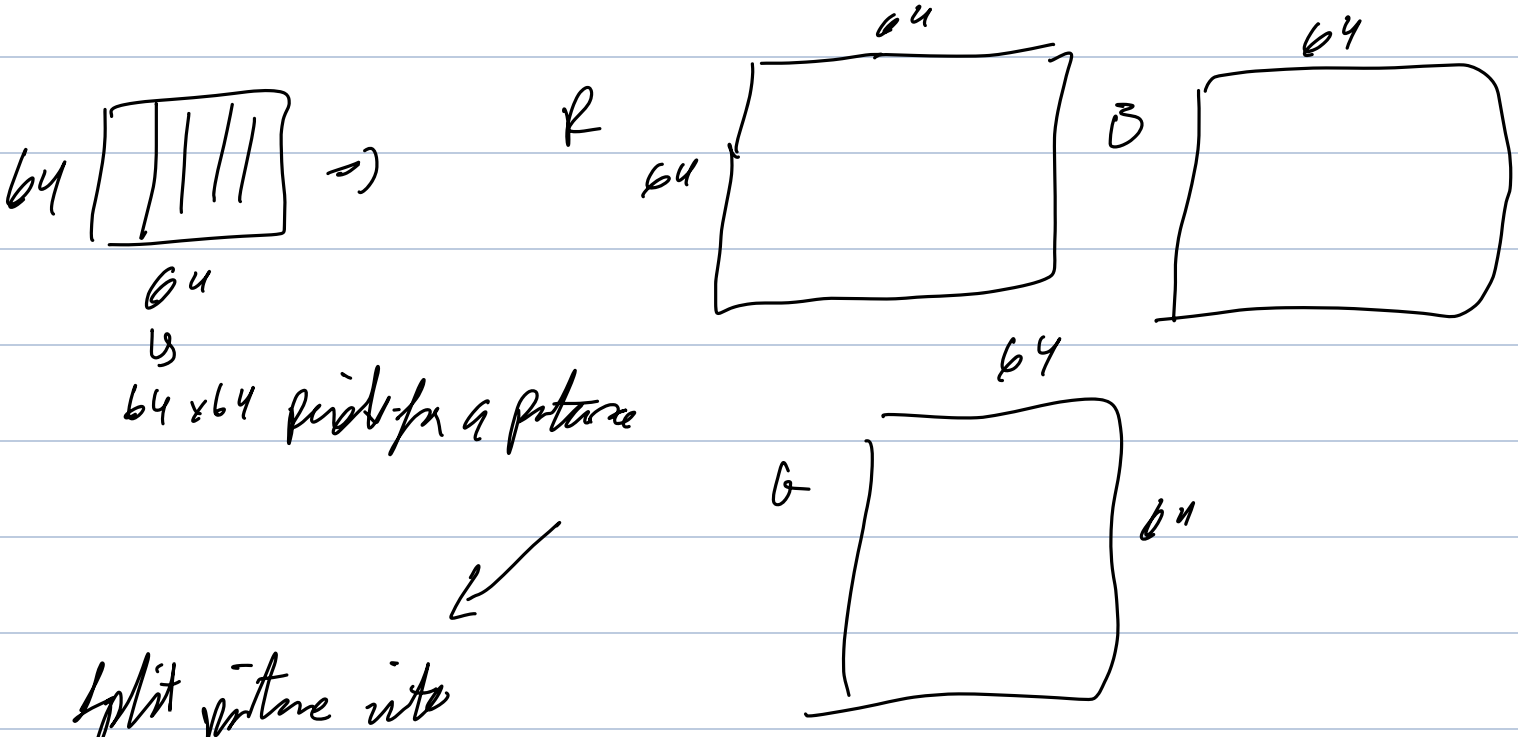
Logistic Regression as a Neural Network

Lec 1: Binary Classification

Logistic Regression for Binary Classification.

Pic of cat ⟶ $y = 1$ (cat) vs 0 (non cat)



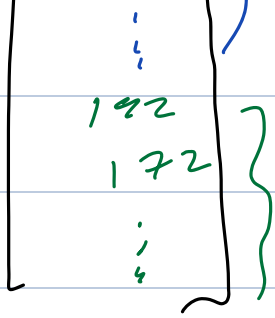
64 × 64 pixels for a picture

Split picture into the R, G, B matrices of the picture

Single Feature Vector ⟶ $x =$

$$\begin{bmatrix} 255 \\ 231 \\ \vdots \\ 255 \\ 192 \\ \vdots \end{bmatrix}$$

all red 64×64 + all blue 64×64



+

64 x 64

=>

x is a 12228 dim feature vector

$64 \times 64 \times 3 = 12288$

$n = n_x = 12288$ → dimension of this input vector / feature

$x \longrightarrow y \ (1 \text{ or } 0)$

using classifier

NOTATION

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

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

$m = m_{train}$

or Test Set m_{test} = # test example

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

MATRIX

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

$X.shape = (n_x, m)$ $\quad < n_x, m >$

↳ n_x by m dimensional matrix

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

↓ matrix

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

$Y.shape = (1, m)$ $\quad < 1, m >$

Ice 2 Logistic Regression

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

if x is a picture, $\hat{y}$ = what is chance that x is a cat.

↳ $0 \le \hat{y} \le 1$

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

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

for logistic reg. weights, bias

Output $\hat{y} = w^T x + b$ → Linear fcn of x

↳ BAD (This is linear regression)

↳ not good b/c we want $\hat{y} \in [0, 1]$

which is hard in linear regression.

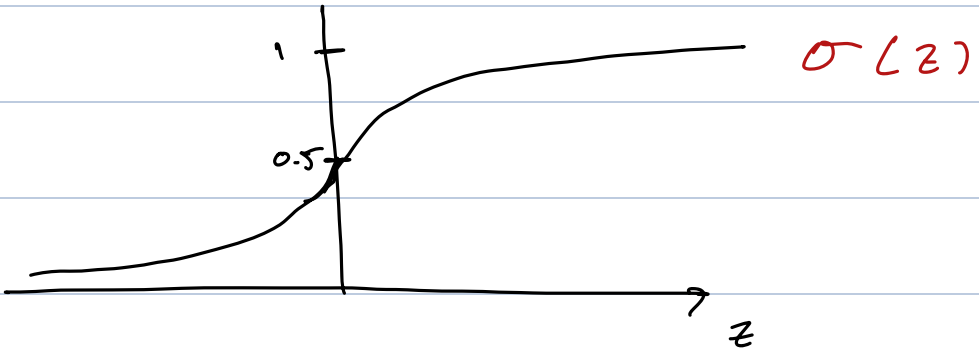
$w^T = 1 \times n_x$

$x = n_x \times 1$

$\hat{y} = 1 \times 1$ (1×1)

But can we Sigmoid fun → (ReLU is better?)

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



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

if z large → $\sigma(z) \simeq \frac{1}{1+0} = 1$.

if z large negative → $\sigma(z) \simeq \frac{1}{1+\infty} = 0$

Usually keep w and b separately

Sometimes you see: $x_0 = 1$, $x \in \mathbb{R}^{n_x+1}$

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

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \begin{matrix} \rightarrow b \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} w. \end{matrix}$$

To implement a neural network, easy to keep w and b separate.

Lec 3: Logistic Regression Cost Function

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

Given $\{(x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})\}$,

want $\hat{y}^{(i)} \approx y^{(i)}$

$x^{(i)} \rightarrow$ (data) - ith example

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

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

$\hat{y}^{(i)} = \sigma(z^{(i)})$

Error (Loss) Function:

Common: Square Error: $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$

Loss fcn, to measure how good our guess

but not good for this b/c we will get multiple optima

is.

while gradient descent.

For Logistic Regression we define a similar loss fn.

error fn for 1 example

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

want to be as small as possible.

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

$\therefore -\log \hat{y}$ to be as small as possible

$\Rightarrow \log \hat{y}$ to be as large as possible

$\Rightarrow \hat{y}$ to be as large as possible (1).

$\hat{y} \to$

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

$\therefore -\log(1-\hat{y})$ to be as small as possible

$\Rightarrow \log(1-\hat{y})$ to be large.

$\Rightarrow 1-\hat{y}$ as large as possible $\Rightarrow \hat{y}$ as small as possible

$\therefore \hat{y} \to 0$

Avg over the entire training set.

Cost Function → How well you're doing over the ENTIRE TRAINING SET

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

Prediction

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

↳ minimize the cost function by using w, b.