Copyright Notice

These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode



deeplearning.ai

Basics of Neural Network Programming

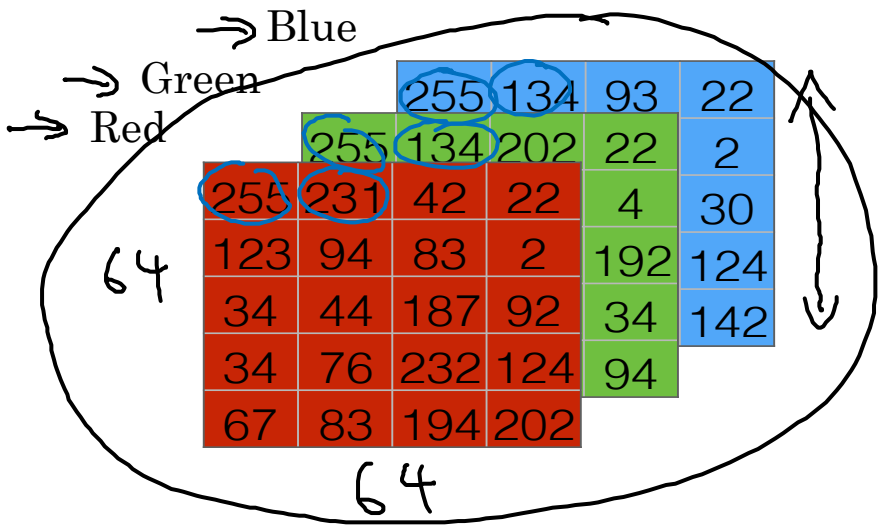
Binary Classification

Binary Classification



1 (cat) vs 0 (non cat)

y



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

$64 \times 64 \times 3 = 12288$

$n = n_x = 12288$

$x \longrightarrow y$

Notation

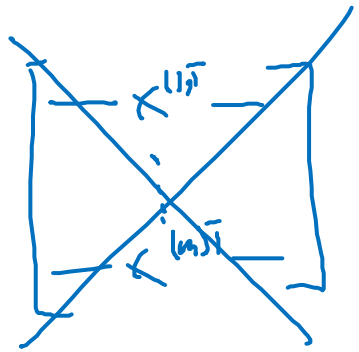
(x, y) $\quad x \in \mathbb{R}^{n_x}$, $y \in \{0, 1\}$

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

$m = m_{train}$ $\qquad$ m_{test} = #test examples.

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

n_x rows, m columns



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

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

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

$Y.shape = (1, m)$

Andrew Ng



deeplearning.ai

Basics of Neural Network Programming

Logistic Regression

Logistic Regression

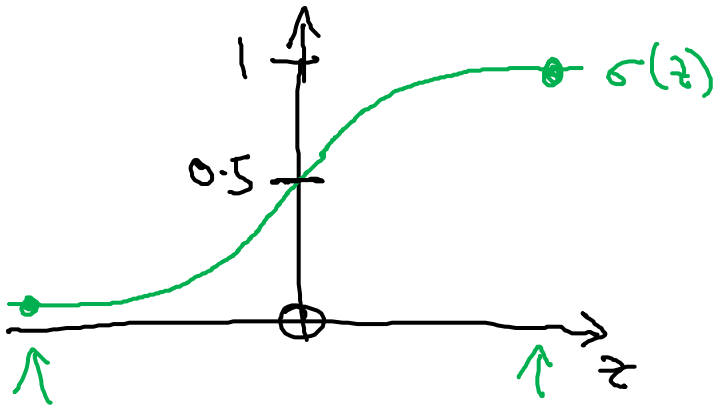
Given x, want $\hat{y} = P(y=1|x)$

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

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

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

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



deeplearning.ai

Basics of Neural Network Programming

Logistic Regression cost function

Logistic Regression cost function

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

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

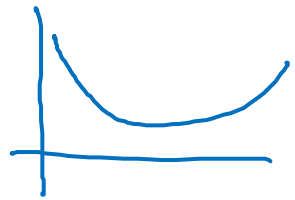
$x^{(i)}$
$y^{(i)}$
$z^{(i)}$
i-th example.

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

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



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



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

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

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

Andrew Ng



deeplearning.ai

Basics of Neural Network Programming

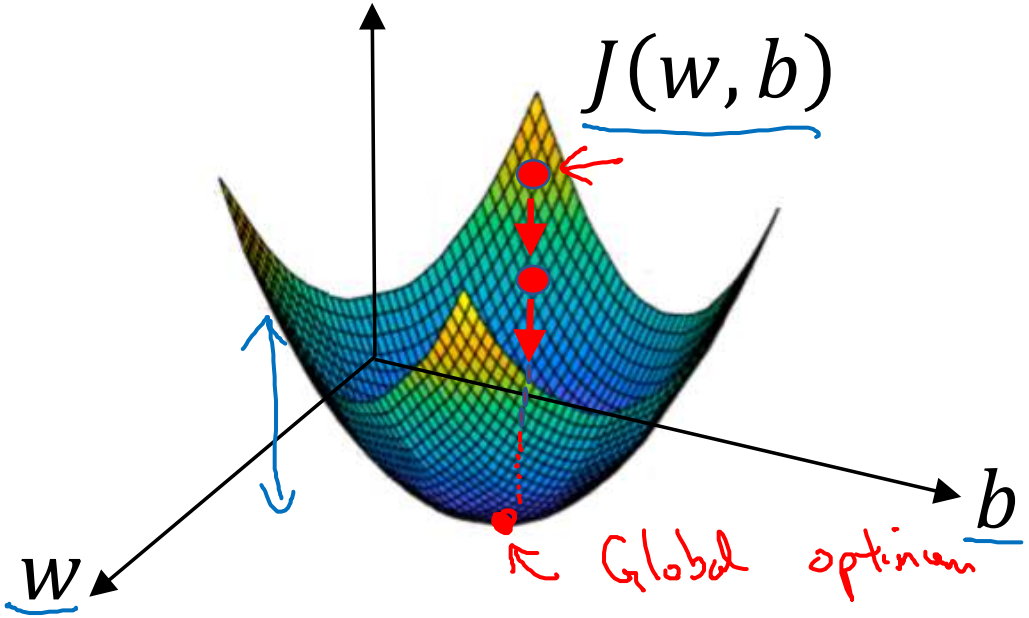
Gradient Descent

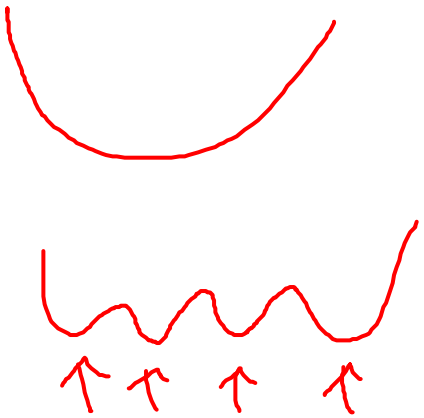
Gradient Descent

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

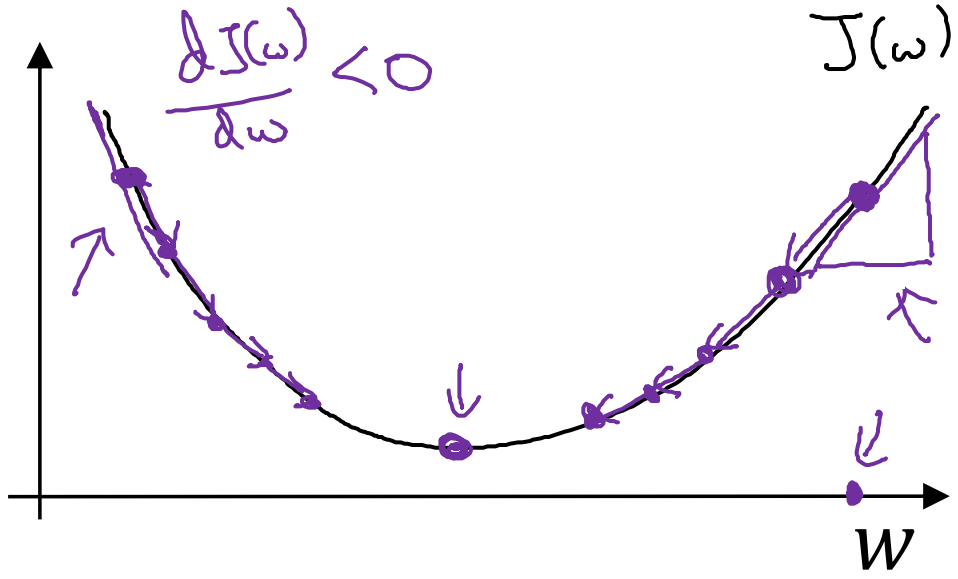
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

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





Gradient Descent



Repeat {

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

}

"dw"

$w := w - \alpha dw$

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

$J(w,b)$

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

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

∂ "partial derivative"

d ← J

Andrew Ng

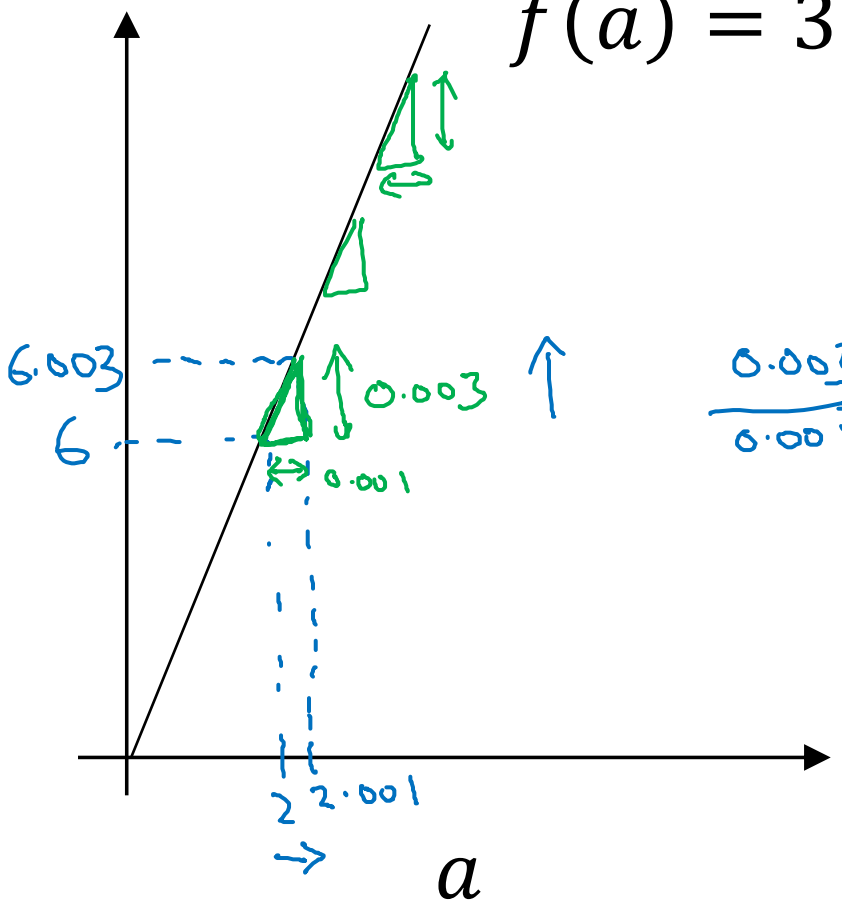


deeplearning.ai

Basics of Neural Network Programming

Derivatives

Intuition about derivatives



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

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

→ $a = 2$ $f(a) = 6$

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

→ $a = 5$ $f(a) = 15$

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

slope at $a=5$ is also 3

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

0.001
0.000000001
0.000000000001

Andrew Ng

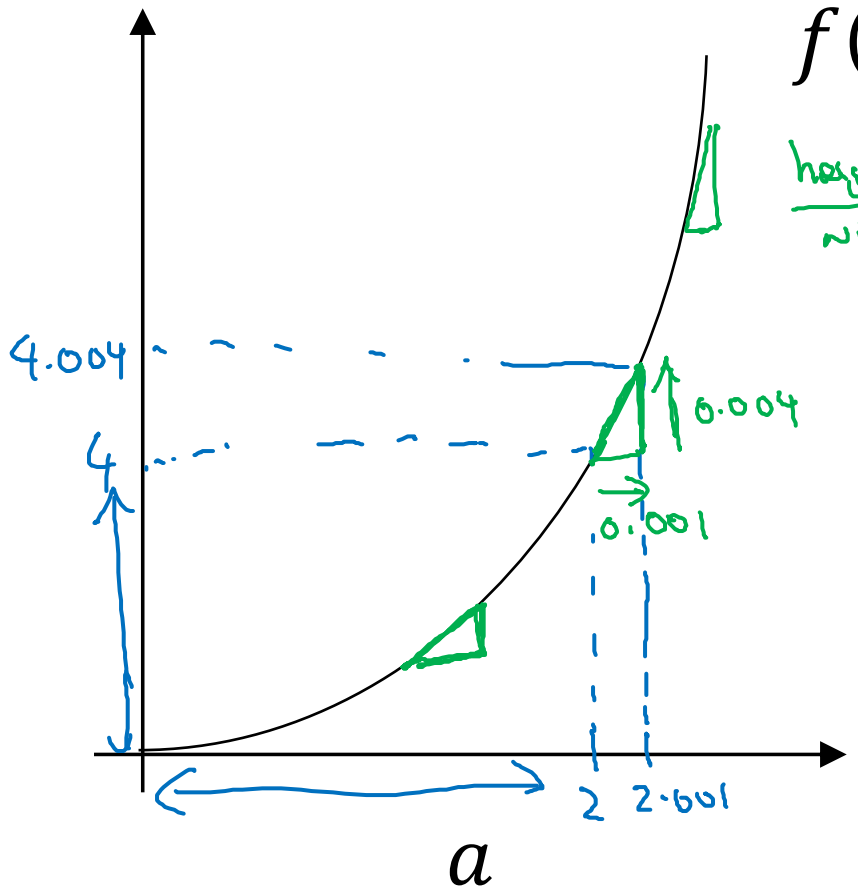


deeplearning.ai

Basics of Neural Network Programming

More derivatives examples

Intuition about derivatives



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

0.001

$(2a) \times 0.001$

0.001 ←

$0.000000\ldots01$ ←

$a = 2$

$a = 2.001$

$f(a) = 4$

$f(a) \approx 4.004$

$(4.004\boxed{001})$

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

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

$a = 5$

$a = 5.001$

$f(a) = 25$

$f(a) \approx 25.010$

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

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

Andrew Ng

More derivative examples

$f(a) = a^2$

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

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

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

$f(a) = a^3$

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

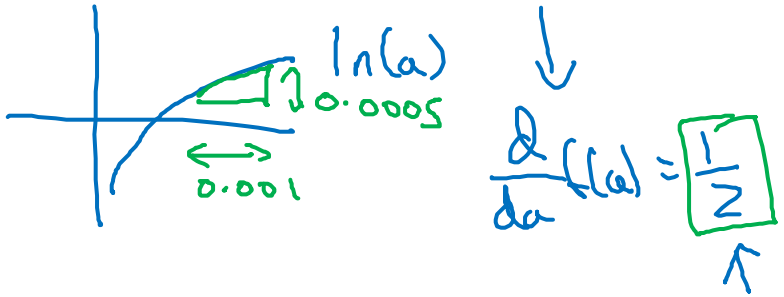
$a = 2$ $\quad f(a) = 8$

$a = 2.001$ $\quad f(a) \approx 8.012$

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

$\ln(a)$

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



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

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

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

0.0005

0.0005

Andrew Ng



deeplearning.ai

Basics of Neural Network Programming

Computation Graph

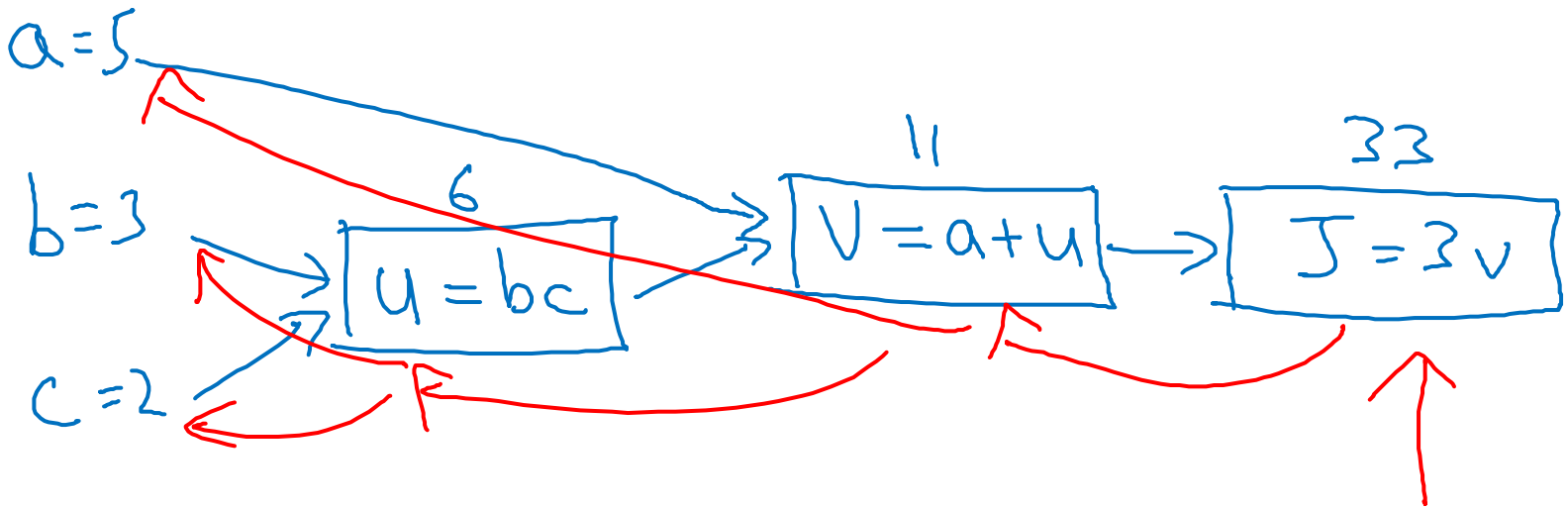
Computation Graph

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

$u = bc$

$v = a+u$

$J = 3v$



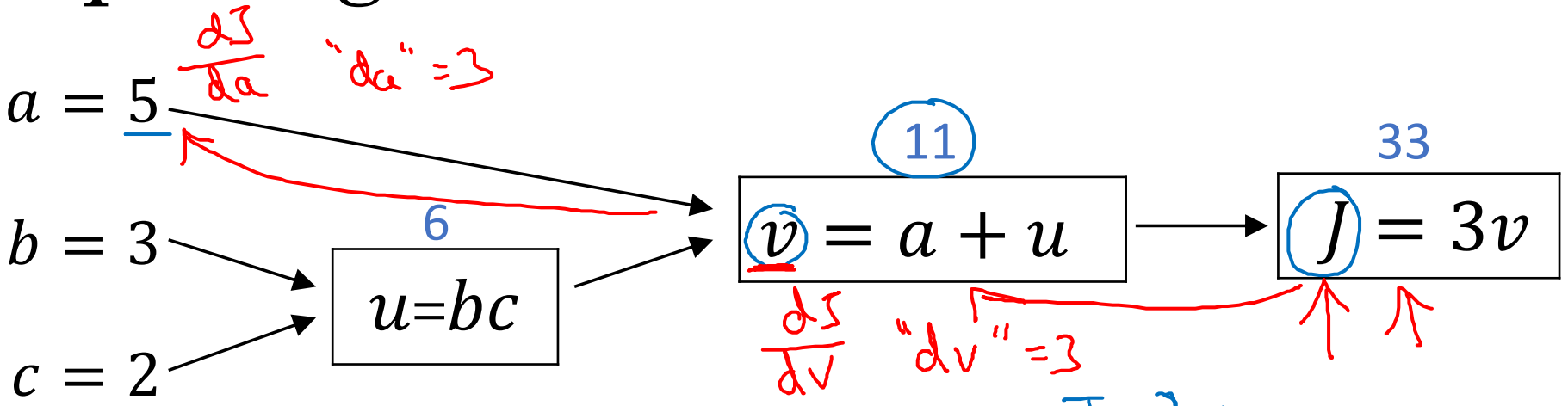


deeplearning.ai

Basics of Neural Network Programming

Derivatives with a Computation Graph

Computing derivatives



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

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

3×1

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

$a \to v \to J$

$J = 3v$

$v = 11 \to 11.001$

$J = 33 \to 33.003$

$a = 5 \to 5.001$

$\to v = 11 \to 11.001$

$J = 33 \to 33.003$

$f(a) = 3a$

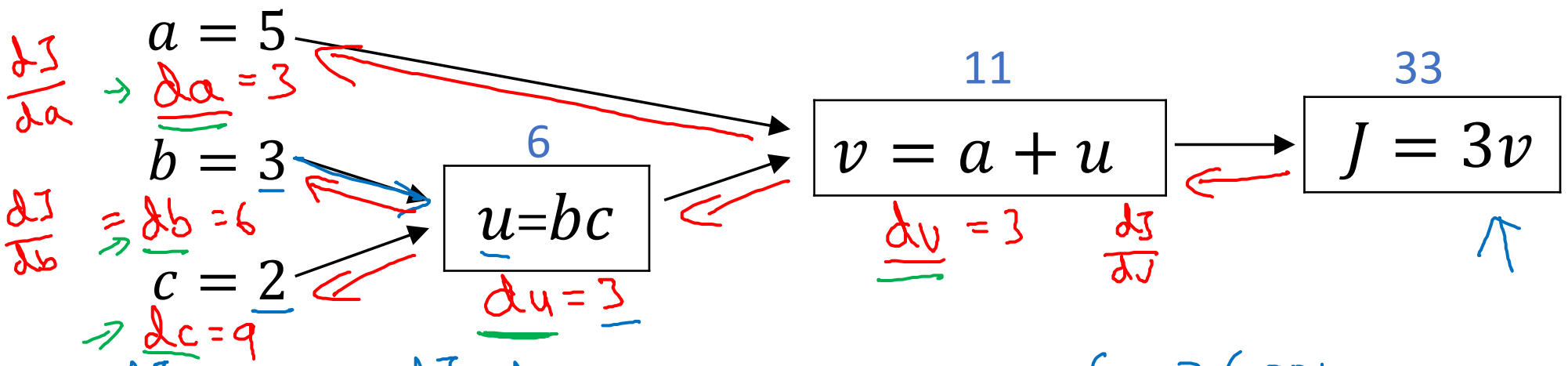
$\frac{df(a)}{da} = \frac{df}{da} = 3$

$J = 3v$

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

$\frac{d\,\text{FinalOutputVar}}{d\,\text{var}}$ → dJdvar → "dvar"

Computing derivatives



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

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

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

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

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

$c = 2$
$.006$

$V = 11.002$
$J = 3V$

Andrew Ng



deeplearning.ai

Basics of Neural Network Programming

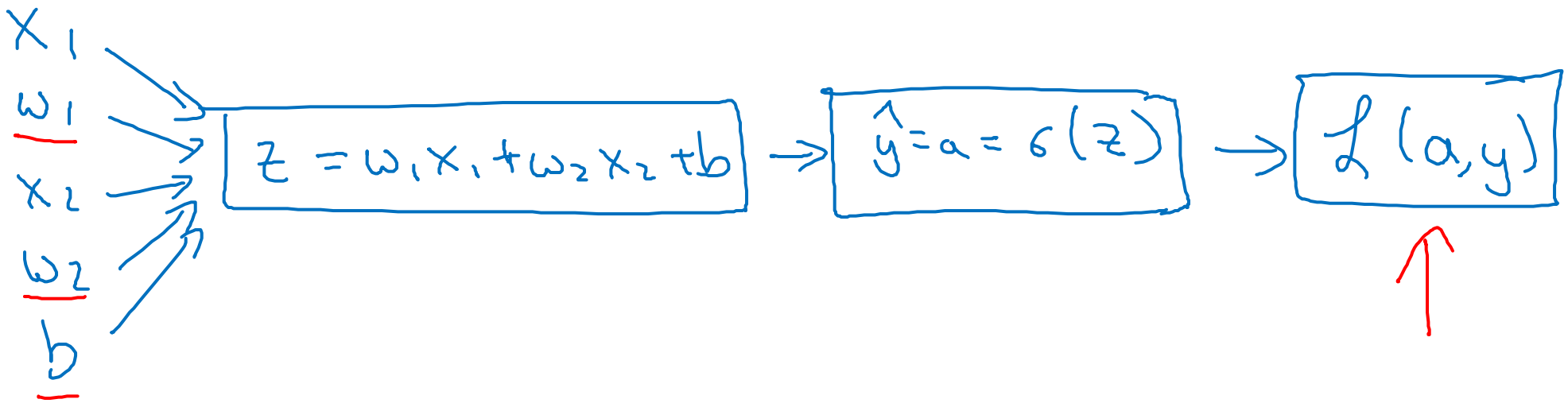
Logistic Regression Gradient descent

Logistic regression recap

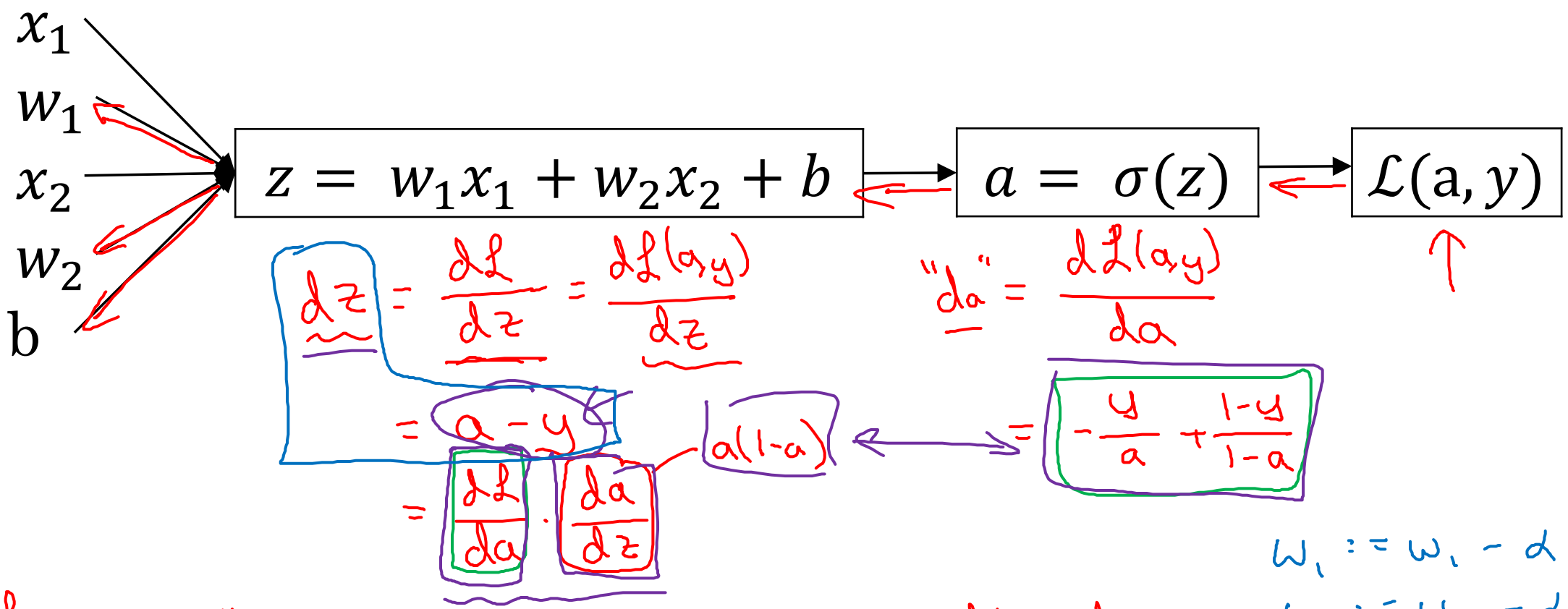
$$z = w^T x + b$$

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

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



Logistic regression derivatives



Andrew Ng



deeplearning.ai

Basics of Neural Network Programming

Gradient descent on m examples

Logistic regression on m examples

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

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

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

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

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

$dw_1^{(i)}$ – $(x^{(i)}, y^{(i)})$

Andrew Ng

Logistic regression on m examples

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

For $i=1$ to m

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

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

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

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

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

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

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

dw_3 ... dw_n

$n=2$

$J /= m$

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

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

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

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

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

Vectorization



deeplearning.ai

Basics of Neural Network Programming

Vectorization

What is vectorization?

$z = w^T x + b$

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

Non-vectorized:

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

Vectorized

```
z = np.dot(w,x) + b
```

$w^T x$

→ 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.

$u = Av$

$u_i = \sum_j A_{ij} v_j$

```
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

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

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

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 $i = 1 \ldots n_x$: $dw_i \mathrel{+}= \ldots$

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

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

$n_x = 2$

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

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

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

$dw /= m$.



deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

Vectorizing Logistic Regression

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

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

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

$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$ (n_x, m) $\mathbb{R}^{n_x \times m}$

$w^T = \begin{bmatrix} - & x^{(1)} & - \\ - & x^{(2)} & - \\ & \vdots & \\ - & x^{(n)} & - \end{bmatrix}$

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

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

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



deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression's Gradient Computation

Vectorizing Logistic Regression

$dz^{(1)} = a^{(1)} - y^{(1)}$ $dz^{(2)} = a^{(2)} - y^{(2)}$ ....

$dZ = [dz^{(1)}\ dz^{(2)} \cdots dz^{(m)}]$ ←

$1 \times m$

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

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

→ $dw = 0$

$dw += X^{(1)} dz^{(1)}$

$dw += X^{(2)} dz^{(2)}$

⋮

$dw /= m$

$db = 0$

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

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

⋮

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

$db /= m.$

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

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

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

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

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

$n \times 1$

Andrew Ng

Implementing Logistic Regression

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

for i = 1 to m:

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

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

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

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

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

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

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

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

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

$db = db/m$

for iter in range(1000):

$Z = w^T X + b$

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

$A = \sigma(Z)$

$dZ = A - Y$

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

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

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

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

Andrew Ng



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 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))
```

(3,4) / (1,4)

Broadcasting example

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

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

(m, n) (2,3) (1, n) ⇝ (m, n) (2,3)

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

(m, n) (m, 1) ⇝ (m, n)

General Principle

(m, n) matrix

$+ \; - \; * \; /$

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

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

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

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

$[1 \; 2 \; 3] + 100 = [101 \quad 102 \quad 103]$

Matlab/Octave: bsxfun



deeplearning.ai

Basics of Neural Network Programming

Explanation of logistic regression cost function (Optional)

Logistic regression cost function

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

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

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

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

Logistic regression cost function

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

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

$p(y|x)$

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

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

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

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

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

Andrew Ng

Cost on m examples

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

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

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

Maximum likelihood estimation

Cost: (minimize)

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