Basics of Neural Network Programming

Binary Classification

Binary Classification

64

64

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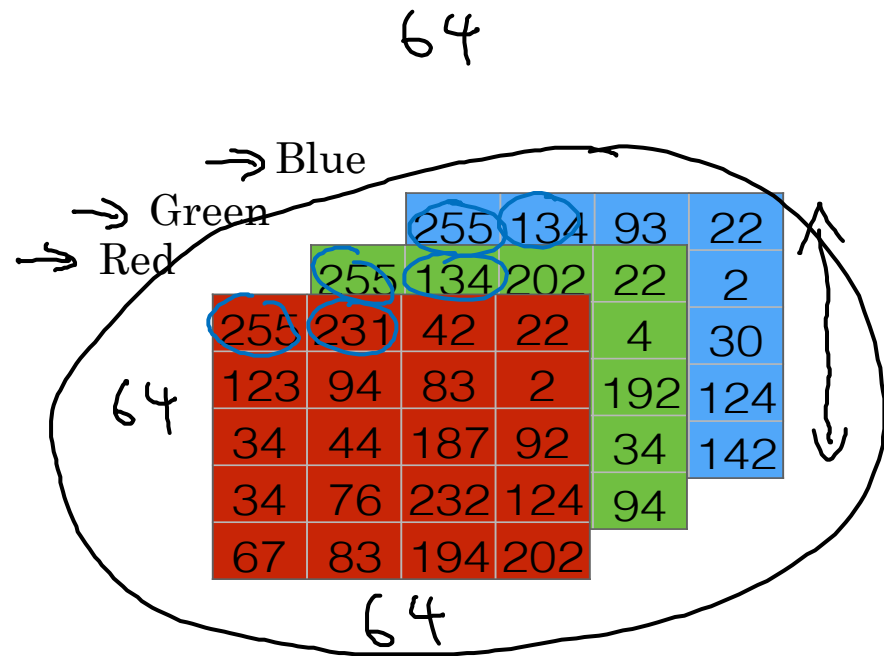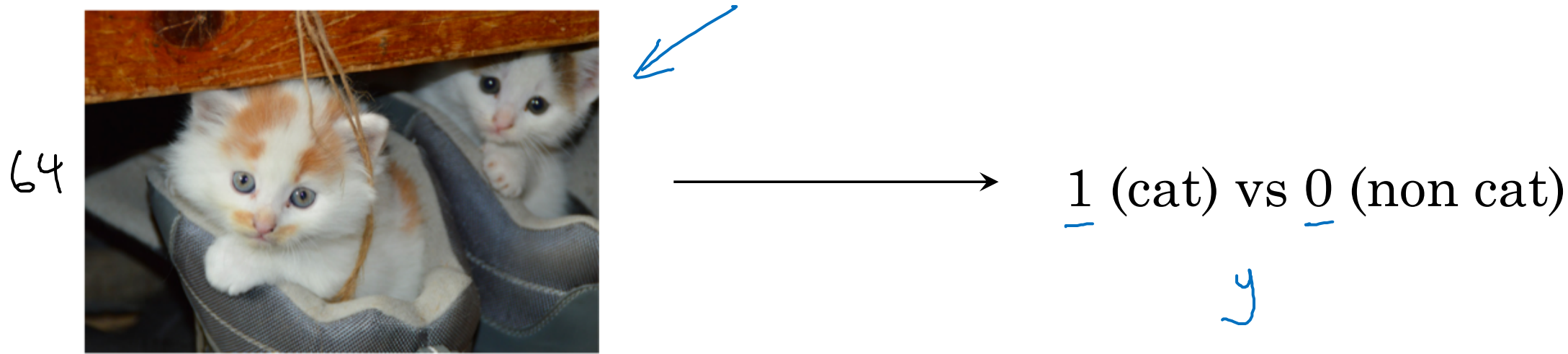
$$x = \begin{bmatrix} 255 \\ 231 \\ \vdots \\ 255 \\ 134 \\ \vdots \end{bmatrix}$$

$64 \times 64 \times 3 = 12288$

$n = n_x = 12288$

$x \longrightarrow y$

Andrew Ng

Notation

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

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

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

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

$\xleftarrow{\quad} m \xrightarrow{\quad}$

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

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

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

$Y.shape = (1, m)$

Basics of Neural
Network Programming

Logistic Regression

deeplearning.ai
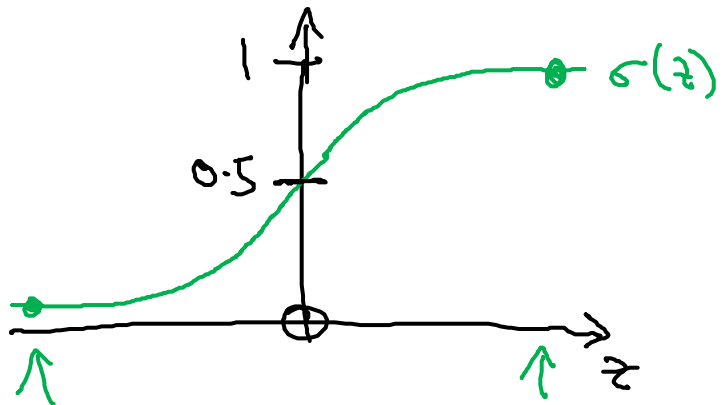
Logistic Regression

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

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

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

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

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

$\sigma(z)$

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

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

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

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

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

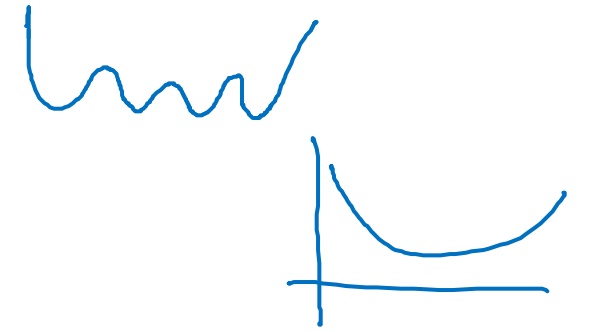# Basics of Neural Network Programming

Logistic Regression cost function

deeplearning.ai

Logistic Regression cost function

$\rightarrow \hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, 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 $\hat{y}^{(i)} \approx y^{(i)}$.

$\begin{array}{l} x^{(i)} \\ y^{(i)} \\ z^{(i)} \end{array}$ $\begin{array}{l} i\text{-th} \\ \text{example.} \end{array}$

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

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

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: $J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})]$
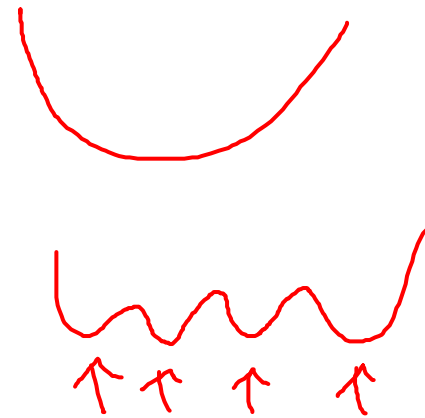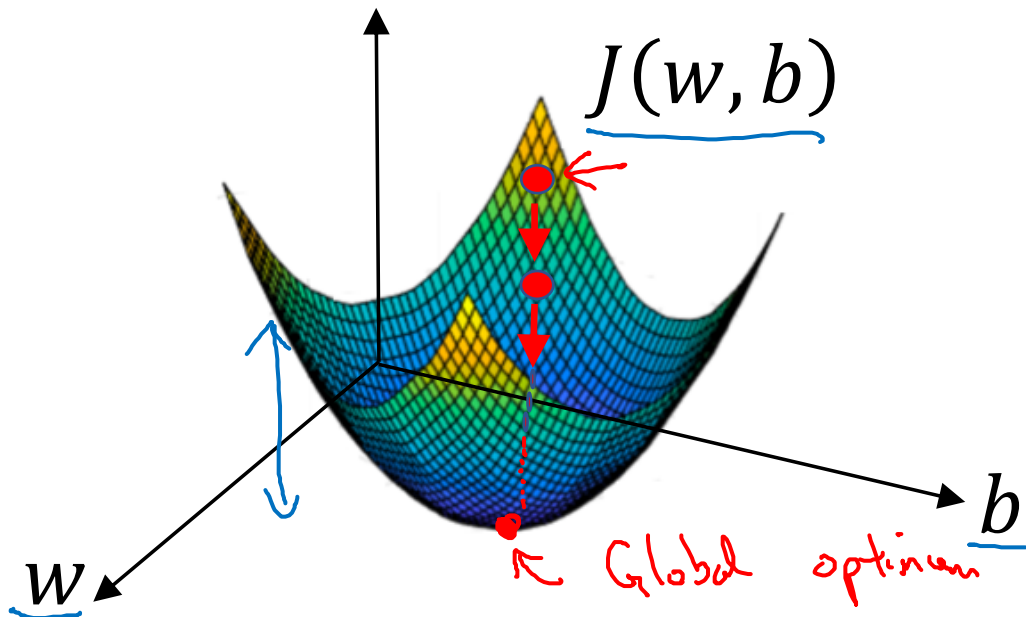
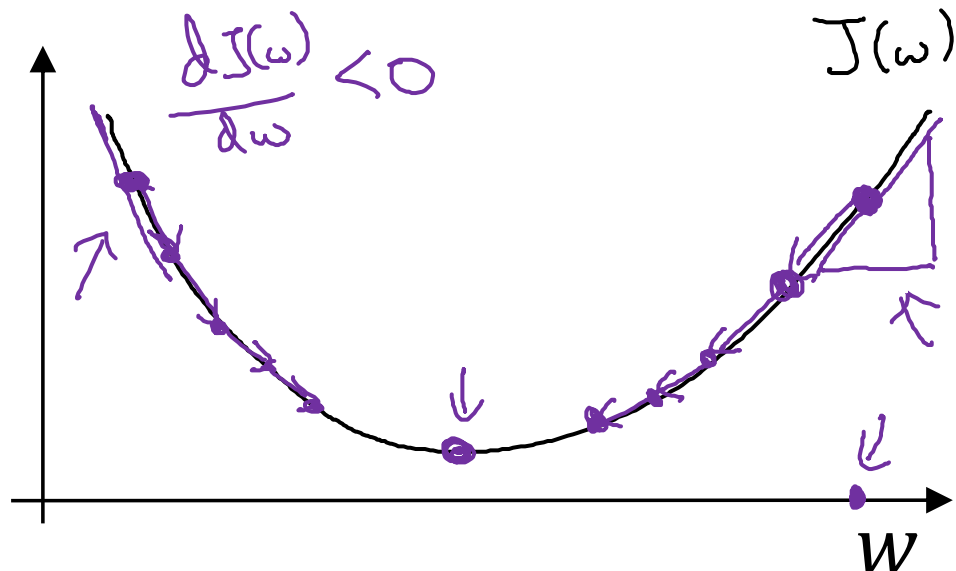# Basics of Neural Network Programming

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 $J(w,b)$

$J(w,b)$

← Global optimum

b

w

Gradient Descent

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

$$\text{Repeat } \{$$

$$w := w - \alpha \frac{dJ(w)}{dw} \qquad \text{learning rate}$$

$$\}$$

$$w := w - \alpha \, dw \qquad \text{``}dw\text{''}$$

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

$$J(w,b)$$

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

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

$$\partial \qquad \text{``partial derivative''}$$

$$\partial \qquad J$$

$$dw$$

$$db$$

Andrew Ng

Basics of Neural Network Programming

Derivatives

deeplearning.ai

Intuition about derivatives

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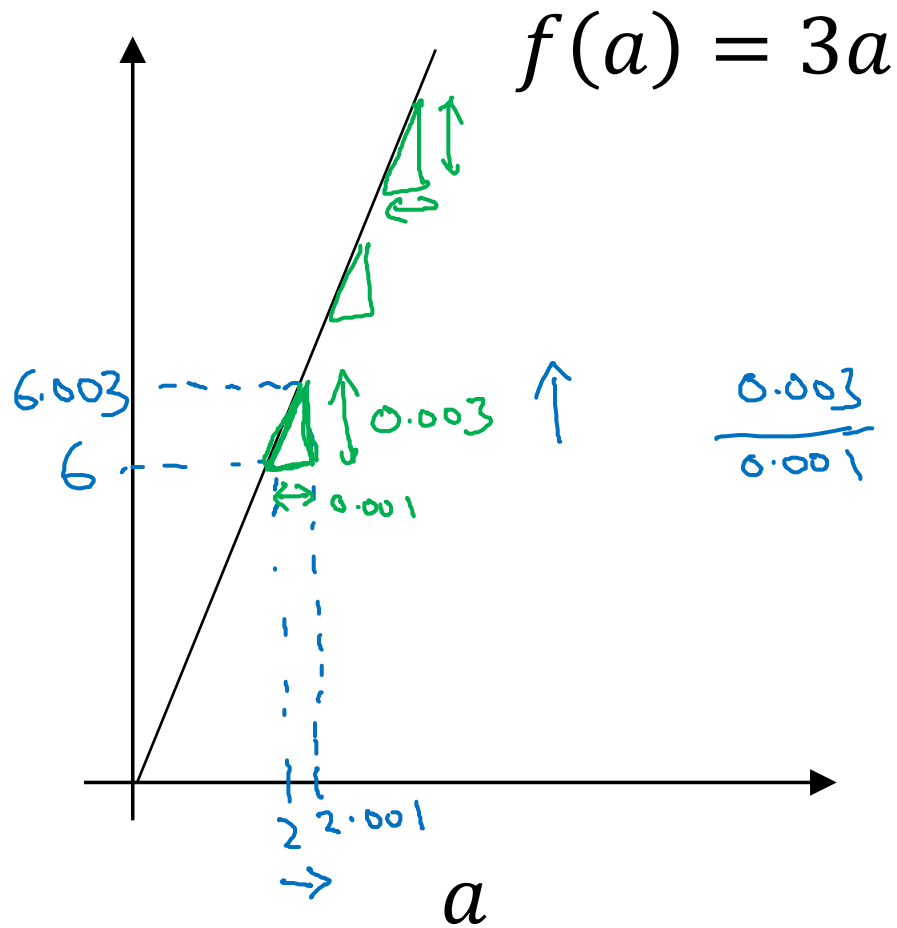
$a = 2 \qquad f(a) = 6$

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

slope (derivative) of $f(a)$

at $a = 2$ is 3

$\dfrac{0.003}{0.001} \quad \dfrac{\text{height}}{\text{width}}$

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

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

slope at $a = 5$ is also 3

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

0.001

0.00000001

0.0000000001

Andrew Ng

Basics of Neural Network Programming

More derivatives examples

deeplearning.ai

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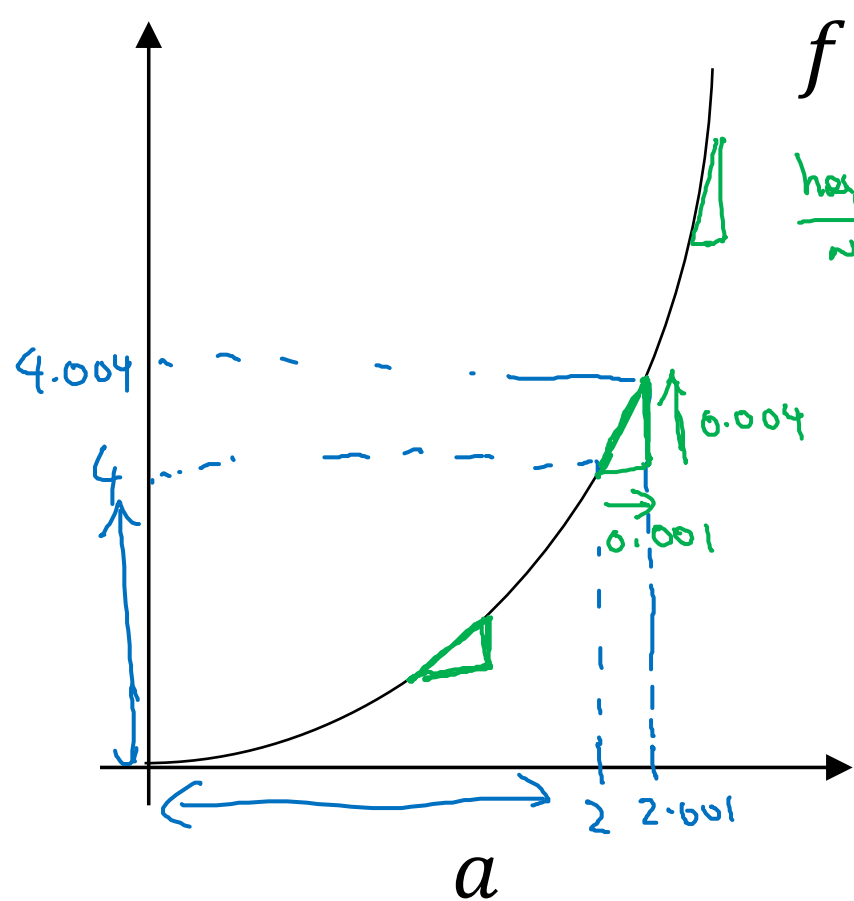
$$f(a) = a^2$$

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

0.001

$(2a) \times 0.001$

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

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

$(4.004 \boxed{004})$

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

$a = 2$ is   4.

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

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

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

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

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

0.001

$0.00000\ldots01$

More derivative examples

$f(a) = a^2$

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

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

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

$f(a) = a^3$

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

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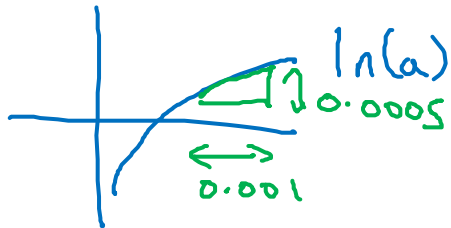
$a = 2.001$ $f(a) \approx 8.012$

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

$\ln(a)$

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

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

$\ln(a)$ 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

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

$\underbrace{}_{u}$

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

$u = bc$

$v = a + u$

$J = 3v$

$a = 5$

$b = 3$ 6

$\boxed{u = bc}$

$c = 2$

11

$\boxed{V = a + u}$

33

$\boxed{J = 3v}$
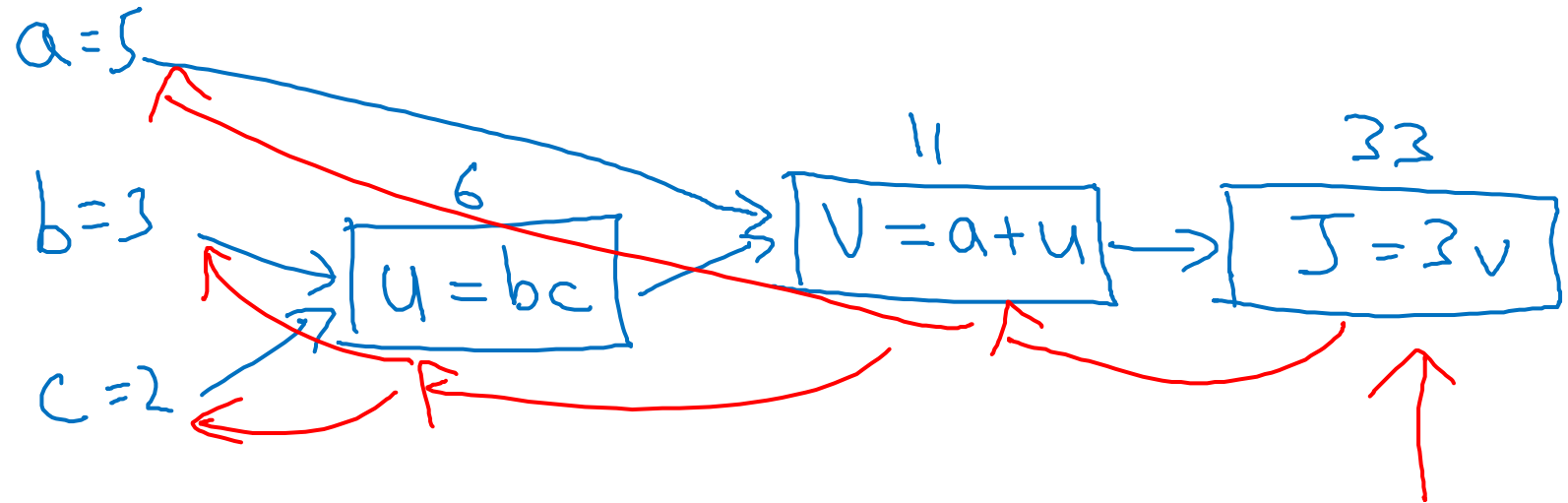
Basics of Neural Network Programming

Derivatives with a Computation Graph

deeplearning.ai

Computing derivatives

$a = 5$

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

$b = 3$

6

$u=bc$

$c = 2$

11

$\boxed{v = a + u}$

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

33

$\boxed{J = 3v}$

$J = 3v$

$v = 11 \;\Rightarrow\; 11.001$

$J = 33 \;\Rightarrow\; 33.003$

$a \rightarrow v \rightarrow J$

$f(a) = 3a$

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

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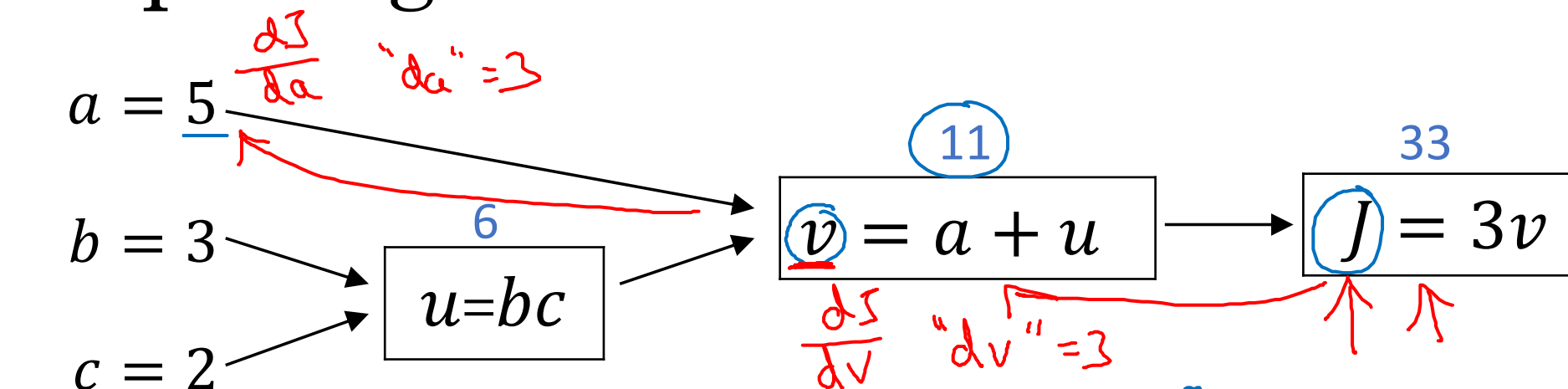
$\dfrac{dJ}{dv} = 3$

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

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

3×1

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

$a = 5 \;\Rightarrow\; 5.001$

$\Rightarrow v = 11 \;\Rightarrow\; 11.001$

$J = 33 \;\Rightarrow\; 33.003$

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

$\dfrac{dJ\,dvar}{}$

"dvar"

Andrew Ng

Computing derivatives

$$\frac{dJ}{da}$$

$a = 5$

$\frac{da}{} = 3$

$b = 3$

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

$c = 2$

$\frac{dc}{} = 9$

$u = bc$

$\frac{du}{} = 3$

$v = a + u$ (11)

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

$J = 3v$ (33)
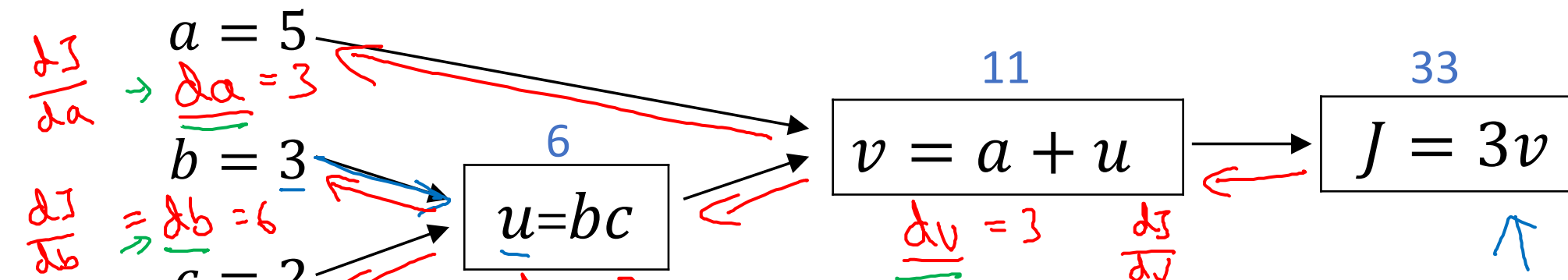
$\frac{dJ}{du} = 3 = \frac{dJ}{dv} \cdot \frac{dv}{du}$
$\qquad\qquad\quad \underset{3}{} \quad \underset{1}{}$

$\frac{dJ}{db} = \boxed{\frac{dJ}{du}} \cdot \frac{du}{db} = 6$
$\qquad\qquad \underset{\to 3}{} \quad \underset{=2}{}$

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

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

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

$v = 11.002$
$J = 3v$

Andrew Ng

Basics of Neural
Network Programming

Logistic Regression
Gradient descent

deeplearning.ai

Logistic regression recap

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

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

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

Logistic regression derivatives

x_1
w_1
x_2
w_2
b

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

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

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

$$= a - y$$

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

$$a(1-a)$$

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

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

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

Andrew Ng

Basics of Neural Network Programming

Gradient descent on m examples

deeplearning.ai

Logistic regression on m examples

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

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

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

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

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

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

Logistic regression on m examples

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

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

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

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

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

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

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

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

dw_3
$\vdots$
dw_n

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

$J /= m$ $\leftarrow$

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

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

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

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

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

Vectorization

Andrew Ng

Basics of Neural Network Programming

Vectorization

deeplearning.ai

What is vectorization?

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

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

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

Non-vectorized:

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

$$z += b$$

Vectorized

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

→ GPU } SIMD — single instruction
→ CPU } multiple data.

Andrew Ng

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]
```

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

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

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

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

→ for i = 1 to n:

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

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

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

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

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

$dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$ $n_x = 2$ $dw \mathrel{+}= x^{(i)} dz^{(i)}$

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

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

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

$dw \mathrel{/}= m.$

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

Vectorizing Logistic Regression

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

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

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

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

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

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

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

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

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

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

"Broadcasting"

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

Andrew Ng

Basics of Neural Network Programming

Vectorizing Logistic Regression's Gradient Computation

deeplearning.ai

Vectorizing Logistic Regression

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

$$\boxed{dz} = [dz^{(1)} \ dz^{(2)} \cdots dz^{(m)}] \qquad \leftarrow$$
$$\underset{1 \times m}{}$$

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

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

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

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

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

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

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

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

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

Andrew Ng

Implementing Logistic Regression

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

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

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

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

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

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

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

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

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

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

$$A = \sigma(z)$$

$$dz = A - Y$$

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

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

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

Andrew Ng
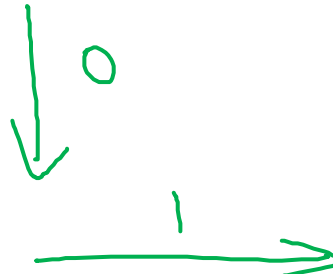
Broadcasting example

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))
```

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

Broadcasting example

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

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

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

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

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

$\downarrow$

(m,n)

General Principle

(m, n)

$\underline{matrix}$

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

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

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

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

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

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

Matlab/Octave: $\underline{bsxfun}$

Logistic regression cost function

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

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

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

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

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

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

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

Cost on m examples

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

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

Maximum likelihood estimate $\nwarrow$

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

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

Andrew Ng