



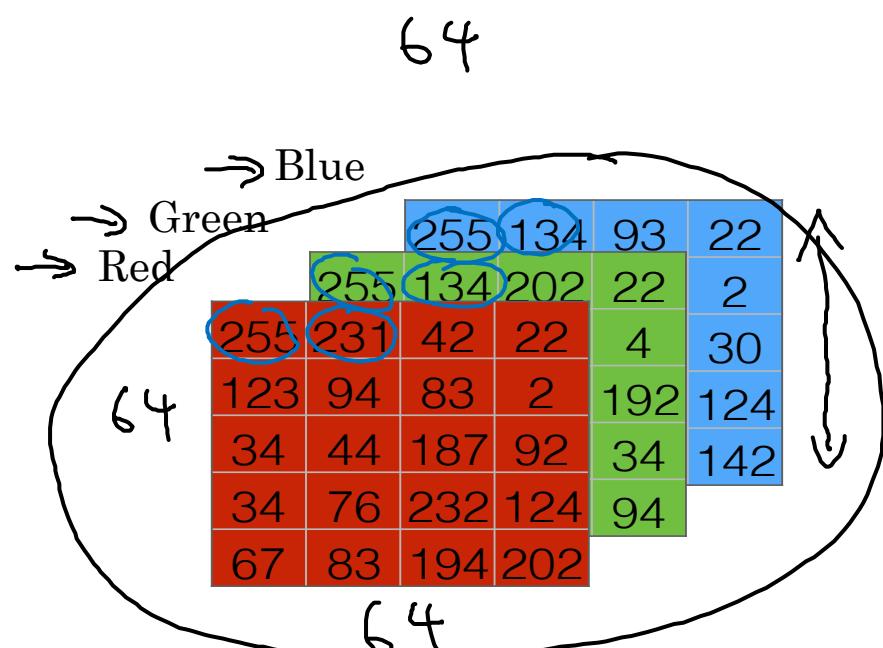
deeplearning.ai

Basics of Neural Network Programming

Binary Classification

Binary Classification

64 →  → 1 (cat) vs 0 (non cat)



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

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

$$n = n_x = 12288$$

$$X \rightarrow y$$

Notation

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

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

$M = M_{\text{train}}$

$M_{\text{test}} = \# \text{test examples.}$

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

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

$X.\text{shape} = (n_x, m)$

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

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

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



deeplearning.ai

Basics of Neural Network Programming

Logistic Regression

Logistic Regression

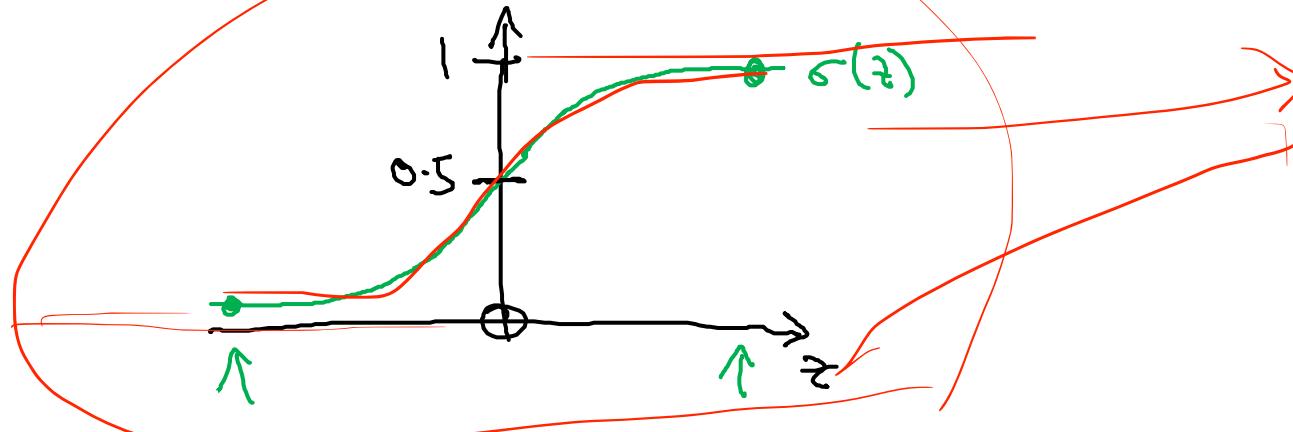
Given x , want

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

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

Output

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



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

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

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

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

$$\hat{y} = \sigma(w^T x) = \sigma(w_0 x_0 + w_1 x_1 + \dots + w_{n_x} x_{n_x})$$

$w = [w_0, w_1, w_2, \dots, w_{n_x}]^T$

$$\sigma(z) =$$

$$\frac{1}{1+e^{-z}}$$

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

If z large negative number

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



deeplearning.ai

Basics of Neural Network Programming

Logistic Regression cost function

Logistic Regression cost function

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

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

Loss (error) function:

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

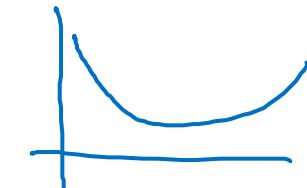
$$L(\hat{y}, y) = - (y \log \hat{y} + (1-y) \log (1-\hat{y}))$$

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

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

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

i -th example.





deeplearning.ai

Basics of Neural Network Programming

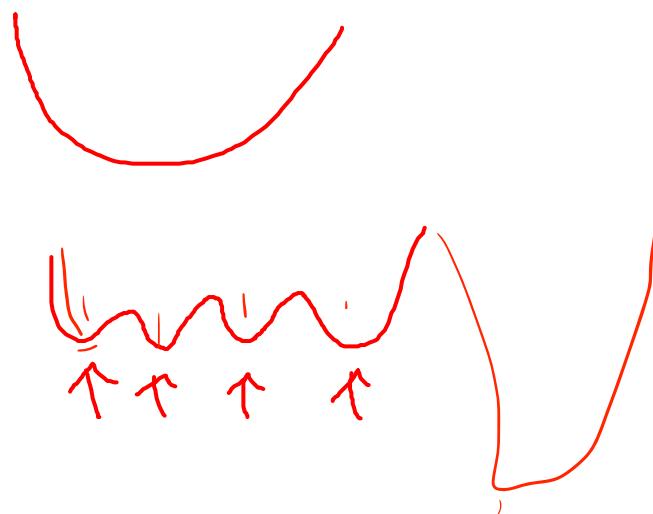
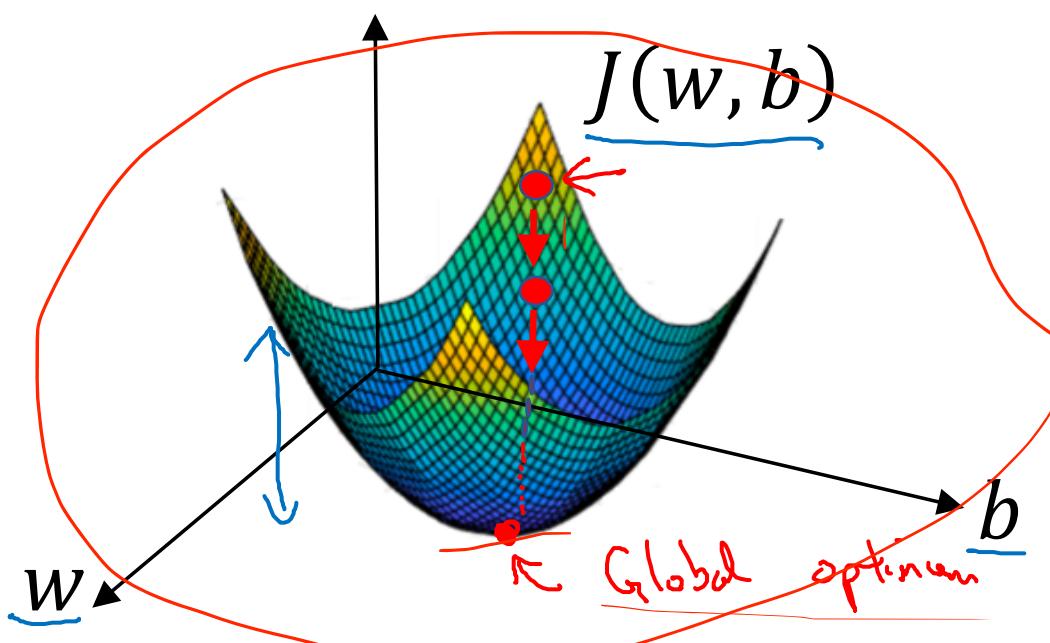
Gradient Descent

Gradient Descent

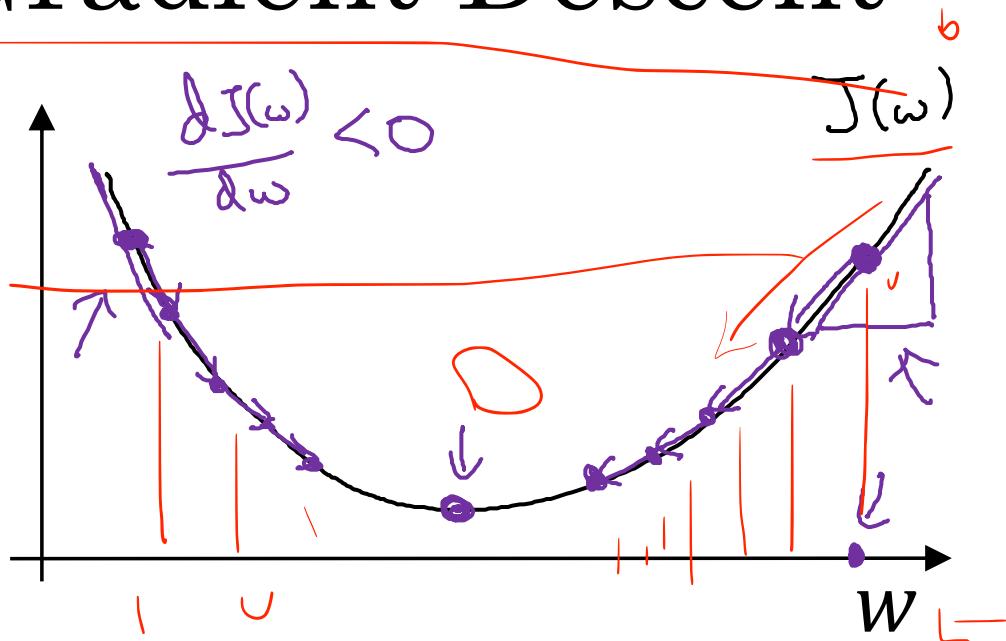
Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$ 

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

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



Gradient Descent



Repeat {

$\omega := \omega - \frac{\alpha}{\text{learning rate}} \frac{\partial J(\omega)}{\partial \omega}$

$\omega := \omega - \alpha \frac{\partial J(\omega)}{\partial \omega}$

$\frac{\partial J(\omega)}{\partial \omega} = ?$

$$\begin{aligned} J(w, b) & \quad \text{---} \\ w := w - \alpha \frac{d J(w, b)}{d w} & \quad \text{---} \\ b := b - \alpha \frac{d J(w, b)}{d b} & \quad \text{---} \end{aligned}$$

Diagram illustrating the backpropagation of gradients for the cost function $J(w, b)$. The diagram shows the flow of gradients from the output layer to the input layer. The top row shows the update rule for w with a red box around the gradient term $\frac{d J(w, b)}{d w}$. The bottom row shows the update rule for b with a red box around the gradient term $\frac{d J(w, b)}{d b}$. Red arrows indicate the flow of gradients from the output layer to the input layer. The text "Partial derivative" is written next to the diagram, with a blue circle containing a partial derivative symbol ($\frac{\partial}{\partial}$) and a blue circle containing a derivative symbol (d) pointing to the red boxes. The text "J" is written next to the bottom row of equations.

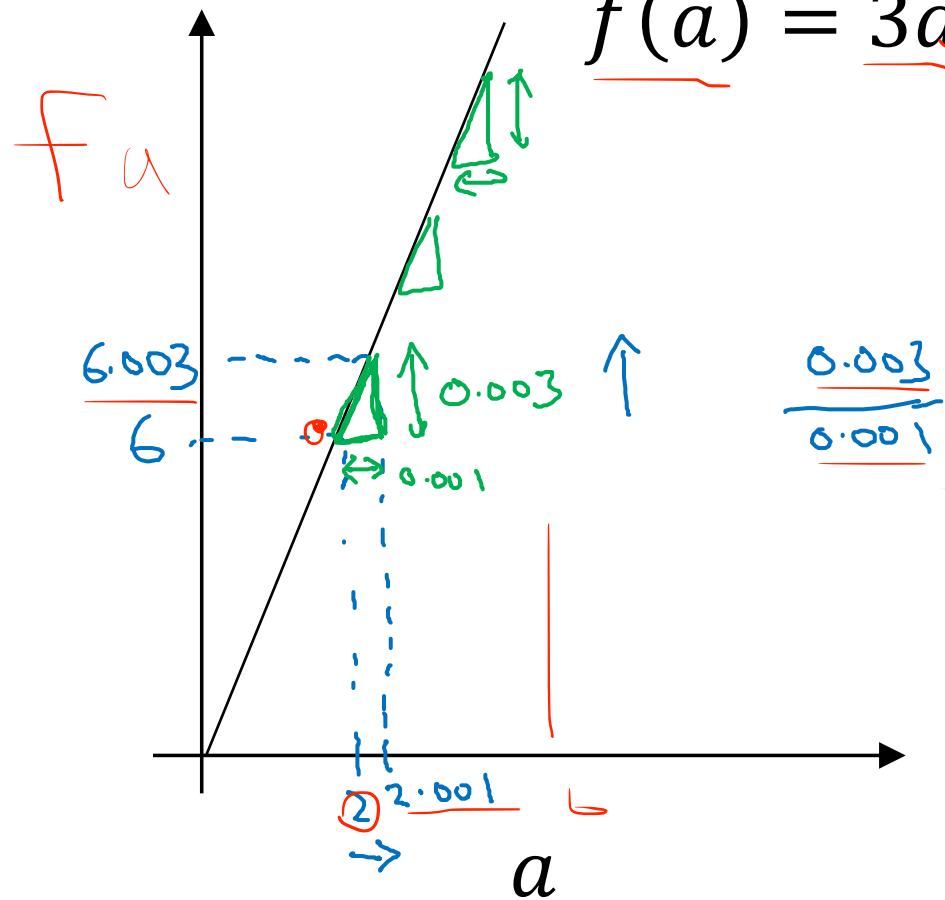


deeplearning.ai

Basics of Neural Network Programming

Derivatives

Intuition about derivatives



$$f(a) = 3a$$

$$\rightarrow a = 2$$

$$f(a) = 6$$

$$a = 2.001$$

$$f(a) = 6.003$$

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

$$\rightarrow a = 5$$

$$f(a) = 15$$

$$a = 5.001$$

$$f(a) = 15.003$$

slope at $a=5$ is also 3

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

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

0.001
0.00000001
0.0000000001

slope (derivative) of $f(a)$

at $a=2$ is 3

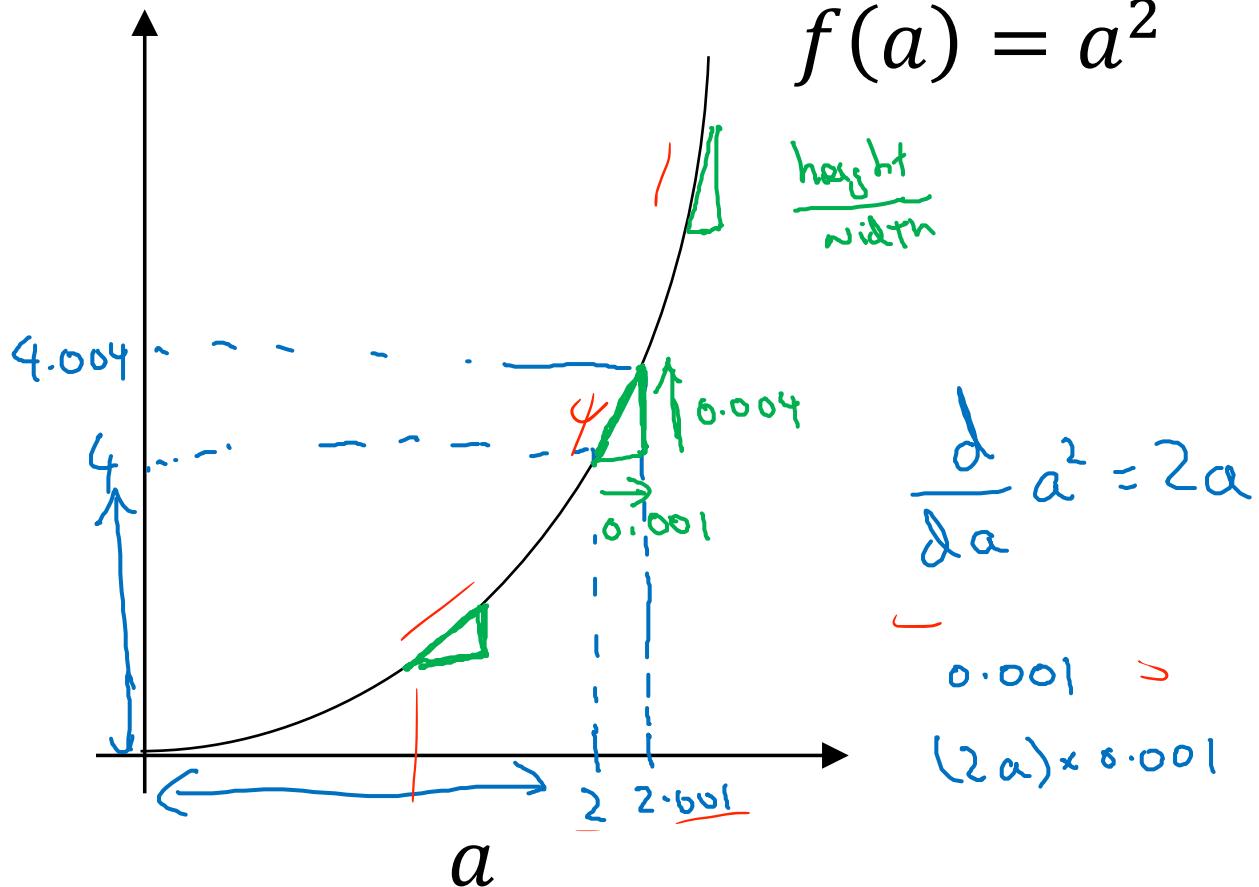


deeplearning.ai

Basics of Neural Network Programming

More derivatives
examples

Intuition about derivatives



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

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

$\frac{4.004 - 4}{0.001} = 4$

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

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

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

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

$\frac{25.010 - 25}{0.001} = 10$

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

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

More derivative examples

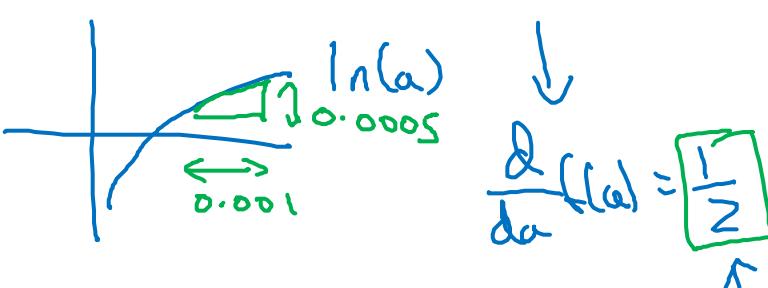
$$\underline{f(a) = a^2}$$

$$\frac{\partial}{\partial a} f(a) = \underbrace{2a}_{4}$$

$$\underline{f(a) = a^3}$$

$$\frac{\partial}{\partial a} f(a) = \underbrace{3a^2}_{3 \times 2^2} = 12$$

$$\underline{f(a) = \frac{\log_e(a)}{\ln(a)}}$$

$$\frac{\partial}{\partial a} f(a) = \frac{1}{a}$$


$$a = 2$$

$$a = 2.001$$

$$f(a) = 4$$

$$f(a) \approx 4.004$$

$$a = 2$$

$$a = 2.001$$

$$f(a) = 8$$

$$f(a) \approx 8.012$$

$$\downarrow a = 2$$

$$\downarrow a = 2.001$$

$$\downarrow f(a) \approx 0.69315$$

$$\downarrow f(a) \approx 0.69365$$

$$\frac{0.0005}{0.0005}$$



deeplearning.ai

Basics of Neural Network Programming

Computation Graph

Computation Graph

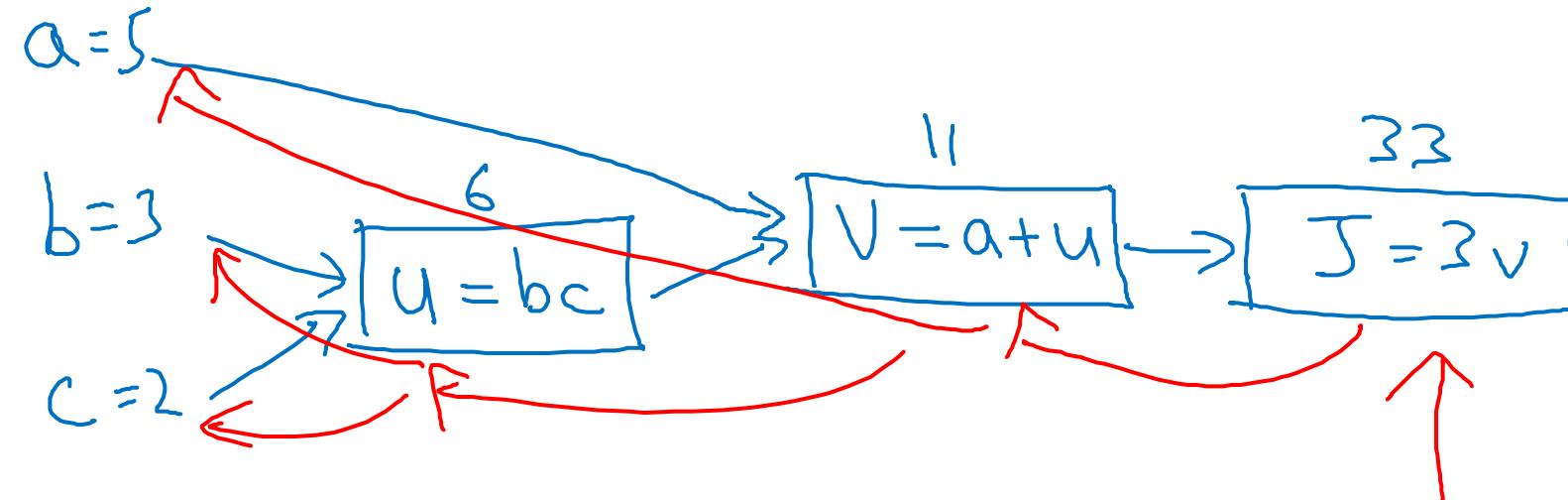
$$J(a, b, c) = 3(a + \underbrace{bc}_u) = 3(5 + 3 \times 2) = 33$$

$\underbrace{}_u$
 $\underbrace{}_v$
 $\underbrace{}_J$

$$u = bc$$

$$v = a + u$$

$$J = 3v$$



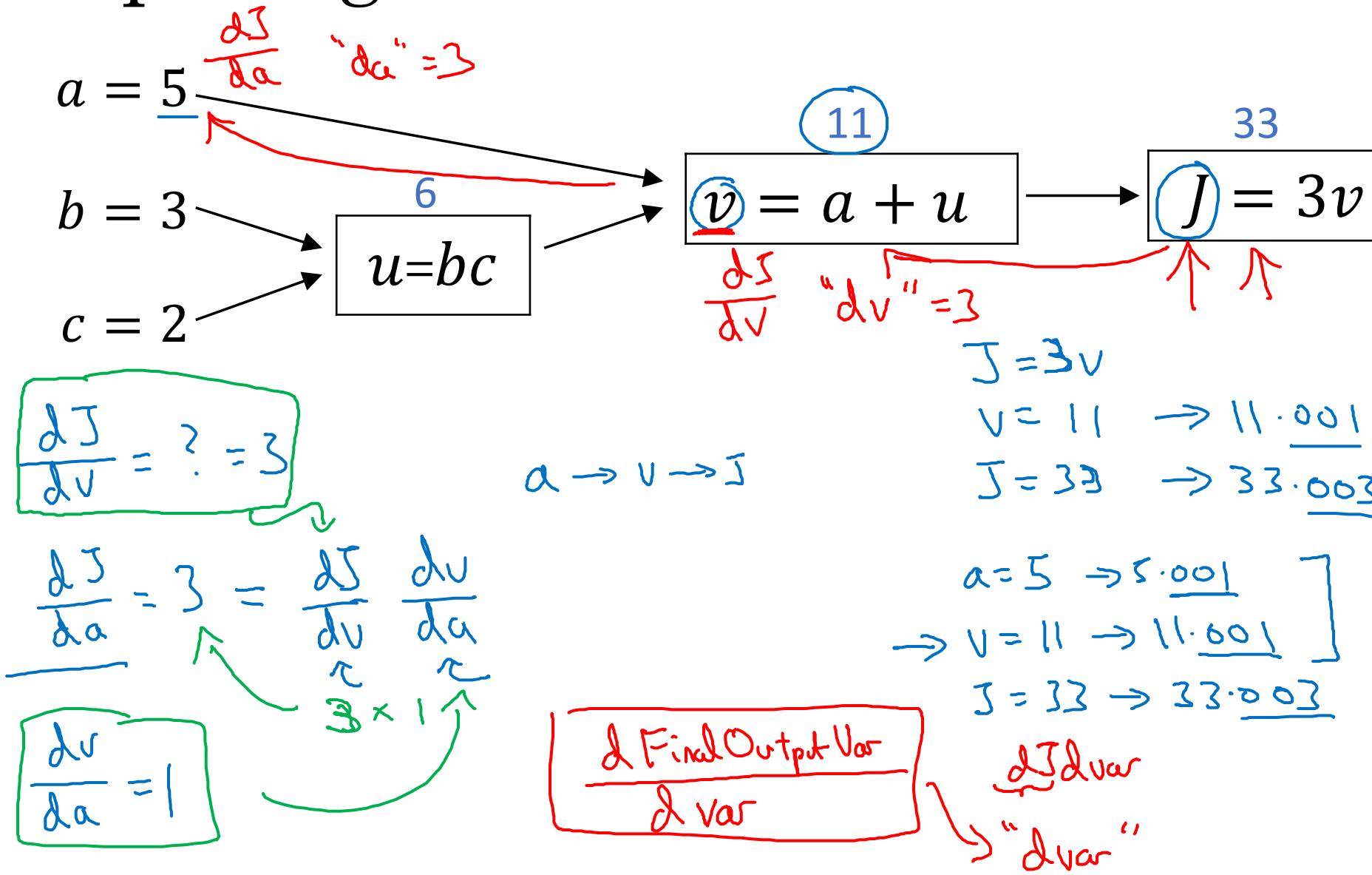


deeplearning.ai

Basics of Neural Network Programming

Derivatives with a Computation Graph

Computing derivatives



$f(a) = 3a$
 $\frac{df(b)}{da} = \frac{df}{da} = 3$
 $J = 3v$
 $\frac{dJ}{dv} = 3$

Computing derivatives

$\frac{\partial J}{\partial a} \rightarrow \frac{\partial a}{\partial a} = 3$
 $\frac{\partial J}{\partial b} \rightarrow \frac{\partial b}{\partial b} = 6$
 $\frac{\partial J}{\partial c} \rightarrow \frac{\partial c}{\partial c} = 9$
 $\frac{\partial J}{\partial u} = 3 = \frac{\partial J}{\partial v} \cdot \frac{\partial v}{\partial u}$
 $\frac{\partial J}{\partial b} = \frac{\partial J}{\partial u} \cdot \frac{\partial u}{\partial b} = 3 \cdot 2 = 6$
 $\frac{\partial J}{\partial a} = \frac{\partial J}{\partial u} \cdot \frac{\partial u}{\partial a} = 3 \cdot 3 = 9$
 $\frac{\partial J}{\partial c} = \frac{\partial J}{\partial u} \cdot \frac{\partial u}{\partial c} = 3 \cdot 3 = 9$

$a = 5 \rightarrow \frac{\partial a}{\partial a} = 3$
 $b = 3 \rightarrow \frac{\partial b}{\partial b} = 6$
 $c = 2 \rightarrow \frac{\partial c}{\partial c} = 9$
 $u = bc \rightarrow \frac{\partial u}{\partial u} = 3$
 $v = a + u \rightarrow \frac{\partial v}{\partial v} = 1$
 $J = 3v \rightarrow \frac{\partial J}{\partial J} = 3$

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

$b = 3 \rightarrow 3.001$
 $u = b \cdot c = 6 \rightarrow 6.002$
 $J = 33.006$

$v = 11.002$
 $J = 3v$



deeplearning.ai

Basics of Neural Network Programming

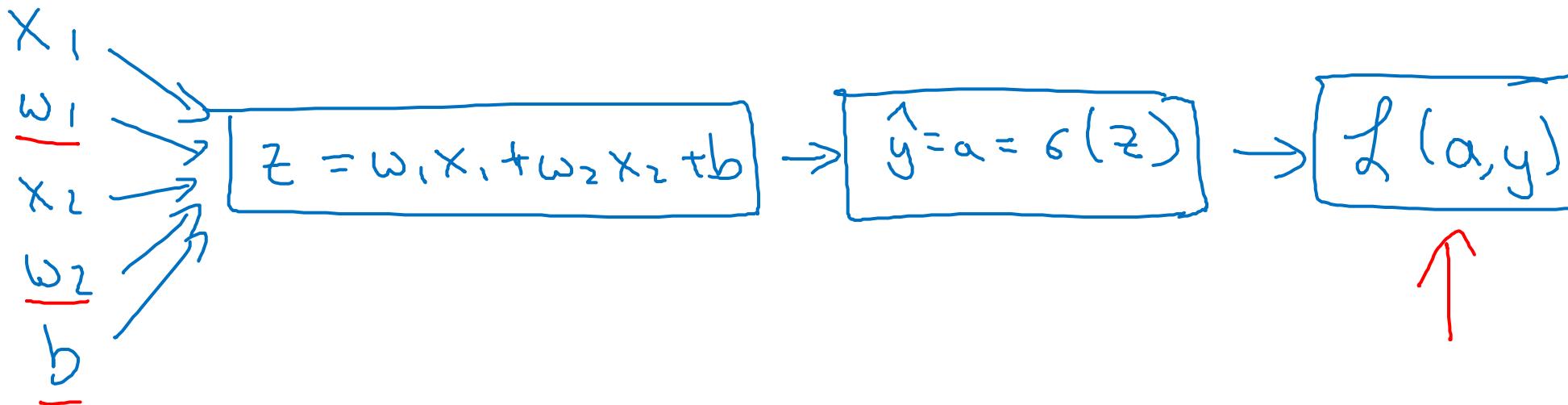
Logistic Regression Gradient descent

Logistic regression recap

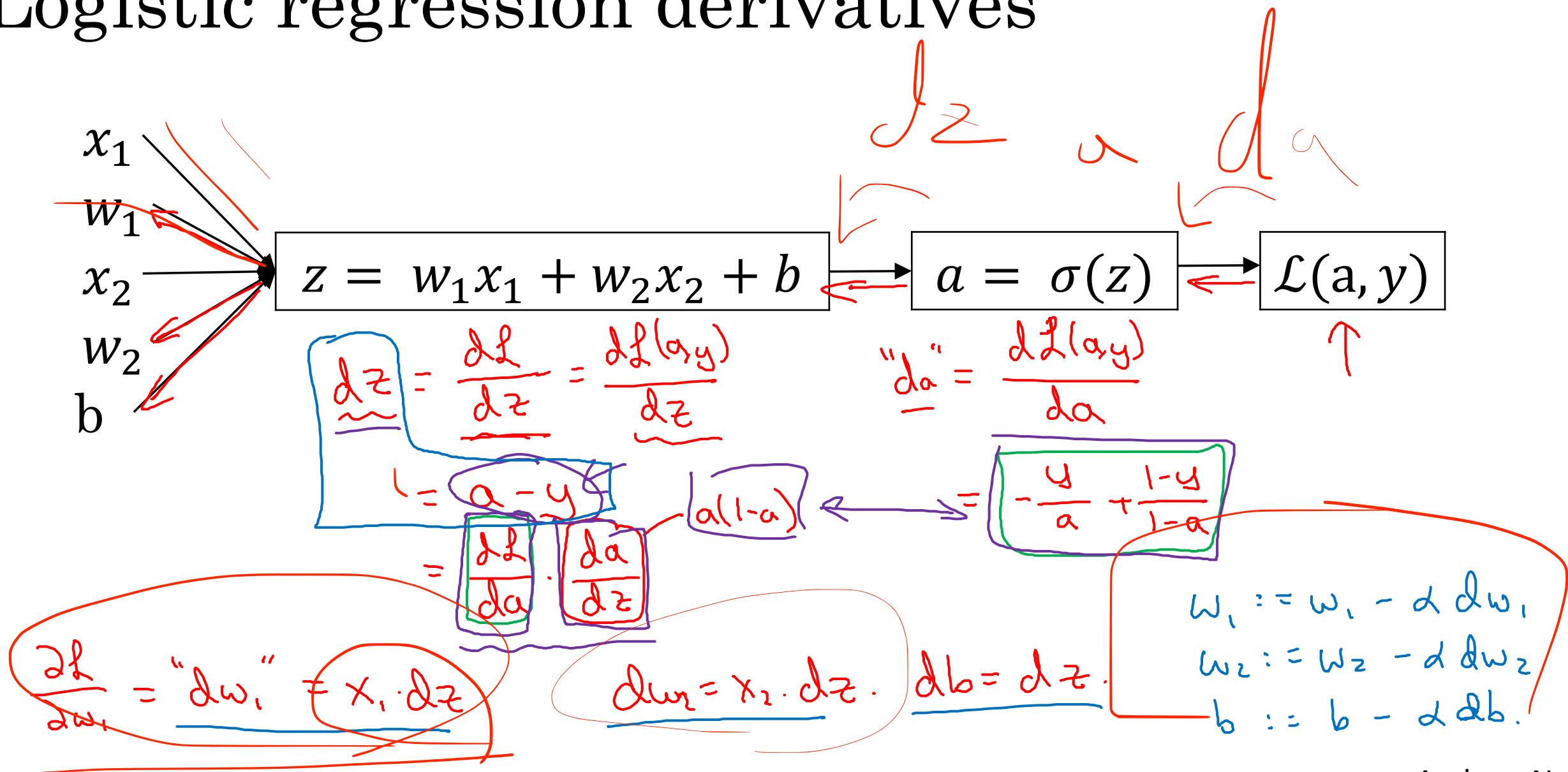
$$\rightarrow z = w^T x + b$$

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

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



Logistic regression derivatives





deeplearning.ai

Basics of Neural Network Programming

Gradient descent
on m examples

Logistic regression on m examples

$$\underline{J(\omega, b)} = \frac{1}{m} \sum_{i=1}^m \ell(a^{(i)}, y^{(i)})$$
$$\Rightarrow a^{(i)} = \hat{y}^{(i)} = g(z^{(i)}) = g(\omega^\top x^{(i)} + b)$$
$$\underline{dw_1^{(i)}}, \underline{dw_2^{(i)}}, \underline{db^{(i)}}$$

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

$$\underline{\frac{\partial}{\partial \omega_1} J(\omega, b)} = \frac{1}{m} \sum_{i=1}^m \underbrace{\frac{\partial}{\partial \omega_1} \ell(a^{(i)}, y^{(i)})}_{\underline{dw_1^{(i)}} - (x^{(i)}, y^{(i)})}$$

Logistic regression on m examples

$$J=0; \underline{\Delta w_1=0}; \underline{\Delta w_2=0}; \underline{\Delta b=0}$$

→ For $i = 1$ to m

$$z^{(i)} = \omega^\top x^{(i)} + b$$

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

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

$$\underline{\Delta z^{(i)}} = a^{(i)} - y^{(i)}$$

$$\begin{aligned} \Delta w_1 &+= x_1^{(i)} \Delta z^{(i)} \\ \Delta w_2 &+= x_2^{(i)} \Delta z^{(i)} \end{aligned}$$

$$\begin{aligned} \Delta w_3 & \\ \vdots & \\ \Delta w_n & \Delta b += \Delta z^{(i)} \end{aligned}$$

$$J / = m \leftarrow$$

$$\Delta w_1 / = m; \Delta w_2 / = m; \Delta b / = m. \leftarrow$$

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

$$w_1 := w_1 - \alpha \underline{\Delta w_1}$$

$$w_2 := w_2 - \alpha \underline{\Delta w_2}$$

$$b := b - \alpha \underline{\Delta b}.$$

Vectorization



deeplearning.ai

Basics of Neural Network Programming

Vectorization

What is vectorization?

$$z = \underbrace{\omega^T x}_{\text{Non-vectorized}} + b$$

Non-vectorized:

$$z = 0$$

for i in range(n - x):
 $z += \omega[i] * x[i]$

$$z += b$$

$$\omega = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix} \quad x = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}$$

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

Vectorized

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

\rightarrow GPU } SIMD - single instruction
 \rightarrow CPU } multiple data.



deeplearning.ai

Basics of Neural Network Programming

More vectorization examples

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

$$u = Av$$

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

$$u = np.zeros((n, 1))$$

for i ...

 for j ...

$$u[i] += A[:, i] * v[i]$$

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

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  
v/v
```

Logistic regression derivatives

$$J = 0, \boxed{dw_1 = 0, dw_2 = 0}, db = 0$$

$$dw = np.zeros((n_x, 1))$$

for i = 1 to n:

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

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

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

for $j=1 \dots n_x$
 $dw_j += \frac{\partial J}{\partial w_j}$

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

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

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

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

$$dw /= m.$$



deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

Vectorizing Logistic Regression

$$\begin{aligned} z^{(1)} &= w^T x^{(1)} + b \\ a^{(1)} &= \sigma(z^{(1)}) \end{aligned}$$

$$\underline{z^{(2)}} = \boxed{w^T x^{(2)} + b}$$
$$\underline{a^{(2)}} = \sigma(z^{(2)})$$

$$\underline{z}^{(3)} = w^T x^{(3)} + b$$
$$\underline{a}^{(3)} = \sigma(z^{(3)})$$

$$X = \underline{\underline{X}} = \underline{\underline{\underline{X}}}$$

Diagram illustrating the structure of a matrix X as a collection of vectors $x^{(1)}, x^{(2)}, \dots, x^{(m)}$. The matrix X is enclosed in a red box, and its columns $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ are enclosed in blue boxes. The vectors $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ are also enclosed in red boxes. A green arrow points to the bottom right corner of the matrix X .

$$\frac{(n_x, m)}{R^{n_x \times m}}$$

A diagram illustrating a sequence of sets $X^{(1)}, X^{(2)}, \dots, X^{(m)}$ within a domain D . The domain D is represented by a red-outlined irregular shape. Inside, a green rectangle labeled I contains a green line segment. A green arrow points from this segment to a green bracket that encloses the first set $X^{(1)}$. Subsequent sets $X^{(2)}, \dots, X^{(m)}$ are shown as overlapping green ovals, each enclosed by a green bracket. The sets are arranged horizontally, with ellipses indicating they continue. The boundary of the domain D is shown as a red line.

$$Z = \begin{bmatrix} Z^{(1)} & Z^{(2)} & \dots & Z^{(m)} \end{bmatrix} = \underbrace{w^T \times}_{1 \times m} + \underbrace{\begin{bmatrix} b & b & \dots & b \end{bmatrix}}_{1 \times m} = \underbrace{\begin{bmatrix} w^T X^{(1)} + b \\ \vdots \\ w^T X^{(m)} + b \end{bmatrix}}_{1 \times m} \dots \underbrace{\begin{bmatrix} w^T X^{(n)} + b \end{bmatrix}}_{1 \times m}$$

$$Z = \text{np.dot}(w.T, x) + b$$

$$\underline{A} = [a^{(1)} \ a^{(2)} \ \dots \ a^{(n)}] = \underline{G}(z)$$

“Broadcasting”



deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression's Gradient Computation

Vectorizing Logistic Regression

Diagram illustrating the vectorization of logistic regression gradients:

Left side (Gradients for \mathbf{w} and b):

- $d\mathbf{z}^{(1)} = a^{(1)} - y^{(1)}$
- $d\mathbf{z}^{(2)} = a^{(2)} - y^{(2)}$
- \dots
- $d\mathbf{z} = [d\mathbf{z}^{(1)} \ d\mathbf{z}^{(2)} \ \dots \ d\mathbf{z}^{(m)}]^T$ (highlighted in green)
- $\mathbf{A} = [a^{(1)} \ \dots \ a^{(m)}]$
- $\mathbf{Y} = [y^{(1)} \ \dots \ y^{(m)}]$
- $\rightarrow d\mathbf{z} = \mathbf{A} - \mathbf{Y} = [a^{(1)} - y^{(1)} \ a^{(2)} - y^{(2)} \ \dots]^T$
- $\rightarrow d\mathbf{w} = 0$
- $d\mathbf{w}^1 = \frac{1}{m} \sum_{i=1}^m d\mathbf{z}^{(i)}$
- $d\mathbf{w}^2 = \frac{1}{m} \sum_{i=1}^m d\mathbf{z}^{(i)}$
- \vdots
- $d\mathbf{w} = \frac{1}{m} \sum_{i=1}^m d\mathbf{z}^{(i)}$
- $d\mathbf{b} = 0$
- $d\mathbf{b}^1 = \frac{1}{m} \sum_{i=1}^m d\mathbf{z}^{(i)}$
- $d\mathbf{b}^2 = \frac{1}{m} \sum_{i=1}^m d\mathbf{z}^{(i)}$
- \vdots
- $d\mathbf{b} = \frac{1}{m} \sum_{i=1}^m d\mathbf{z}^{(i)}$

Right side (Final gradient calculations):

- $d\mathbf{b} = \frac{1}{m} \sum_{i=1}^m d\mathbf{z}^{(i)}$
- $= \frac{1}{m} \text{np. sum } (d\mathbf{z})$
- $d\mathbf{w} = \frac{1}{m} \mathbf{X}^T d\mathbf{z}$
- $= \frac{1}{m} \left[\begin{matrix} \mathbf{X}^{(1)} & \dots & \mathbf{X}^{(m)} \end{matrix} \right] \left[\begin{matrix} d\mathbf{z}^{(1)} \\ \vdots \\ d\mathbf{z}^{(m)} \end{matrix} \right]$
- $= \frac{1}{m} \left[\frac{1}{m} \sum_{i=1}^m \mathbf{X}^{(i)} d\mathbf{z}^{(i)} \right]$
- $= \frac{1}{m} \left[\frac{1}{m} \sum_{i=1}^m \mathbf{X}^{(1)} d\mathbf{z}^{(i)} + \dots + \frac{1}{m} \sum_{i=1}^m \mathbf{X}^{(m)} d\mathbf{z}^{(i)} \right]$
- $= \frac{1}{m} \left[\mathbf{X}^1 d\mathbf{z}^{(1)} + \dots + \mathbf{X}^m d\mathbf{z}^{(m)} \right]$

Implementing Logistic Regression

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

~~for i = 1 to m:~~

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

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

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

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

$$\left. \begin{array}{l} dw_1 += x_1^{(i)} dz^{(i)} \\ dw_2 += x_2^{(i)} dz^{(i)} \end{array} \right\} dw += X^{(i)} * dz^{(i)}$$

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

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

$$db = db/m$$

```
for iter in range(1000):  
    z = w^T X + b  
    = n p.dot(w.T, X) + b  
    A = σ(z)  
    d_z = A - Y  
    dw = 1/m * X * d_z^T  
    db = 1/m * np.sum(d_z)  
  
    w := w - α dw  
    b := b - α db
```



deeplearning.ai

Basics of Neural Network Programming

Broadcasting in Python

Broadcasting example

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

	Apples	Beef	Eggs	Potatoes
Carb	56.0	0.0	4.4	68.0
Protein	1.2	104.0	52.0	8.0
Fat	1.8	135.0	99.0	0.9

= A
(3,4)

$$59_{\text{cal}} \quad \frac{56}{59} \approx 94.9\%$$

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

```
cal = A.sum(axis = 0)
```

$$\text{percentage} = \frac{100 * \text{A} / (\text{cal.reshape}(1, 4))}{\uparrow(3, 4) / (1, 4)}$$

Broadcasting example

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

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

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

General Principle

$$\begin{array}{ccc} (m, n) & \begin{array}{c} + \\ - \\ \times \\ \diagup \end{array} & (1, n) \rightsquigarrow (m, n) \\ \underline{\text{matrix}} & & (m, 1) \rightsquigarrow (m, n) \end{array}$$

$$\begin{array}{ccc} (m, 1) & + & \mathbb{R} \\ \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} & + & 100 & = & \begin{bmatrix} 101 \\ 102 \\ 103 \end{bmatrix} \\ \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} & + & 100 & = & \begin{bmatrix} 101 & 102 & 103 \end{bmatrix} \end{array}$$

Matlab/Octave: bsxfun



deeplearning.ai

Basics of Neural Network Programming

Explanation of logistic
regression cost function
(Optional)

Logistic regression cost function

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

Interpret $\hat{y} = p(y=1|x)$

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

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

Logistic regression cost function

→ If $y = 1$: $p(y|x) = \hat{y}$

→ If $y = 0$: $p(y|x) = 1 - \hat{y}$

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

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

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

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

$= -\frac{1}{m} \sum \log (\hat{y} \cdot y)$

Cost on m examples

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

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

Maximum likelihood
estimation \nearrow

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

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