Logistic Regression Model

1) Preprocess data into h5 file.
   - Loop through all images
   - resize to 64x64. => convert to RGB.
   - assign label + (1 = cat, 0 = not cat).
   - Convert to array
   - train_test_split (   ) 80% training
     20% testing.

   h5 create file

2) Load data out of h5.

Spend more time doing this.

3) Shapes: train_x = (209, 64, 64, 3) → m = Pictures. 209

   (labels): train_y = (209, 1). + [ ] col vector, y or n of cat.

   test_x = (50 ...)

   test_y = (50, 1)

Then flatten into a matrix:

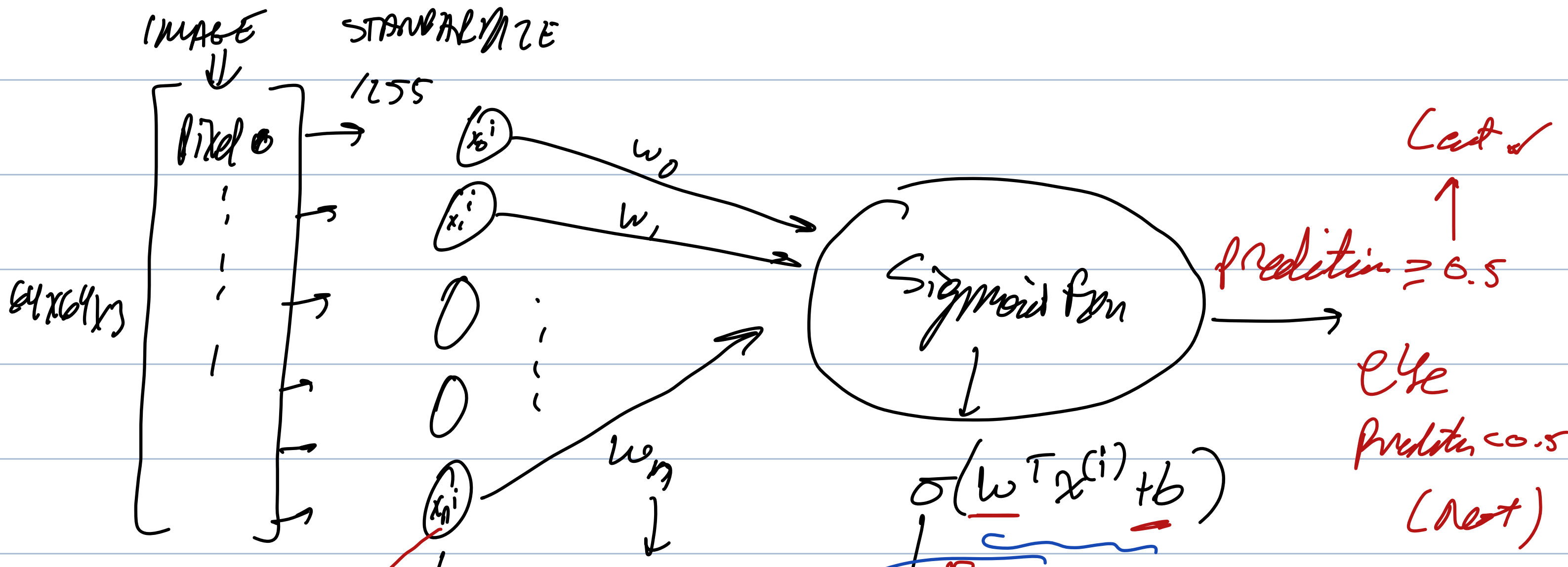
train_x => (209, 64, 64, 3)

$$\Rightarrow (64\times 64\times 3,\ 209)$$

$$n \text{ parameters} \left\{ \underbrace{\begin{bmatrix} | & | & | & & | \\ x_1 & x_2 & x_2 & \cdots & x_m \\ | & | & | & & | \end{bmatrix}}_{m \text{ Samples}} \right.$$

n parameters ↓ or inputs corresponding to each pixel for RGB.

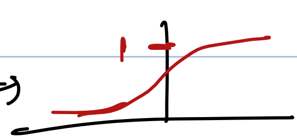
Neural Architecture



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

Learning is finding the w and b

$\sigma \Rightarrow$



$z^{(i)}$

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

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

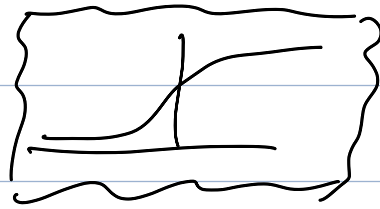
$$\underbrace{\mathcal{L}(a^{(i)}, y^{(i)})}_{\text{Loss Fxn}} = -y^{(i)} \log(a^{(i)}) - (1 - y^{(i)}) \log(1 - a^{(i)})$$

Ultimately this is a MLE problem

(MAximum LIKELIHOOD ESTIMATE)

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

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



What is the likelihood fxn?

Definition: $P(\text{data} \mid p) = ?$ Given some unknown probability p, we assume we know p, what is the probability of getting the data, if we assume p.

Assuming we have x, what is probability of y? $\hat{y}$

B/c we're using $\hat{y}$ to estimate y.

if $y = 1 \Rightarrow P(y \mid x) = \hat{y} = \sigma(w^T x + b)$

if $y = 0 \Rightarrow P(y \mid x) = 1 - \hat{y} = 1 - \sigma(w^T x + b)$

b/c the mapping of x to yhat $\in [0, 1]$, is for two interrelated reasons of our classification being 0 or 1, and that the $\hat{y}$ is the probability of a cat.

In logistic regression, assume $P(y=1 \mid x) = \hat{y} = \sigma(z)$

is a BERNOULLI DISTRIBUTION

SUCCESS or failure.

$\therefore P(y=0 \mid x) = 1 - \hat{y}$.

∴ The main reason is the probabilities are b/c often cases we could use a SOFTMAX Fcn instead of sigmoid.

Now we have the two, but what is

$P(Y=1 \mid x) = \hat{y}$. $\quad P(Y=0 \mid x) = 1 - \hat{y}$.

$P(Y=y \mid x) = \hat{y}^{y} \cdot (1-\hat{y})^{1-y}$

$\therefore$ if $y=1 \Rightarrow \hat{y}^{1} \cdot (1-\hat{y})^{1-1} = \hat{y}$

if $y=0 \Rightarrow \hat{y}^{0} \cdot (1-\hat{y})^{1} = 1-\hat{y}$

$\therefore P(Y=y \mid x) = \hat{y}^{y} (1-\hat{y})^{1-y}$. Likelihood fcn

We want to find the MLE for $\hat{y}$.

Take log.

not just x, → $\boxed{w \text{ and } b}$

$\log(P(Y=y \mid x)) = y \log \hat{y} + (1-y) \log(1-\hat{y})$

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

↓
error fun
or likelihood?

→ log probability of the observed data given parameters.

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

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

So we need to maximize MLE of $\log(L(w,b)) \Rightarrow$

$\log(P(Y=y|x,w,b)) = y\log(\underbrace{\sigma(w^T x+b)}_{\hat{y}}) + (1-y)\log(\underbrace{1-\sigma(w^T x+b)}_{1-\hat{y}})$

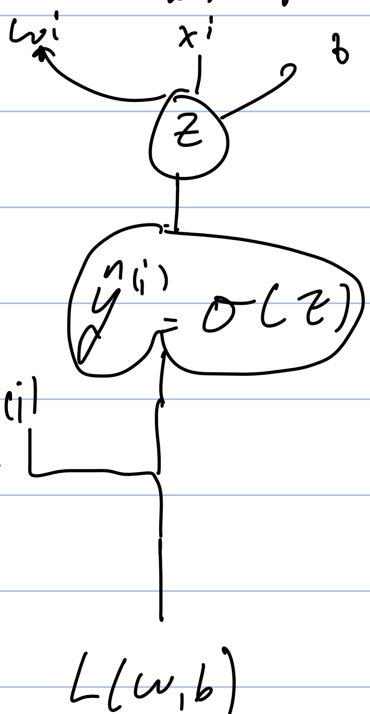
∴ TAKE PARTIALS TO OPTIMIZE

$\dfrac{\partial L(w,b)}{\partial w}$ and $\dfrac{\partial L(w,b)}{\partial b}$ → but there are n w's
$w^1 \ldots w^n$

So first, recall we have m examples:

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

Now consider the chain rule and then there are



$$\frac{\partial L(w,b)}{\partial w^{(i)}} = \frac{\partial L(w,b)}{\partial \hat{y}^{(i)}} \cdot \frac{\partial \hat{y}^{(i)}}{\partial z^{(i)}} \cdot \frac{\partial z^{(i)}}{w^{(i)}}$$

$$= \left(\sum_{i=1}^{m} \frac{y^{(i)}}{\hat{y}^{(i)}} + \frac{1-y^{i}}{1-\hat{y}^{(i)}}(-1)\right)\left(\sigma'(z^{(i)})\right)\left(x^{(i)}\right)$$

$$= \left(\sum_{i=1}^{m} \frac{y - \cancel{y\hat{y}} - \hat{y} + \cancel{y\hat{y}}}{\hat{y}(1-\hat{y})} \right) \left(\sigma'(z^{(i)})\right)\left(x^{(i)}\right)$$

$$= \left(\sum_{i=1}^{M} \frac{y - \hat{y}}{\hat{y}(1-\hat{y})} \right) \left(\sigma(z^{(i)})\left(1-\sigma(z^{(i)})\right) \right) \left(x^{(i)}\right)$$

$$= \left(\sum_{i=1}^{M} \frac{y^{i} - \hat{y}^{i}}{\cancel{\hat{y}^{i}(1-\hat{y}^{i})}} \right) \left(\cancel{\hat{y}^{(i)}(1-\hat{y}^{(i)})} \right) \left(x^{(i)}\right)$$

$$\boxed{\frac{\partial}{\partial w} = \left(\sum_{i=1}^{m} y^{(i)} - \hat{y}^{(i)} \right) x^{(i)}}$$

Similarly we can get $\frac{\partial}{\partial b} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial b} \to 1$

$$\Rightarrow \boxed{\frac{\partial}{\partial b} = \left(\sum_{i=1}^{M} y^{(i)} - \hat{y}^{(i)} \right)}$$

📌 **Step 4: Could We Just Use Gradient Ascent?**

Yes, we **could** use gradient ascent to maximize $\log L(w, b)$ directly. However:

- Most ML frameworks (like TensorFlow, PyTorch, Scikit-Learn) are **built around minimization**.
- Using a **minimization approach keeps it consistent with other models** (like linear regression, which minimizes squared error).
- It's easier to **interpret a "loss function"** than a "likelihood function."

📌 **Step 5: Summary**

Approach	Goal	Why We Do It
MLE (Maximize Likelihood)	Find parameters that maximize $\log L(w, b)$	Theoretical foundation for logistic regression
Gradient Descent (Minimization)	Find parameters that minimize the **negative log-likelihood** $J(w, b)$	Works better with standard ML optimization algorithms
Why Negative?	So we can **minimize instead of maximize**	Consistent with other ML models (e.g., squared error in linear regression)

Although even here we want to MAXIMIZE MLE

(Likelihood fun)

it makes more logical sense to want to minimize the loss error.

So Loss fun $= -\log(L(w,b))$

$$\Rightarrow \frac{\partial L}{\partial w} = \sum(\hat{y}^{(i)} - y^{(i)})x^{(i)}$$

$$\frac{\partial L}{\partial b} = \sum(\hat{y}^{(i)} - y^{(i)})$$

We want to carry the error fun into the loss function

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

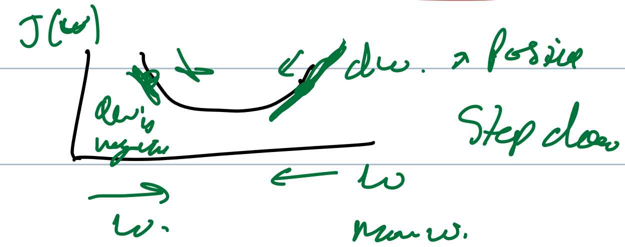
$$\frac{\partial J}{\partial w} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})x^{(i)}$$

$$\frac{\partial J}{\partial b} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})$$

Sigmoid fun is nonlinear, can't just set $=0$ and solve for w and b.

USE ITERATIVE OPTIMIZATION TECHNIQUES INSTEAD

GRADIENT DESCENT. Image



$$w = w - \alpha\frac{\partial J}{\partial w}.$$ TO MINIMIZE $J(w,b)$ and

learning rate.

$$b = b - \alpha \frac{\partial J}{\partial w}.$$

How big of a step to take.

maximizing likelihood.

Different things like newtons method to find but is expensive
(TANGENT, ETC. CALC 2 NOTES)

∴ TRAINED (GRADIENT DESCENT)
why it works b/c $\hat{y}$ is the derivative depends on w, and b,
So it'll update itself as it goes.