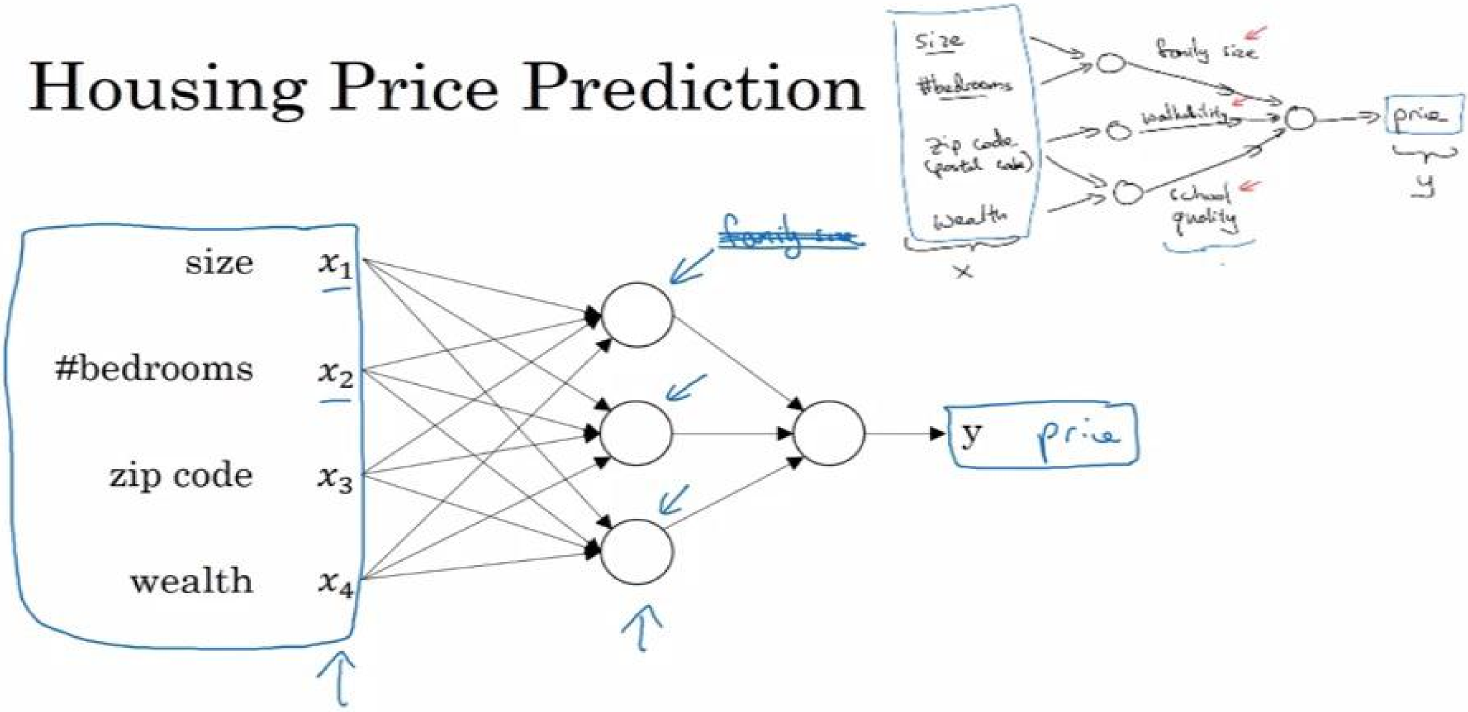
What you'll learn

Courses in this sequence (Specialization):
1. Neural Networks and Deep Learning
2. Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization
3. Structuring your Machine Learning project train/dev/test end-to-end
4. Convolutional Neural Networks CNN
5. Natural Language Processing: Building sequence models
RNN, LSTM

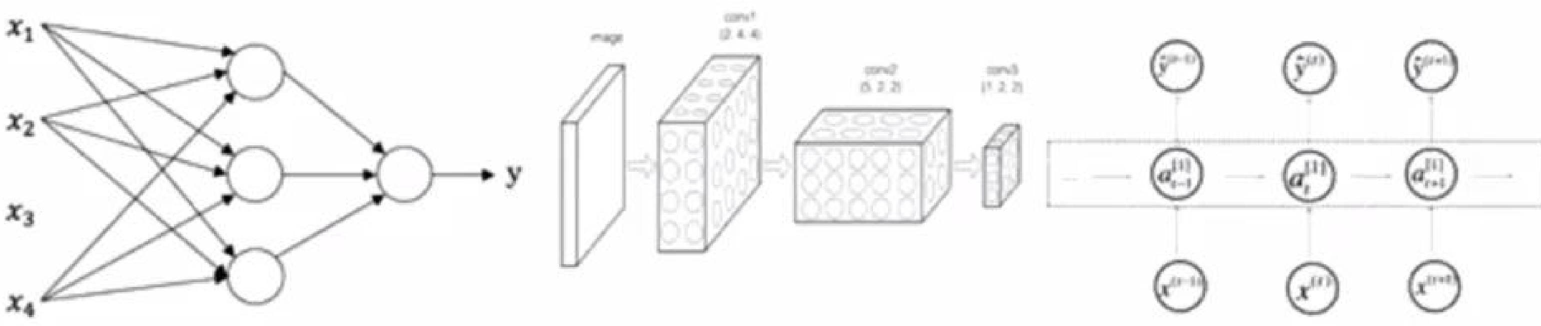
Housing Price Prediction



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

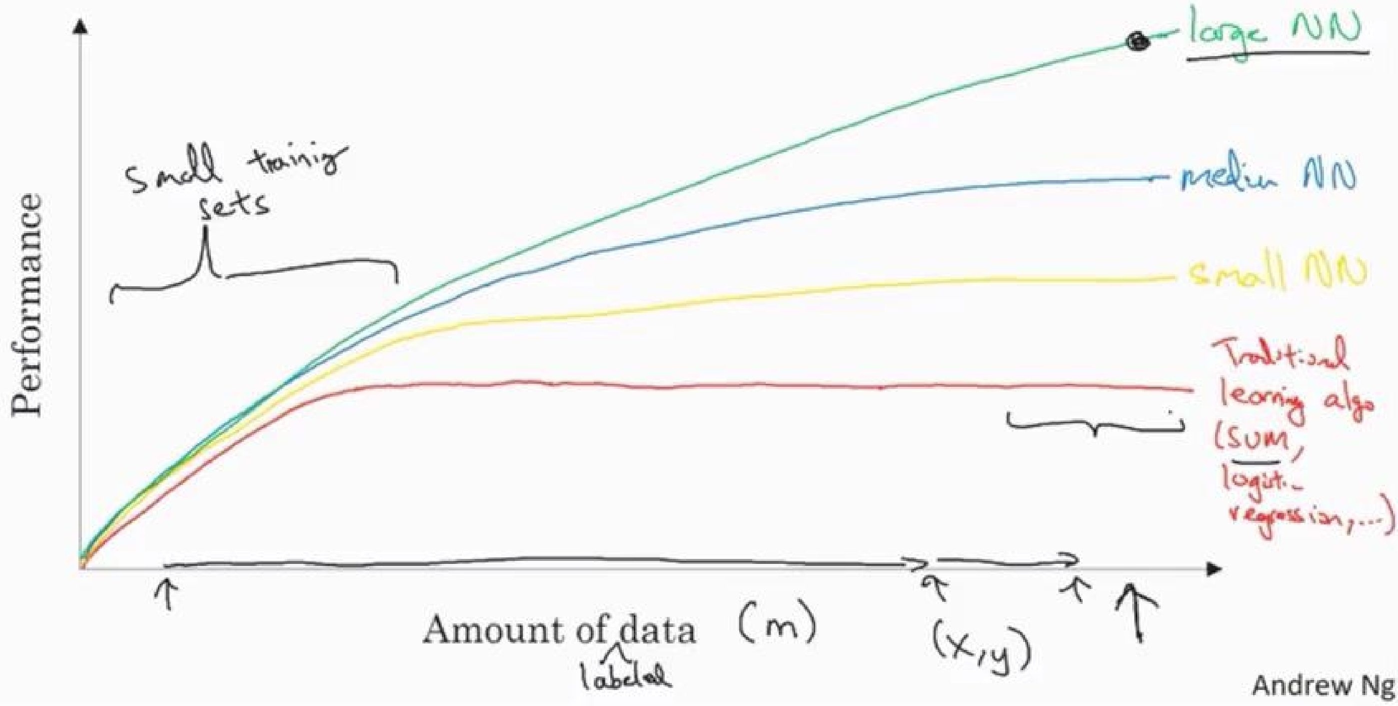
Audio

Image

Four scores and seven years ago...

Text

Scale drives deep learning progress



Andrew Ng

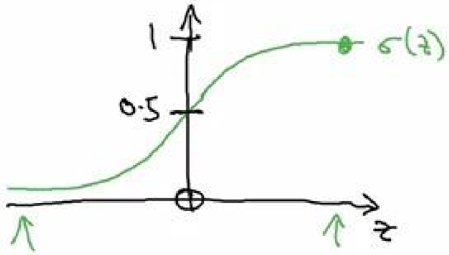
Logistic Regression

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

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

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

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



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

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

If z large negative number

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

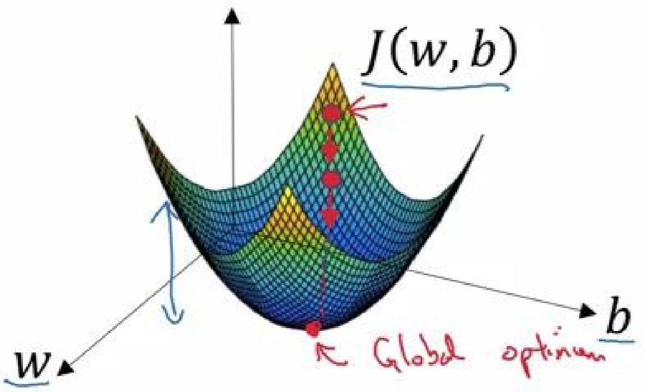
Andrew Ng

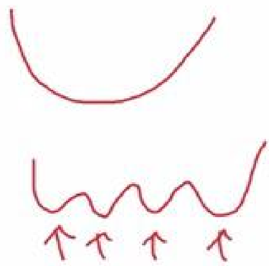
Gradient Descent

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

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

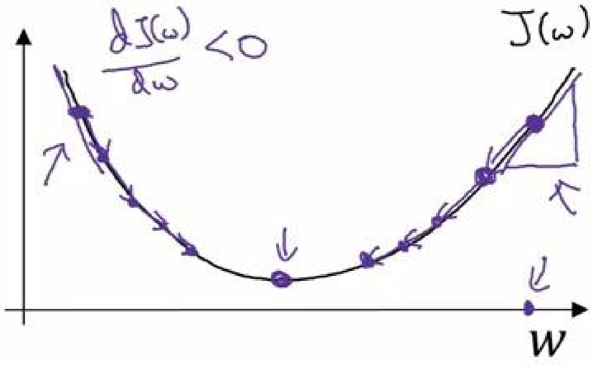
Want to find w, b that minimize $J(w,b)$





Andrew Ng

Gradient Descent



learning rate

Repeat {

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

}

"dw"

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

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

$J(w,b)$

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

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

∂ "partial derivative"

d ← J

Andrew Ng

More derivative examples

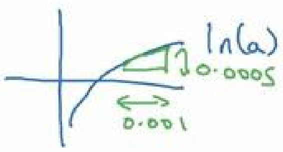
$f(a) = a^2$ $\qquad \frac{d}{da} f(a) = \underbrace{2a}_{4}$ $\qquad a = 2 \quad f(a) = 4$ $\qquad a = 2.001 \quad f(a) \approx 4.004$

$f(a) = a^3$ $\qquad \frac{d}{da} f(a) = \underbrace{3a^2}_{3 \times 2^2 = 12}$ $\qquad a = 2 \quad f(a) = 8$ $\qquad a = 2.001 \quad f(a) \approx 8.012$

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

$\ln(a)$

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



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

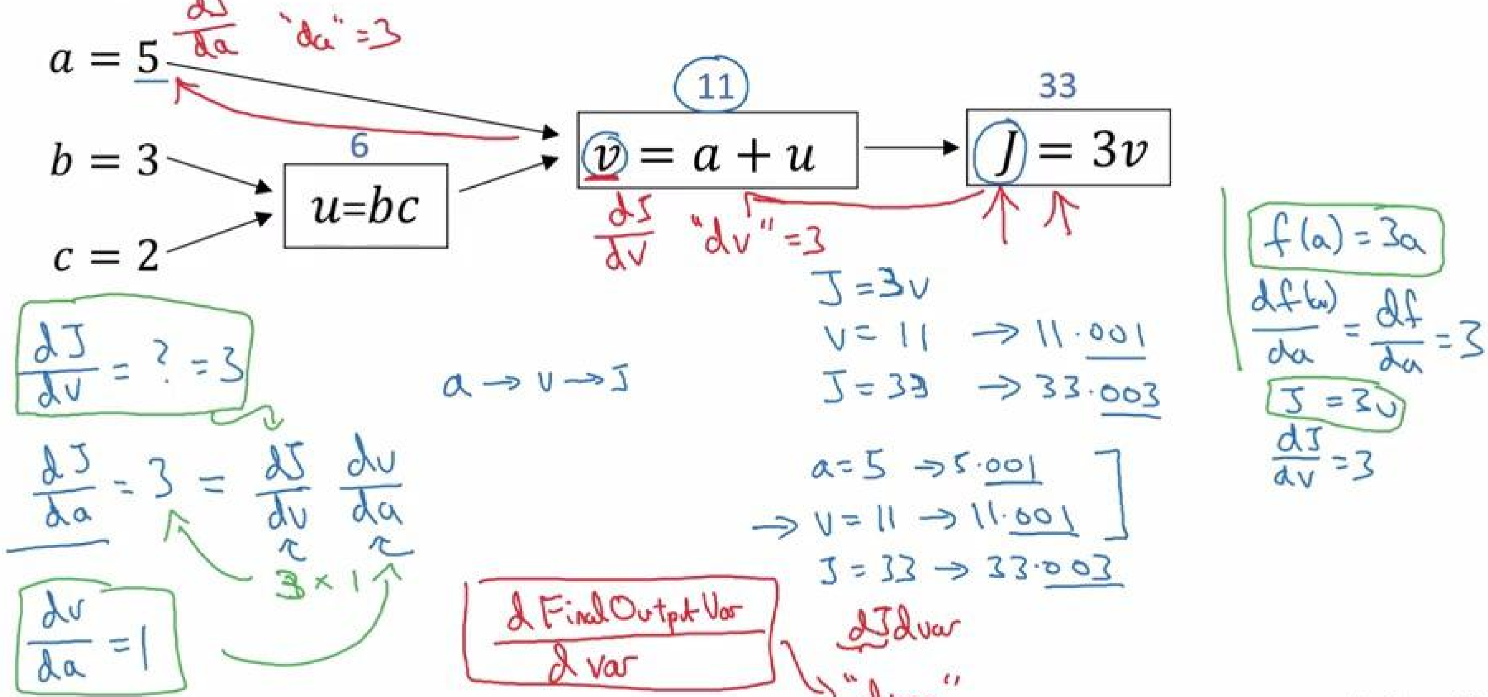
$a = 2 \quad f(a) \approx 0.69315$

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

0.0005 ⟷ 0.0005

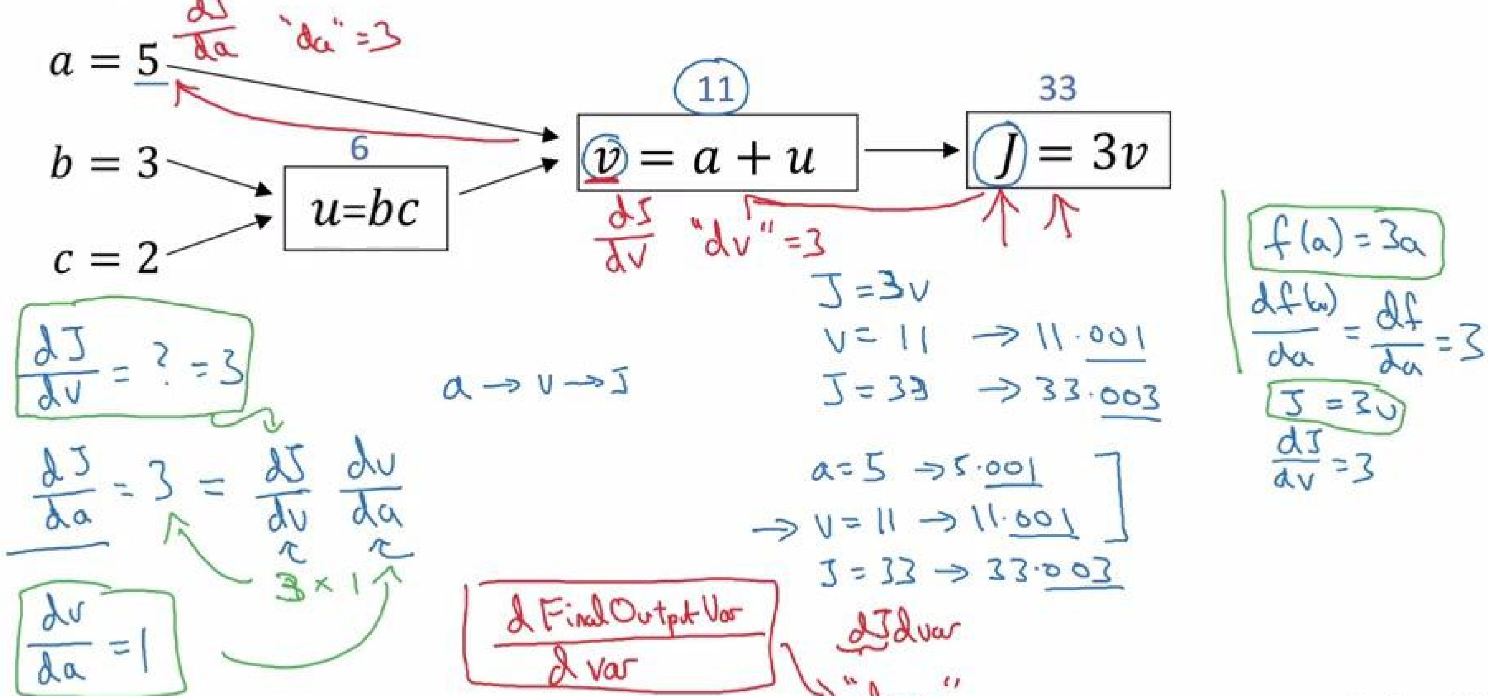
Andrew Ng

Computing derivatives



Andrew Ng

Computing derivatives



Andrew Ng

Computing derivatives

$a = 5$ $\quad \frac{dJ}{da} \rightarrow \underline{da} = 3$

$b = 3$ $\quad \frac{dJ}{db} = db = 6$

$c = 2$ $\quad \rightarrow dc = 9$

$$\overset{6}{\boxed{u=bc}} \rightarrow \overset{11}{\boxed{v = a + u}} \rightarrow \overset{33}{\boxed{J = 3v}}$$

$\underline{du} = 3$ $\quad \underline{dv} = 3$ $\quad \frac{dJ}{dJ}$

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

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

$$\frac{dJ}{dc} = \boxed{\frac{dJ}{du}} \cdot \frac{du}{dc} = 9$$
$$3 \times 3$$

$u = 6 \rightarrow 6.00\underline{1}$
$v = 11 \rightarrow 11.001$
$J = 33 \rightarrow 33.0\underline{03}$

$b = 3 \rightarrow \underline{3.001}$
$u = b \cdot c = 6 \rightarrow \underline{6.002}$ $\quad c = 2$
$J = 33.006$ $\quad .006$

$v = 11.002$
$J = 3v$

Andrew Ng

Logistic regression derivatives

x_1, w_1, x_2, w_2, b

$$\boxed{z = w_1x_1 + w_2x_2 + b} \rightarrow \boxed{a = \sigma(z)} \rightarrow \boxed{\mathcal{L}(\mathrm{a}, y)}$$

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

$$dz = \frac{d\mathcal{L}}{dz} = \frac{d\mathcal{L}(a,y)}{dz}$$
$$= a - y$$
$$= \boxed{\frac{d\mathcal{L}}{da}} \cdot \boxed{\frac{da}{dz}} \qquad \frac{da}{dz} = \boxed{a(1-a)}$$

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

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

Andrew Ng

Logistic regression on m examples

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

For $i=1$ to m

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

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

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

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

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

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

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

dw_3 ⋮ dw_n

$n=2$

$J /= m$

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

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

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

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

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

Vectorization

Andrew Ng

Implementing Logistic Regression

```
J = 0, dw1 = 0, dw2 = 0, db = 0
for i = 1 to m:
```

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

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

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

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

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

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

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

$db = db/m$

for iter in range(1000):

$Z=w^TX+b$

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

$A=\sigma(Z)$

$dZ=A-Y$

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

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

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

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

Andrew Ng

Python/numpy vectors

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

a.shape = (5,)

"rank 1 array"

Don't use

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

$\rightarrow$ a.shape = (5,1) column vector

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

$\rightarrow$ a.shape = (1,5) row vec.

Logistic regression cost function

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

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

$\Big\}\; p(y|x)$

Logistic regression cost function

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

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

$p(y|x)$

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

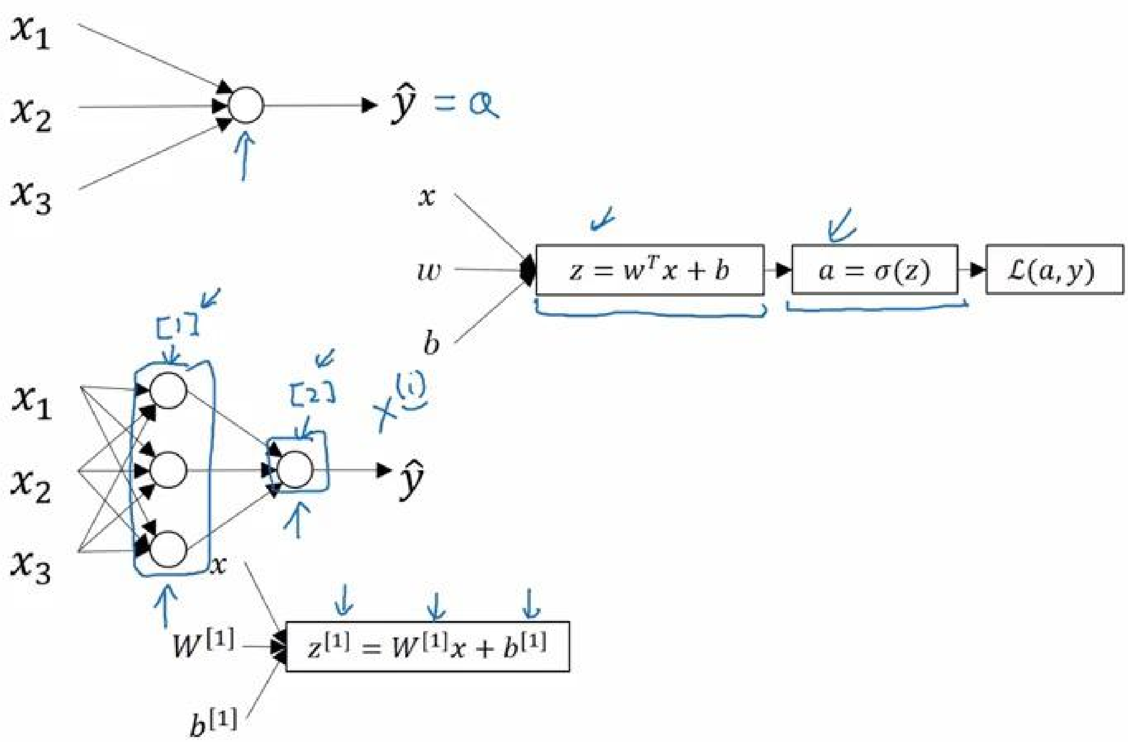
If $y=1$: $\underline{p(y|x) = \hat{y}}\ \underbrace{(1-\hat{y})^{0}}_{=1}$

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

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

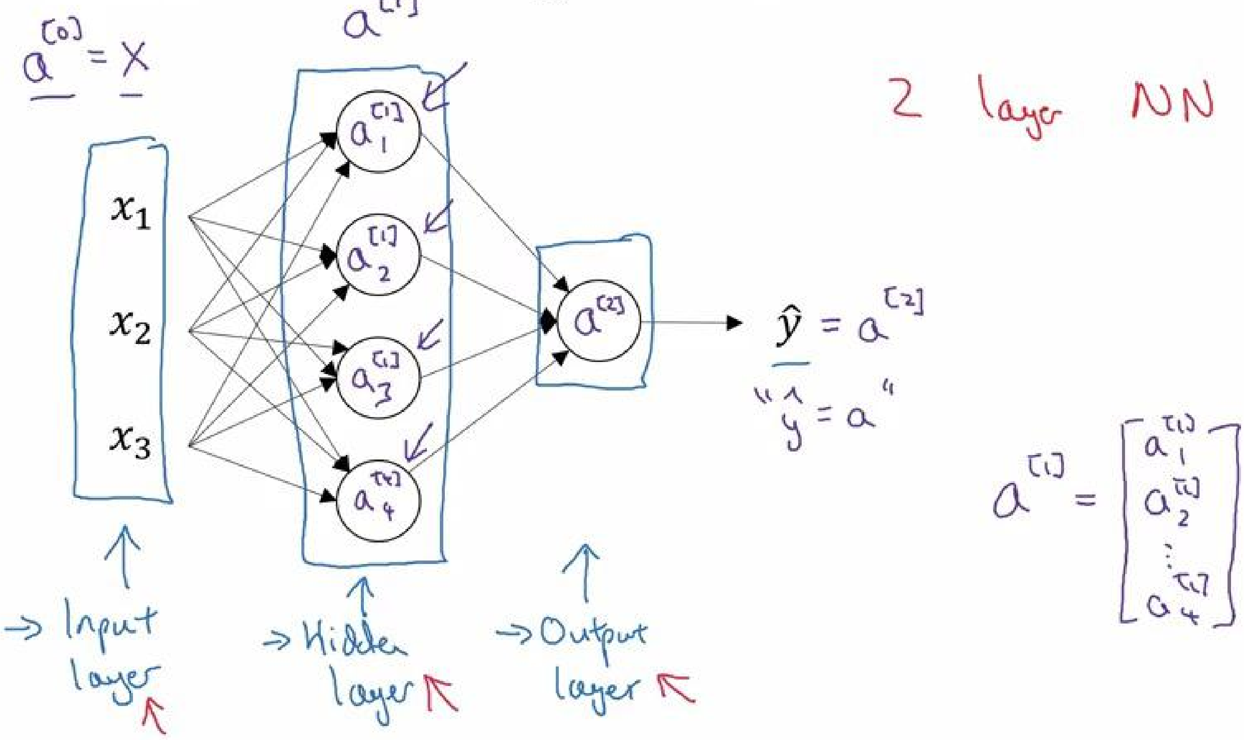
Andrew Ng

What is a Neural Network?



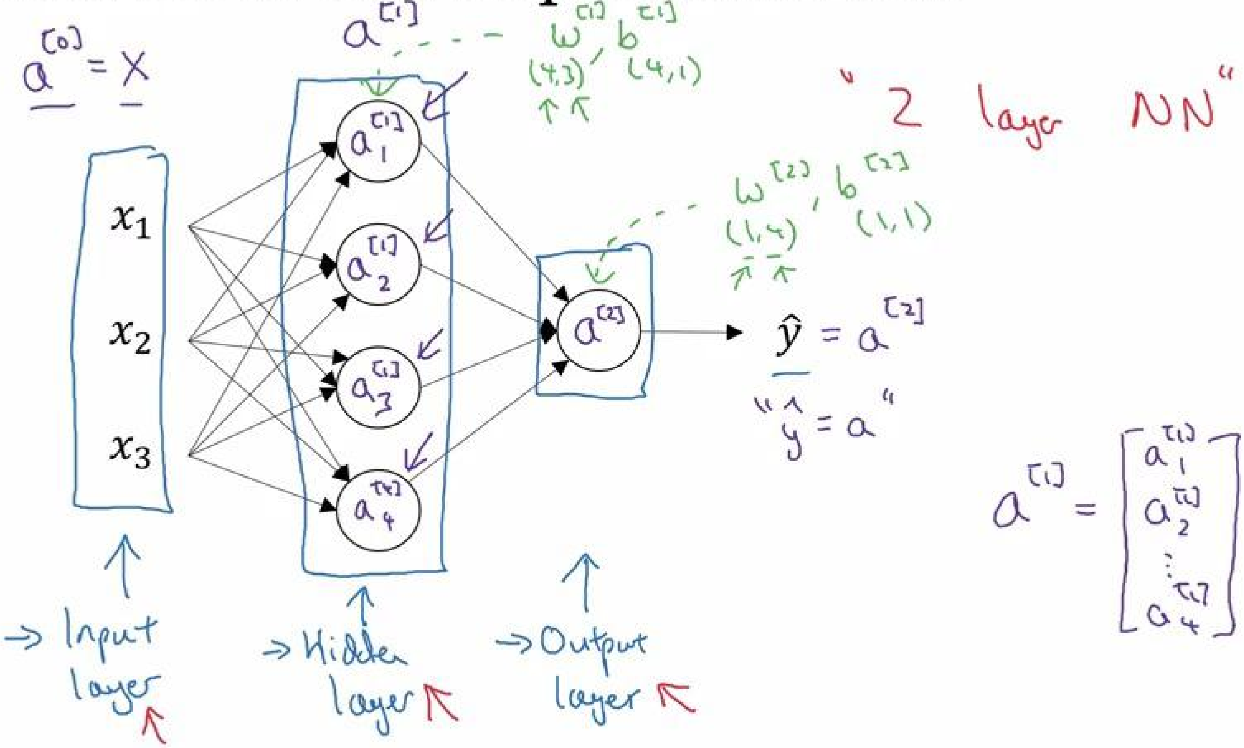
Andrew Ng

Neural Network Representation



Andrew Ng

Neural Network Representation



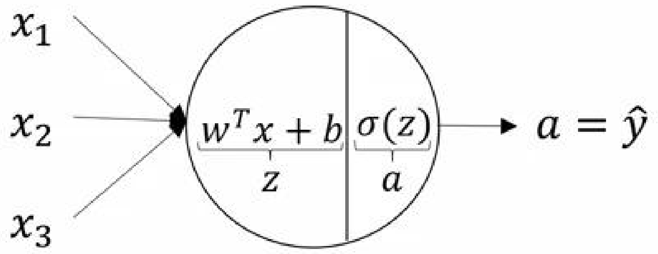
Andrew Ng

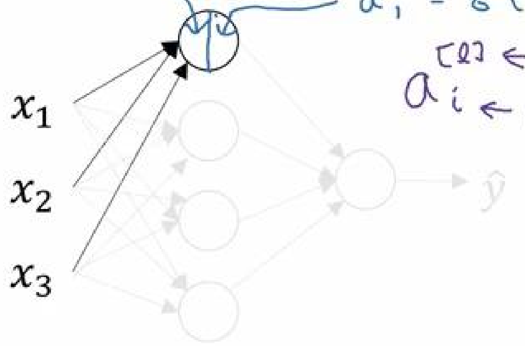
Neural Network Representation

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

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

$a_i^{[l]}$ ← layer, ← node in layer.





$z = w^T x + b$

$a = \sigma(z)$

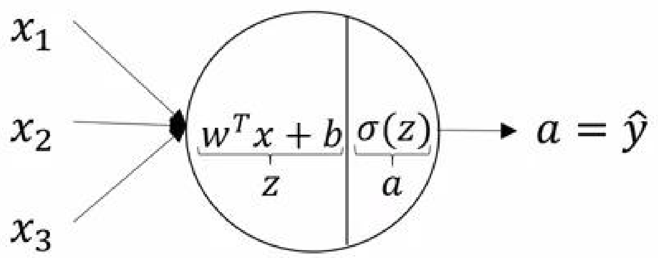
Andrew Ng

Neural Network Representation

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

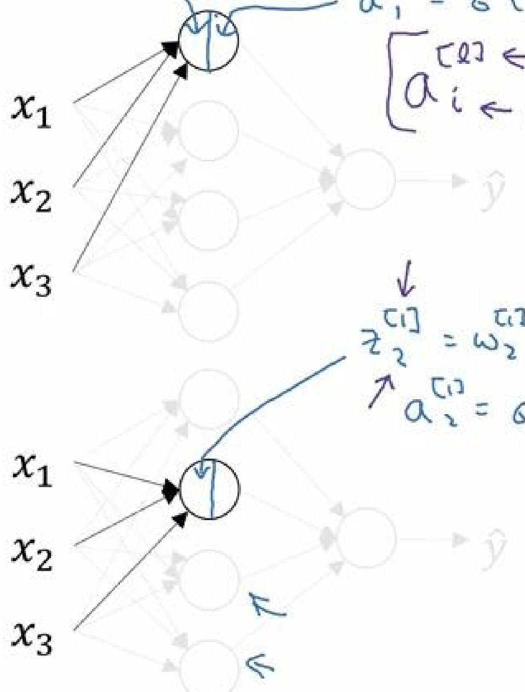
$a_1^{[1]} = \sigma(z_1^{[1]})$

$a_i^{[l]}$ ← layer, ← node in layer.



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

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



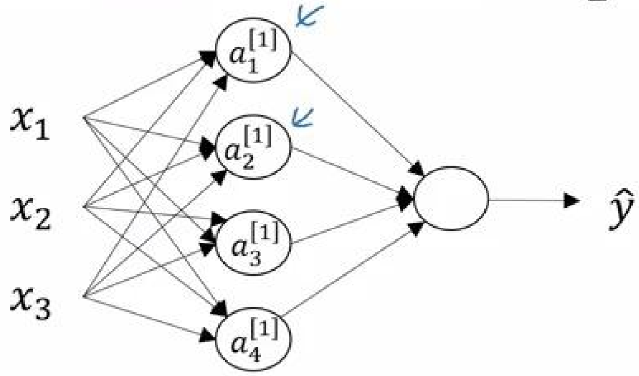
$z = w^T x + b$

$a = \sigma(z)$

Andrew Ng

Neural Network Representation

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



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

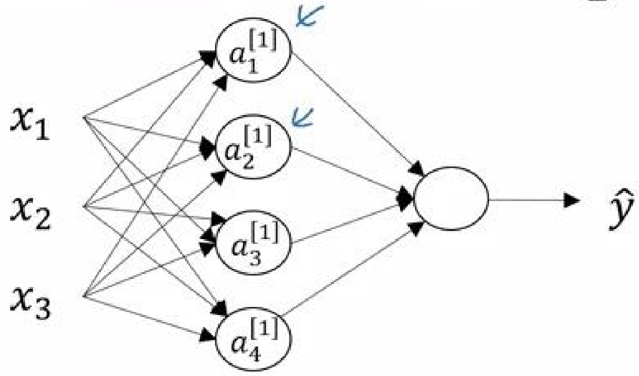
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, \ a_2^{[1]} = \sigma(z_2^{[1]})$$

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

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

Neural Network Representation

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



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

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

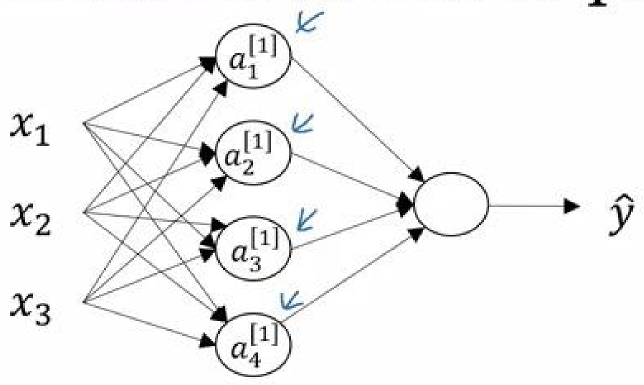
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \ a_3^{[1]} = \sigma(z_3^{[1]})$$

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

$$\begin{bmatrix} — w_1^{[1]T} — \\ — w_2^{[1]T} — \\ — w_3^{[1]T} — \\ — w_4^{[1]T} — \end{bmatrix}$$

Neural Network Representation

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



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

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

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

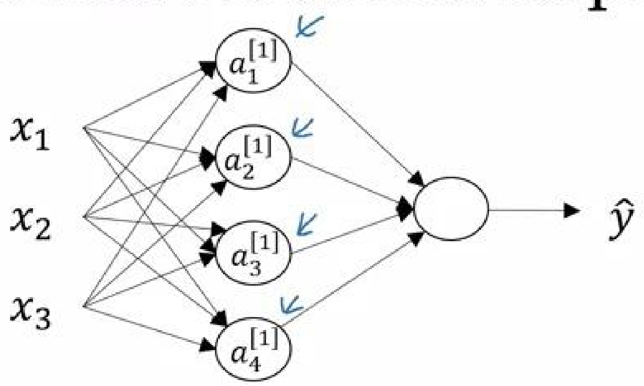
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

$$\underset{(4,3)}{\begin{bmatrix} — w_1^{[1]T} — \\ — w_2^{[1]T} — \\ — w_3^{[1]T} — \\ — w_4^{[1]T} — \end{bmatrix}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} \rightarrow w_1^{[1]T} x + b_1^{[1]} \\ \rightarrow w_2^{[1]T} x + b_2^{[1]} \\ \rightarrow w_3^{[1]T} x + b_3^{[1]} \\ \rightarrow w_4^{[1]T} x + b_4^{[1]} \end{bmatrix}$$

Andrew Ng

Neural Network Representation

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



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

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

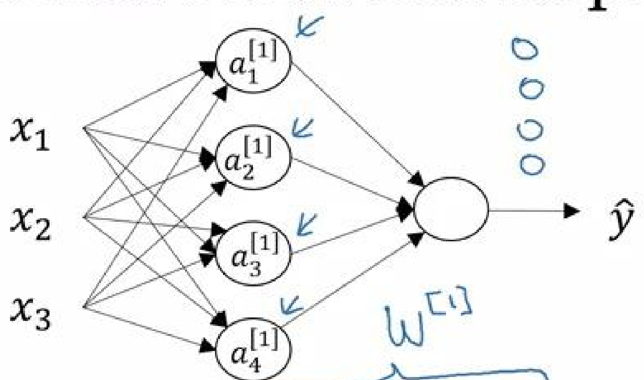
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$

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

$$\underset{(4,3)}{\begin{bmatrix} — w_1^{[1]T} — \\ — w_2^{[1]T} — \\ — w_3^{[1]T} — \\ — w_4^{[1]T} — \end{bmatrix}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} \rightarrow w_1^{[1]T} x + b_1^{[1]} \\ \rightarrow w_2^{[1]T} x + b_2^{[1]} \\ \rightarrow w_3^{[1]T} x + b_3^{[1]} \\ \rightarrow w_4^{[1]T} x + b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

Andrew Ng

Neural Network Representation



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

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

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

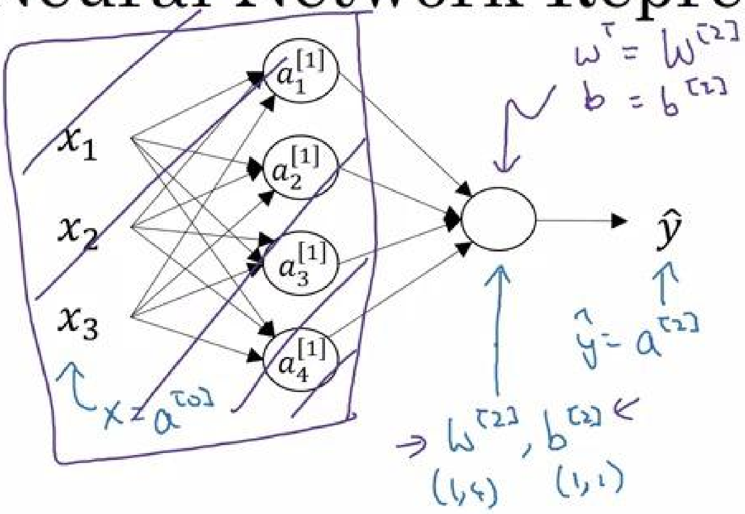
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$

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

$$z^{[1]} = \overbrace{\underset{(4,3)}{\begin{bmatrix} — w_1^{[1]T} — \\ — w_2^{[1]T} — \\ — w_3^{[1]T} — \\ — w_4^{[1]T} — \end{bmatrix}}}^{W^{[1]}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \underbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}}_{b^{[1]}\ (4,1)} = \begin{bmatrix} w_1^{[1]T} x + b_1^{[1]} \\ w_2^{[1]T} x + b_2^{[1]} \\ w_3^{[1]T} x + b_3^{[1]} \\ w_4^{[1]T} x + b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

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

Neural Network Representation learning



Given input x:

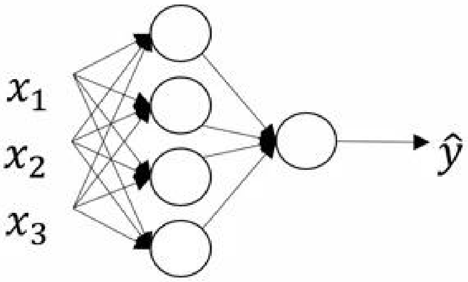
$$\underset{(4,1)}{z^{[1]}} = \underset{(4,3)}{W^{[1]}} \underset{(3,1)}{\overset{a^{[0]}}{x}} + \underset{(4,1)}{b^{[1]}}$$

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

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

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

Vectorizing across multiple examples



$$\left\{\begin{aligned} z^{[1]} &= W^{[1]}x + b^{[1]} \\ a^{[1]} &= \sigma(z^{[1]}) \\ z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\ a^{[2]} &= \sigma(z^{[2]}) \end{aligned}\right.$$

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

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

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

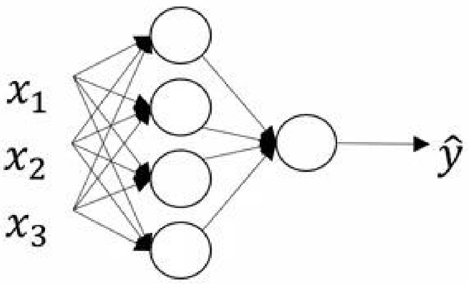
$\vdots$

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

$a^{[2](i)}$ ← example i ← layer 2

Andrew Ng

Vectorizing across multiple examples



$$\left\{\begin{aligned} z^{[1]} &= W^{[1]}x + b^{[1]} \\ a^{[1]} &= \sigma(z^{[1]}) \\ z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\ a^{[2]} &= \sigma(z^{[2]}) \end{aligned}\right\} -$$

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

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

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

$\vdots$

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

$a^{[2](i)}$ ← example i ← layer 2

for $i = 1$ to m,

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

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

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

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

Andrew Ng

Vectorizing across multiple examples

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

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

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

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

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

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

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

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

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

Andrew Ng

Vectorizing across multiple examples

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

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

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

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

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

$$\boxed{\begin{aligned} Z^{[1]} &= W^{[1]}X + b^{[1]} \\ A^{[1]} &= \sigma(Z^{[1]}) \\ Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\ A^{[2]} &= \sigma(Z^{[2]}) \end{aligned}}$$

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

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

Andrew Ng

Justification for vectorized implementation

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

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

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

Andrew Ng

Justification for vectorized implementation

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

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

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

Andrew Ng

Justification for vectorized implementation

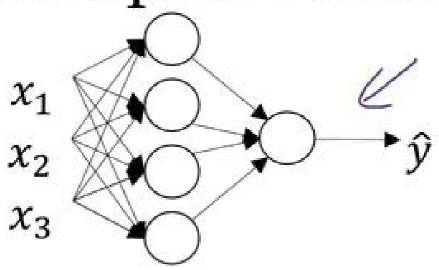
$z^{[1](1)} = W^{[1]}x^{(1)} + \cancel{b^{[1]}} \to 0, \quad z^{[1](2)} = W^{[1]}x^{(2)} + \cancel{b^{[1]}} \to 0, \quad z^{[1](3)} = W^{[1]}x^{(3)} + \cancel{b^{[1]}} \to 0$

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

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

Andrew Ng

Recap of vectorizing across multiple examples



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

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

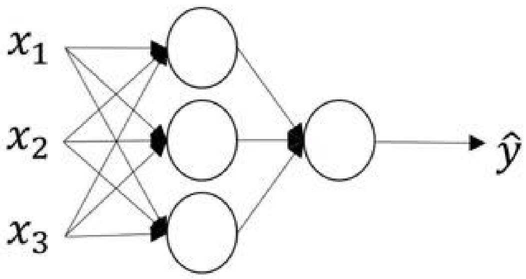
```
for i = 1 to m
```

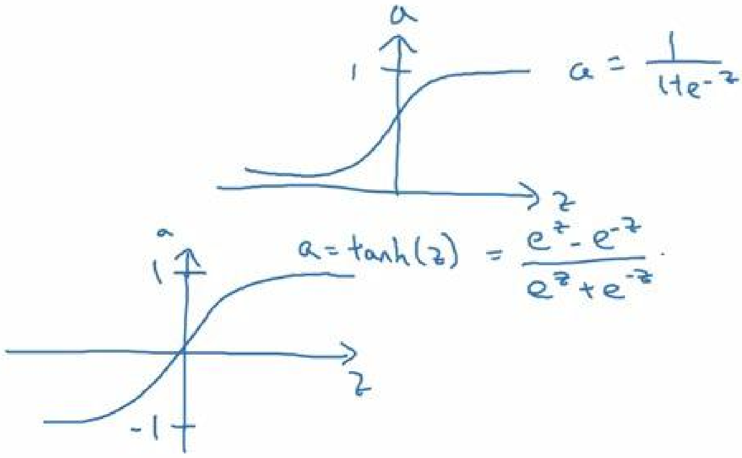
$$\boxed{z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}}$$
$$a^{[1](i)} = \sigma(z^{[1](i)})$$
$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$
$$a^{[2](i)} = \sigma(z^{[2](i)})$$

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

Andrew Ng

Activation functions





Given x:

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

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

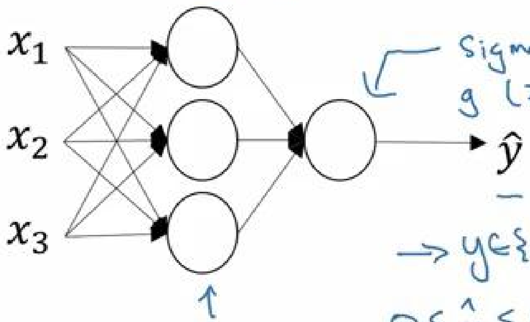
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

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

Andrew Ng

Activation functions

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



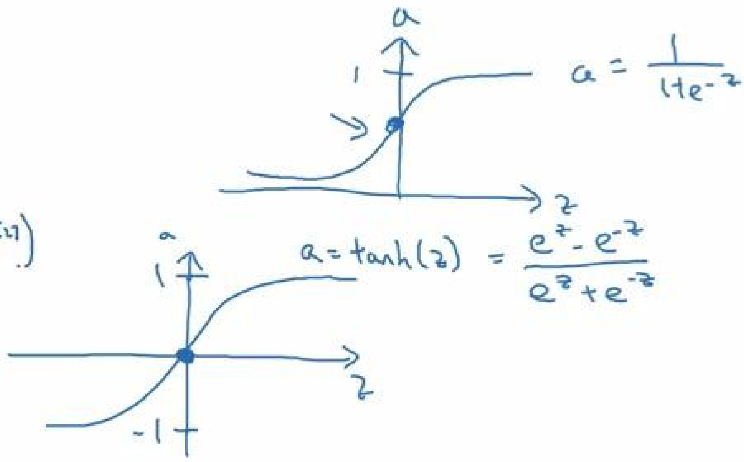
sigmoid

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

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

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

$-1 \quad 1$



Given x:

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

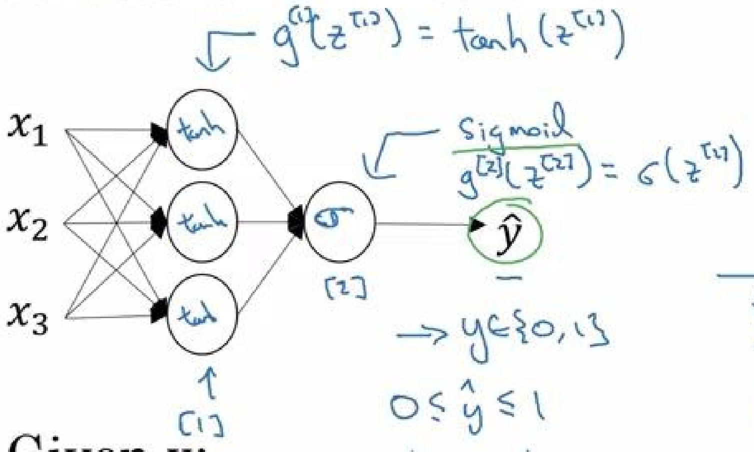
$$\rightarrow a^{[1]} = \cancel{\sigma(z^{[1]})}\ g(z^{[1]})$$

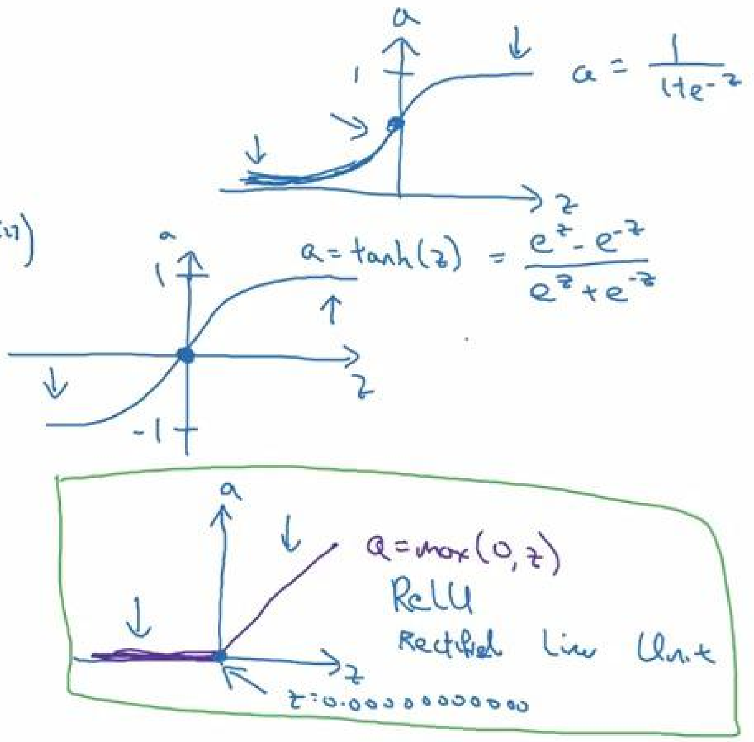
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

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

Andrew Ng

Activation functions



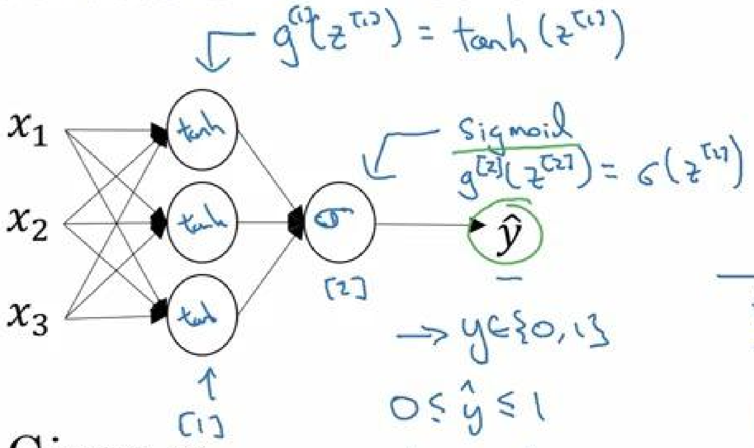


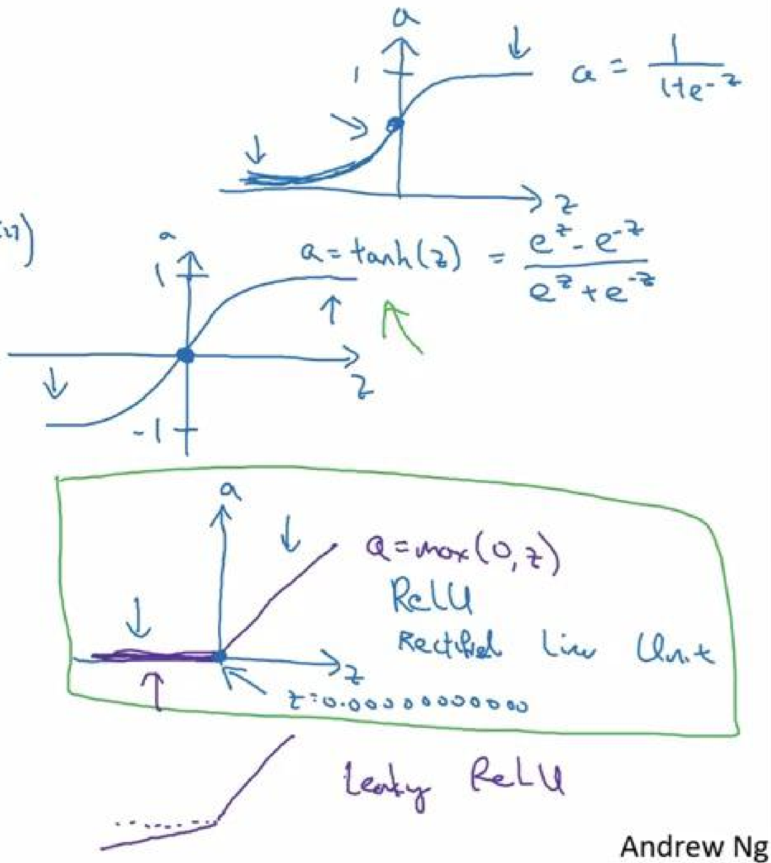
Given x:

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

Andrew Ng

Activation functions



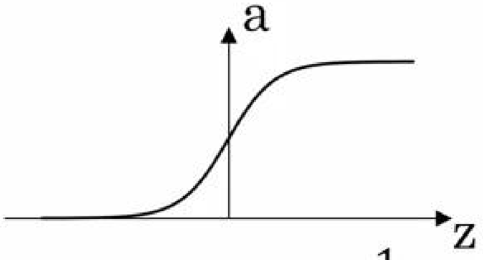


Given x:

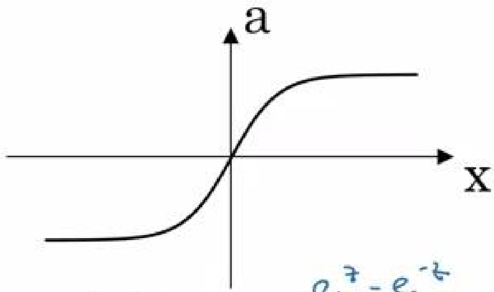
$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = \cancel{\sigma(z^{[1]})}\ g^{[1]}(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = \cancel{\sigma(z^{[2]})}\ g^{[2]}(z^{[2]})$$

Andrew Ng

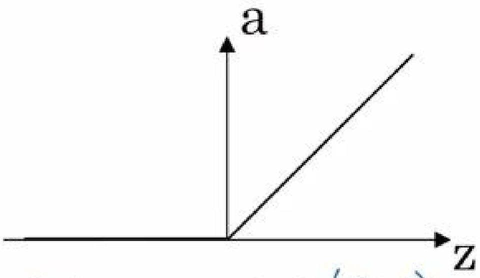
Pros and cons of activation functions



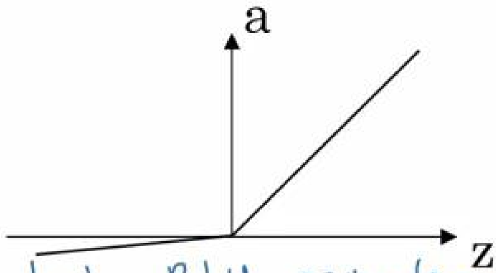
sigmoid: $a = \frac{1}{1+e^{-z}}$



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



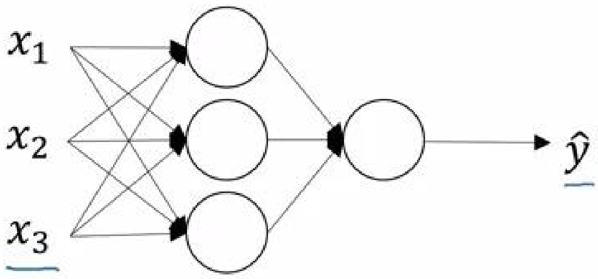
ReLU $a = \max(0, z)$



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

Andrew Ng

Activation function



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

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

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

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

$$= W'x + b'$$

Given x:

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

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

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

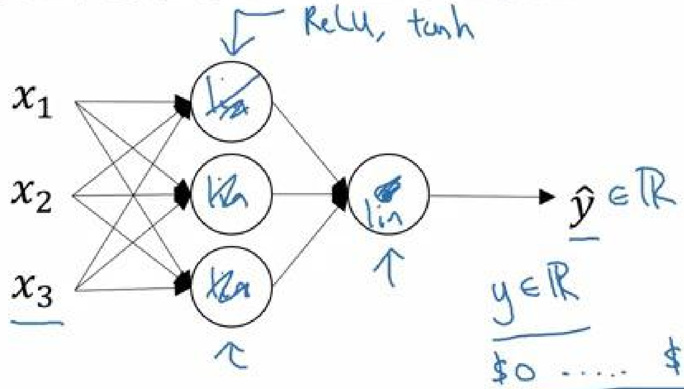
$\rightarrow$ $a^{[2]} = \cancel{g^{[2]}(z^{[2]})}\ z^{[2]}$

$g(z) = z$

"linear activation function"

Andrew Ng

Activation function



$y \in \mathbb{R}$

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

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

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

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

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

$= W'x + b'$

Given x:

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

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

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

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

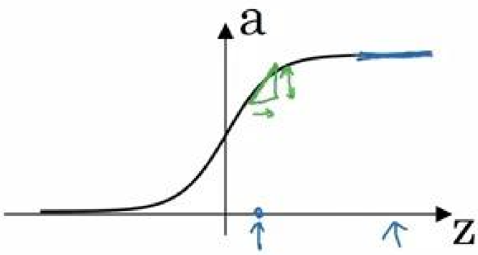
$g(z) = z$

"linear activation function"

$g(z) = z$

Andrew Ng

Sigmoid activation function



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

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

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

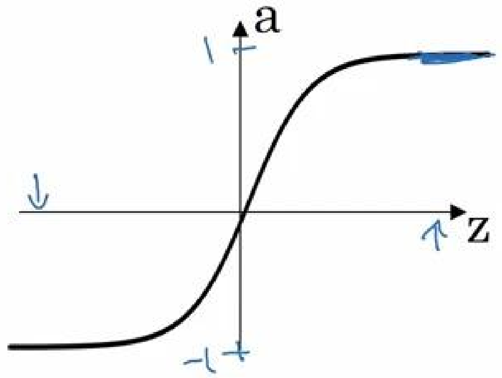
$= g(z)(1 - g(z))$

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

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

Andrew Ng

Tanh activation function



$g(z) = \tanh(z)$

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

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

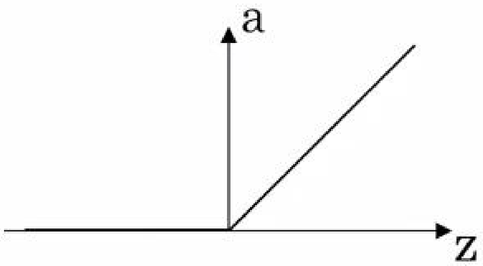
$= 1 - (\tanh(z))^2 \leftarrow$

$z = 10 \quad \tanh(z) \approx 1$

$g'(z) \approx 0$

Andrew Ng

ReLU and Leaky ReLU



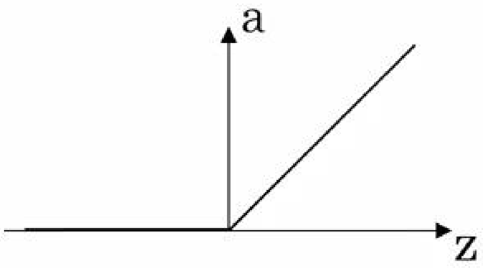
ReLU

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

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

Andrew Ng

ReLU and Leaky ReLU

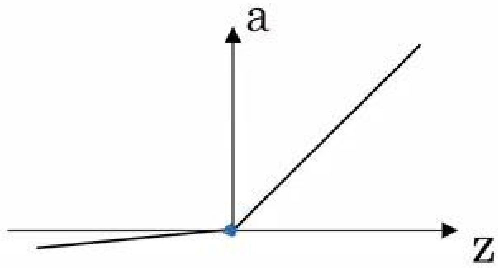


ReLU

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

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

$z = 0.00000\ldots0$



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

Gradient descent for neural networks

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

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

Gradient descent:

Repeat {

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

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

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

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

$b^{[2]} := \ldots \; b^{[2]} -$

Formulas for computing derivatives

Forward propagation:

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

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

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

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

Formulas for computing derivatives

Forward propagation:

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

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

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

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

Back propagation:

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

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

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

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

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

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

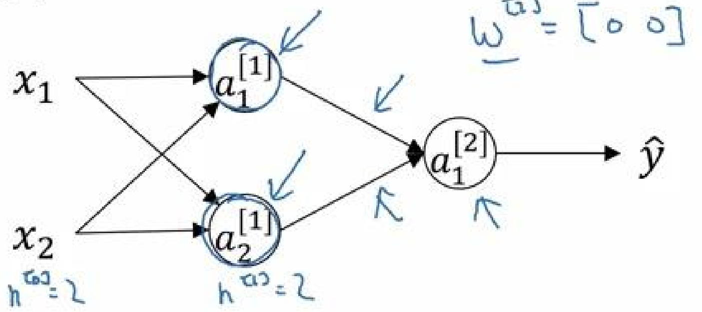
$$dZ^{[1]} = \underbrace{W^{[2]T} dZ^{[2]}}_{(n^{[1]},m)} * \underbrace{g^{[1]\prime}(Z^{[1]})}_{(n^{[1]},m)}$$

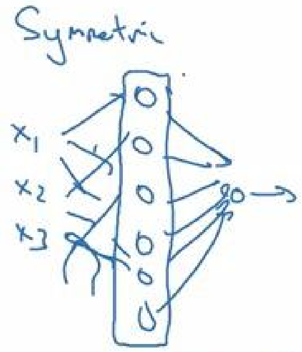
element-wise product

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

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

What happens if you initialize weights to zero?





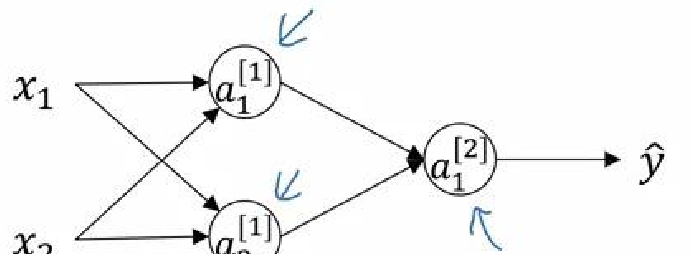
$W^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ $\quad b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

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

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

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

Random initialization



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

100?

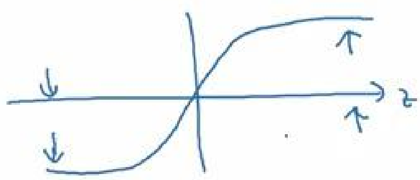
$b^{[1]} = np.zero((2,1))$

$W^{[2]} = \ldots$

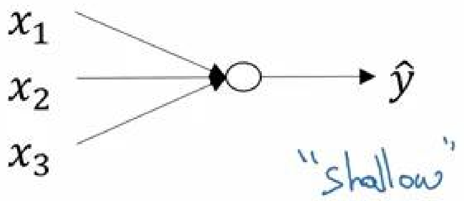
$b^{[2]} = 0$

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

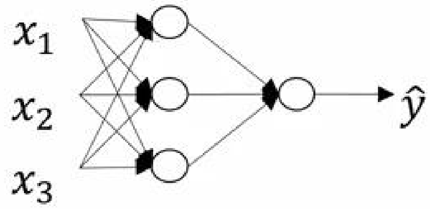
$a^{[1]} = g^{[1]}(z^{[1]})$



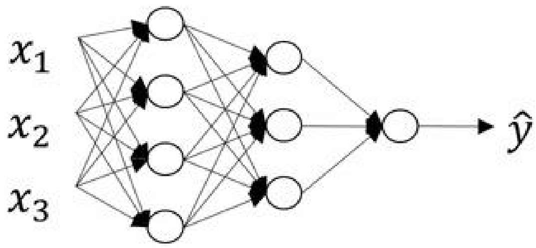
What is a deep neural network?



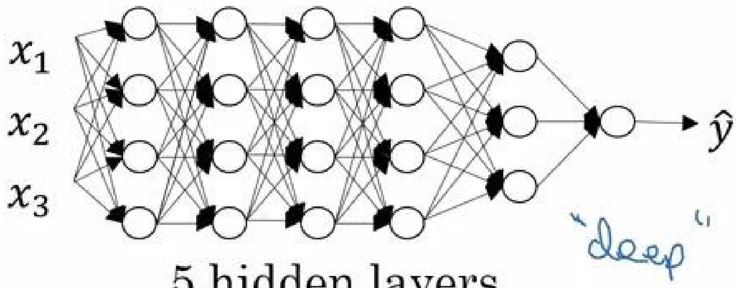
logistic regression



1 hidden layer



2 hidden layers

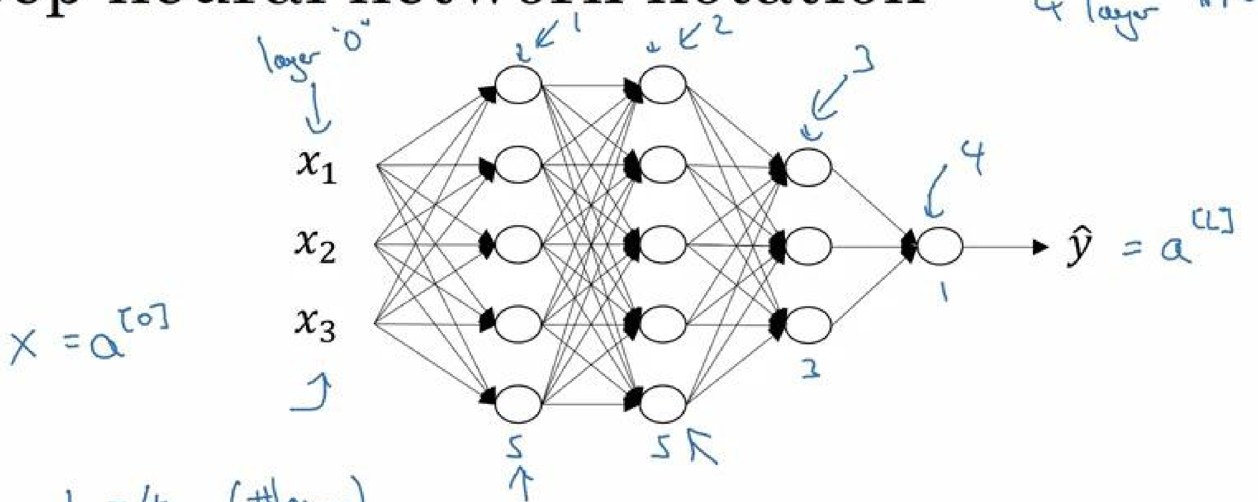


5 hidden layers

Andrew Ng

Deep neural network notation

4 layer NN



$L = 4$ (#layers)

$n^{[l]}$ = #units in layer l $\quad n^{[1]} = 5,\ n^{[2]} = 5,\ n^{[3]} = 3,\ n^{[4]} = n^{[L]} = 1$

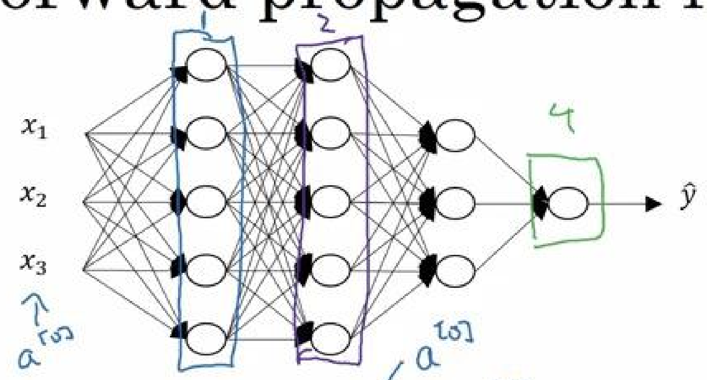
$n^{[0]} = n_x = 3$

$a^{[l]}$ = activations in layer l

$a^{[l]} = g^{[l]}(z^{[l]})$, $W^{[l]}$ = weights for $\underline{z^{[l]}}$, $b^{[l]}$

Andrew Ng

Forward propagation in a deep network



$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g^{[l]}(z^{[l]})$$

$X:\ z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$

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

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

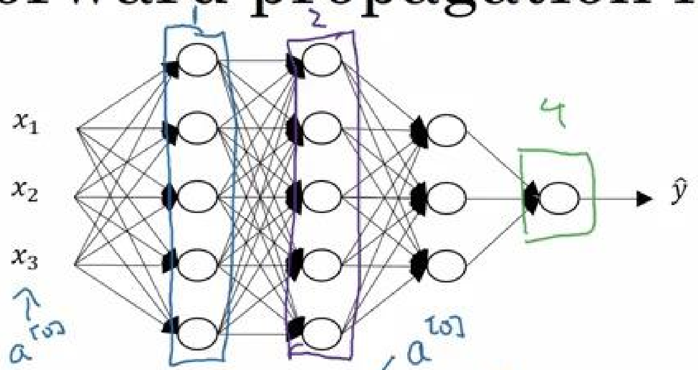
$a^{[2]} = g^{[2]}(z^{[2]})$

$\vdots$

$z^{[4]} = W^{[4]} a^{[3]} + b^{[4]}, \quad a^{[4]} = g^{[4]}(z^{[4]}) = \hat{y}$

Andrew Ng

Forward propagation in a deep network



$A^{[0]} = X$

$$\boxed{\begin{aligned} Z^{[l]} &= W^{[l]} A^{[l-1]} + b^{[l]} \\ A^{[l]} &= g^{[l]}(Z^{[l]}) \end{aligned}}$$

$X:\ z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$

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

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

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

$\vdots$

$z^{[4]} = W^{[4]} a^{[3]} + b^{[4]}, \quad a^{[4]} = g^{[4]}(z^{[4]}) = \hat{y}$

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

Vectorized:

$X = A^{[0]}$

$Z^{[1]} = W^{[1]} \cancel{X} A^{[0]} + b^{[1]}$

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

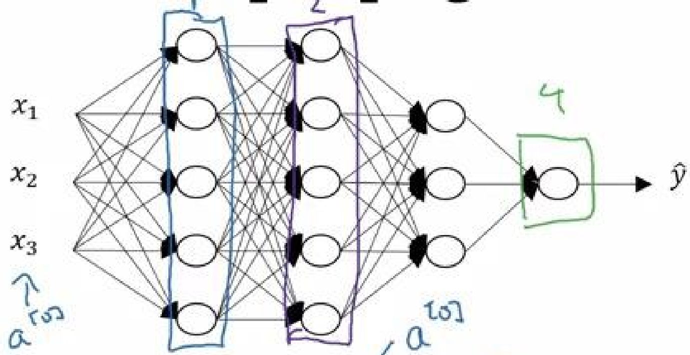
$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$

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

$\hat{Y} = g(Z^{[4]}) = A^{[4]}$

Andrew Ng

Forward propagation in a deep network



$A^{[0]} = X$

$$\boxed{\begin{aligned} Z^{[l]} &= W^{[l]} A^{[l-1]} + b^{[l]} \\ A^{[l]} &= g^{[l]}(Z^{[l]}) \end{aligned}}$$

x : $z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$

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

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

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

$z^{[4]} = W^{[4]} a^{[3]} + b^{[4]}$, $a^{[4]} = g^{[4]}(z^{[4]}) = \hat{y}$

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

Vectorized:

$X = A^{[0]}$

for $l = 1 \ldots 4$

$Z^{[1]} = W^{[1]} A^{[0]} + b^{[1]}$

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

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

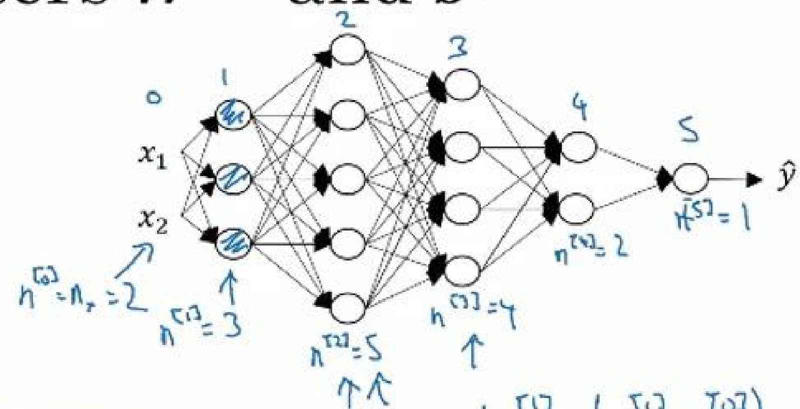
$A^{[2]} = g^{[2]}(Z^{[2]})$

$\hat{Y} = g(Z^{[4]}) = A^{[4]}$

Andrew Ng

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

$L = 5$



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

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

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

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

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

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

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

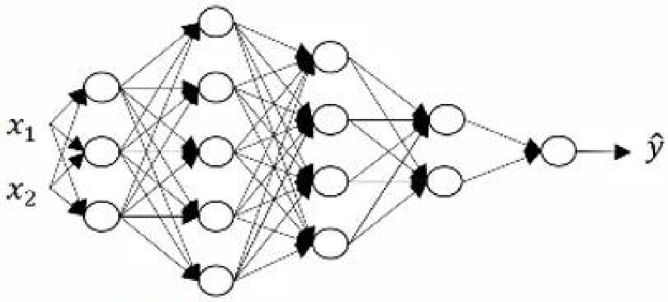
$(5,1) \quad (5,3) \quad (3,1) \quad (5,1)$

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

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

Andrew Ng

Vectorized implementation



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

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

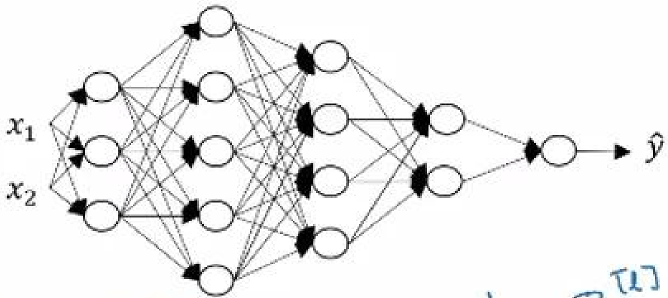
$$\left[z^{[1](1)} \; z^{[1](2)} \cdots z^{[1](m)} \right]$$

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

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

$n^{[1]},$

Vectorized implementation



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

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

$$\left[z^{[1](1)} \; z^{[1](2)} \cdots z^{[1](m)} \right]$$

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

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

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

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

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

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

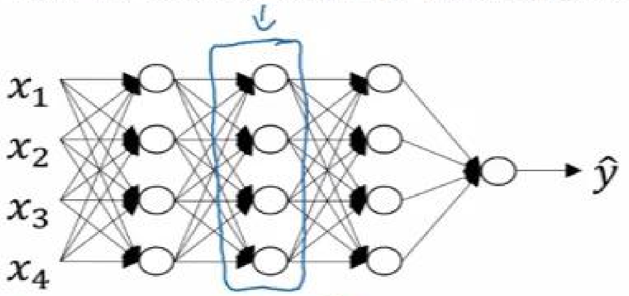
$dZ^{[l]}, dA^{[l]} : (n^{[l]},m)$

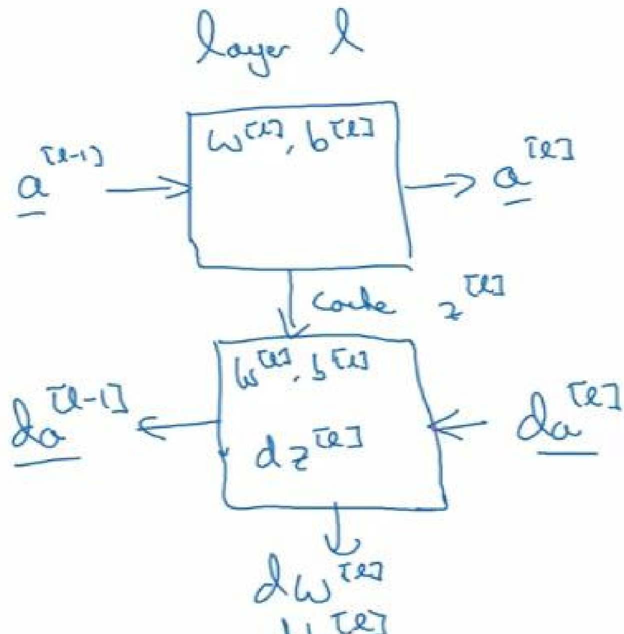
Circuit theory and deep learning

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.

Andrew Ng

Forward and backward functions





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

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

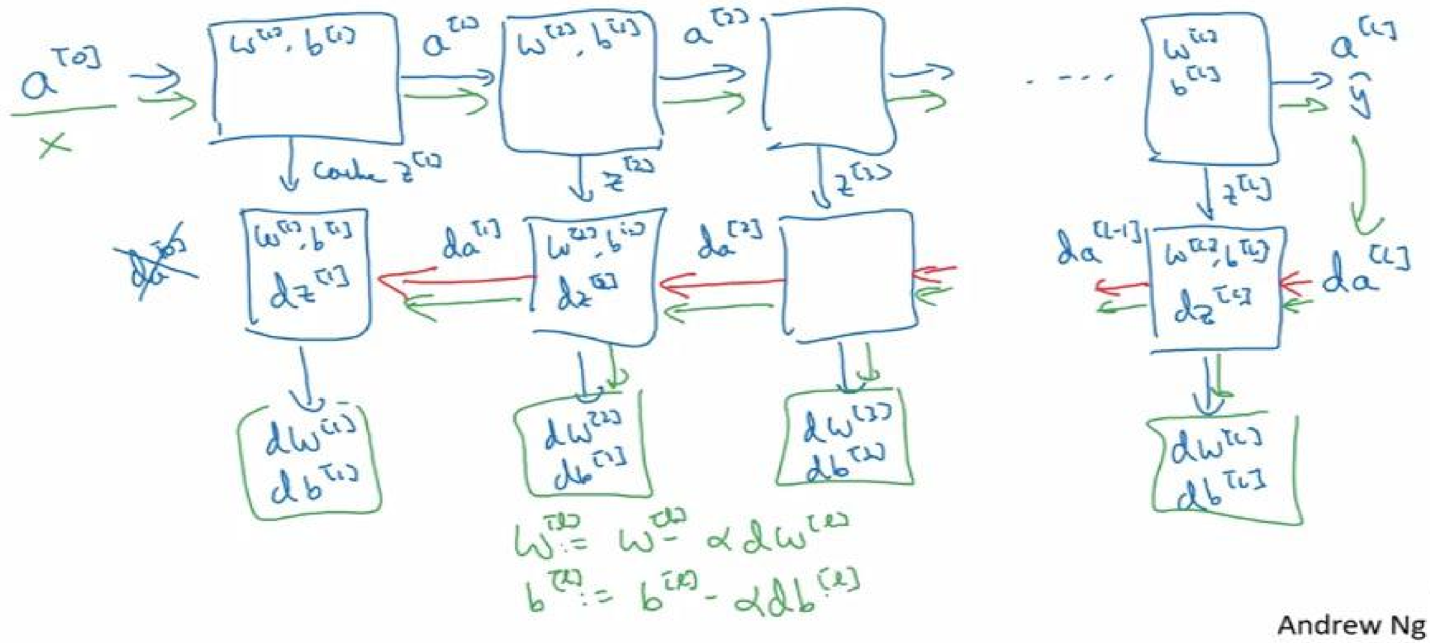
$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$ cache $z^{[l]}$

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

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

Andrew Ng

Forward and backward functions



Andrew Ng

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

Andrew Ng

What are hyperparameters?

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

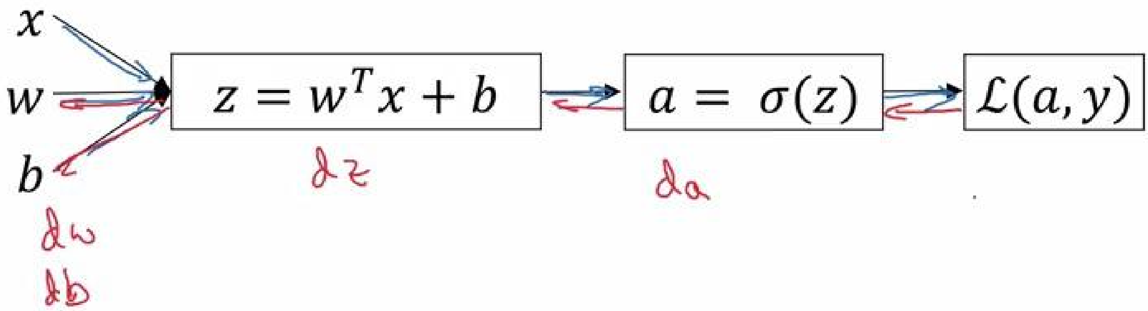
Hyperparameters: learning rate α
#iterations
#hidden layers L
#hidden units $n^{[1]}, n^{[2]}, \ldots$
choice of activation function

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

Andrew Ng

Computing gradients

Logistic regression



Andrew Ng

Neural network gradients

$x_1, x_2, x_3 \to \circ \to \hat{y}$

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

$W^{[2]}$, $b^{[2]}$ $\quad dW^{[2]}$, $db^{[2]}$

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

$dW^{[1]}$, $db^{[1]}$

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

$* \, g^{[1]\prime}(z^{[1]})$
element wise product

$dW^{[2]} = dz^{[2]} a^{[1]T}$ $\quad \to$ "$dw = dz \cdot x$"

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

$w^{[1]}$: [———]

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

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

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

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

$dW^{[1]} = dz^{[1]} \cdot x^T$ $\quad a^{[0]T}$

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

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

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

Andrew Ng

Computing gradients

Logistic regression

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

dw, db $\quad dz$

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

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

Andrew Ng

Computing gradients

Logistic regression

$$\left.\begin{matrix} x \\ w \\ b \end{matrix}\right\} \rightarrow \boxed{z = w^T x + b} \rightarrow \boxed{a = \sigma(z)} \rightarrow \boxed{\mathcal{L}(a, y)}$$

dw

db

$dz = a - y$

$dz = da \cdot g'(z)$

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

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

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

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:

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

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

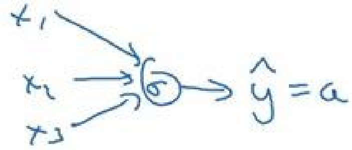
$Z^{[1]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$

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

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

Andrew Ng

Computing gradients



Logistic regression

$$\left.\begin{array}{l} x \\ w \\ b \end{array}\right\} \to \boxed{z = w^T x + b} \to \boxed{a = \sigma(z)} \to \boxed{\mathcal{L}(a, y)}$$

dw

db

$dz = a - y$

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

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

$dz = da \cdot g'(z)$

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

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

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

Andrew Ng

Neural network gradients

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

$$\left.\begin{array}{l} x \\ W^{[1]} \\ b^{[1]} \end{array}\right\} \to \boxed{z^{[1]} = W^{[1]}x + b^{[1]}} \to \boxed{a^{[1]} = \sigma(z^{[1]})} \to \boxed{z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}} \to \boxed{a^{[2]} = \sigma(z^{[2]})} \to \boxed{\mathcal{L}(a^{[2]}, y)}$$

$W^{[2]}$, $b^{[2]}$

$dW^{[2]}$

$db^{[2]}$

$dz^{[1]} = W^{[2]T} dz^{[2]}$

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

$da^{[1]} = \ldots$

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

$da^{[2]}$

$dW^{[2]} = dz^{[2]} a^{[1]T}$ $\to$ "$dw = dz \cdot x$"

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

$W^{[1]}$: [———]

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

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

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

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

dfoo $\quad dW^{[1]}$

Andrew Ng