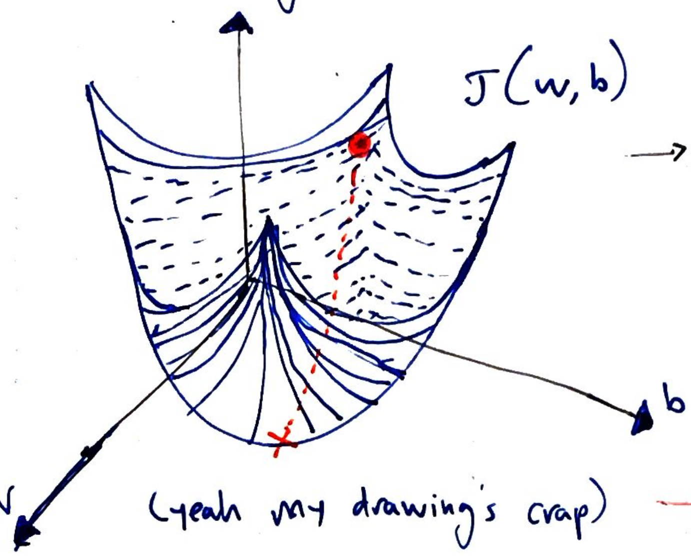
$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)}) \right]$$

basically gives average of $L(\hat{y}, y)$

→ In training log. reg. model, we're going to find params w & b that minimize overall cost f^n J.

Gradient Descent:

Now let's actually try & use Gradient descent to learn params w & b!



(yeah my drawing's crap)

→ want w, b to minimize $J(w,b)$

→ initialize w, b to some initial val

→ for log reg, anything works,

→ random → 0

→ gradient descent starts at that initial pt. & takes a step in steepest downhill direction

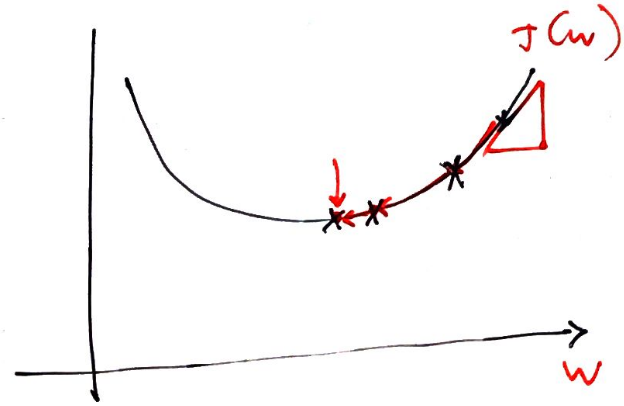
algo

Repeat {

$$w := w - \alpha \underbrace{\frac{dJ(w)}{dw}}_{\text{update, derivative}}$$

α ← learning rate

}



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

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

→ also can write as $\frac{\partial J(w,b)}{\partial w}$

* use ∂ if 2 or more params → partial deriv

d if one

means the same thing mostly ¯_(ツ)_/¯

Side Quest: Computational Graphs

Visualization trick/tool

→ these come in handy when there is some distinguished or some special output variable,

→ ~~*~~ handy in visualizing components of a much complex equation that you're tying to optimize.

→ you can visualize each component to find bottlenecks!

$J(a,b,c) = 3(a+\underbrace{bc}_{u})$ $= 3(5+3\cdot 2) = 33$

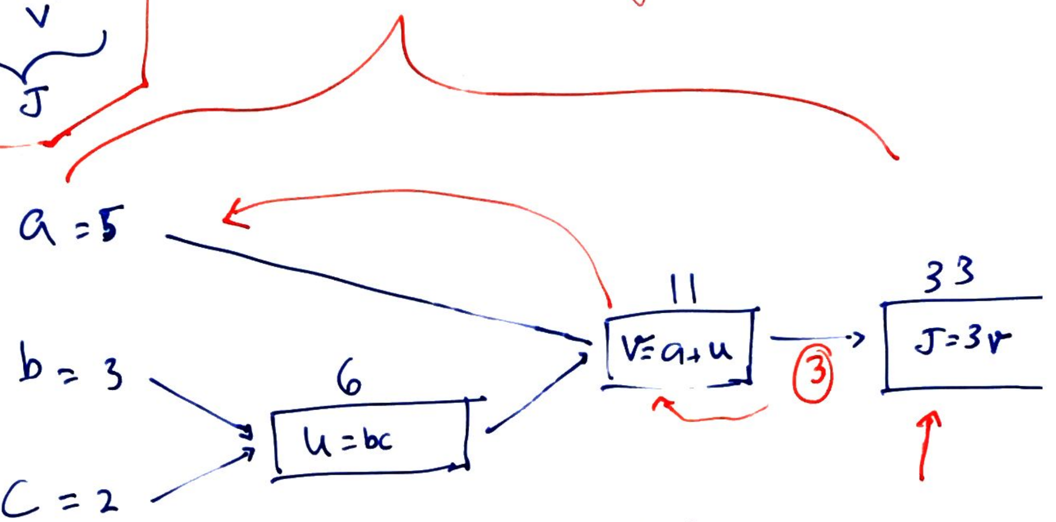
assume { $\underbrace{a+u}_{v}$, $\underbrace{3v}_{J}$ }

Computational graph

$u = bc$

$v = a+u$

$J = 3v$



for computing derivatives we move R to L

$J = 3v$

① $\frac{dJ}{dv} = ? \rightarrow = 3$

② $\frac{dJ}{da} = ?$

If we bump up a, whats effect on J?

$a = 5 \rightarrow 5.001$

$v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

$a \xrightarrow{\Delta} 0.001$ $\quad J \xrightarrow{\Delta} 0.003$

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

③ $\frac{dJ}{du} = ?$

$u = 6 \rightarrow 6.001$

$v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

$\Rightarrow \frac{dJ}{du} = 3$

i.e. $\underbrace{\frac{dJ}{dv}}_{3} \cdot \underbrace{\frac{dv}{du}}_{1}$

④ $\frac{dJ}{db} = \frac{dJ}{du} \cdot \frac{du}{db}$

$b = 3 \rightarrow 3.001$

$u = b.c = 6 \rightarrow 6.002$ $\boxed{c=2}$

$v = 11.002$

$J = 33.006$

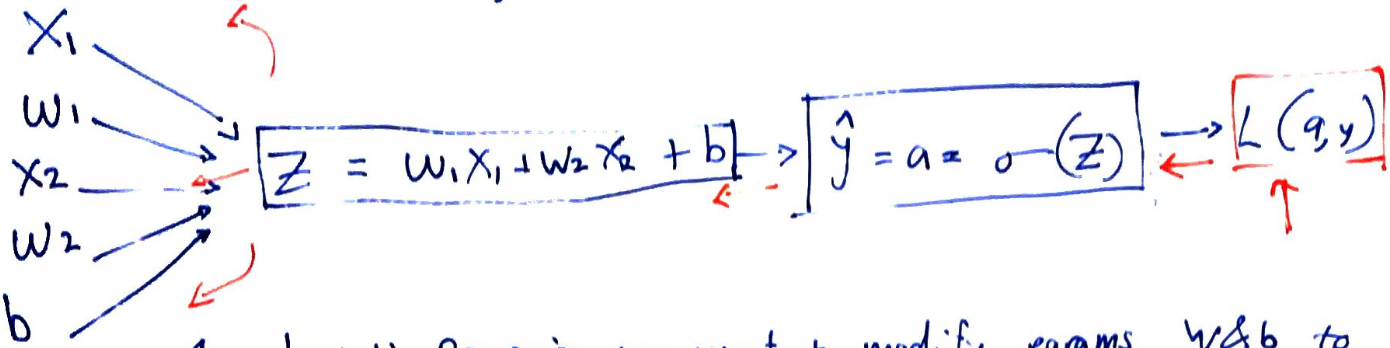
$\frac{dJ}{db} = 6$

Tldr: forward L→R ⇒ compute cost f^n

backward R→L ⇒ compute derivati

Logistic Regression Gradient Descent

for one observation: $\hat{y} = a = \sigma(z)$ $\quad z = w^T x + b$



In logistic Regression, we want to modify params w&b to reduce the loss

How you actually compute the loss on a single training eg.

we want to compute derivatives wrt this loss

1) go backward to compute derivative of loss wrt a

$$\text{"}da\text{"} = \frac{dL(a,y)}{da}$$

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

$$\boxed{dz = \frac{dL}{dz} = \frac{dL(a,y)}{dz} = a - y}$$

$$= \frac{dL}{da} \cdot \frac{da}{dz} \quad \frac{da}{dz} = a(1-a)$$

$$\frac{dL}{dw_1} = \text{"}dw_1\text{"} = x_1 \cdot dz \; ; \; dw_2 = x_2 \cdot dz \; ; \; db = dz$$

$$w_1 := w_1 - \alpha dw_1 \quad w_2 := w_2 - \alpha dw_2 \quad b := b - \alpha db$$

* formula for derivative of loss wrt z ? $a - y$

Note * this explaination is for a single observation (assuming 2 features x_1, x_2)

Logistic Regression on m examples:

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

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

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

initiate

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

for $i = 1$ to m

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

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

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

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

$\left.\begin{array}{l} dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)} \\ dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)} \end{array}\right\}$ assuming 2 features $n=2$

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

$J /= m$

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

$\partial w_1 = \frac{\partial J}{\partial w_1}$

$w_1 := w_1 - \alpha dw_1$

$w_2 := w_2 - \alpha dw_2$

$b := b - \alpha db$

Vectorization