# Copyright Notice

deeplearning.ai

# Basics of Neural Network Programming

## Binary Classification

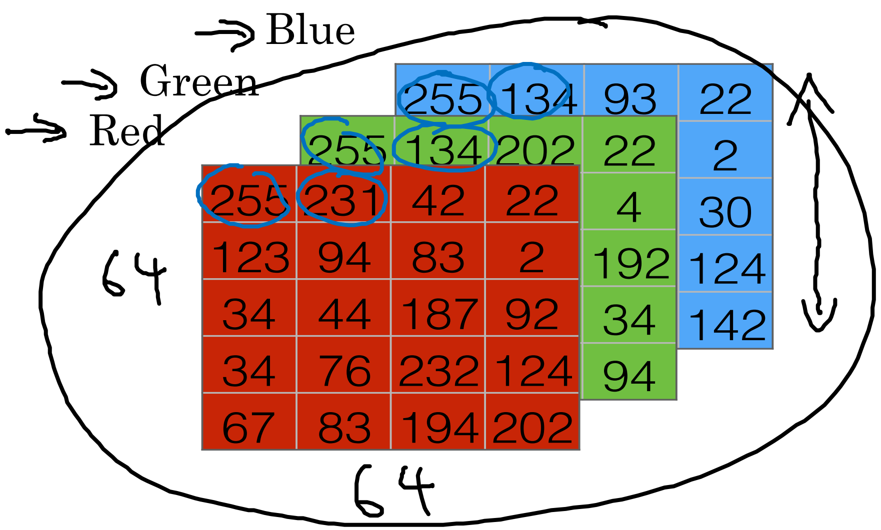
# Binary Classification

64

64

$\longrightarrow$ $\underline{1}$ (cat) vs $\underline{0}$ (non cat)

$y$




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

$64 \times 64 \times 3 = 12288$

$n = n_x = 12288$

$x \longrightarrow y$

Andrew Ng

# Notation

$(x, y)$ $\quad x \in \mathbb{R}^{n_x}, \; y \in \{0, 1\}$

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

$M = M_{train}$ $\qquad M_{test}$ = #test examples.

$$X = \begin{array}{c} \left[\begin{array}{cccc} | & | & & | \\ x^{(1)} & x^{(2)} & \ldots & x^{(m)} \\ | & | & & | \end{array}\right] \\ \xleftrightarrow{\quad m \quad} \end{array} \updownarrow n_x \qquad \cancel{\begin{bmatrix} - \; x^{(1)T} \; - \\ \vdots \\ - \; x^{(m)T} \; - \end{bmatrix}}$$

$X \in \mathbb{R}^{n_x \times m}$ $\qquad X.\text{shape} = (n_x, m)$

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

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

$Y.\text{shape} = (1, m)$

Andrew Ng

# Basics of Neural Network Programming

## Logistic Regression

deeplearning.ai

# Logistic Regression

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

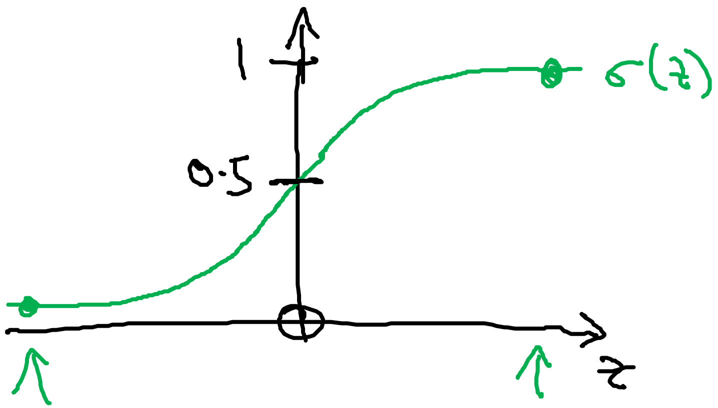
$0 \le \hat{y} \le 1$

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

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

where b is the intercept

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




Logistic regression utilises the sigmoid function to fulfill probability bounds

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

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

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

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

If $z$ large $\sigma(z) \approx \frac{1}{1+0} = 1$

If $z$ large negative number

$$\sigma(z) = \frac{1}{1+e^{-z}} \approx \frac{1}{1+\text{Bignum}} \approx 0$$

Andrew Ng

# Basics of Neural Network Programming

## Logistic Regression cost function

deeplearning.ai

# Logistic Regression cost function

i is the i-th observation in the training data

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

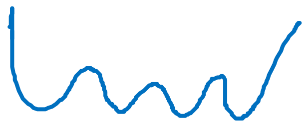
$z^{(i)} = w^T x^{(i)} + b$

Given $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

$x^{(i)}$
$y^{(i)}$
$z^{(i)}$

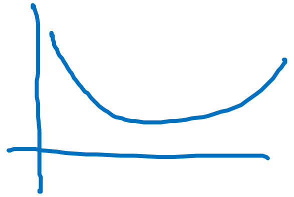
$i$-th example.

$\boxed{\text{Loss}}$ (error) function: $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$



instead of squared error, we use this (which we minimise as well)

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



If $y = 1$: $\mathcal{L}(\hat{y}, y) = -\log \hat{y} \leftarrow$ Want $\log \hat{y}$ large, want $\hat{y}$ large.

If $y = 0$: $\mathcal{L}(\hat{y}, y) = -\log(1-\hat{y}) \leftarrow$ Want $\log 1-\hat{y}$ large .... want $\hat{y}$ small

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

Cost function is the average of the Loss function across training observations m

Andrew Ng

# Basics of Neural Network Programming

## Gradient Descent

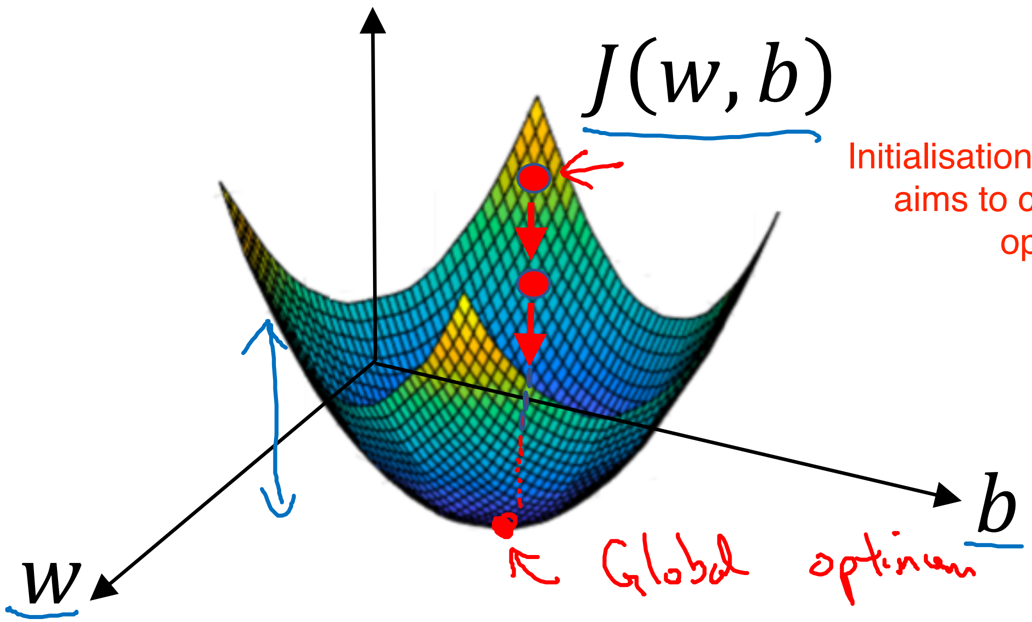
deeplearning.ai

# Gradient Descent

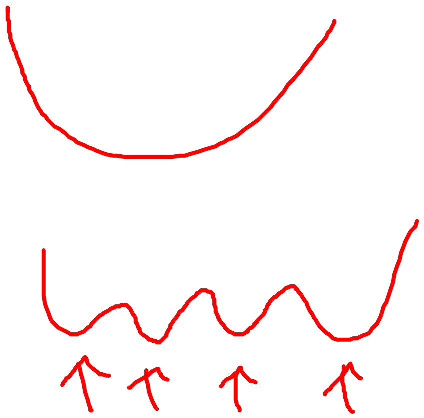
Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$

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

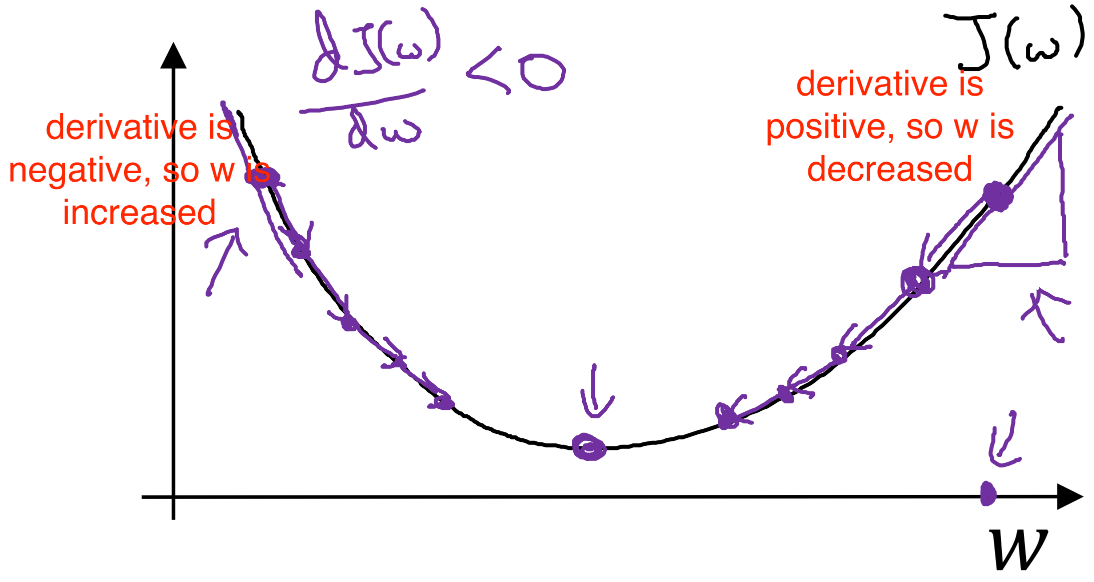
Want to find $w, b$ that minimize $J(w,b)$




Initialisation point, and then the algo aims to converge to the global optimum minima



# Gradient Descent




learning rate

Repeat {

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

}

"dw"

$w := w - \alpha \, dw$

learning rate x the derivative is the update to parameters W (and b)

$\frac{dJ(w)}{dw} = ?$

$J(w,b)$

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

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

$\partial$ "partial derivative"

$d$ J

Andrew Ng

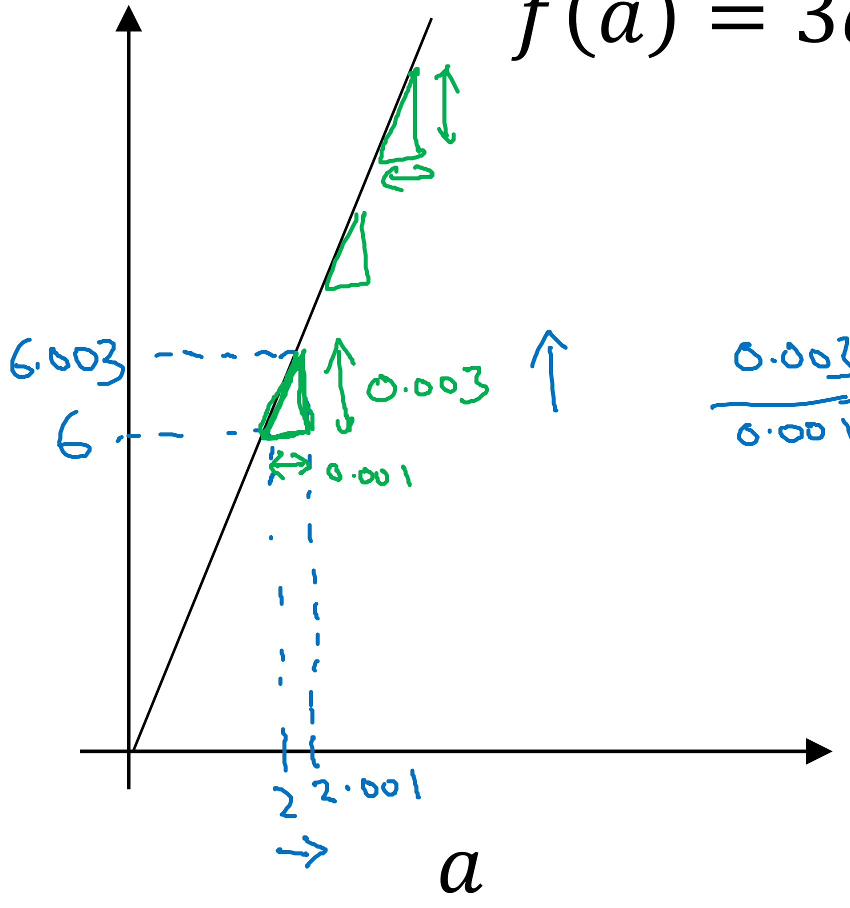
# Basics of Neural Network Programming

## Derivatives

deeplearning.ai

# Intuition about derivatives

$f(a) = 3a$




$\frac{0.003}{0.001}$ $\frac{\text{height}}{\text{width}}$

$a = 2$ $f(a) = 6$

$a = 2.\underline{001}$ $f(a) = 6.\underline{003}$

slope (derivative) of $f(a)$ at $a = 2$ is 3

$a = 5$ $f(a) = 15$

$a = 5.\underline{001}$ $f(a) = 15.\underline{003}$

slope at $a = 5$ is also 3

$\frac{d f(a)}{da} = 3 = \frac{d}{da} f(a)$

0.001

0.00000001

0.0000000001

deeplearning.ai

# Basics of Neural Network Programming

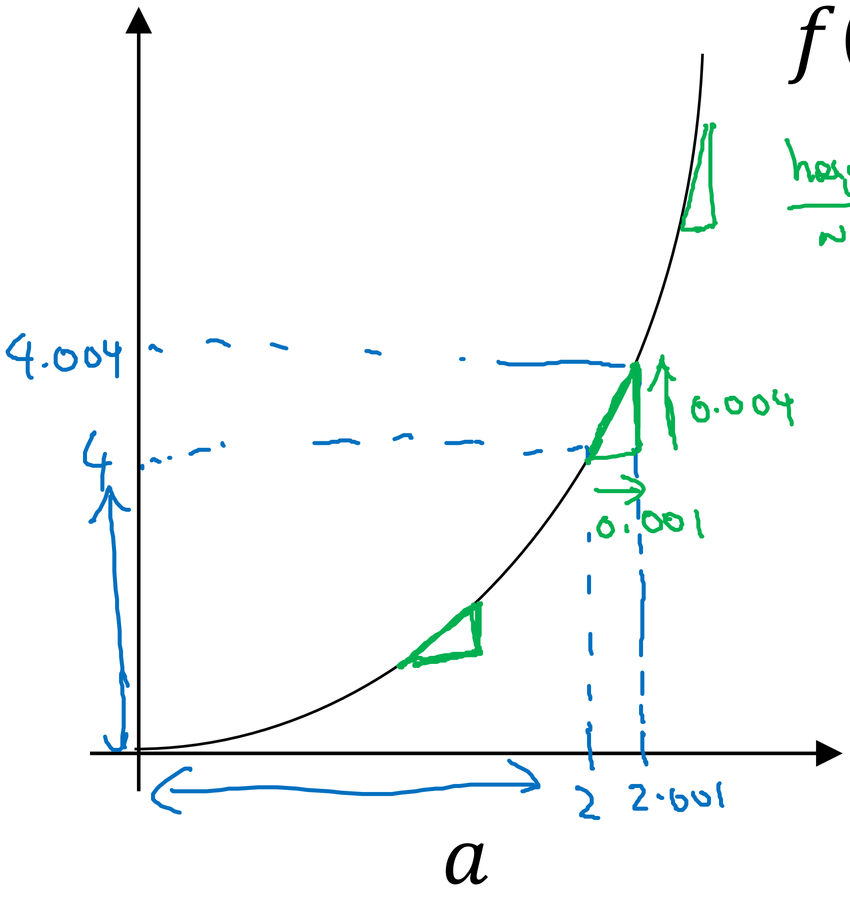
## More derivatives examples

# Intuition about derivatives

$0.001 \leftarrow$

$0.00000\ldots01 \leftarrow$

$f(a) = a^2$




$\frac{\text{height}}{\text{width}}$

$\frac{d}{da} a^2 = 2a$

$0.001$

$(2a) \times 0.001$

$a = 2 \qquad f(a) = 4$

$a = 2.001 \qquad f(a) \approx 4.004$

$(4.004\boxed{001})$

slope (derivative) of f(a) at a=2 is 4.

$\boxed{\frac{d}{da} f(a) = 4}$ when $\boxed{a = 2}$.

$a = 5 \qquad f(a) = 25$

$a = 5.001 \qquad f(a) \approx 25.010$

$\boxed{\frac{d}{da} f(a) = 10}$ when $\boxed{a = 5}$

$\frac{d}{da} f(a) = \frac{d}{da} a^2 = \boxed{2a}$

Andrew Ng

# More derivative examples

$f(a) = a^2$ $\qquad \frac{d}{da} f(a) = \underbrace{2a}_{4}$ $\qquad a = 2 \quad f(a) = 4$

$a = 2.001 \quad f(a) \approx 4.004$

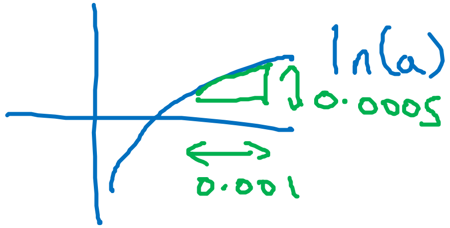
$f(a) = a^3$ $\qquad \frac{d}{da} f(a) = \underbrace{3a^2}_{3 \times 2^2 = 12}$ $\qquad a = 2 \quad f(a) = 8$

$a = \underline{2.001} \quad f(a) \approx 8.\underline{012}$

$f(a) = \log_e(a)$

$\ln(a)$

$\frac{d}{da} f(a) = \underline{\frac{1}{a}}$




$\frac{d}{da} f(a) = \boxed{\frac{1}{2}}$

$a = 2 \quad f(a) \approx 0.69315$

$a = 2.\underline{001} \quad \underline{f(a)} \approx 0.69365$

$0.0005 \quad \underline{0.0005}$

Andrew Ng

# Basics of Neural Network Programming

## Computation Graph

deeplearning.ai

# Computation Graph

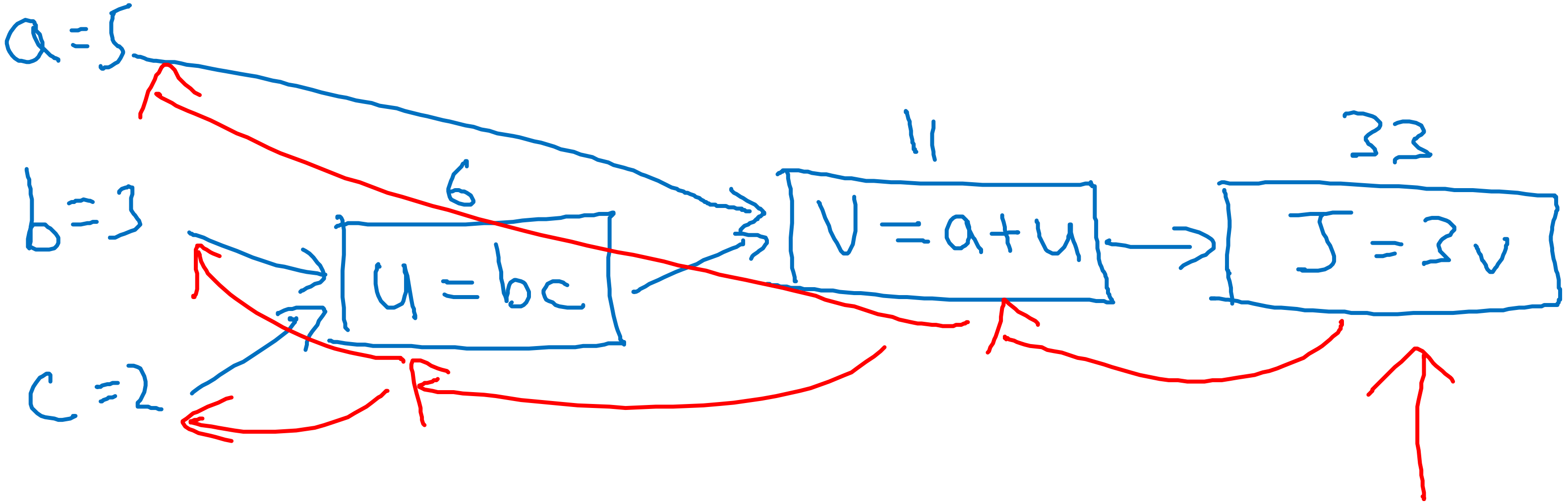
$$J(a,b,c) = \underbrace{3(\underbrace{a + \underbrace{bc}_{u}}_{v})}_{J} = 3(5 + 3\times 2) = 33$$

$u = bc$

$v = a + u$

$J = 3v$

left to right pass to compute outputs (J), right to left to compute the derivatives



deeplearning.ai

# Basics of Neural Network Programming

# Derivatives with a Computation Graph

# Computing derivatives

Back propogation to identify derivatives, by breaking down the impact of the net change of a value and how it affects the output

$a = 5$ $\quad \frac{dJ}{da}$ "da" = 3

$b = 3$

$c = 2$

6: $u=bc$

11: $v = a + u$

33: $J = 3v$

$\frac{dJ}{dv}$ "dv" = 3

$J = 3v$
$v = 11 \rightarrow 11.001$
$J = 33 \rightarrow 33.003$

$f(a) = 3a$

$\frac{df(a)}{da} = \frac{df}{da} = 3$

$J = 3v$

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

$\boxed{\frac{dJ}{dv} = ? = 3}$

$a \rightarrow v \rightarrow J$

Chain Rule

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

$3 \times 1$

$\boxed{\frac{dv}{da} = 1}$

$a = 5 \rightarrow 5.001$
$\rightarrow v = 11 \rightarrow 11.001$
$J = 33 \rightarrow 33.003$

$\boxed{\frac{d\,FinalOutputVar}{d\,var}}$

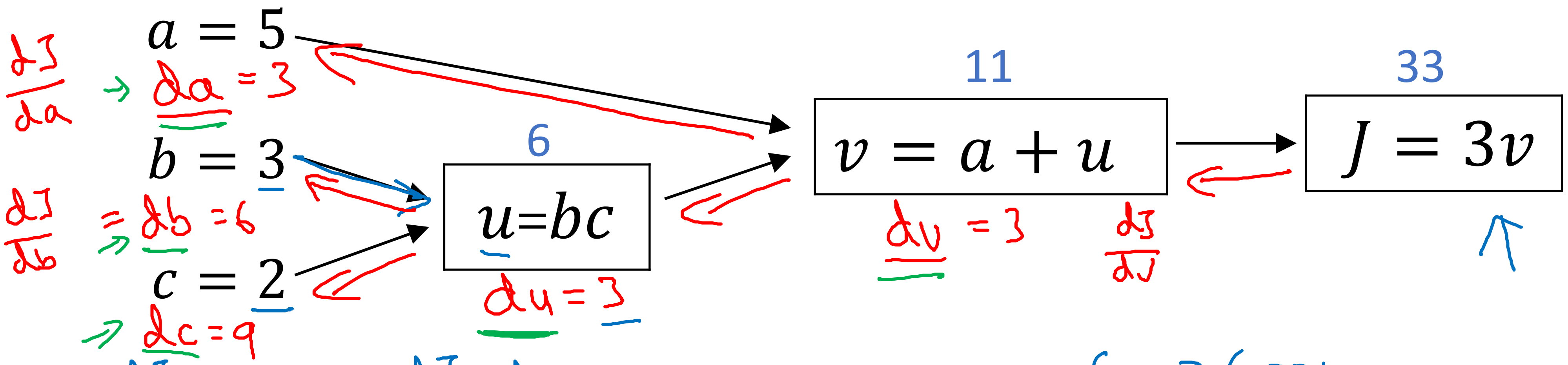
$dJdvar$

"dvar"

We tend to take the derivative of the Final Output anyway… its the with respect to what that changes

Andrew Ng

# Computing derivatives




$$\frac{dJ}{du} = 3 = \underbrace{\frac{dJ}{dv}}_{3} \cdot \underbrace{\frac{dv}{du}}_{1}$$

$$\frac{dJ}{db} = \underbrace{\boxed{\frac{dJ}{du}}}_{\rightarrow 3} \cdot \underbrace{\frac{du}{db}}_{=2} = 6$$

$$\frac{dJ}{da} = \boxed{\frac{dJ}{du}} \cdot \frac{du}{da} = 9$$
$$3 \times 3$$

$u = 6 \rightarrow 6.001$
$v = 11 \rightarrow 11.001$
$J = 33 \rightarrow 33.003$

$b = 3 \rightarrow 3.001$
$u = b \cdot c = 6 \rightarrow 6.002$ $\quad c = 2$
$J = 33.006$ $\quad .006$

$v = 11.002$
$J = 3v$

Andrew Ng

deeplearning.ai

# Basics of Neural Network Programming

## Logistic Regression Gradient descent

# Logistic regression recap

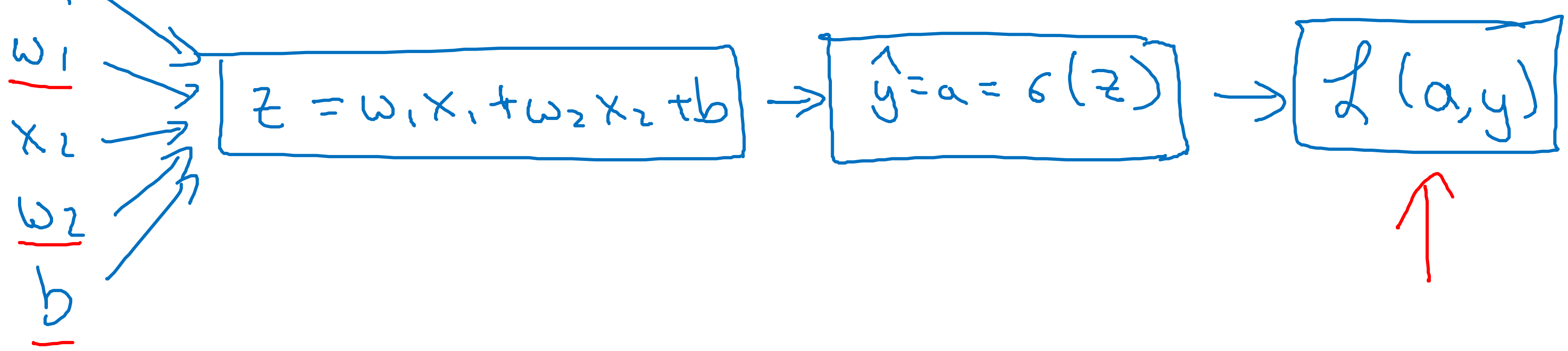
$$\rightarrow z = w^T x + b$$

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

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

Forward propagation to compute the loss function, and backwards propagation to compute the derivatives
(params w and b that minimises the loss function)

2 feature example



# Logistic regression derivatives

$x_1$, $w_1$, $x_2$, $w_2$, $b$

$$\boxed{z = w_1 x_1 + w_2 x_2 + b} \rightarrow \boxed{a = \sigma(z)} \rightarrow \boxed{\mathcal{L}(a, y)}$$

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

$$= a - y$$

$$= \frac{d\mathcal{L}}{da} \cdot \frac{da}{dz}$$

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

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

$$a(1-a)$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \text{"}dw_1\text{"} = x_1 \cdot dz. \qquad dw_2 = x_2 \cdot dz. \qquad db = dz.$$

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

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

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

The final goal of back propagation - calculate how much we need to change the params w and b.
The params then get updated by this calculated change (derivatives)N

Andrew Ng

deeplearning.ai

# Basics of Neural Network Programming

---

## Gradient descent on $m$ examples

# Logistic regression on $m$ examples

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

$$(x^{(i)}, y^{(i)})$$

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

$$\underline{dw_1^{(i)}}, \underline{dw_2^{(i)}}, \underline{db^{(i)}}$$

Now we expand the calculation from 1 example to across the entire training set (Cost function)

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

# Logistic regression on $m$ examples

$J = 0;\ dw_1 = 0;\ dw_2 = 0;\ db = 0$

For $i = 1$ to $m$

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

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

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

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

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

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

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

$n = 2$

$dw_3$
$\vdots$
$dw_n$

$J /= m$

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

$$dw_1 = \frac{\partial J}{\partial w_1}$$

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

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

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

Vectorization

instead of For loops, for more efficiency and speed

Andrew Ng

# Basics of Neural Network Programming

## Vectorization

deeplearning.ai

Code efficiency to prevent for-loops

# What is vectorization?

$z = \underbrace{w^T x} + b$

$w = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$ $x = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$ $w \in \mathbb{R}^{n_x}$ $x \in \mathbb{R}^{n_x}$

Non-vectorized:

```
z = 0
for i in range(n-x):
    z += w[i] * x[i]
z += b
```

Vectorized

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

→ GPU
→ CPU

SIMD - single instruction multiple data.

deeplearning.ai

# Basics of Neural Network Programming

## More vectorization examples

# Neural network programming guideline

Whenever possible, avoid explicit for-loops.

$u = Av$

$u_i = \sum_i \sum_j A_{ij} v_j$

```
u = np.zeros((n,1))
for i ...
    for j ...
        u[i] += A[i][j]*v[j]
```

```
u = np.dot(A,v)
```

Andrew Ng

# Vectors and matrix valued functions

Say you need to apply the exponential operation on every element of a matrix/vector.

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \rightarrow u = \begin{bmatrix} e^{v_1} \\ e^{v_2} \\ \vdots \\ e^{v_n} \end{bmatrix}$$

```
u = np.zeros((n,1))
for i in range(n):
    u[i]=math.exp(v[i])
```

```
import numpy as np
u = np.exp(v)

np.log(v)
np.abs(v)
np.maximum(v, 0)
v**2          1/v
```

# Logistic regression derivatives

$J = 0$, $\boxed{\cancel{dw1 = 0, dw2 = 0}}$, $db = 0$ $\quad dw = np.zeros((n\_x, 1))$

for i = 1 to n:

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

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

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

$$dz^{(i)} = a^{(i)}(1 - a^{(i)})$$

for $j = 1 \ldots n_x$

$dw_j \mathrel{+}= \ldots$

$$\boxed{\cancel{\begin{aligned} dw_1 &\mathrel{+}= x_1^{(i)} dz^{(i)} \\ dw_2 &\mathrel{+}= x_2^{(i)} dz^{(i)} \end{aligned}}}$$

$n_x = 2$ $\qquad dw \mathrel{+}= x^{(i)} dz^{(i)}$

eliminate the second for loop that loops through the number of features / params

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

$J = J/m$, $\boxed{\cancel{dw_1 = dw_1/m, \; dw_2 = dw_2/m}}$, $db = db/m$

$dw /= m.$

Andrew Ng

deeplearning.ai

# Basics of Neural Network Programming

## Vectorizing Logistic Regression

# Vectorizing Logistic Regression

$$z^{(1)} = w^T x^{(1)} + b \qquad z^{(2)} = w^T x^{(2)} + b \qquad z^{(3)} = w^T x^{(3)} + b$$

$$a^{(1)} = \sigma(z^{(1)}) \qquad a^{(2)} = \sigma(z^{(2)}) \qquad a^{(3)} = \sigma(z^{(3)})$$

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

$(n_x, m)$

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

$$[\text{———}]\ w^T \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

$$Z = [z^{(1)}\ z^{(2)} \cdots z^{(m)}] = \underbrace{w^T X + \underbrace{[b\ b \cdots b]}_{1 \times m}} = [\underbrace{w^T x^{(1)} + b \quad w^T x^{(2)} + b \quad \cdots \quad w^T x^{(m)} + b}_{1 \times m}]$$

$$\rightarrow Z = np.dot(w.T, X) + b$$

$\uparrow$ (1,1) $\mathbb{R}$

"Broadcasting"

$$A = [a^{(1)}\ a^{(2)} \cdots a^{(m)}] = \sigma(Z)$$

Andrew Ng

deeplearning.ai

# Basics of Neural Network Programming

---

## Vectorizing Logistic Regression's Gradient Computation

# Vectorizing Logistic Regression

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

$$\boxed{dZ = [dz^{(1)} \; dz^{(2)} \; \ldots \; dz^{(m)}]} \leftarrow$$

$1 \times m$

$A = [a^{(1)} \; \ldots \; a^{(m)}]$, $\quad Y = [y^{(1)} \; \ldots \; y^{(m)}]$

$\rightarrow dZ = A - Y = [a^{(1)} - y^{(1)} \quad a^{(2)} - y^{(2)} \quad \ldots]$

$\rightarrow dw = 0$

$dw \mathrel{+}= x^{(1)} dz^{(1)}$

$dw \mathrel{+}= x^{(2)} dz^{(2)}$

$\vdots$

$dw \mathrel{/}= m$

$\cancel{dw_1}$

$\cancel{dw_2}$

$\vdots$

$db = 0$

$db \mathrel{+}= dz^{(1)}$

$db \mathrel{+}= dz^{(2)}$

$\vdots$

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

$db \mathrel{/}= m.$

$$\boxed{db = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)} = \frac{1}{m} \, np.sum(dZ)}$$

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

$$= \frac{1}{m} \begin{bmatrix} | & & | \\ x^{(1)} & \ldots & x^{(m)} \\ | & & | \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$$

$$= \frac{1}{m} \left[ x^{(1)} dz^{(1)} + \ldots + x^{(m)} dz^{(m)} \right]$$

$n \times 1$

# Implementing Logistic Regression

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

$\boxed{\text{for } i = 1 \text{ to } m:}$

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

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

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

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

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

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

$dw \mathrel{+}= x^{(i)} * dz^{(i)}$

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

$J = J/m,\ dw_1 = dw_1/m,\ dw_2 = dw_2/m$

$db = db/m$

for iter in range(1000):

$Z = w^T X + b$

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

$A = \sigma(Z)$

$dZ = A - Y$

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

$db = \frac{1}{m} np.sum(dZ)$

$w := w - \alpha dw$

$b := b - \alpha db$

Andrew Ng

deeplearning.ai

# Basics of Neural Network Programming

## Broadcasting in Python

# Broadcasting example

Calories from Carbs, Proteins, Fats in 100g of different foods:

$$
\begin{array}{r c}
 & \begin{array}{cccc} \text{Apples} & \text{Beef} & \text{Eggs} & \text{Potatoes} \end{array} \\
\begin{array}{r} \text{Carb} \\ \text{Protein} \\ \text{Fat} \end{array} &
\begin{bmatrix} 56.0 & 0.0 & 4.4 & 68.0 \\ 1.2 & 104.0 & 52.0 & 8.0 \\ 1.8 & 135.0 & 99.0 & 0.9 \end{bmatrix}
\end{array}
= \underset{(3,4)}{A}
$$

59 cal

$\frac{56}{59} \approx 94.9\%$

0

1

Calculate % of calories from Carb, Protein, Fat. Can you do this without explicit for-loop?

```
cal = A.sum(axis = 0)
percentage = 100*A/(cal.reshape(1,4))
```

(3,4) / (1,4)

# Broadcasting example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100}$$

$$\underset{(m,n)\ \ (2,3)}{\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}} + \begin{bmatrix}100 & 200 & 300\\100 & 200 & 300\end{bmatrix}$$

$(1,n) \leadsto (m,n)$ $(2,3)$

$$\underset{(m,n)}{\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}} + \begin{bmatrix}100 & 100 & 100\\200 & 200 & 200\end{bmatrix} =$$

$(m,1) \to (m,n)$

# General Principle

$(m, n)$ matrix $\quad + \; - \; * \; / \quad (1, n) \rightsquigarrow (m, n)$

$(m, 1) \rightsquigarrow (m, n)$

$(m, 1) \quad + \quad \mathbb{R}$

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

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

Matlab/Octave: bsxfun

# Basics of Neural Network Programming

deeplearning.ai

## Explanation of logistic regression cost function (Optional)

# Logistic regression cost function

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

Interpret $\hat{y} = P(y=1 \mid x)$

If $y=1$: $P(y \mid x) = \hat{y}$

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

Andrew Ng

# Logistic regression cost function

$$\boxed{\text{If} \quad y = 1: \quad p(y|x) = \hat{y}}$$

$$\boxed{\text{If} \quad y = 0: \quad p(y|x) = 1 - \hat{y}}$$

$p(y|x)$

$$\boxed{p(y|x) = \underbrace{\hat{y}^{y}}\ \underbrace{(1-\hat{y})^{(1-y)}}}$$

If $y=1$: $\underline{p(y|x) = \hat{y}}\ \underbrace{(1-\hat{y})^{0}}_{=1}$

$\underline{\text{If } y=0}$: $\underline{p(y|x)} = \underbrace{\hat{y}^{0}}_{=1}\ (1-\hat{y})^{(1-y)} = 1 \times (1-\hat{y}) = \underline{1-\hat{y}}$

$$\uparrow \log \underline{p(y|x)} = \log \hat{y}^{y}(1-\hat{y})^{(1-y)} = y\log\hat{y} + (1-y)\log(1-\hat{y})$$

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

# Cost on $m$ examples

$$\log p(\text{labels in training set}) = \log \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)})$$

$$\log p(\cdots\cdots) = \sum_{i=1}^{m} \underbrace{\log p(y^{(i)} \mid x^{(i)})}_{-\mathcal{L}(\hat{y}^{(i)}, y^{(i)})}$$

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

Maximum likelihood estimation

Cost: (minimize)

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

Andrew Ng