Basics of Neural Network Programming

Binary Classification

deeplearning.ai

Binary Classification

64

64

$\longrightarrow$ 1 (cat) vs 0 (non cat)

y

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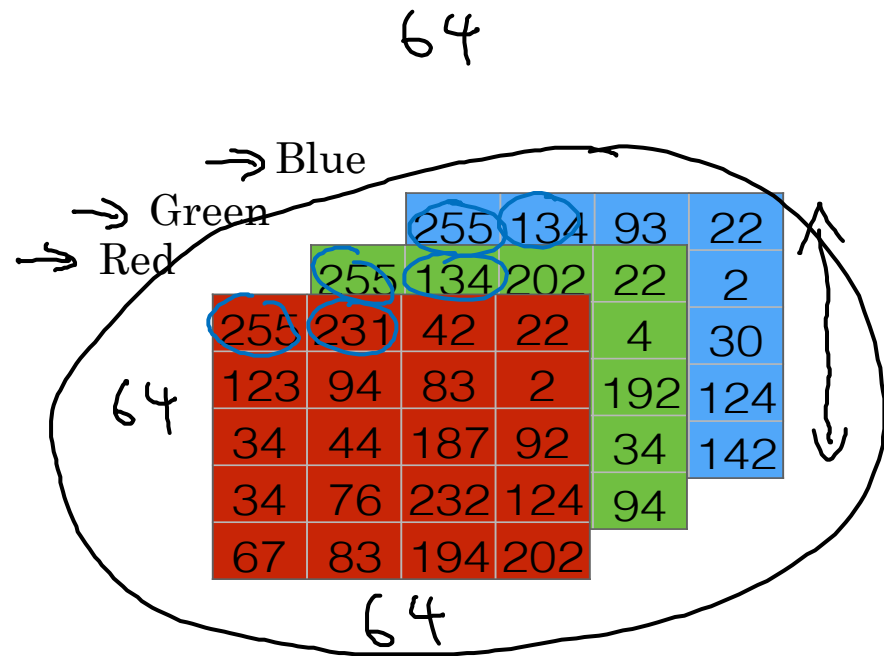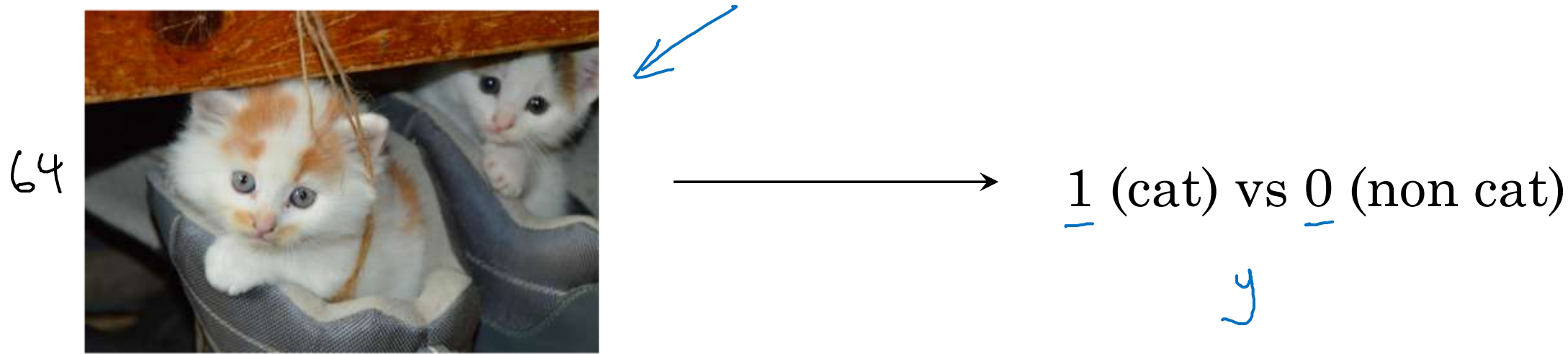
$$x = \begin{bmatrix} 255 \\ 231 \\ \vdots \\ 255 \\ 134 \\ \vdots \end{bmatrix}$$

$64 \times 64 \times 3 = 12288$

$n = n_x = 12288$

$x \longrightarrow y$

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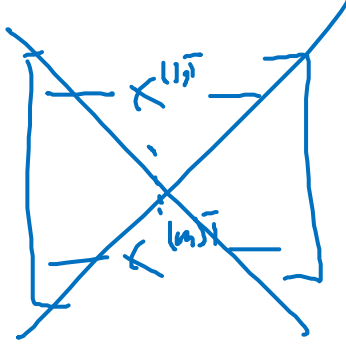
$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \ldots & x^{(m)} \\ | & | & & | \end{bmatrix} \Big\} n_x$$

$$\xleftarrow{\hspace{1cm}} m \xrightarrow{\hspace{1cm}}$$

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

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

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

$Y.shape = (1, m)$

Logistic Regression

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

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

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

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

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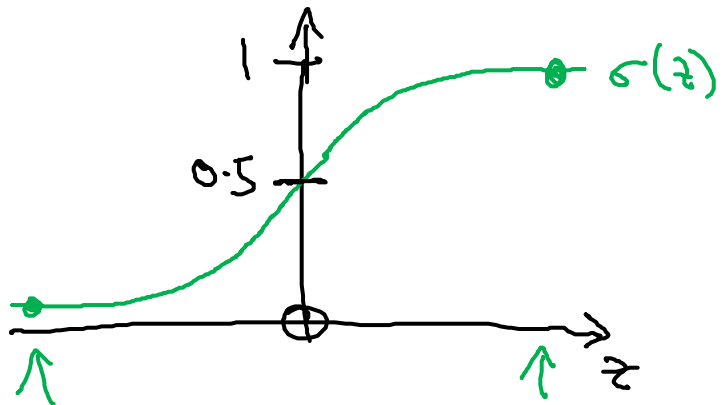
$x_0 = 1$, $x \in \mathbb{R}^{n_x+1}$

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

$$\Theta = \begin{bmatrix} \Theta_0 \\ \Theta_1 \\ \Theta_2 \\ \vdots \\ \Theta_{n_x} \end{bmatrix} \begin{matrix} \}b \leftarrow \\ \\ \}w \leftarrow \\ \\ \end{matrix}$$

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

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

If z large negative number

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

Andrew Ng

Basics of Neural Network Programming
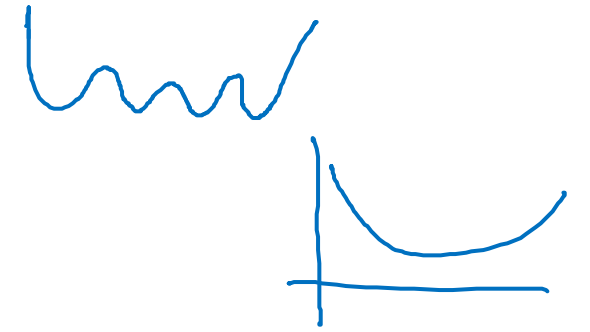
Logistic Regression cost function

deeplearning.ai

Logistic Regression cost function

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

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

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

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

$\boxed{\text{Loss}}$ (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 $\underline{y=1}$: $\mathcal{L}(\hat{y}, y) = - \log \hat{y} \leftarrow$ Want $\log \hat{y}$ large, Want $\hat{y}$ large.

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

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

Andrew Ng

Basics of Neural Network Programming

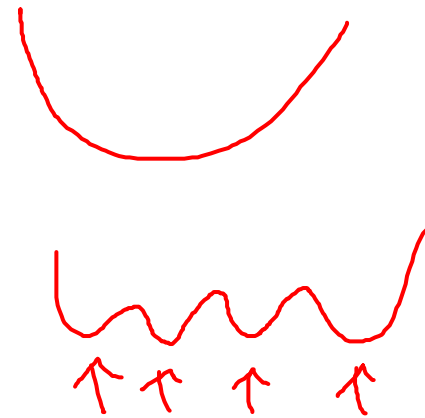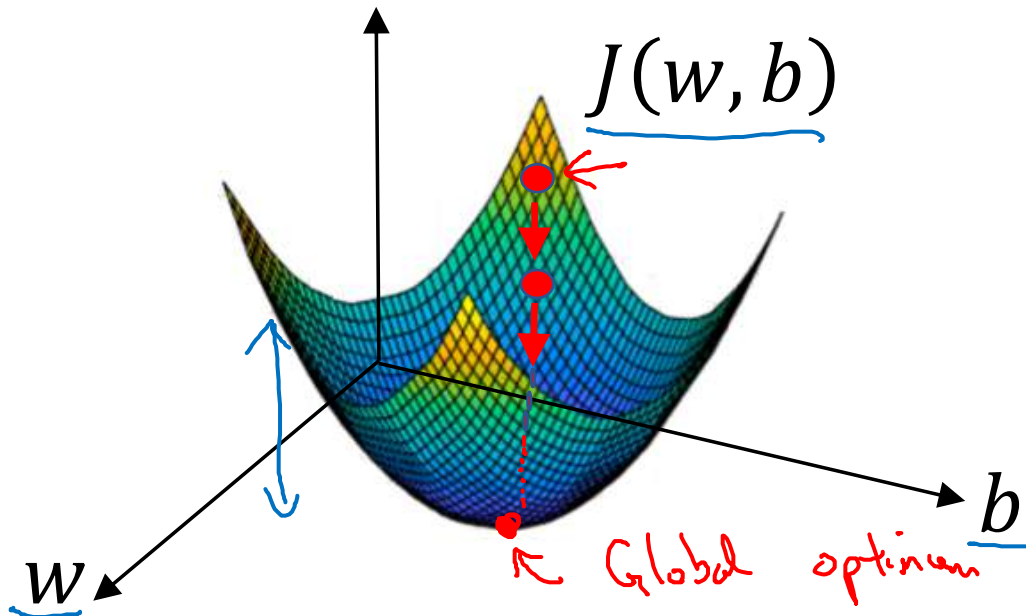## Gradient Descent

deeplearning.ai

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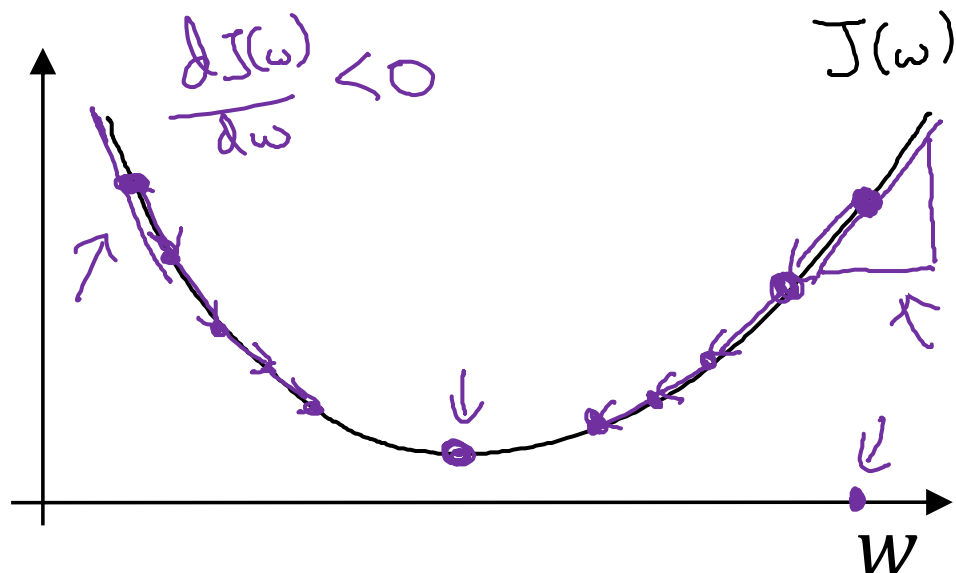
Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$ ⟵

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

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

$J(w,b)$

w

b

← Global optimum

Gradient Descent

$J(w)$

$$\frac{dJ(w)}{dw} < 0$$

w

Repeat {

learning rate

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

}

"dw"

$$w := w - \alpha \underline{dw}$$

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

$J(w,b)$

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

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

$$\boxed{\frac{\partial J(w,b)}{\partial w}}$$

$$\boxed{\frac{\partial J(w,b)}{\partial b}}$$

∂ "partial derivative"

∂ J

dw

db

Andrew Ng

Basics of Neural Network Programming

Derivatives

deeplearning.ai

Intuition about derivatives

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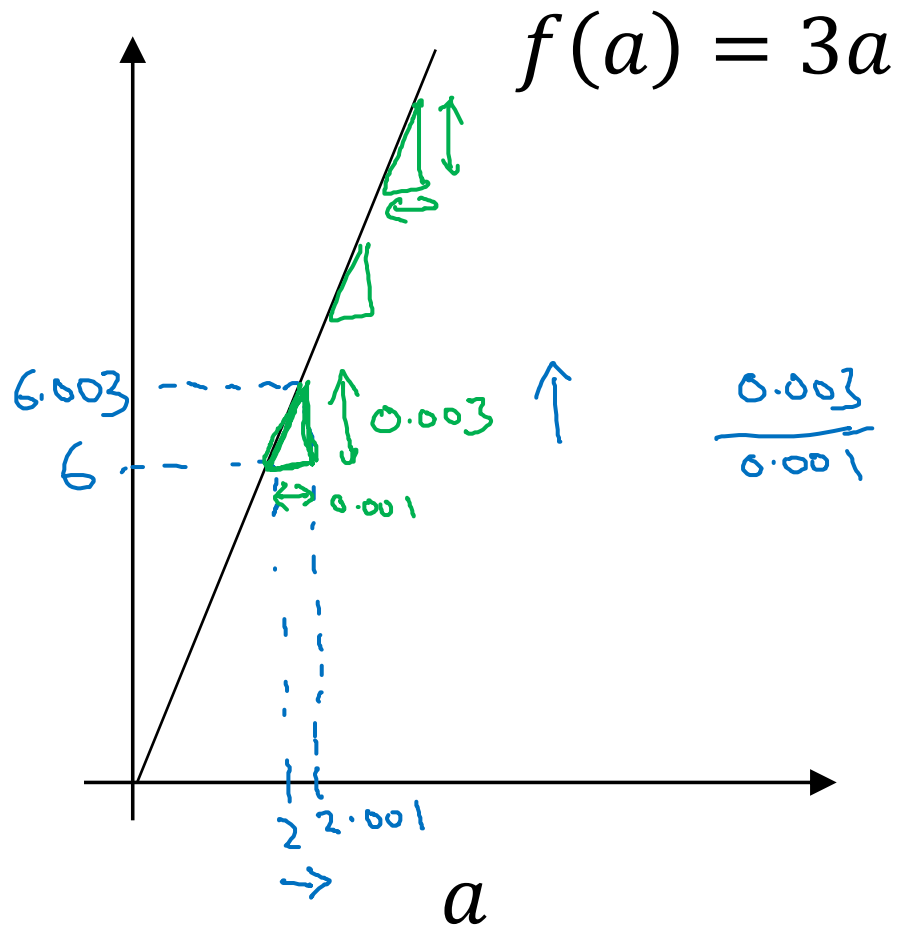
$a = 2$ $f(a) = 6$

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

slope (derivative) of $f(a)$

$\dfrac{0.003}{0.001}$ $\dfrac{height}{width}$ at $a=2$ is 3

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

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

slope at $a=5$ is also 3

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

0.001

0.00000001

0.00000001

Andrew Ng

Basics of Neural
Network Programming

More derivatives
examples

deeplearning.ai

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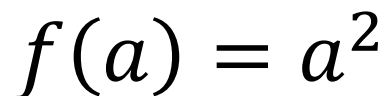
$$f(a) = a^2$$

height / width

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

0.001

$(2a) \times 0.001$

4.004

0.004

0.001

4

2 2.001

a

0.001 ←

$0.00000\ldots.01$ ←

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

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

$(4.004\boxed{004})$

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

$a = 2$ is   4.

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

$f(a) = 25$

$a = 5$

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

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

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

More derivative examples

$f(a) = a^2$

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

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

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

$f(a) = a^3$

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

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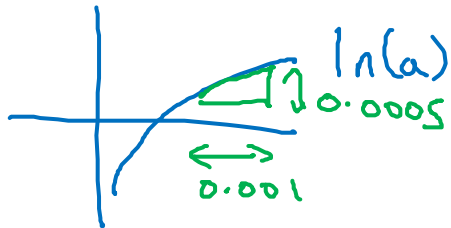
$a = 2.001$ $f(a) \approx 8.012$

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

$\ln(a)$

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

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

$\ln(a)$ 0.0005 0.001

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

$a = 2.001$ $f(a) \approx 0.69365$

0.0005

0.0005

Andrew Ng

Basics of Neural Network Programming

Computation Graph

deeplearning.ai

Computation Graph

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

u ,

v

J

$u = bc$

$V = a + u$

$J = 3v$

$a = 5$

$b = 3$ 6 $\boxed{u = bc}$ 11 $\boxed{V = a + u}$ 33 $\boxed{J = 3v}$

$c = 2$
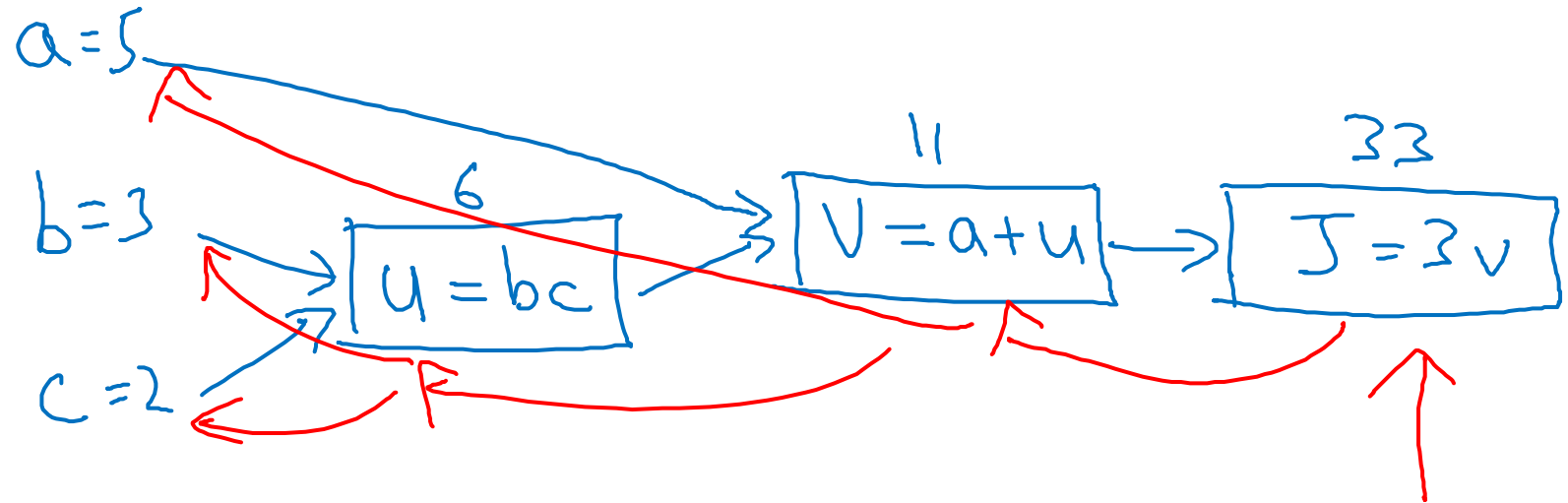
Basics of Neural Network Programming

Derivatives with a Computation Graph

deeplearning.ai

Computing derivatives

$a = 5$

$$\frac{dJ}{da} \quad \text{``}da\text{''} = 3$$

$b = 3$

$c = 2$

$$\boxed{u=bc}$$
6

$$\boxed{v = a + u}$$
11

$$\frac{dJ}{dv} \quad \text{``}dv\text{''} = 3$$

$$\boxed{J = 3v}$$
33

$a \to v \to J$

$J = 3v$

$v = 11 \to 11.001$

$J = 33 \to 33.003$

$f(a) = 3a$

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

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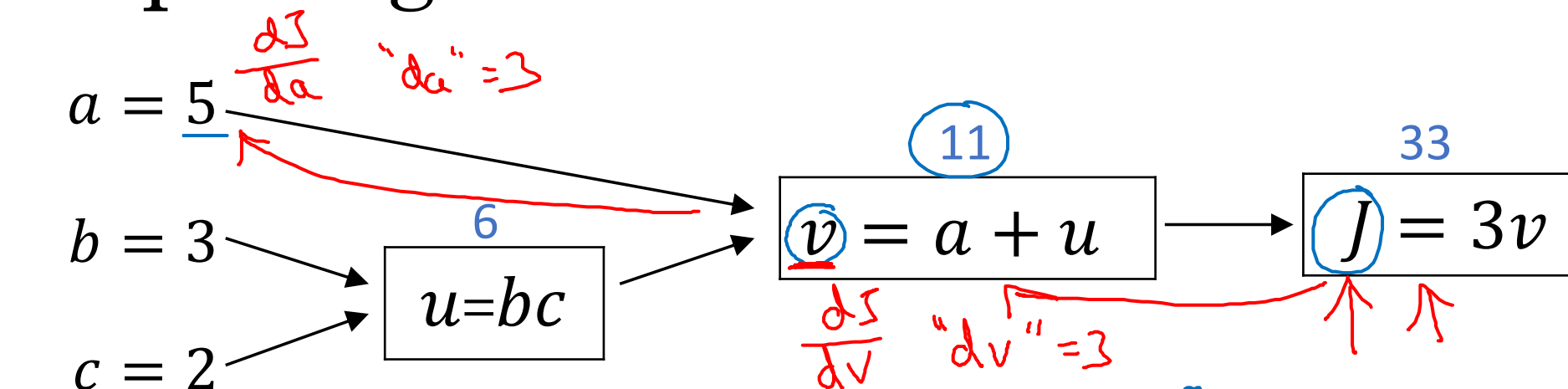
$$\frac{dJ}{dv} = 3$$

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

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

3×1

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

$a = 5 \to 5.001$

$\to v = 11 \to 11.001$

$J = 33 \to 33.003$

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

$\frac{dJ}{dvar}$

``$dvar$''

Andrew Ng

Computing derivatives

$\frac{dJ}{da}$ → $\frac{da}{=3}$

$a = 5$

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

$b = 3$

$u = bc$ 6

→ $\frac{dc}{=9}$

$c = 2$

$du = 3$

$v = a + u$ 11

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

$J = 3v$ 33
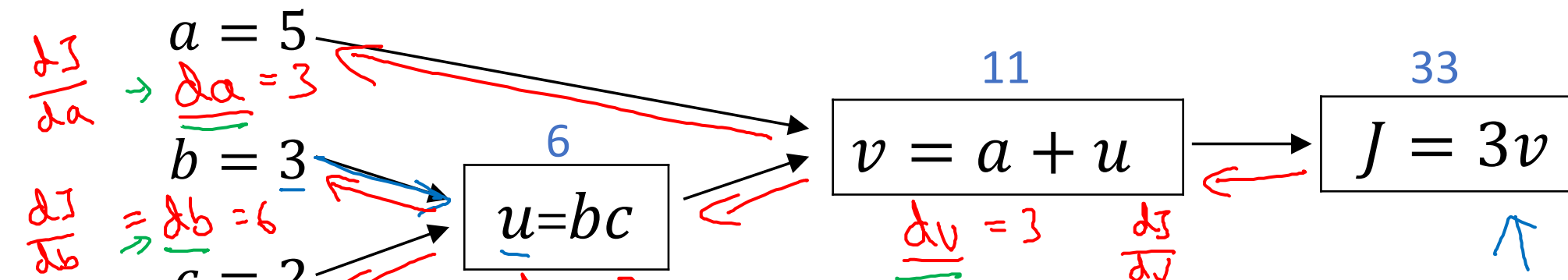
$\frac{dJ}{du} = 3 = \frac{dJ}{dv} \cdot \frac{dv}{du}$

3   1

$\frac{dJ}{db} = \boxed{\frac{dJ}{du}} \cdot \frac{du}{db} = \frac{6}{}$

→3   =2

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

3 × 3

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

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

$v = 11.002$
$J = 3v$

Andrew Ng

Logistic regression recap

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

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

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

Logistic regression derivatives

x_1

w_1

x_2

w_2

b

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

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

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

$$= a - y$$

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

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

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

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

Andrew Ng
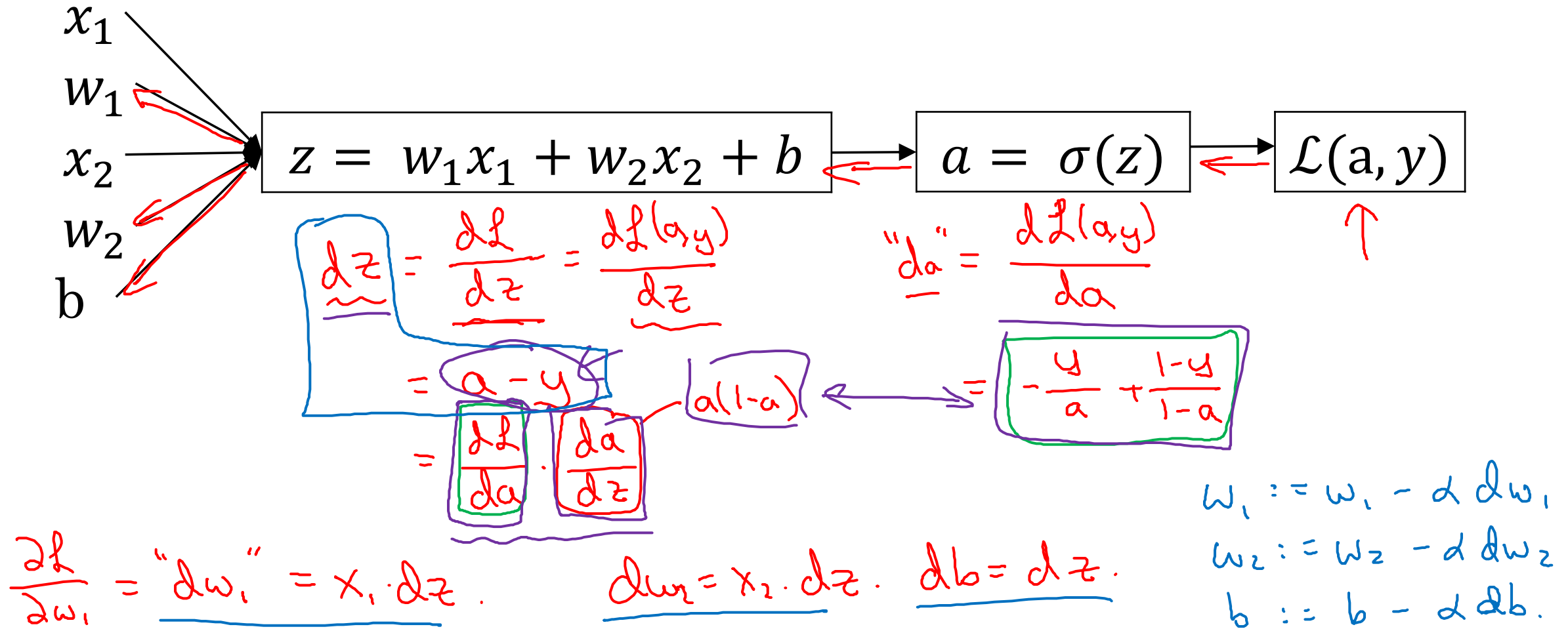
Basics of Neural Network Programming

deeplearning.ai

Gradient descent on m examples

Logistic regression on m examples

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

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

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

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

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

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

Logistic regression on m examples

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

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

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

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

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

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

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

$\quad dw_2 += x_2^{(i)} dz^{(i)} \qquad \Big] \; n = 2$

dw_3
$\vdots$
dw_n

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

$J /= m$

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

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

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

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

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

Vectorization

Andrew Ng

What is vectorization?

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

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

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

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 $\Big\}$ SIMD – single instruction

→ 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.

$$u = Av$$

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

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

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

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   1/v

Andrew Ng

Logistic regression derivatives

$J = 0,$ ~~$dw1 = 0, dw2 = 0$~~$, db = 0$

$dw = np.zeros((n\text{-}x, 1))$

$\rightarrow$ for i = 1 to n:

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

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

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

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

for j=1...nx
$dw_j \mathrel{+}= ...$

~~$dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$~~
~~$dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}$~~

$n_x = 2$

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

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

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

$dw /= m.$

Andrew Ng

Basics of Neural Network Programming

Vectorizing Logistic Regression

deeplearning.ai

Vectorizing Logistic Regression

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

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

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

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

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

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

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

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

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

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

"Broadcasting"

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

Basics of Neural Network Programming

deeplearning.ai

Vectorizing Logistic Regression's Gradient Computation

Vectorizing Logistic Regression

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

$$dZ = [\, dz^{(1)} \; dz^{(2)} \; \cdots \; dz^{(m)} \,]$$
$$\underset{1 \times m}{}$$

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

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

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

$$dw_1$$
$$dw_2$$
$$\vdots$$

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

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

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

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

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

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

Andrew Ng

Implementing Logistic Regression

J = 0, dw_1 = 0, dw_2 = 0, db = 0
for i = 1 to m:

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

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

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

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

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

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

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

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

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

for iter in range(1000): ←

$$Z = w^T X + b$$

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

$$A = \sigma(Z)$$

$$dZ = A - Y$$

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

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

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

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

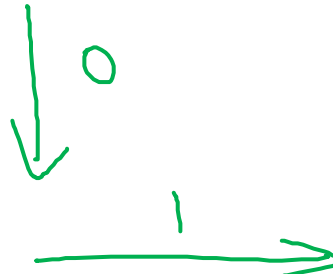# Basics of Neural Network Programming

deeplearning.ai

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 cal

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

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

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

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

Broadcasting example

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

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

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

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

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

ξ

(m,n)

General Principle

(m, n) $\underline{+}$ $(1, n)$ $\leadsto$ (m, n)

matrix $\underline{\displaystyle \ast}$

 $/$ $(m, 1)$ $\leadsto$ (m, n)

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

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

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

Matlab/Octave: $\underline{\text{bsxfun}}$

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

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

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

Andrew Ng

Logistic regression cost function

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

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

$$\Big\} \quad p(y|x)$$

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

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

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

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

$= -\underbrace{\ell(\hat{y}, y)}_{} \downarrow$

Andrew Ng

Cost on m examples

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

$\log p(\ldots) = \sum_{i=1}^{m} \log p(y^{(i)} | x^{(i)})$

$\underbrace{}_{-\mathcal{L}(\hat{y}^{(i)}, y^{(i)})}$

Maximum likelihood estimate ↖

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

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

Andrew Ng