Basics of Neural Network Programming

Binary Classification

deeplearning.ai

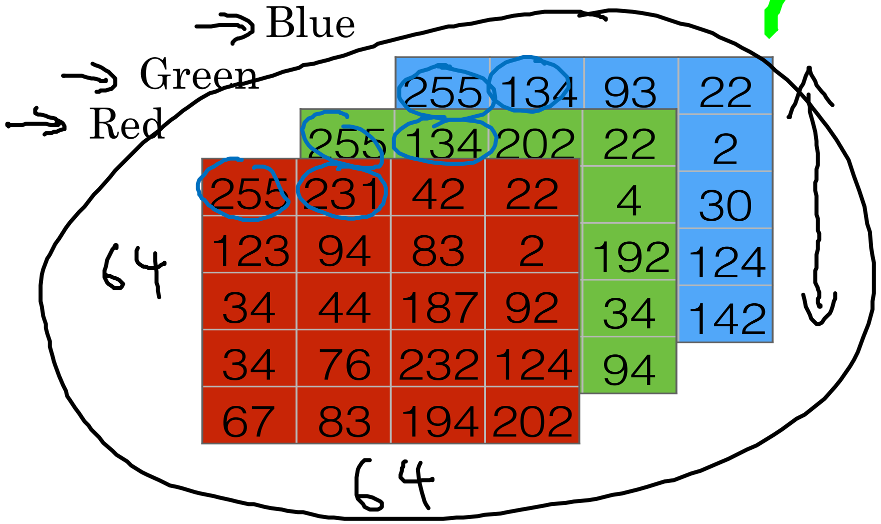
Binary Classification

64

64

$\rightarrow$ 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 is nx dimensional

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

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

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

$$X = \underbrace{\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \ldots & x^{(m)} \\ | & | & & | \end{bmatrix}}_{m} \Big\updownarrow n_x \qquad \cancel{\begin{bmatrix} — & x^{(1)T} & — \\ & \vdots & \\ — & x^{(m)T} & — \end{bmatrix}}$$

$$Y = [y^{(1)} \; y^{(2)} \; \ldots, y^{(m)}]$$

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

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

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

Andrew Ng

Basics of Neural Network Programming

Logistic Regression

deeplearning.ai

Logistic Regression

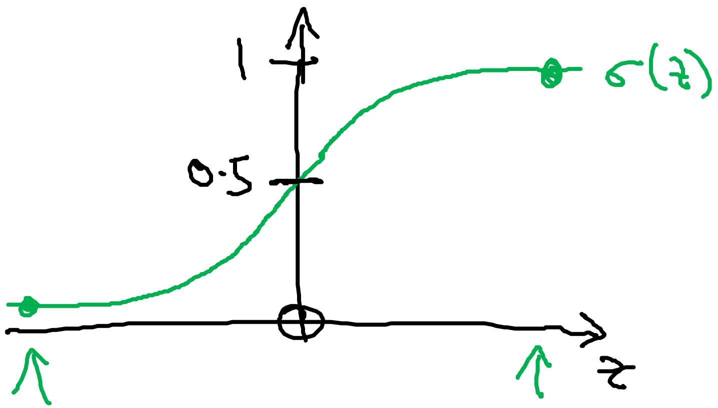
Given x, want $\hat{y} = P(y=1 \mid x)$

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

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

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

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



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

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

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \quad \begin{array}{l} \} \, b \leftarrow \\ \\ \} \, w \leftarrow \end{array}$$

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

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

If z large negative number

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

deeplearning.ai

Basics of Neural Network Programming

Logistic Regression cost function

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 $\hat{y}^{(i)} \approx y^{(i)}$.

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

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

we don't use this squared error loss function in logistic as it awill

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

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

cost is for entire training set , while loss is with individual trainign examolle

Andrew Ng

Basics of Neural Network Programming

Gradient Descent

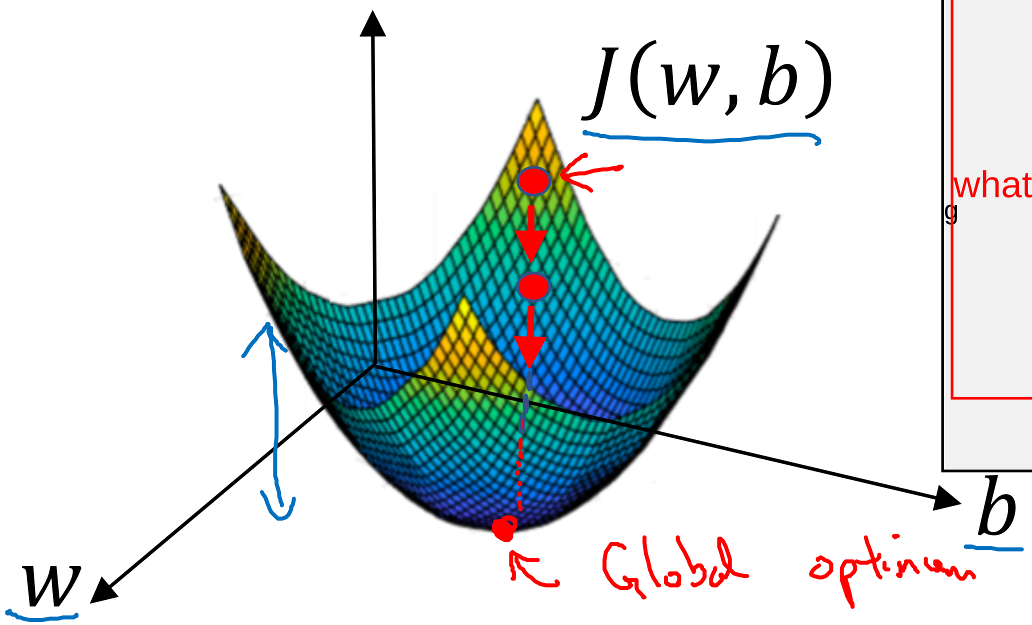
deeplearning.ai

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 $J(w,b)$

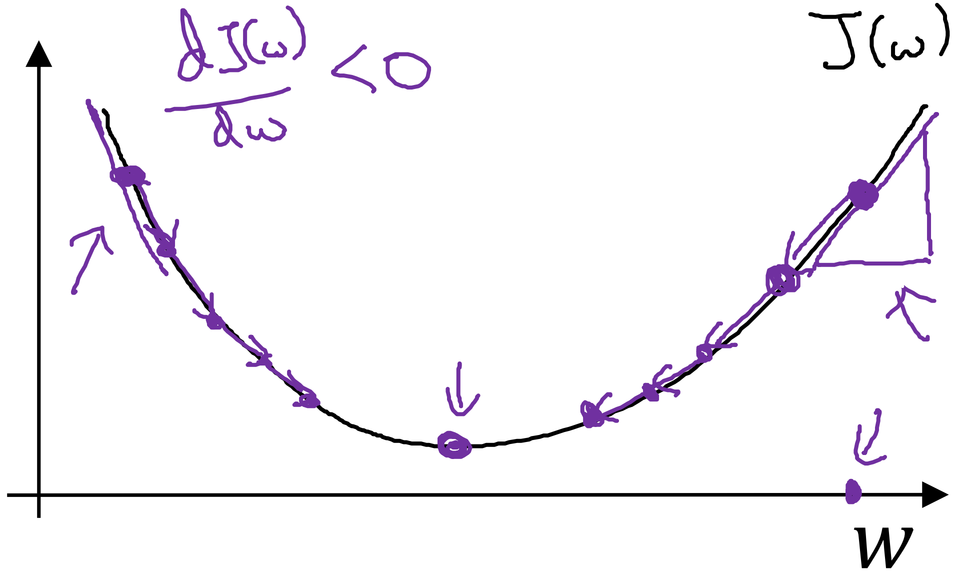


what gradient descent does is it starts at that initial point and then takes a step in the ste

g

Andrew Ng

Gradient Descent



learning rate

Repeat {

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

}

"dw"

$w := w - \alpha dw$

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

$J(w, b)$

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

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

∂ "partial derivative"

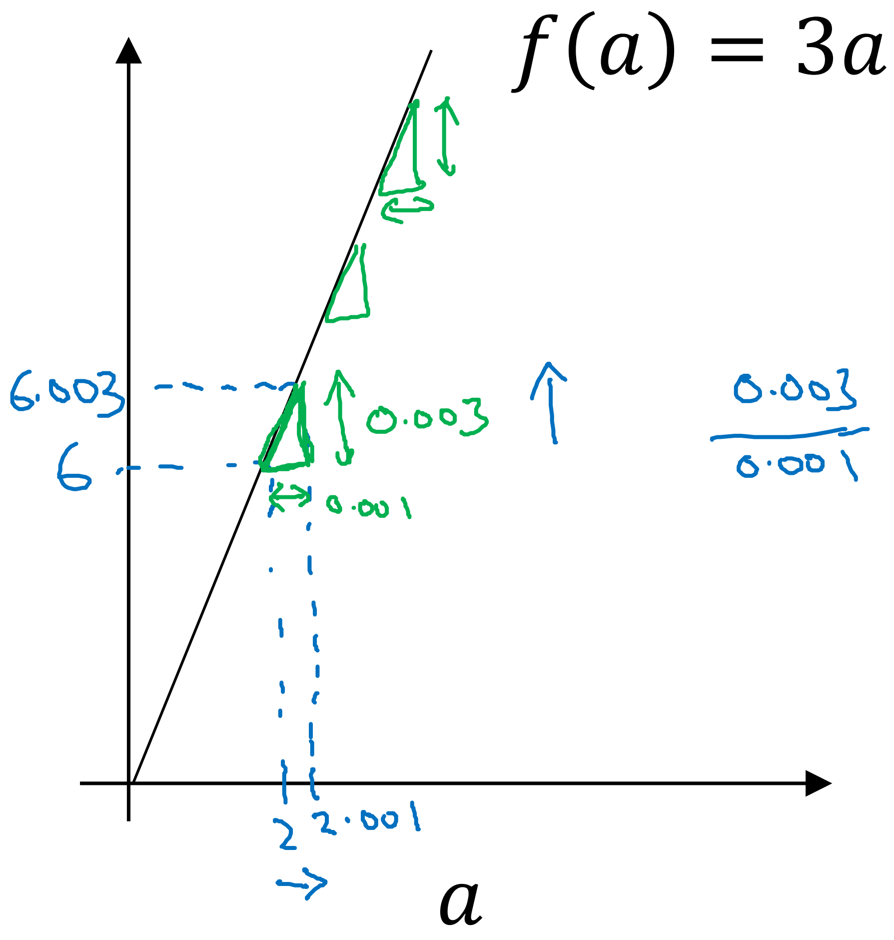
d J

Basics of Neural Network Programming

Derivatives

deeplearning.ai

Intuition about derivatives



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

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

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

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

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

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

slope at $a = 5$ is also 3

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

0.001

0.00000001

0.000000000001

Andrew Ng

Basics of Neural Network Programming

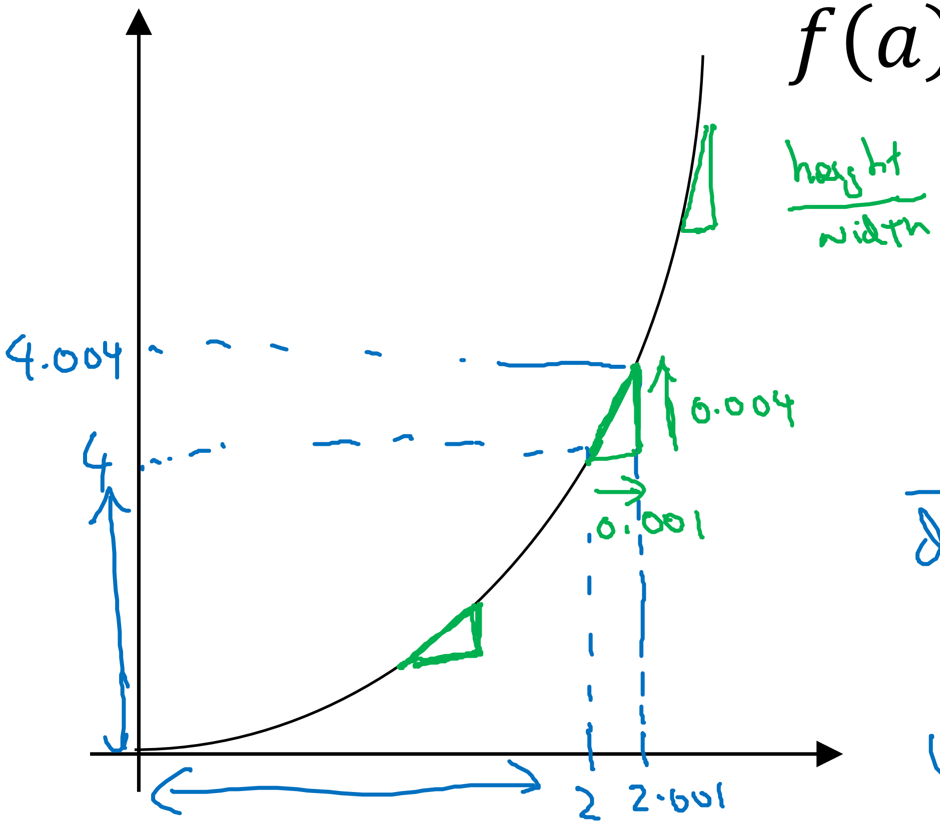
More derivatives examples

deeplearning.ai

Intuition about derivatives

0.001
$0.00000\ldots01$

$f(a) = a^2$



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

0.001

$(2a) \times 0.001$

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

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

(4.004001)

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

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

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

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

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

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

Andrew Ng

More derivative examples

$f(a) = a^2$

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

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

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

$f(a) = a^3$

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

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

$a = \underline{2.001}$ $\quad f(a) \approx 8.\underline{012}$

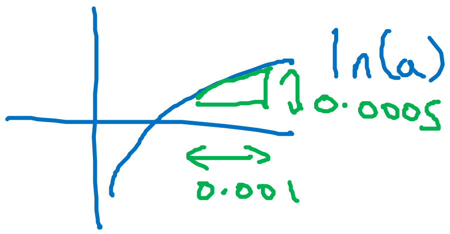
$f(a) = \log_e(a)$

$\ln(a)$

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

$\downarrow$

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



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

$a = 2.\underline{001}$ $\quad \underline{f(a)} \approx 0.69365$

$\downarrow$

$0.0005 \longleftrightarrow \underline{0.0005}$

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Computation Graph

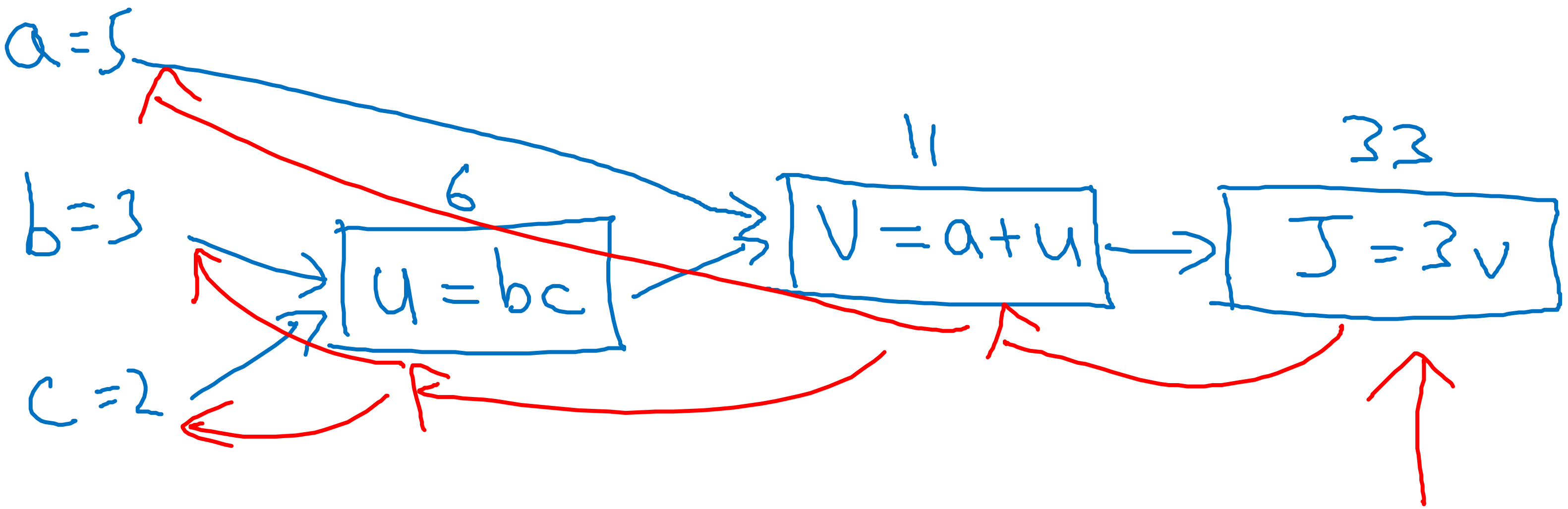
Computation Graph

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

$u = bc$

$v = a + u$

$J = 3v$



deeplearning.ai

Basics of Neural Network Programming

Derivatives with a Computation Graph

Computing derivatives

$a = 5$ $\frac{dJ}{da}$ "da" = 3

$b = 3$

$c = 2$

$u = bc$ 6

$v = a + u$ 11

$J = 3v$ 33

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

$J = 3v$

$v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

$f(a) = 3a$

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

$J = 3v$

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

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

$a \rightarrow v \rightarrow J$

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

3×1

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

$a = 5 \rightarrow 5.001$

$\rightarrow v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

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

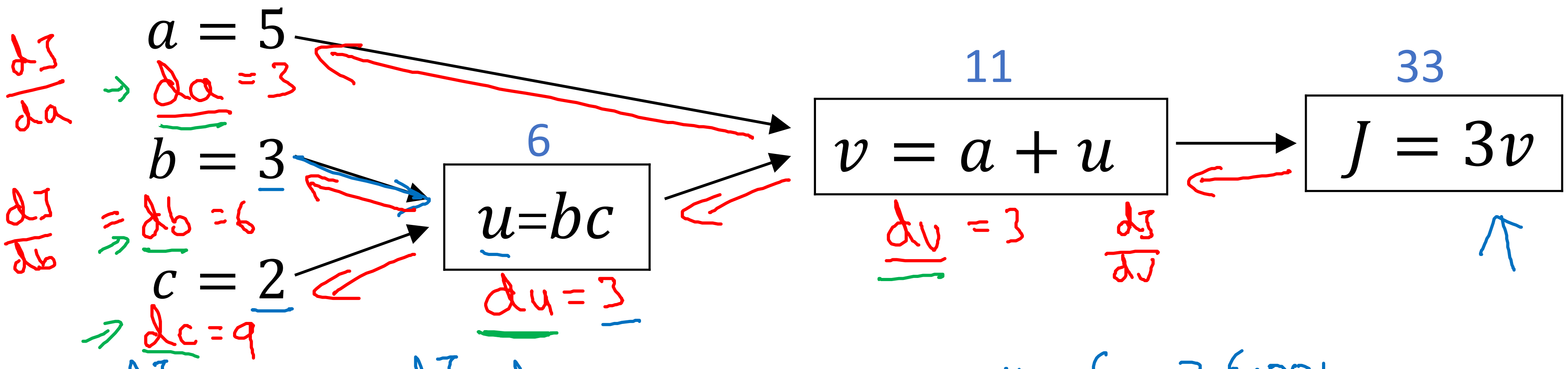
dJdvar

$\boxed{\text{"dvar"}}$

In code we will write only this dvar, when we me

Andrew Ng

Computing derivatives



$$\frac{dJ}{du} = 3 = \underbrace{\frac{dJ}{dv}}_{3} \cdot \underbrace{\frac{dv}{du}}_{1}$$

$$\frac{dJ}{db} = \underbrace{\boxed{\frac{dJ}{du}}}_{\to 3} \cdot \underbrace{\frac{du}{db}}_{=2} = 6$$

$$\frac{dJ}{da} = \underbrace{\boxed{\frac{dJ}{du}}}_{3} \cdot \underbrace{\frac{du}{da}}_{3} = 9$$

3 × 3

$u = 6 \rightarrow 6.001$

$v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

$b = 3 \rightarrow 3.001$

$u = b \cdot c = 6 \rightarrow 6.002$ $\qquad c = 2$

$J = 33.006$ $\qquad .006$

$v = 11.002$

$J = 3v$

deeplearning.ai

Basics of Neural Network Programming

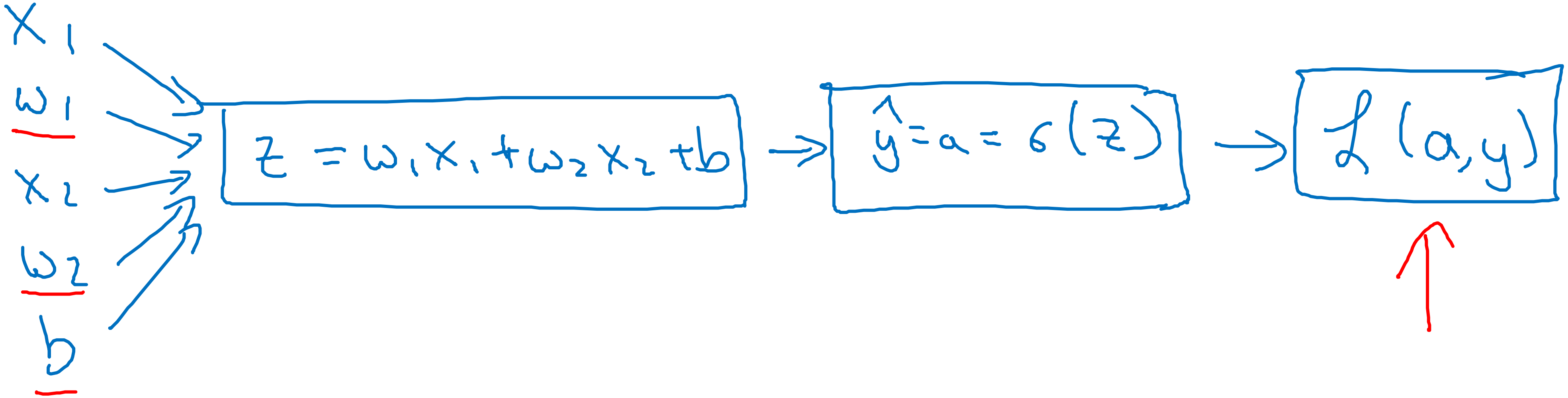
Logistic Regression Gradient descent

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

$$
\begin{tikzcd}
x_1 \arrow[drr] & & & & \\
w_1 \arrow[dr] & & & & \\
x_2 \arrow[r] & \boxed{z = w_1x_1 + w_2x_2 + b} \arrow[r] & \boxed{a = \sigma(z)} \arrow[r] & \boxed{\mathcal{L}(a, y)} \\
w_2 \arrow[ur] & & & & \\
b \arrow[uur] & & & &
\end{tikzcd}
$$

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

$$= a - y$$

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

$$\frac{da}{dz} = a(1-a)$$

$$\text{"}da\text{"} = \frac{d\mathcal{L}(a,y)}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \text{"}dw_1\text{"} = x_1 \cdot dz. \qquad dw_2 = x_2 \cdot dz. \qquad db = dz.$$

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

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

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

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Gradient descent on m examples

Logistic regression on m examples

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

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

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

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

$$\frac{\partial}{\partial w_1} J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{\frac{\partial}{\partial w_1}\mathcal{L}(a^{(i)}, y^{(i)})}_{dw_1^{(i)}}$$

$dw_1^{(i)}$ — $(x^{(i)}, y^{(i)})$

Logistic regression on m examples

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

For $i = 1$ to m

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

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

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

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

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

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

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

$n = 2$

dw_3
$\vdots$
dw_n

$J /= m$

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

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

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

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

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

Vectorization

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Vectorization

What is vectorization?

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

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

Non-vectorized:

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

Vectorized

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

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

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)
```

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
```

Andrew Ng

Logistic regression derivatives

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

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

$\to$ 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 $j = 1 \ldots n_x$

$dw_j \mathrel{+}= \ldots$

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

$n_x = 2$

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

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

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

$dw \mathrel{/}= m.$

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

Vectorizing Logistic Regression

$z^{(1)} = w^T x^{(1)} + b$ $\qquad$ $z^{(2)} = w^T x^{(2)} + b$ $\qquad$ $z^{(3)} = w^T x^{(3)} + b$

$a^{(1)} = \sigma(z^{(1)})$ $\qquad$ $a^{(2)} = \sigma(z^{(2)})$ $\qquad$ $a^{(3)} = \sigma(z^{(3)})$

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

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

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

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

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

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression's Gradient Computation

Vectorizing Logistic Regression

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

$$\boxed{dZ = [dz^{(1)} \; dz^{(2)} \; \dots \; dz^{(m)}]} \leftarrow$$

$1 \times m$

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

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

$\rightarrow dw = 0$

$dw \mathrel{+}= x^{(1)} dz^{(1)}$

$dw \mathrel{+}= x^{(2)} dz^{(2)}$

$\vdots$

$dw \mathrel{/}= m$

$\cancel{dw_1}$

$\cancel{dw_2}$

$\vdots$

$db = 0$

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

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

$\vdots$

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

$db \mathrel{/}= m.$

$$\boxed{db = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)} = \frac{1}{m} \text{np.sum}(dZ)}$$

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

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

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

$n \times 1$

Andrew Ng

Implementing Logistic Regression

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

$\boxed{\texttt{for i = 1 to m:}}$

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

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

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

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

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

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

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

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:

$$
\begin{array}{r} \\ \text{Carb} \\ \text{Protein} \\ \text{Fat} \end{array}
\begin{array}{c}
\begin{array}{cccc} \text{Apples} & \text{Beef} & \text{Eggs} & \text{Potatoes} \end{array} \\
\begin{bmatrix} 56.0 & 0.0 & 4.4 & 68.0 \\ 1.2 & 104.0 & 52.0 & 8.0 \\ 1.8 & 135.0 & 99.0 & 0.9 \end{bmatrix}
\end{array}
= \underset{(3,4)}{A}
$$

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

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

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

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

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

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

General Principle

$$\underset{\text{matrix}}{(m, n)} \quad \begin{matrix} + \\ - \\ * \\ / \end{matrix} \quad \begin{matrix} (1, n) \rightsquigarrow (m, n) \\ \\ (m, 1) \rightsquigarrow (m, n) \end{matrix}$$

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

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

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

Matlab/Octave: <u>bsxfun</u>

Basics of Neural Network Programming

deeplearning.ai

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

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

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

Logistic regression cost function

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

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

$p(y|x)$

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

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

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

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

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

Andrew Ng

Cost on m examples

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

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

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

Maximum likelihood estimation

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

(minimize)