Basics of Neural Network Programming

Binary Classification

deeplearning.ai

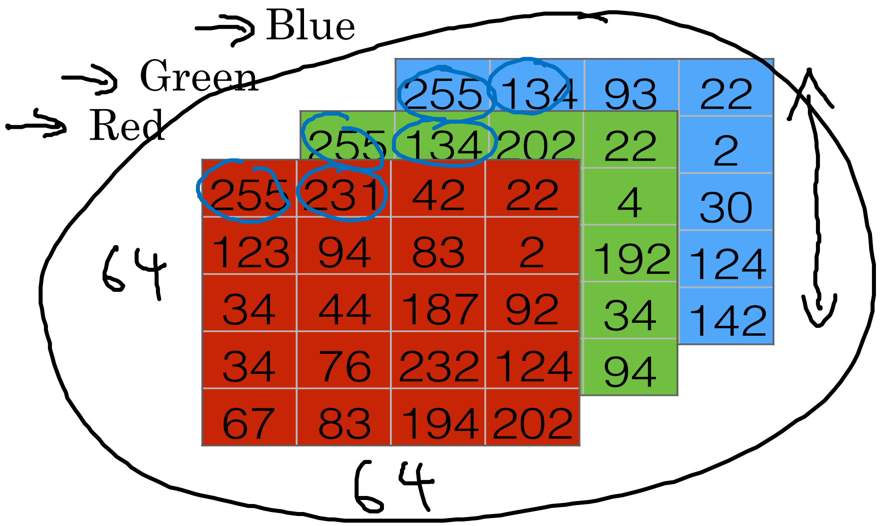
Binary Classification

64

64

$\underline{1}$ (cat) vs $\underline{0}$ (non cat)

y



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

$64 \times 64 \times 3 = 12288$

$n = n_x = 12288$

$x \longrightarrow y$

Andrew Ng

Notation

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

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

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

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

dont do it like this
as it complicates
implementation

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

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

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

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

.shape this is python command to find the shape of the matrix

Basics of Neural Network Programming

Logistic Regression

deeplearning.ai

Logistic Regression

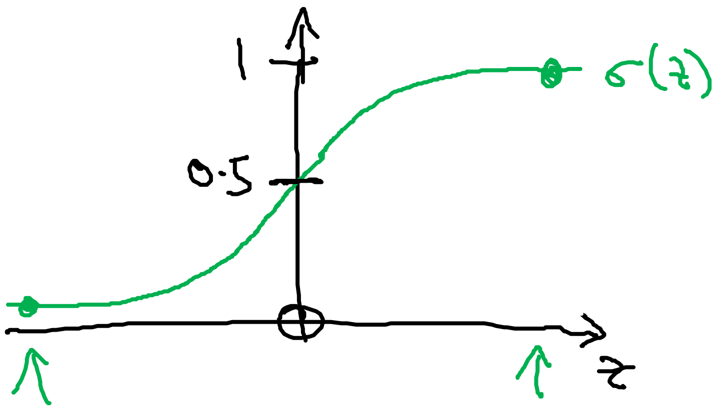
Given x, want $\hat{y} = P(y=1 \mid x)$

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

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

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

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



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

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

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

these are alternative denotation conventions that we are not gone follow here(he used this in other course i did)

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

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

If z large negative number

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

Basics of Neural Network Programming

Logistic Regression cost function

deeplearning.ai

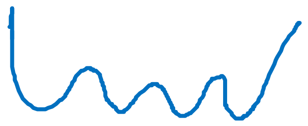
Logistic Regression cost function

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

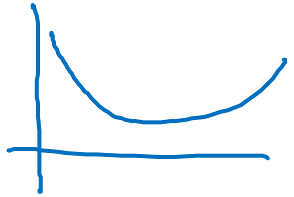
Given $\{(x^{(1)}, y^{(1)}),\dots,(x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

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

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



we dont use this since its not convex
and we have local minnimum



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

If $y = 1$: $\mathcal{L}(\hat{y}, y) = -\log \hat{y} \leftarrow$ Want $\log \hat{y}$ large, want $\hat{y}$ large. · y heat is sigmund fct

If $y = 0$: $\mathcal{L}(\hat{y}, y) = -\log(1-\hat{y}) \leftarrow$ 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

deeplearning.ai

Basics of Neural Network Programming

Logistic Regression cost function

Andrew Ng

Logistic Regression cost function

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

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

$x^{(i)}$ $\quad$ i-th

$y^{(i)}$ $\quad$ example.

$z^{(i)}$

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

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

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

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

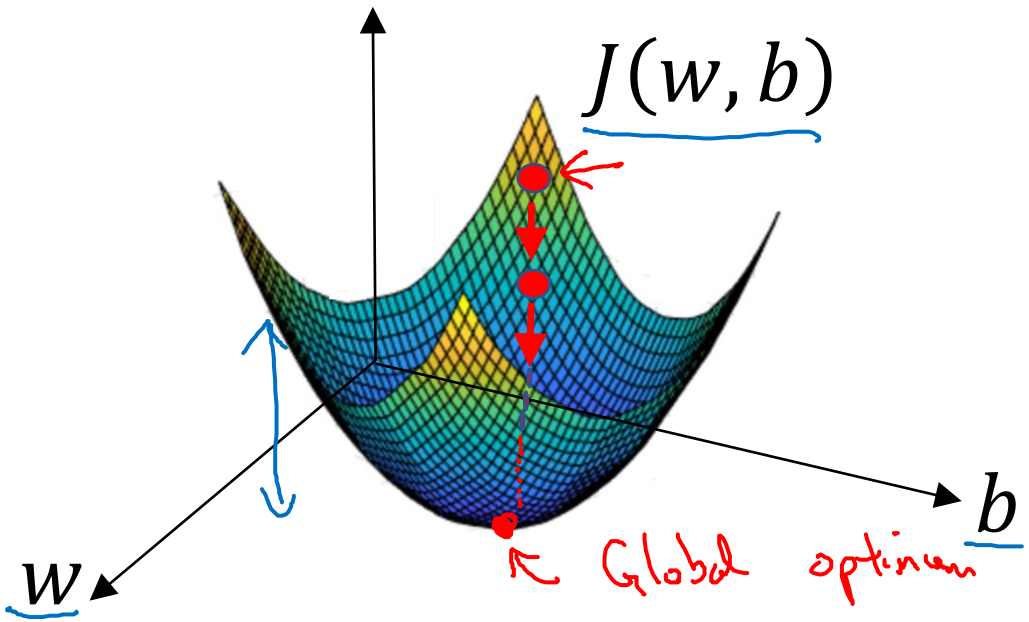
Gradient Descent

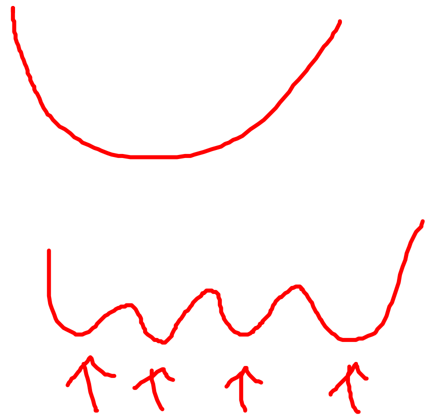
Gradient Descent

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

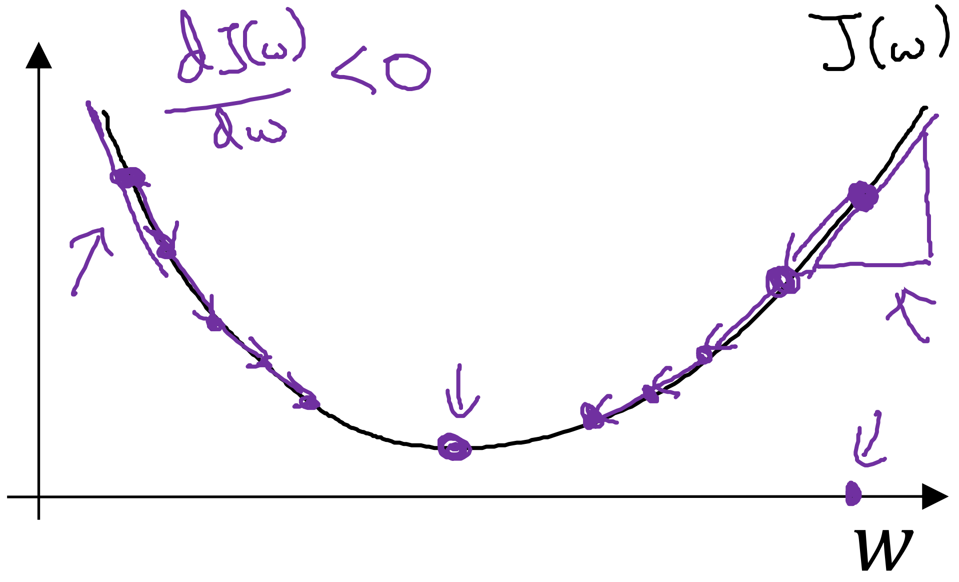
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\boxed{\hat{y}^{(i)}}, \boxed{y^{(i)}}) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

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





Gradient Descent



Repeat {

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

"dw"

}

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

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

We need to find these partial derivatives.

$J(w,b)$

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

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

∂ "partial derivative"

d J

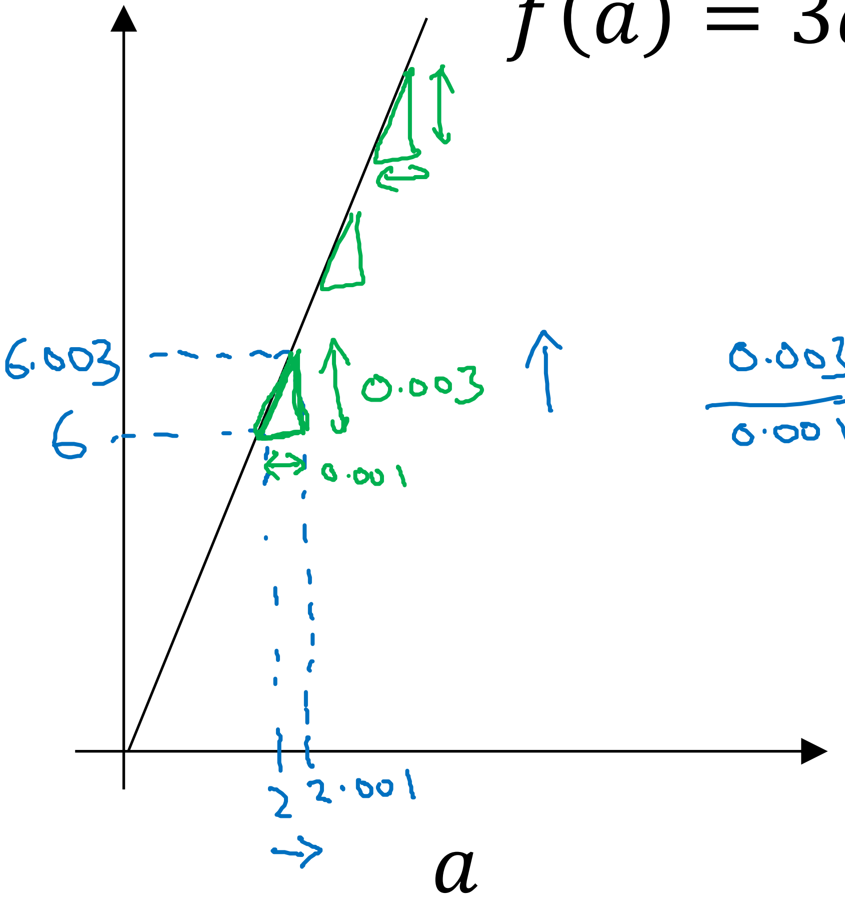
Basics of Neural Network Programming

Derivatives

deeplearning.ai

Intuition about derivatives

$$f(a) = 3a$$



$$\frac{0.003}{0.001} \qquad \frac{\text{height}}{\text{width}} \rightarrow \underline{\text{slope}}\ (\underline{\text{derivative}}) \text{ of } f(a) \text{ at } a=2 \text{ is } 3$$

→ $a = 2$ $\qquad f(a) = 6$

$a = 2.\underline{001}$ $\qquad f(a) = 6.\underline{003}$

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

→ $a = 5$ $\qquad f(a) = 15$

$a = 5.\underline{001}$ $\qquad f(a) = 15.\underline{003}$

slope at $a=5$ is also 3

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

0.001

0.00000001

0.0000000001

Andrew Ng

Basics of Neural Network Programming

deeplearning.ai

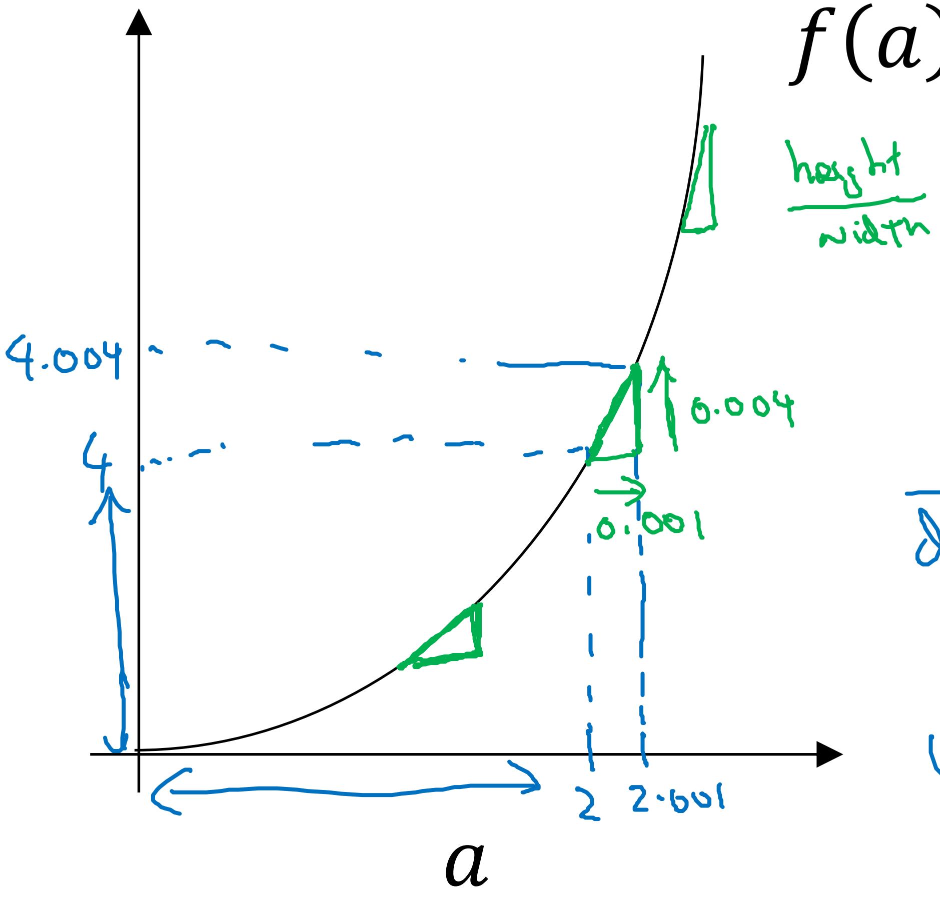
More derivatives examples

Intuition about derivatives

0.001

$0.00000\ldots01$

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



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

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

$(4.004\boxed{001})$

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

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

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

0.001

$(2a) \times 0.001$

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

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

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

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

Andrew Ng

More derivative examples

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

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

$f(a) = a^3$ $\qquad \frac{d}{da} f(a) = \underbrace{3a^2}_{3 \times 2^2 = 12}$ $\qquad a = 2 \quad f(a) = 8$

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

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

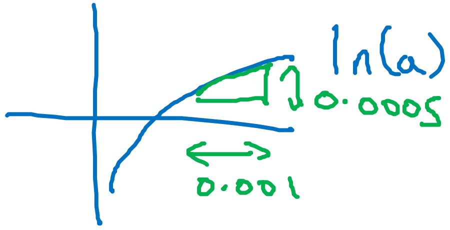
$\ln(a)$

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

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

$\downarrow$

$0.0005 \qquad \underline{0.0005}$



$\downarrow$

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

Andrew Ng

Basics of Neural Network Programming

Computation Graph

deeplearning.ai

Andrew Ng

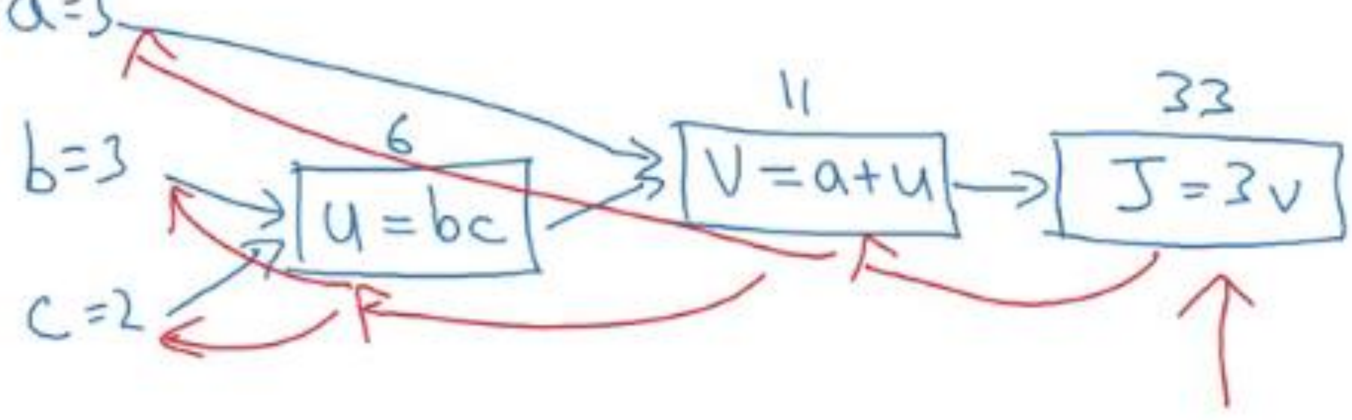
Computation Graph

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

$u = bc$

$v = a + u$

$J = 3v$



Andrew Ng

So, the computation graph comes in handy when there is some distinguished or some special output variable, such as J in this case, that you want to optimize. And in the case of a logistic regression, J is of course the cos function that we're trying to minimize. And what we're seeing in this little example is that, through a left-to-right pass, you can compute the value of J. And what we'll see in the next couple of slides is that in order to compute derivatives, there'll be a right-to-left pass like this, kind of going in the opposite direction as the blue arrows. That would be most natural for computing the derivatives. So to recap, the computation graph organizes a computation with this blue arrow, left-to-right computation. Let's refer to the next video how you can do the backward red arrow right-to-left computation of the derivatives. Let's go on to the next video.

Andrew Ng

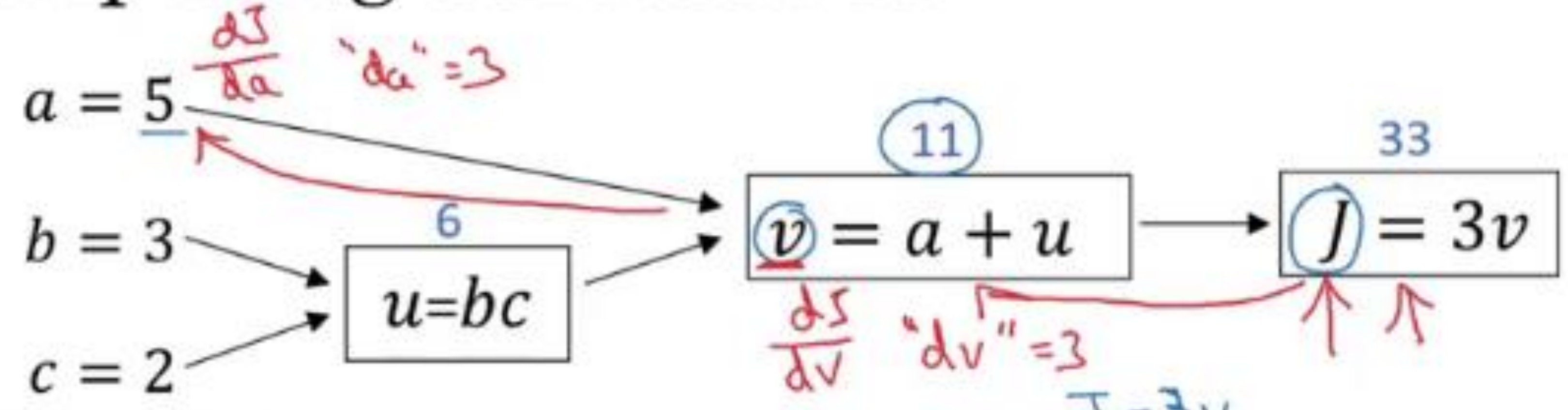
deeplearning.ai

Basics of Neural Network Programming

Derivatives with a Computation Graph

Andrew Ng

Computing derivatives



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

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

3×1

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

$a \rightarrow v \rightarrow J$

$J = 3v$

$v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

$a = 5 \rightarrow 5.001$

$\rightarrow v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

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

dJdvar

"dvar"

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

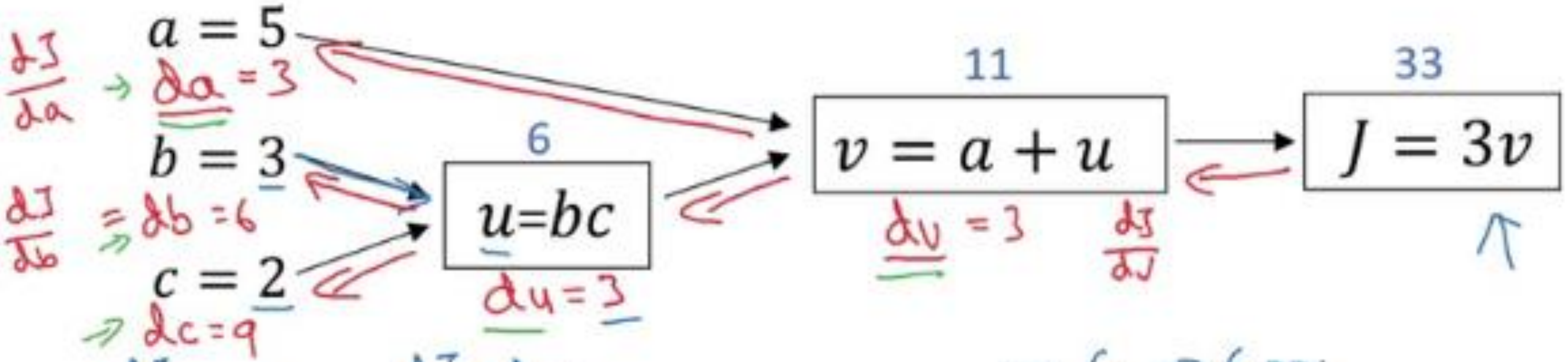
$$\frac{df(a)}{da} = \frac{df}{da} = 3$$

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

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

Andrew Ng

Computing derivatives



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

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

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

3×3

$u = 6 \to 6.001$

$v = 11 \to 11.001$

$J = 33 \to 33.003$

$b = 3 \to 3.001$

$u = b \cdot c = 6 \to 6.002$ $\quad c = 2$

$J = 33.006$ $\quad .006$

$v = 11.002$

$J = 3v$

Andrew Ng

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Logistic Regression Gradient descent

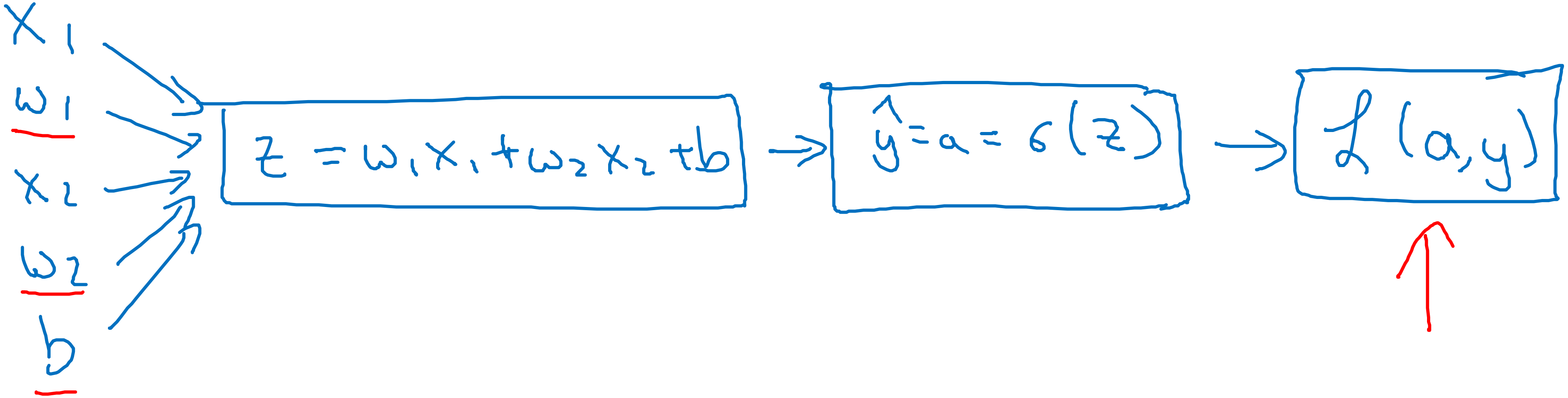
thsi si for one training example

Logistic regression recap

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

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

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



Andrew Ng

Logistic regression derivatives

$$\left.\begin{matrix} x_1 \\ w_1 \\ x_2 \\ w_2 \\ b \end{matrix}\right\} \rightarrow \boxed{z = w_1 x_1 + w_2 x_2 + b} \rightarrow \boxed{a = \sigma(z)} \rightarrow \boxed{\mathcal{L}(\mathrm{a}, y)}$$

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

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

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

this is solution of dL(a,y)/da

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

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

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Gradient descent on m examples

Logistic regression on m examples

we just using this training example

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

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

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

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

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

$$\underline{dw_1^{(i)}} \rightarrow (x^{(i)}, y^{(i)})$$

wwe showed in previews slide how to compute this for a single training sample

So for m training examples u just compute those partial derivatives for all the samples and do an average as is done above.

Logistic regression on m examples

$J=0;\ \underline{dw_1}=0;\ \underline{dw_2}=0;\ \underline{db}=0$

initialize them a those values.

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

this is for the training sets, from i to m

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

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

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

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

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

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

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

dw_3
$\vdots$
dw_n

$\underline{n}=2$

we assume we have just 2 features, in reality they will be a lot.

$J /= m \leftarrow$

You still need to divide by n becouse we are computing averages.

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

We do average, check previews slide why.

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

$$w_1 := w_1 - \alpha \underline{dw_1}$$

$$w_2 := w_2 - \alpha \underline{dw_2}$$

$$b := b - \alpha \underline{db}$$

Vectorization

deeplearning.ai

Basics of Neural Network Programming

Vectorization

What is vectorization?

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

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

Non-vectorized:

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

Vectorized

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

→ GPU
→ CPU

} SIMD - single instruction multiple data.

Andrew Ng

Basics of Neural Network Programming

deeplearning.ai

More vectorization examples

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

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, ~~dw1 = 0, dw2 = 0~~, db = 0

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

initialize a vector of zeros with nx elements(the number of weights)

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 $j = 1 \ldots n_x$

$dw_j \mathrel{+}= \ldots$

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

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

$n_x = 2$

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

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

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

$dw \mathrel{/}= m.$

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

Here we vectorise implementation without using any loop

Vectorizing Logistic Regression

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

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

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

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

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

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

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

b here is just real number but python expands it to a row vector b like above. Its called BROADCASTING

SEE DETTAIL OF THIS IN PROG ASSIGNMENT.

Andrew Ng

Basics of Neural Network Programming

deeplearning.ai

Vectorizing Logistic Regression's Gradient Computation

Vectorizing Logistic Regression

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

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

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

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

this is how we compute dz in vectorised form

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

$\cancel{dw_1}$
$\cancel{dw_2}$
$\vdots$

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

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

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

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

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

$n \times 1$

this is a vector of all the X input values

Andrew Ng

Implementing Logistic Regression

NO WAY TO GET RID OF THIS LOOP FOR NR OF ITERATIONS

THIS IS OUR NON EFFICIENT IMPLEMENTATION

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

for i = 1 to m:

WE NOW GOT RID OF THIS ONE ALSO

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

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

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

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

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

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

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

we got rid of a potential loop here.

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

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

$db = db/m$

for iter in range(1000):

$Z = w^T X + b$

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

$A = \sigma(Z)$

$dZ = A - Y$

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

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

$w := w - \alpha dw$

$b := b - \alpha db$

FINALD CODE HOW TO IMPLEMENT LOGISTIC REGRESSION

Andrew Ng

Basics of Neural Network Programming

Broadcasting in Python

deeplearning.ai

Broadcasting example

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

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

59cal

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

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))
```

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

```
>>> import numpy as np
>>> A = np.array([[56.0,0.0,4.4,68.0],
[1.2,104.0,52.0,8.0],[1.8,135.0,99.0,0.9]])
>>> A
array([[ 56. ,   0. ,   4.4,  68. ],
       [  1.2, 104. ,  52. ,   8. ],
       [  1.8, 135. ,  99. ,   0.9]])
>>> calc = A.sum(axis = 0)
>>> calc
array([ 59. , 239. , 155.4,  76.9])
>>> calc.reshape(1,4)
array([[ 59. , 239. , 155.4,  76.9]])
>>> percentage = 100*A/calc
>>> percentage
array([[94.91525424,  0.        ,  2.83140283,
88.42652796],
       [ 2.03389831, 43.51464435, 33.46203346,
10.40312094],
       [ 3.05084746, 56.48535565, 63.70656371,
1.17035111]])
>>>
```

Andrew Ng

Broadcasting example

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

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

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

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

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

General Principle

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

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

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

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

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

Basics of Neural Network Programming

A note on python/ numpy vectors

deeplearning.ai

Python Demo

Python / numpy vectors

```
import numpy as np

a = np.random.randn(5)

a = np.random.randn((5,1))

a = np.random.randn((1,5))

assert(a.shape = (5,1))
```

deeplearning.ai

Basics of Neural Network Programming

Quick tour of Jupyter/ipython notebooks

Andrew Ng

coursera

Logistic Regression with a Neural Network mindset - Student 06-16 Last Checkpoint: 06/24/2017 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Python 3

Code

CellToolbar

Submit Assignment

Control Panel Logout

1) Packages

Let's first import all the packages that you will need during this assignment.

- numpy is the fundamental package for scientific computing with Python.
- h5py is a common package to interact with a dataset that is stored on an H5 file.
- matplotlib is a famous library to plot graphs in Python.

In [6]:

```
import numpy as np
import matplotlib.pyplot as plt
import h5py

%matplotlib inline


### START CODE HERE ### (≈ 3 lines of code)
print("Hello World")
### END CODE HERE ###
```

```
Hello World
```

2) Overview of the Problem set

Problem Statement: You are given a dataset ("data.h5") containing:

```
- a training set of m_train images labeled as cat (1) or non-cat (0)
- a test set of m_test images labeled as cat and non-cat
```

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Explanation of logistic regression cost function (Optional)

Logistic regression cost function

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

$p(y|x)$

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

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

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

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

Andrew Ng

Andrew Ng

Cost on m examples

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

Maximum likelihood estimation

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

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

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

Andrew Ng

Andrew Ng