deeplearning.ai

Basics of Neural Network Programming

Binary Classification

Binary Classification

Input cat image

64

64

Aim to classify

1 (cat) vs 0 (non cat)

y

Input feature

Total Dimension of the vector

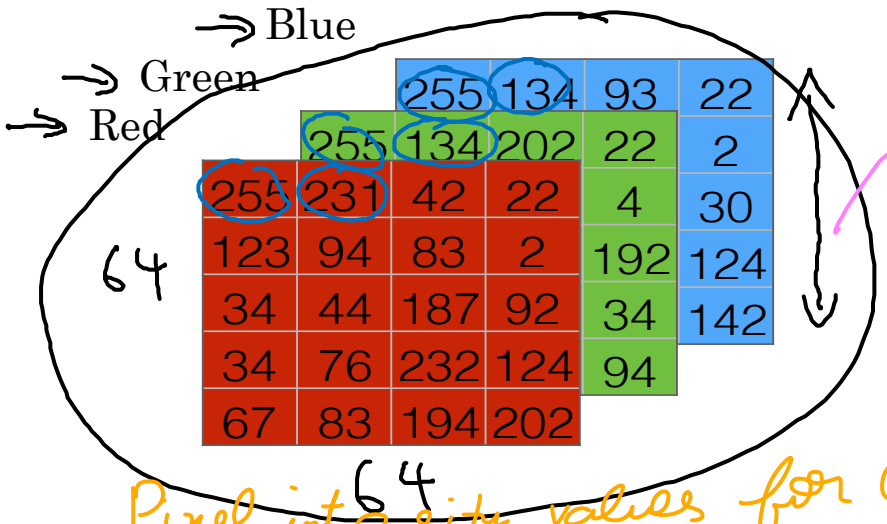
Pixel intensity vector

$$x = \begin{bmatrix} 255 \\ 231 \\ \vdots \\ 255 \\ 134 \\ \vdots \\ 255 \\ 134 \end{bmatrix} \begin{matrix} \} \text{ red} \\ \\ \\ \} \text{ green} \\ \\ \\ \} \text{ blue} \end{matrix}$$

$64 \times 64 \times 3 = 12288$

$n = n_x = 12288$

$x \longrightarrow y$



Pixel intensity values for cat image

Andrew Ng

Notation

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

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

(1^{st} training example; 2^{nd} training example; m^{th} training example)

$m = m_{train}$

m_{test} = # test examples.

Neural Networks representation

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

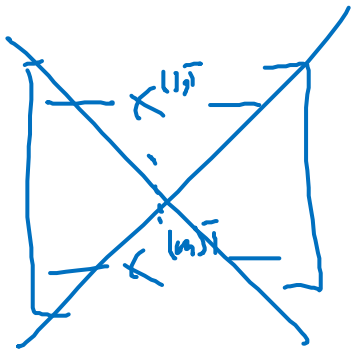
(1^{st} train, 2^{nd} train, m^{th} train; n_x; m)

$m \Rightarrow$ # of train examples

$n_x \Rightarrow$ # of values in a particular train example

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

$X.shape = (n_x, m)$



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

(1^{st} train, 2^{nd} train, m^{th} train)

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

$Y.shape = (1, m)$

deeplearning.ai

Basics of Neural Network Programming

Logistic Regression

Logistic Regression

If x is a picture, we want to tell what is the probability of a cat picture?

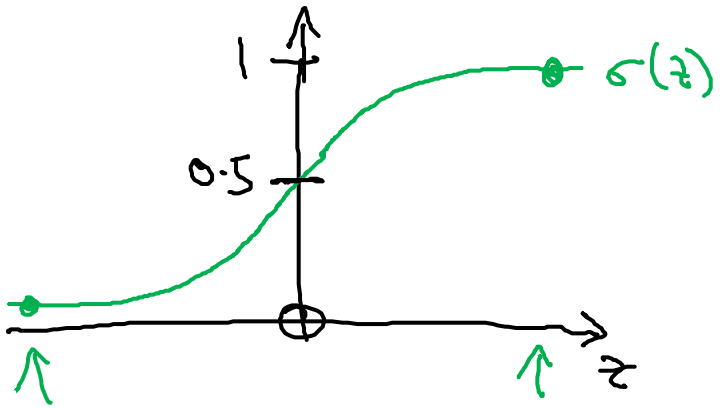
Given x, want $\hat{y} = P(y=1|x)$

$x \in \mathbb{R}^{n_x}$ (n_x dimensional vector)

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

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

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



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

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

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix}$ θ_0 ← b ← Interapt; $\theta_1 \ldots \theta_{n_x}$ ← w ← slope

Not going to use this Notation

Sigmoid function

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

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

If z large negative number

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

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Logistic Regression cost function

Logistic Regression cost function

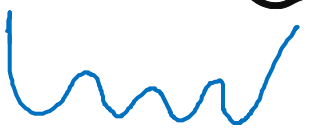
$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, where $\sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$

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

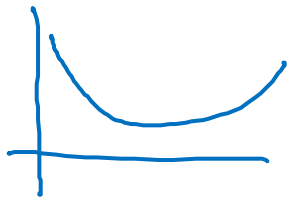
$x^{(i)}$, $y^{(i)}$, $z^{(i)}$ — Associated with i-th example.

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

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



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



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

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

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

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

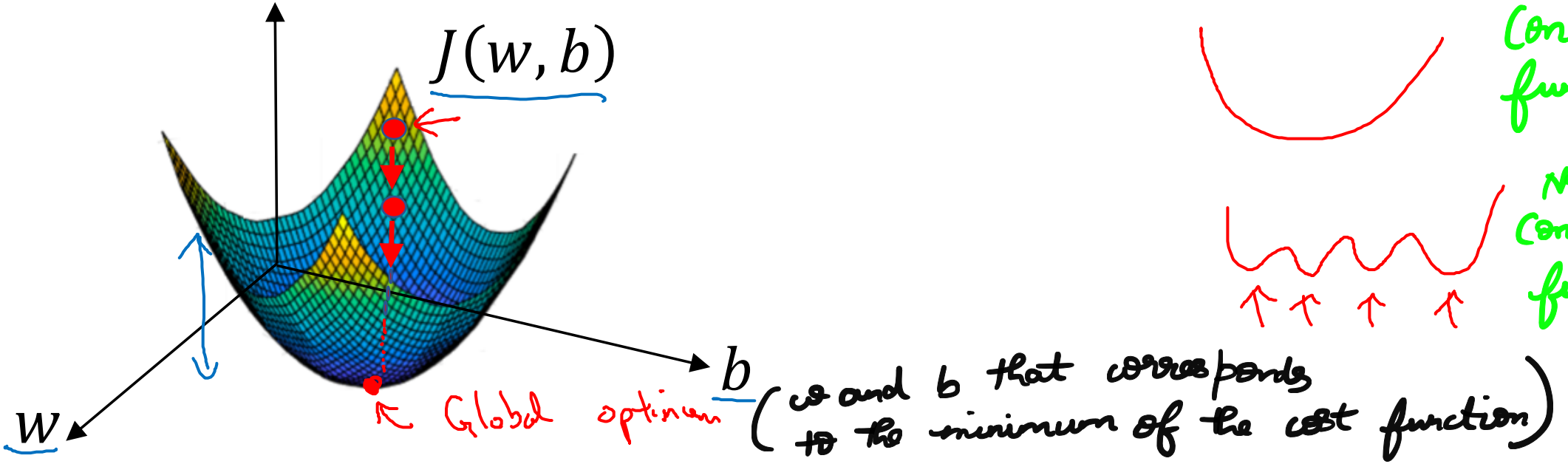
Gradient Descent

Gradient Descent

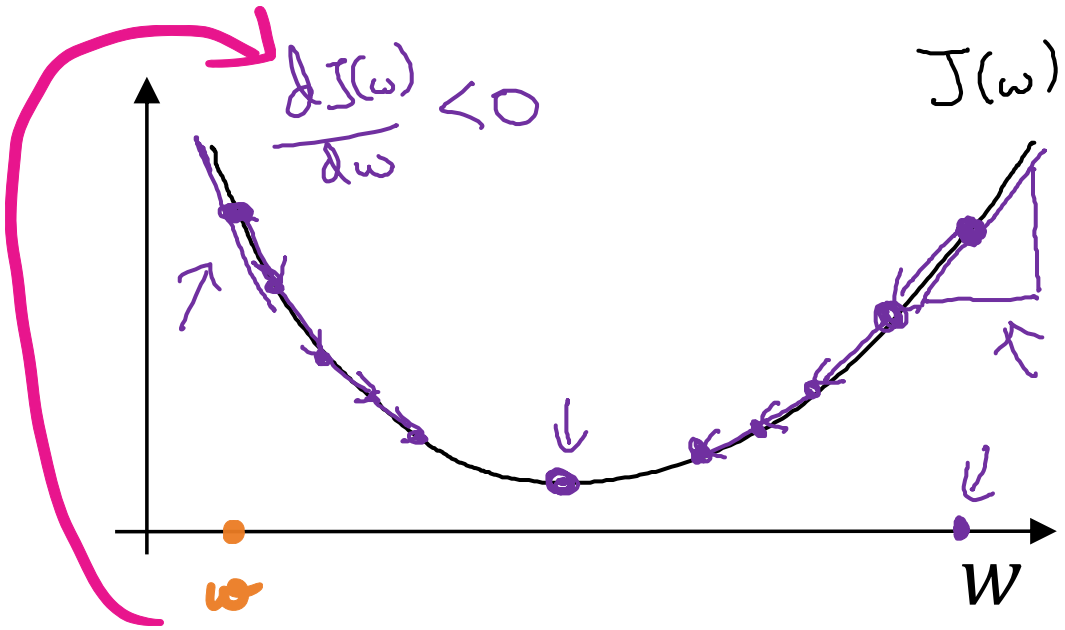
Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$

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

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



Gradient Descent



$\frac{dJ(w)}{dw} > 0$

Repeat {

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

learning rate

"dw"

$w := w - \alpha dw$

}

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

$J(w,b)$

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

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

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

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

∂ "partial derivative"

d ← J

Andrew Ng

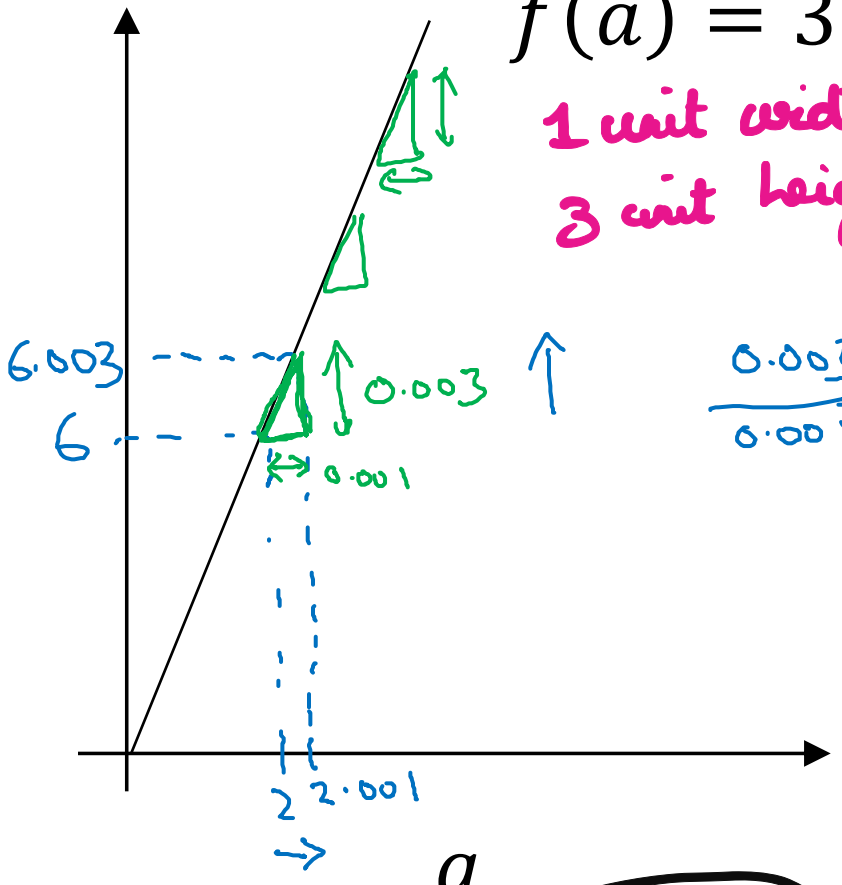
deeplearning.ai

Basics of Neural Network Programming

Derivatives

Intuition about derivatives

$f(a) = 3a$



1 unit width ↑
3 unit height will increase

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

→ a = 2 f(a) = 6

a = 2.001 f(a) = 6.003

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

→ a = 5 f(a) = 15

a = 5.001 f(a) = 15.003

slope at a = 5 is also 3

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

0.001
0.000000001
0.000000000001

So Height : width ⇒ 3 : 1

Andrew Ng

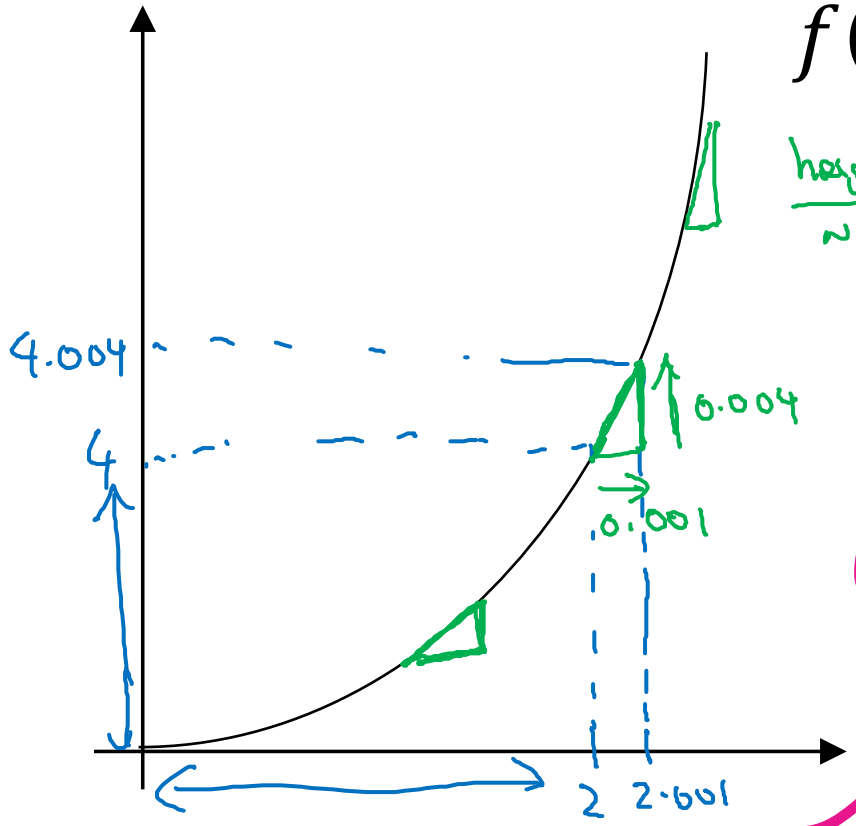
deeplearning.ai

Basics of Neural Network Programming

More derivatives examples

Intuition about derivatives

$f(a) = a^2$



height / width

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

0.001

$(2a) \times 0.001$

If width increase by 1 unit, then the height increases by 2 units.

0.001 ←

0.000000....01 ←

$a = 2$

$a = 2.001$

$f(a) = 4$

$f(a) \approx 4.004$

(4.004001)

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

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

$a = 5$

$a = 5.001$

$f(a) = 25$

$f(a) \approx 25.010$

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

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

Andrew Ng

More derivative examples

$f(a) = a^2$

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

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

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

$f(a) = a^3$

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

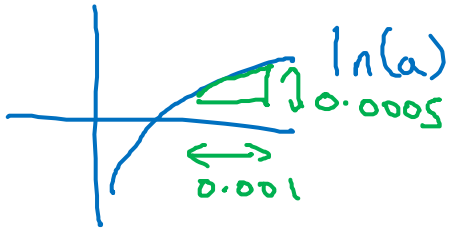
$a = 2$ $\quad f(a) = 8$

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

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

$\ln(a)$

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



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

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

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

0.0005

0.0005

$a \uparrow$ by 0.001

$f(a) \uparrow$ by 0.0005

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

deeplearning.ai

Basics of Neural Network Programming

Computation Graph

Computation Graph

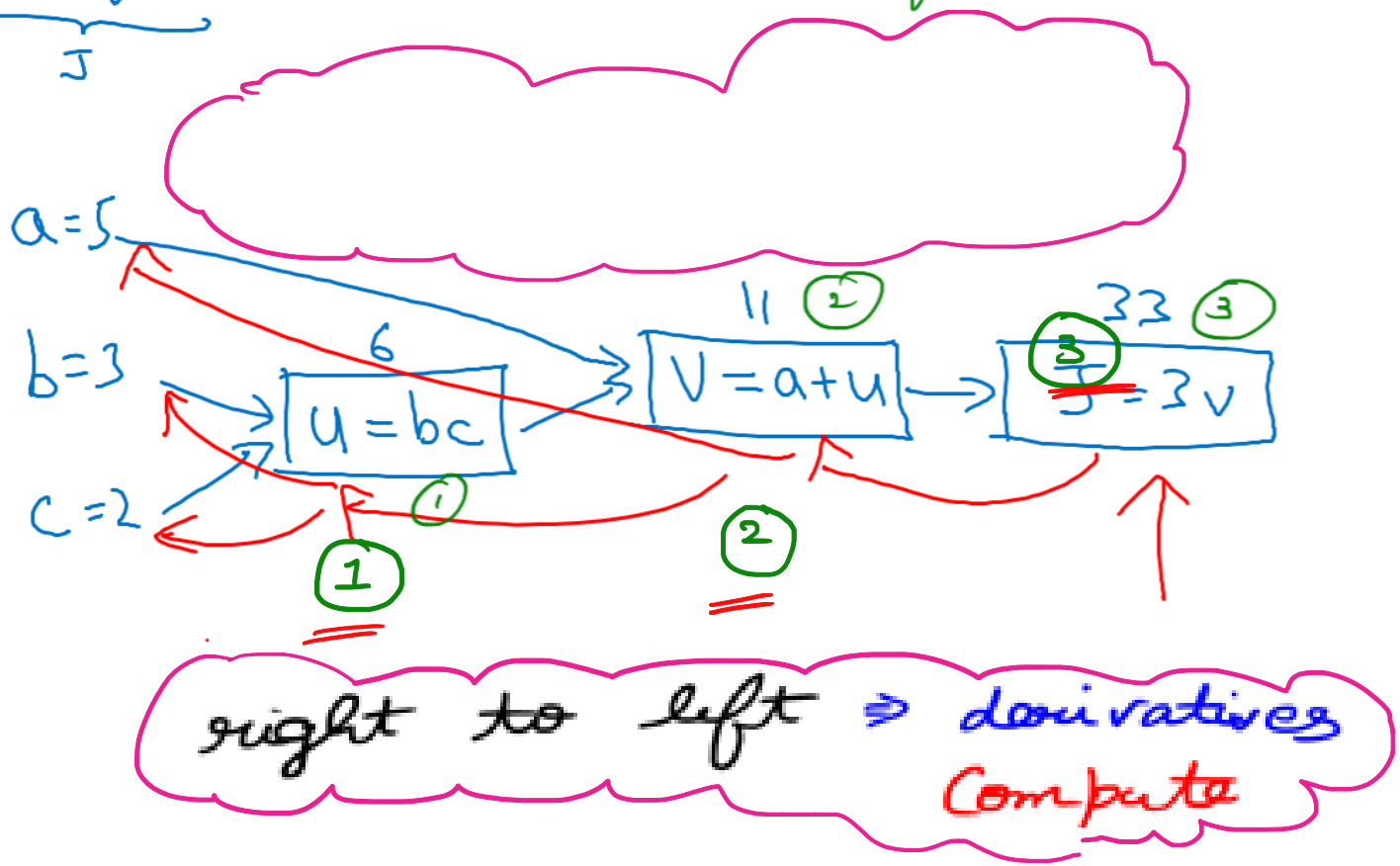
$J(a,b,c) = 3(a+bc) = 3(5+3\times 2) = 33$

u, v, J

Compute this function $J(a,b,c)$

1) $u = bc$
2) $V = a+u$
3) $J = 3v$

We have to find all the three in-order to compute ① ② and ③



Computation Graph

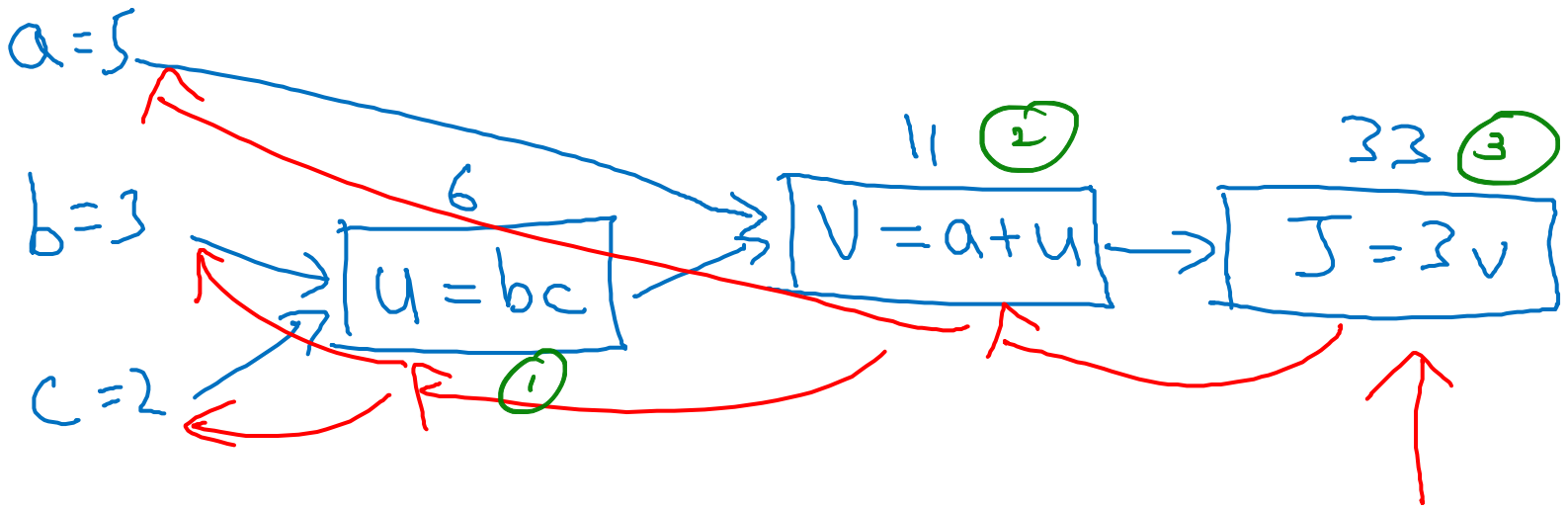
$J(a,b,c) = 3(a+bc) = 3(5+3\times 2) = 33$

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

Compute this function $J(a,b,c)$

1) $u = bc$
2) $V = a+u$
3) $J = 3v$

We have to find all the three in-order to compute ① ② and ③

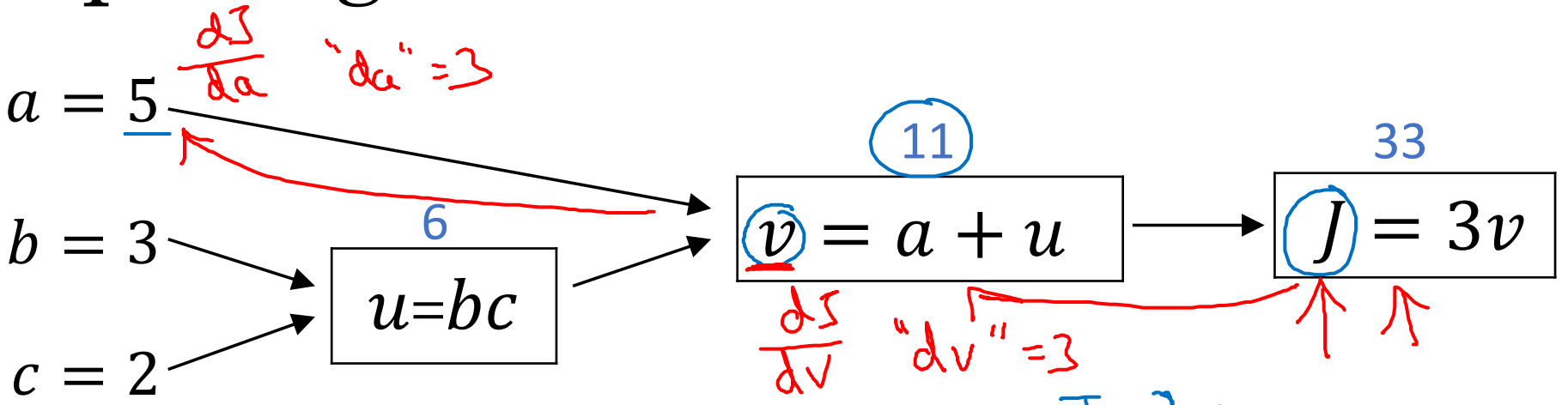


deeplearning.ai

Basics of Neural Network Programming

Derivatives with a Computation Graph

Computing derivatives



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

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

3×1

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

$a \to v \to J$

$J = 3v$

$v = 11 \to 11.001$

$J = 33 \to 33.003$

$a = 5 \to 5.001$

$\to v = 11 \to 11.001$

$J = 33 \to 33.003$

$f(a) = 3a$

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

$J = 3v$

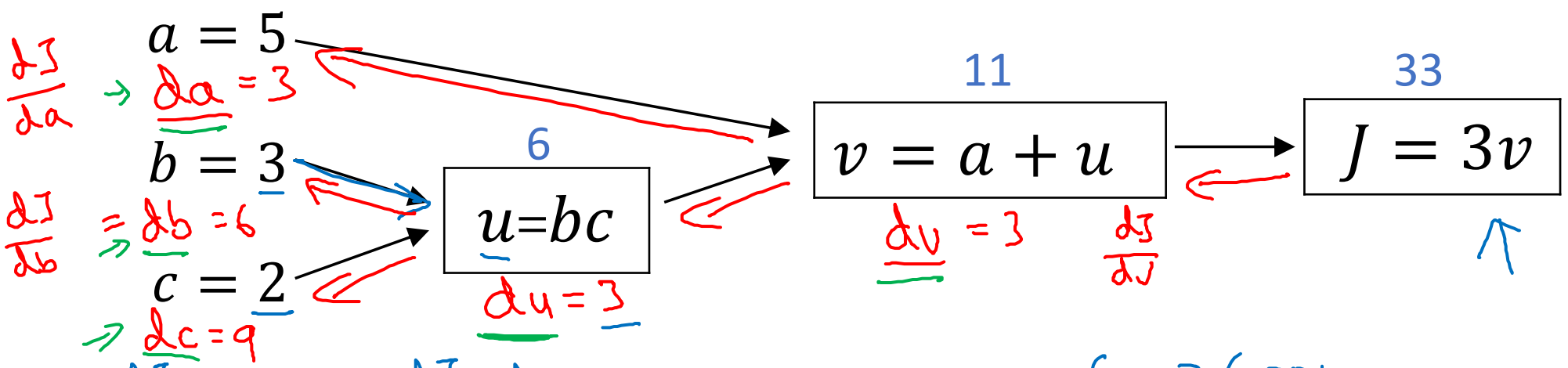
$\frac{dJ}{dv} = 3$

dFinalOutputVar / dvar

dJdvar

"dvar"

Computing derivatives



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

$3 \quad 1$

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

$\rightarrow 3 \quad = 2$

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

3×3

$u = 6 \rightarrow 6.001$

$v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

$b = 3 \rightarrow 3.001$

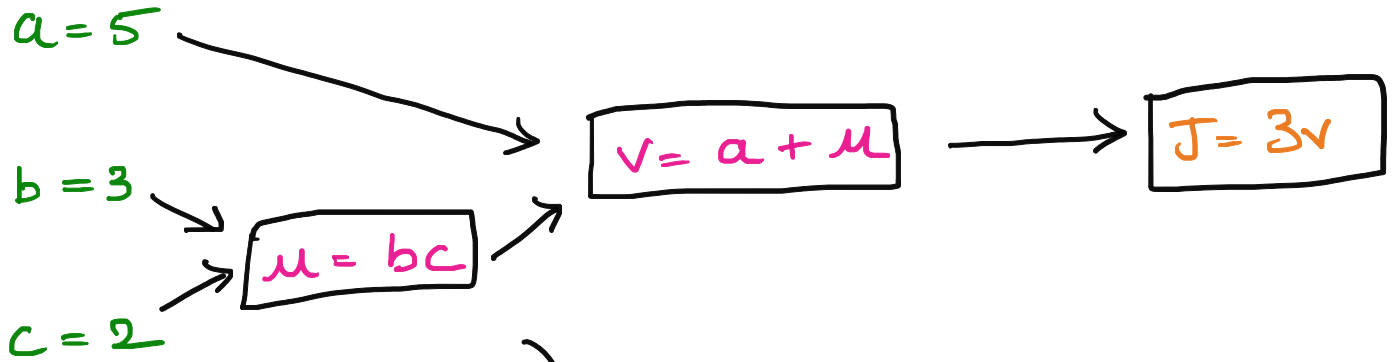
$u = b \cdot c = 6 \rightarrow 6.002$ $\quad c = 2$

$J = 33.006$ $\quad .006$

$v = 11.002$

$J = 3v$

Andrew Ng



Step-1 (a's effect)

J = 3v

v = 11 → 11.001

J = 33 → 33.003

3 times increase

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

Therfore slight change in v will increase the J, 3 times.

v = a + u

a = 5 → 5.001

v ⇒ 6+5 ⇒ 11 → 6+5.001 ⇒ 11.001

J ⇒ 3*11 ⇒ 33 → 3*11.001 ⇒ 33.003

↑ v = a + u ↑

↑ J = 3v ↑

$\frac{dJ}{da} \Rightarrow \frac{dv}{da} * \frac{dJ}{dv}$ (1, 3)

$\frac{dJ}{da} \Rightarrow 1 * 3 \Rightarrow 3$

Step-2 (b's effect)

deeplearning.ai

Basics of Neural Network Programming

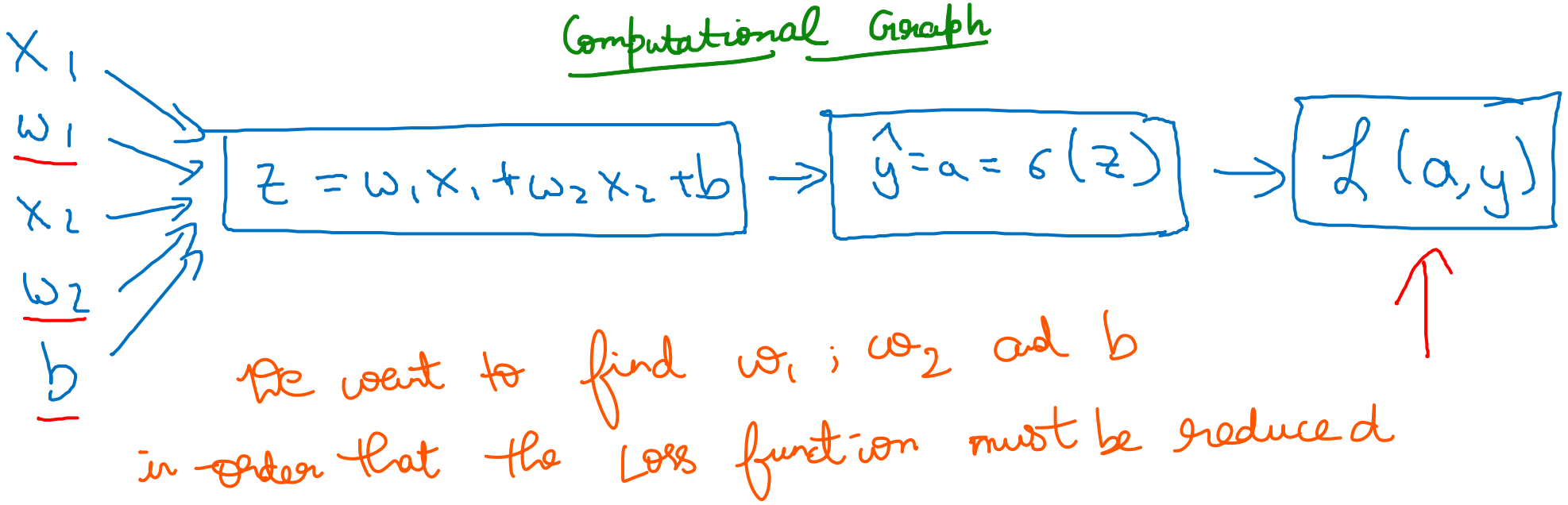
Logistic Regression Gradient descent

Logistic regression recap

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

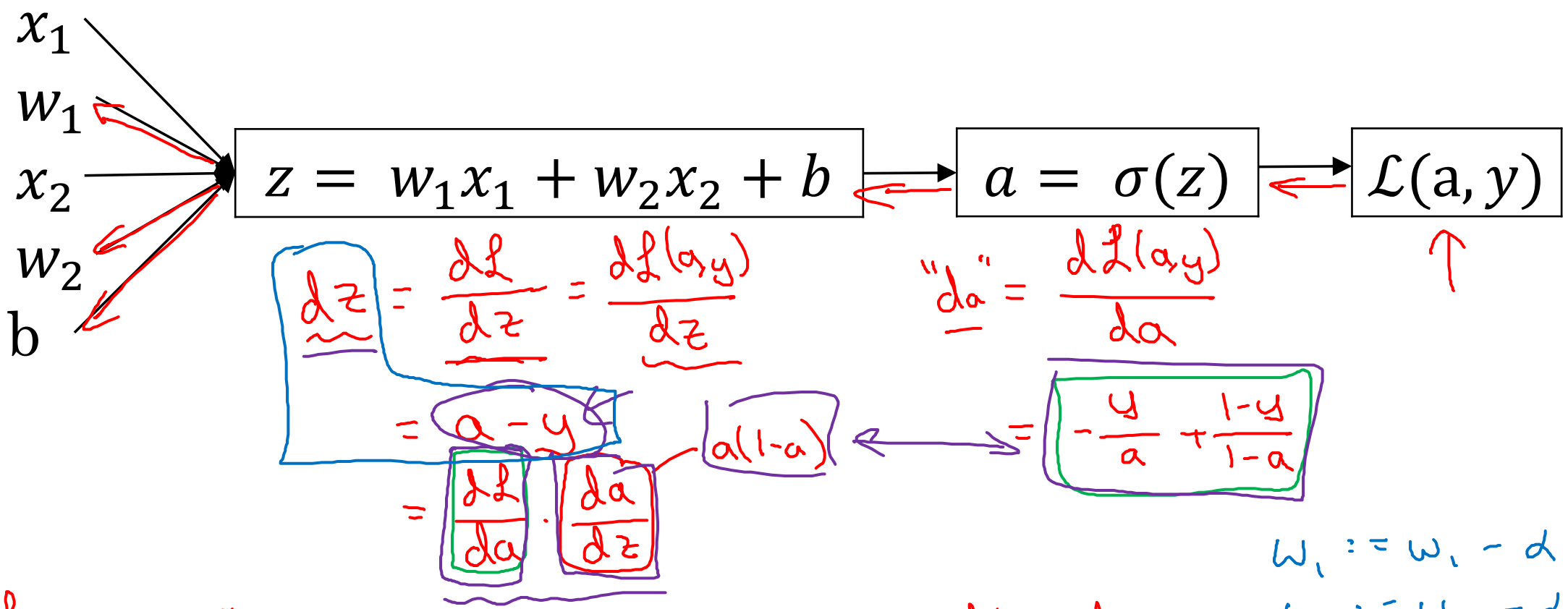
$$\hat{y} = a = \sigma(z)$$

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



Andrew Ng

Logistic regression derivatives



Andrew Ng

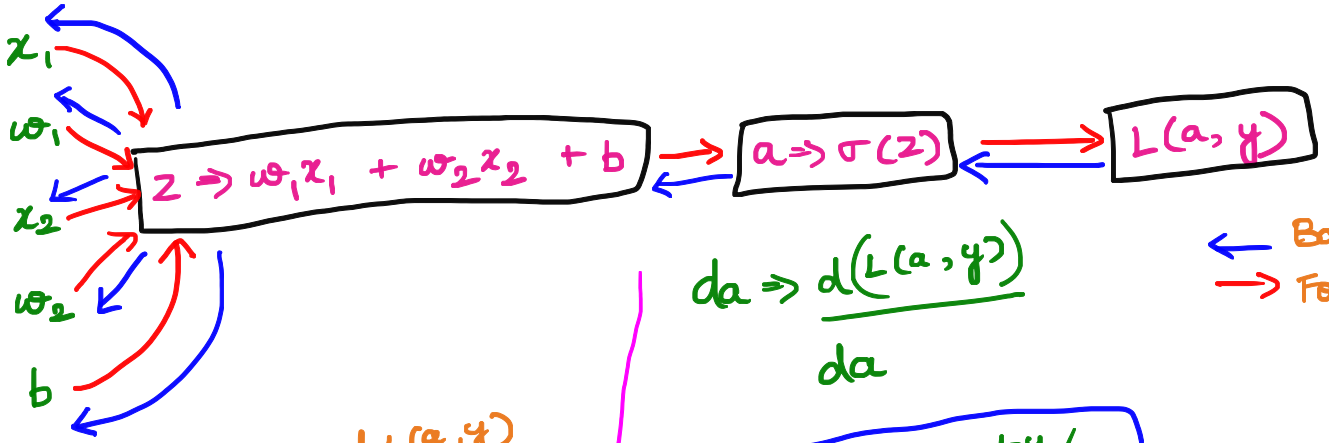
1) $dz \Rightarrow \frac{dL}{dz} \Rightarrow a - y$

2) $dw_1 \Rightarrow \frac{dL}{dw_1} \Rightarrow x_1 * dz$

$dw_2 \Rightarrow \frac{dL}{dw_2} \Rightarrow x_2 * dz$

*) $\frac{dL(a,y)}{da} \Rightarrow -y/a + (1-y)/(1-a)$

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



← Backward propagation
→ Forward propagation

$$da \Rightarrow \frac{d(L(a,y))}{da}$$

$$da \Rightarrow -y/a + (1-y)/(1-a)$$

$$dz \Rightarrow \frac{dL}{dz} \Rightarrow \frac{da}{dz} * \frac{dL(a,y)}{da}$$

$$\Rightarrow a(1-a) * \left[-y/a + (1-y)/(1-a)\right]$$

$$dz \Rightarrow a - y$$

$$\frac{dL}{dw_1} \Rightarrow dw_1 \Rightarrow x_1 \cdot dz$$

$$\frac{dL}{dw_2} \Rightarrow dw_2 \Rightarrow x_2 \cdot dz$$

$$db \Rightarrow dz$$

With these formulae, the slope and intercept are going to be updated in the gradient descent

$$w_1 \Rightarrow w_1 - \alpha\, dw_1$$
$$w_2 \Rightarrow w_2 - \alpha\, dw_2$$
$$b \Rightarrow b - \alpha\, db$$

1) Derivative with respect to the Activation function

$$\frac{dL(a,y)}{da} \Rightarrow ?$$

$$L(a,y) \Rightarrow -\frac{1}{m}\left[\sum_{i=1}^{m} y \ln(a) + (1-y)\ln(1-a)\right]$$

$$\frac{d(L(a,y))}{da} \Rightarrow -y\ln(a) - (1-y)\ln(1-a)$$

$$\Rightarrow -y\left/\frac{1}{a}\right. - \frac{1-y}{(1-a)} * (-1)$$

$$\boxed{\frac{d(L(a,y))}{da} \Rightarrow -y/a + \frac{1-y}{1-a}}$$

2) Derivative with respect to sigmoid function

$\frac{da}{dz} \Rightarrow ?$

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

$a = \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2 + b)}}$

$\frac{d}{dx}(x^2) \Rightarrow 2x^{2-1}$

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

$\frac{da}{dz} \Rightarrow -1\,(1+e^{-z})^{-1-1}\,(0+e^{-z})\,(-1)$

$\Rightarrow -1\,(1+e^{-z})^{-2}\,(-e^{-z})$

$\Rightarrow \frac{e^{-z}}{(1+e^{-z})^2}$

Add +1 and -1

$\Rightarrow \frac{1}{(1+e^{-z})} * \frac{1+e^{-z}-1}{(1+e^{-z})}$

$$\frac{da}{dz} \Rightarrow \frac{1}{1+e^{-z}} * \frac{1+e^{-z}-1}{1+e^{-z}}$$

$$\Rightarrow sig(z) * \left[1 - \frac{1}{1+e^{-z}}\right] \rightarrow \frac{1+e^{-z}+1}{1+e^{-z}}$$

$$\frac{da}{dz} \Rightarrow sig(z) * 1 - sig(z)$$

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

3) Combining 1) and 2) together

$$dz \Rightarrow \frac{dL}{dz} \Rightarrow \frac{da}{dz} * \frac{dL(a,y)}{da}$$

$$\Rightarrow [a * (1-a)] * \left[-y/a + (1-y)/(1-a)\right]$$

$$\Rightarrow a * \cancel{(1-a)} * \frac{(1-a)(-y) + a(1-y)}{a * \cancel{(1-a)}}$$

$$dz \Rightarrow \frac{dL}{dz} \Rightarrow -y + \cancel{ay} + a - \cancel{ay} \Rightarrow a - y$$

$$\boxed{dz = a - y}$$

Derivatives with respect to weights:

$$z = w_1 x_1 + w_2 x_2 + b$$

$$z = w^T x$$

$$\frac{dL(a,y)}{dw} \Rightarrow ?$$

$$\frac{dz}{dw} \Rightarrow x$$

$$\frac{dL(a,y)}{dw} \Rightarrow \frac{dL}{dw} \Rightarrow \frac{dz}{dw} * \frac{da}{dz} * \frac{dL}{da} \Rightarrow \frac{dz}{dw} * \frac{dL}{dz}$$

TP

$$\Rightarrow x * \frac{dL}{dz}$$

$$\left[dz \rightarrow \frac{dL}{dz} \Rightarrow a - y \right]$$

$$\frac{dL}{dw} \Rightarrow x * dz$$

$$\frac{dL}{dw_1} \Rightarrow dw_1 \Rightarrow x_1 * dz$$

(or)

$$\frac{dL}{dw_2} \Rightarrow dw_2 \Rightarrow x_2 * dz$$

(or)

$$\frac{dL}{dw_1} \Rightarrow dw_1 \Rightarrow x_1 * (a - y)$$

$$\frac{dL}{dw_2} \Rightarrow dw_2 \Rightarrow x_2 * (a - y)$$

Derivatives with respect to weighs:

$$\frac{dL(a,y)}{dw} \Rightarrow ?$$

$$\frac{dL}{dz} \Rightarrow a-y$$

$$\frac{dL(a,y)}{dw} \Rightarrow \frac{dz}{dw} * \frac{da}{dz} * \frac{dL}{da} \Rightarrow \frac{dz}{dw} * \frac{dL}{dz}$$

$$\frac{dL(a,y)}{dw} \Rightarrow \frac{dz}{dw} * (a-y) \longrightarrow (1)$$

Let's take the cost function

$$(2) \longrightarrow L(a,y) \Rightarrow -1/m \left[\sum_{i=1}^{m} \left(y \ln(a) + (1-y) \ln(1-a) \right) \right]$$

$$\ln\left(\frac{a}{b}\right) \Rightarrow \ln(a) - \ln(b)$$

$$\ln(a) \Rightarrow \ln\left(\frac{1}{1+e^{-z}}\right)$$

$$\Rightarrow \ln\left[(1+e^{-z})^{-1}\right]$$

$$\ln(a) \Rightarrow -\ln\left[(1+e^{-z})\right]$$

$$\ln(1-a) \Rightarrow \ln\left[1 - \frac{1}{1+e^{-z}}\right]$$

$$\Rightarrow \ln\left[\frac{1+e^{-z}-1}{1+e^{-z}}\right] \Rightarrow \ln\left[\frac{e^{-z}}{1+e^{-z}}\right] \Rightarrow \ln(e^{-z}) - \ln(1+e^{-z})$$

ln and e are inverse functions

$$\ln(1-a) \Rightarrow -z - \ln(1+e^{-z})$$

Put ln(a) and ln(1-a) in (2)

$$L(a,y) \Rightarrow -\frac{1}{m}\left[\sum_{i=1}^{m}\left(y \ln(a) + (1-y)\ln(a)\right)\right]$$

Note : We will use the outer -ve at the last

$$\Rightarrow \sum_{i=1}^{m}\left[y\left[-\ln(1+e^{-z})\right] + (1-y)\left[-z-\ln(1+e^{-z})\right]\right]$$

$$\Rightarrow \sum_{i=1}^{m}\left[-y\ln(1+e^{-z}) - z - \ln(1+e^{-z}) + yz + y\ln(1+e^{-z})\right]$$

$$\Rightarrow -z - \ln(1+e^{-z}) + yz$$

$$\Rightarrow yz - \left[z + \ln(1+e^{-z})\right]$$

$$\ln(ab) \Rightarrow \ln(a) + \ln(b)$$

Applying log and e on z

$$\Rightarrow yz - \left[\ln(e^{z}) + \ln(1+e^{-z})\right]$$

$$\Rightarrow yz - \left[\ln\left[(e^{z})\cdot(1+e^{-z})\right]\right]$$

$$\Rightarrow yz - \left[\ln\left[e^{z}(1) + e^{z}e^{-z}\right]\right]$$

$$\Rightarrow yz - \left[\ln\left[1+e^{z}\right]\right]$$

$L \Rightarrow yz - \ln(1+e^{z})$

$z \Rightarrow wx+b$

$L \Rightarrow y(wx+b) - \ln(1+e^{wx+b})$

Taking the derivative with respect to w

$$\frac{dL}{dw} \Rightarrow \frac{dL}{dw}\left[y(wx+b) - \ln(1+e^{wx+b})\right]$$

$$\frac{dL}{dw} \Rightarrow yx - \left(\frac{e^{wx+b}}{1+e^{wx+b}} * x\right) \Rightarrow yx - \left(\frac{x * e^{wx+b}}{1+e^{wx+b}}\right)$$

$$\frac{dL}{dw} \Rightarrow yx - x\,sig(z)$$

$$\Rightarrow x\left[y - sig(z)\right]$$

(-ve) on both sides

$$\frac{dL}{dw} \Rightarrow -x\left[y - sig(z)\right]$$

$$\frac{dL}{dw} \Rightarrow x\left[sig(z) - y\right]$$

$$\boxed{\frac{dL}{dw} \Rightarrow x(a-y)}$$

$$\frac{e^{wx+b}}{1+e^{wx+b}} \Rightarrow \frac{1}{\frac{1+e^{wx+b}}{e^{wx+b}}} \Rightarrow \frac{1}{\frac{1}{e^{wx+b}} + \frac{e^{wx+b}}{e^{wx+b}}}$$

$$\Rightarrow \frac{1}{\frac{1}{e^{wx+b}} + 1}$$

$$\Rightarrow \frac{1}{e^{-(wx+b)} + 1} \Rightarrow \frac{1}{1+e^{-(wx+b)}}$$

$$\boxed{\frac{e^{wx+b}}{1+e^{wx+b}} \Rightarrow \frac{1}{1+e^{-(wx+b)}} \Rightarrow sig(z)}$$

Cross Entropy Loss function

$$L(a,y) \Rightarrow -1/m\left[\sum_{i=1}^{m} y \ln(a) + (1-y)\ln(1-a)\right]$$

sigmoid function

Using the -ve which was left at the beginning.

Concluding the above findings

$$\frac{dL}{dw} \Rightarrow dw \Rightarrow x * (a-y) \Rightarrow x * dz$$

Since we have x_1 and x_2

$$\frac{dL}{dw_1} \Rightarrow dw_1 \Rightarrow x_1 * (a-y)$$

$$\frac{dL}{dw_2} \Rightarrow dw_2 \Rightarrow x_2 * (a-y)$$

Derivative with respect to bias :

$$\frac{dL(a,y)}{db} \Rightarrow ?$$

$$\frac{dL}{db} \Rightarrow \frac{dz}{db} * \frac{da}{dz} * \frac{dL}{da} \Rightarrow \frac{dz}{db} * \frac{dL}{dz}$$

$$dz \Rightarrow \frac{dL}{dz} \Rightarrow a - y$$

Let's calculate $\frac{dz}{db} \Rightarrow ?$

deeplearning.ai

Basics of Neural Network Programming

Gradient descent on m examples

Logistic regression on m examples

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

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

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

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

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

$dw_1^{(i)}$ — $(x^{(i)}, y^{(i)})$

Andrew Ng

Logistic regression on m examples

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

For $i=1$ to m

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

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

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

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

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

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

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

dw_3 ... dw_n

$n=2$

$J /= m$

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

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

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

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

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

Vectorization

$J=0$; $dw_1 \Rightarrow \frac{dL}{dw_1} \Rightarrow 0$; $dw_2 \Rightarrow \frac{dL}{dw_2} \Rightarrow 0$; $db \Rightarrow \frac{dL}{db} \Rightarrow 0$; $dz \Rightarrow \frac{dL}{dz}$

for $i = 1$ to m { → number of train examples

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

$$a^{(i)} = \sigma(z^{(i)}) \quad \left(\because \sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}} \right)$$

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

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

for loop { → number of features

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

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

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

}

$J = J/m$; $dw_1 \Rightarrow \frac{dw_1}{m}$; $dw_2 \Rightarrow \frac{dw_2}{m}$; $db \Rightarrow \frac{db}{m}$ } Taking the averages

$w_1 \Rightarrow w_1 - \alpha\, dw_1$

$w_2 \Rightarrow w_2 - \alpha\, dw_2$

$b \Rightarrow b - \alpha\, db$

} This step will be done after finding the derivatives of both the slopes and intercept

}

Vectorization

↓

To avoid this

↓

As the # of features, the complexity too increases

} Here only there are 2 features. Think of n features

deeplearning.ai

Basics of Neural Network Programming

Vectorization

What is vectorization?

$z = w^T x + b$

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

Non-vectorized:

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

Vectorized

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

$w^T x$

→ GPU
→ CPU

} SIMD - single instruction multiple data.

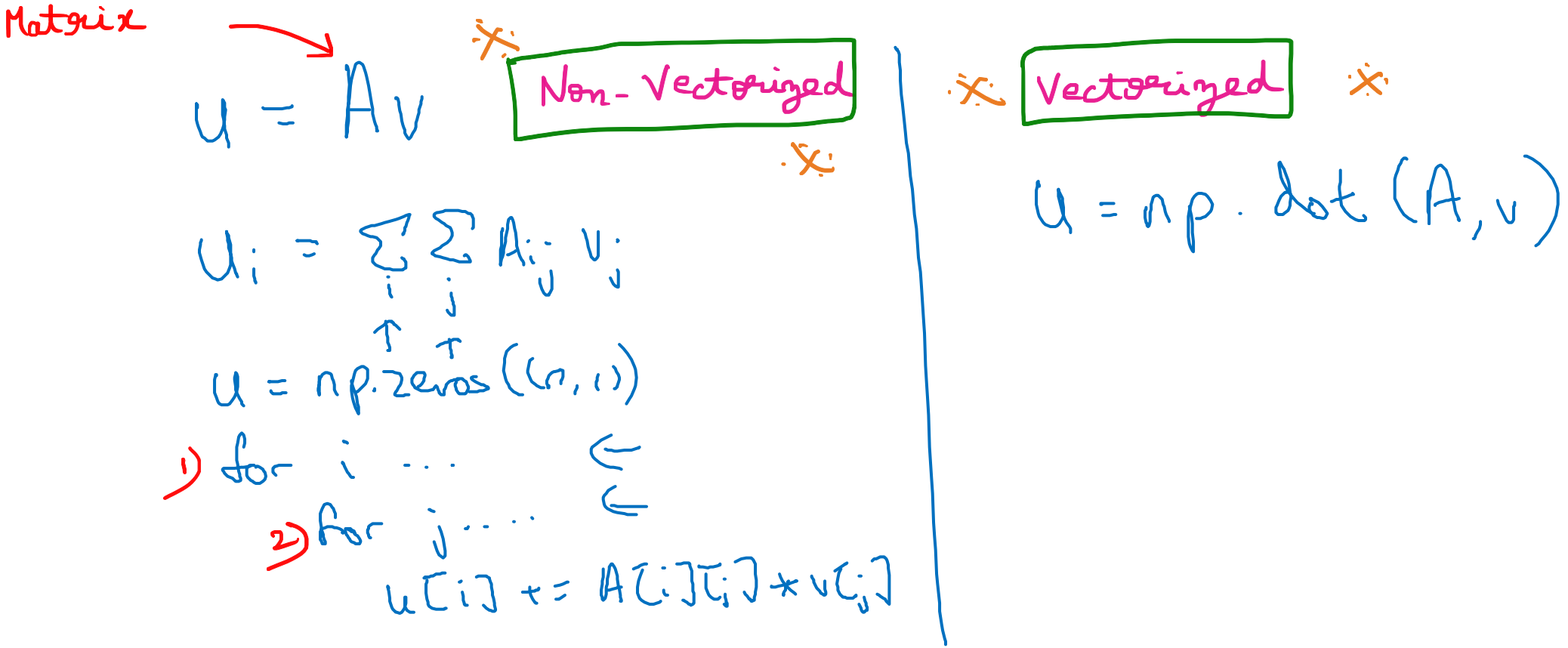
deeplearning.ai

Basics of Neural Network Programming

More vectorization examples

Neural network programming guideline

Whenever possible, avoid explicit for-loops.



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])
```

Non-vectorized version

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

Vectorized version

```
np.log(v)
np.abs(v)
np.maximum(v,0)
v**2
1/v
```

some of the examples

Logistic regression derivatives

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

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

for i = 1 to n:

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

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

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

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

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

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

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

$n_x = 2$

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

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

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

$dw /= m$.

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

Vectorizing Logistic Regression

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

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

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

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

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

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

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

$$A = \begin{bmatrix} a^{(1)} & a^{(2)} & \cdots & a^{(m)} \end{bmatrix} = \sigma(Z)$$

Andrew Ng

Cost function → $z^{(1)} = w^T x^{(1)} + b$ | $z^{(2)} = w^T x^{(2)} + b$ | $z^{(3)} = w^T x^{(3)} + b$ | ..... m training examples

Activation function → $a^{(1)} = \sigma(z^{(1)})$ | $a^{(2)} = \sigma(z^{(2)})$ | $a^{(3)} = \sigma(z^{(3)})$

1st training example | 2nd training example | 3rd training example

$$Z = [z^{(1)}, z^{(2)}, \dots z^{(m)}] \Rightarrow w^T \underline{X} + [b, b, \dots b]$$

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

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

$$\Rightarrow w^T X + [b, b, \dots b]$$

$$\Rightarrow [w^T x^{(1)} + w^T x^{(2)} + \dots + w^T x^{(m)}] + [b, b, \dots b]$$

$1 \times m$

$$Z = [z^{(1)}, z^{(2)}, \dots z^{(m)}] \Rightarrow [\underbrace{w^T x^{(1)} b}_{z^{(1)}} + \underbrace{w^T x^{(2)} b}_{z^{(2)}} + \dots + \underbrace{w^T x^{(m)} b}_{z^{(m)}}]$$

$$Z \Rightarrow \text{np.dot}(w^T X) + b$$

Although b is a single value while adding with the vector multiplication, it spreads like the above. for addition

$$A = [a^{(1)}, a^{(2)}, \dots a^{(m)}] = \sigma(z)$$

Python Broadcasting

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression's Gradient Computation

Vectorizing Logistic Regression

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

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

$1 \times m$

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

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

→ $dw = 0$

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

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

⋮

$dw /= m$

$db = 0$

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

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

⋮

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

$db /= m.$

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

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

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

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

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

$n \times 1$

Andrew Ng

Activation function $\Rightarrow A \Rightarrow [a^{(1)}, a^{(2)}, \ldots, a^{(m)}] \Rightarrow \sigma(z)$

Dependent variable $\Rightarrow Y \Rightarrow [y^{(1)}, y^{(2)}, \ldots y^{(m)}]$

$dz \Rightarrow [dz^{(1)}, dz^{(2)}, dz^{(3)}, \ldots, dz^{(m)}]$

$dz \Rightarrow A - Y \Rightarrow [a^{(1)}-y^{(1)} ; a^{(2)}-y^{(2)} ; \ldots , a^{(m)}-y^{(m)}]$

Removed 1 for loop

Looping over all the training examples

$dw \Rightarrow 0$

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

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

$\vdots$

$dw \Rightarrow dw/m$

$db \Rightarrow 0$

$db \Rightarrow db + dz^{(1)}$

$db \Rightarrow db + dz^{(2)}$

$\vdots$

$db \Rightarrow db/m$

Derivative of the intercept

Derivative of the slope

We can exclude the for-loop by introducing Vectorization

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

$dw \Rightarrow \frac{1}{m} X (dz)^T$

$\Rightarrow \frac{1}{m} \begin{bmatrix} x^{(1)} & x^{(2)} & \cdots & x^{(m)} \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ dz^{(2)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$

Vectorized Format

$dw \Rightarrow 1/m \left[x^{(1)} dz^{(1)} + x^{(2)} dz^{(2)} + \cdots x^{(m)} dz^{(m)} \right]$

Implementing Logistic Regression

Non vectorized form

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

for i = 1 to m:

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

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

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

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

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

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

$dw_1 += x^{(i)} * dz^{(i)}$

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

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

$db = db/m$

Vectorized Form

for iter in range(1000):

$Z = w^T X + b$

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

$A = \sigma(Z)$

$dZ = A - Y$

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

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

$w := w - \alpha dw$

$b := b - \alpha db$

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Broadcasting in Python

From the data-set

↳ 100 grams of Apple contains 56 calories from carbohydrates.

↳ 100 grams of Beef contains 104 calories from proteins.

↳ 100 grams of Eggs contains 99 calories from Fat.
52 calories from protein.
4.4 calories from carbohydrates.

Question

Calculate the % of calories from carbohydrates, protein and fat for each food ?

Let's take Apple

	(1)
	56.0
	1.2
	1.8
	59.0

% of calories from carbohydrate ⇒ $56/59$ ⇒ 94.9%

protein ⇒ $1.2/59.0$ ⇒ 2%

fat ⇒ $1.8/59.0$ ⇒ 3%

Calculate all the % of calories from the foods without using explicit for loops ?

Broadcasting example

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

	Apples	Beef	Eggs	Potatoes
Carb	56.0	0.0	4.4	68.0
Protein	1.2	104.0	52.0	8.0
Fat	1.8	135.0	99.0	0.9

$= A$ (3,4)

59 cal

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

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

```
cal = A.sum(axis = 0)
percentage = 100*A/(cal.reshape(1,4))
```

(3,4) / (1,4)

Broadcasting example

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

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

(m, n) (2,3) (1, n) ⇝ (m, n) (2,3)

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

(m, n) (m, 1) ⇝ (m, n)

Question - 1

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix}_{m\times 1} + \underbrace{100}_{1\times 1} \Rightarrow \text{Expanding this 100} \Rightarrow \begin{bmatrix}1\\2\\3\\4\end{bmatrix}_{m\times 1} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix}_{m\times 1} \Rightarrow \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

Question-2

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}_{m\times n} + \underbrace{\begin{bmatrix}100 & 200 & 300\end{bmatrix}}_{1\times n} \Rightarrow \begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}_{m\times n} + \begin{bmatrix}100 & 200 & 300\\100 & 200 & 300\end{bmatrix}_{n\times m} \Rightarrow \begin{bmatrix}101 & 202 & 303\\104 & 205 & 306\end{bmatrix}$$

Question-3

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}_{m\times n} + \underbrace{\begin{bmatrix}100\\200\end{bmatrix}}_{m\times 1} \Rightarrow \begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}_{m\times n} + \begin{bmatrix}100 & 100 & 100\\200 & 200 & 200\end{bmatrix}_{m\times n} \Rightarrow \begin{bmatrix}101 & 102 & 103\\204 & 205 & 206\end{bmatrix}$$

General Principle

converting

Main Matrix (m, n) matrix $+ \; - \; * \; /$ $(1, n) \rightsquigarrow (m, n)$

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

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

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

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

Matlab/Octave: bsxfun

deeplearning.ai

Basics of Neural Network Programming

Explanation of logistic regression cost function (Optional)

Logistic regression cost function

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

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

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

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

Logistic regression cost function

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

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

$p(y|x)$

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

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

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

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

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

Andrew Ng

$$P(y|x) \Rightarrow (\hat{y})^{y}(1-\hat{y})^{(1-y)}$$

If $y \to 1 \Rightarrow P(y|x) \to (\hat{y})^{1}(1-\hat{y})^{(1-1)} \Rightarrow \hat{y}(1-\hat{y})^{0} \Rightarrow \hat{y}$

$y \to 0 \Rightarrow P(y|x) \Rightarrow (\hat{y})^{0}(1-\hat{y})^{(1-0)} \to 1*(1-\hat{y}) \Rightarrow 1-\hat{y}$

proof

Positive example $\Rightarrow P(y|x) \Rightarrow \hat{y}$

Negative example $\Rightarrow P(y|x) \Rightarrow 1-\hat{y}$

Applying log on both sides

$$\log[P(y|x)] \Rightarrow \log[(\hat{y})^{y}(1-\hat{y})^{(1-y)}]$$

$\log(ab) \Rightarrow \log(a) + \log(b)$

$$\Rightarrow \log(\hat{y}^{y}) + \log(1-\hat{y})^{(1-y)}$$

$$\log[P(y|x)] \Rightarrow y\log(\hat{y}) + (1-y)\log(1-\hat{y}) \Rightarrow -L(\hat{y}, y)$$

Logistic Regression: While training an algorithm we want the probabilities to be large

Minimizing the Loss is equal to Maximizing the probability.

$$\log[P(y|x)] \Rightarrow -\text{Loss}(\hat{y}, y)$$

Cost on m examples

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

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

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

Maximum likelihood estimation

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

For minimizing the cost function, we are using the Maximum Likelihood Estimation