Week2

06 May 2020 18:43

Basics of Neural Network Programming

Binary Classification

deeplearning.ai

Hello, and welcome back.

Binary Classification

→ 1 (cat) vs 0 (non cat)

Here's an example of a binary classification problem.

Andrew Ng

Binary Classification

1 (cat) vs 0 (non cat)

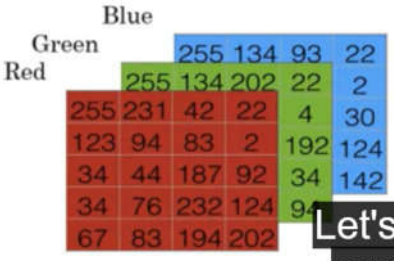
in which case you output 1, or
not-cat in which case you output 0,

Andrew Ng

Binary Classification

1 (cat) vs 0 (non cat)

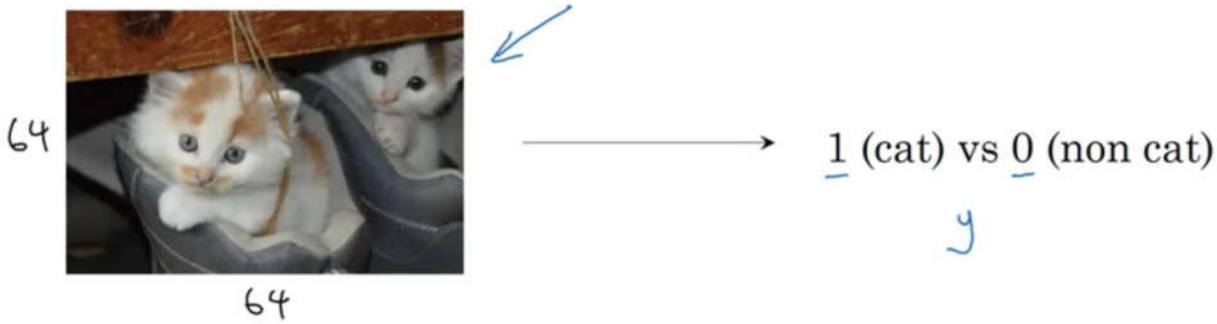
y

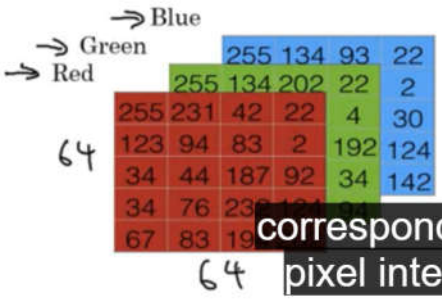


Let's look at how an image is
represented in a computer.

Andrew Ng

Binary Classification

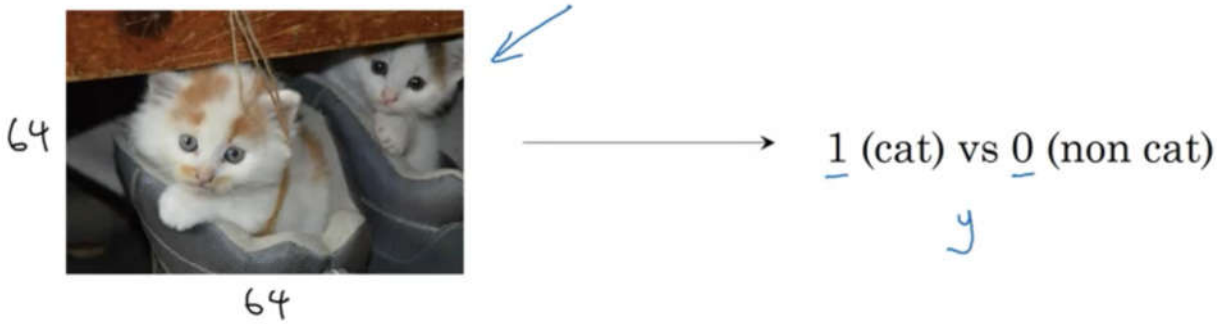


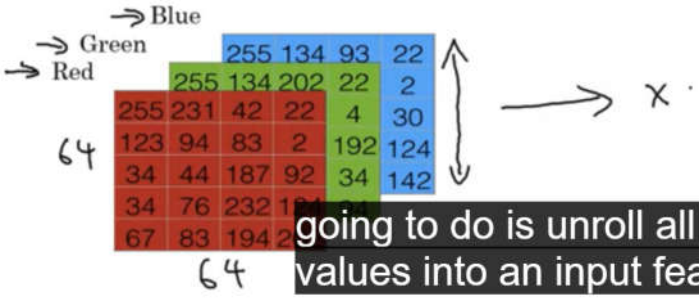


corresponding to the red, green and blue pixel intensity values for your images.

Andrew Ng

Binary Classification

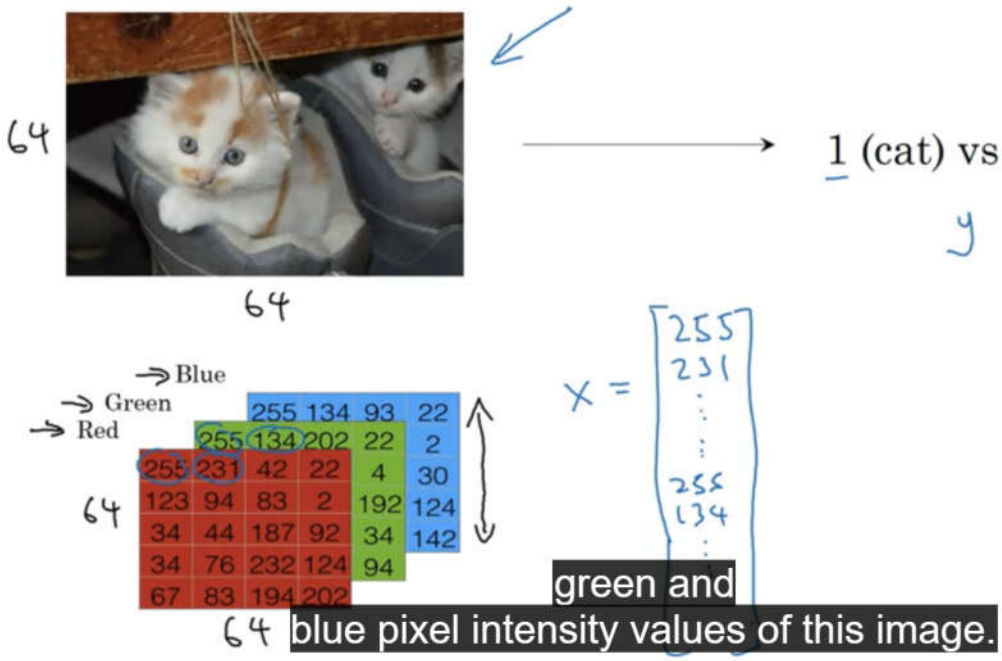




going to do is unroll all of these pixel values into an input feature vector x.

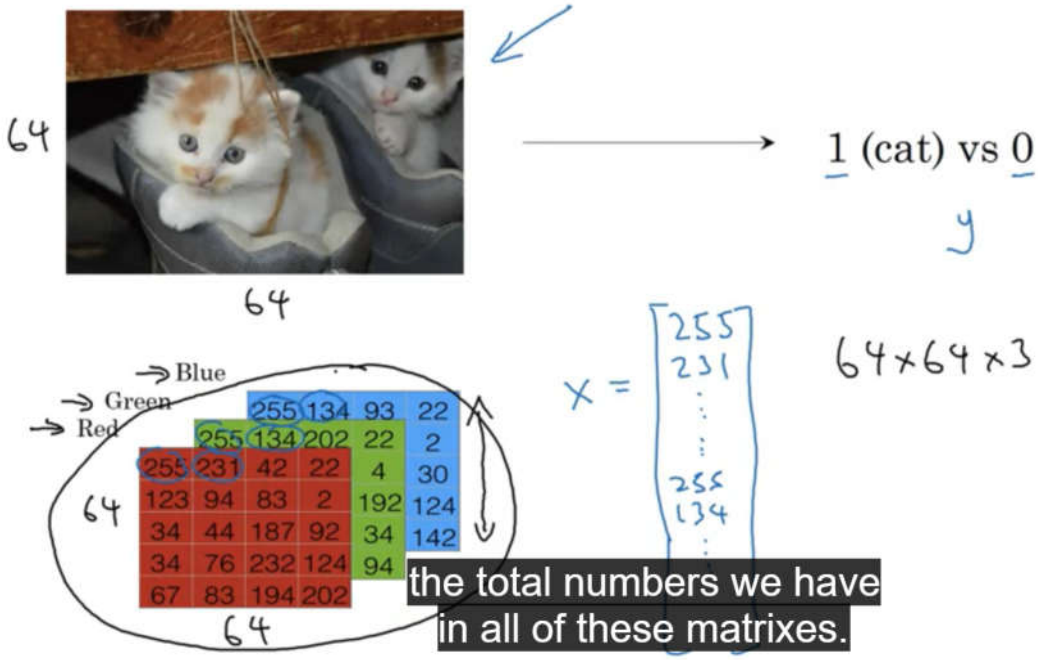
Andrew Ng

Binary Classification



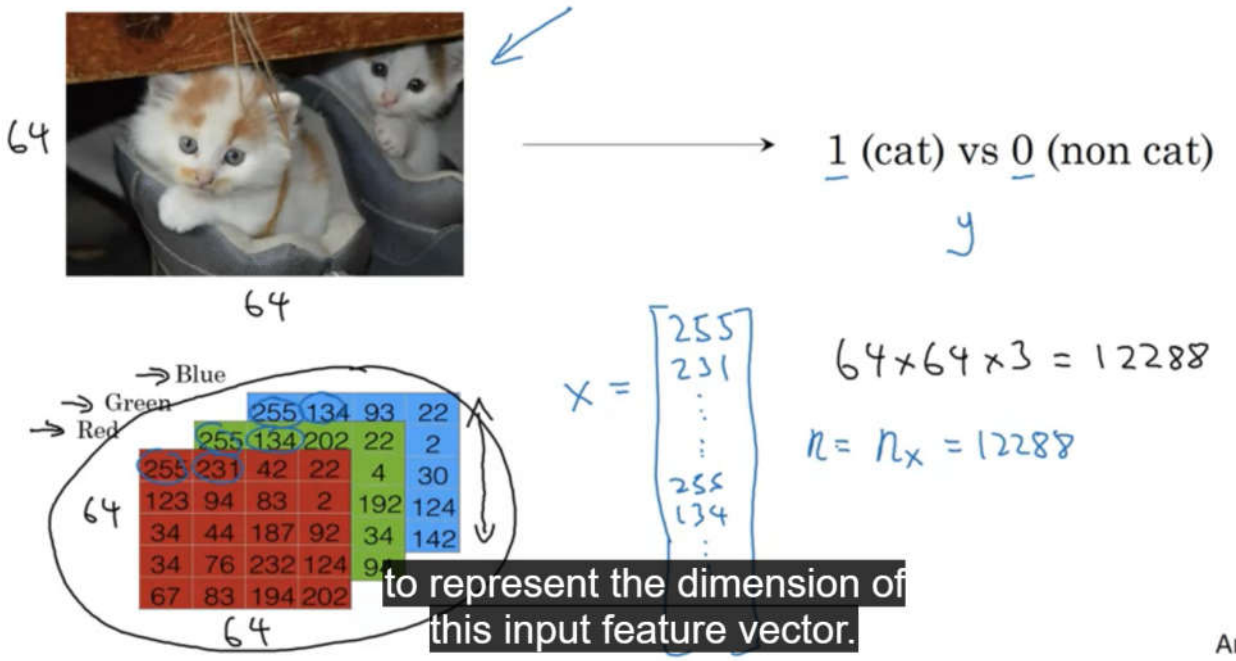
Andrew Ng

Binary Classification



Andrew Ng

Binary Classification



to represent the dimension of this input feature vector.

Andrew Ng

Notation

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

vector and y, the label, is either 0 or 1.

Andrew Ng

Notation

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

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

And then that altogether is
your entire training set.

Andrew Ng

Notation

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

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

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

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

$\longleftarrow m \longrightarrow$

$$\cancel{\begin{bmatrix} - & x^{(1)T} & - \\ & \vdots & \\ - & x^{(m)T} & - \end{bmatrix}}$$

$X \in \mathbb{R}^{n_x \times m}$

when you implement this in Python,

Andrew Ng

Notation

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

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

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

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

$\longleftarrow m \longrightarrow$

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

6:51 / 8:23

Andrew Ng

Notation

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

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

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

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \updownarrow n_x \qquad \cancel{\begin{bmatrix} - x^{(1)T} - \\ \vdots \\ - x^{(m)T} - \end{bmatrix}} \qquad Y = [y^{(1)} \; y^{(2)} \; \cdots, y^{(m)}]$$

$\longleftarrow m \longrightarrow$

$X \in \mathbb{R}^{n_x \times m} \qquad X.\text{shape}$ up to Y m like so.

Andrew Ng

Notation

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

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

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

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

$\longleftarrow m \longrightarrow$

$X \in \mathbb{R}^{n_x \times m}$

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

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

$Y.shape = (1, m)$

Which just means this is a 1 by m matrix.

Andrew Ng

Press Esc to exit full screen

Basics of Neural Network Programming

Logistic Regression

deeplearning.ai

In this video, we'll go over logistic regression.

0:00 / 5:58

deeplearning.ai

in a supervised learning problem are all either zero or one,

Logistic Regression

Press Esc to exit full screen

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

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

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

Output $\hat{y} = w^T x + b$

And in fact, this is what you use if
you were doing linear regression.

1:35 / 5:58

Andrew Ng

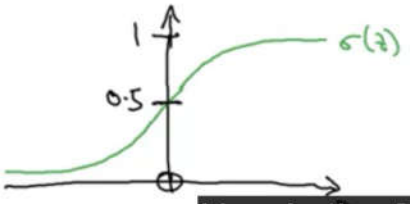
Logistic Regression

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

$x \in \mathbb{R}^{n_x}$ $\qquad 0 \le \hat{y} \le 1$

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

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



Here's the formula for the sigmoid function.

Andrew Ng

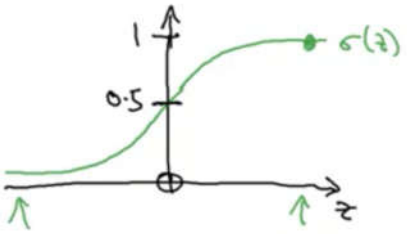
Logistic Regression

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

$x \in \mathbb{R}^{n_x}$ $\qquad 0 \le \hat{y} \le 1$

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

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



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

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

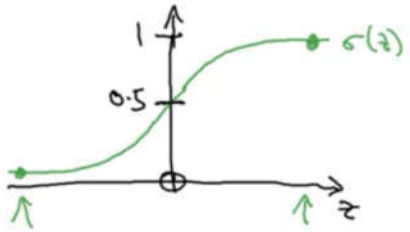
Logistic Regression

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

$x \in \mathbb{R}^{n_x}$ $\qquad 0 \le \hat{y} \le 1$

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

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



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

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

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \begin{matrix} \} b \leftarrow \\ \\ \} w \leftarrow \\ \\ \\ \end{matrix}$$

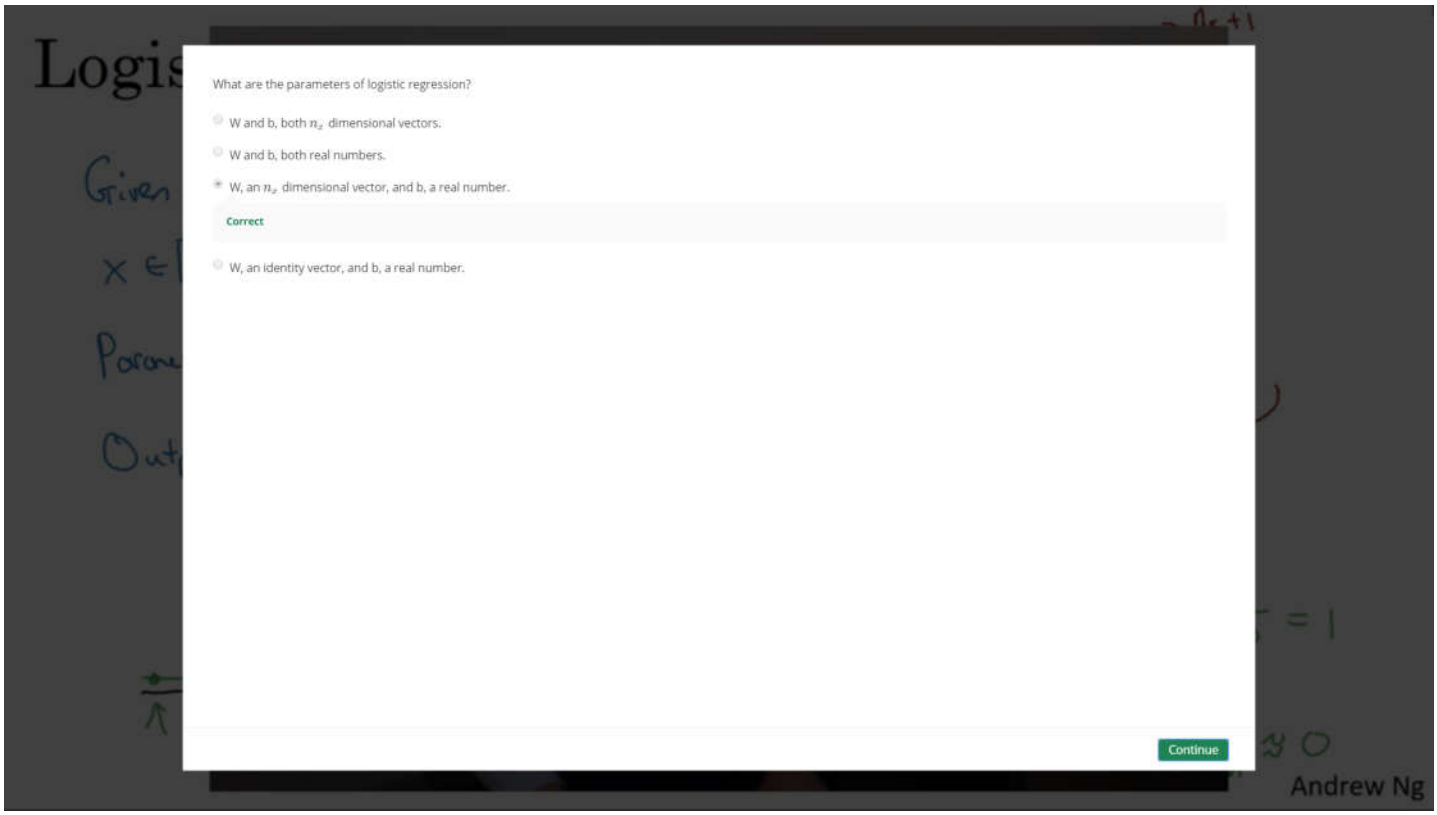
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

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

In a previous video,

0:00 / 8:11

Logistic Regression cost function

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

So your output y-hat is sigmoid of w

0:20 / 8:11

Andrew Ng

Logistic Regression cost function

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

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

seems natural that you want to find parameters W

Andrew Ng

Logistic Regression cost function

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

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

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

but in logistic regression people don't usually do

Andrew Ng

Logistic Regression cost function

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, 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 $\hat{y}^{(i)} \approx y^{(i)}$. $\quad x^{(i)}, y^{(i)}, z^{(i)}$ $\quad i$-th example.

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

need to define to measure how good our output y-hat is when the true label is y.

Andrew Ng

Logistic Regression cost function

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, 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 $\hat{y}^{(i)} \approx y^{(i)}$. $\quad x^{(i)}, y^{(i)}, z^{(i)}$ $\quad i$-th example.

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

convex and so we'll see in that later video becomes much easier to optimize.

Andrew Ng

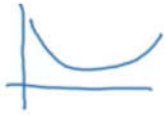
Logistic Regression cost function

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, 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 $\hat{y}^{(i)} \approx y^{(i)}$. $\quad x^{(i)}, y^{(i)}, z^{(i)}$ i-th example.

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

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



Here's some intuition for why this loss function makes sense.

Andrew Ng

Logistic Regression cost function

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, 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 $\hat{y}^{(i)} \approx y^{(i)}$. $\quad x^{(i)}, y^{(i)}, z^{(i)}$ i-th example.

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

But it can't ever be bigger than one so saying you want y-hat to be close to one as well.

4:45 / 8:11

Andrew Ng

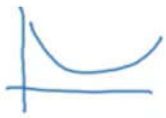
Logistic Regression cost function

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, 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 $\hat{y}^{(i)} \approx y^{(i)}$. $\quad x^{(i)}, y^{(i)}, z^{(i)}$ i-th example.

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

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



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

that this loss function is trying to make y-hat as small as possible.

Andrew Ng

Logistic Regression cost function

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, 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 $\hat{y}^{(i)} \approx y^{(i)}$. $\quad x^{(i)}, y^{(i)}, z^{(i)}$ i-th example.

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

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

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

What is the difference between the cost function and the loss function for logistic regression?

- The cost function computes the error for a single training example; the loss function is the average of the cost functions of the entire training set.
- They are different names for the same function.
- The loss function computes the error for a single training example; the cost function is the average of the loss functions of the entire training set.

Correct

Continue

Andrew Ng

coursera.org/learn/neural-networks-deep-learning/supplement/F7xbU/clarification-about-upcoming-gradient-descent-video

Neural Networks and Deep Learning > Week 2 > Clarification about Upcoming Gradient Descent Video

Prev | Next

Logistic Regression as a Neural Network

- **Video:** Binary Classification 8 min
- **Video:** Logistic Regression 5 min
- **Reading:** Clarification about Upcoming Logistic Regression Cost Function Video 1 min
- **Video:** Logistic Regression Cost Function 8 min
- **Reading:** Clarification about Upcoming Gradient Descent Video 1 min
- **Video:** Gradient Descent 11 min
- **Video:** Derivatives 7 min
- **Video:** More Derivative Examples 10 min
- **Video:** Computation

Please note that in the next video at the second slide, there is a missing parenthesis.

The negative sign should apply to the entire cost function (both terms in the summation).

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(yhat^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} logyhat^{(i)} + (1-y^{(i)}) log(1-yhat^{(i)})\right)$$

✓ Complete Go to next item

Press Esc to exit full screen

Basics of Neural Network Programming

Gradient Descent

deeplearning.ai

You've seen the logistic regression model.

11:12 / 11:23

Gradient Descent

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

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

To recap, here is the familiar logistic regression algorithm.

0:26 / 11:23

Andrew Ng

Gradient Descent

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

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

And so the loss function
measures how well your algorithms

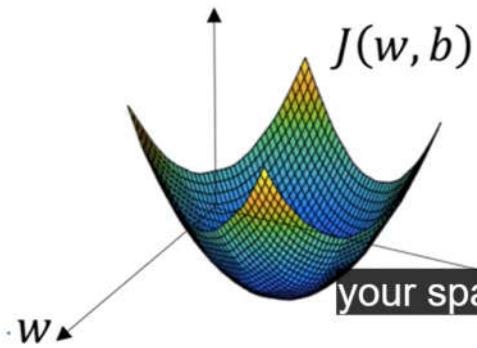
Andrew Ng

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 $J(w,b)$



your spatial parameters, w and b.

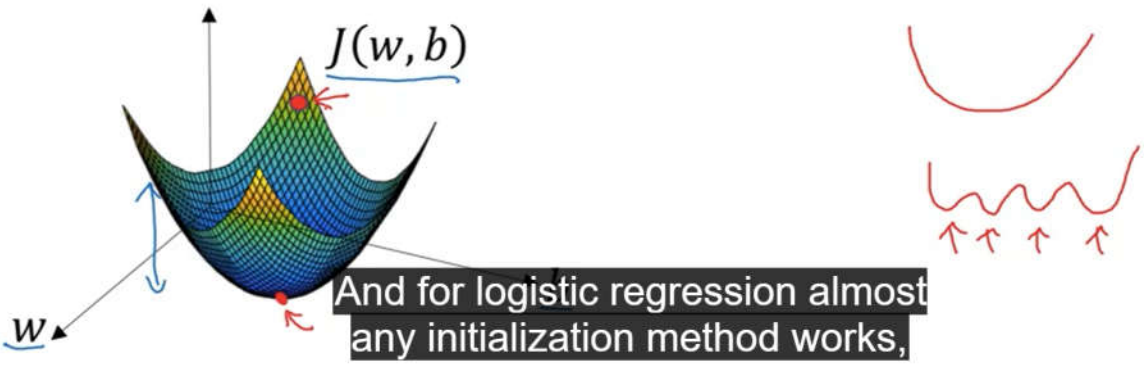
Andrew Ng

Gradient Descent

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

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

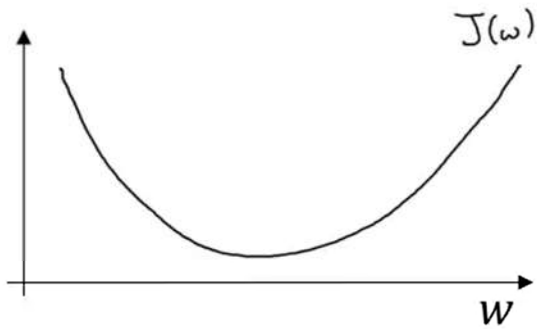
Want to find w, b that minimize $J(w,b)$



And for logistic regression almost any initialization method works,

Andrew Ng

Gradient Descent



Repeat {

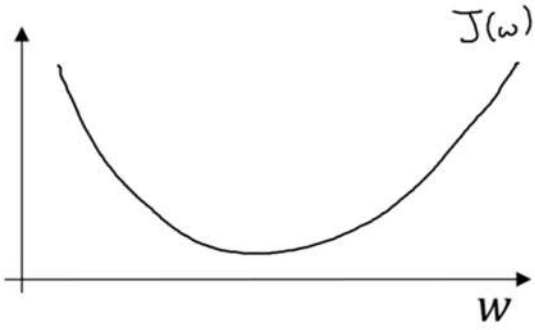
$$w := w - \alpha \frac{dJ(w)}{dw}$$

}

I will repeatedly do that until the algorithm converges.

Andrew Ng

Gradient Descent



learning rate

Repeat {

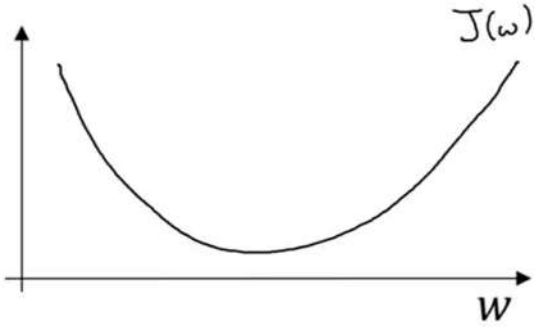
$$w := w - \alpha \frac{dJ(w)}{dw}$$

}

controls how big a step we take on each iteration or gradient descent.

Andrew Ng

Gradient Descent



learning rate

Repeat {

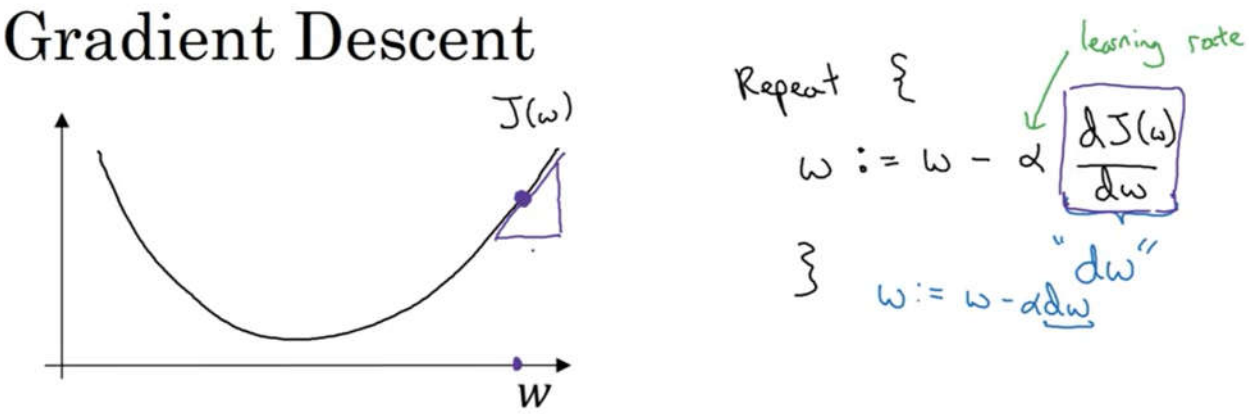
$$w := w - \alpha \underbrace{\frac{dJ(w)}{dw}}_{\text{"}dw\text{"}}$$

}

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

And so we use dw to be the variable name to represent this derivative term.

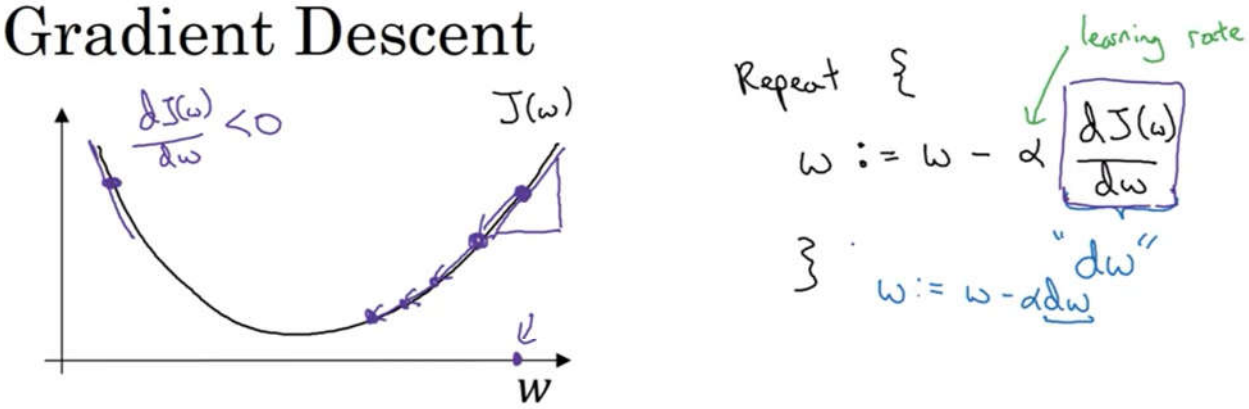
Andrew Ng



And so, here the derivative is positive.

5:43 / 11:23

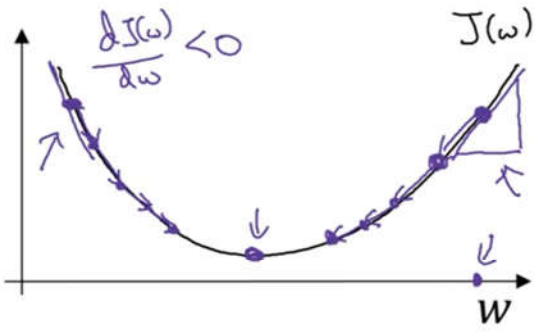
Andrew Ng



the gradient descent update would subtract alpha times a negative number.

Andrew Ng

Gradient Descent



Repeat {

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

learning rate

"dw"

}

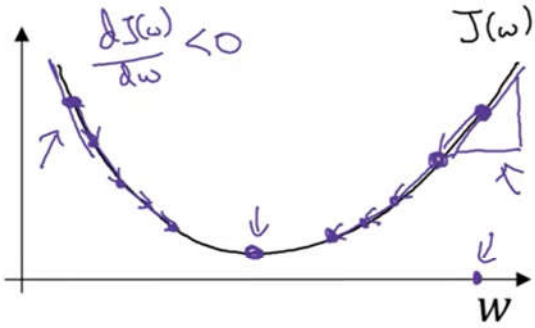
$$w := w - \alpha dw$$

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

now is that this term represents
the slope of the function, and

Andrew Ng

Gradient Descent



Repeat {

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

learning rate

"dw"

}

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

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

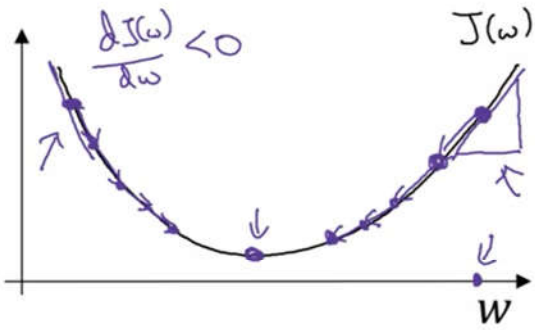
$J(w, b)$

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

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

Andrew Ng

Gradient Descent



Repeat {

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

learning rate

"dw"

}

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

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

$J(w,b)$

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

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

∂ "partial derivative"

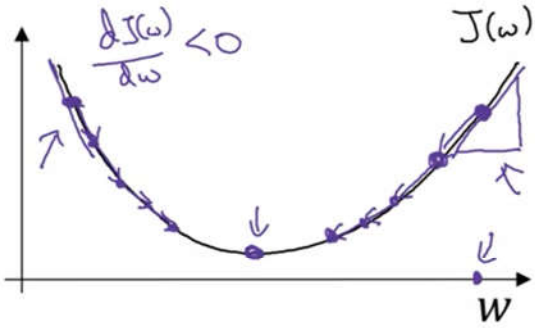
d ← J

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

But if you see this partial derivative symbol all it means is you're measure

Andrew Ng

Gradient Descent



Repeat {

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

learning rate

"dw"

}

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

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

$J(w,b)$

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

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

∂ "partial derivative"

d ← J

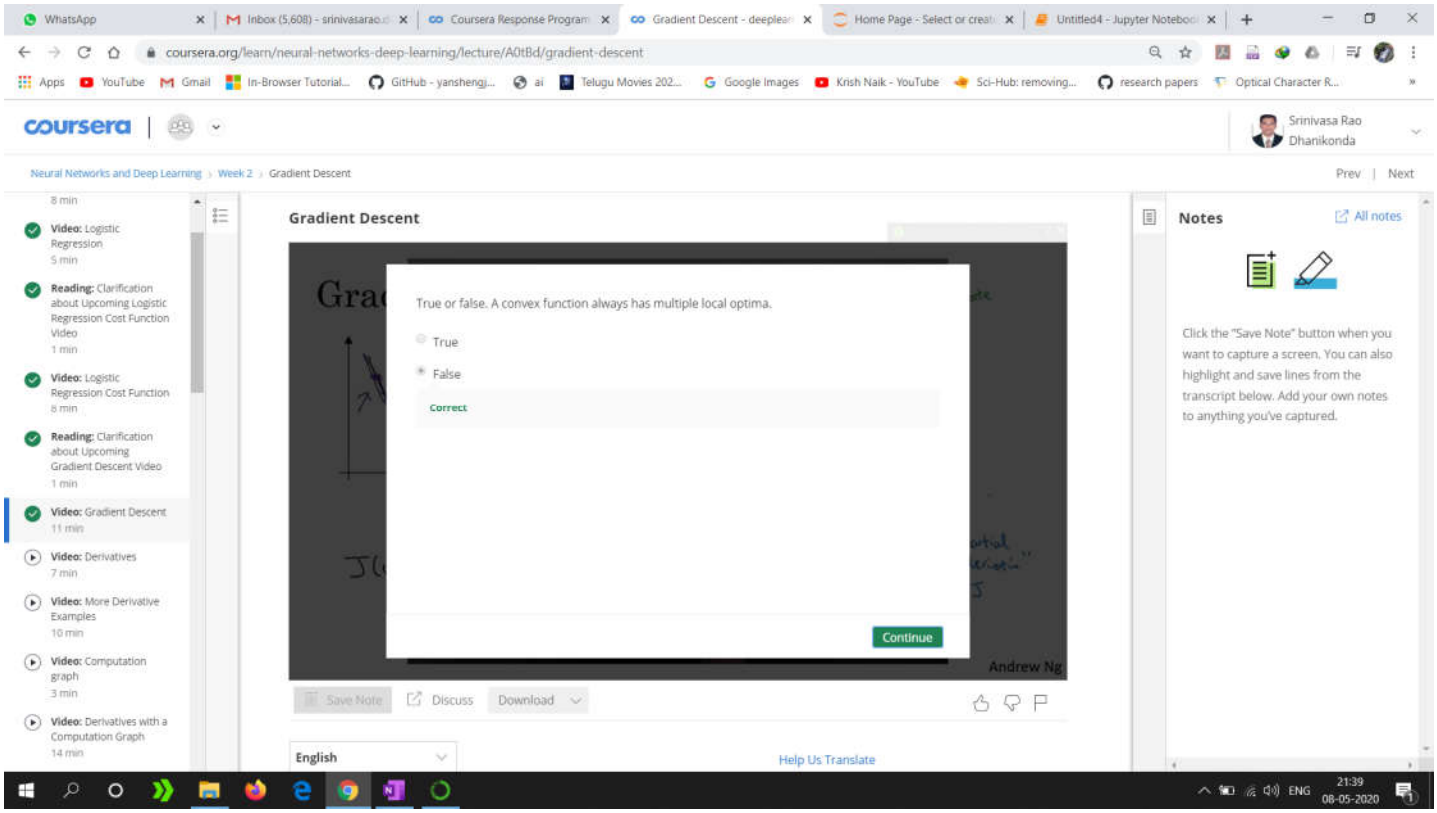
$$b := b - \alpha \boxed{\frac{\partial J(w,b)}{\partial b}}$$

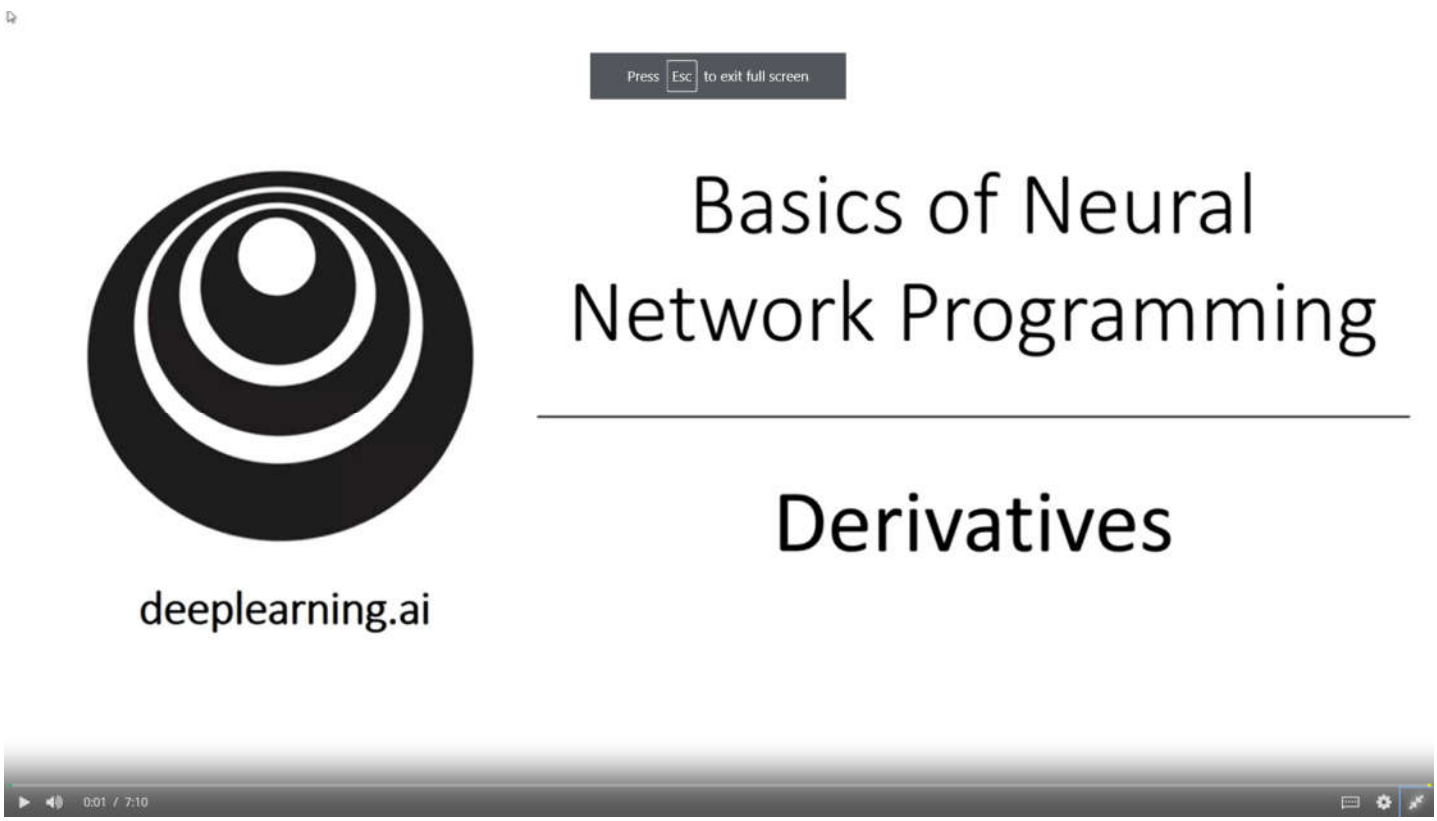
$$\frac{\partial J(w,b)}{\partial b}$$

dw

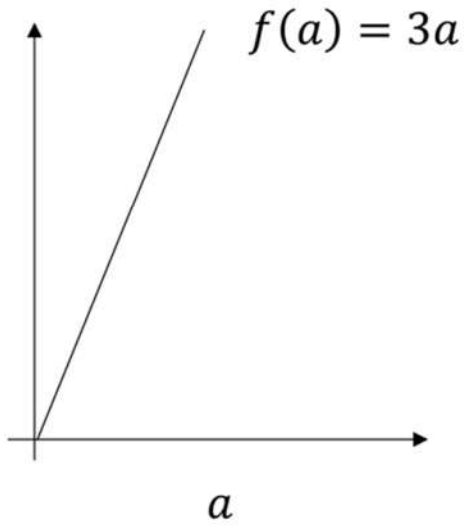
db

Andrew Ng


coursera
Neural Networks and Deep Learning › Week 2 › Gradient Descent
Prev | Next
Gradient Descent
True or false. A convex function always has multiple local optima.
True
False
Correct
Continue
Andrew Ng
Save Note
Discuss
Download
English
Help Us Translate
Notes
All notes
Click the "Save Note" button when you want to capture a screen. You can also highlight and save lines from the transcript below. Add your own notes to anything you've captured.

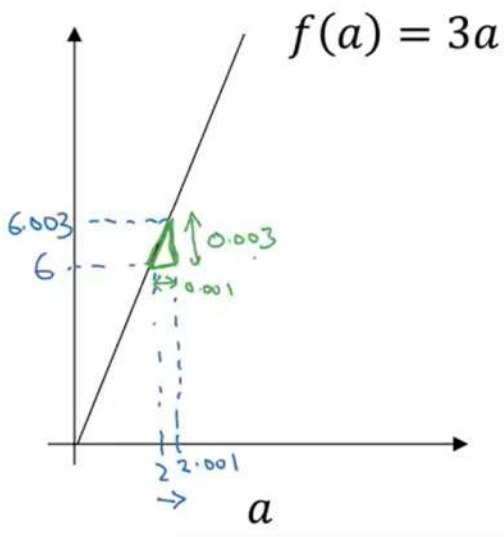

Press Esc to exit full screen
Basics of Neural Network Programming
Derivatives
deeplearning.ai
0:01 / 7:10

Intuition about derivatives



Andrew Ng

Intuition about derivatives



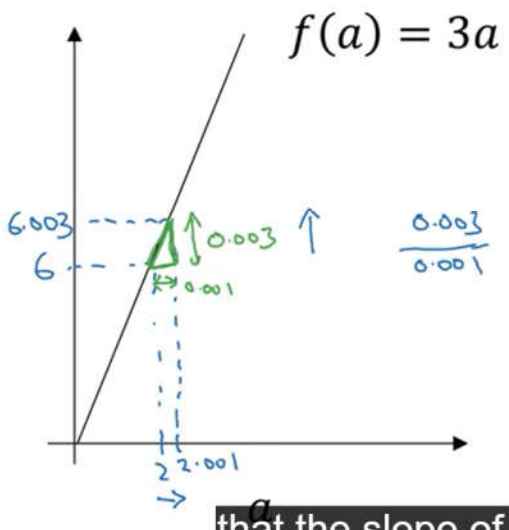
$a = 2$ $f(a) = 6$

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

2:51 / 7:10

Andrew Ng

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

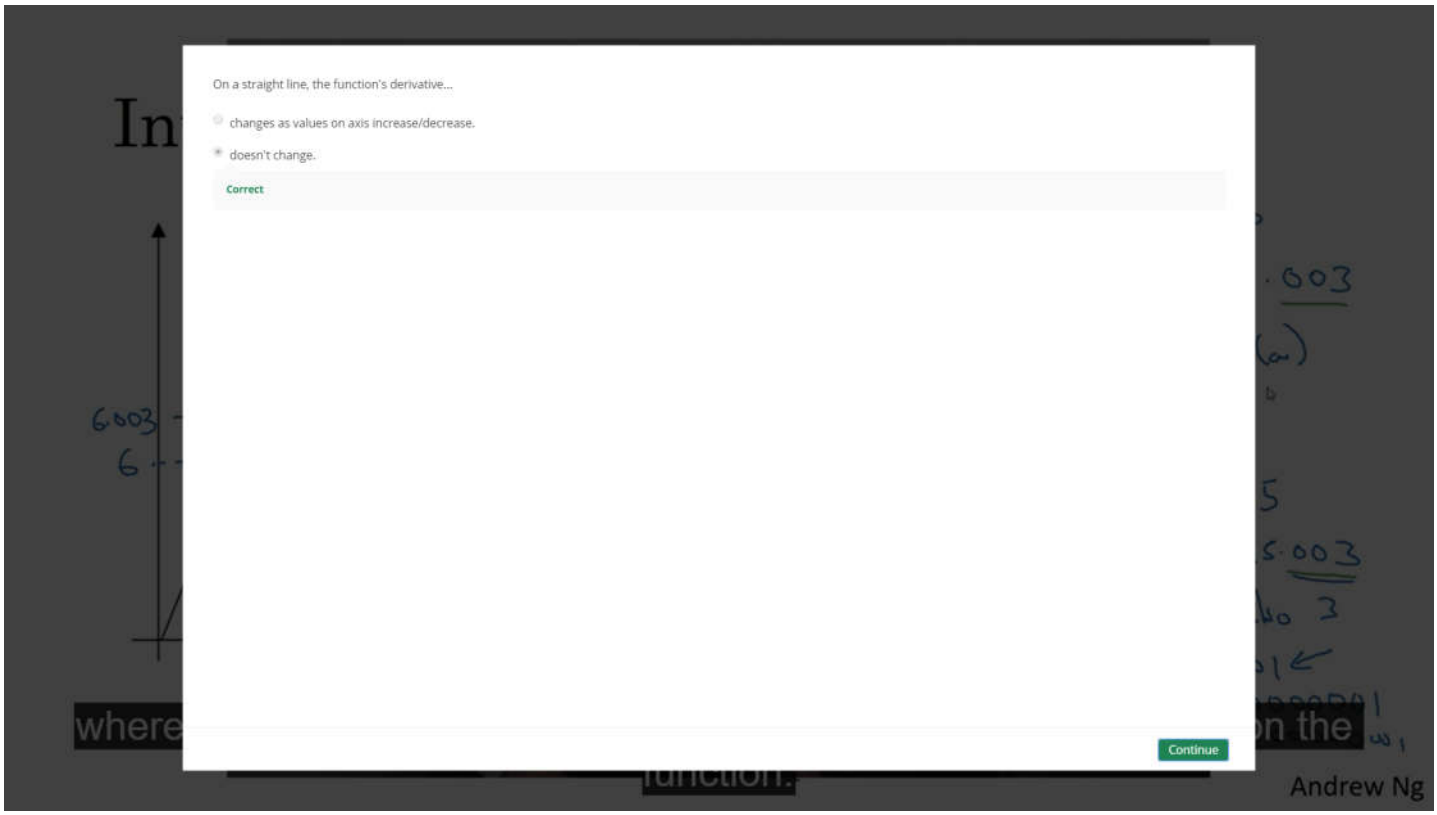
$a = 5$ $f(a) = 15$

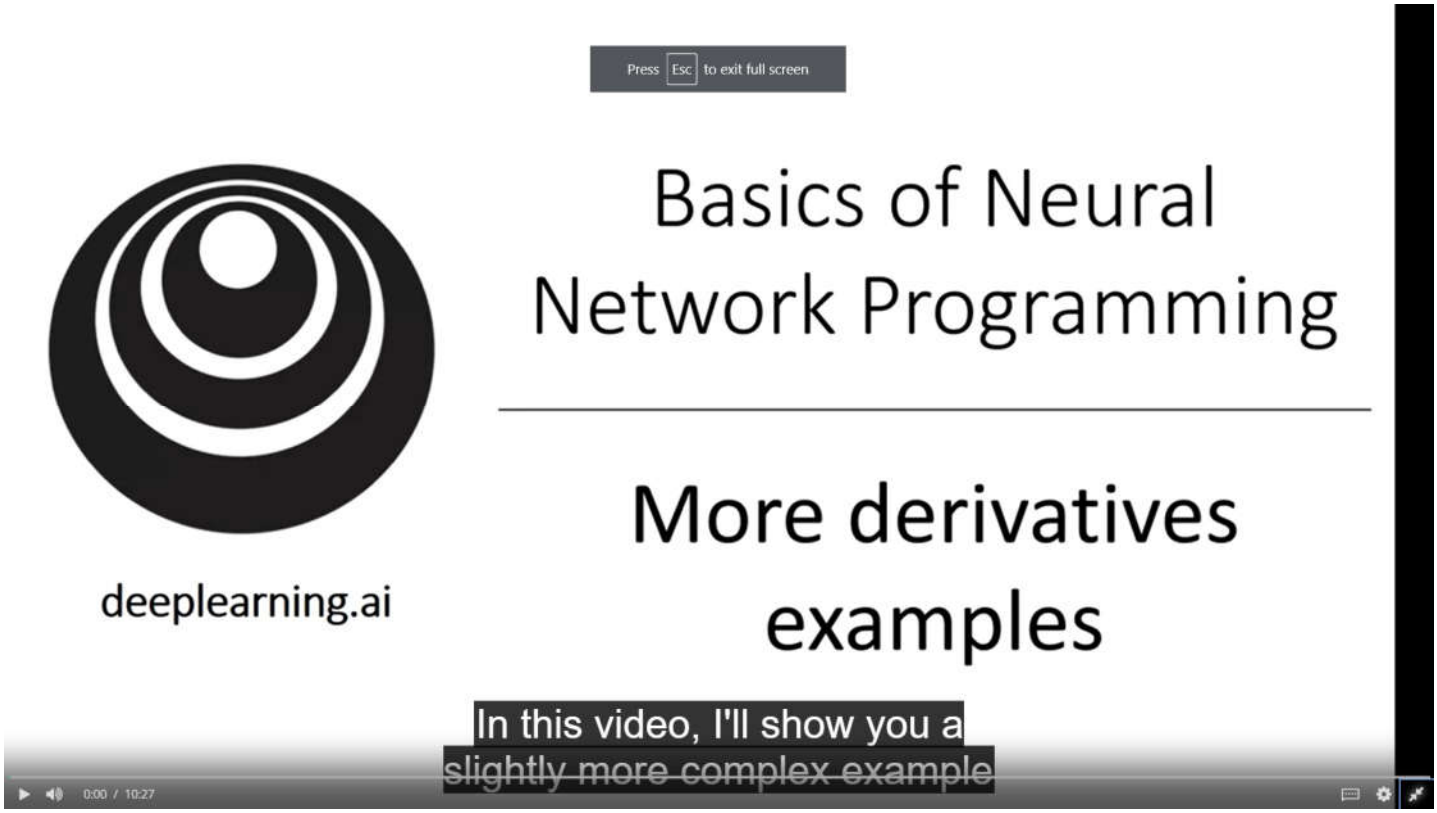
$a = 5.001$ $f(a) = 15.003$

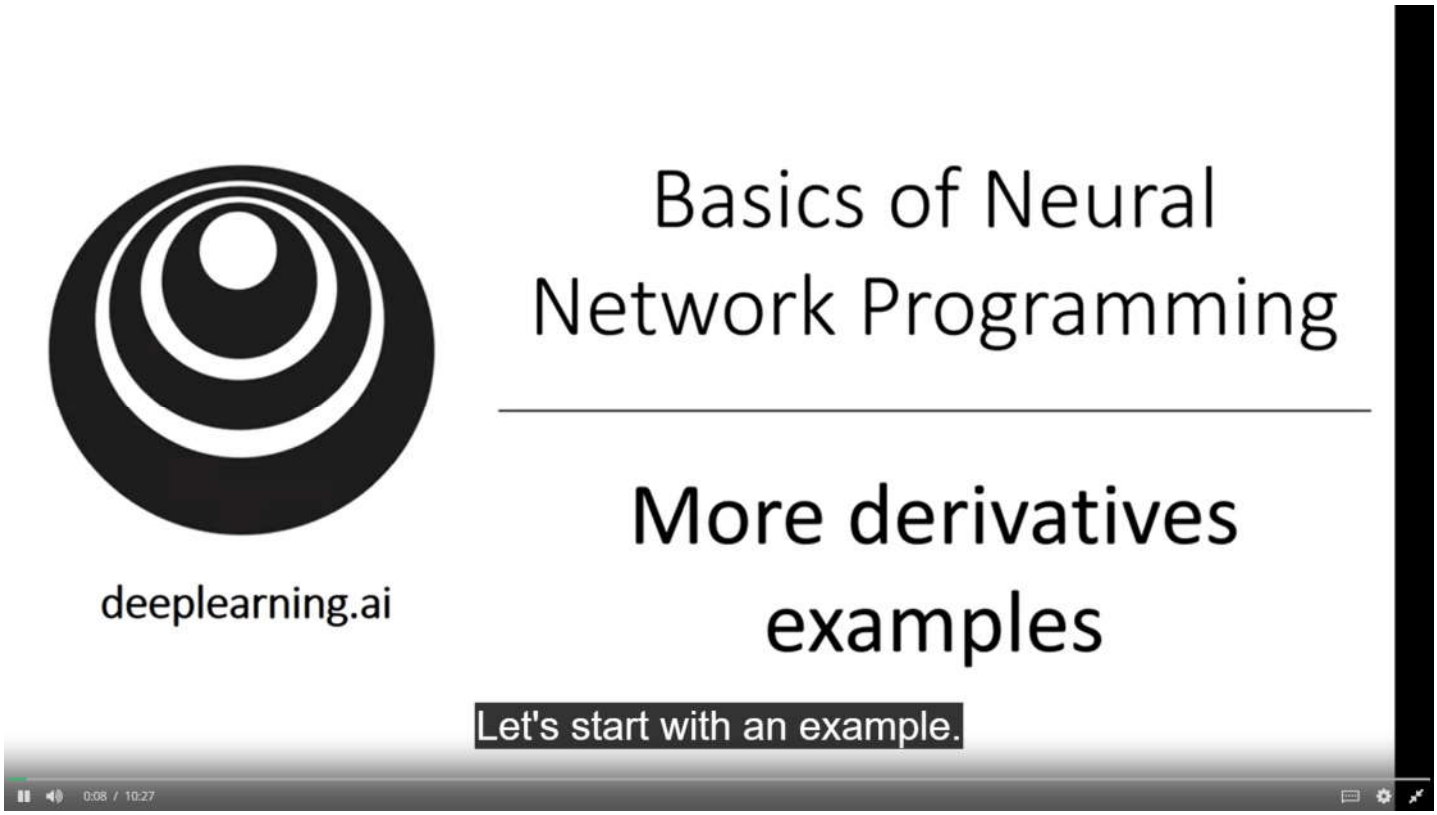
slope at $a=5$ is also 3

that the slope of the function f is equal to three:

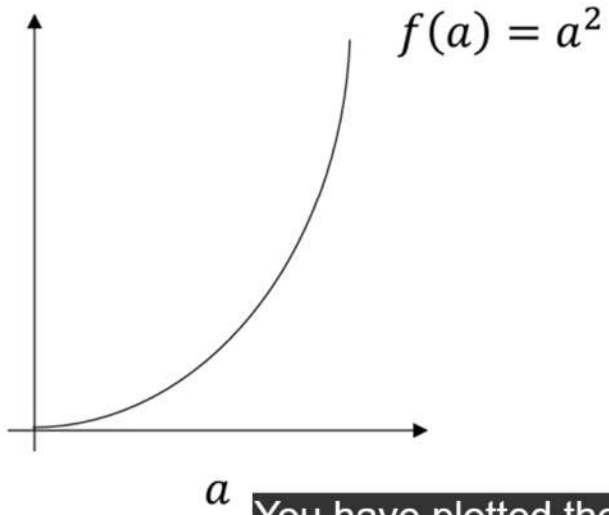
Andrew Ng




Press Esc to exit full screen
Basics of Neural Network Programming
deeplearning.ai
More derivatives examples
In this video, I'll show you a slightly more complex example
0:00 / 10:27


Basics of Neural Network Programming
deeplearning.ai
More derivatives examples
Let's start with an example.
0:08 / 10:27

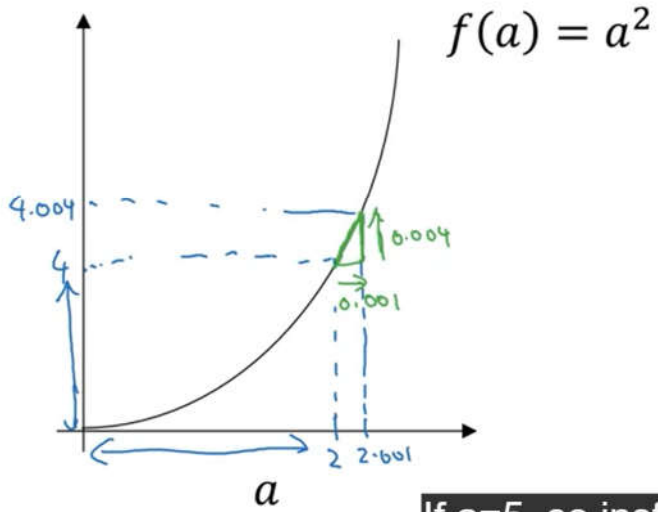
Intuition about derivatives



You have plotted the function f(a) = a².

Andrew Ng

Intuition about derivatives



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

$a = 2.001$ $f(a) \approx 4.004$ (4.004001)

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

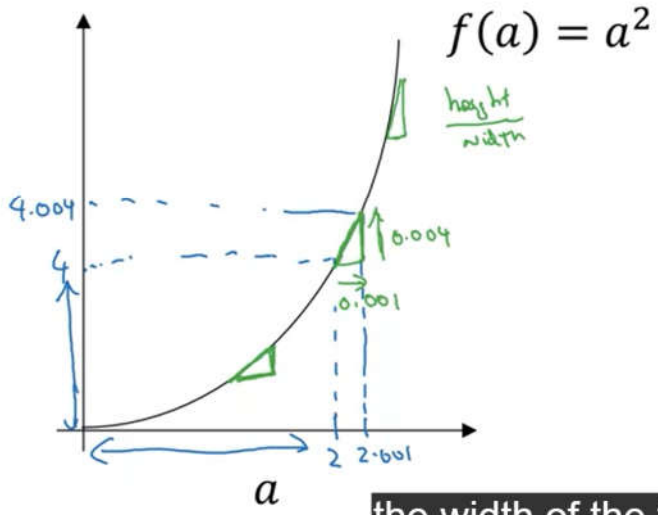
$\frac{d}{da} f(a) = 4$ when $a = 2$.

a=

If a=5, so instead of a=2,

Andrew Ng

Intuition about derivatives



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

$a = 2.001$ $\quad f(a) \approx 4.004$ (4.004001)

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

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

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

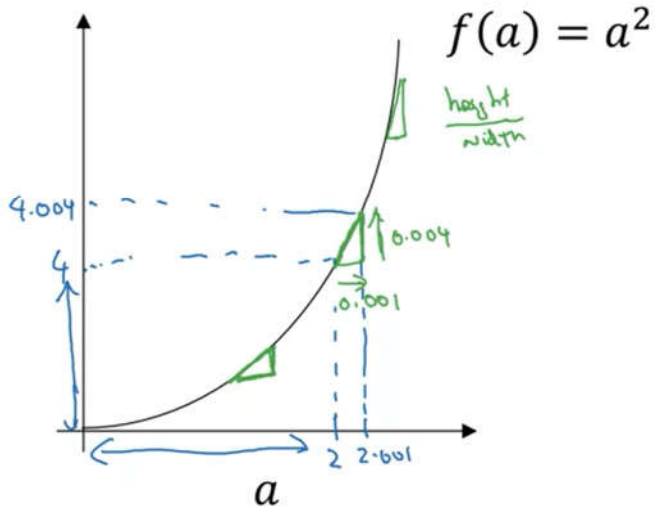
$a = 5.001$ $\quad f(a) \approx 25.010$

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

the width of the triangle is very different at different points on the curve.

Andrew Ng

Intuition about derivatives



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

$a = 2.001$ $\quad f(a) \approx 4.004$ (4.004001)

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

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

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

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

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

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

Andrew Ng

More derivative examples

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

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

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

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

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

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

0.0005

$\ln(a)$

$\downarrow$

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

Andrew Ng

Press Esc to exit full screen

Basics of Neural Network Programming

Computation Graph

deeplearning.ai

0:00 / 3:33

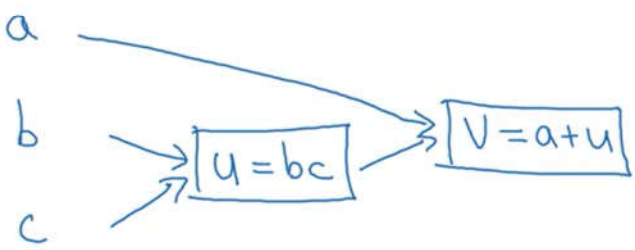
Computation Graph

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

$u = bc$

$v = a + u$

$J = 3v$



are V. So the inputs to that are u

Andrew Ng

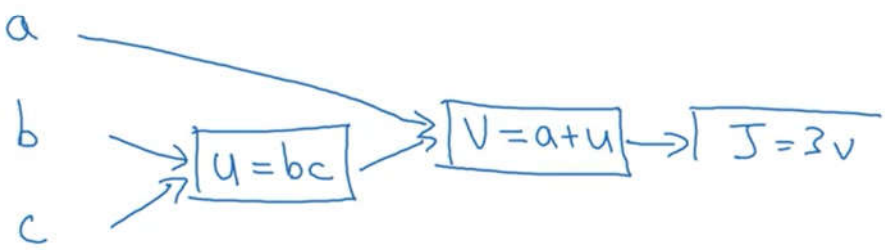
Computation Graph

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

$u = bc$

$v = a + u$

$J = 3v$



And then finally, we have J=3V.

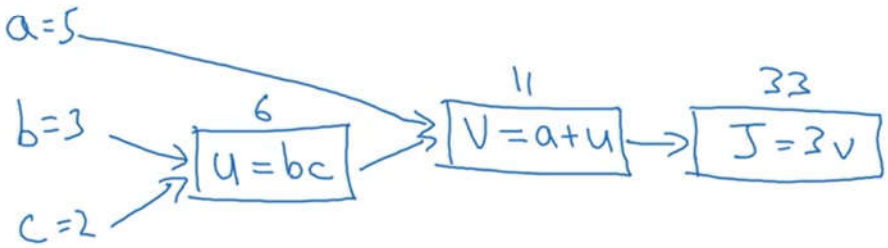
2:01 / 3:33

Andrew Ng

Computation Graph

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

$u = bc$
$v = a + u$
$J = 3v$



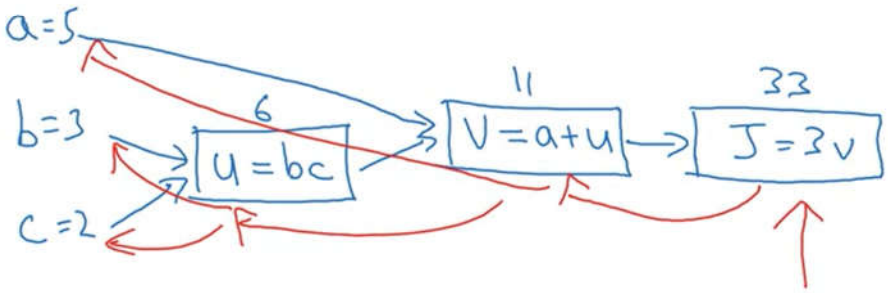
you actually get 33 as the value of J.

Andrew Ng

Computation Graph

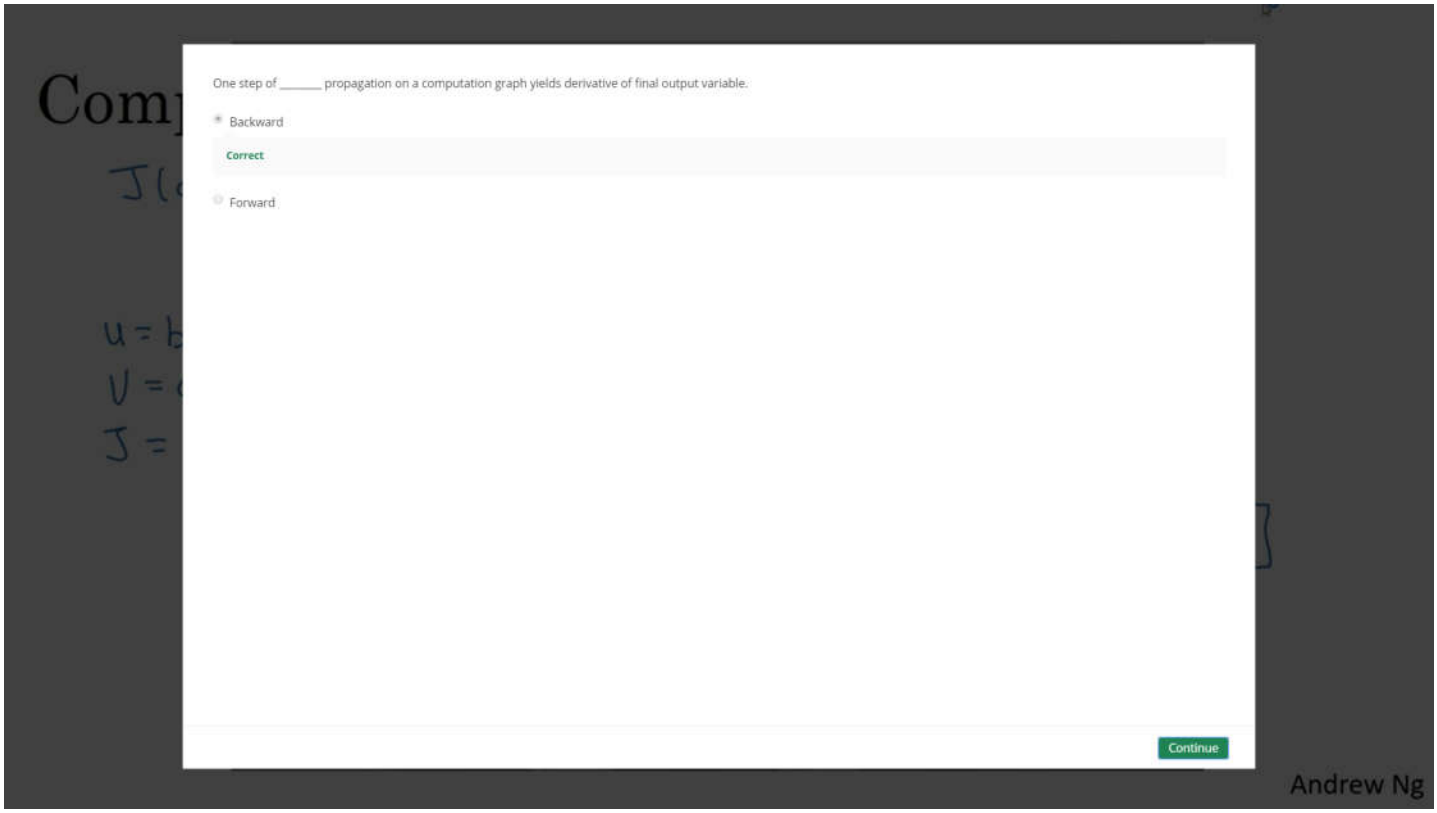
$$J(a,b,c) = \underbrace{3(\underbrace{a + \underbrace{bc}_{u}}_{v})}_{J} = 3(5 + 3\times 2) = 33$$

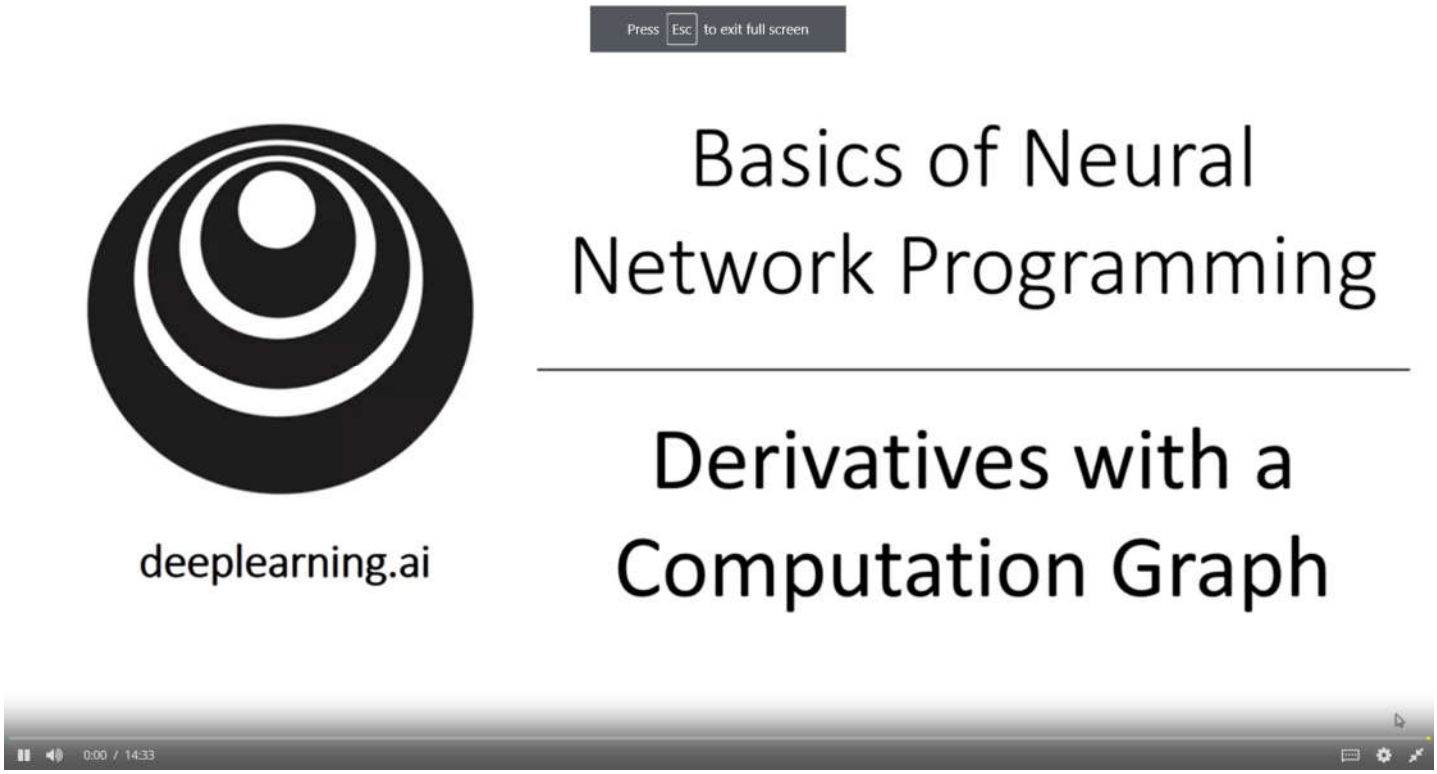
$u = bc$
$v = a + u$
$J = 3v$



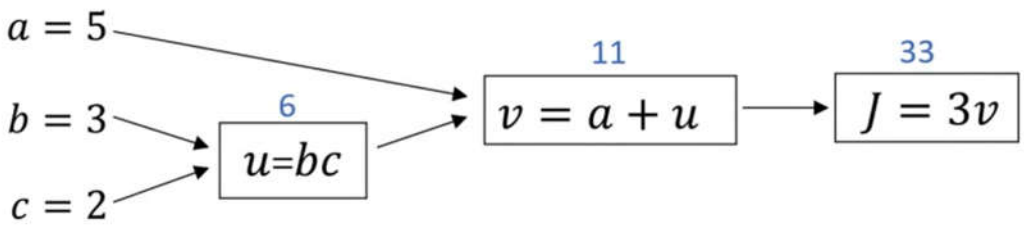
as the blue arrows.

Andrew Ng


One step of _____ propagation on a computation graph yields derivative of final output variable.
Backward
Correct
Forward
Continue
Andrew Ng


Press Esc to exit full screen
Basics of Neural Network Programming
Derivatives with a Computation Graph
deeplearning.ai
0:00 / 14:33

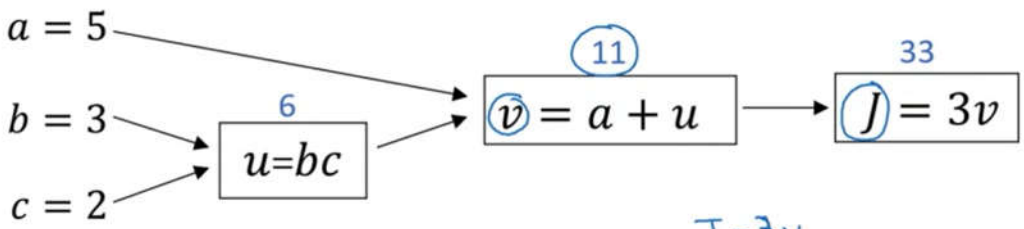
Computing derivatives



So here's a computation graph.

Andrew Ng

Computing derivatives



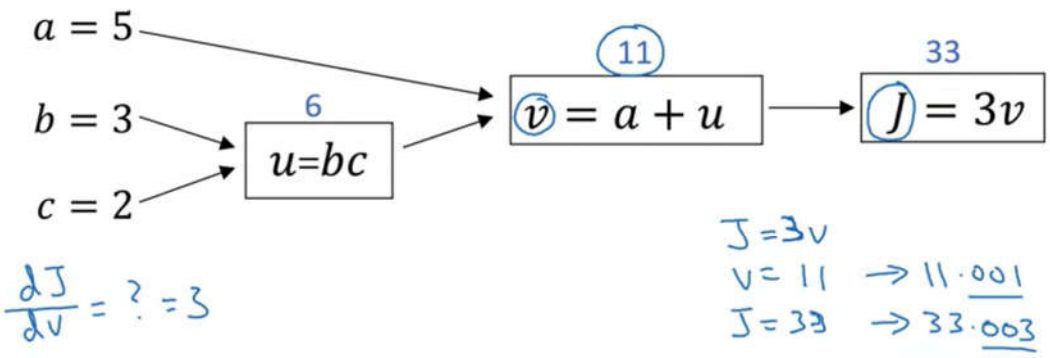
$\frac{dJ}{dv} = ?$

$J = 3v$
$v = 11 \rightarrow 11.001$
$J = 33 \rightarrow 33.003$

So here, we've increased v by 0.001.

Andrew Ng

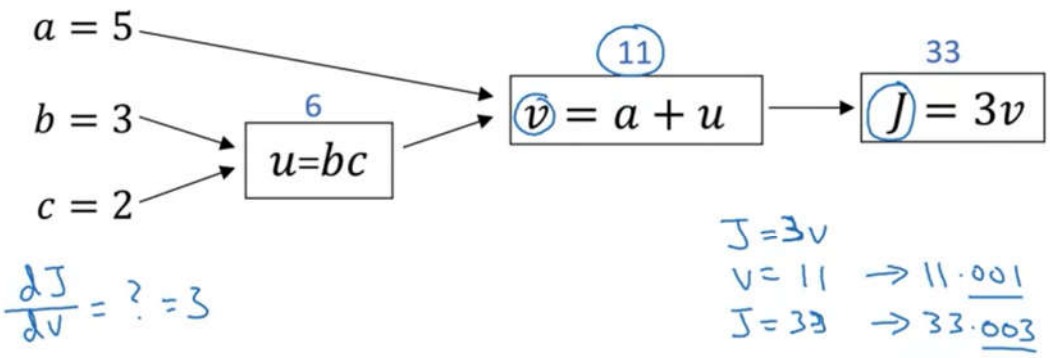
Computing derivatives



Because the increase in J is
3 times the increase in v.

Andrew Ng

Computing derivatives



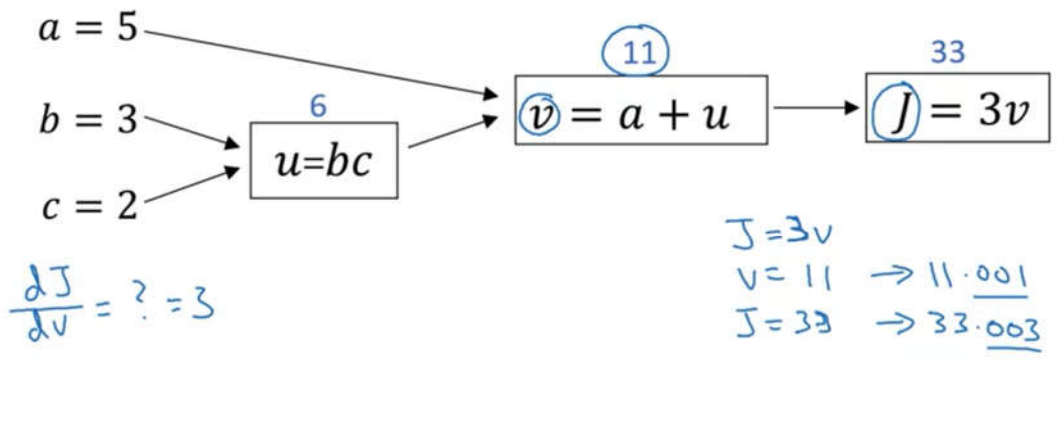
$f(a) = 3a$

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

a slightly sloppy notation,
you can write as df/da = 3.

Andrew Ng

Computing derivatives



$f(a) = 3a$

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

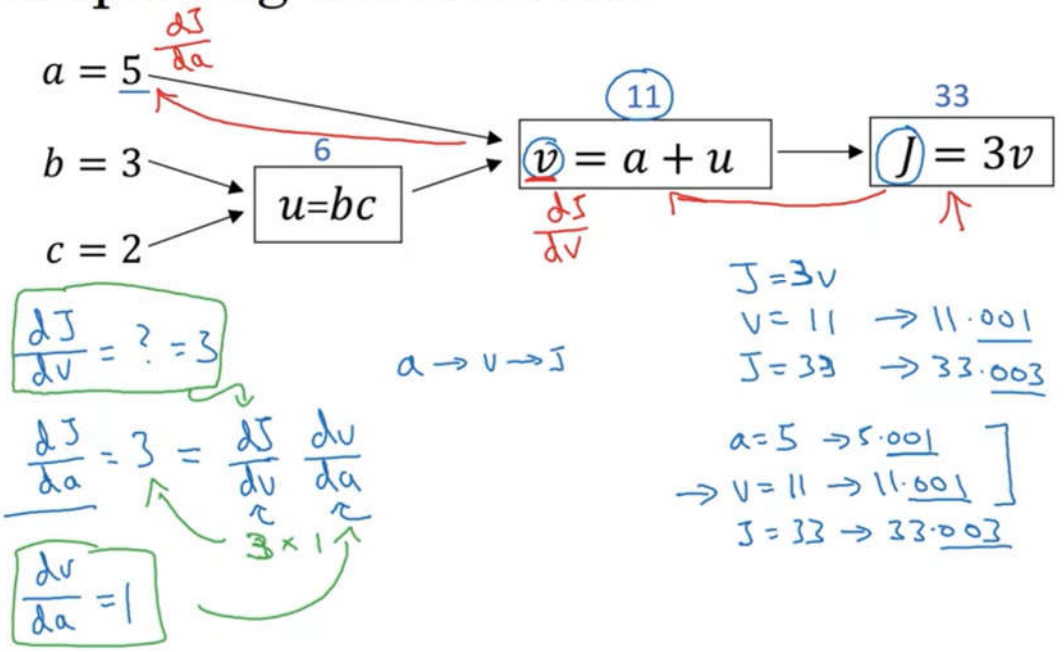
$J = 3v$

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

v playing the role of a in this previous example that we had from an earlier video.

Andrew Ng

Computing derivatives



$f(a) = 3a$

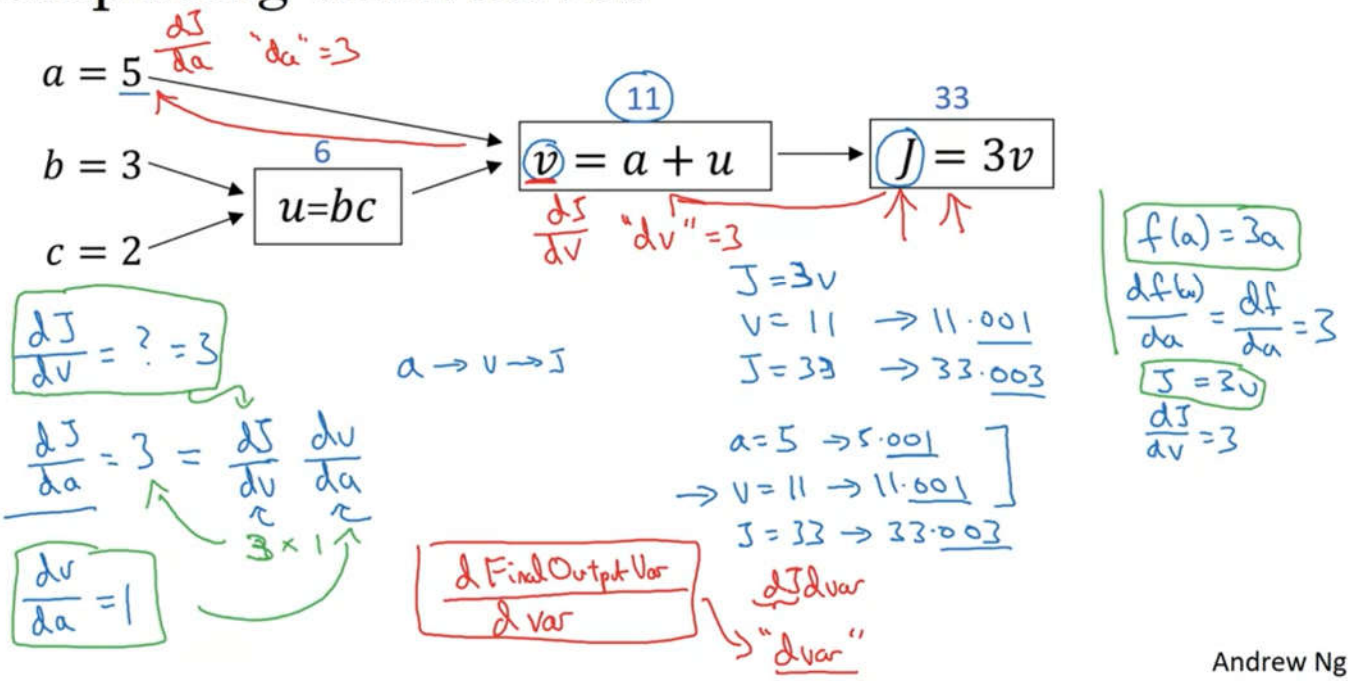
$\frac{df(a)}{da} = \frac{df}{da} = 3$

$J = 3v$

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

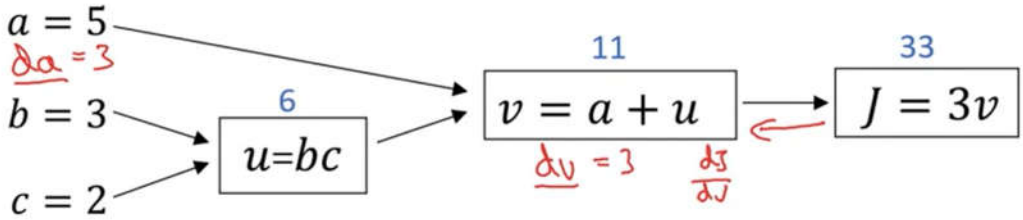
Andrew Ng

Computing derivatives



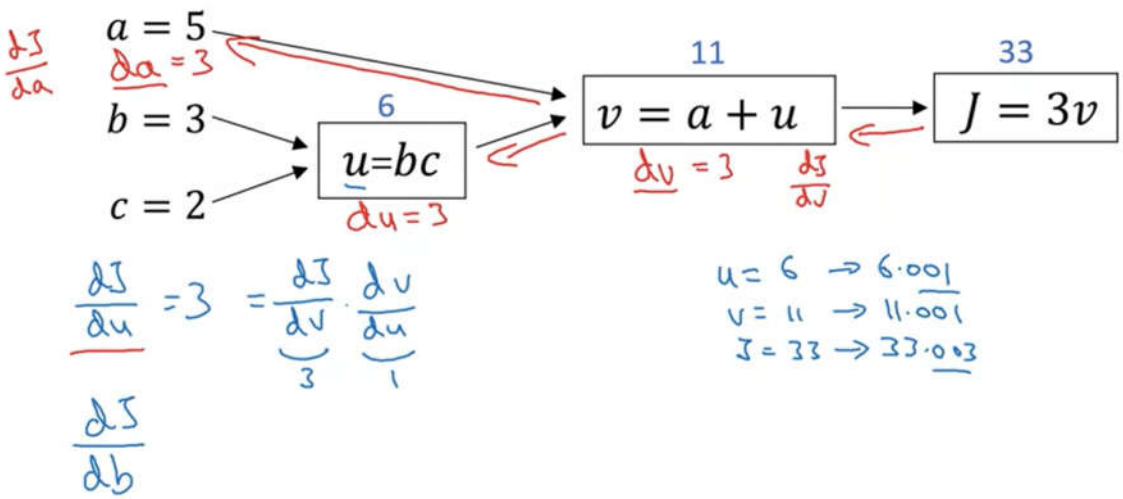
Andrew Ng

Computing derivatives



Andrew Ng

Computing derivatives



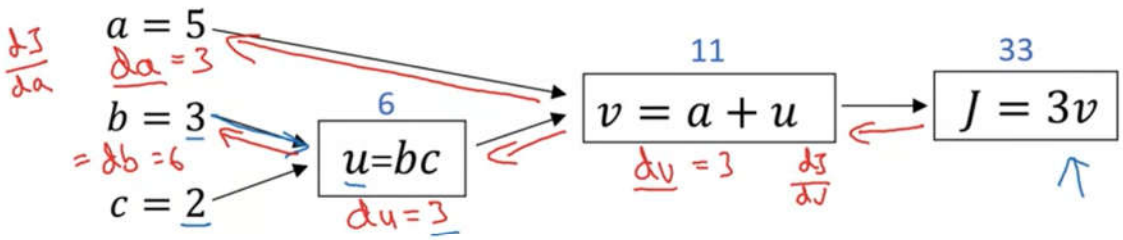
$$\frac{dJ}{du} = 3 = \underbrace{\frac{dJ}{dv}}_{3} \cdot \underbrace{\frac{dv}{du}}_{1}$$

$$\frac{dJ}{db}$$

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

Andrew Ng

Computing derivatives



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

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

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

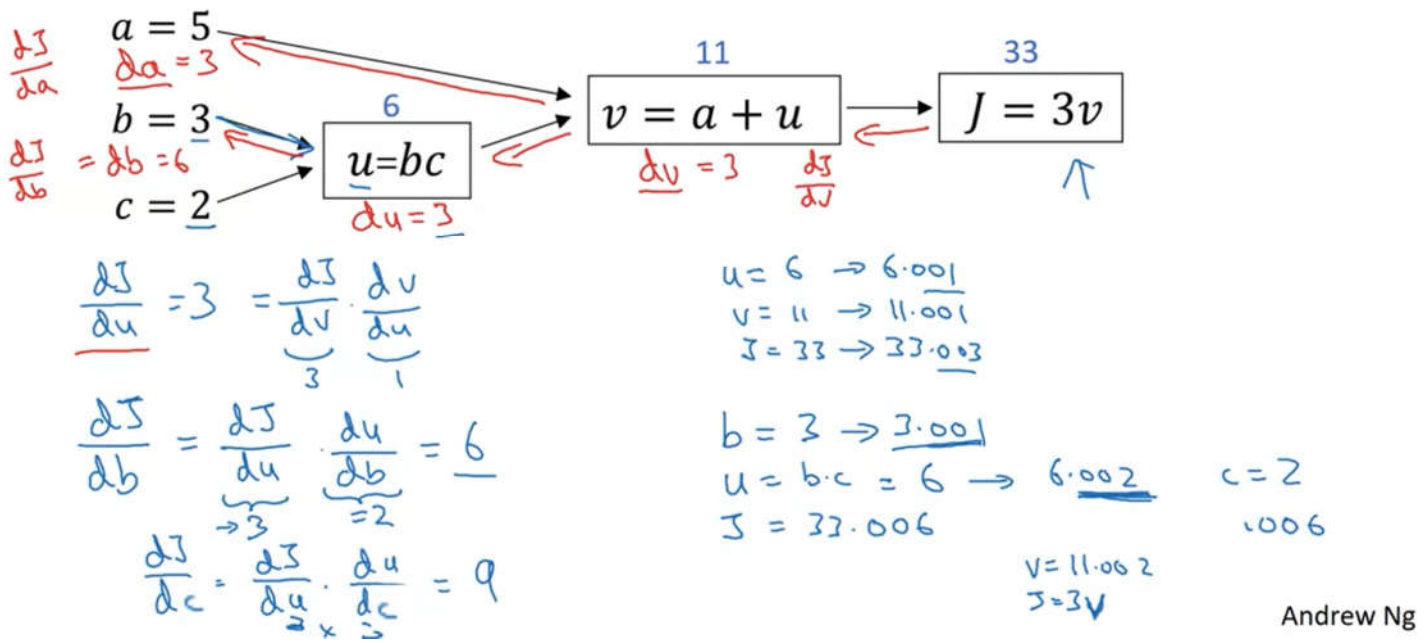
$b = 3 \rightarrow 3.001$
$u = b \cdot c = 6 \rightarrow 6.002$
$J = 33.006$

$c = 2$
$.006$

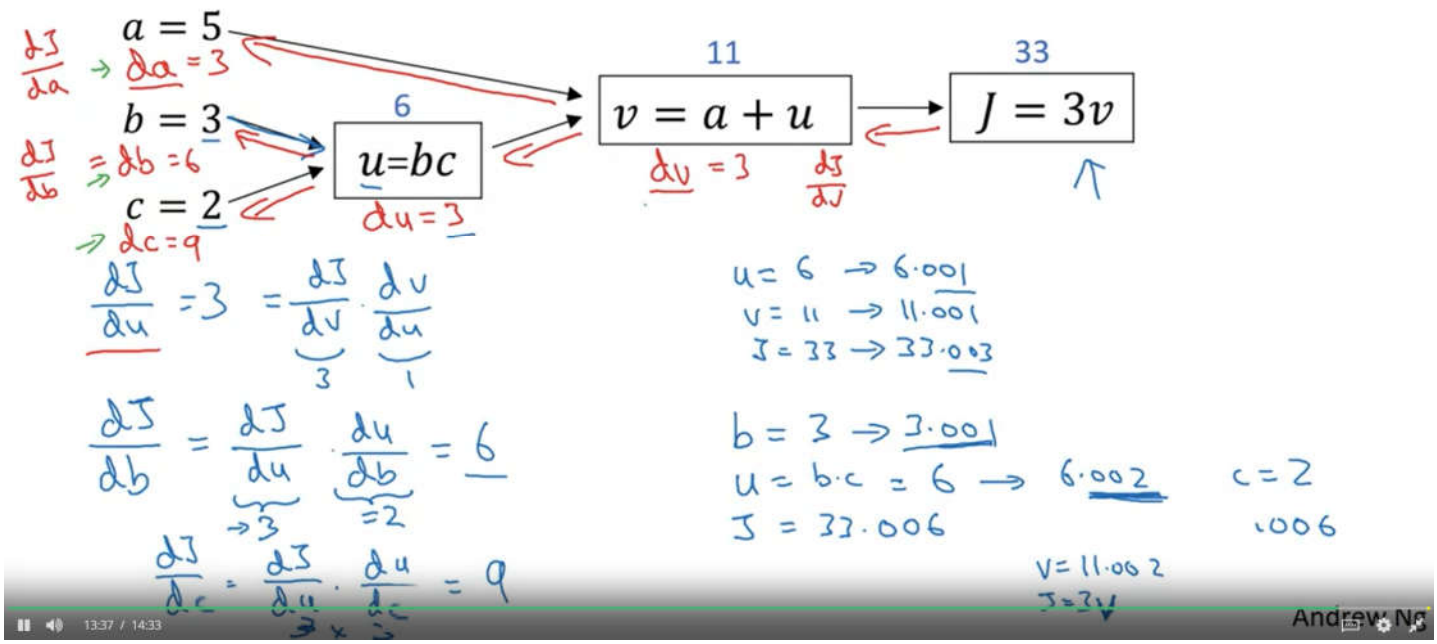
$v = 11.002$
$J = 3v$

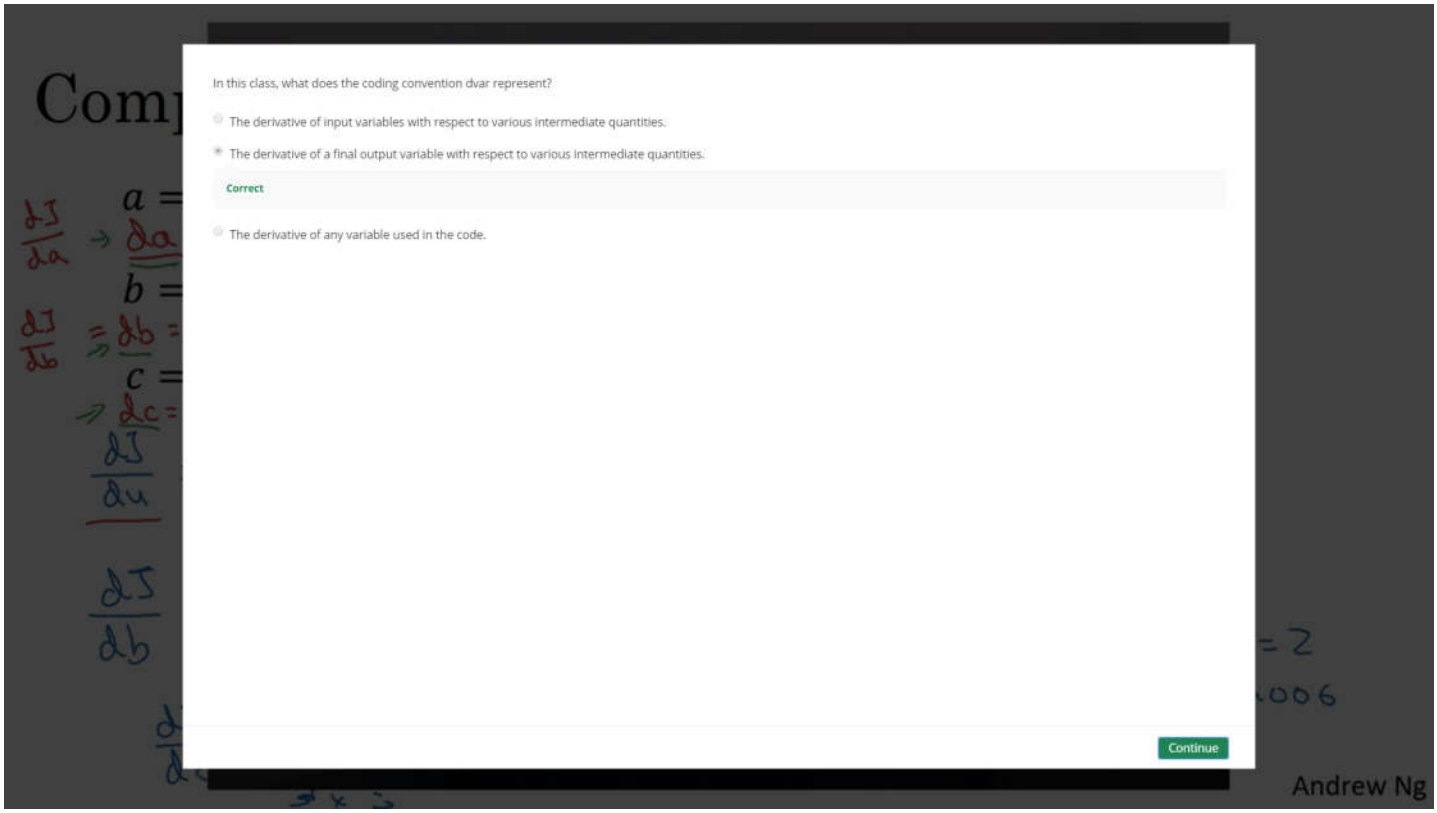
Andrew Ng

Computing derivatives



Computing derivatives





Press Esc to exit full screen

deeplearning.ai

Basics of Neural Network Programming

Logistic Regression Gradient descent

Logistic regression recap

$$z = w^T x + b$$
$$\hat{y} = a = \sigma(z)$$
$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

To that, let's dive into gradient descent for logistic regression.

Andrew Ng

Logistic regression recap

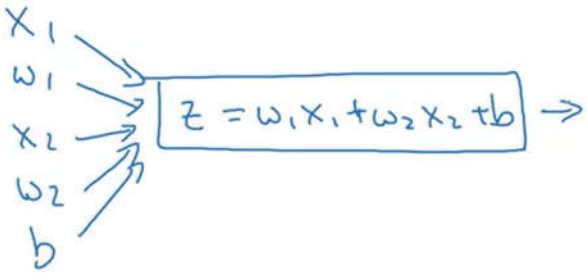
$$z = w^T x + b$$
$$\hat{y} = a = \sigma(z)$$
$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

Let's write this out as a computation graph and for this example,

Andrew Ng

Logistic regression recap

$$z = w^T x + b$$
$$\hat{y} = a = \sigma(z)$$
$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

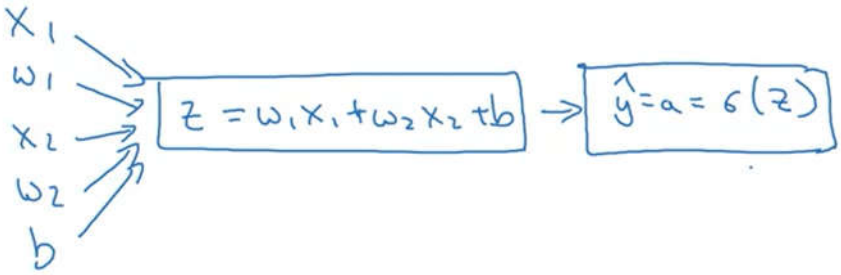


Then, we compute Y_hat,

Andrew Ng

Logistic regression recap

$$z = w^T x + b$$
$$\hat{y} = a = \sigma(z)$$
$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

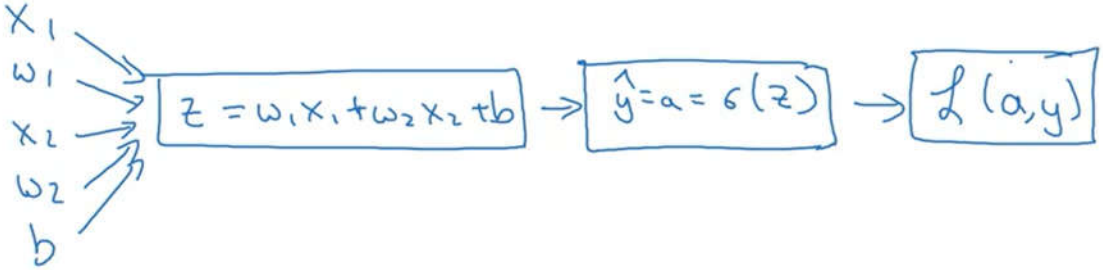


that's the next step in the computation graph, and then, finally,

Andrew Ng

Logistic regression recap

$$z = w^T x + b$$
$$\hat{y} = a = \sigma(z)$$
$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

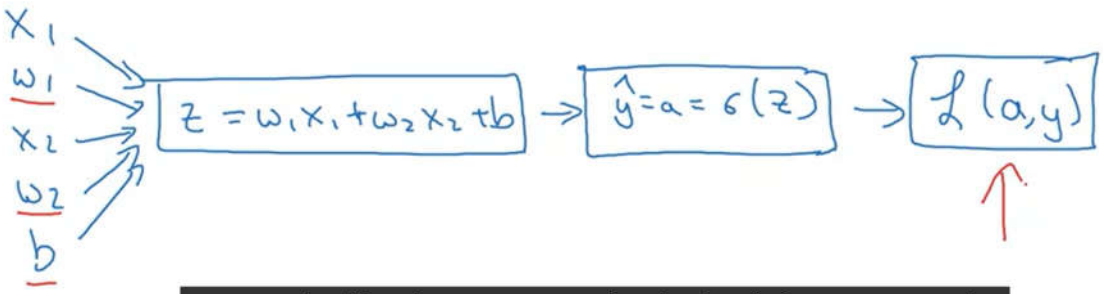


and I won't copy the formula again.

Andrew Ng

Logistic regression recap

$$z = w^T x + b$$
$$\hat{y} = a = \sigma(z)$$
$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

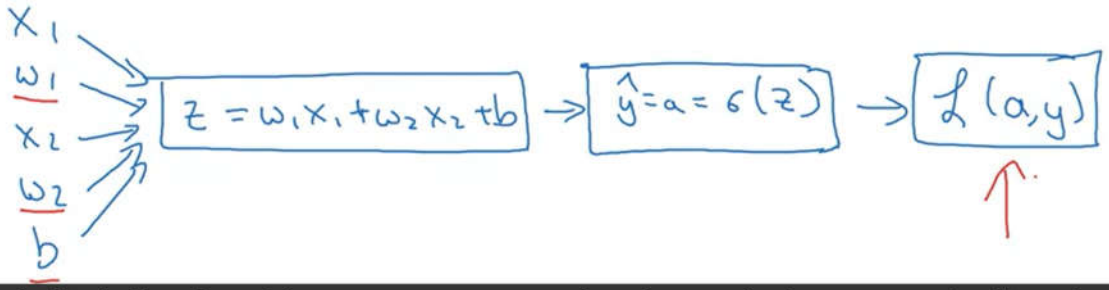


compute the loss on a single training example,

Andrew Ng

Logistic regression recap

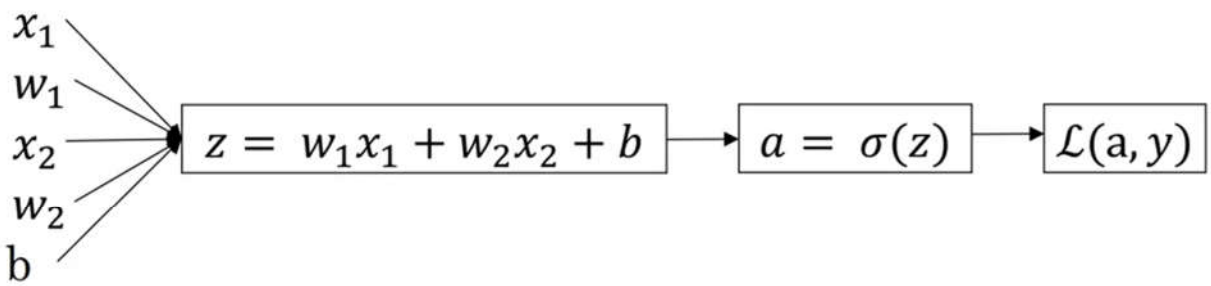
$$z = w^T x + b$$
$$\hat{y} = a = \sigma(z)$$
$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$



now let's talk about how you can go backwards to compute the derivatives.

Andrew Ng

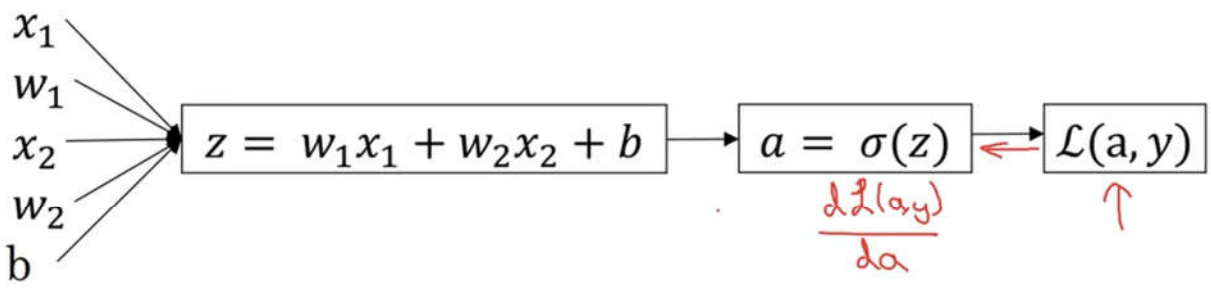
Logistic regression derivatives



Here's a cleaned-up version of the diagram.

Andrew Ng

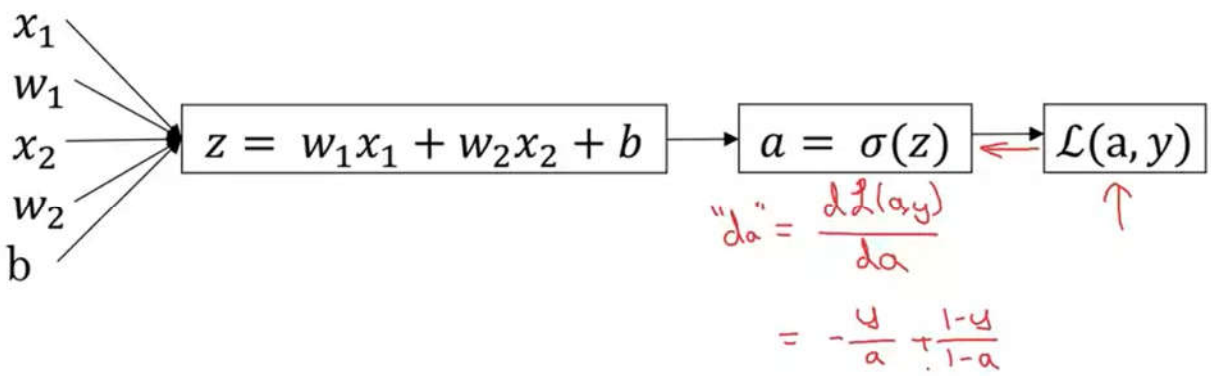
Logistic regression derivatives



So, in the code,

Andrew Ng

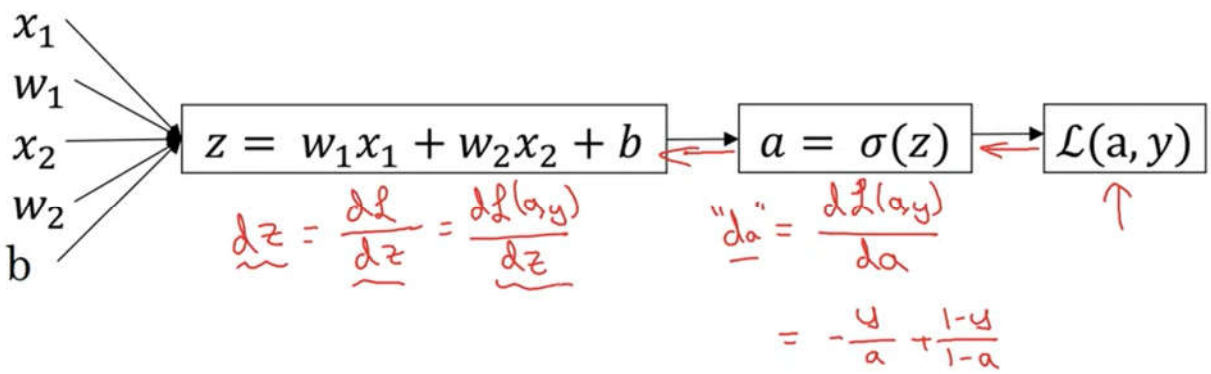
Logistic regression derivatives



And the way you do that is you take the formula for the loss and,

Andrew Ng

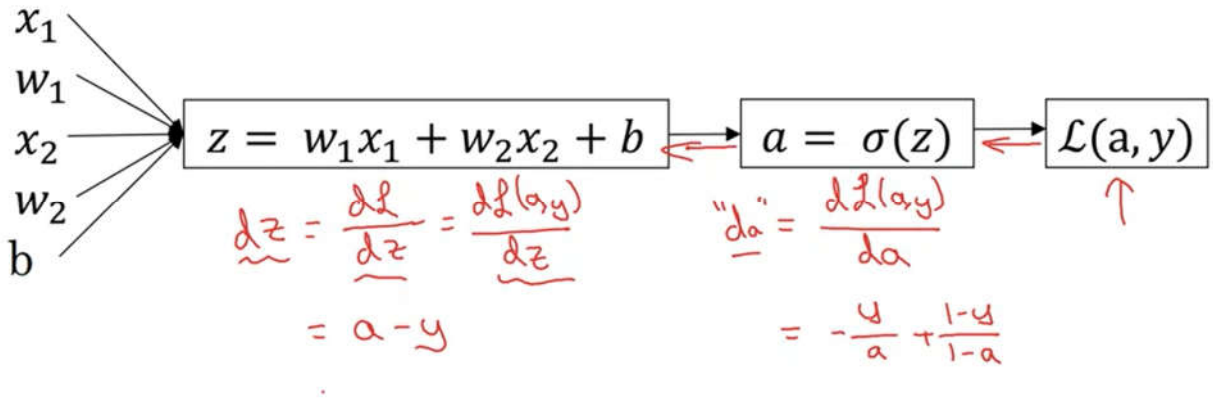
Logistic regression derivatives



Either type of notation is equally acceptable.

Andrew Ng

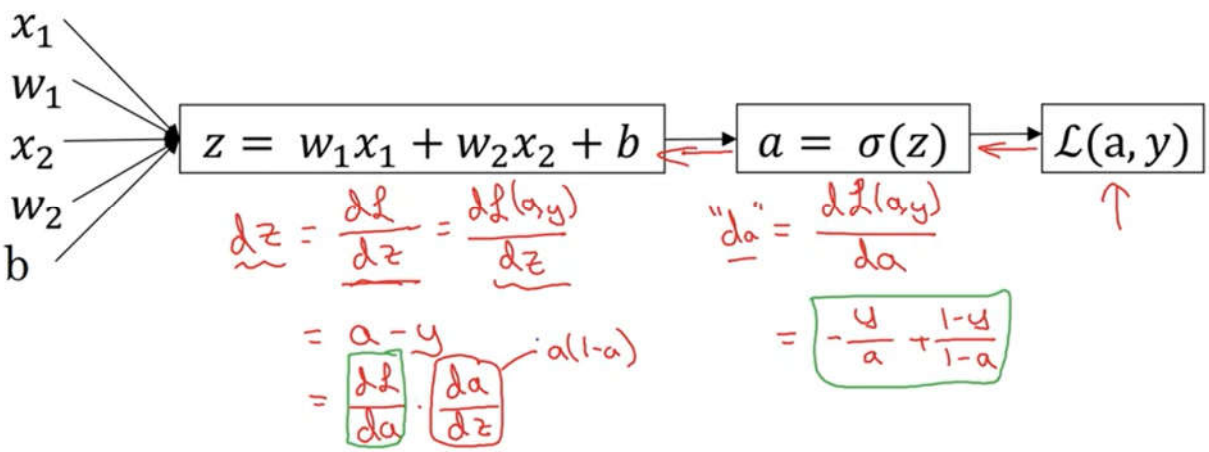
Logistic regression derivatives



Just a couple of comments only for those of you experts in calculus,

Andrew Ng

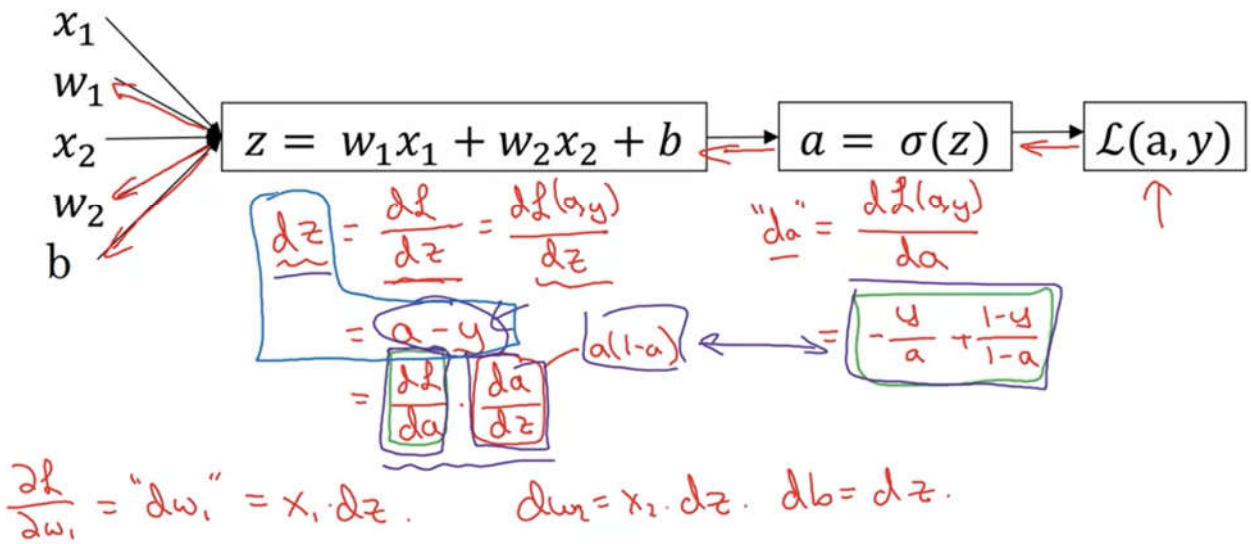
Logistic regression derivatives



if you take these two quantities, DL DA,

Andrew Ng

Logistic regression derivatives



Andrew Ng

In this video, what is the simplified formula for the derivative of the losswith respect to z?
a (1 - y)
a - y
Correct
a / (1-a)
Continue
Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Gradient descent on m examples

Logistic regression on m examples

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

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

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

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

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

Andrew Ng

Logistic regression on m examples

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

For $i=1$ to m

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

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

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

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

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

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

Andrew Ng

Logistic regression on m examples

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

For $i=1$ to m

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

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

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

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

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

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

$n=2$

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

$J /= m$

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

Andrew Ng

Logistic regression on m examples

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

For $i=1$ to m

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

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

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

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

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

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

$n=2$

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

$J /= m$

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

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

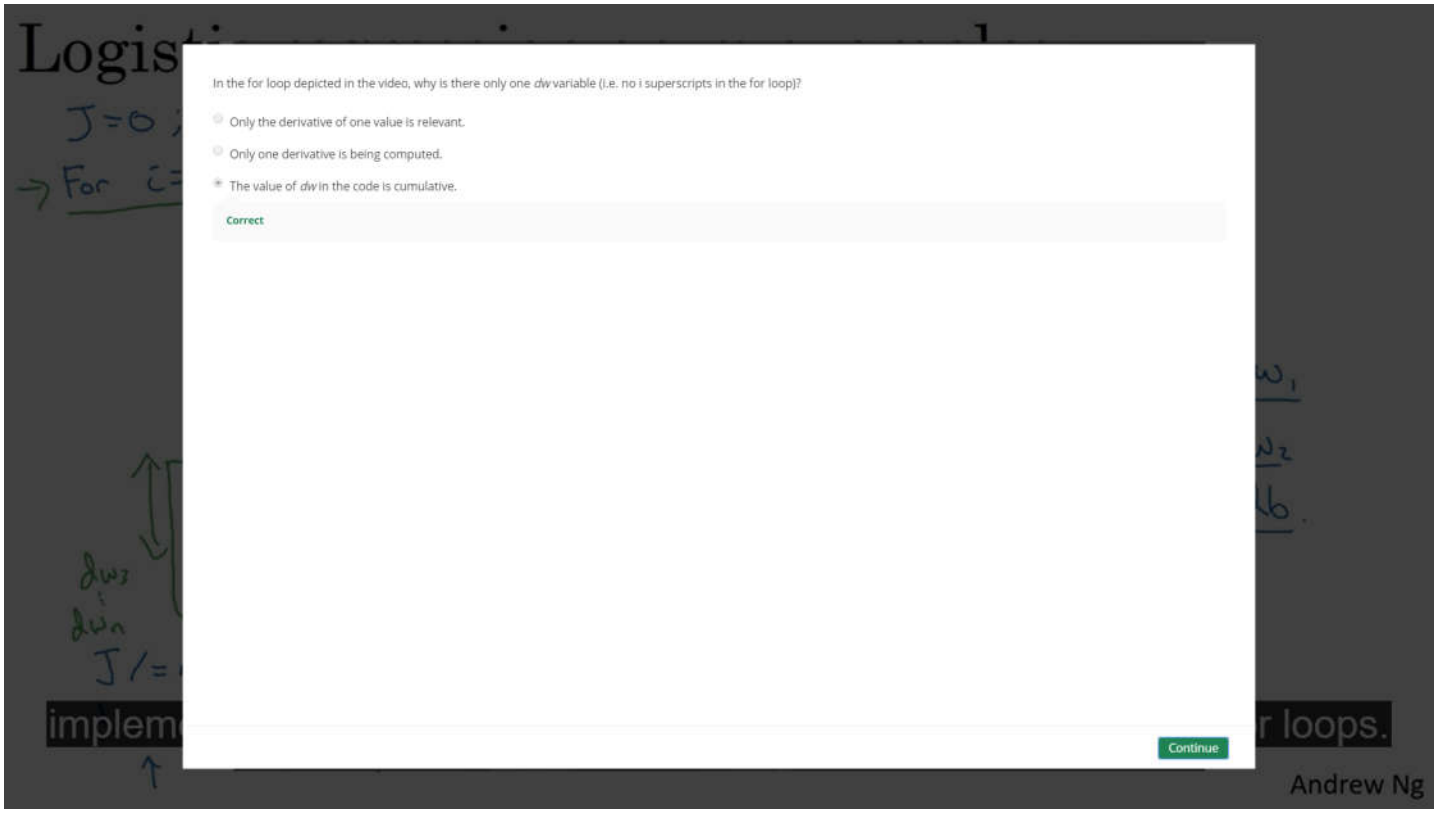
$w_1 := w_1 - \alpha\, dw_1$

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

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

5:30 / 8:00

Andrew Ng



Logistic regression on m examples

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

For $i=1$ to m

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

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

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

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

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

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

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

$n=2$

dw_3
$\vdots$
dw_n

$J/=m$

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

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

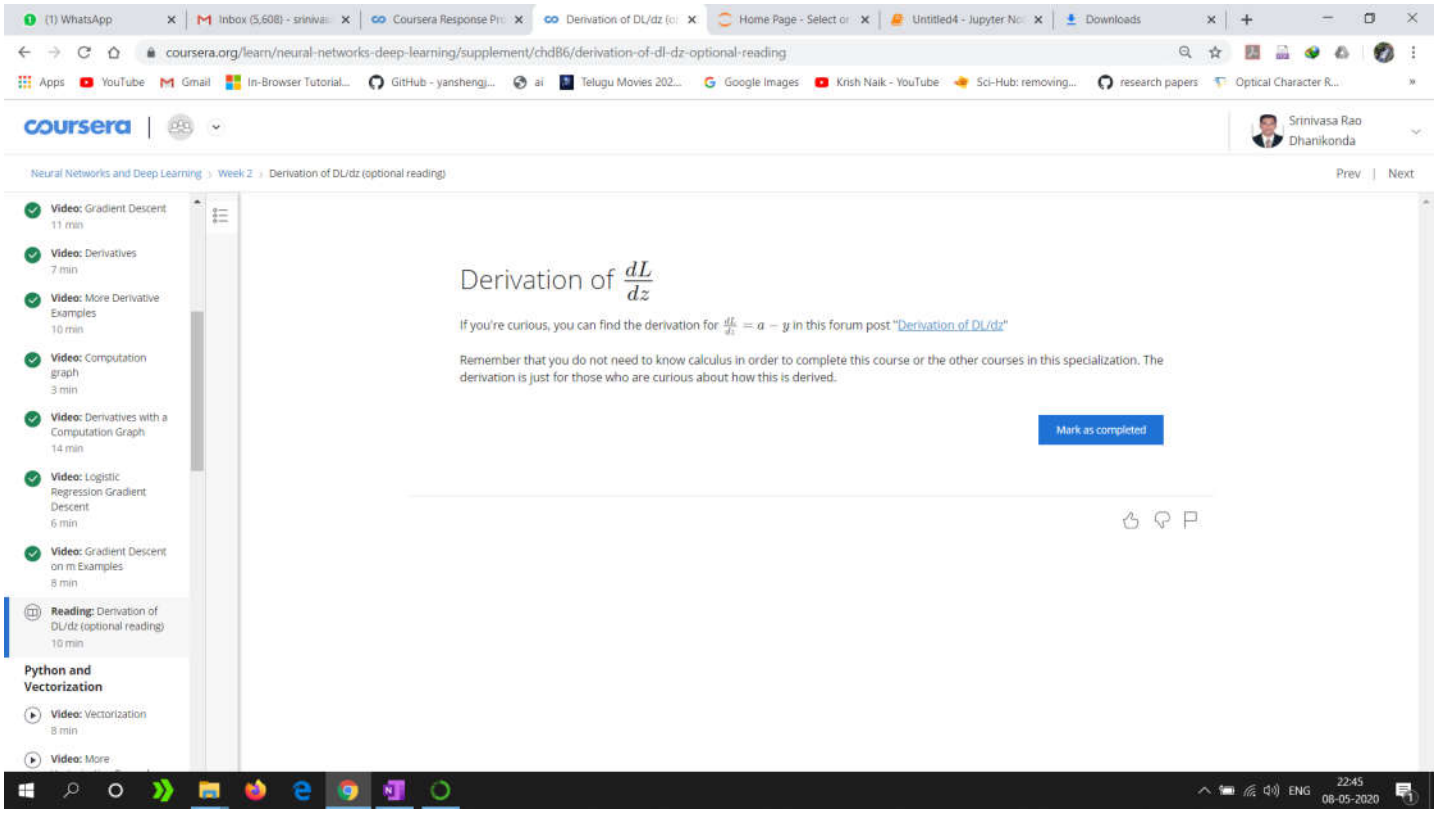
$w_1 := w_1 - \alpha\, dw_1$

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

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

Vectorization

Andrew Ng



Basics of Neural Network Programming

Vectorization

deeplearning.ai

0:02 / 8:04

What is vectorization?

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

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

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

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

Non-vectorized:

if you had a non-vectorized implementation,

Andrew Ng

What is vectorization?

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

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

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

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

Non-vectorized:

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

Vectorized

In contrast, a vectorized implementation would just compute W transpose X directly.

Andrew Ng

What is vectorization?

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

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

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

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

Non-vectorized:

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

Vectorized

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

X, so this computes W transpose X.

Andrew Ng

What is vectorization?

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

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

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

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

Non-vectorized:

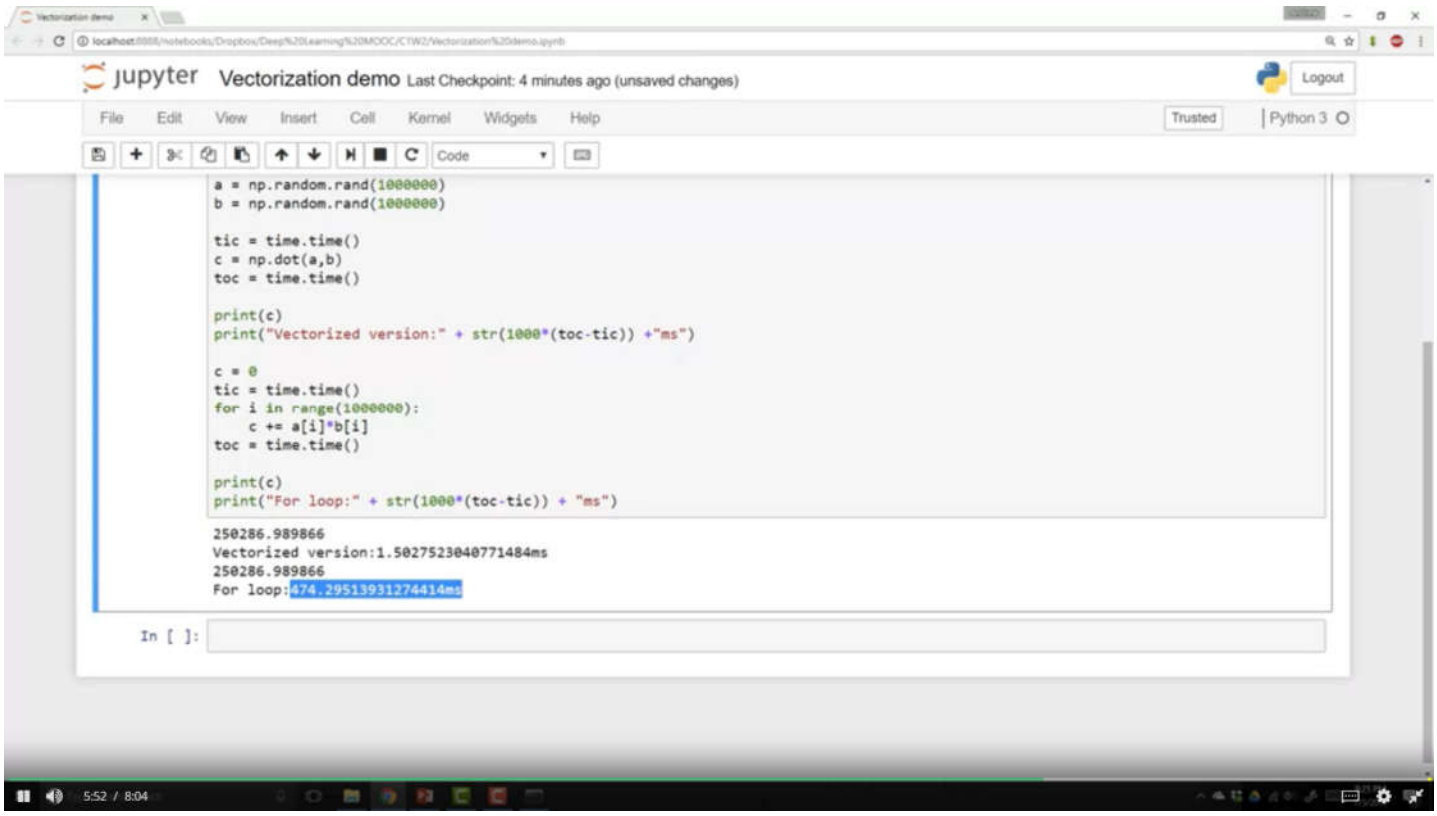
```
z = 0
for i in range(n-x):
    z += w[i] * x[i]
z += b
```

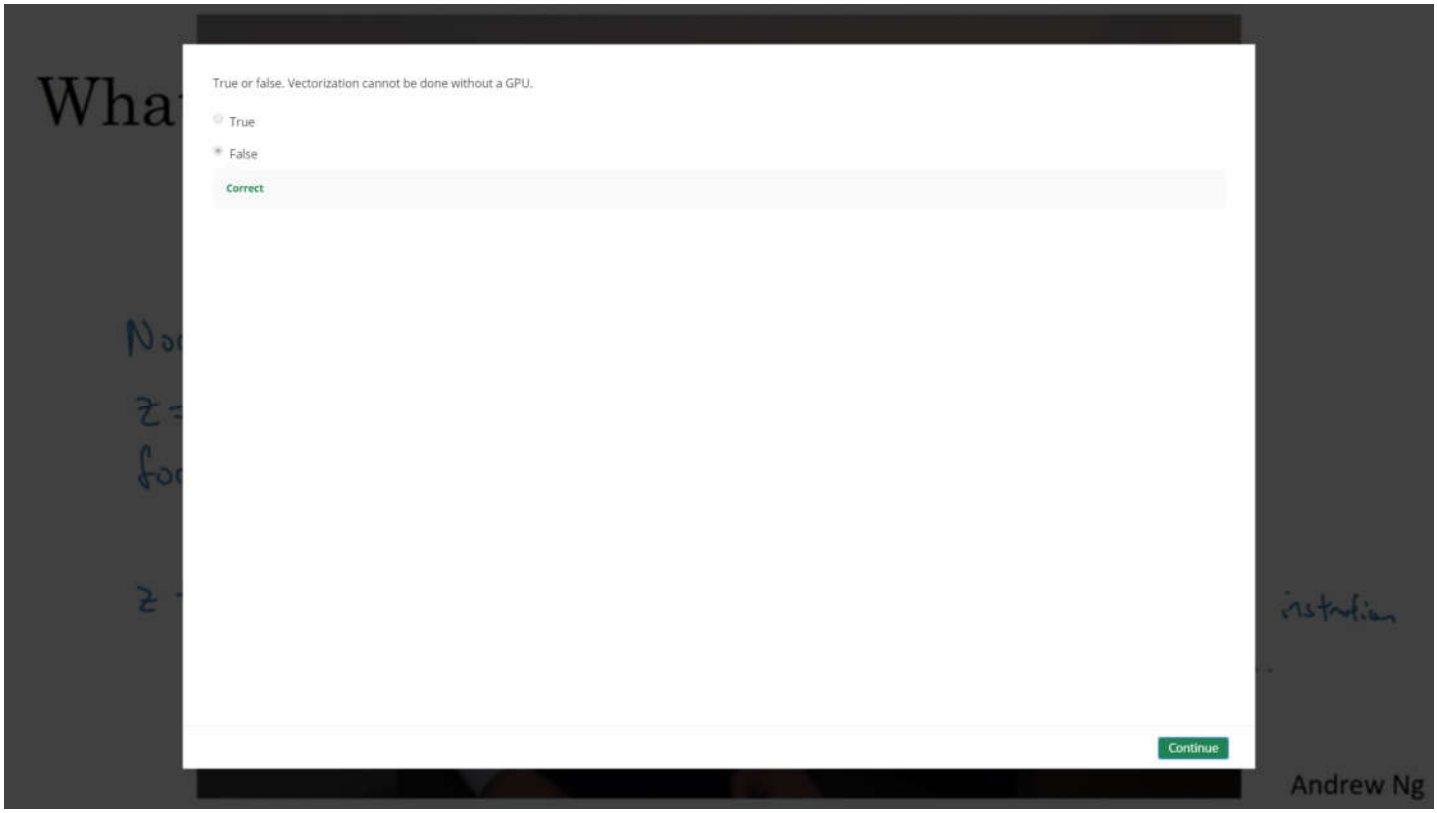
Vectorized

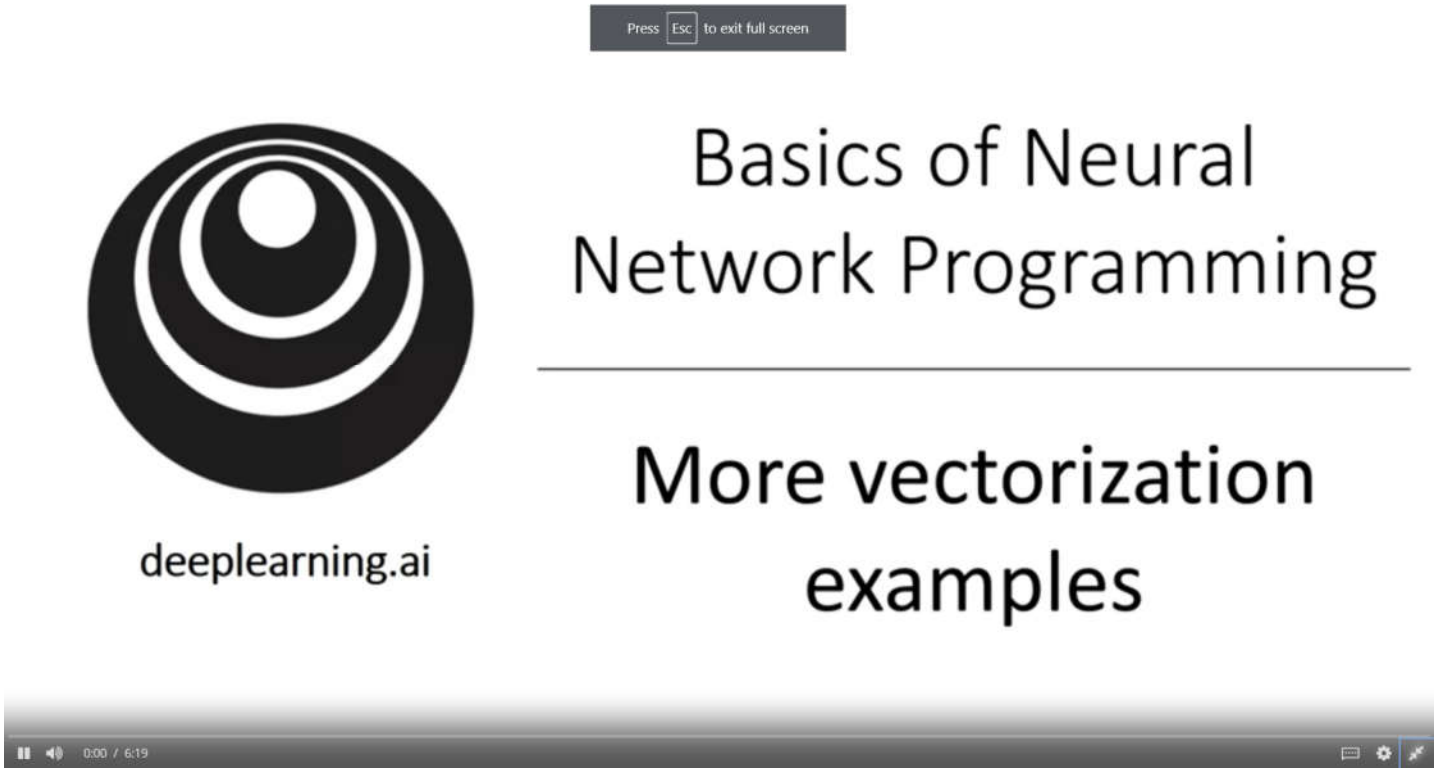
$z = \underbrace{np.dot(w,x)}_{w^T x} + b$

Let's actually illustrate this with a little demo.

Andrew Ng




True or false. Vectorization cannot be done without a GPU.
True
False
Correct
Continue
Andrew Ng


Press Esc to exit full screen
Basics of Neural Network Programming
More vectorization examples
deeplearning.ai
0:00 / 6:19

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

Andrew Ng

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]
```

So now, this is two for-loops,
looping over both i and j.

Andrew Ng

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

$u = Av$

u = np.dot(A,v)

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[i]
```

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

now eliminates two different for-loops,
and it's going to be way faster.

Andrew Ng

Vectors and matrix valued functions

Say you need to apply the exponential operation on every element of a matrix/vector.

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$$

Let's say you already have a vector,
v, in memory and you

Andrew Ng

Vectors and matrix valued functions

Say you need to apply the exponential operation on every element of a matrix/vector.

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \qquad u = \begin{bmatrix} e^{v_1} \\ e^{v_2} \\ \vdots \\ e^{v_n} \end{bmatrix}$$

```
u = np.zeros((n,1))
for i in range(n):
    u[i]=math.exp(v[i])
```

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)
```

Andrew Ng

Vectors and matrix valued functions

Say you need to apply the exponential operation on every element of a matrix/vector.

$$v = \begin{bmatrix} v_1 \\ \vdots \end{bmatrix} \rightarrow u = \begin{bmatrix} e^{v_1} \\ e^{v_2} \\ \vdots \end{bmatrix}$$

```
import numpy as np
u = np.exp(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

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

for $i = 1$ to m:

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

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

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

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

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

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

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

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

Andrew Ng

Logistic regression derivatives

$J = 0$, $\boxed{\cancel{dw1 = 0, dw2 = 0}}$, $db = 0$ $\qquad dw = np.zeros((n_x, 1))$

for $i = 1$ to m:

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

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

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

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

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

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

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

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

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

Andrew Ng

Logistic regression derivatives

$J = 0$, $\boxed{\cancel{dw1 = 0, dw2 = 0}}$, $db = 0$ $\qquad dw = np.zeros((n_x, 1))$

for $i = 1$ to m:

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

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

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

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

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

$\boxed{\cancel{dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}} \quad \cancel{dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}}}$ $\quad n_x = 2$ $\qquad dw \mathrel{+}= x^{(i)} dz^{(i)}$

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

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

Andrew Ng

Logistic regression derivatives

$J = 0$, $\boxed{\cancel{dw1 = 0, dw2 = 0}}$, $db = 0$ $\qquad dw = np.zeros((n_x, 1))$

for $i = 1$ to m:

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

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

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

Logistic regression derivatives

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

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

for i = 1 to m:

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

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

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

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

for $j = 1 \ldots n_x$

$dw_j \mathrel{+}= \ldots$

$\boxed{\cancel{dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}}, \cancel{dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}}}$ $n_x = 2$

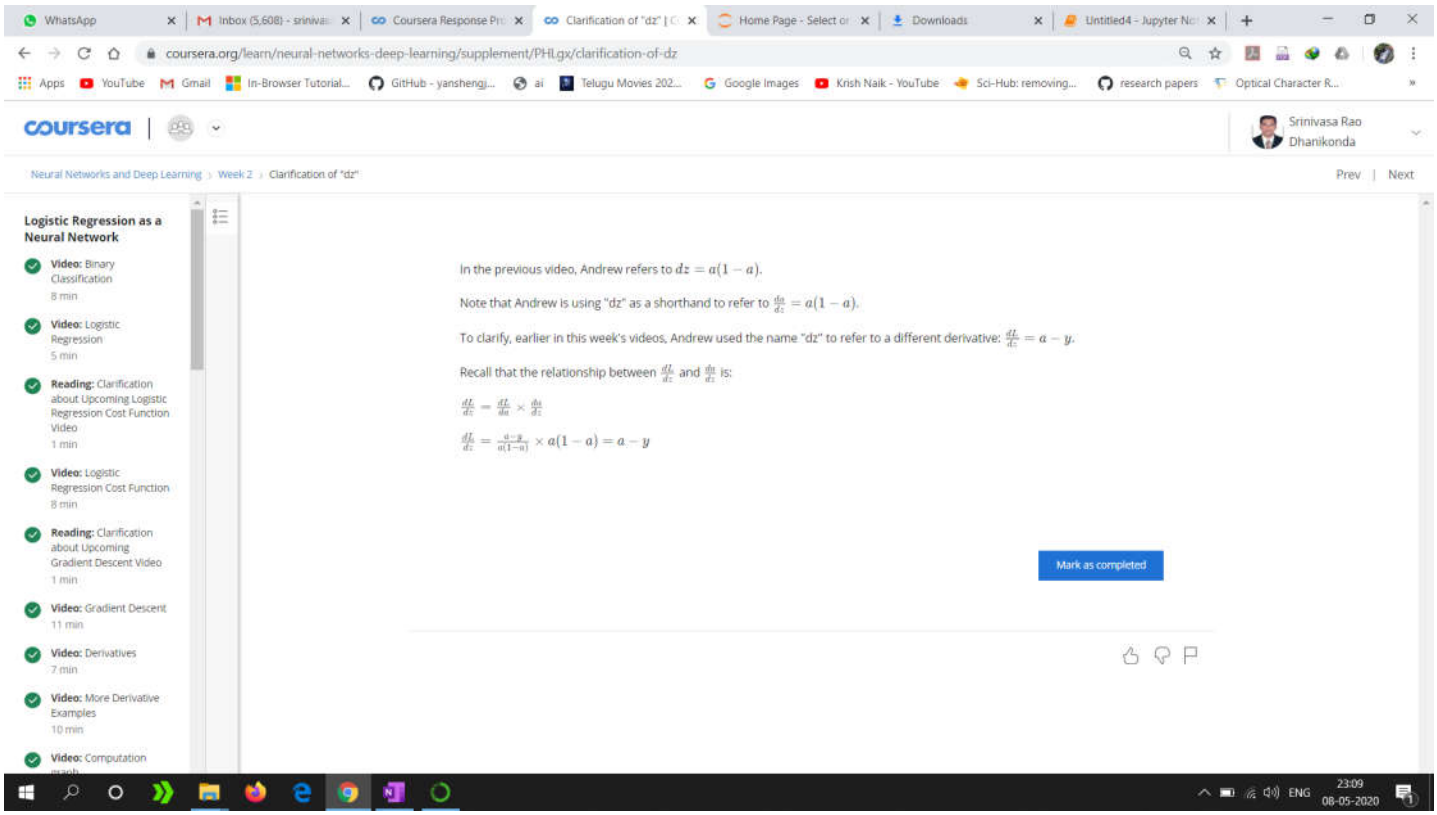
$dw \mathrel{+}= x^{(i)} dz^{(i)}$

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

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

$dw \mathrel{/}= m.$

Andrew Ng



Press Esc to exit full screen

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

0:00 / 7:32

Vectorizing Logistic Regression

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

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

Andrew Ng

Vectorizing Logistic Regression

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

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

Vectorizing Logistic Regression

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

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

Andrew Ng

Vectorizing Logistic Regression

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

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

Andrew Ng

Vectorizing Logistic Regression

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

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

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

Andrew Ng

Vectorizing Logistic Regression

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

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

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

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

Andrew Ng

Vectorizing Logistic Regression

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

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

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

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

Andrew Ng

Vectorizing Logistic Regression

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

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

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

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

Andrew Ng

Vectorizing Logistic Regression

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

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

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

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

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

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

Andrew Ng

Vectorizing Logistic Regression

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

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

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

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

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

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

"Broadcasting"

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

Andrew Ng

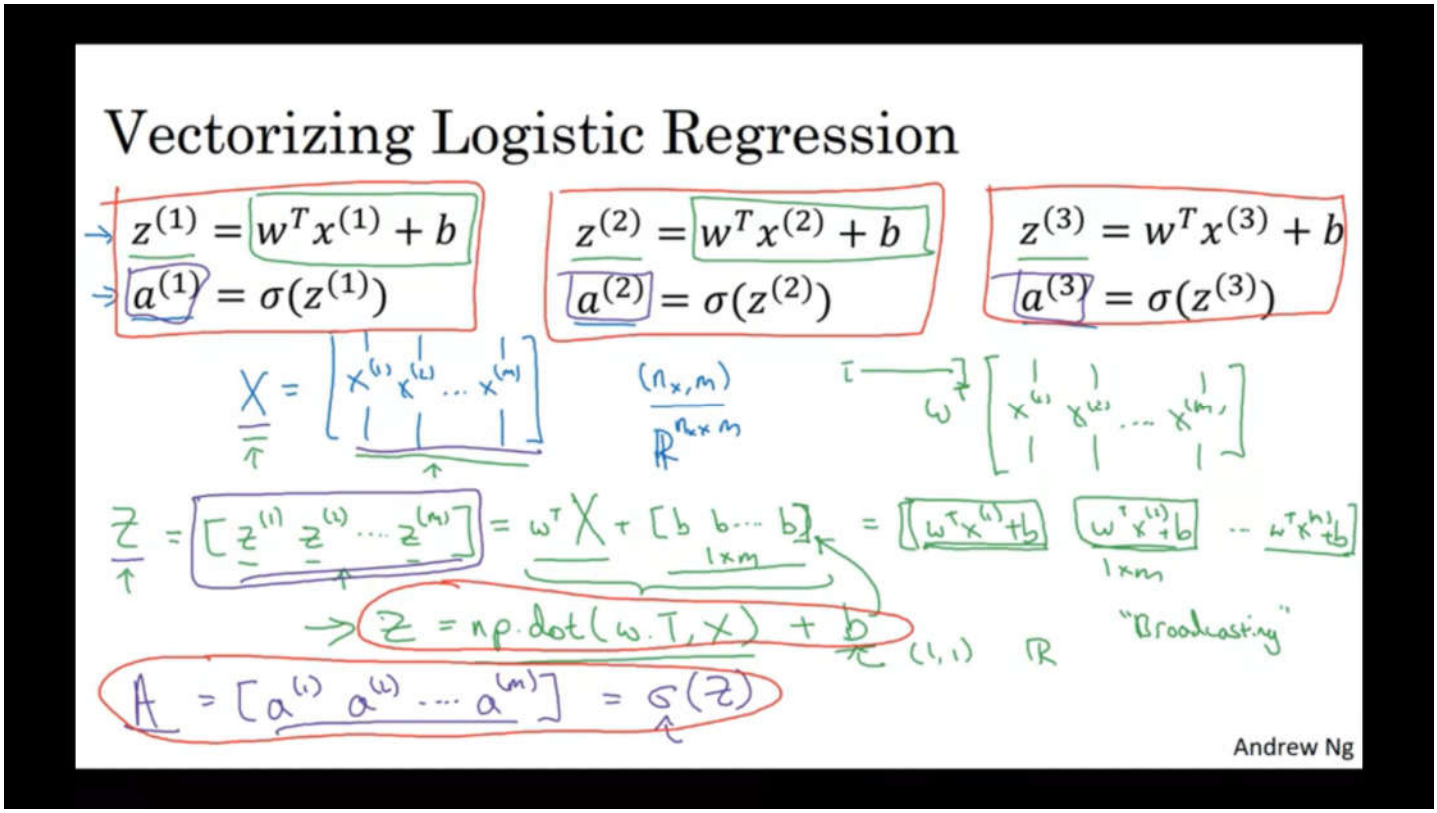
Vectorizing Logistic Regression

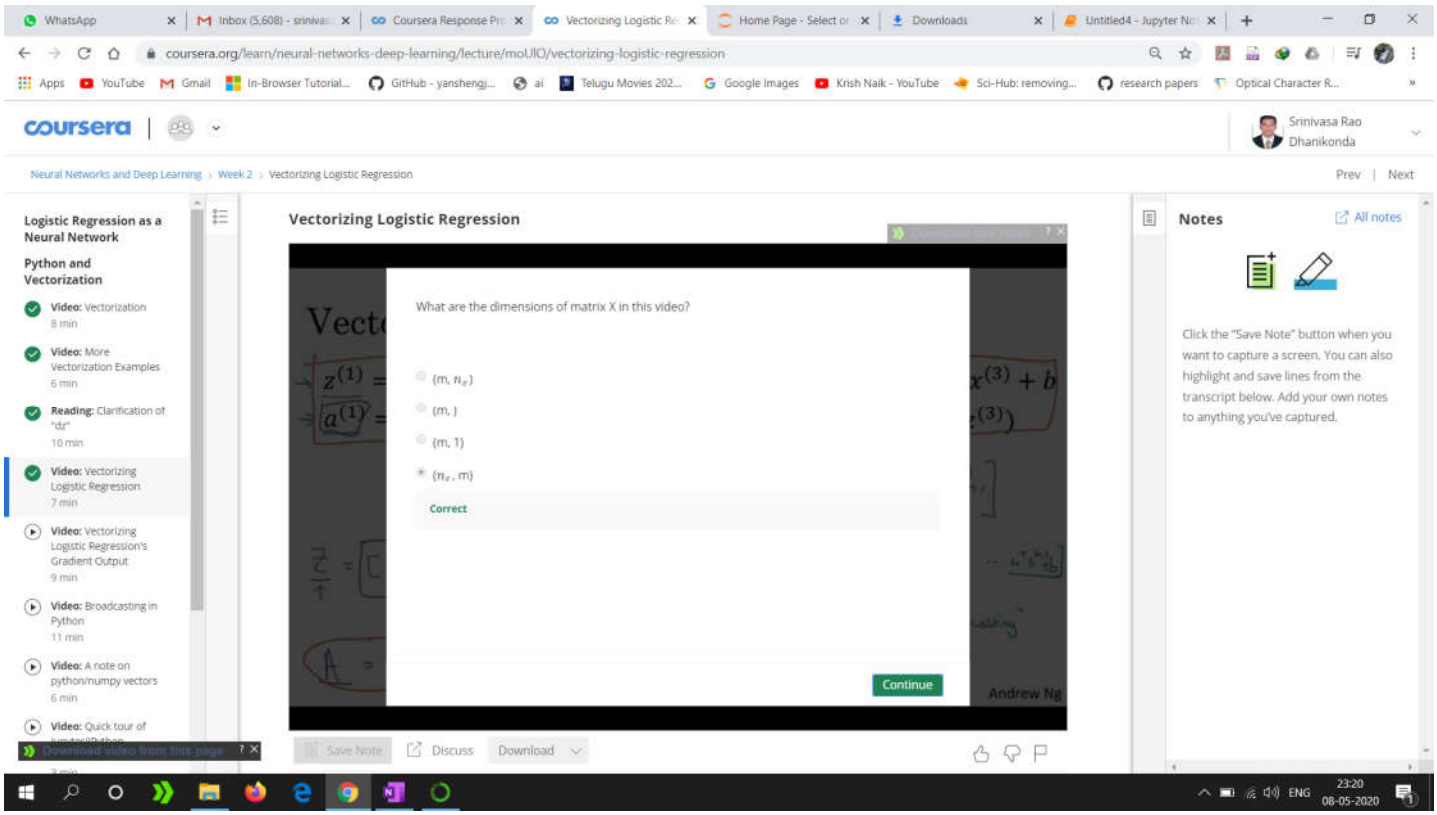
$z^{(1)} = \boxed{w^T x^{(1)} + b}$ $\quad z^{(2)} = \boxed{w^T x^{(2)} + b}$ $\quad z^{(3)} = w^T x^{(3)} + b$

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

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

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


Vectorizing Logistic Regression
$z^{(1)} = w^T x^{(1)} + b$
$a^{(1)} = \sigma(z^{(1)})$
$z^{(2)} = w^T x^{(2)} + b$
$a^{(2)} = \sigma(z^{(2)})$
$z^{(3)} = w^T x^{(3)} + b$
$a^{(3)} = \sigma(z^{(3)})$
$X = [x^{(1)}\ x^{(2)} \dots x^{(m)}]$
(n_x, m)
$\mathbb{R}^{n_x \times m}$
$Z = [z^{(1)}\ z^{(2)} \dots z^{(m)}] = w^T X + [b\ b \dots b]$
$1 \times m$
$= [w^T x^{(1)} + b\ \ w^T x^{(2)} + b \dots w^T x^{(m)} + b]$
$Z = np.dot(w.T, X) + b$
$(1,1)$
$\mathbb{R}$
"Broadcasting"
$A = [a^{(1)}\ a^{(2)} \dots a^{(m)}] = \sigma(Z)$
Andrew Ng


coursera.org/learn/neural-networks-deep-learning/lecture/moUlO/vectorizing-logistic-regression
coursera
Neural Networks and Deep Learning > Week 2 > Vectorizing Logistic Regression
Prev | Next
Logistic Regression as a Neural Network
Python and Vectorization
Video: Vectorization 8 min
Video: More Vectorization Examples 6 min
Reading: Clarification of "dz" 10 min
Video: Vectorizing Logistic Regression 7 min
Video: Vectorizing Logistic Regression's Gradient Output 9 min
Video: Broadcasting in Python 11 min
Video: A note on python/numpy vectors 6 min
Video: Quick tour of
Vectorizing Logistic Regression
What are the dimensions of matrix X in this video?
(m, n_x)
$(m,)$
$(m, 1)$
(n_x, m)
Correct
Continue
Andrew Ng
Save Note
Discuss
Download
Notes
All notes
Click the "Save Note" button when you want to capture a screen. You can also highlight and save lines from the transcript below. Add your own notes to anything you've captured.

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression's Gradient Computation

Vectorizing Logistic Regression

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

$$\boxed{dZ = [\underbrace{dz^{(1)}\ dz^{(2)} \cdots dz^{(m)}}_{1\times m}]}$$

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

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

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

Andrew Ng

Vectorizing Logistic Regression

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

$$\boxed{dZ = [\underbrace{dz^{(1)}\ dz^{(2)} \cdots dz^{(m)}}_{1\times m}]} \leftarrow$$

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

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

$\rightarrow dw = 0$

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

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

$\vdots$

$dw /= m$

$db = 0$

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

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

$\vdots$

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

$db /= m.$

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

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

Andrew Ng

Vectorizing Logistic Regression

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

$$\boxed{dZ = [\underbrace{dz^{(1)}\ dz^{(2)} \cdots dz^{(m)}}_{1\times m}]} \leftarrow$$

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

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

$\rightarrow dw = 0$

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

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

$\vdots$

$dw /= m$

$db = 0$

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

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

$\vdots$

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

$db /= m.$

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

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

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

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

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

Andrew Ng

Vectorizing Logistic Regression

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

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

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

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

→ $dw = 0$

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

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

$\vdots$

$dw /= m$

$db = 0$

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

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

$\vdots$

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

$db /= m.$

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

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

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

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

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

$n \times 1$

Andrew Ng

Implementing Logistic Regression

```
J = 0, dw1 = 0, dw2 = 0, db = 0
for i = 1 to m:
    z(i) = w^T x(i) + b
    a(i) = σ(z(i))
    J += -[y(i) log a(i) + (1 - y(i)) log(1 - a(i))]
    dz(i) = a(i) - y(i)
    dw1 += x1(i) dz(i)
    dw2 += x2(i) dz(i)
    db += dz(i)
J = J/m, dw1 = dw1/m, dw2 = dw2/m
db = db/m
```

Andrew Ng

Implementing Logistic Regression

```
J = 0, dw1 = 0, dw2 = 0, db = 0
for i = 1 to m:
    z(i) = w^T x(i) + b
    a(i) = σ(z(i))
```

$Z = w^T X + b$

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

$A = \sigma(Z)$

$dZ = A - Y$

Implementing Logistic Regression

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

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

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

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

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

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

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

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

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

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

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

$db = db/m$

$Z = w^T X + b$

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

$A = \sigma(Z)$

$dZ = A - Y$

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

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

Andrew Ng

Implementing Logistic Regression

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

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

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

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

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

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

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

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

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

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

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

$db = db/m$

$Z = w^T X + b$

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

$A = \sigma(Z)$

$dZ = A - Y$

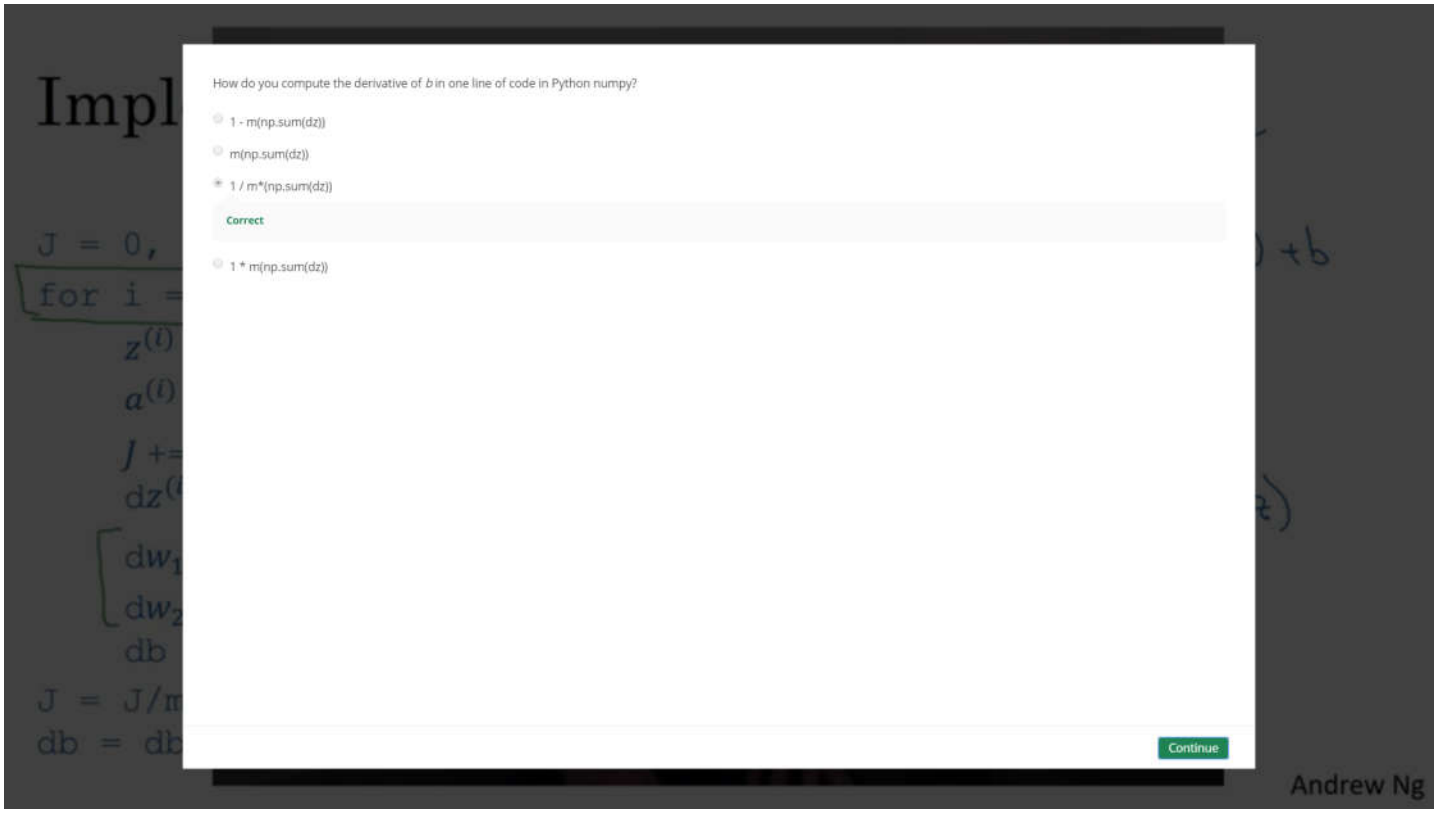
$dw = \frac{1}{m} X dZ^T$

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

$w := w - \alpha dw$

$b := b - \alpha db$

Andrew Ng



deeplearning.ai

Basics of Neural Network Programming

Broadcasting in Python

Broadcasting example

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

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

Broadcasting example

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

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

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

Calculate % of calories from Carb, Protein, Fat.

Broadcasting example

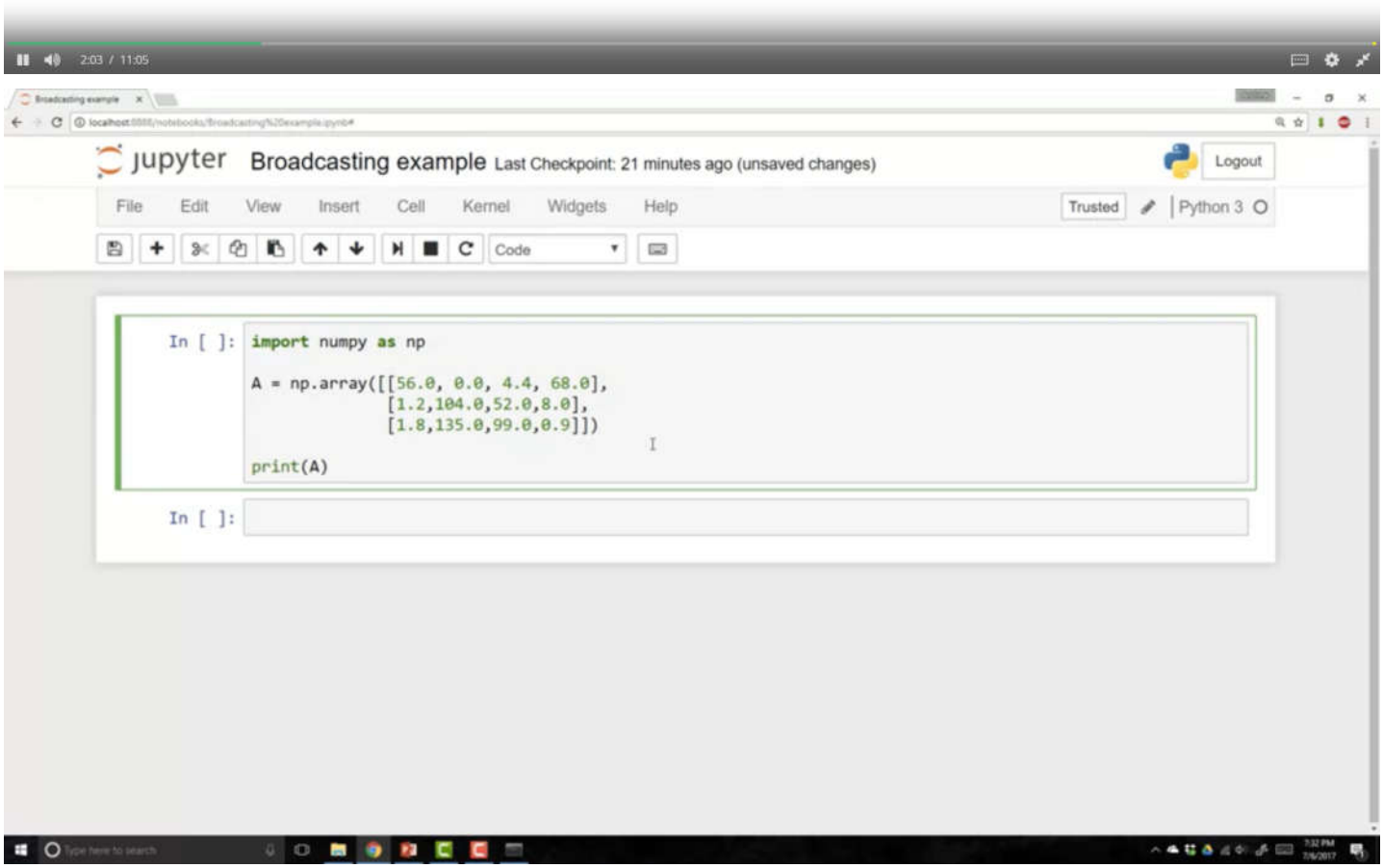
Calories from Carbs, Proteins, Fats in 100g of different foods:

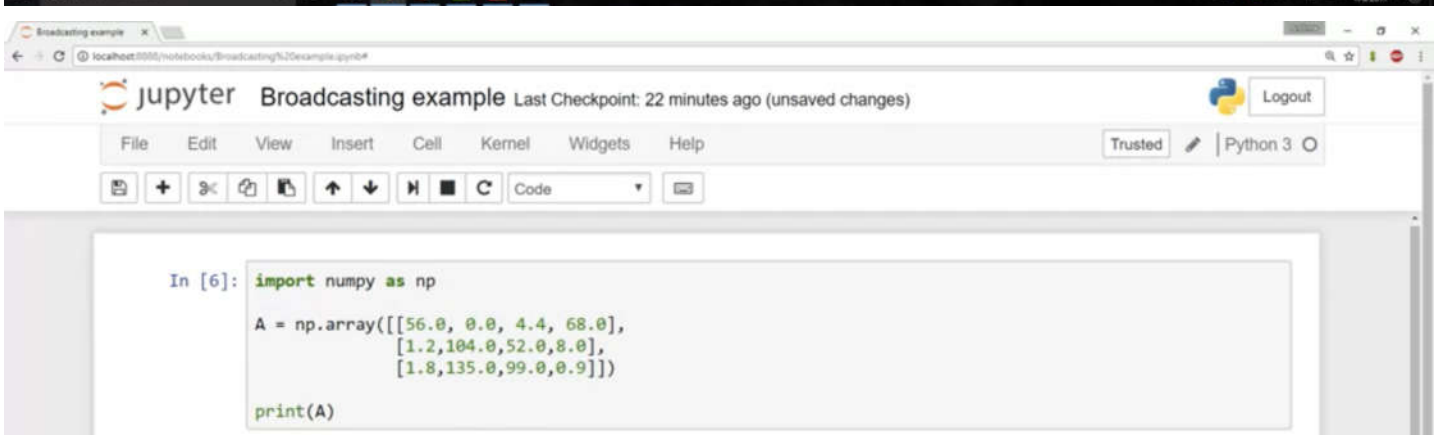
$$\begin{array}{r} \\ \text{Carb} \\ \text{Protein} \\ \text{Fat} \end{array} \begin{array}{c} \begin{array}{cccc} \text{Apples} & \text{Beef} & \text{Eggs} & \text{Potatoes} \end{array} \\ \begin{bmatrix} 56.0 & 0.0 & 4.4 & 68.0 \\ 1.2 & 104.0 & 52.0 & 8.0 \\ 1.8 & 135.0 & 99.0 & 0.9 \end{bmatrix} \end{array}$$

59 cal

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

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





jupyter Broadcasting example Last Checkpoint: 22 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

In [6]:
```
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]])

print(A)
```

```
[[  56.     0.     4.4   68. ]
 [   1.2  104.    52.     8. ]
 [   1.8  135.    99.     0.9]]
```

In [7]:
```
cal = A.sum(axis=0)
print(cal)
```

```
[  59.   239.   155.4   76.9]
```

In [ ]:

jupyter Broadcasting example Last Checkpoint: 22 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

```
              [1.2,104.0,52.0,8.0],
              [1.8,135.0,99.0,0.9]])

print(A)
```

```
[[  56.     0.     4.4   68. ]
 [   1.2  104.    52.     8. ]
 [   1.8  135.    99.     0.9]]
```

In [7]:
```
cal = A.sum(axis=0)
print(cal)
```

```
[  59.   239.   155.4   76.9]
```

In [8]:
```
percentage = 100*A/cal.reshape(1,4)
print(percentage)
```

```
[[ 94.91525424   0.           2.83140283  88.42652796]
 [  2.03389831  43.51464435  33.46203346  10.40312094]
 [  3.05084746  56.48535565  63.70656371   1.17035111]]
```

In [ ]:

Broadcasting example

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

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

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

0

1

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))
```

Broadcasting example

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

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

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

0

1

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))
```

Broadcasting example

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

Broadcasting example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100} = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

Broadcasting example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100} = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

Broadcasting example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100} = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

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

Broadcasting example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100} = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

$$\begin{bmatrix}1 & 2 & 3\end{bmatrix} \quad \begin{bmatrix}100 & 200 & 300\end{bmatrix}$$

Broadcasting example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100} = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

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

Broadcasting example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100} = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

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

Broadcasting example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100} = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

$$\underset{(m,n)\ (2,3)}{\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}} + \underset{(1,n) \rightsquigarrow (m,n)\ (2,3)}{\begin{bmatrix}100 & 200 & 300\\100 & 200 & 300\end{bmatrix}} = \begin{bmatrix}101 & 202 & 303\\104 & 205 & 306\end{bmatrix}$$

Broadcasting example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100} = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

$$\underset{(m,n)\ (2,3)}{\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}} + \underset{(1,n) \rightsquigarrow (m,n)\ (2,3)}{\begin{bmatrix}100 & 200 & 300\\100 & 200 & 300\end{bmatrix}} = \begin{bmatrix}101 & 202 & 303\\104 & 205 & 306\end{bmatrix}$$

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

Broadcasting example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100} = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

$$\underset{(m,n)}{\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}} \underset{(2,3)}{+} \underset{(1,n) \rightsquigarrow (m,n)}{\begin{bmatrix}100 & 200 & 300\\100 & 200 & 300\end{bmatrix}} \underset{(2,3)}{=} \begin{bmatrix}101 & 202 & 303\\104 & 205 & 306\end{bmatrix}$$

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

Broadcasting example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100} = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

$$\underset{(m,n)}{\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}} \underset{(2,3)}{+} \underset{(1,n) \rightsquigarrow (m,n)}{\begin{bmatrix}100 & 200 & 300\\100 & 200 & 300\end{bmatrix}} \underset{(2,3)}{=} \begin{bmatrix}101 & 202 & 303\\104 & 205 & 306\end{bmatrix}$$

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

Broadcasting example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} \cancel{100} = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

$$\underset{(m,n)}{\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}} \underset{(2,3)}{+} \underset{(1,n) \rightsquigarrow (m,n)}{\begin{bmatrix}100 & 200 & 300\\100 & 200 & 300\end{bmatrix}} = \begin{bmatrix}101 & 202 & 303\\104 & 205 & 306\end{bmatrix}$$

$$\underset{(m,n)}{\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}} + \underset{\substack{(m,1)\\ \Downarrow \\ (m,n)}}{\begin{bmatrix}100\\200\end{bmatrix}} \begin{bmatrix}100 & 100\\200 & 200\end{bmatrix} = \begin{bmatrix}101 & 102 & 103\\204 & 205 & 206\end{bmatrix}$$

General Principle

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

General Principle

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

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

General Principle

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

matrix $\quad - \quad$

$\quad * \quad$

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

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

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

General Principle

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

matrix $\quad - \quad$

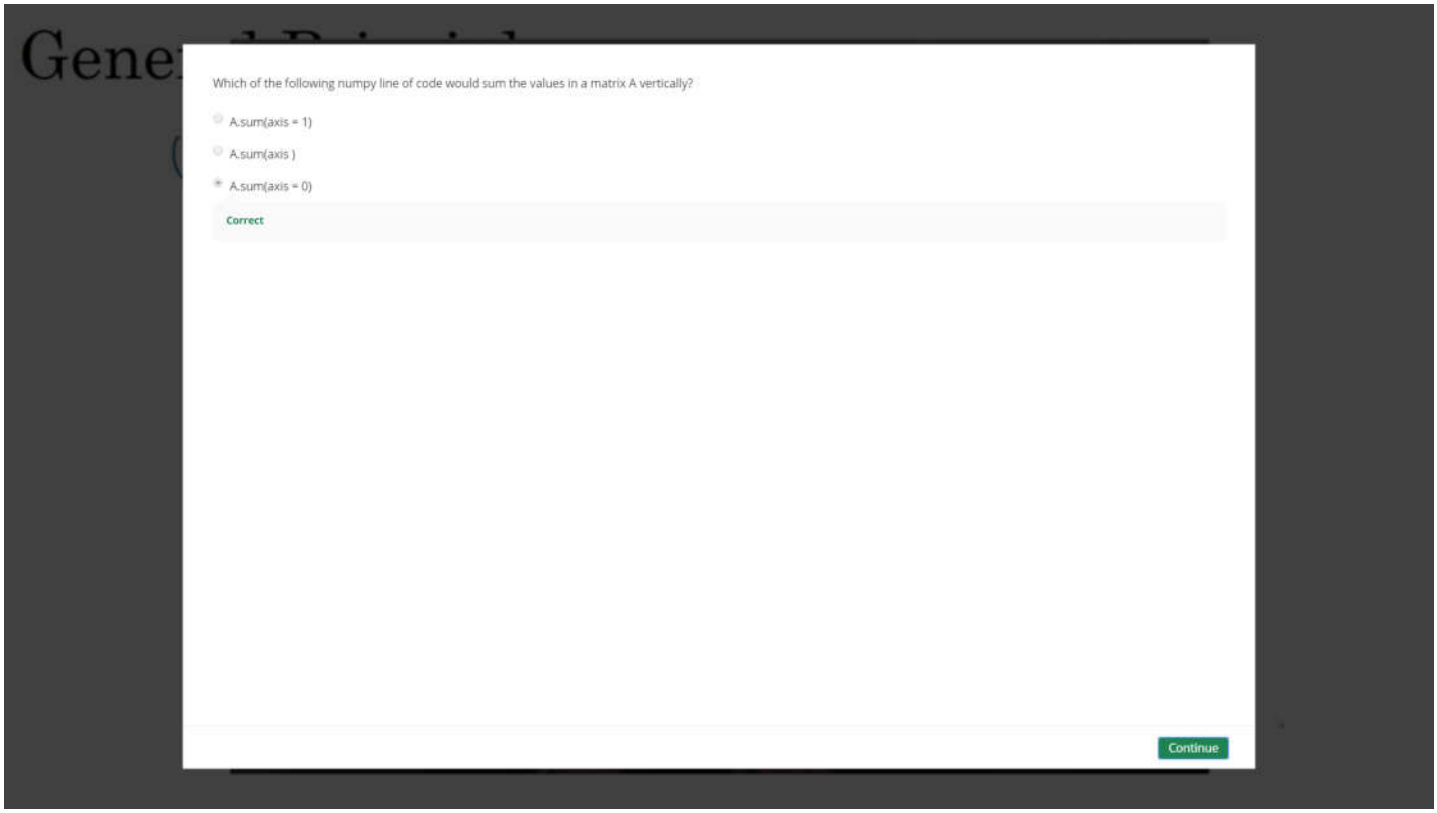
$\quad * \quad$

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

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

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

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



deeplearning.ai

Basics of Neural Network Programming

A note on python/ numpy vectors

jupyter Python-Numpy vectors Last Checkpoint: 7 minutes ago (unsaved changes)

Press Esc to exit full screen

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

```
In [1]: import numpy as np
        a = np.random.randn(5)

In [2]: print(a)
        [ 0.50290632 -0.29691149  0.95429604 -0.82126861 -1.46269164]

In [ ]: print
```

jupyter Python-Numpy vectors Last Checkpoint: 7 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

```
In [1]: import numpy as np
        a = np.random.randn(5)

In [2]: print(a)
        [ 0.50290632 -0.29691149  0.95429604 -0.82126861 -1.46269164]

In [3]: print(a.shape)
        (5,)

In [ ]:
```

jupyter Python-Numpy vectors Last Checkpoint: 8 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

```
In [1]: import numpy as np

        a = np.random.randn(5)

In [2]: print(a)

        [ 0.50290632 -0.29691149  0.95429604 -0.82126861 -1.46269164]

In [3]: print(a.shape)

        (5,)

In [4]: print(a.T)

        [ 0.50290632 -0.29691149  0.95429604 -0.82126861 -1.46269164]

In [5]: print(np.dot(a,a.T))

        4.06570109321

In [ ]:
```

2:32 / 6:49

jupyter Python-Numpy vectors Last Checkpoint: 8 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

```
        [ 0.50290632 -0.29691149  0.95429604 -0.82126861 -1.46269164]

In [3]: print(a.shape)

        (5,)

In [4]: print(a.T)

        [ 0.50290632 -0.29691149  0.95429604 -0.82126861 -1.46269164]

In [5]: print(np.dot(a,a.T))

        4.06570109321

In [6]: a = np.random.randn(5,1)
        print(a)

        [[-0.0967311 ]
         [-2.38617377]
         [-0.3243588 ]
         [-0.96216349]
         [ 0.54410384]]

In [ ]:
```

2:57 / 6:49

jupyter Python-Numpy vectors Last Checkpoint: 8 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

```
In [4]: print(a.T)

[ 0.50290632 -0.29691149  0.95429604 -0.82126861 -1.46269164]

In [5]: print(np.dot(a,a.T))

4.06570109321

In [6]: a = np.random.randn(5,1)
        print(a)

[[-0.0967311 ]
 [-2.38617377]
 [-0.3243588 ]
 [-0.96216349]
 [ 0.54410384]]

In [7]: print(a.T)

[[-0.0967311  -2.38617377 -0.3243588  -0.96216349  0.54410384]]

In [ ]:
```

jupyter Python-Numpy vectors Last Checkpoint: 9 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

```
In [5]: print(np.dot(a,a.T))

4.06570109321

In [6]: a = np.random.randn(5,1)
        print(a)

[[-0.0967311 ]
 [-2.38617377]
 [-0.3243588 ]
 [-0.96216349]
 [ 0.54410384]]

In [7]: print(a.T)

[[-0.0967311  -2.38617377 -0.3243588  -0.96216349  0.54410384]]

In [8]: print(np.dot(a,a.T))

[[ 0.00935691  0.23081721  0.03137558  0.09307113 -0.05263176]
 [ 0.23081721  5.69382526  0.77397645  2.29588928 -1.2983263 ]
 [ 0.03137558  0.77397645  0.10520863  0.31208619 -0.17648487]
 [ 0.09307113  2.29588928  0.31208619  0.92575858 -0.52351684]
 [-0.05263176 -1.2983263  -0.17648487 -0.52351684  0.29604898]]

In [ ]:
```

Python/numpy vectors

```
a = np.random.randn(5)
```

a.shape = (5,)

"rank 1 array"

Andrew Ng

Python/numpy vectors

```
a = np.random.randn(5)
```

a.shape = (5,)

"rank 1 array"

Don't use

Andrew Ng

Python/numpy vectors

```
a = np.random.randn(5)
```

a.shape = (5,)

"rank 1 array"

Don't use

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

Andrew Ng

Python/numpy vectors

```
a = np.random.randn(5)
```

a.shape = (5,)

"rank 1 array"

Don't use

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

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

Andrew Ng

Python/numpy vectors

```
a = np.random.randn(5)
```

a.shape = (5,)

"rank 1 array"

Don't use

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

→ a.shape = (5,1) column vector

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

→ a.shape = (1,5) row vector.

Andrew Ng

Python/numpy vectors

```
a = np.random.randn(5)
```

a.shape = (5,)

"rank 1 array"

Don't use

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

→ a.shape = (5,1) column vector ✓

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

→ a.shape = (1,5) row vector. ✓

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

Andrew Ng

Python/numpy vectors

```
a = np.random.randn(5)
```

a.shape = (5,)

"rank 1 array"

Don't use

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

→ a.shape = (5,1) column vector ✓

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

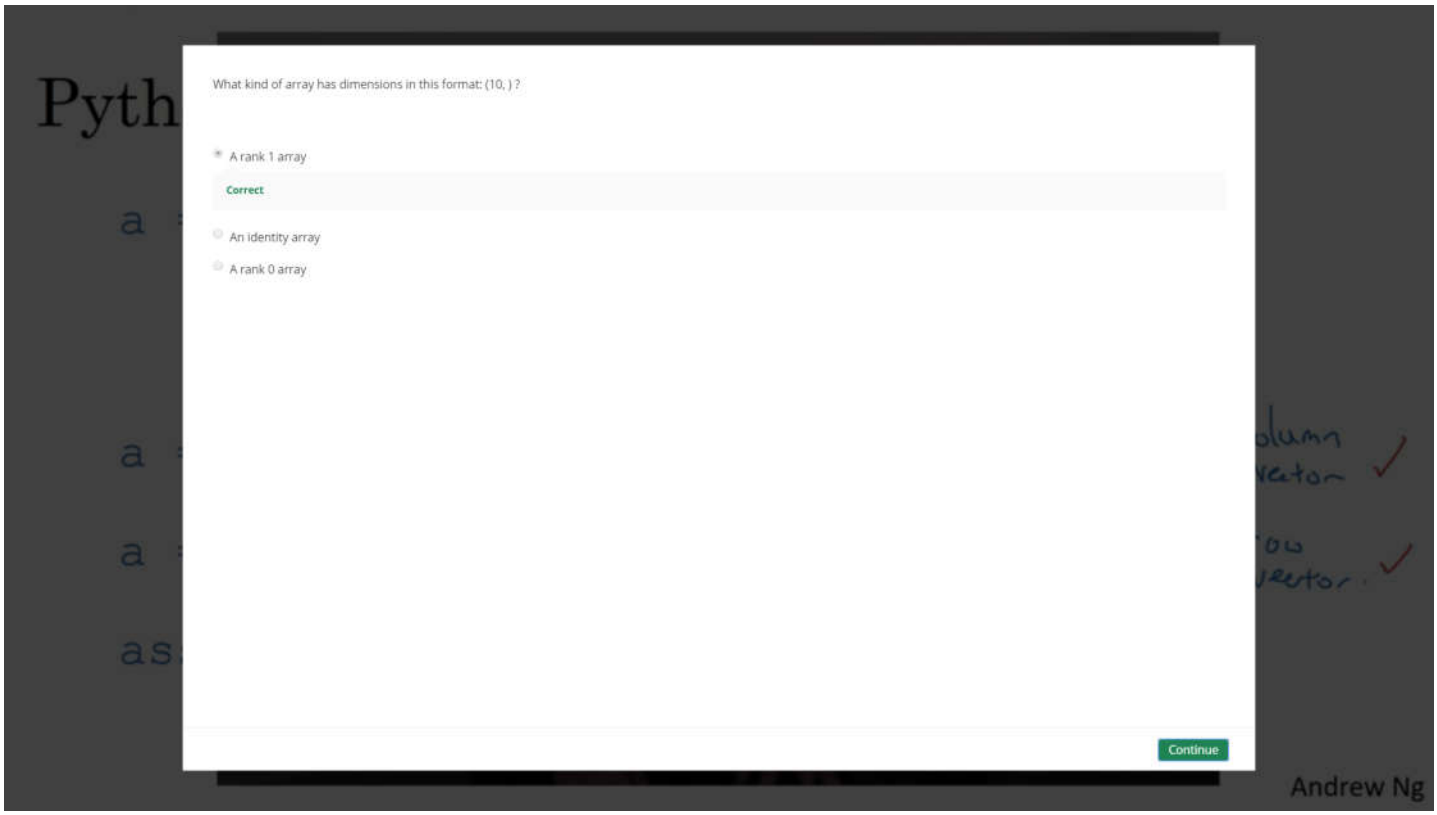
→ a.shape = (1,5) row vector. ✓

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

←

a = a.reshape((5,1))

Andrew Ng



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

Andrew Ng

Logistic regression cost function

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

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

Andrew Ng

Logistic regression cost function

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

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

$p(y|x)$

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

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

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

Andrew Ng

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

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

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

Andrew Ng

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

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

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

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

Andrew Ng

Cost on m examples

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

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

Maximum likelihood estimation

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

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

Andrew Ng

Cost on m examples

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

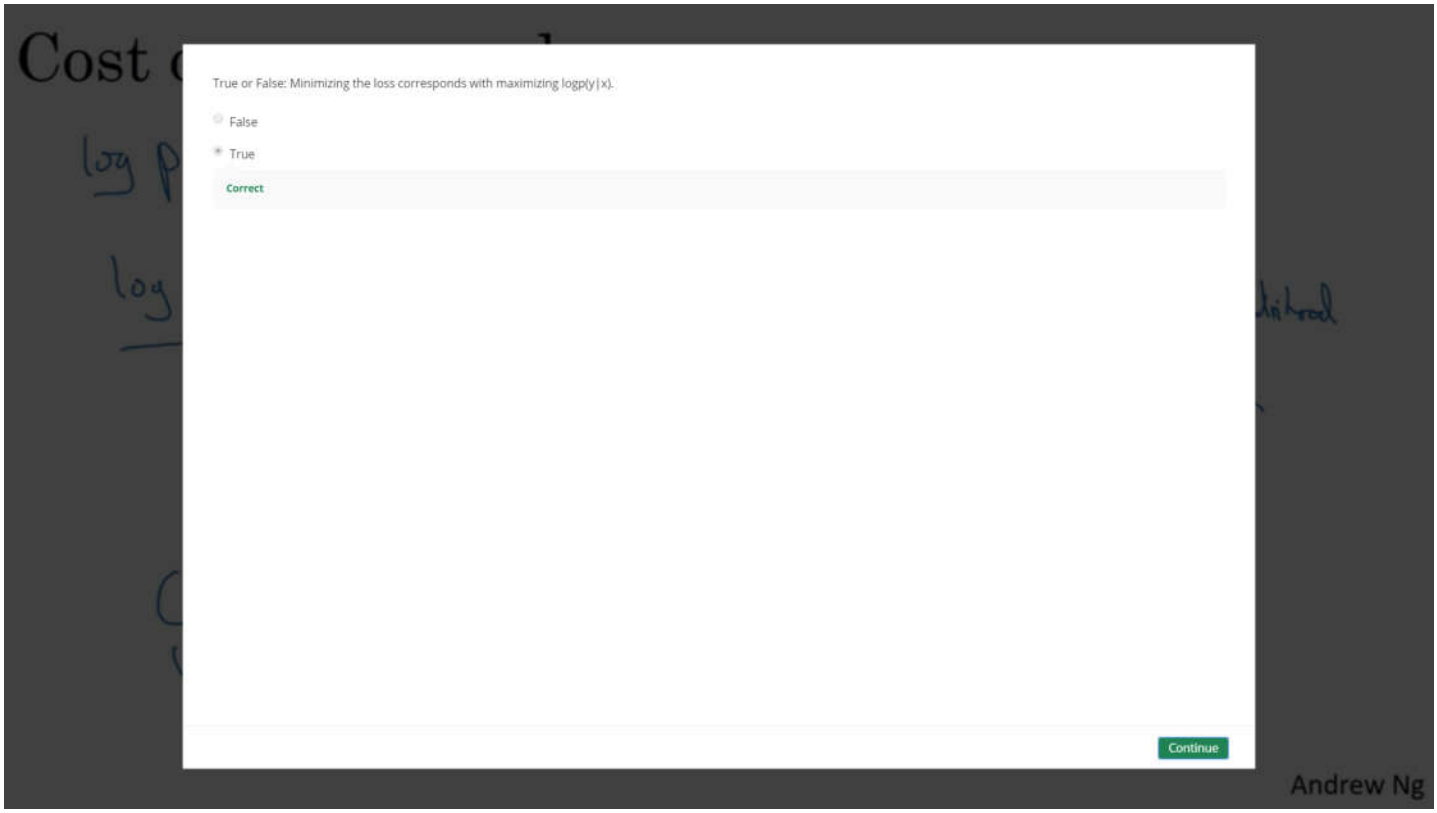
$$\log p(\ldots\ldots) = \sum_{i=1}^{m} \underbrace{\log p(y^{(i)} \mid x^{(i)})}_{-\mathcal{L}(\hat{y}^{(i)},\, y^{(i)})}$$

Maximum likelihood estimation

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

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

Andrew Ng


Cost
log p
log
True or False: Minimizing the loss corresponds with maximizing logp(y|x).
False
True
Correct
Continue
Andrew Ng