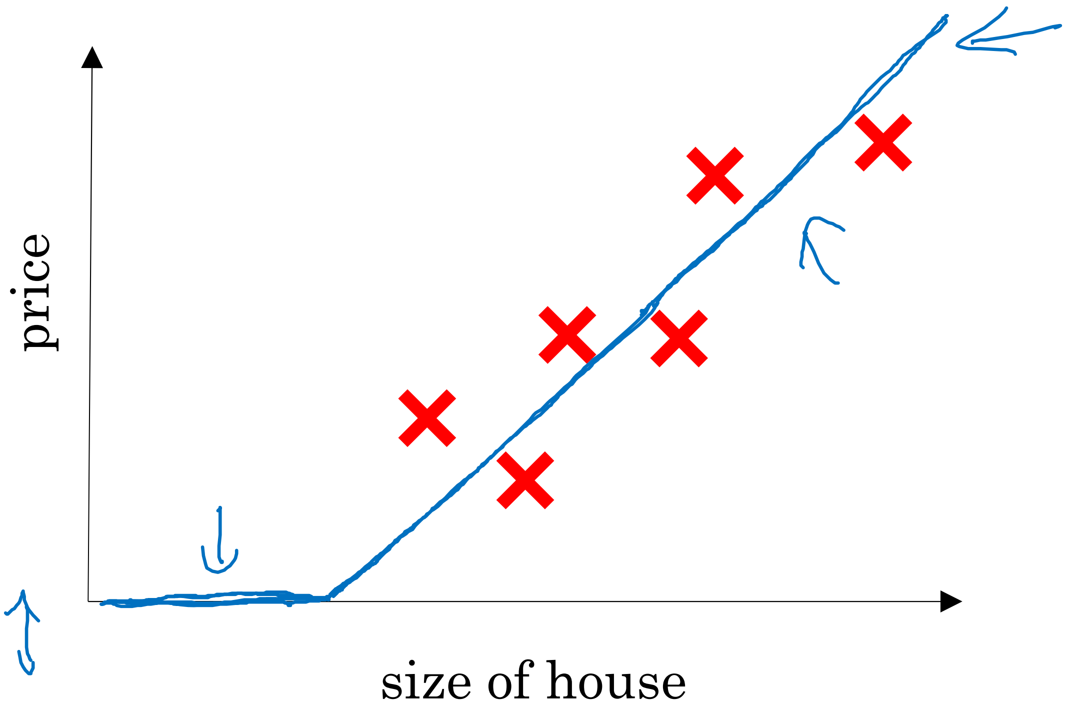
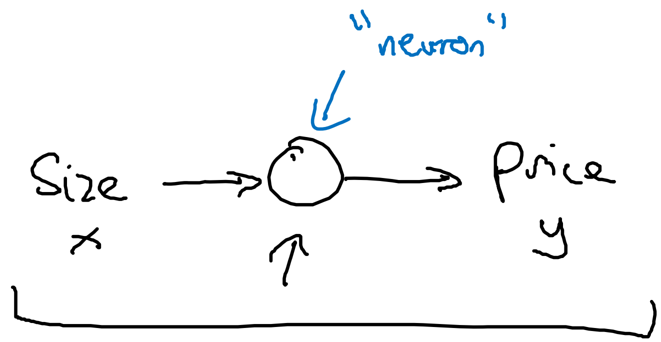
deeplearning.ai

Introduction to Deep Learning

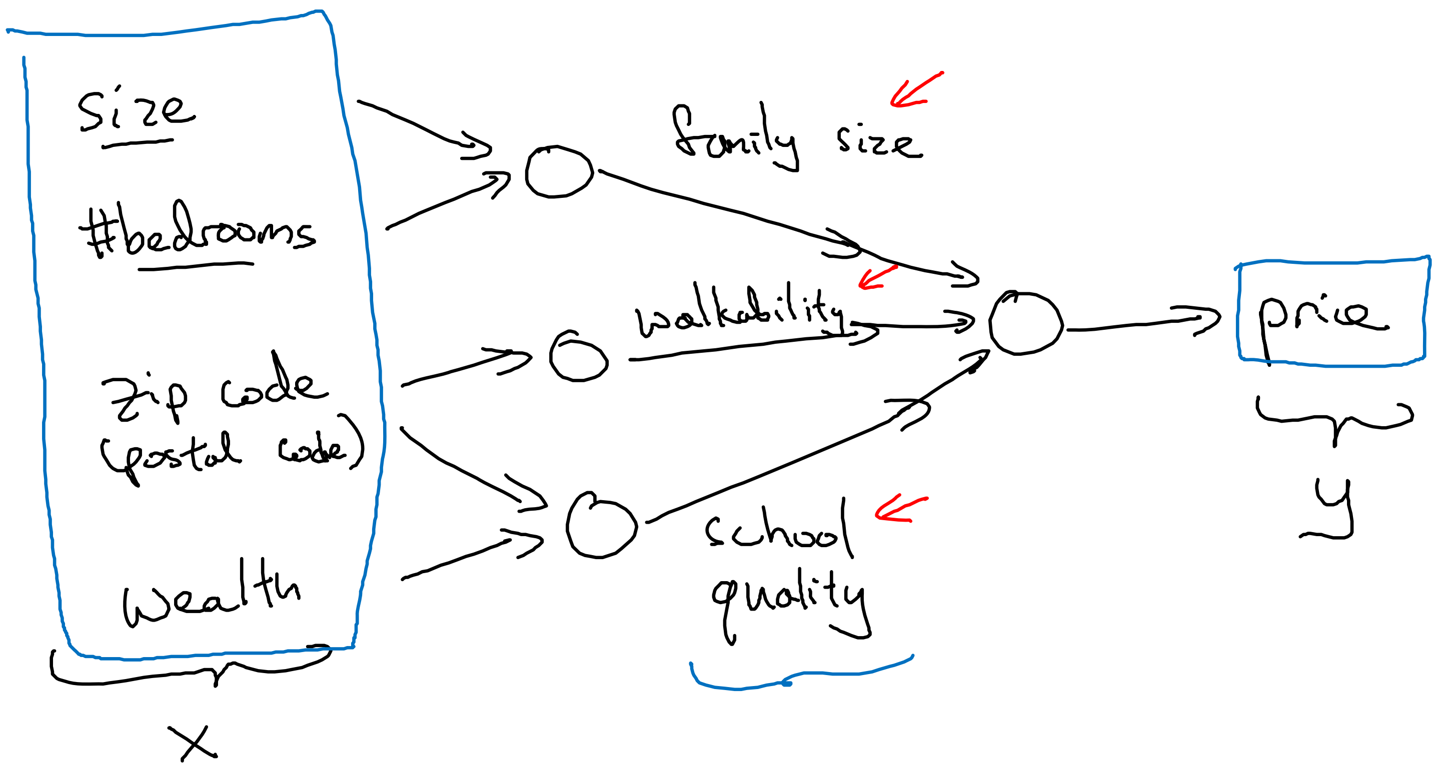
What is a Neural Network?

Housing Price Prediction



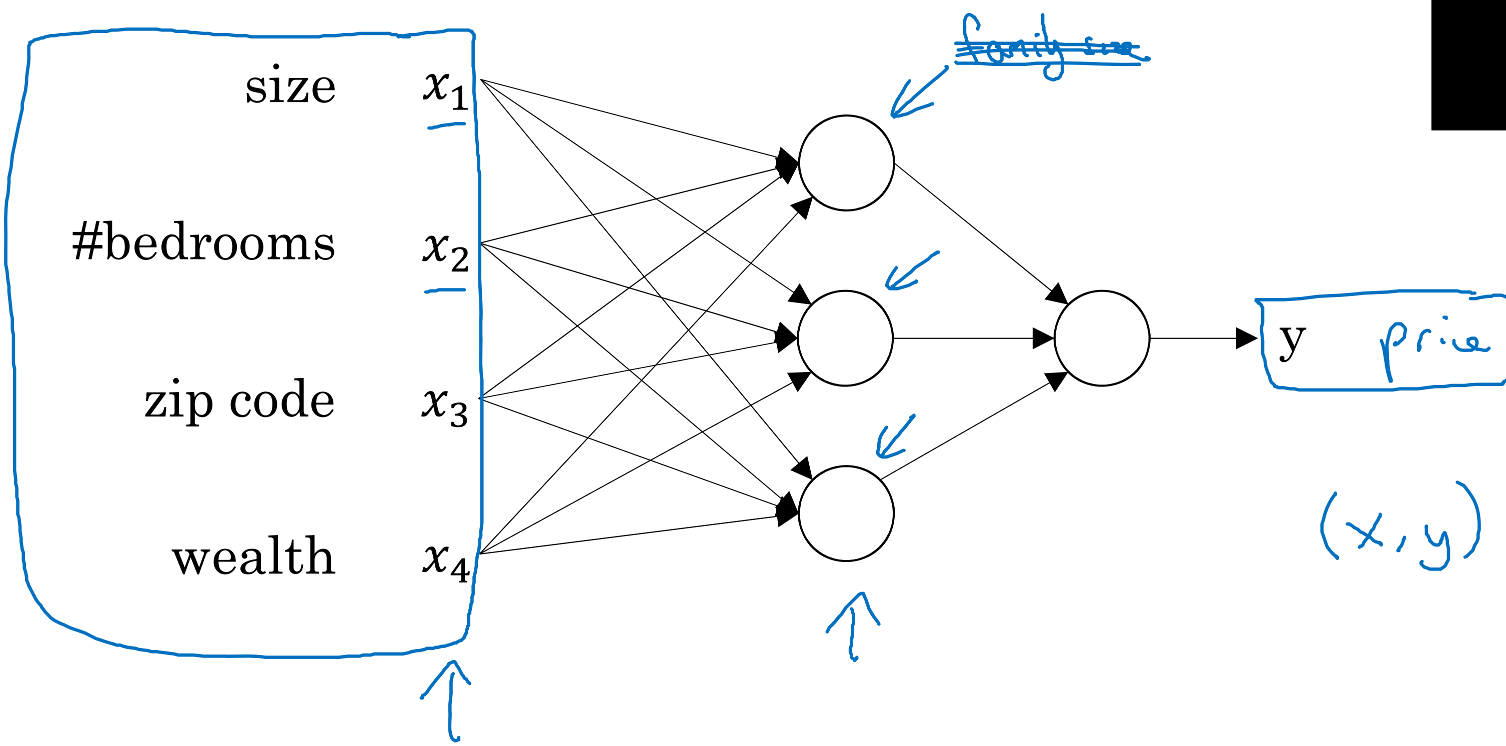


Housing Price Prediction



Housing Price Prediction

Drawing of previous Image



deeplearning.ai

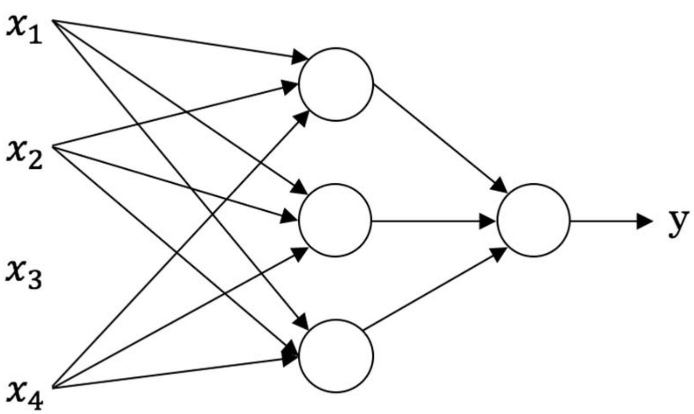
Introduction to Deep Learning

Supervised Learning with Neural Networks

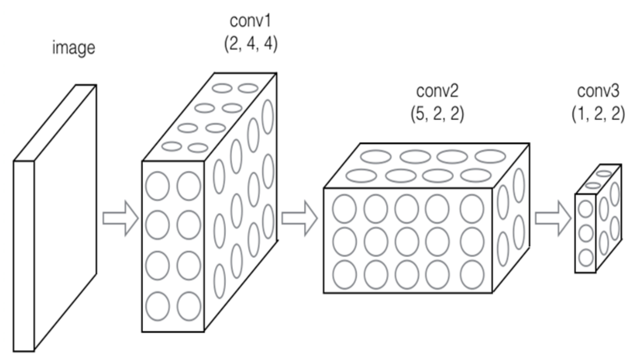
Supervised Learning

Input(x)	Output (y)	Application	
Home features	Price	Real Estate	Standard NN
Ad, user info	Click on ad? (0/1)	Online Advertising	
Image	Object (1,…,1000)	Photo tagging	CNN
Audio	Text transcript	Speech recognition	RNN
English	Chinese	Machine translation	
Image, Radar info	Position of other cars	Autonomous driving	Custom/ Hybrid

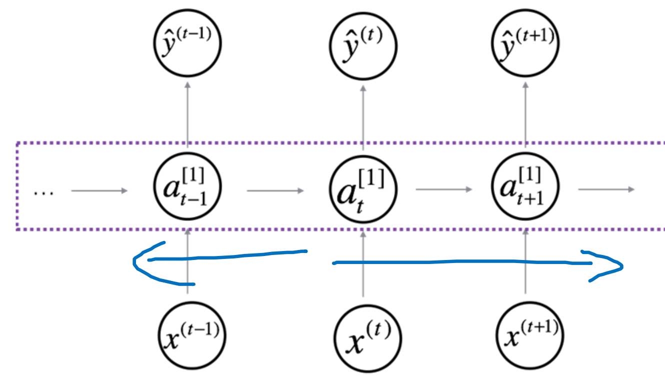
Neural Network examples



Standard NN



Convolutional NN



Recurrent NN

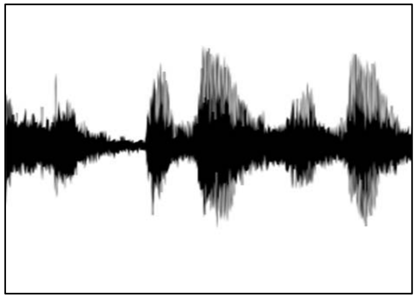
Supervised Learning

Structured Data

Size	#bedrooms	⋯	Price (1000$s)
2104	3		400
1600	3		330
2400	3		369
⋮	⋮		⋮
3000	4		540

User Age	Ad Id	⋯	Click
41	93242		1
80	93287		0
18	87312		1
⋮	⋮		⋮
27	71244		1

Unstructured Data



Audio

Image

Four scores and seven years ago…

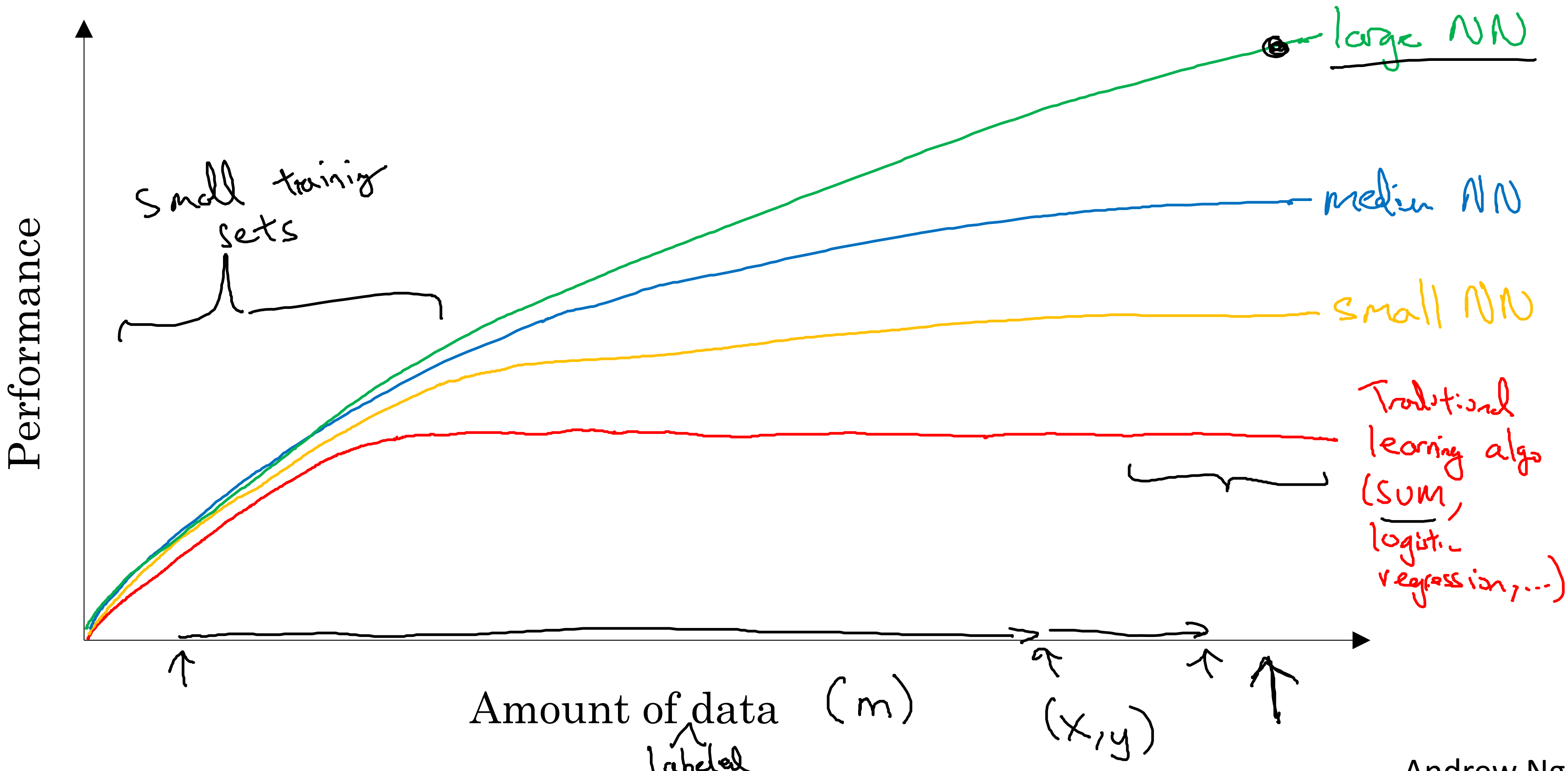
Text

deeplearning.ai

Introduction to Neural Networks

Why is Deep Learning taking off?

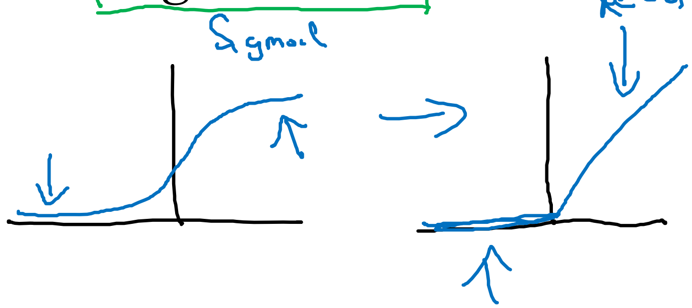
Scale drives deep learning progress



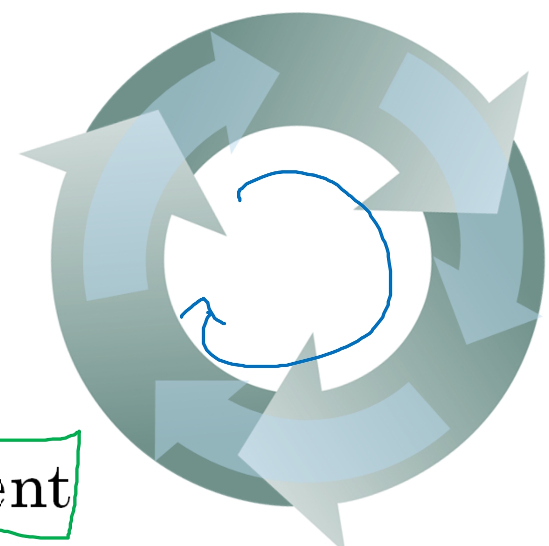
Andrew Ng

Scale drives deep learning progress

- Data
- Computation
- Algorithms



Idea



Experiment

Code

10 min
1 day
1 month

Andrew Ng

Basics of Neural Network Programming

Binary Classification

deeplearning.ai

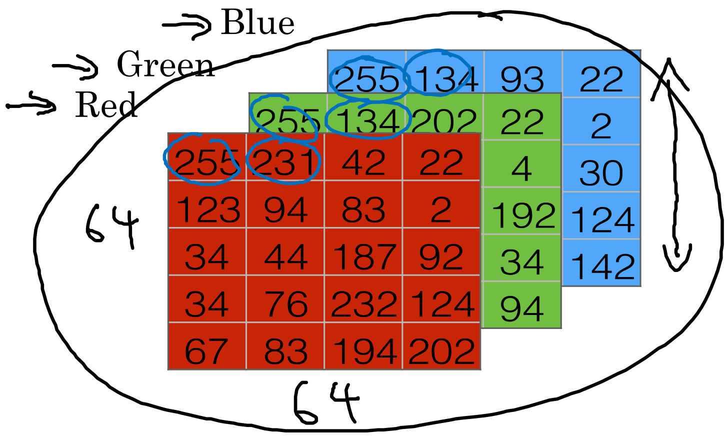
Binary Classification

64

64

$\longrightarrow$ 1 (cat) vs 0 (non cat)

y



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

$$64 \times 64 \times 3 = 12288$$

$$n = n_x = 12288$$

$$X \longrightarrow y$$

Andrew Ng

Notation

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

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

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

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

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

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

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

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

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Logistic Regression

Logistic Regression

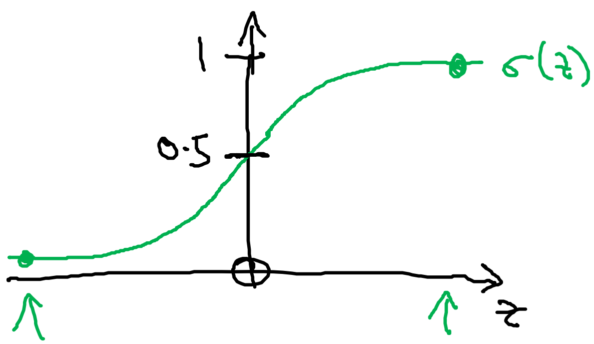
Given x, want $\hat{y} = P(y=1 \mid x)$

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

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

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

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



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

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

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

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

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

If z large negative number

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

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Logistic Regression cost function

Logistic Regression cost function

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

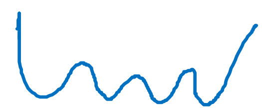
$z^{(i)} = w^T x^{(i)} + b$

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

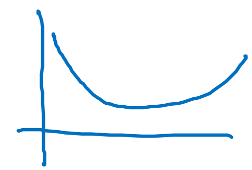
$x^{(i)}$
$y^{(i)}$
$z^{(i)}$

i-th example.

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



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



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

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

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

Andrew Ng

Basics of Neural Network Programming

Gradient Descent

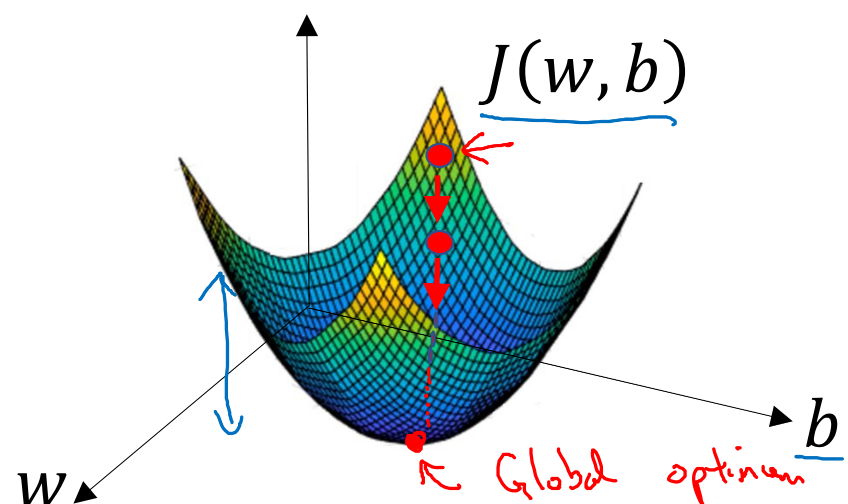
deeplearning.ai

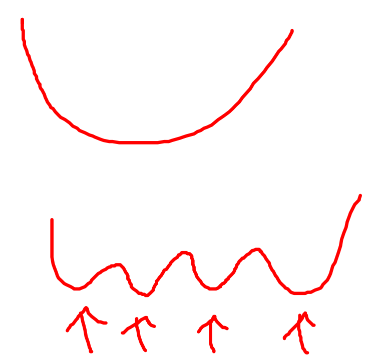
Gradient Descent

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

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

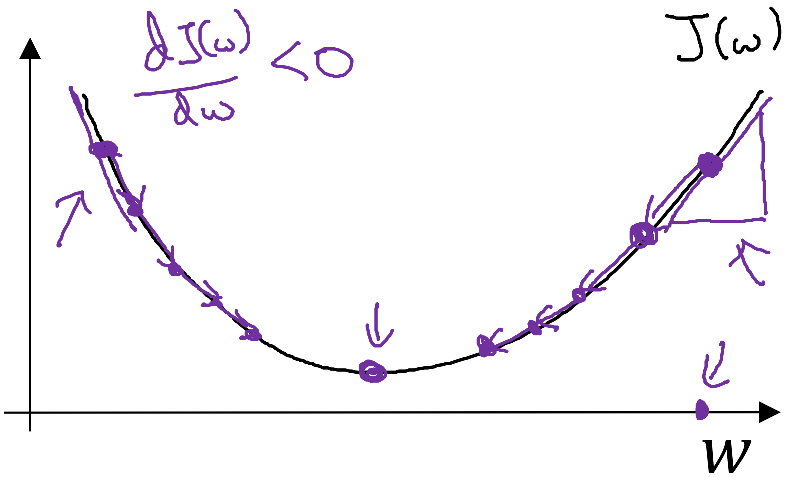
Want to find w, b that minimize $J(w,b)$





Andrew Ng

Gradient Descent



learning rate

Repeat {

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

}

"dw"

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

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

$J(w, b)$

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

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

∂ "partial derivative"

d J

Andrew Ng

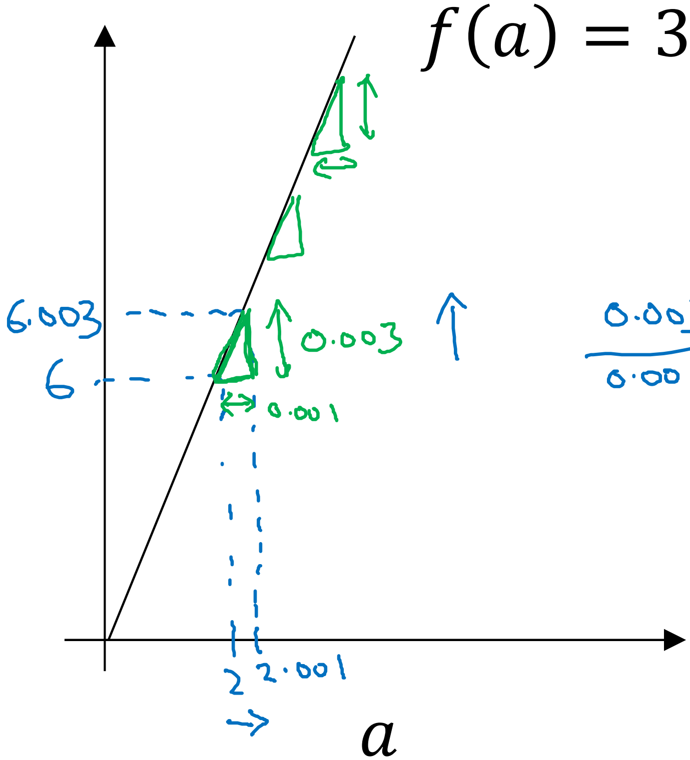
Basics of Neural Network Programming

Derivatives

deeplearning.ai

Intuition about derivatives

$f(a) = 3a$



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

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

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

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
0.00000001
0.0000000001

Andrew Ng

deeplearning.ai

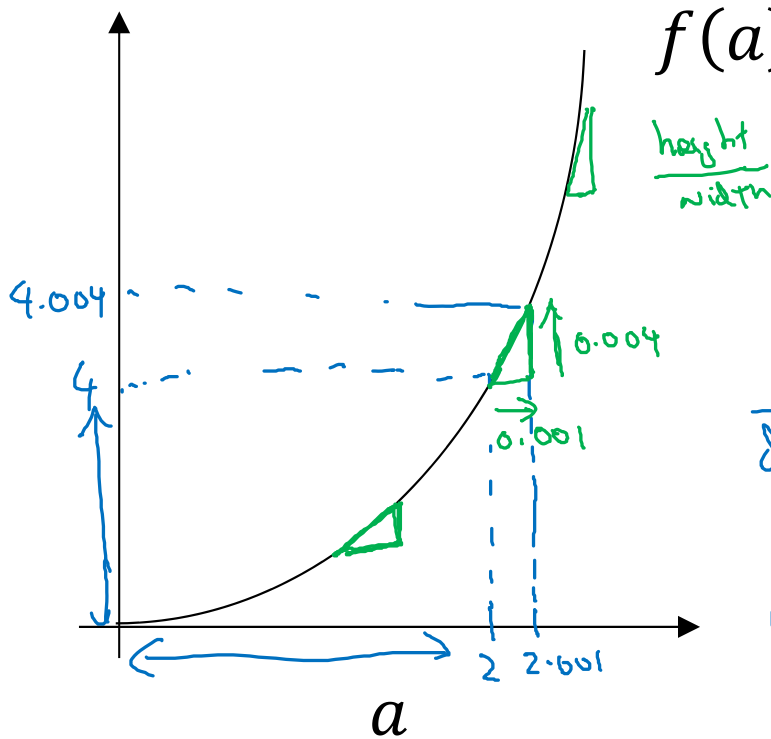
Basics of Neural Network Programming

More derivatives examples

Intuition about derivatives

0.001
$0.00000\ldots01$

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



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

0.001

$(2a) \times 0.001$

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

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

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

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

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

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

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

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

Andrew Ng

More derivative examples

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

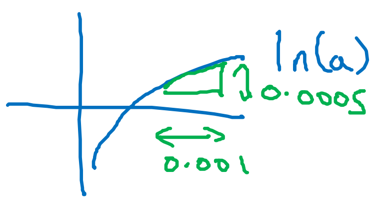
$\qquad a = 2.001 \quad f(a) \approx 4.004$

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

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

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

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



$\frac{d}{da} f(a) = \boxed{\frac{1}{2}}$ $\qquad 0.0005 \qquad 0.0005$

Andrew Ng

deeplearning.ai

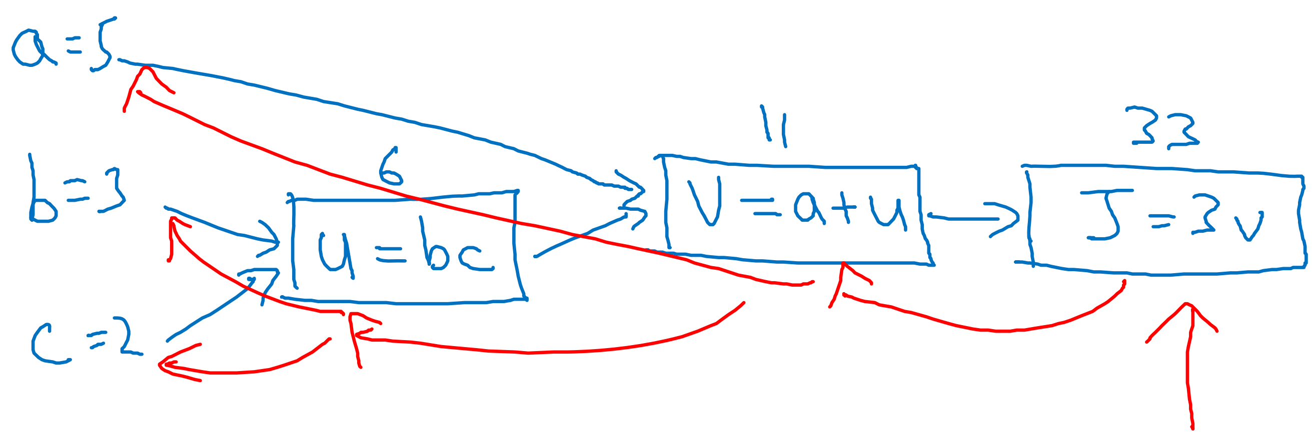
Basics of Neural Network Programming

Computation Graph

Computation Graph

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

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



Andrew Ng

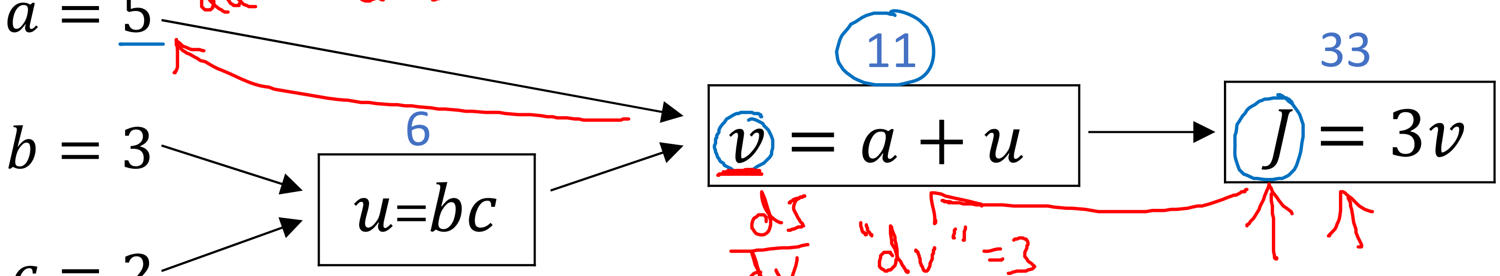
deeplearning.ai

Basics of Neural Network Programming

Derivatives with a Computation Graph

Computing derivatives

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



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

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

$a \rightarrow v \rightarrow J$

$J = 3v$

$v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

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

3×1

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

$a = 5 \rightarrow 5.001$

$\rightarrow v = 11 \rightarrow 11.001$

$J = 33 \rightarrow 33.003$

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

dJdvar

$\rightarrow$ "dvar"

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

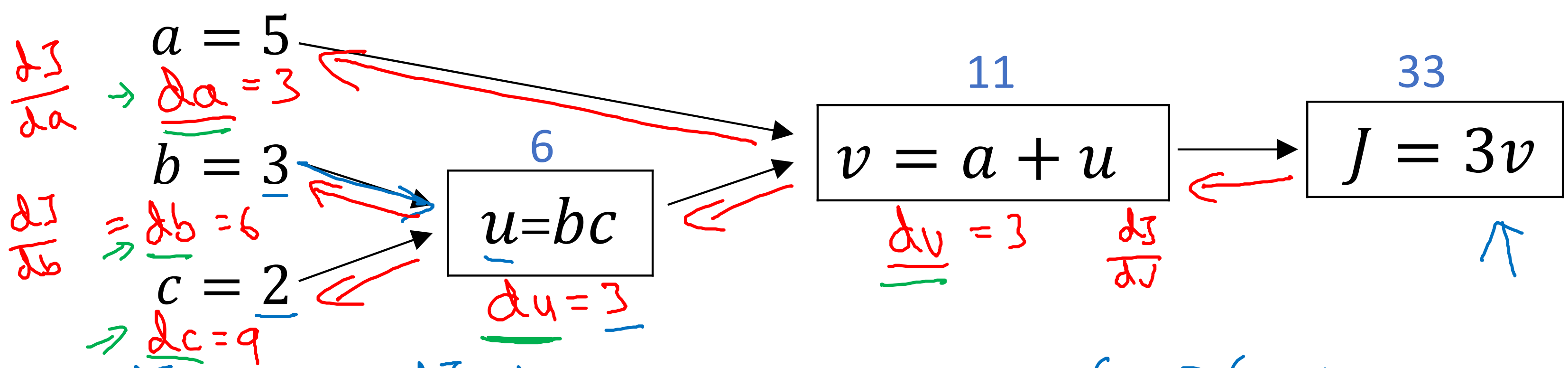
$\frac{df(a)}{da} = \frac{df}{da} = 3$

$\boxed{J = 3v}$

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

Andrew Ng

Computing derivatives



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

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

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

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

deeplearning.ai

Basics of Neural Network Programming

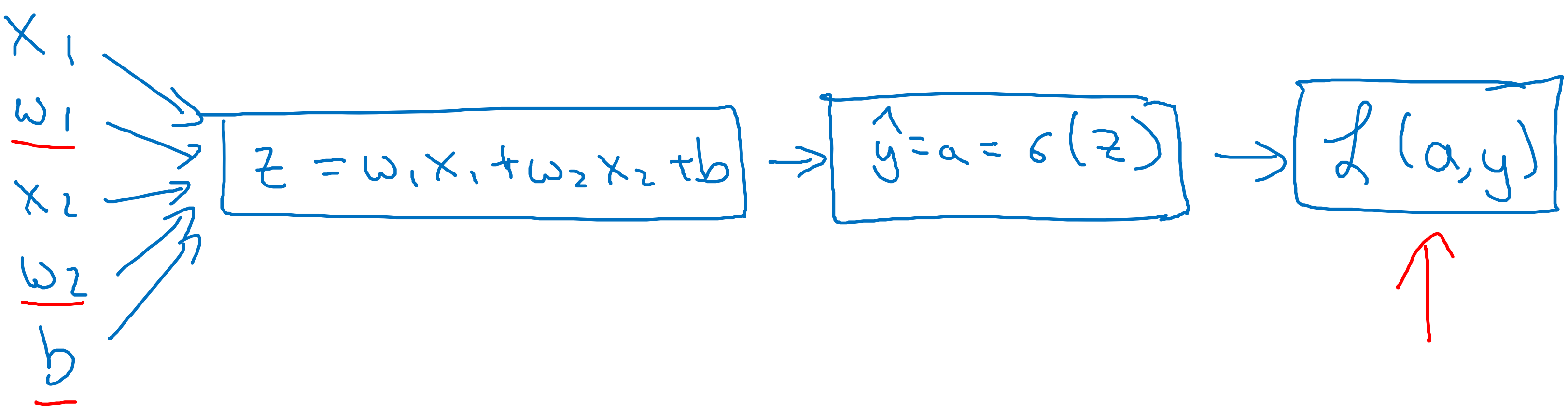
Logistic Regression Gradient descent

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

$$x_1,\ w_1,\ x_2,\ w_2,\ b \longrightarrow \boxed{z = w_1x_1 + w_2x_2 + b} \longrightarrow \boxed{a = \sigma(z)} \longrightarrow \boxed{\mathcal{L}(a, y)}$$

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

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

$$= a - y$$

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

$$a(1-a)$$

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

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

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

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

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Gradient descent on m examples

Logistic regression on m examples

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

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

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

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

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

Andrew Ng

Logistic regression on m examples

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

For $i=1$ to m

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

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

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

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

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

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

$n=2$

dw_3
$\vdots$
dw_n

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

$J /= m$

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

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

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

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

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

Vectorization

Basics of Neural Network Programming

Vectorization

deeplearning.ai

What is vectorization?

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

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

Non-vectorized:

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

Vectorized

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

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

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)
```

Andrew Ng

Vectors and matrix valued functions

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

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

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

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

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

Andrew Ng

Logistic regression derivatives

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

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

$\texttt{for i = 1 to n:}$

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

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

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

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

for $j = 1 \ldots n_x$

$dw_j \mathrel{+}= \ldots$

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

$n_x = 2$

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

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

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

$dw /= m.$

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

Vectorizing Logistic Regression

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

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

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

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

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

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

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

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

$1 \times m$

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

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

$\rightarrow dw = 0$

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

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

$\vdots$

$dw \mathrel{/}= m$

$\cancel{dw_1}$

$\cancel{dw_2}$

$\vdots$

$db = 0$

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

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

$\vdots$

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

$db \mathrel{/}= m.$

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

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

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

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

Andrew Ng

Implementing Logistic Regression

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

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

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

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

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

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

$$\left.\begin{aligned} dw_1 &\mathrel{+}= x_1^{(i)} dz^{(i)} \\ dw_2 &\mathrel{+}= x_2^{(i)} dz^{(i)} \end{aligned}\right\} \; dw \mathrel{+}= x^{(i)} * dz^{(i)}$$

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

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

$db = db/m$

for iter in range(1000):

$$Z = w^T X + b$$
$$= np.dot(w.T, X) + b$$
$$A = \sigma(Z)$$
$$dZ = A - Y$$
$$dw = \frac{1}{m} X dZ^T$$
$$db = \frac{1}{m} np.sum(dZ)$$
$$w := w - \alpha dw$$
$$b := b - \alpha db$$

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Broadcasting in Python

Broadcasting example

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

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

59 cal

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

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))
```

(3,4) / (1,4)

Broadcasting example

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

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

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

General Principle

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

matrix

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

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

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

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

deeplearning.ai

Basics of Neural Network Programming

A note on python/ numpy vectors

Python Demo

Python / numpy vectors

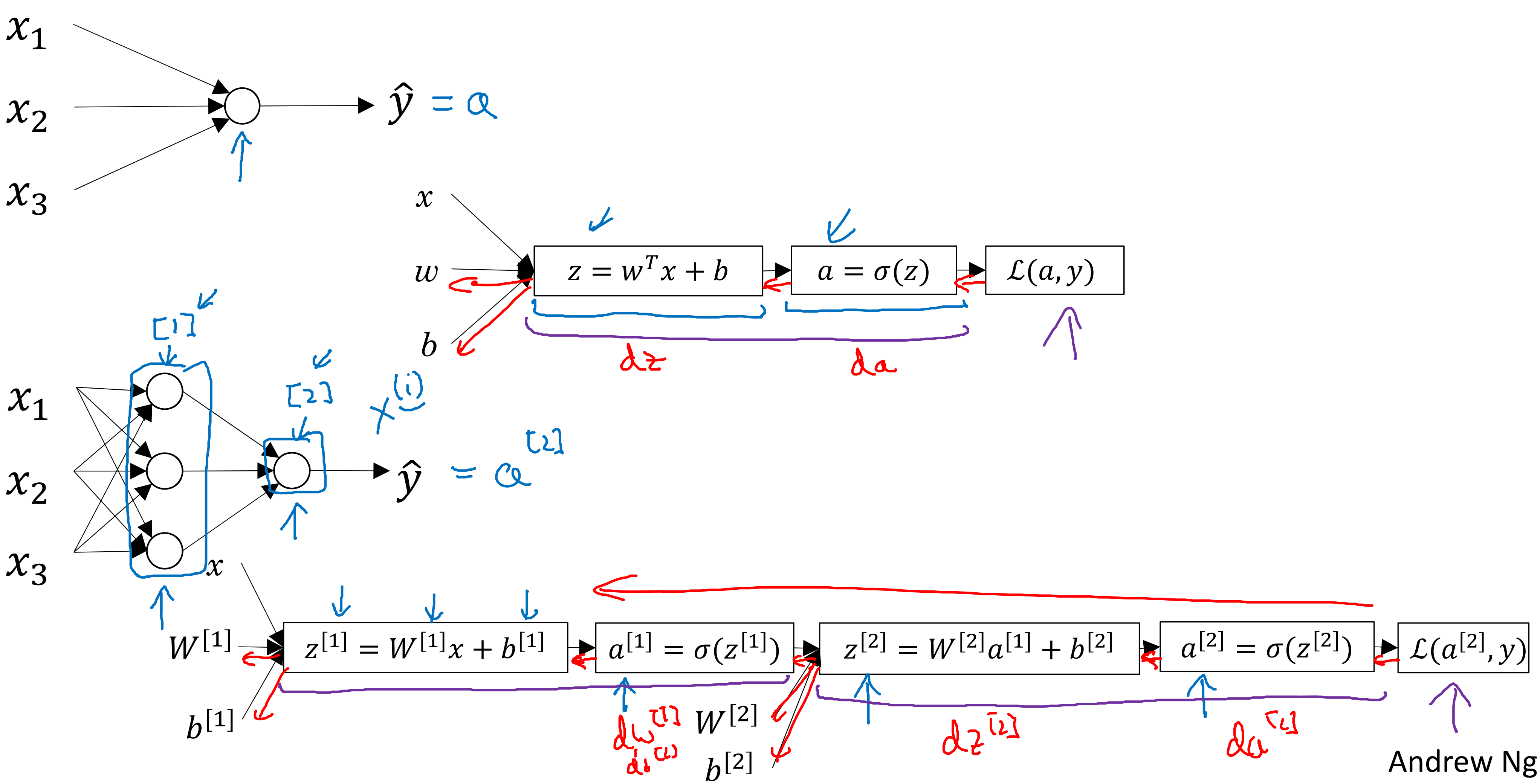
```
import numpy as np

a = np.random.randn(5)

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

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

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

deeplearning.ai

One hidden layer Neural Network

Neural Networks Overview

What is a Neural Network?



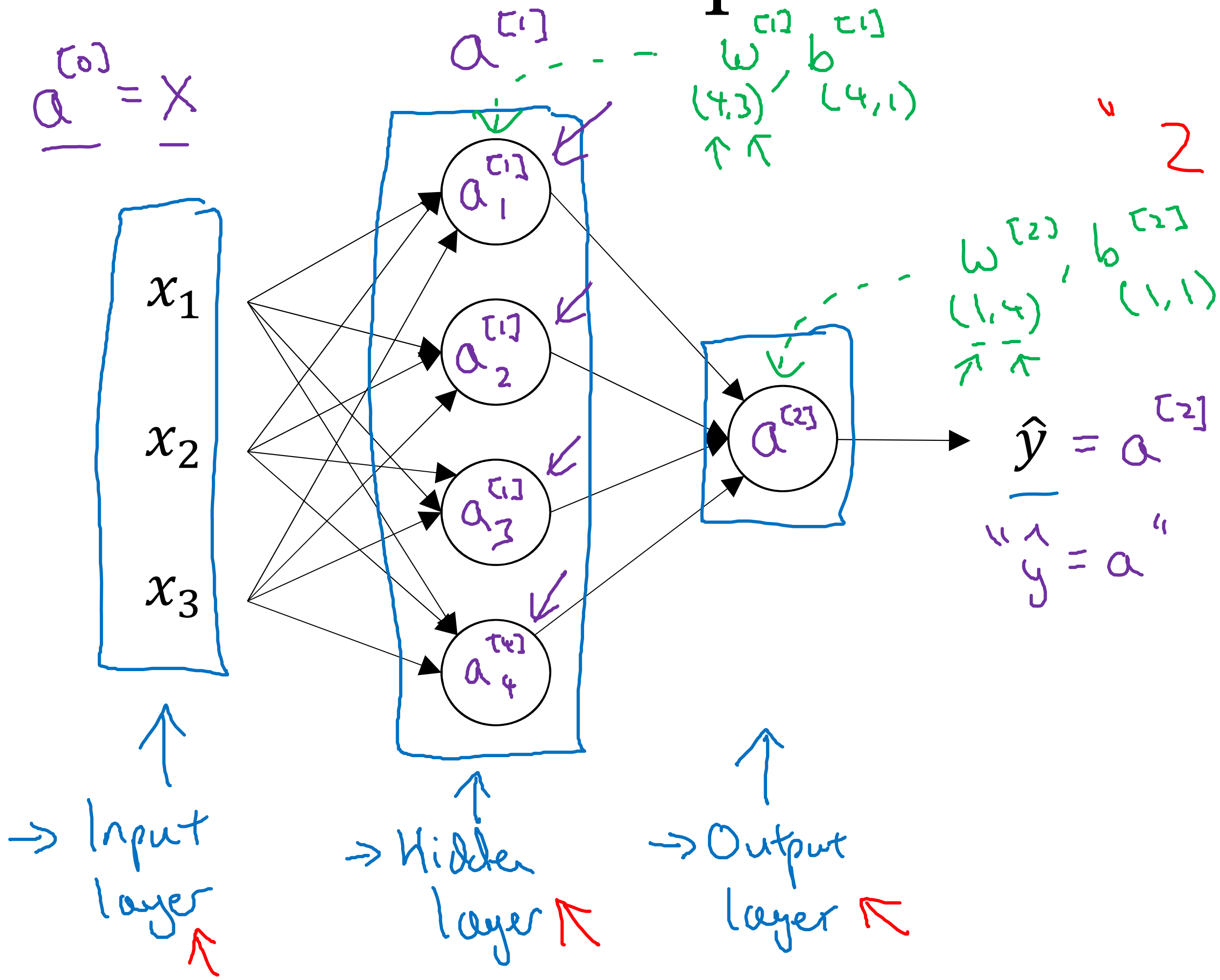
Andrew Ng

One hidden layer Neural Network

deeplearning.ai

Neural Network Representation

Neural Network Representation



"2 layer NN"

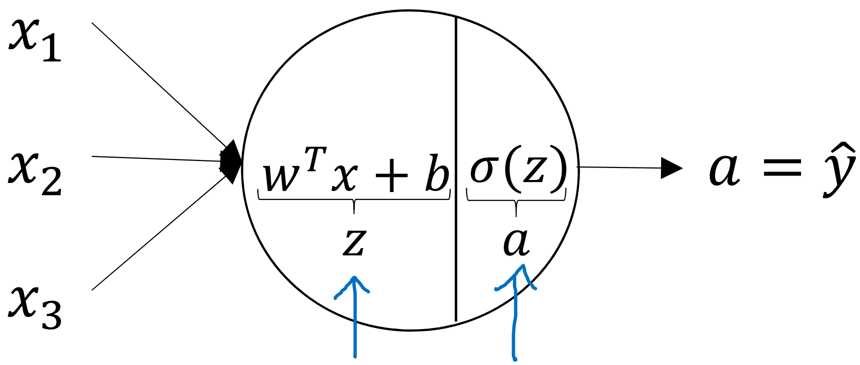
$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ \vdots \\ a_4^{[1]} \end{bmatrix}$$

deeplearning.ai

One hidden layer Neural Network

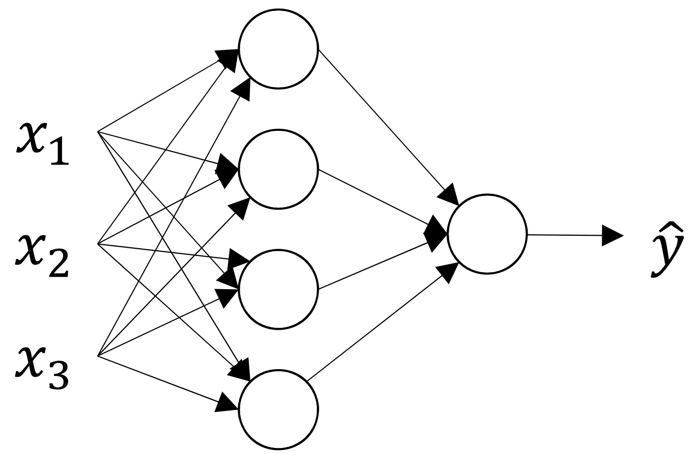
Computing a Neural Network's Output

Neural Network Representation

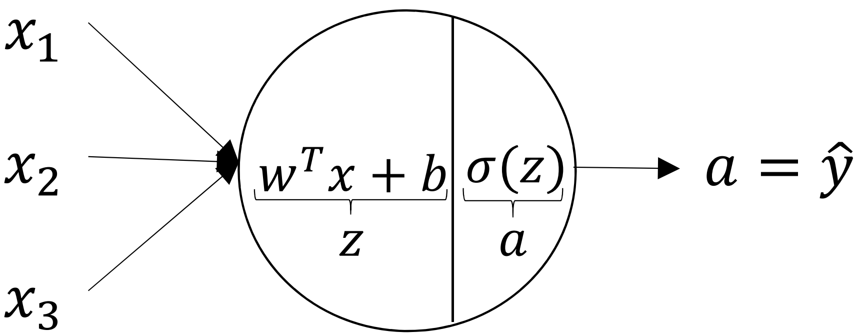


$z = w^T x + b$

$a = \sigma(z)$

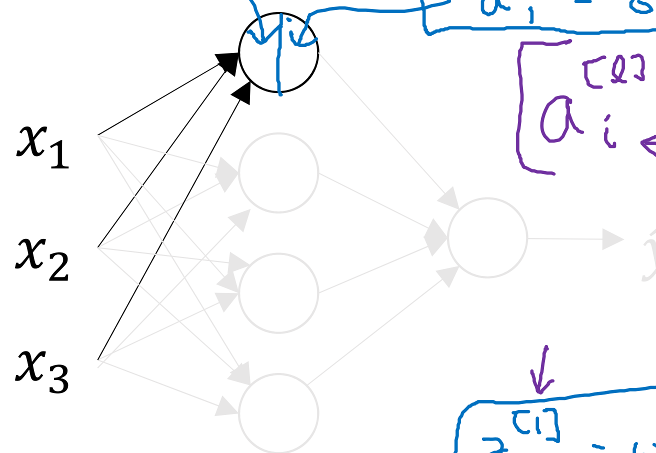


Neural Network Representation



$$z = w^Tx + b$$

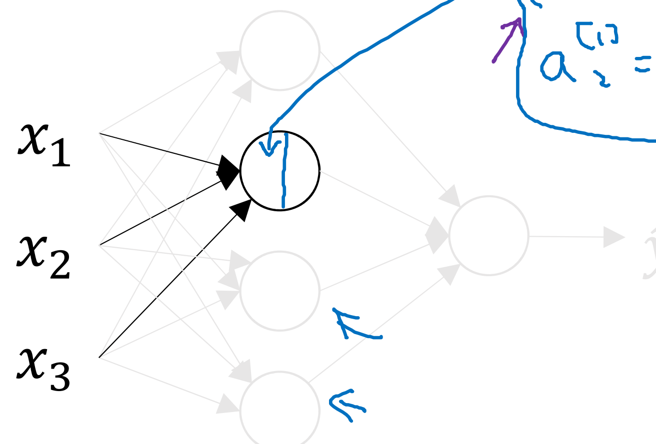
$$a = \sigma(z)$$



$$z_1^{[1]} = w_1^{[1]T}x + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$

$a_i^{[l]}$ ← layer (l), ← node in layer (i).

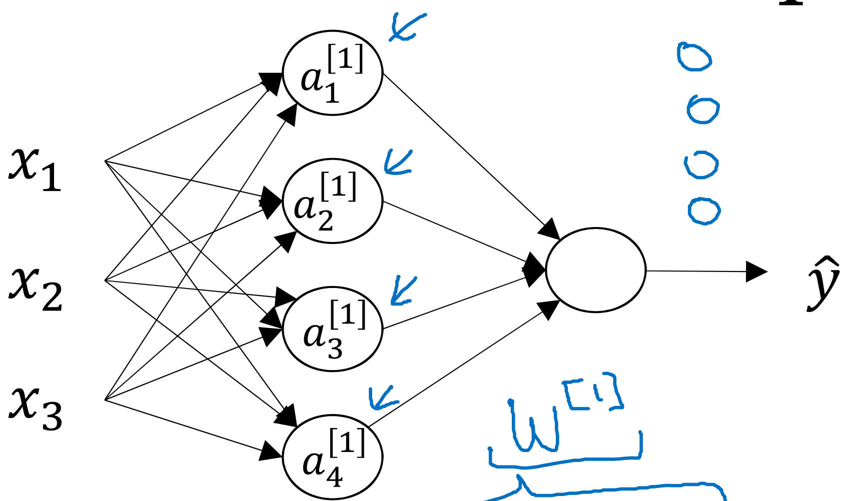


$$z_2^{[1]} = w_2^{[1]T}x + b_2^{[1]}$$

$$a_2^{[1]} = \sigma(z_2^{[1]})$$

Neural Network Representation

$(w_1^{[1]})^T x$ $\qquad a^{[1]}$



$$z_1^{[1]} = w_1^{[1]T}x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_2^{[1]T}x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_3^{[1]T}x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$

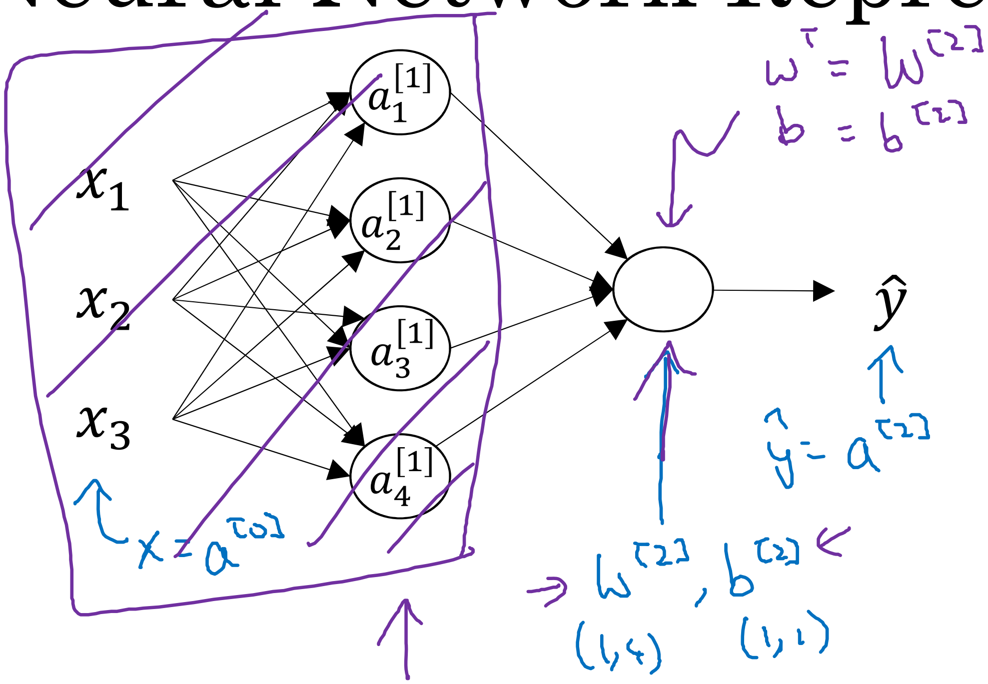
$$z_4^{[1]} = w_4^{[1]T}x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

$$z^{[1]} = \overbrace{\begin{bmatrix} — w_1^{[1]T} — \\ — w_2^{[1]T} — \\ — w_3^{[1]T} — \\ — w_4^{[1]T} — \end{bmatrix}}^{W^{[1]}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \underbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}}_{b^{[1]}} = \begin{bmatrix} w_1^{[1]T}x + b_1^{[1]} \\ w_2^{[1]T}x + b_2^{[1]} \\ w_3^{[1]T}x + b_3^{[1]} \\ w_4^{[1]T}x + b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

$(4, 3)$ $\qquad (4, 1)$

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ \vdots \\ a_4^{[1]} \end{bmatrix} = \sigma(z^{[1]})$$

Neural Network Representation learning



$z = w^T x + b$

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

Given input x:

$$\underset{(4,1)}{z^{[1]}} = \underset{(4,3)}{W^{[1]}}\,\underset{(3,1)}{\overset{a^{[0]}}{x}} + \underset{(4,1)}{b^{[1]}}$$

$$\underset{(4,1)}{a^{[1]}} = \sigma(\underset{(4,1)}{z^{[1]}})$$

$$\underset{(1,1)}{z^{[2]}} = \underset{(1,4)}{W^{[2]}}\,\underset{(4,1)}{a^{[1]}} + \underset{(1,1)}{b^{[2]}}$$

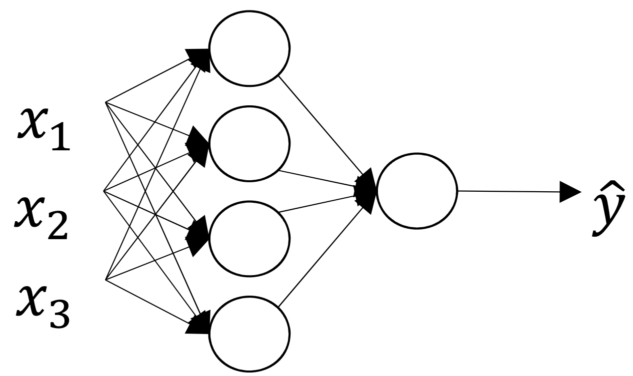
$$\underset{(1,1)}{a^{[2]}} = \sigma(\underset{(1,1)}{z^{[2]}})$$

deeplearning.ai

One hidden layer Neural Network

Vectorizing across multiple examples

Vectorizing across multiple examples



$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = \sigma(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = \sigma(z^{[2]})$$

$X \longrightarrow a^{[2]} = \hat{y}$

$x^{(1)} \longrightarrow a^{[2](1)} = \hat{y}^{(1)}$

$x^{(2)} \longrightarrow a^{[2](2)} \quad \hat{y}^{(2)}$

$\vdots$

$x^{(m)} \longrightarrow a^{[2](m)} \quad \hat{y}^{(m)}$

$a^{[2](i)}$ ← example i

← layer 2

for $i = 1$ to m,

$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$
$$a^{[1](i)} = \sigma(z^{[1](i)})$$
$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$
$$a^{[2](i)} = \sigma(z^{[2](i)})$$

Vectorizing across multiple examples

```
for i = 1 to m:
```

$$\boxed{z^{[1](i)}} = W^{[1]}x^{(i)} + b^{[1]}$$

$$\boxed{a^{[1](i)}} = \sigma(z^{[1](i)})$$

$$\boxed{z^{[2](i)}} = W^{[2]}a^{[1](i)} + b^{[2]}$$

$$\boxed{a^{[2](i)}} = \sigma(z^{[2](i)})$$

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$A^{[1]} = \sigma(Z^{[1]})$$

$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

$$A^{[2]} = \sigma(Z^{[2]})$$

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

(n_x, m)

training examples

hidden units

$$Z^{[1]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$$

$$A^{[1]} = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

hidden units

deeplearning.ai

One hidden layer Neural Network

Explanation for vectorized implementation

Justification for vectorized implementation

$z^{[1](1)} = W^{[1]}x^{(1)} + \cancel{b^{[1]}}$ 0, $\quad z^{[1](2)} = W^{[1]}x^{(2)} + \cancel{b^{[1]}}$ 0, $\quad z^{[1](3)} = W^{[1]}x^{(3)} + \cancel{b^{[1]}}$ 0

$$W^{[1]} = \begin{bmatrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{bmatrix} \qquad W^{[1]}x^{(1)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \qquad W^{[1]}x^{(2)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \qquad W^{[1]}x^{(3)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

$$W^{[1]} \underbrace{\begin{bmatrix} | & | & | & \\ x^{(1)} & x^{(2)} & x^{(3)} & \cdots \\ | & | & | & \end{bmatrix}}_{X} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} = \begin{bmatrix} | & | & | & \\ z^{[1](1)} & z^{[1](2)} & z^{[1](3)} & \cdots \\ | & | & | & \end{bmatrix} = Z^{[1]}$$

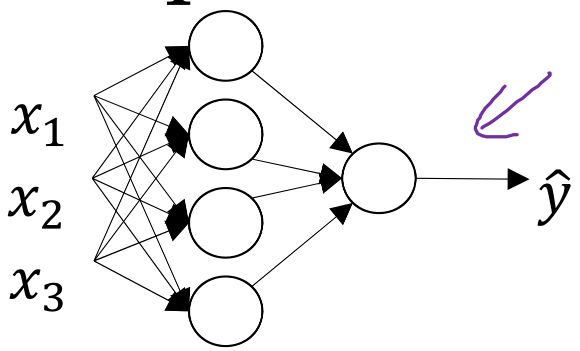
$+b^{[1]} \quad +b^{[1]} \quad +b^{[1]}$

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$W^{[1]}x^{(i)} = z^{[1](i)}$$

Andrew Ng

Recap of vectorizing across multiple examples



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

$$A^{[1]} = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

```
for i = 1 to m
```

$$\boxed{z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}}$$
$$a^{[1](i)} = \sigma(z^{[1](i)})$$
$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$
$$a^{[2](i)} = \sigma(z^{[2](i)})$$

$A^{[0]}$ $X = a^{[0]}$ $x^{(i)} = a^{[0](i)}$

$$Z^{[1]} = W^{[1]}X + b^{[1]} \leftarrow W^{[1]}A^{[0]} + b^{[1]}$$
$$A^{[1]} = \sigma(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = \sigma(Z^{[2]})$$

Andrew Ng

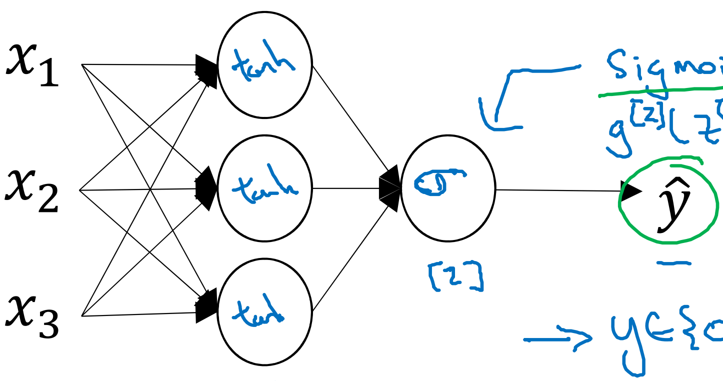
deeplearning.ai

One hidden layer Neural Network

Activation functions

Activation functions

$g^{[1]}(z^{[1]}) = \tanh(z^{[1]})$



Sigmoid

$g^{[2]}(z^{[2]}) = \sigma(z^{[2]})$

$\rightarrow y \in \{0,1\}$

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

-1 1

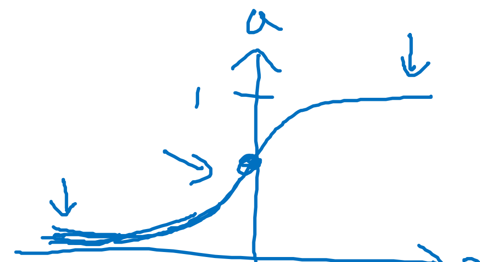
Given x:

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

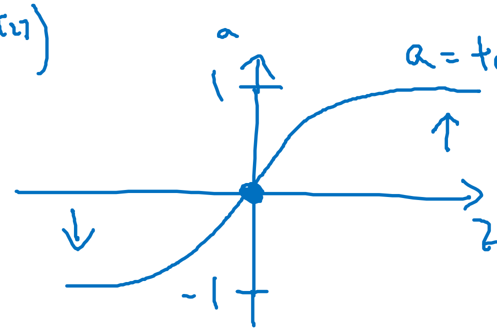
$$\rightarrow a^{[1]} = \cancel{\sigma(z^{[1]})}\ g^{[1]}(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

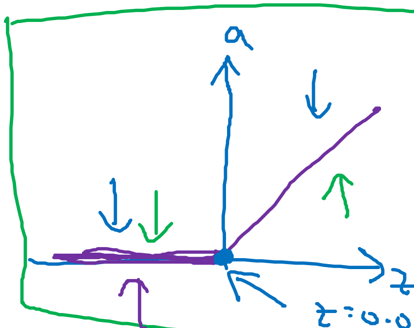
$$\rightarrow a^{[2]} = \cancel{\sigma(z^{[2]})}\ g^{[2]}(z^{[2]})$$



$a = \frac{1}{1+e^{-z}}$



$a = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$

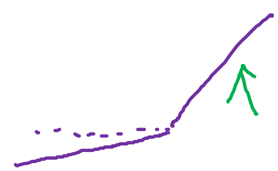


$z > 0$

$a = \max(0, z)$

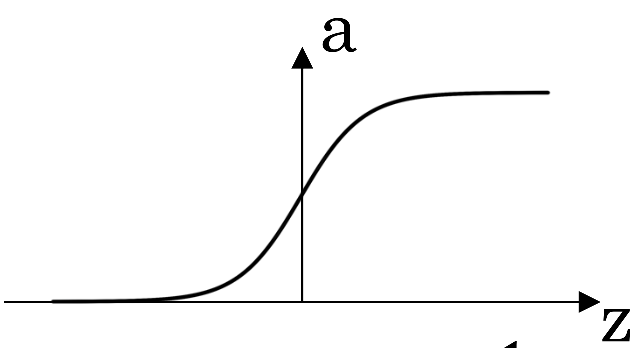
ReLU

Rectified Linear Unit

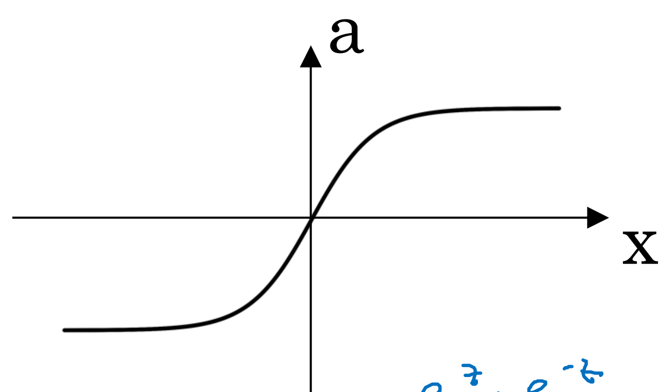


Leaky ReLU

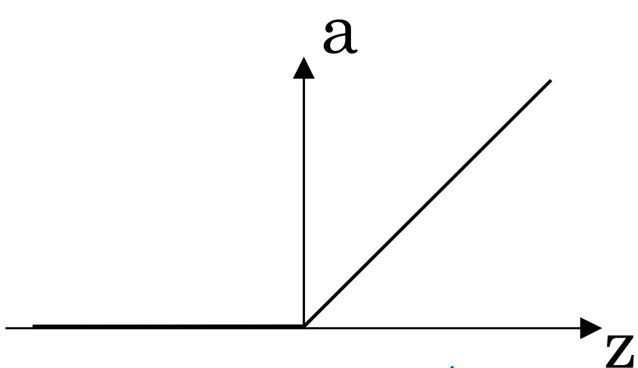
Pros and cons of activation functions



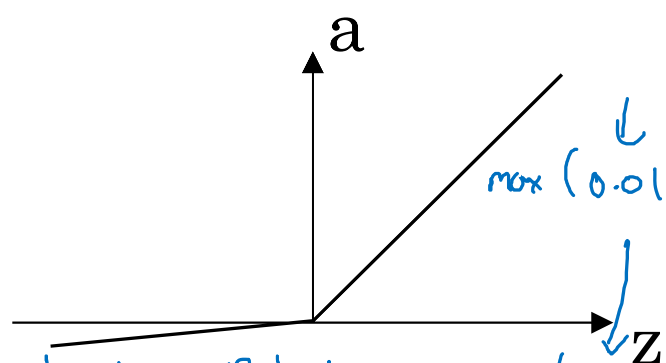
sigmoid: $a = \dfrac{1}{1+e^{-z}}$



tanh: $a = \dfrac{e^{z}-e^{-z}}{e^{z}+e^{z}}$



ReLU $a = \max(0, z)$



$\max(0.01z, z)$

Leaky ReLU $a = \max(0.01z, z)$

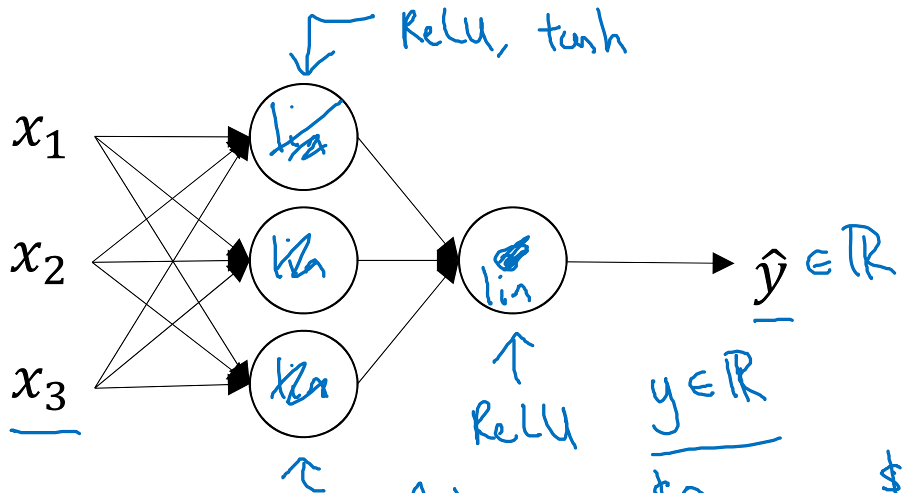
Andrew Ng

deeplearning.ai

One hidden layer Neural Network

Why do you need non-linear activation functions?

Activation function



$y \in \mathbb{R}$

\$0 ..... \$1,000,000s

$\hat{y} \geq 0$.

Given x:

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[1]} = \cancel{g^{[1]}(z^{[1]})}\ z^{[1]}$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = \cancel{g^{[2]}(z^{[2]})}\ z^{[2]}$$

$g(z) = z$

"linear activation function"

$$a^{[1]} = z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[2]} = z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = W^{[2]}\underbrace{(W^{[1]}x + b^{[1]})}_{a^{[1]}} + b^{[2]}$$

$$= \underbrace{(W^{[2]}W^{[1]})}_{W'}x + \underbrace{(W^{[2]}b^{[1]} + b^{[2]})}_{b'}$$

$$= W'x + b'$$

$g(z) = z$

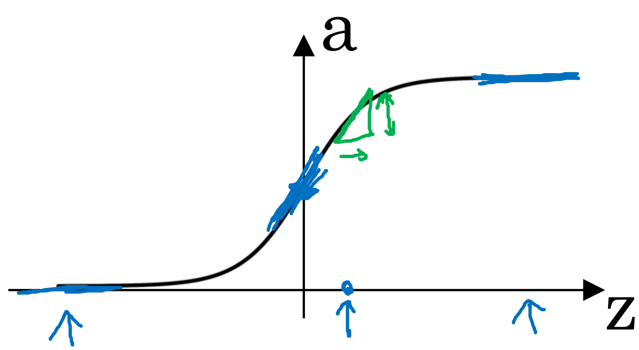
Andrew Ng

deeplearning.ai

One hidden layer Neural Network

Derivatives of activation functions

Sigmoid activation function



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

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

$$\boxed{g'(z)} = \boxed{\frac{d}{dz} g(z)} = \text{slope of } g(x) \text{ at } z$$

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

$$= \underbrace{g(z)\left(1-g(z)\right)}$$

$$= \boxed{a\,(1-a)}$$

$$g'(z) = a(1-a)$$

$z = 10$. $\quad g(z) \approx 1$

$$\frac{d}{dz} g(z) \approx 1\,(1-1) \approx 0$$

$z = -10 \quad g(z) \approx 0$

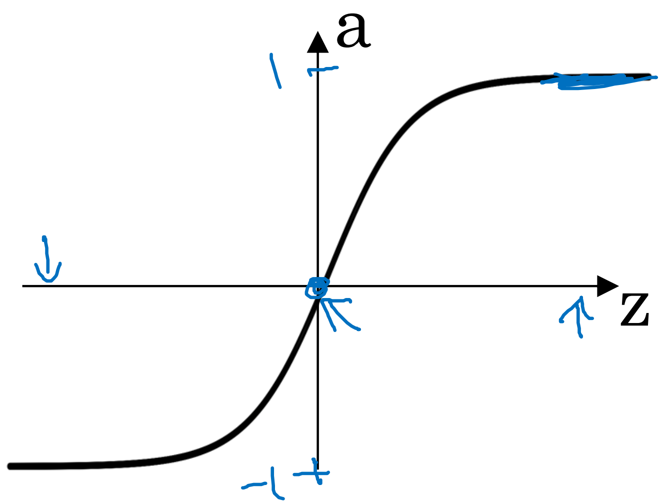
$$\frac{d}{dz} g(z) \approx 0 \cdot (1-0) \approx 0$$

$z = 0 \quad g(z) = \frac{1}{2}$

$$\frac{d}{dz} g(z) = \frac{1}{2}\left(1 - \frac{1}{2}\right) = \frac{1}{4}$$

Andrew Ng

Tanh activation function



$$g(z) = \tanh(z)$$
$$= \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$z = 10$ $\tanh(z) \approx 1$
$g'(z) \approx 0$

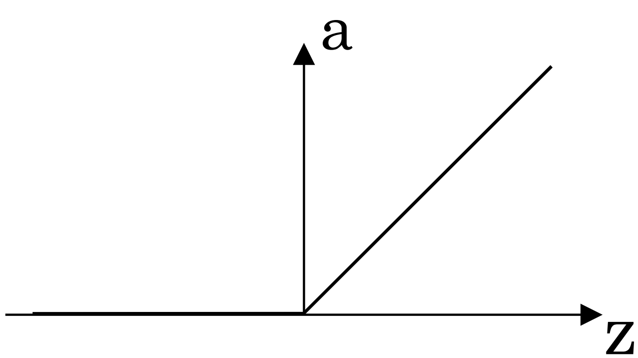
$z = -10$ $\tanh(z) \approx -1$
$g'(z) \approx 0$

$z = 0$ $\tanh(z) = 0$
$g'(z) = 1$

$$g'(z) = \frac{d}{dz} g(z) = \text{slope of } g(z) \text{ at } z$$
$$= 1 - (\tanh(z))^2$$

$a = g(z)$, $\quad g'(z) = 1 - a^2$

ReLU and Leaky ReLU

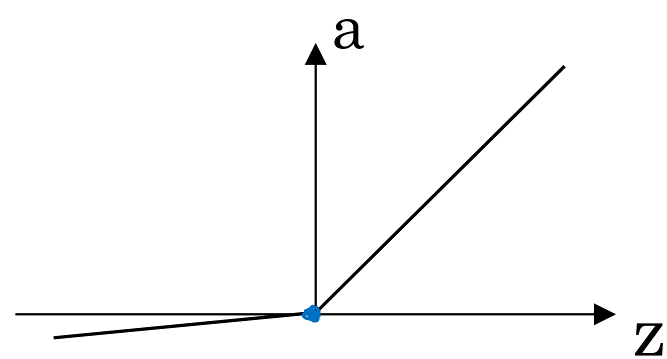


ReLU

$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \\ \cancel{\text{undef} \quad \text{if } z = 0} \end{cases}$$

$$z = 0.00000 \ldots 0$$



Leaky ReLU

$$g(z) = \max(0.01z, z)$$

$$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

Andrew Ng

deeplearning.ai

One hidden layer Neural Network

Gradient descent for neural networks

Gradient descent for neural networks

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}$ $\qquad n_x = n^{[0]}, n^{[1]}, \underline{n^{[2]} = 1}$

$(n^{[1]}, n^{[0]})\quad (n^{[1]}, 1)\quad (n^{[2]}, n^{[1]})\quad (n^{[2]}, 1)$

Cost function: $J(W^{[1]}, b^{[1]}, \underline{W^{[2]}}, \underline{b^{[2]}}) = \frac{1}{m} \sum_{i=1}^{n} \mathcal{L}(\hat{y}, y)$

$\hat{y} \leftarrow a^{[2]}$

Gradient descent:

→ Repeat {

→ Compute predicts $(\hat{y}^{(i)}, i = 1 \ldots, m)$

$\underline{dW^{[1]}} = \frac{\partial J}{\partial W^{[1]}}, \quad \underline{db^{[1]}} = \frac{\partial J}{\partial b^{[1]}}, \ldots$

$W^{[1]} := W^{[1]} - \alpha \, dW^{[1]}$

$b^{[1]} := b^{[1]} - \alpha \, db^{[1]}$

$W^{[2]} := \ldots \qquad b^{[2]} := \ldots$

}

Andrew Ng

Formulas for computing derivatives

Forward propagation:

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$A^{[1]} = g^{[1]}(Z^{[1]}) \leftarrow$$

$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

$$A^{[2]} = g^{[2]}(Z^{[2]}) = \sigma(Z^{[2]})$$

Back propagation:

$$dZ^{[2]} = A^{[2]} - Y \quad \swarrow \qquad Y = [y^{(1)} \; y^{(2)} \; \dots, y^{(m)}]$$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$$

$$(n^{[2]},) \leftarrow$$

$$(n^{[2]}, 1) \leftarrow$$

$$db^{[2]} = \frac{1}{m} \underline{np.sum}(dZ^{[2]}, \underline{axis=1}, \underline{keepdims=True})$$

$$dZ^{[1]} = \underbrace{W^{[2]T} dZ^{[2]}}_{(n^{[1]}, m)} * \underbrace{g^{[1]'}(Z^{[1]})}_{(n^{[1]}, m)}$$

$\uparrow$ element-wise product

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$$

$$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis=1, \underline{keepdims=True})$$

$$\underline{(n^{[1]}, 1)} \qquad (n^{[1]},)$$

reshape $\uparrow$

Andrew Ng

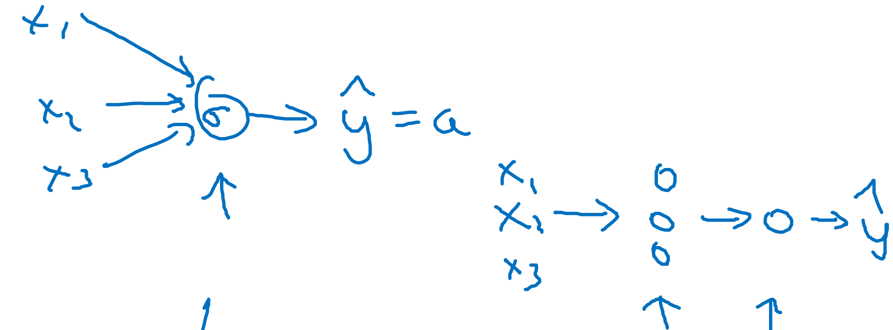
deeplearning.ai

One hidden layer Neural Network

Backpropagation intuition (Optional)

Computing gradients

Logistic regression



$$x,\ w,\ b \longrightarrow \boxed{z = w^T x + b} \longrightarrow \boxed{a = \sigma(z)} \longrightarrow \boxed{\mathcal{L}(a, y)}$$

$dz = a - y$

$dw = dz \cdot x$

$db = dz$

$$\boxed{dz = da \cdot g'(z)}$$

$g(z) = \sigma(z)$

$$\boxed{\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial a} \cdot \boxed{\frac{da}{dz}}}$$

"dz" = "da"

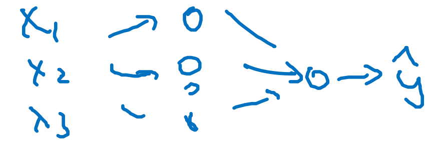
$$\frac{d}{dz} g(z) = g'(z)$$

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

$$= -\frac{y}{a} + \frac{1-y}{1-a}$$

Andrew Ng

Neural network gradients



$n_x = n^{[0]}$ $\quad n^{[1]}$ $\quad n^{[2]} = 1$

$x,\ W^{[1]},\ b^{[1]} \rightarrow \boxed{z^{[1]} = W^{[1]}x + b^{[1]}} \rightarrow \boxed{a^{[1]} = \sigma(z^{[1]})} \rightarrow \boxed{z^{[2]} = W^{[2]}x + b^{[2]}} \rightarrow \boxed{a^{[2]} = \sigma(z^{[2]})} \rightarrow \boxed{\mathcal{L}(a^{[2]}, y)}$

$W^{[2]}$, $b^{[2]}$ → $z^{[2]}$

$dW^{[1]}$ $\quad db^{[1]}$ $\quad dW^{[2]}$ $\quad db^{[2]}$

$dz^{[1]} = W^{[2]T} dz^{[2]}$ $\quad da^{[1]} = \ldots$ $\quad dz^{[2]} = a^{[2]} - y$ $\quad da^{[2]}$

$* \ g^{[1]'}(z^{[1]})$

element wise product

$dW^{[2]} = dz^{[2]} a^{[1]T}$

$db^{[2]} = dz^{[2]}$

"$dw = dz \cdot x$"

$w^{[1]}: [\ \underline{\qquad}\ ]$

$W^{[2]} \quad (n^{[2]}, n^{[1]})$

$z^{[2]}, dz^{[2]} \quad (n^{[2]}, 1) \ - (1,1)$

$z^{[1]}, dz^{[1]} \quad (n^{[1]}, 1)$

$dz^{[1]} = \boxed{W^{[2]T} dz^{[2]}} * \boxed{g^{[1]'}(z^{[1]})}$

$(n^{[1]}, 1) = (n^{[1]}, n^{[2]})\ (n^{[2]}, 1) * (n^{[1]}, 1)$

foo $\quad W^{[1]}$

dfoo $\quad dW^{[2]}$

$dW^{[1]} = dz^{[1]} \cdot x^T$

$db^{[1]} = dz^{[1]}$

$a^{[0]T}$

Andrew Ng

Summary of gradient descent

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]} a^{[1]^T}$$

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(z^{[1]})$$

$$dW^{[1]} = dz^{[1]} x^T$$

$$db^{[1]} = dz^{[1]}$$

Vectorized Implementation:

$$z^{[1]} = W^{[1]} x + b^{[1]}$$

$$a^{[1]} = g^{[1]}(z^{[1]})$$

$$Z^{[1]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$$

$$Z^{[1]} = W^{[1]} X + b^{[1]}$$

$$A^{[1]} = g^{[1]}(Z^{[1]})$$

Andrew Ng

Summary of gradient descent

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]} a^{[1]^T}$$

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(z^{[1]})$$

$(n^{[1]}, 1)$

$$dW^{[1]} = dz^{[1]} x^T$$

$$db^{[1]} = dz^{[1]}$$

$$dZ^{[2]} = A^{[2]} - Y$$

$$J(\ldots) = \frac{1}{m} \sum_{i=1}^{n} \mathcal{L}(\hat{y}, y)$$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]^T}$$

$$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$$

elementwise product

$$dZ^{[1]} = \underbrace{W^{[2]T} dZ^{[2]}}_{(n^{[1]}, m)} * \underbrace{g^{[1]\prime}(Z^{[1]})}_{(n^{[1]}, m)}$$

$(n^{[1]}, m)$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$$

$$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)$$

Andrew Ng

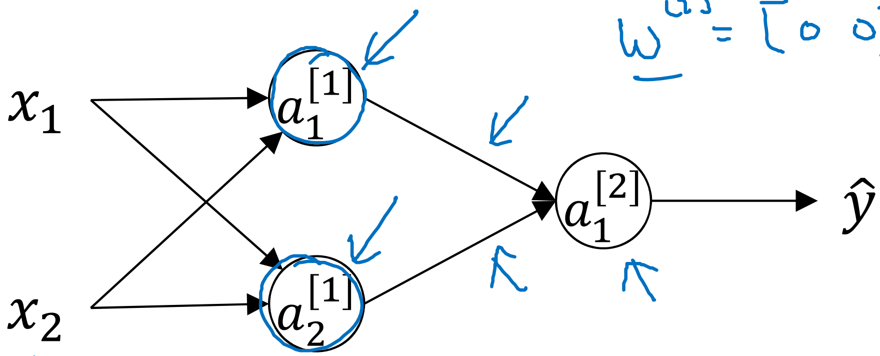
One hidden layer Neural Network

deeplearning.ai

Random Initialization

What happens if you initialize weights to zero?

$W^{[2]} = [0 \quad 0]$



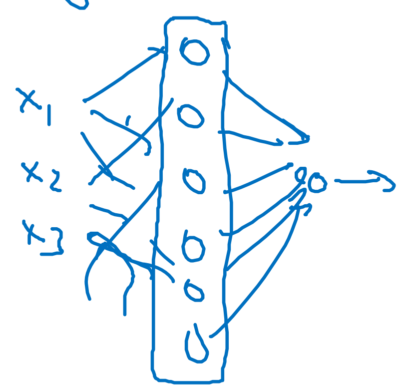
$n^{[0]} = 2$ $\qquad$ $n^{[1]} = 2$

$W^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ $\qquad$ $b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$a_1^{[1]} = a_2^{[1]}$ $\qquad$ $dz_1^{[1]} = dz_2^{[1]}$

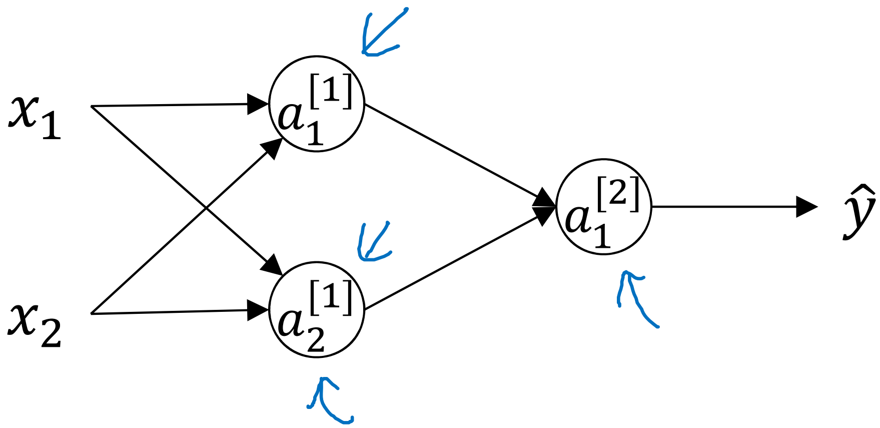
$dW = \begin{bmatrix} u & v \\ u & v \end{bmatrix}$ $\qquad$ $W^{[1]} = W^{[1]} - \alpha \, dW$

Symmetric



$W^{[1]} = \begin{bmatrix} \cdots\cdots \\ \cdots\cdots \end{bmatrix}$

Random initialization



$\rightarrow W^{[1]} = \text{np.random.randn}((2,2)) * 0.01$

100?

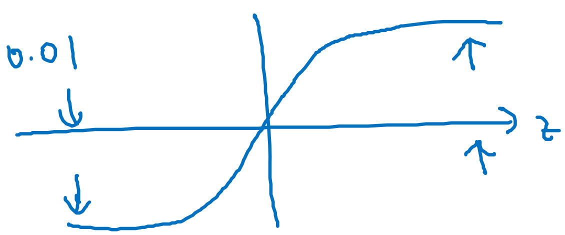
$b^{[1]} = \text{np.zeros}((2,1))$

$W^{[2]} = \text{np.random.randn}((1,2)) * 0.01$

$b^{[2]} = 0$

$\rightarrow z^{[1]} = W^{[1]} x + b^{[1]}$

$a^{[1]} = g^{[1]}(z^{[1]})$



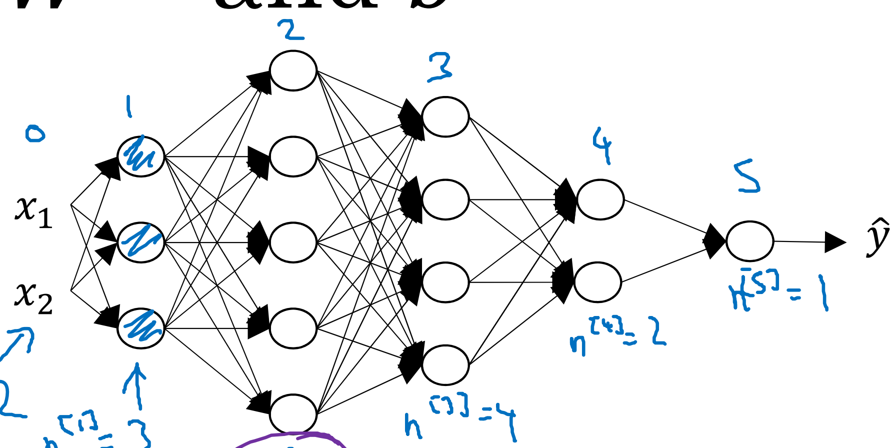
deeplearning.ai

Deep Neural Networks

Getting your matrix dimensions right

Parameters $W^{[l]}$ and $b^{[l]}$

$L = 5$



$z^{[l]} = g^{[l]}(a^{[l]})$

$a^{[l]}$

$$W^{[l]} : (n^{[l]}, n^{[l-1]})$$
$$b^{[l]} : (n^{[l]}, 1)$$
$$dW^{[l]} : (n^{[l]}, n^{[l-1]})$$
$$db^{[l]} : (n^{[l]}, 1)$$

$$z^{[1]} = \boxed{W^{[1]} \cdot x} + \boxed{b^{[1]}}$$

$(3,1) \leftarrow (3,2) \quad (2,1) \quad (3,1)$

$(n^{[1]}, 1) \quad (n^{[1]}, n^{[0]}) \quad (n^{[0]}, 1) \quad (n^{[1]}, 1)$

$$\begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot \\ \cdot \end{bmatrix}$$

$W^{[1]} : (n^{[1]}, n^{[0]})$

$W^{[2]} : (5, 3) \quad (n^{[2]}, n^{[1]})$

$$z^{[2]} = \boxed{W^{[2]} \cdot a^{[1]}} + \boxed{b^{[2]}}$$

$\rightarrow (5,1) \quad (5,3) \quad (3,1) \quad (5,1)$

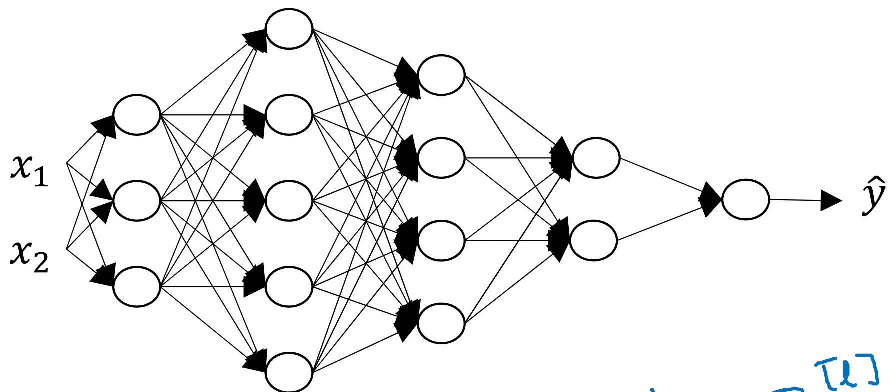
$(n^{[2]}, 1)$

$W^{[3]} : (4, 5)$

$W^{[4]} : (2, 4) \quad , \quad W^{[5]} : (1, 2)$

Andrew Ng

Vectorized implementation



$$z^{[1]} = W^{[1]} \cdot x + b^{[1]}$$

$(n^{[1]},1)$ $(n^{[1]},n^{[0]})$ $(n^{[0]},1)$ $(n^{[1]},1)$

$$\begin{bmatrix} z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \end{bmatrix}$$

$$Z^{[1]} = W^{[1]} \cdot X + b^{[1]}$$

$(n^{[1]},m)$ $(n^{[1]},n^{[0]})$ $(n^{[0]},m)$ $(n^{[1]},1)$

$(n^{[1]},m)$

$z^{[l]}, a^{[l]} : (n^{[l]},1)$

$Z^{[l]}, A^{[l]} : (n^{[l]},m)$

$l=0 \quad A^{[0]} = X = (n^{[0]},m)$

$dZ^{[l]}, dA^{[l]} : (n^{[l]},m)$

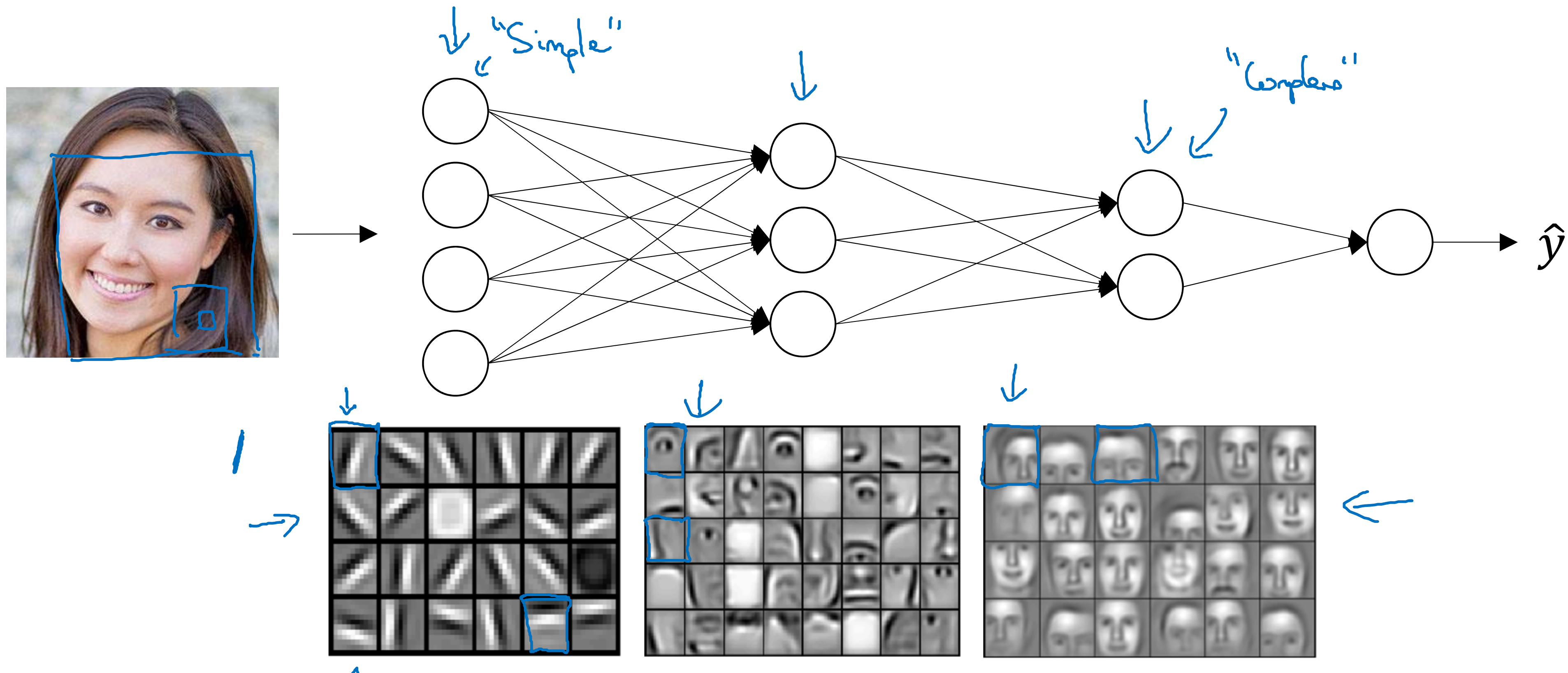
Andrew Ng

deeplearning.ai

Deep Neural Networks

Why deep representations?

Intuition about deep representation

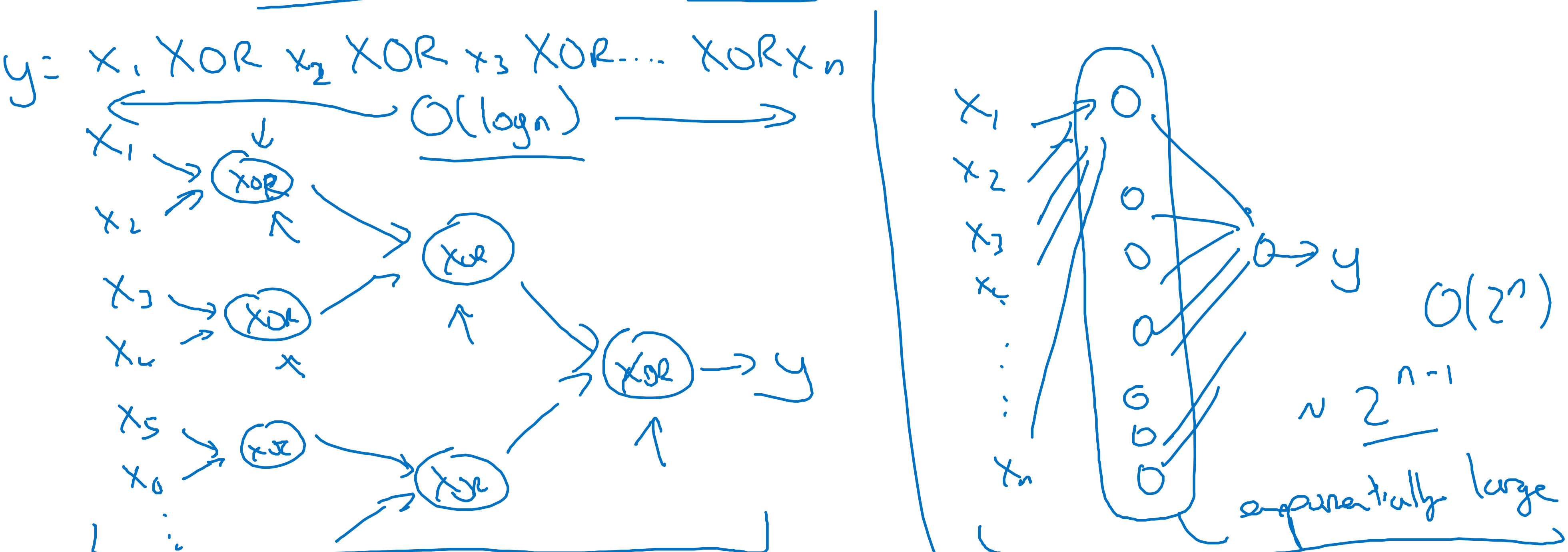


Audio → low level audio waveform → Phonemes C A T → Words → Sentence phrase

Andrew Ng

Circuit theory and deep learning

Informally: There are functions you can compute with a "small" L-layer deep neural network that shallower networks require exponentially more hidden units to compute.

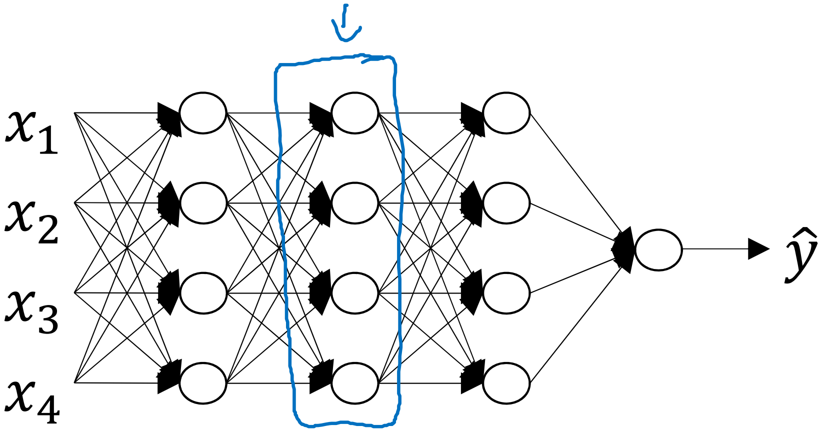


deeplearning.ai

Deep Neural Networks

Building blocks of deep neural networks

Forward and backward functions



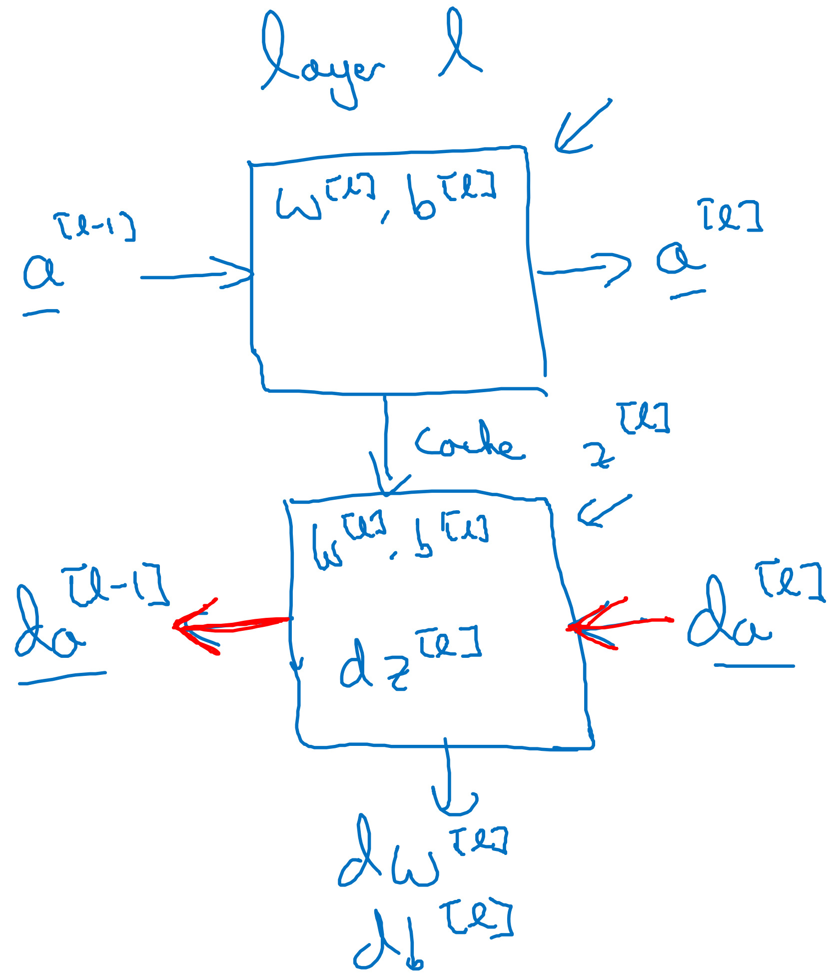
Layer l: $W^{[l]}, b^{[l]}$

→ Forward: Input $a^{[l-1]}$, output $a^{[l]}$

$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$ cache $z^{[l]}$

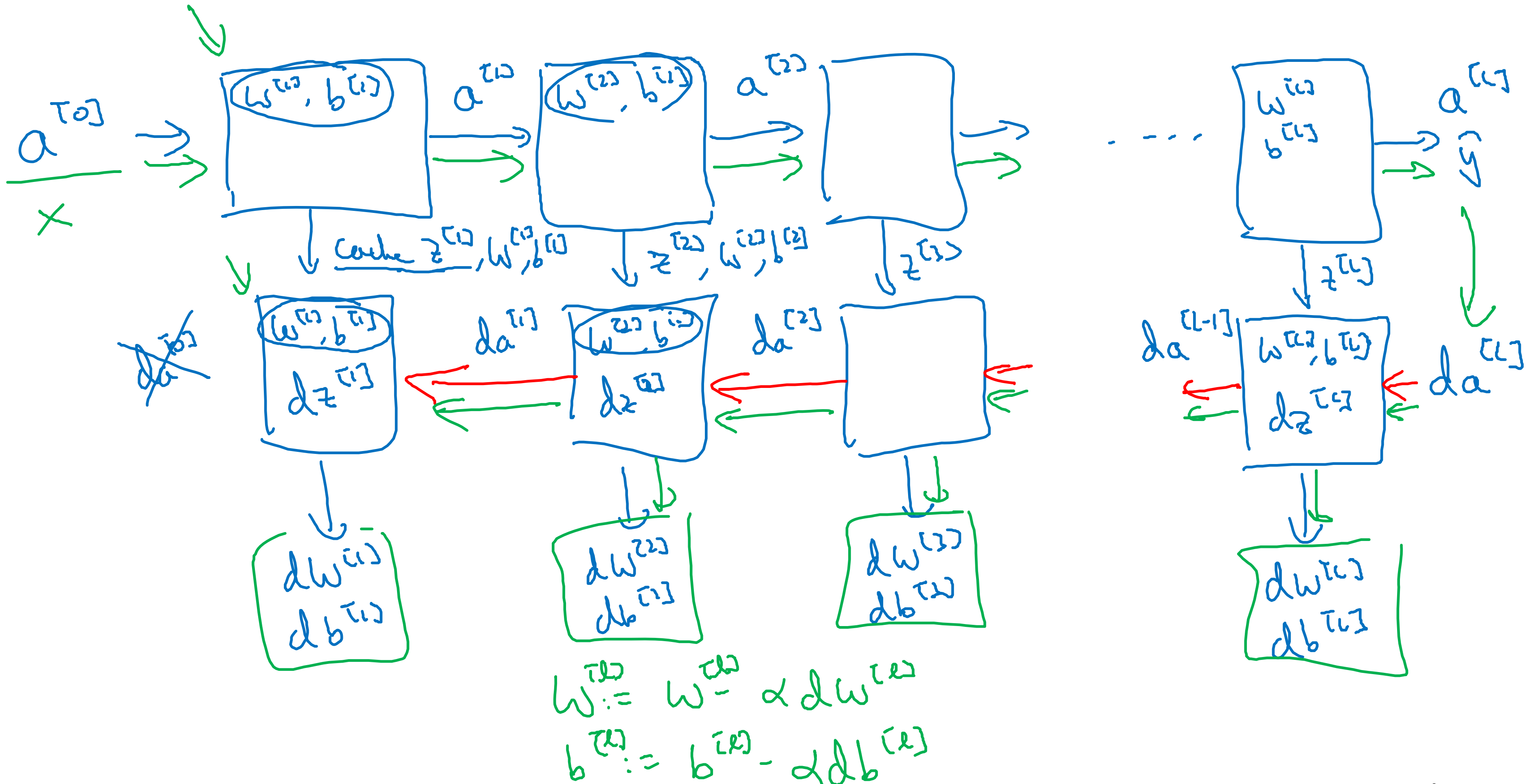
$a^{[l]} = g^{[l]}(z^{[l]})$

→ Backward: Input $da^{[l]}$, cache($z^{[l]}$) output $da^{[l-1]}$, $dW^{[l]}$, $db^{[l]}$



Andrew Ng

Forward and backward functions



Andrew Ng

deeplearning.ai

Deep Neural Networks

Forward and backward propagation

Forward propagation for layer l

- Input $a^{[l-1]}$
- Output $a^{[l]}$, cache ($z^{[l]}$)

$W^{[l]}, b^{[l]}$

$$z^{[l]} = W^{[l]} \cdot a^{[l-1]} + b^{[l]}$$

$$a^{[l]} = g^{[l]}(z^{[l]})$$

Vectorized:

$$Z^{[l]} = W^{[l]} \cdot A^{[l-1]} + b^{[l]}$$

$$A^{[l]} = g^{[l]}(Z^{[l]})$$

$a^{[0]}$

$A^{[0]}$

$X = A^{[0]} \rightarrow \square \rightarrow \square \rightarrow \square \rightarrow$

Andrew Ng

Backward propagation for layer l

- Input $da^{[l]}$
- Output $\boxed{da^{[l-1]}}$, $dW^{[l]}$, $db^{[l]}$

$$dz^{[l]} = da^{[l]} * g^{[l]\prime}(z^{[l]})$$

$$dW^{[l]} = dz^{[l]} \cdot a^{[l-1]}$$

$$db^{[l]} = dz^{[l]}$$

$$\boxed{da^{[l-1]}} = W^{[l]T} \cdot dz^{[l]}$$

$$dz^{[l]} = W^{[l+1]T} dz^{[l+1]} * g^{[l]\prime}(z^{[l]})$$

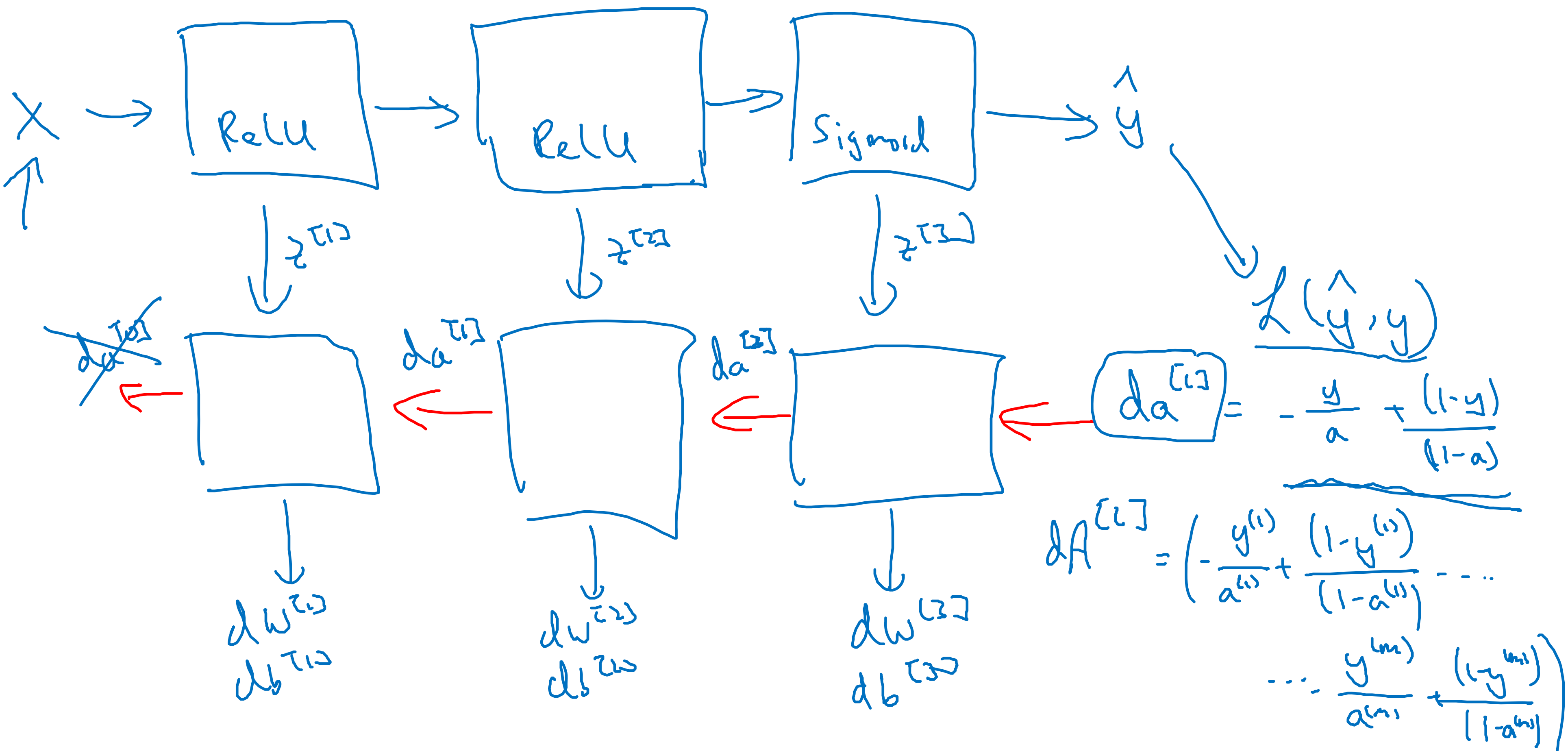
$$dZ^{[l]} = dA^{[l]} * g^{[l]\prime}(Z^{[l]})$$

$$dW^{[l]} = \frac{1}{m} dZ^{[l]} \cdot A^{[l-1]T}$$

$$db^{[l]} = \frac{1}{m} \text{np.sum}(dZ^{[l]}, \text{axis}=1, \text{keepdims}=\text{True})$$

$$\boxed{dA^{[l-1]}} = W^{[l]T} \cdot dZ^{[l]}$$

Summary



Andrew Ng

deeplearning.ai

Deep Neural Networks

Parameters vs Hyperparameters

What are hyperparameters?

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}$ ...

Hyperparameters: learning rate α

#iterations

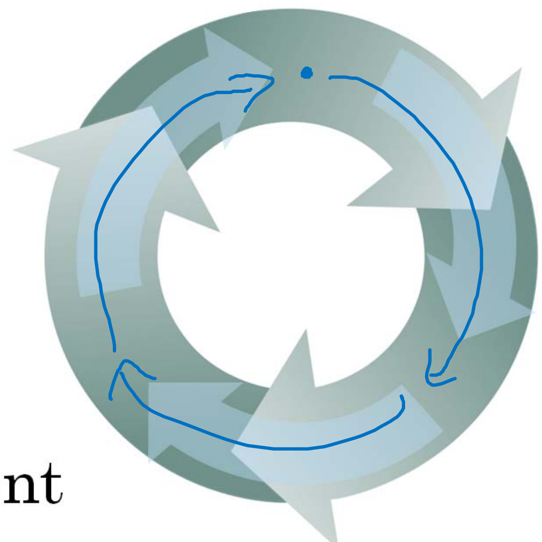
#hidden layers L

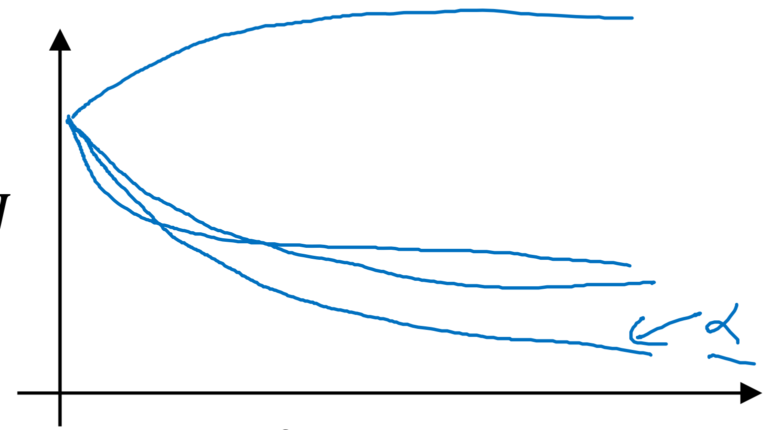
#hidden units $n^{[1]}, n^{[2]}, \ldots$

choice of activation function

Later: Momentum, mini-batch size, regularizations, ...

Applied deep learning is a very empirical process





Vision, Speech, NLP, Ad, Search, Recommendation.

deeplearning.ai

Deep Neural Networks

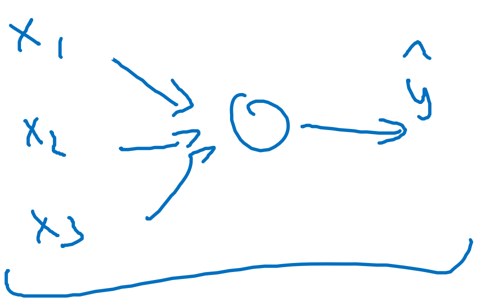
What does this have to do with the brain?

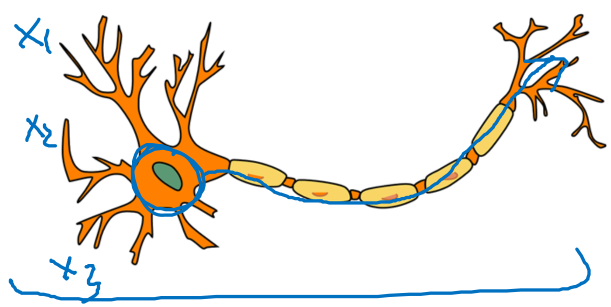
Forward and backward propagation

$$
\begin{aligned}
Z^{[1]} &= W^{[1]}X + b^{[1]} \\
A^{[1]} &= g^{[1]}(Z^{[1]}) \\
Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\
A^{[2]} &= g^{[2]}(Z^{[2]}) \\
&\vdots \\
A^{[L]} &= g^{[L]}\left(Z^{[L]}\right) = \hat{Y}
\end{aligned}
$$

$$
\begin{aligned}
dZ^{[L]} &= A^{[L]} - Y \\
dW^{[L]} &= \frac{1}{m} dZ^{[L]} A^{[L]^T} \\
db^{[L]} &= \frac{1}{m} np.\mathrm{sum}(\mathrm{d}Z^{[L]}, axis = 1, keepdims = True) \\
dZ^{[L-1]} &= dW^{[L]^T} dZ^{[L]} g'^{[L]}(Z^{[L-1]}) \\
&\vdots \\
dZ^{[1]} &= dW^{[L]^T} dZ^{[2]} g'^{[1]}(Z^{[1]}) \\
dW^{[1]} &= \frac{1}{m} dZ^{[1]} A^{[1]^T} \\
db^{[1]} &= \frac{1}{m} np.\mathrm{sum}(\mathrm{d}Z^{[1]}, axis = 1, keepdims = True)
\end{aligned}
$$

"It's like the brain"





$X \longrightarrow y$

Andrew Ng