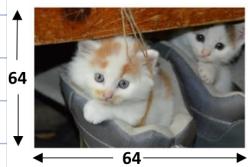


# Binary Classification



| Red | Green | Blue |
|-----|-------|------|
| 255 | 231   | 22   |
| 231 | 134   | 42   |
| 134 | 93    | 22   |
| 93  | 2     | 2    |
| 2   | 30    | 30   |
| 123 | 94    | 83   |
| 94  | 2     | 2    |
| 83  | 192   | 124  |
| 192 | 142   | 34   |
| 124 | 34    | 44   |
| 34  | 44    | 187  |
| 44  | 187   | 92   |
| 187 | 92    | 34   |
| 92  | 34    | 142  |
| 34  | 76    | 232  |
| 76  | 232   | 124  |
| 232 | 124   | 94   |
| 124 | 94    | 67   |
| 94  | 67    | 83   |
| 67  | 83    | 194  |
| 194 | 202   | 202  |

$$x = \begin{bmatrix} 255 \\ 231 \\ 42 \\ \vdots \\ 255 \\ 134 \\ 202 \\ \vdots \\ 255 \\ 134 \\ 93 \\ \vdots \end{bmatrix}$$

red

green

blue

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

$n_x$

the dimension of the input feature vector

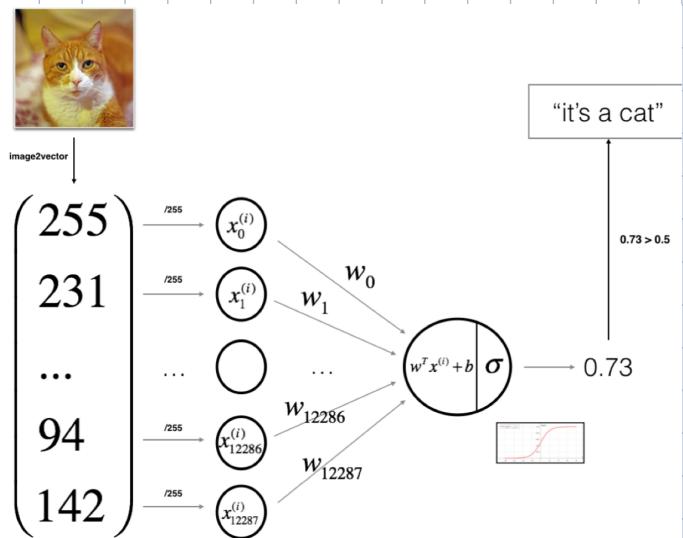
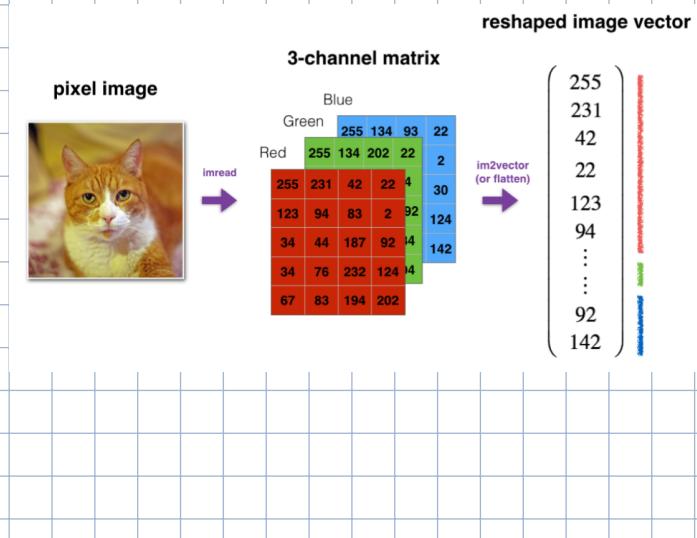
Notation:

A single training example:  $(x, y)$ ,  $x \in \mathbb{R}^{n_x}$ ,  $y \in \{0, 1\}$

There are  $m$  training samples:  $(x^{(1)}, y^{(1)})$ ,  $(x^{(2)}, y^{(2)})$ , ...,  $(x^{(m)}, y^{(m)})$

$$X = \begin{bmatrix} | & | & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & | \end{bmatrix} \in \mathbb{R}^{n_x \times m}$$

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



Logistic Regression: probability of the picture being a cat

Given  $x$ , we want  $\hat{y} = P(y=1|x)$ ,  $0 \leq \hat{y} \leq 1$

A picture of cat or non-cat      estimated output

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

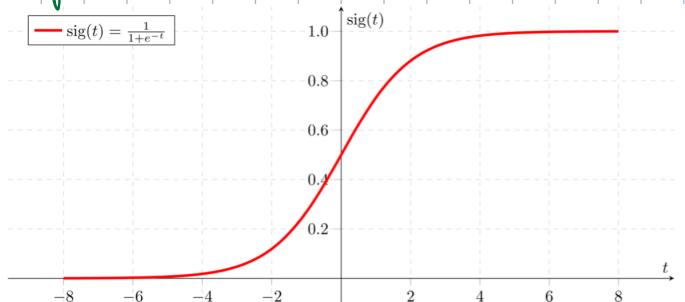
The parameters:  $w \in \mathbb{R}^{n_x}$ ,  $b \in \mathbb{R}$

Output  $\hat{y} = w^T x + b \rightarrow$  This is a bad algorithm. We need

$\hat{y}$  to be  $0 \leq \hat{y} \leq 1$ , but  $w^T x + b$

can be larger than 1 or negative.

This is why we need sigmoid function.



Apply sigmoid function  $\sigma$  to  $\hat{y}$

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

$\underbrace{w^T x + b}_z$

Thus,  $0 \leq \hat{y} \leq 1$

In logistic regression, we want to train  $w$  and  $b$

so that we have a good estimate of  $\hat{y}$ .

Another notation which combines  $w$  and  $b$  in  $\theta$ .

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

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

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \quad \left\{ \begin{array}{l} b \\ w \end{array} \right\}$$

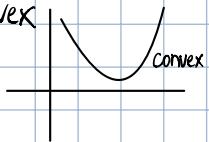
## Cost function in Logistic Regression

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$ , where  $\sigma(z) = \frac{1}{1+e^{-z}}$ , we use  $x^{(i)}$  to denote the i-th sample.

Given training examples  $\{(x^{(i)}, y^{(i)}), \dots, (x^{(m)}, y^{(m)})\}$ , we want  $\hat{y}^{(i)} \approx y^{(i)}$

Loss (error) function:  $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ , In logistic regression, people usually  
 (Only apply to single sample)

Thus, we need a different loss function in logistic regression which produces convex in optimization problem.



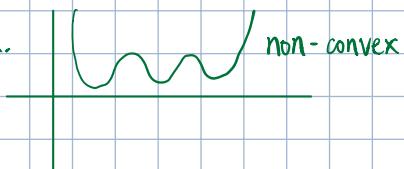
$$L(\hat{y}, y) = -[y \log \hat{y} + (1-y) \log(1-\hat{y})]$$

Intuition: We want  $L(\hat{y}, y)$  as small as possible.

If  $y=1$ ,  $L(\hat{y}, y) = -\log \hat{y}$  so  $\hat{y}$  as large as possible.

If  $y=0$ ,  $L(\hat{y}, y) = -\log(1-\hat{y})$  so  $\hat{y}$  as small as possible.

In gradient descent, it can't find the global optima.



Now, let define a Cost function which measures the loss function on the entire training set.

Cost function:  $J(w, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})$   
 (the cost for all samples)

In training, we want to find  $w$  and  $b$  which minimize the cost function  $J(w, b)$  in our Logistic Regression model.

Justification of  $L(\hat{y}, y)$ :  $\hat{y} = \sigma(z)$ , where  $\sigma(z) = 1/(1+e^{-z})$

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

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

$P(y|x)$  can be written as

$$\hat{y}^y (1-\hat{y})^{1-y}$$

Now, maximizing  $P(y|x)$  is the same as maximizing  $\log P(y|x)$ .

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

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

Now, cost on  $m$  samples.

$$P(\text{labels in training set}) = \prod_{i=1}^m P(y^{(i)}|X^{(i)})$$

$$\rightarrow \log P(\text{labels in training set}) = \log \prod_{i=1}^m P(y^{(i)}|X^{(i)})$$

$$\rightarrow = \sum_{i=1}^m \log P(y^{(i)}|X^{(i)})$$

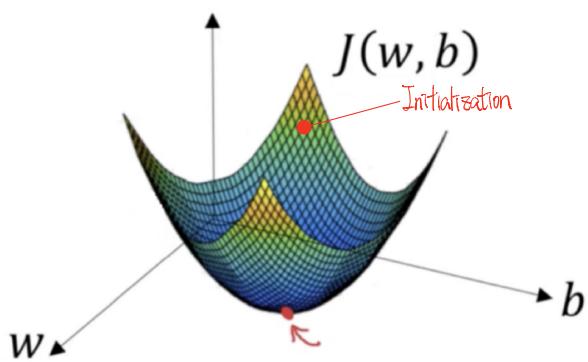
$$= \sum_{i=1}^m -L(\hat{y}^{(i)}, y^{(i)})$$

Maximum Likelihood Estimation

$$\frac{\partial}{\partial w} \log P(y^{(i)}|X^{(i)})$$

It turns out that logistic regression can be viewed as a very small neural network.

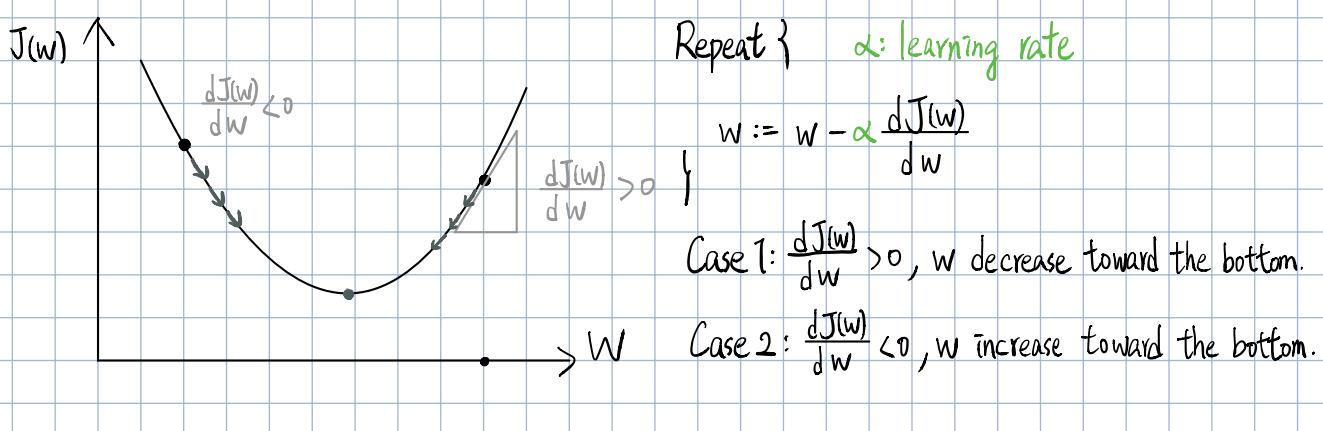
How to use Gradient Descent to train our logistic regression model.



For the case of  $J(w, b)$

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

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



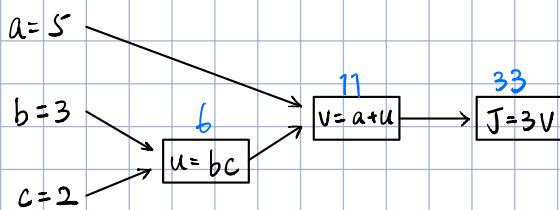
Computation Graph: Forward

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

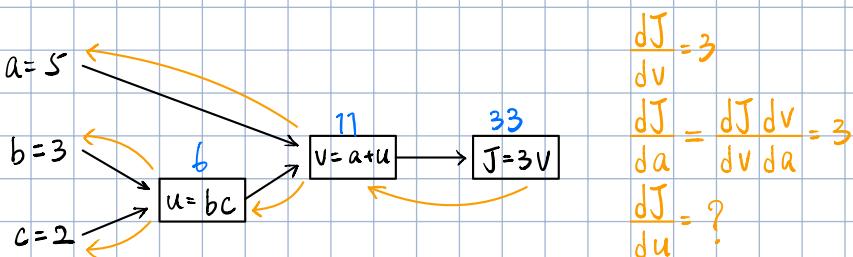
$$u = bc$$

$$v = a + u$$

$$J = 3v$$



## Derivatives with Computation Graph: Backward



We need to figure out  $\frac{dJ}{da}$ ,  $\frac{dJ}{db}$ ,  $\frac{dJ}{dc}$ . Hint: Use chain rule.

---

Logistic Regression Gradient Descent : On one sample.

$$z = w^T x + b$$

estimated output

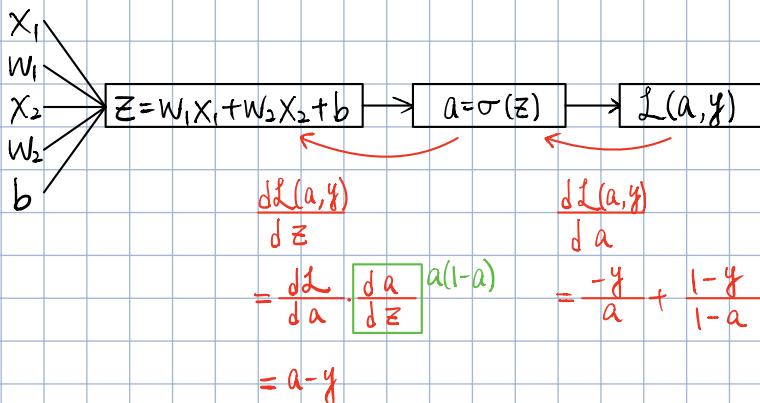
sigmoid fun.

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

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

E.g.: With two features  $x_1$  and  $x_2$

Thus, the computation graph is:



Now, we can calculate  $\frac{\partial L}{\partial w_1}$ ,  $\frac{\partial L}{\partial w_2}$ ,  $\frac{\partial L}{\partial b}$ .

$$\frac{\partial L}{\partial w_1} = "d w_1" = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w_1} = x_1 \cdot \frac{\partial L}{\partial z}$$

Once  $d w_1, d w_2, d b$  are known, we can update  $w_1, w_2$  and  $b$ .

$$\frac{\partial L}{\partial w_2} = "d w_2" = x_2 \frac{\partial L}{\partial z}$$

$$\begin{aligned} w_1 &:= w_1 - \alpha \frac{d w_1}{d w_1} \\ w_2 &:= w_2 - \alpha \frac{d w_2}{d w_2} \\ b &:= b - \alpha \frac{d b}{d b} \end{aligned}$$

$$\frac{\partial L}{\partial b} = "d b" = \boxed{\frac{\partial L}{\partial z}} \rightarrow a - y$$

Logistic Regression Gradient Descent : On  $m$  samples.

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

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

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

$$\text{Notice } dw_1 = \frac{\partial J}{\partial w_1}, dw_2 = \frac{\partial J}{\partial w_2}, db = \frac{\partial J}{\partial b}$$

for  $i = 1$  to  $m$ :

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

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

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

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

$$dw_1 += X_1^{(i)} dz^{(i)} \quad \left. \begin{array}{l} n=2 \text{ features} \\ \text{This could be a for loop} \end{array} \right.$$

$$dw_2 += X_2^{(i)} dz^{(i)} \quad \left. \begin{array}{l} \\ \text{if there are more features} \end{array} \right.$$

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

}

$$J /= m$$

$$dw_1 /= m$$

$$dw_2 /= m$$

$$db /= m$$

$$w_1 := w_1 - \alpha dw_1$$

$$w_2 := w_2 - \alpha dw_2$$

$$b := b - \alpha db$$

There are two for loops here.  
It's too slow. Don't even think about it. Use **Vectorization**.

Avoid explicit for-loop whenever possible.

$$z = w^T x + b$$

$$a = \sigma(z)$$

$$dz = a - y$$

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

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

$$w -= \alpha dw$$

$$b -= \alpha db$$