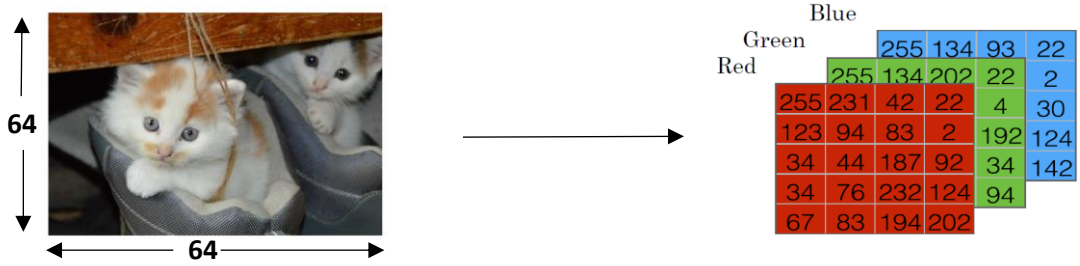
Binary Classification

In a binary classification problem, the result is a discrete value output.

For example - account hacked (1) or compromised (0)
- a tumor malign (1) or benign (0)

Example: Cat vs Non-Cat

The goal is to train a classifier that the input is an image represented by a feature vector, x, and predicts whether the corresponding label y is 1 or 0. In this case, whether this is a cat image (1) or a non-cat image (0).



An image is store in the computer in three separate matrices corresponding to the Red, Green, and Blue color channels of the image. The three matrices have the same size as the image, for example, the resolution of the cat image is 64 pixels X 64 pixels, the three matrices (RGB) are 64 X 64 each.

The value in a cell represents the pixel intensity which will be used to create a feature vector of n-dimension. In pattern recognition and machine learning, a feature vector represents an object, in this case, a cat or no cat.

To create a feature vector, x, the pixel intensity values will be "unroll" or "reshape" for each color. The dimension of the input feature vector x is $n_x = 64\ x\ 64\ x\ 3 = 12\ 288$.

$$x = \begin{bmatrix} 255 \\ 231 \\ 42 \\ \vdots \\ 255 \\ 134 \\ 202 \\ \vdots \\ 255 \\ 134 \\ 93 \\ \vdots \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \text{red} \\ \\ \\ \\ \text{green} \\ \\ \\ \\ \text{blue} \\ \\ \end{matrix}$$

Logistic Regression

Logistic regression is a learning algorithm used in a supervised learning problem when the output y are all either zero or one. The goal of logistic regression is to minimize the error between its predictions and training data.

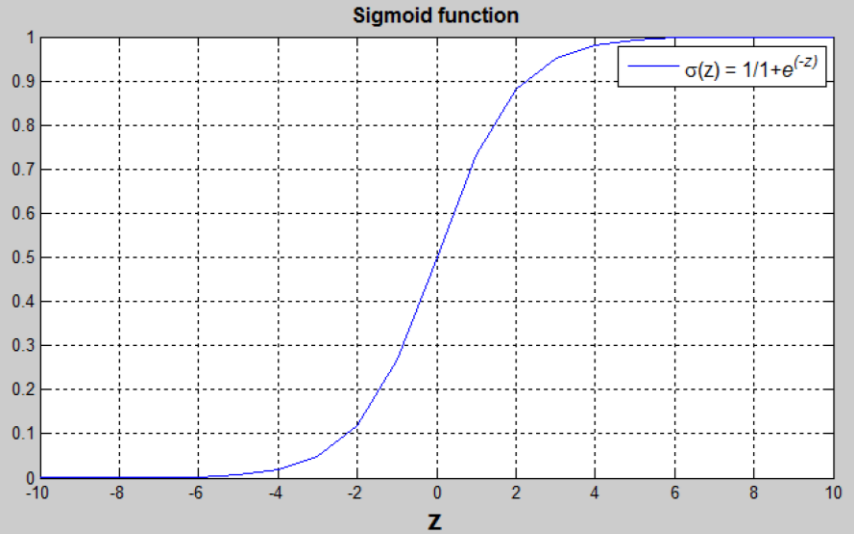
Example: Cat vs No - cat

Given an image represented by a feature vector x, the algorithm will evaluate the probability of a cat being in that image.

$$Given\ x\ ,\ \hat{y} = P(y = 1|x), \text{where } 0\ \leq \hat{y} \leq 1$$

The parameters used in Logistic regression are:

- The input features vector: $x\ \in \mathbb{R}^{n_x}$, where n_x is the number of features
- The training label: $y \in 0,1$
- The weights: $w\ \in \mathbb{R}^{n_x}$, where n_x is the number of features
- The threshold: $b\ \in \mathbb{R}$
- The output: $\hat{y} = \sigma(w^T x + b)$
- Sigmoid function: s = $\sigma(w^T x + b)$ = $\sigma(z)$= $\frac{1}{1+ e^{-z}}$



$(w^T x + b)$ is a linear function $(ax + b)$, but since we are looking for a probability constraint between [0,1], the sigmoid function is used. The function is bounded between [0,1] as shown in the graph above.

Some observations from the graph:

- If z is a large positive number, then $\sigma(z)$ = 1
- If z is small or large negative number, then $\sigma(z)$ = 0
- If $z = 0$, then $\sigma(z)$ = 0.5

Logistic Regression: Cost Function

To train the parameters w and b, we need to define a cost function.

Recap:

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

$x^{(i)}$ the i-th training example

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

Loss (error) function:

The loss function measures the discrepancy between the prediction ($\hat{y}^{(i)}$) and the desired output ($y^{(i)}$). In other words, the loss function computes the error for a single training example.

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

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

- If $y^{(i)} = 1$: $L(\hat{y}^{(i)}, y^{(i)}) = -\log(\hat{y}^{(i)})$ where $\log(\hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 1
- If $y^{(i)} = 0$: $L(\hat{y}^{(i)}, y^{(i)}) = -\log(1 - \hat{y}^{(i)})$ where $\log(1 - \hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 0

Cost function

The cost function is the average of the loss function of the entire training set. We are going to find the parameters $w\ and\ b$ that minimize the overall cost function.

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

deeplearning.ai

Basics of Neural Network Programming

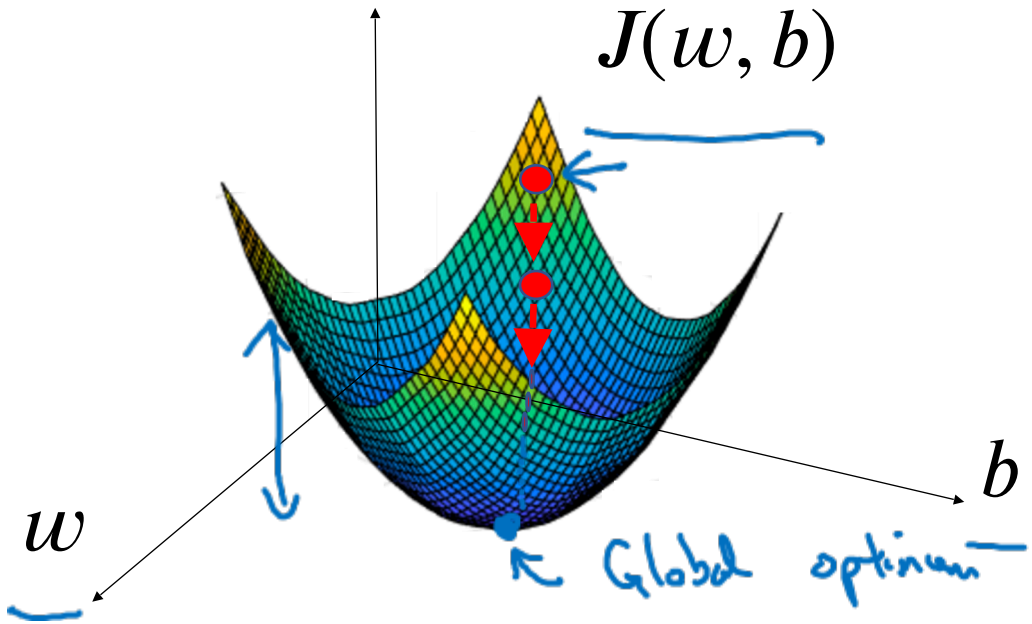
Gradient Descent

Gradient Descent

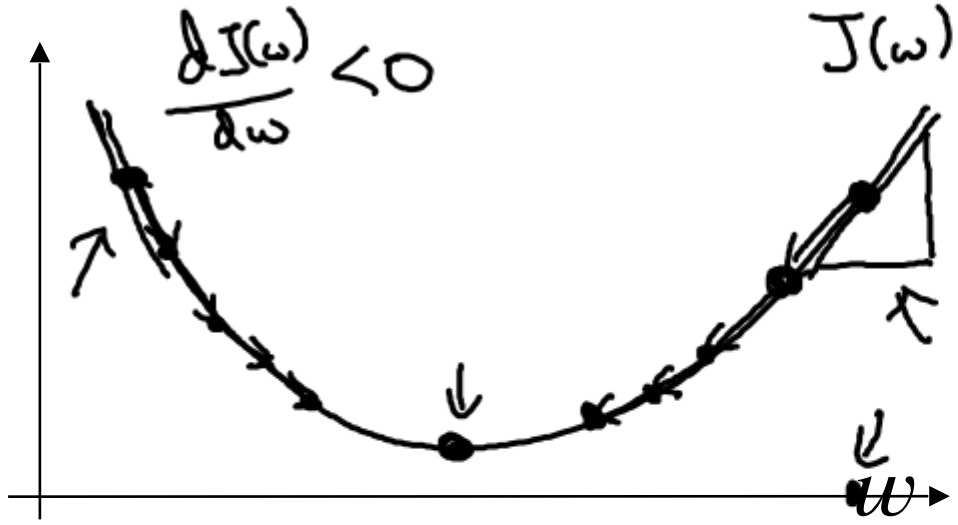
Recap: ,

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \square\,\square - \frac{1}{m}\sum_{i=1}^{m}$$

Want to find that minimize



Gradient Descent



Repeat {

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

"dw"

}

$w := w - \alpha dw$

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

$J(w,b)$

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

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

∂ "partial derivative"

J

Andrew Ng

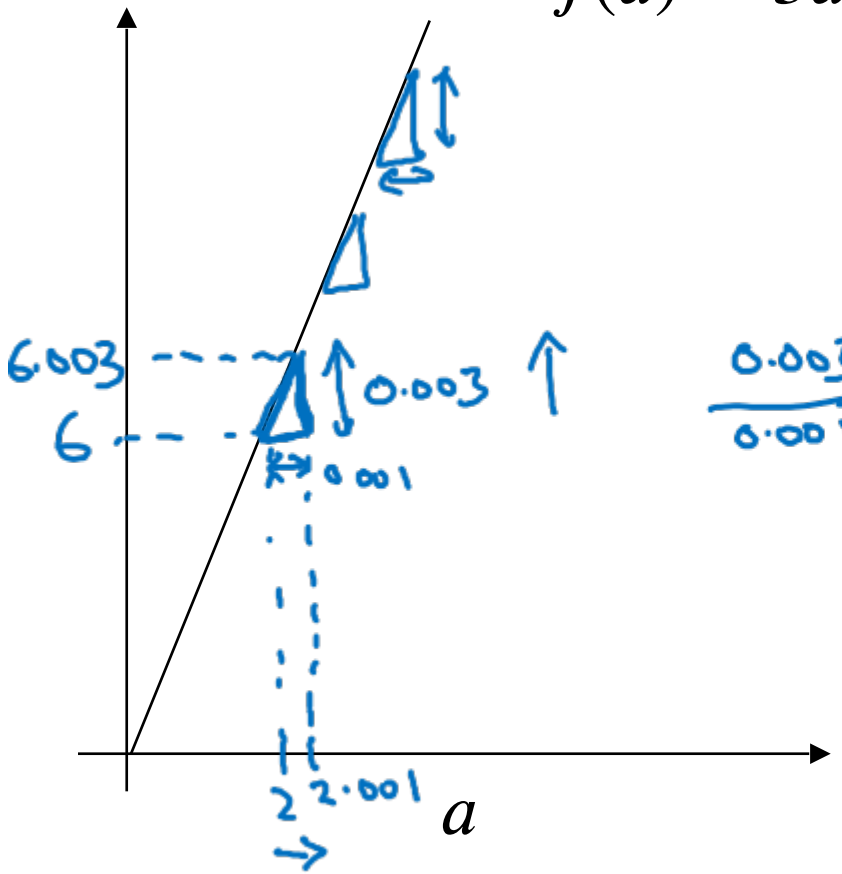
deeplearning.ai

Basics of Neural Network Programming

Derivatives

Intuition about derivatives

$f(a) = 3a$



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

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

→ a = 2 f(a) = 6

a = 2.001 f(a) = 6.003

→ a = 5 f(a) = 15

a = 5.001 f(a) = 15.003

slope at a=5 is also 3

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

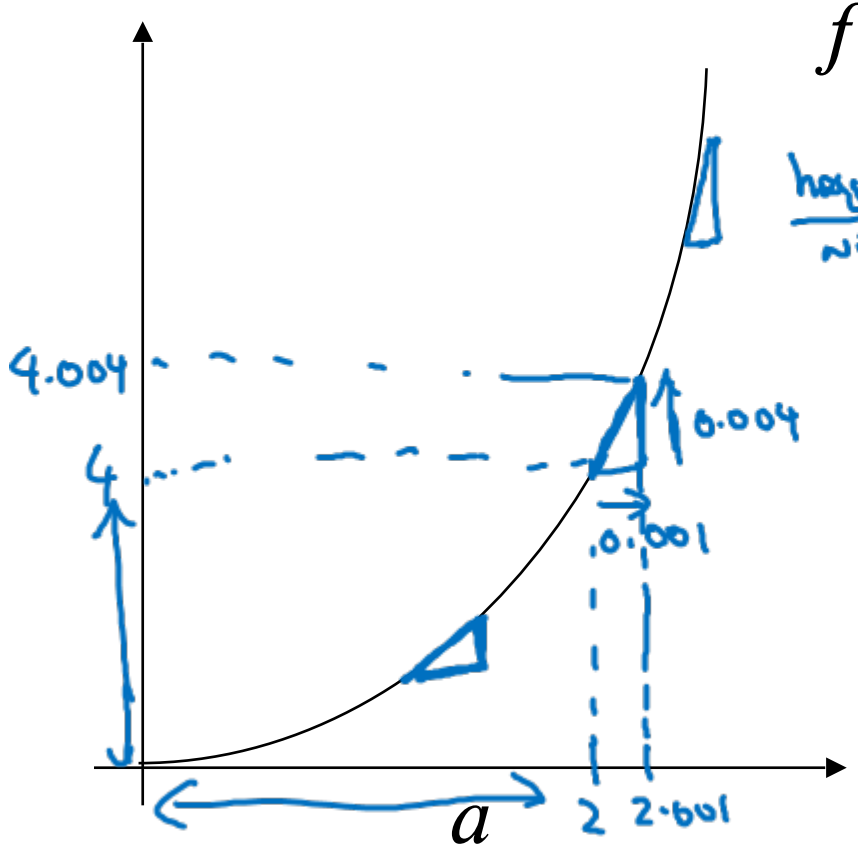
0.001

0.000000001

0.000000000001

Intuition about derivatives

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



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

0.001

$(2a) \times 0.001$

0.001 ←

$0.000000\ldots01$ ←

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

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

$(4.004\boxed{004})$

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

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

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

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

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

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

More derivative examples

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

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

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

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

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

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

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

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

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

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

0.0005 $\quad 0.0005$

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Computation Graph

Computation Graph

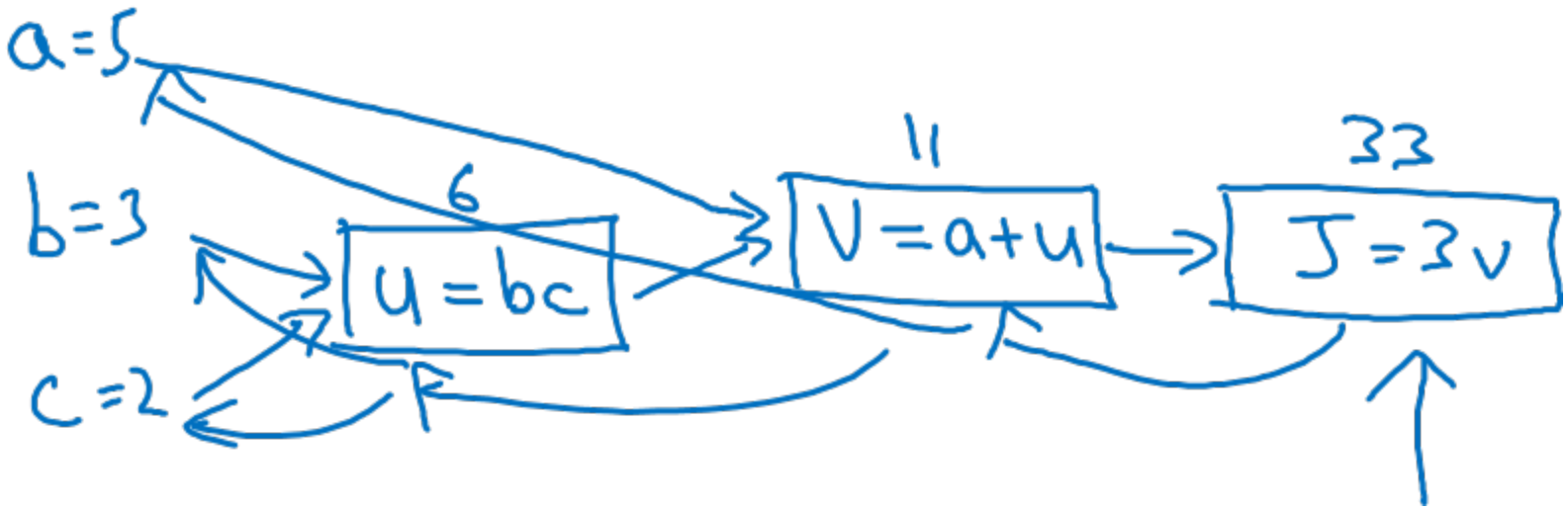
$J(a,b,c) = 3(a+bc) = 3(5+3\times 2) = 33$

u, v, J

$u = bc$

$V = a+u$

$J = 3v$

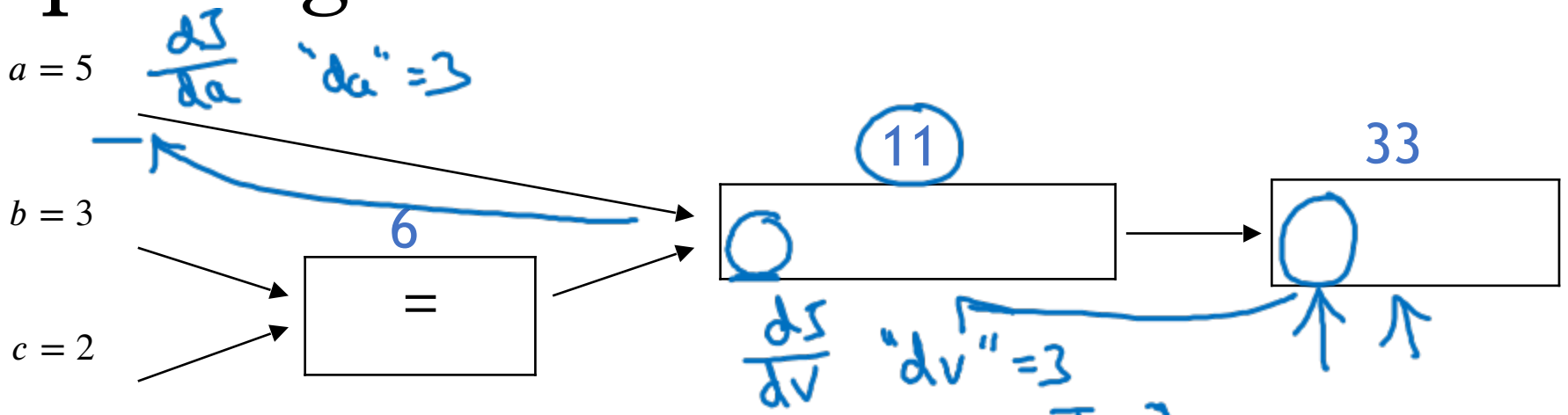


deeplearning.ai

Basics of Neural Network Programming

Derivatives with a Computation Graph

Computing derivatives



$J = 3v$

$v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

$a \rightarrow v \rightarrow J$

$f(a) = 3a$

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

$J = 3v$

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

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

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

3×1

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

$a = 5 \rightarrow 5.001$

$\rightarrow v = 11 \rightarrow 11.001$

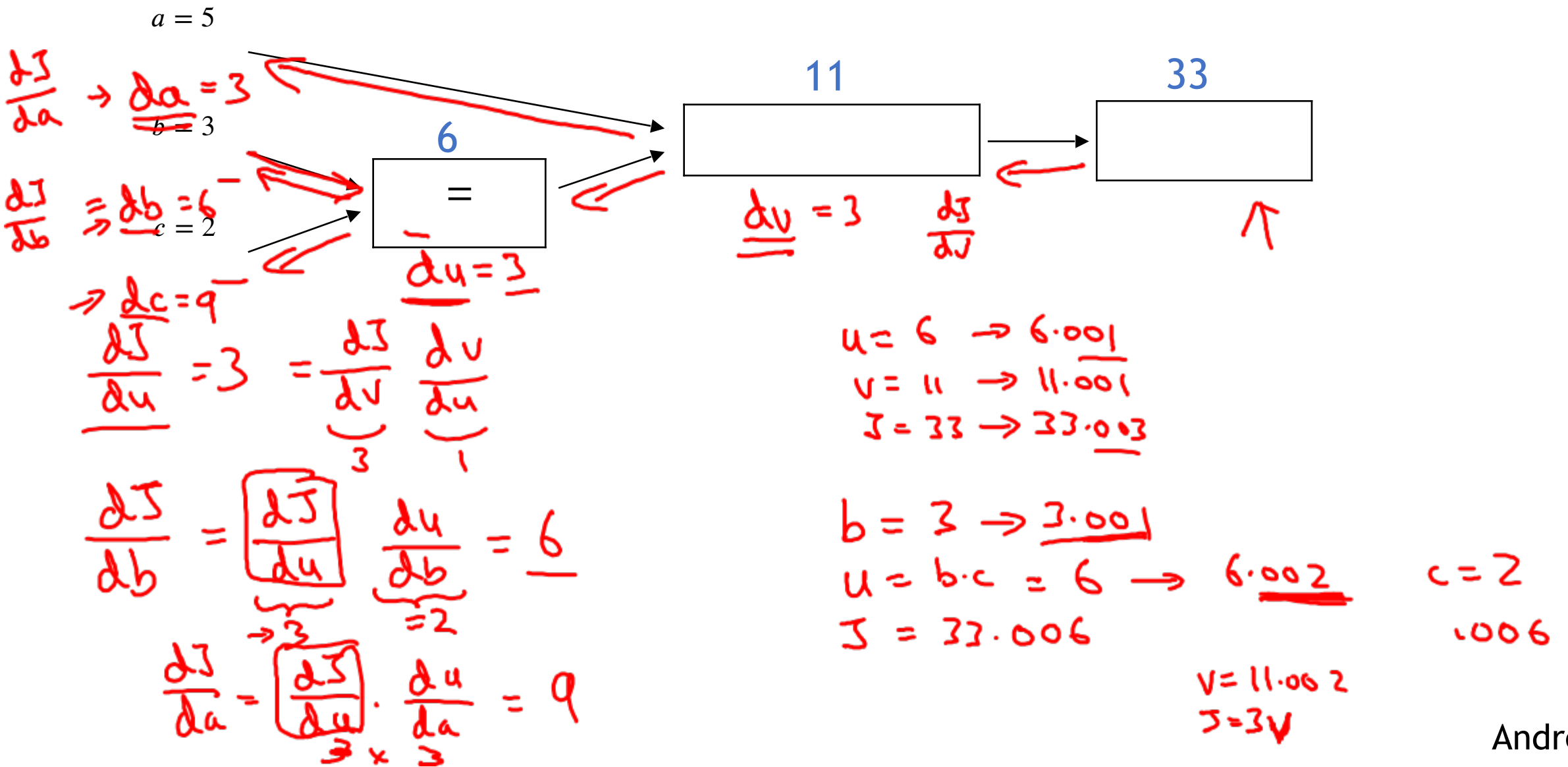
$J = 33 \rightarrow 33.003$

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

dJ dvar

Andrew Ng

Computing derivatives



Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

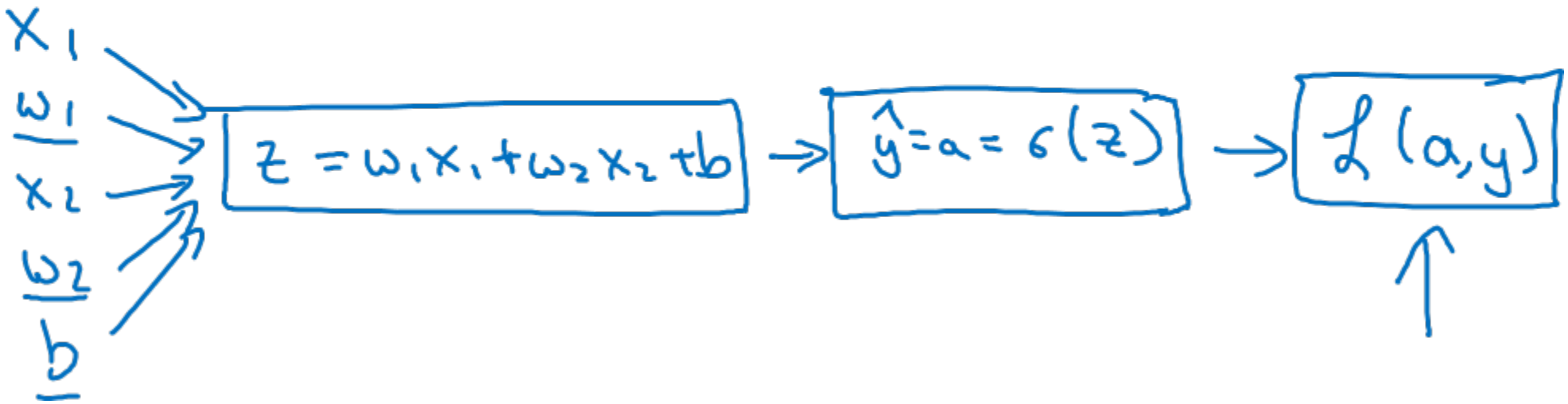
Logistic Regression Gradient descent

Logistic regression recap

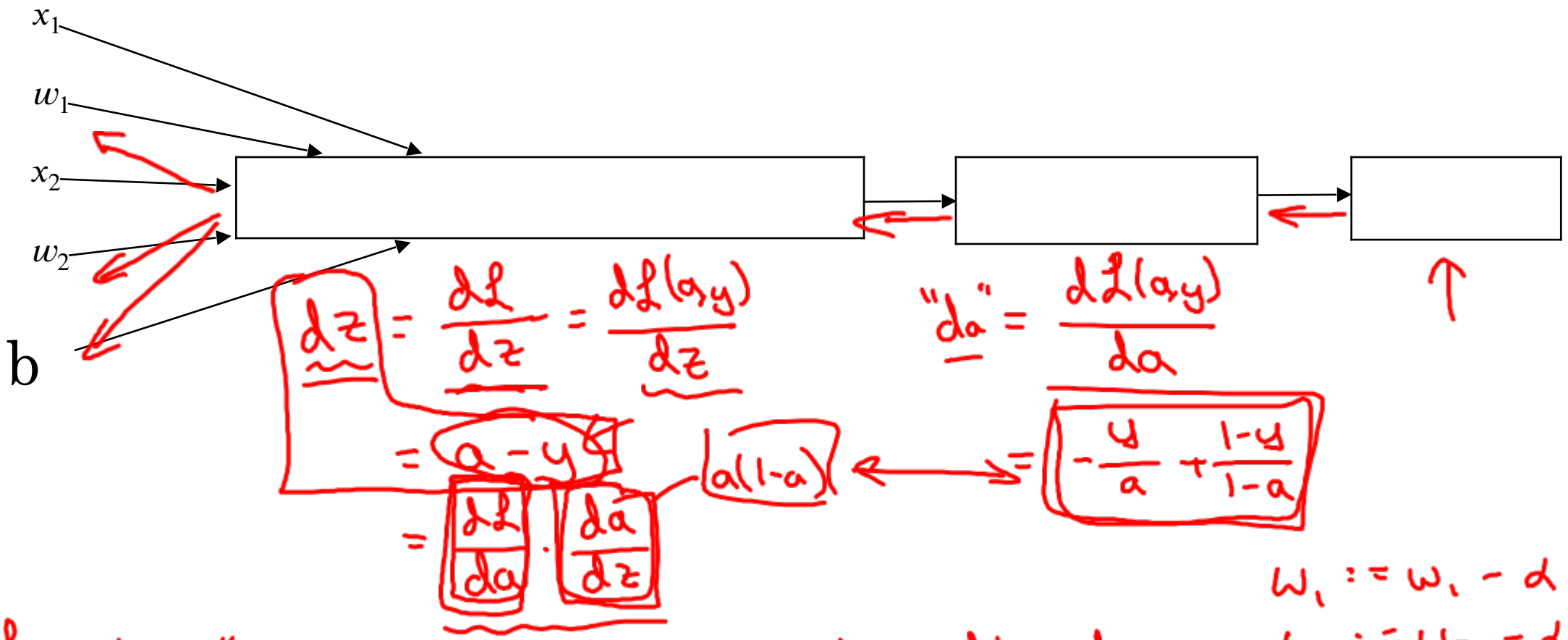
$$z = w^T x + b$$

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

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



Logistic regression derivatives



Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Gradient descent on m examples

Logistic regression on m examples

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

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

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

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

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

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

Andrew Ng

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.

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

More vectorization examples

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

$u = Av$

$u_i = \sum_i \sum_j A_{ij} v_j$

```
u = np.zeros((n,1))
for i ...
    for j ...
        u[i] += A[i][j] * v[j]
```

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

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
```

Logistic regression derivatives

```
J = 0, dw1 = 0, dw2 = 0, db = 0
for i = 1 to n:
        =
        =
       +=
       d =
       d+=
       d+=
       db += d
J = J/m, d = d/m, d = d/m, db = db/m
```

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

for $j = 1 \ldots n_x$: $dw_j += \ldots$

$n_x = 2$

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

$dw /= m$.

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

Vectorizing Logistic Regression

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

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

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

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

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

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

$1 \times m$ $\qquad$ $1 \times m$

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

$(1,1)$ $\qquad$ $\mathbb{R}$ $\qquad$ "Broadcasting"

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

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression's Gradient Computation

Vectorizing Logistic Regression

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

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

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

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

$\rightarrow dw = 0$

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

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

⋮

$dw /= m$

$db = 0$

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

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

⋮

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

$db /= m.$

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

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

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

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

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

$n \times 1$

Andrew Ng

Implementing Logistic Regression

```
J = 0, d = 0, d = 0, db = 0
for i = 1 to m:
      =
      =
     +=
     d =
    d+=
    d+=
     db += d
J = J/m, d = d/m, d = d/m
db = db/m
```

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

```
for iter in range(1000):
    Z = w^T X + b
      = np.dot(w.T, X) + b
    A = σ(Z)
    dZ = A - Y
    dw = 1/m X dZ^T
    db = 1/m np.sum(dZ)

    w := w - α dw
    b := b - α db
```

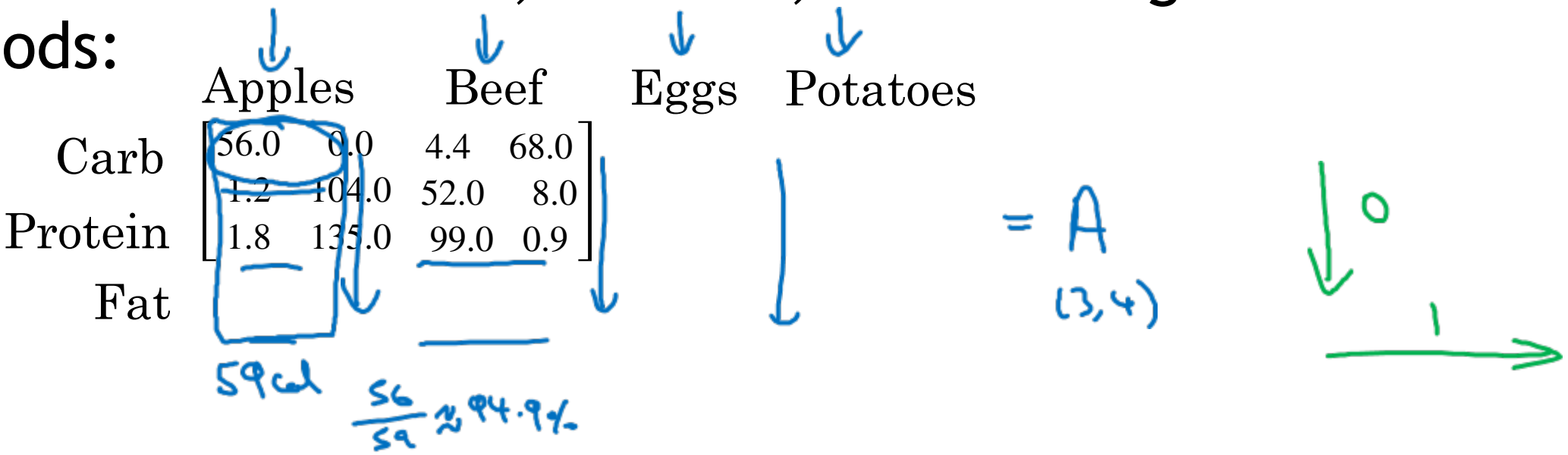
deeplearning.ai

Basics of Neural Network Programming

Broadcasting in Python

Broadcasting example

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



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

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

100

$$+ \begin{bmatrix}100 & 200 & 300\\100 & 200 & 300\end{bmatrix}$$

(m, n) (2,3) (1, n) ⇝ (m, n) (2,3)

$$+ \begin{bmatrix}100\\200\end{bmatrix}$$

(m, n) (m, 1) ⇝ (m, n)

$$= \begin{bmatrix}100 & 100 & 100\\200 & 200 & 200\end{bmatrix}$$

General Principle

(m, n) matrix

$+$ $-$ $*$ $/$

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

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

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

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

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

Matlab/Octave: bsxfun

deeplearning.ai

Basics of Neural Network Programming

A note on python/ numpy vectors

Python Demo

Python / numpy vectors

```
import numpy as np

a = np.random.randn(5)

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

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

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