Basics of Neural Network Programming

Binary Classification

Binary Classification

64

64

→ 1 (cat) vs 0 (non cat)

y

Blue
Green
Red

255	134	93	22
255	134	202	22	2
255	231	42	22	4	30
123	94	83	2	192	124
34	44	187	92	34	142
34	76	232	124	94
67	83	194	202

64

64
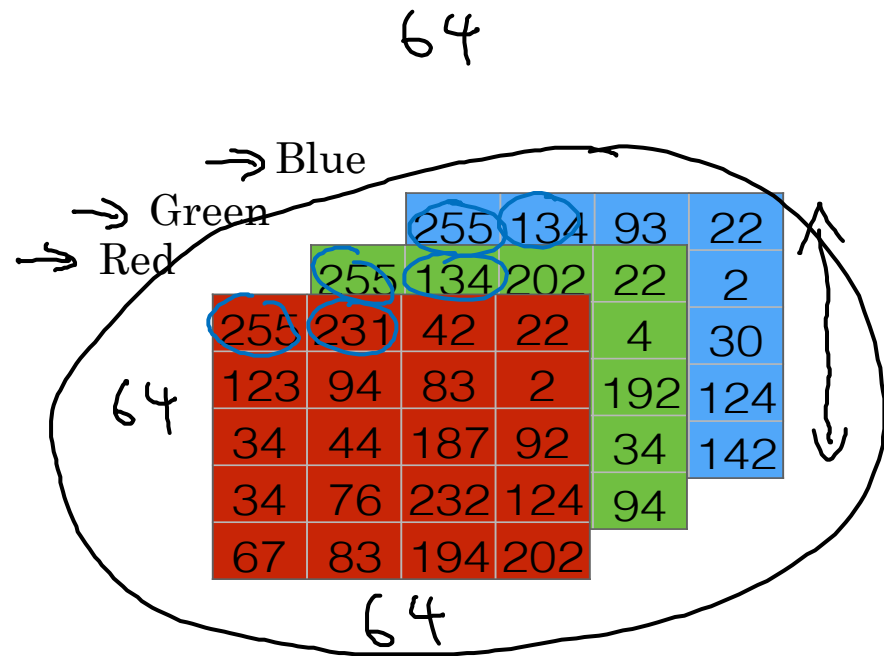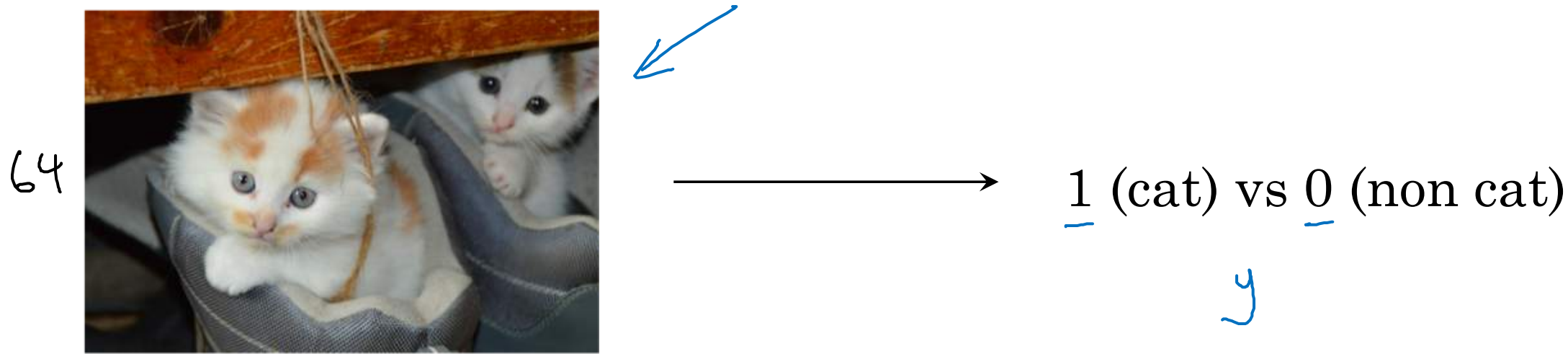
$$x = \begin{bmatrix} 255 \\ 231 \\ \vdots \\ 255 \\ 134 \\ \vdots \end{bmatrix}$$

$64 \times 64 \times 3 = 12288$

$n = n_x = 12288$

$x \longrightarrow y$

Andrew Ng

Notation

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

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

$m = M_{train}$ $M_{test} = \#test\ examples.$

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

$$\underleftrightarrow{\hspace{2cm} m \hspace{2cm}}$$

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

$Y = \begin{bmatrix} y^{(1)} & y^{(2)} & \ldots & , y^{(m)} \end{bmatrix}$

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

$Y.shape = (1, m)$

Basics of Neural Network Programming

Logistic Regression

deeplearning.ai

Logistic Regression

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

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

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

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

Output $\hat{y} = \sigma(\underbrace{w^T x + b}_{z})$
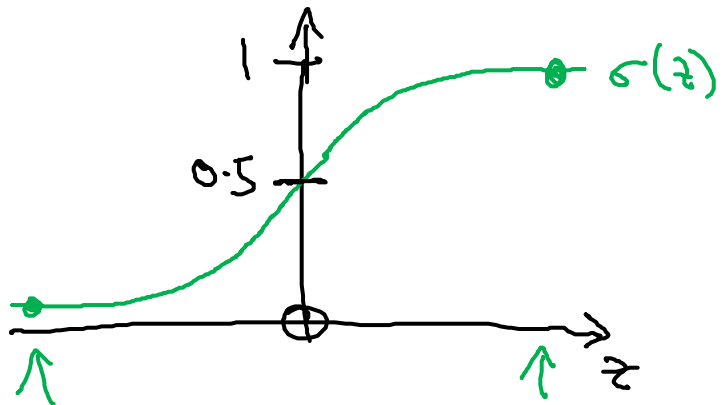
$x_0 = 1$, $x \in \mathbb{R}^{n_x + 1}$

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

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \begin{matrix} \}b \leftarrow \\ \\ \}w \leftarrow \\ \\ \end{matrix}$$

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

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

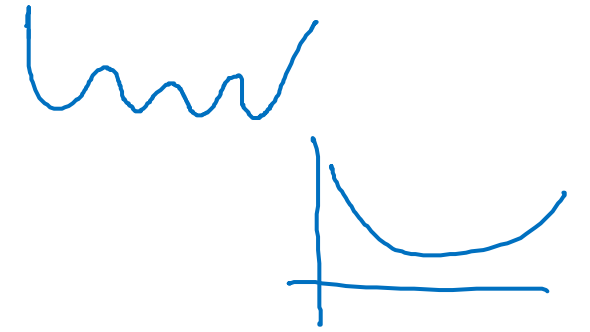## Logistic Regression cost function

deeplearning.ai

Logistic Regression cost function

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

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

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

$x^{(i)}$
$y^{(i)}$ $i-th$ example.
$z^{(i)}$

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

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

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

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

Andrew Ng

Basics of Neural Network Programming

Gradient Descent

deeplearning.ai

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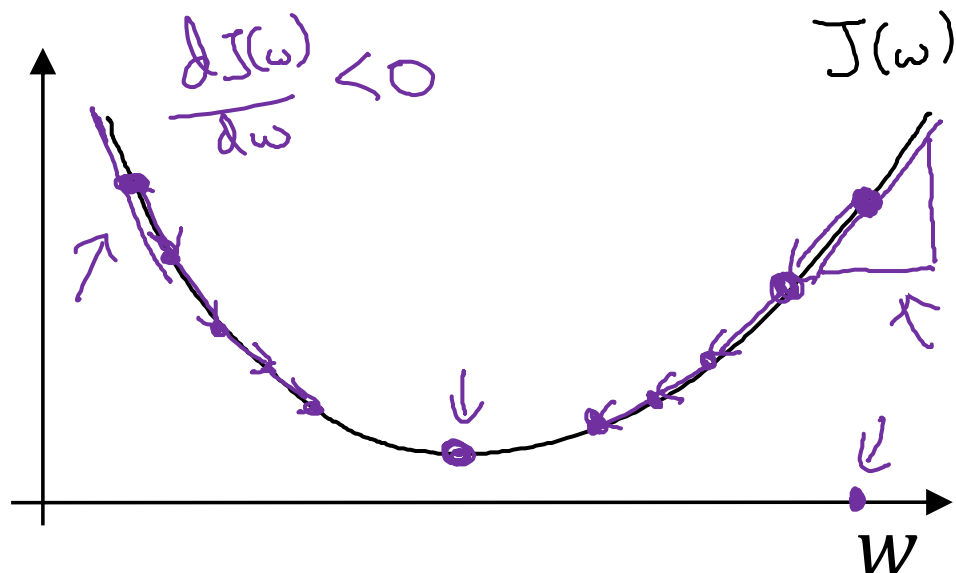
Recap: $\hat{y} = \sigma(w^T x + b), \ \sigma(z) = \frac{1}{1+e^{-z}}$ ⟵

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

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

$J(w,b)$

b

w

⟵ Global optinum

Gradient Descent

$\dfrac{dJ(\omega)}{d\omega} < 0$

$J(\omega)$

w

Repeat {

learning rate

$\omega := \omega - \alpha \dfrac{dJ(\omega)}{d\omega}$

}

"dw"

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

$\dfrac{dJ(\omega)}{d\omega} = ?$

$J(\omega, b)$

$\omega := \omega - \alpha \dfrac{d\,J(\omega,b)}{d\omega}$

$b := b - \alpha \dfrac{d\,J(\omega,b)}{db}$

$\dfrac{\partial J(\omega,b)}{\partial \omega} \rightarrow dw$

$\dfrac{\partial J(\omega,b)}{\partial b} \rightarrow db$

∂ "partial derivative"

∂ J

Basics of Neural Network Programming

Derivatives

deeplearning.ai

Intuition about derivatives

$f(a) = 3a$
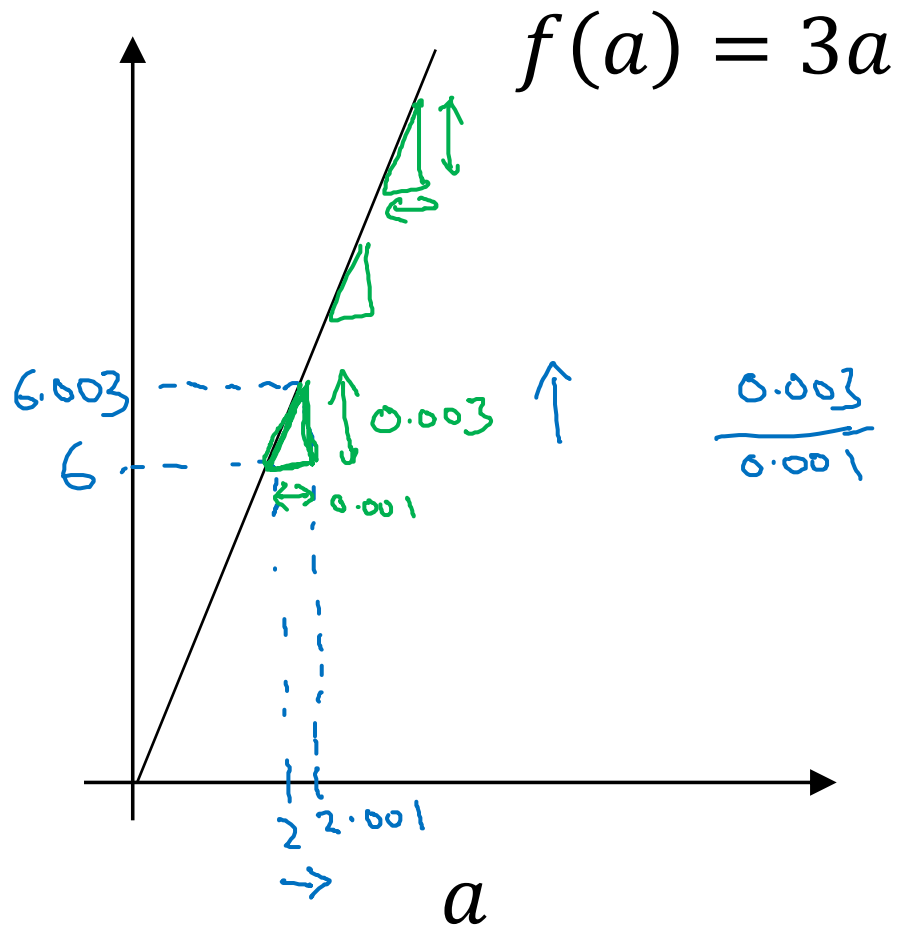
$a = 2$ $f(a) = 6$

$a = 2.001$ $f(a) = 6.003$

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

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

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

$a = 5.001$ $f(a) = 15.003$

slope at $a = 5$ is also 3

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

0.001
0.0000001
0.000000001

deeplearning.ai

Basics of Neural
Network Programming

More derivatives
examples

Intuition about derivatives
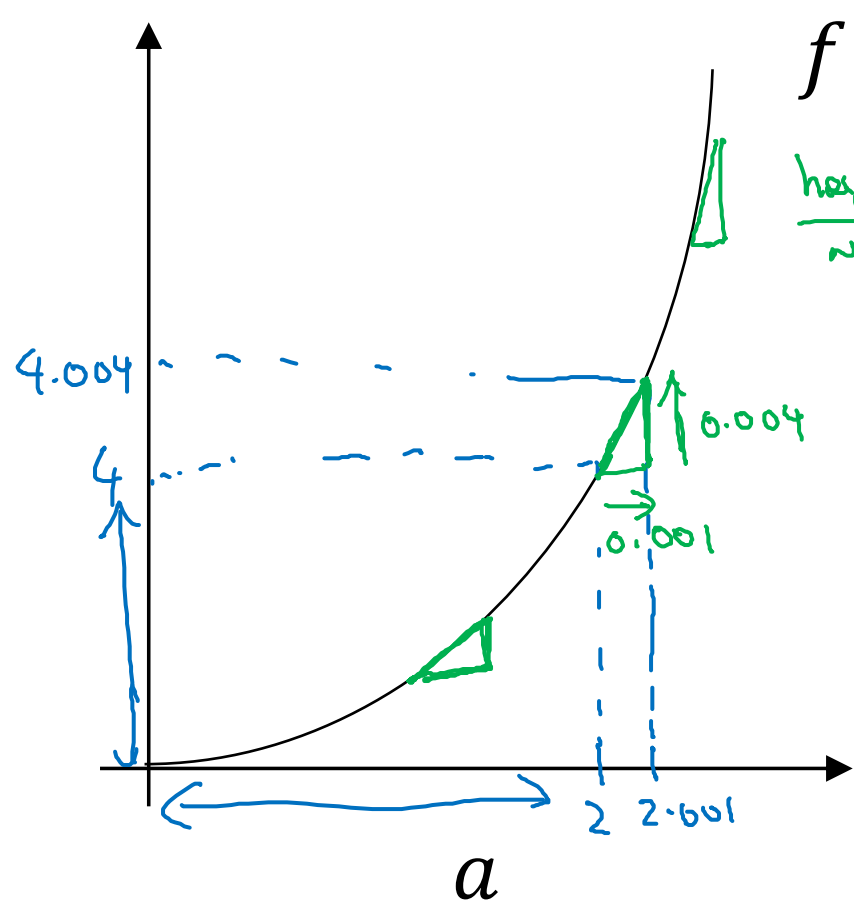
$f(a) = a^2$

$0.001 \leftarrow$
$0.00000....01 \leftarrow$

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

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

$(4.004\boxed{004})$

slope (derivative) of $f(a)$ at

$a = 2$ is   4.

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

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

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

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

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

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

0.001

$(2a) \times 0.001$

height / width

4.004

4

0.004

0.001

$2 \quad 2.001$

a

Andrew Ng

More derivative examples

$$f(a) = a^2$$

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

$$a = 2 \qquad f(a) = 4$$
$$a = 2.001 \qquad f(a) \approx 4.004$$

$$f(a) = a^3$$

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

$$a = 2 \qquad f(a) = 8$$
$$a = 2.001 \qquad f(a) \approx 8.012$$

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

$$\frac{d}{da} f(a) = \frac{1}{a}$$
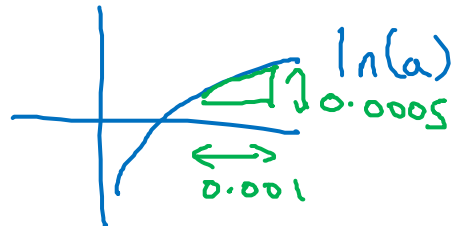
$$a = 2 \qquad f(a) \approx 0.69315$$
$$a = 2.001 \qquad f(a) \approx 0.69365$$

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

$$\ln(a)$$
$$0.0005$$
$$0.001$$

$$0.0005$$
$$0.0005$$

Andrew Ng

Basics of Neural Network Programming

Computation Graph

deeplearning.ai

Computation Graph

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

$$\underbrace{\underbrace{}_{u},}_{}$$
$$\underbrace{}_{J}$$

$u = bc$

$V = a + u$

$J = 3v$

$a = 5$

$b = 3$ 6   11   33

$u = bc$ $V = a + u$ $J = 3v$

$c = 2$
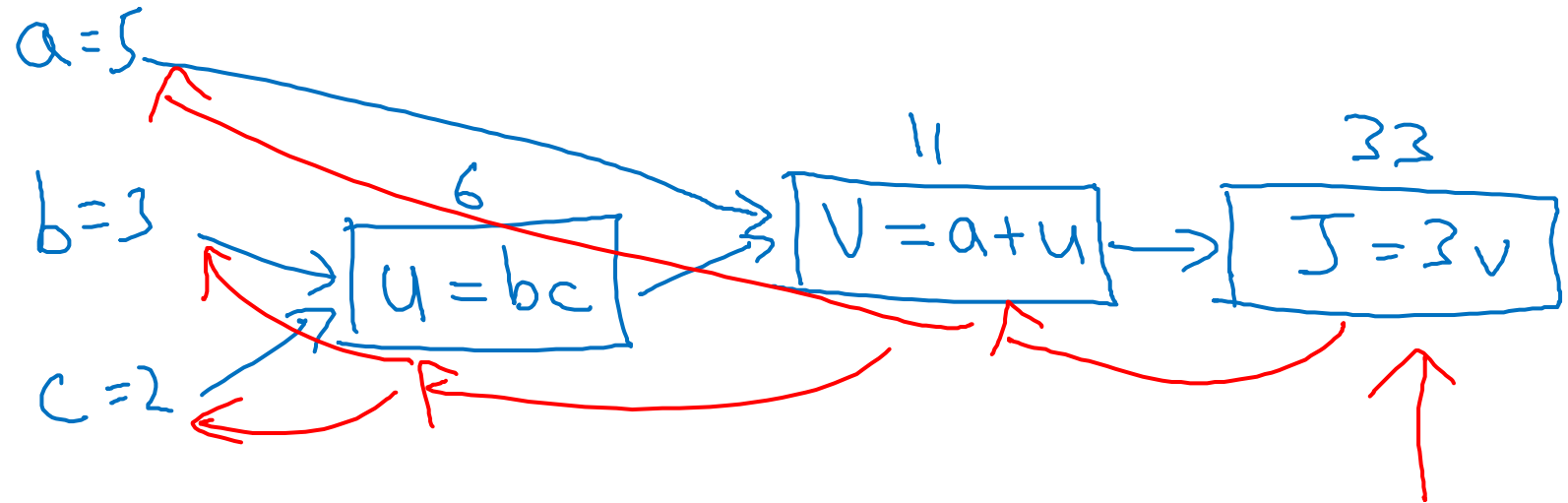
Basics of Neural Network Programming

Derivatives with a Computation Graph

Computing derivatives
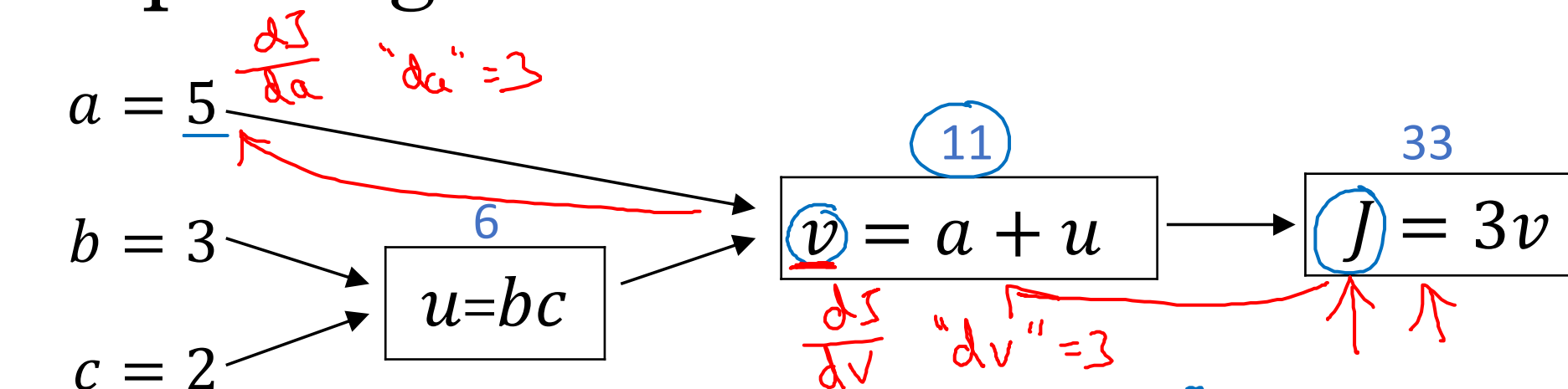
$\dfrac{dJ}{da}$ "da" = 3

$a = 5$

11

33

$b = 3$

6

$u=bc$

$\boxed{v = a + u}$

$\boxed{J = 3v}$

$c = 2$

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

$a \rightarrow v \rightarrow J$

$J = 3v$

$v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

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

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

3×1

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

$a = 5 \rightarrow 5.001$

$\rightarrow v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

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

$\dfrac{dJ}{d\,var}$

"dvar"

$\boxed{f(a) = 3a}$

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

$\boxed{J = 3v}$

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

Andrew Ng

Computing derivatives

$\dfrac{dJ}{da}$

$\rightarrow \dfrac{da}{} = 3$

$a = 5$

$\dfrac{dJ}{db} = \dfrac{db}{} = 6$

$b = 3$

$\rightarrow \dfrac{dc}{} = 9$

$c = 2$

6

$\boxed{u = bc}$

$\dfrac{du}{} = 3$

11

$\boxed{v = a + u}$

$\dfrac{dv}{} = 3 \qquad \dfrac{dJ}{dJ}$

33

$\boxed{J = 3v}$

$\dfrac{dJ}{du} = 3 = \dfrac{dJ}{dv} \cdot \dfrac{dv}{du}$
$\qquad\qquad\quad \underbrace{}_{3} \quad \underbrace{}_{1}$

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

$\dfrac{dJ}{da} = \boxed{\dfrac{dJ}{du}} \cdot \dfrac{du}{da} = 9$
$\qquad\qquad\qquad \underbrace{}_{3 \times 3}$

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

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

$v = 11.002$
$J = 3v$

Andrew Ng
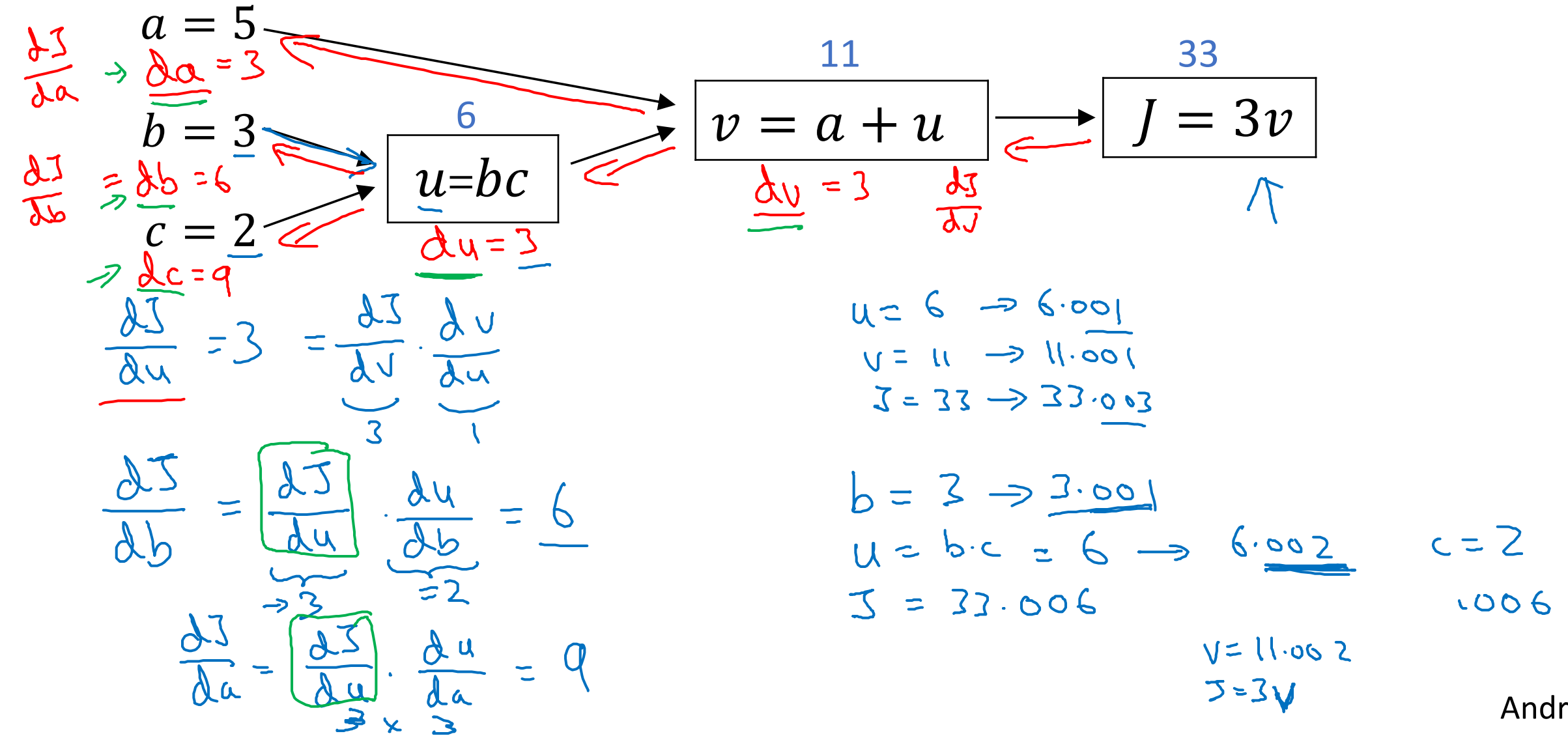
Basics of Neural
Network Programming

Logistic Regression
Gradient descent

deeplearning.ai

Logistic regression recap

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

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

$$\mathcal{L}(a, y) = -(y \log(a) + (1-y) \log(1-a))$$
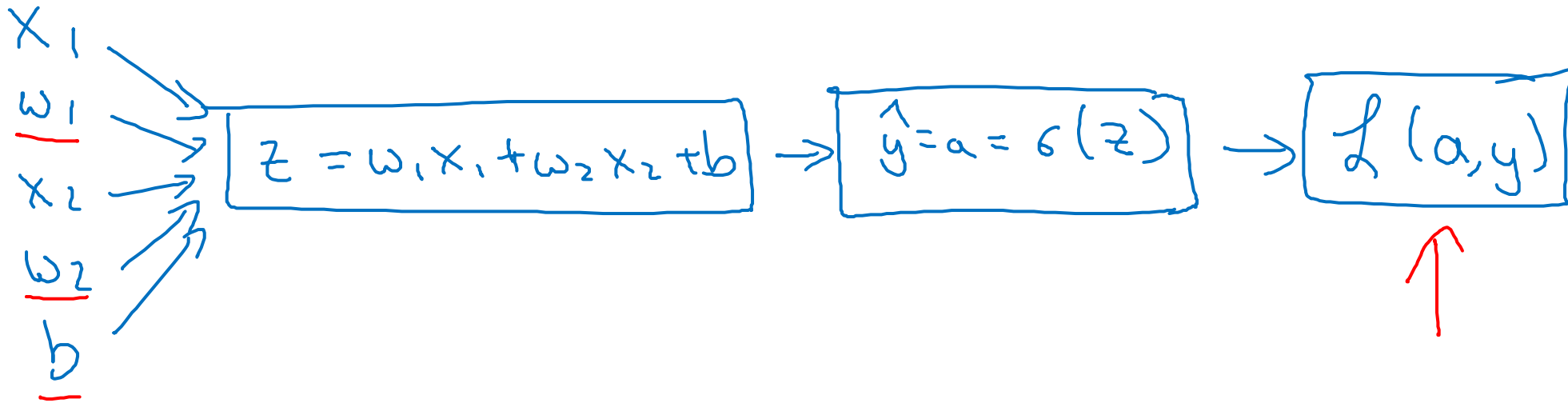
$$x_1$$
$$w_1$$
$$x_2$$
$$w_2$$
$$b$$

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

Logistic regression derivatives

x_1

w_1

x_2

w_2

b

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

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

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

$$= a - y$$

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

$$a(1-a) \longleftarrow \quad = -\frac{y}{a} + \frac{1-y}{1-a}$$

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

$$w_1 := w_1 - \alpha \, dw_1$$
$$w_2 := w_2 - \alpha \, dw_2$$
$$b := b - \alpha \, db.$$

Andrew Ng
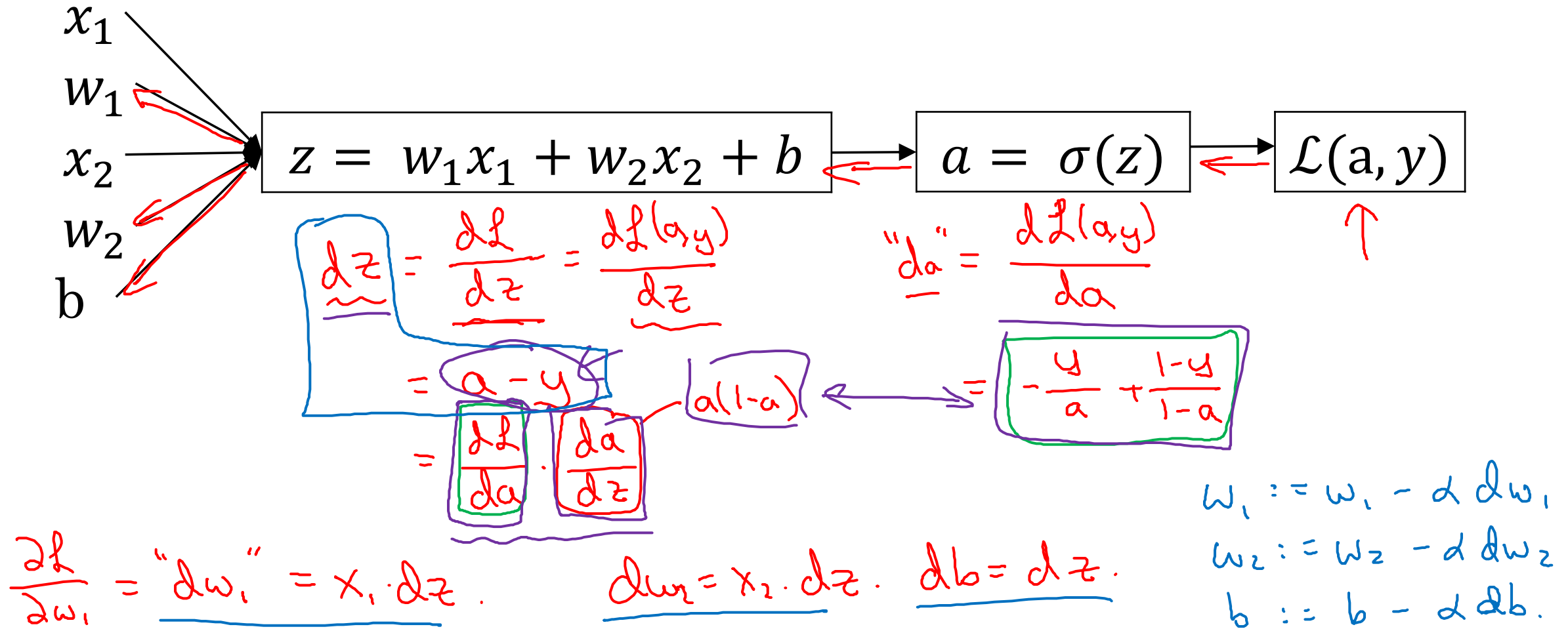
Basics of Neural Network Programming

Gradient descent on m examples

Logistic regression on m examples

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

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

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

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

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

$$dw_1^{(i)} \quad - \quad (x^{(i)}, y^{(i)})$$

Logistic regression on m examples

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

$\rightarrow$ For $i = 1$ to m

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

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

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

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

$\qquad dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$ $\Big\}$ $n = 2$

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

dw_3
$\vdots$
dw_n

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

$J \mathrel{/}= m$ $\Leftarrow$

$dw_1 \mathrel{/}= m$; $dw_2 \mathrel{/}= m$; $db \mathrel{/}= m$. $\Leftarrow$

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

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

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

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

Vectorization

Basics of Neural Network Programming

Vectorization

deeplearning.ai

What is vectorization?

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

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

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

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

Non-vectorized:

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

$$z += b$$

Vectorized

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

$\underbrace{\qquad\qquad\qquad}_{w^T x}$

$\rightarrow$ GPU $\Big\}$ SIMD — single instruction

$\rightarrow$ CPU $\Big\}$ multiple data.

Andrew Ng

Basics of Neural Network Programming

More vectorization examples

deeplearning.ai

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

$$u = Av$$

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

$$u = np.zeros((n,1))$$
$$\text{for } i \dots$$
$$\quad \text{for } j \dots$$
$$\qquad u[i] += A[i][j] * v[j]$$

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

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

$$J = 0, \boxed{dw1 = 0, \; dw2 = 0}, db = 0$$

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

$\rightarrow$ 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...n_x
$dw_j += \ldots$

$$\boxed{\begin{array}{l} dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)} \\ dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)} \end{array}}$$

$n_x = 2$

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

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

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

$dw \mathbin{/}= m.$

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

Vectorizing Logistic Regression

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

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

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

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

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

$$w^T \begin{bmatrix} x^{(1)} & x^{(2)} & \cdots & x^{(m)} \end{bmatrix}$$

$$Z = \begin{bmatrix} z^{(1)} & z^{(2)} & \cdots & z^{(m)} \end{bmatrix} = w^T X + \begin{bmatrix} b & b \cdots b \end{bmatrix} = \begin{bmatrix} w^T x^{(1)} + b & w^T x^{(2)} + b & \cdots & w^T x^{(m)} + b \end{bmatrix}$$

$$1 \times m \qquad 1 \times m$$

"Broadcasting"

$$Z = np.dot(w.T, X) + b \qquad (1,1) \quad \mathbb{R}$$

$$A = \begin{bmatrix} a^{(1)} & a^{(2)} & \cdots & a^{(m)} \end{bmatrix} = \sigma(Z)$$

Andrew Ng

Basics of Neural Network Programming

deeplearning.ai

Vectorizing Logistic Regression's Gradient Computation

Vectorizing Logistic Regression

$$dz^{(1)} = a^{(1)} - y^{(1)} \qquad dz^{(2)} = a^{(2)} - y^{(2)} \qquad \cdots$$

$$\boxed{dz} = \underbrace{[dz^{(1)} \; dz^{(2)} \cdots dz^{(m)}]}_{1 \times m} \quad \Leftarrow$$

$$A = [a^{(1)} \cdots a^{(m)}] . \qquad Y = [y^{(1)} \cdots y^{(m)}]$$

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

$$\rightarrow dw = 0$$
$$dw \mathrel{+}= x^{(1)} dz^{(1)}$$
$$dw \mathrel{+}= x^{(2)} dz^{(2)}$$
$$\vdots$$
$$dw / = m$$

$$db = 0$$
$$db \mathrel{+}= dz^{(1)}$$
$$db \mathrel{+}= dz^{(2)}$$
$$\vdots$$
$$db \mathrel{+}= dz^{(m)}$$
$$db / = m.$$

$$db = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)}$$

$$= \frac{1}{m} \, np.\,sum(dz)$$

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

$$= \frac{1}{m} \begin{bmatrix} x^{(1)} \cdots x^{(m)} \\ 1 \quad 1 \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$$

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

$$n \times 1$$

Andrew Ng

Implementing Logistic Regression

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

for i = 1 to m:

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

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

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

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

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

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

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

db = db/m

for iter in range(1000): $\Leftarrow$

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

$$A = \sigma(Z)$$

$$dZ = A - Y$$

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

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

$$w := w - \alpha \, dw$$
$$b := b - \alpha \, db$$
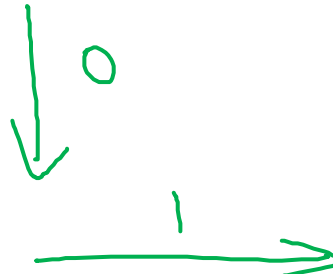
Andrew Ng

Basics of Neural Network Programming

Broadcasting in Python

deeplearning.ai

Broadcasting example

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

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

$(3,4)$

59 cal

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

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

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

$\uparrow (3,4) \quad / \quad (1,4)$

Broadcasting example

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

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 100 & 200 & 300 \\ 100 & 200 & 300 \end{bmatrix}$$

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

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 100 & 100 & 100 \\ 200 & 200 & 200 \end{bmatrix} =$$

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

General Principle

(m, n)

matrix

$+$
$-$
$*$
$/$

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

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

$(m, 1)$

$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

$[1 \ 2 \ 3]$

$+$

$+$

$+$

$\mathbb{R}$

100

100

$= \begin{bmatrix} 101 \\ 102 \\ 103 \end{bmatrix}$

$= [101 \quad 102 \quad 103]$

Matlab/Octave: bsxfun

deeplearning.ai

Basics of Neural Network Programming

Explanation of logistic regression cost function (Optional)

Logistic regression cost function

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

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

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

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

Andrew Ng

Logistic regression cost function

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

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

$\qquad\qquad\qquad\qquad\qquad\qquad$ $p(y|x)$

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

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

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

$\uparrow \log p(y|x) = \log \hat{y}^{y}(1-\hat{y})^{(1-y)} = y \log \hat{y} + (1-y)\log(1-\hat{y})$
$\qquad\qquad\qquad = -\mathcal{L}(\hat{y}, y) \downarrow$

Cost on m examples

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

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

Maximum likelihood estimate $\nwarrow$

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

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

Andrew Ng