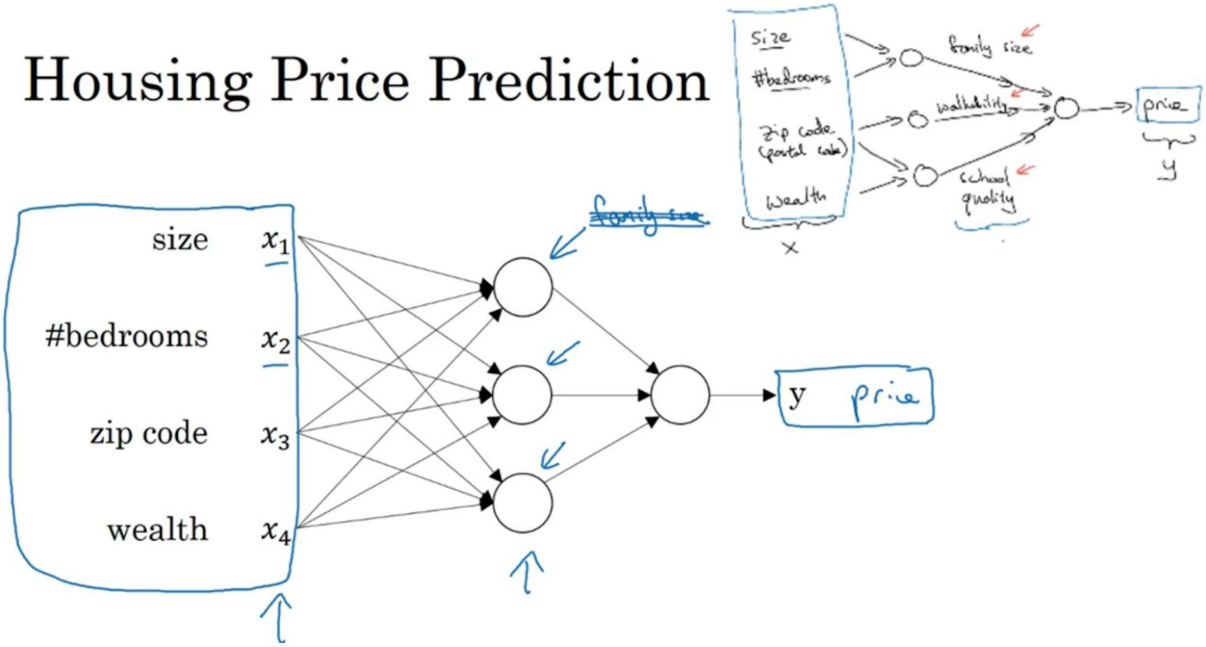
Course 1: NEURAL NETWORKS AND DEEP LEARNING



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

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

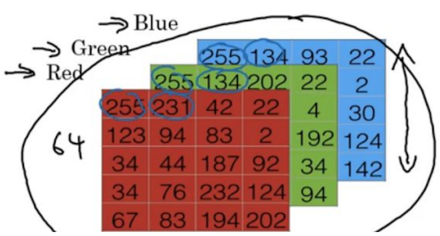
Binary Classification

64

64

→ 1 (cat) vs 0 (non cat)

y



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

$64 \times 64 \times 3 = 12288$

$n = n_x = 12288$

$x \longrightarrow y$

Notation

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

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

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

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

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

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

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

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

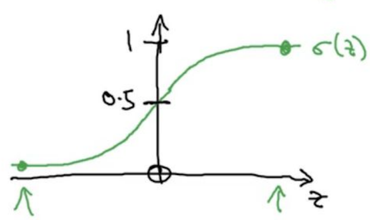
Logistic Regression

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

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

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

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



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

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

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

If z large negative number

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

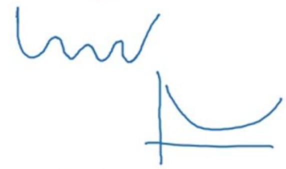
Logistic Regression cost function

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

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



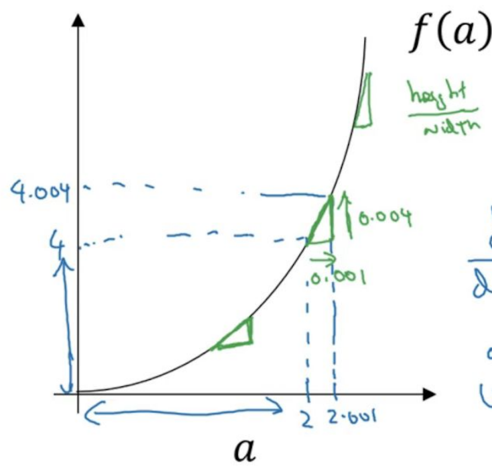
$$\boxed{\mathcal{L}(\hat{y}, y)} = -\left(\boxed{y \log \hat{y}} + (1-y)\log(1-\hat{y})\right) \leftarrow$$

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

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

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

Intuition about derivatives



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

0.001

(2a).

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

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

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

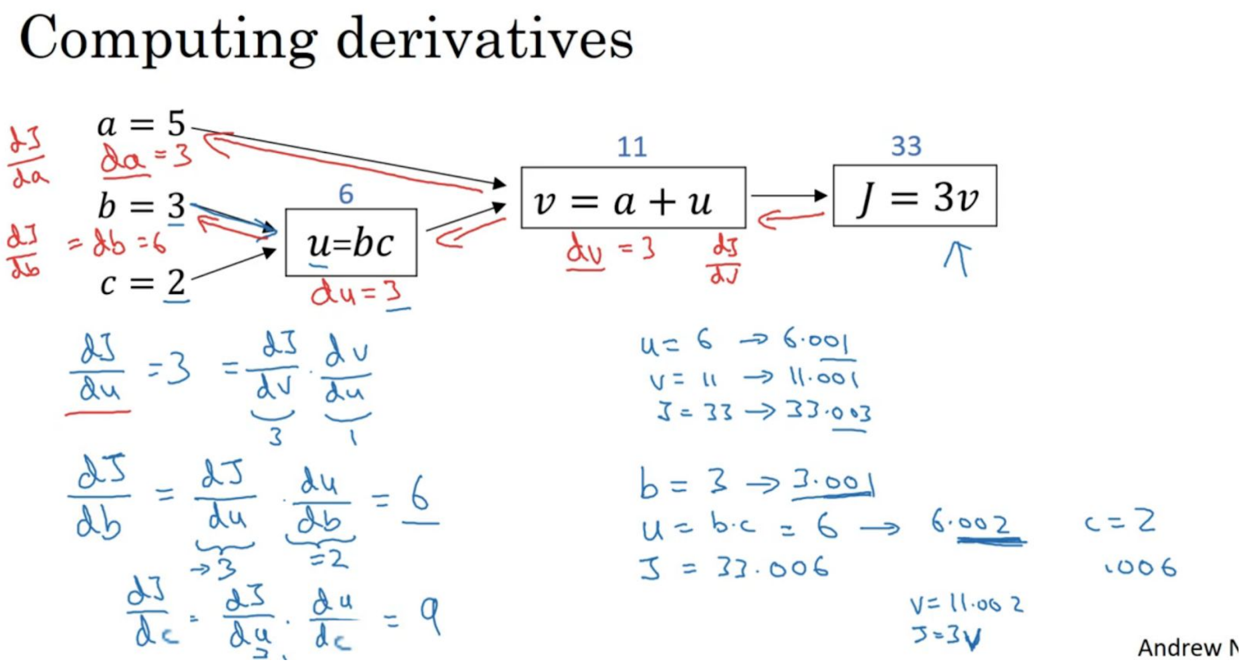
$\boxed{\frac{d}{da} f(a) = 4}$ when $\boxed{a=2}$.

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

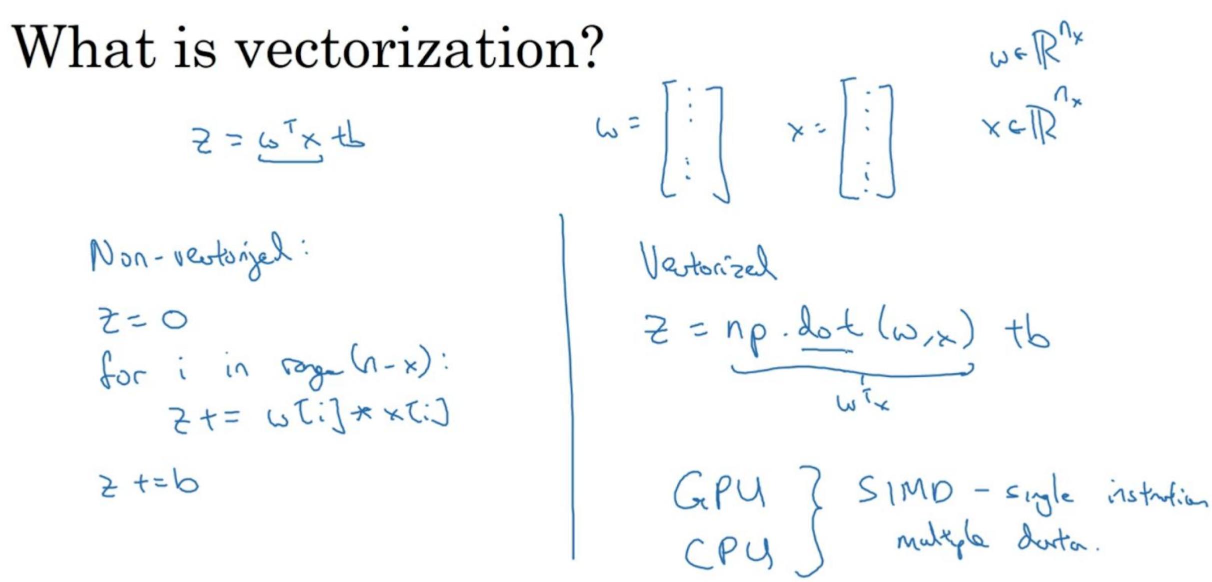
$a = 5.001$ $f(a) \approx 25.010$

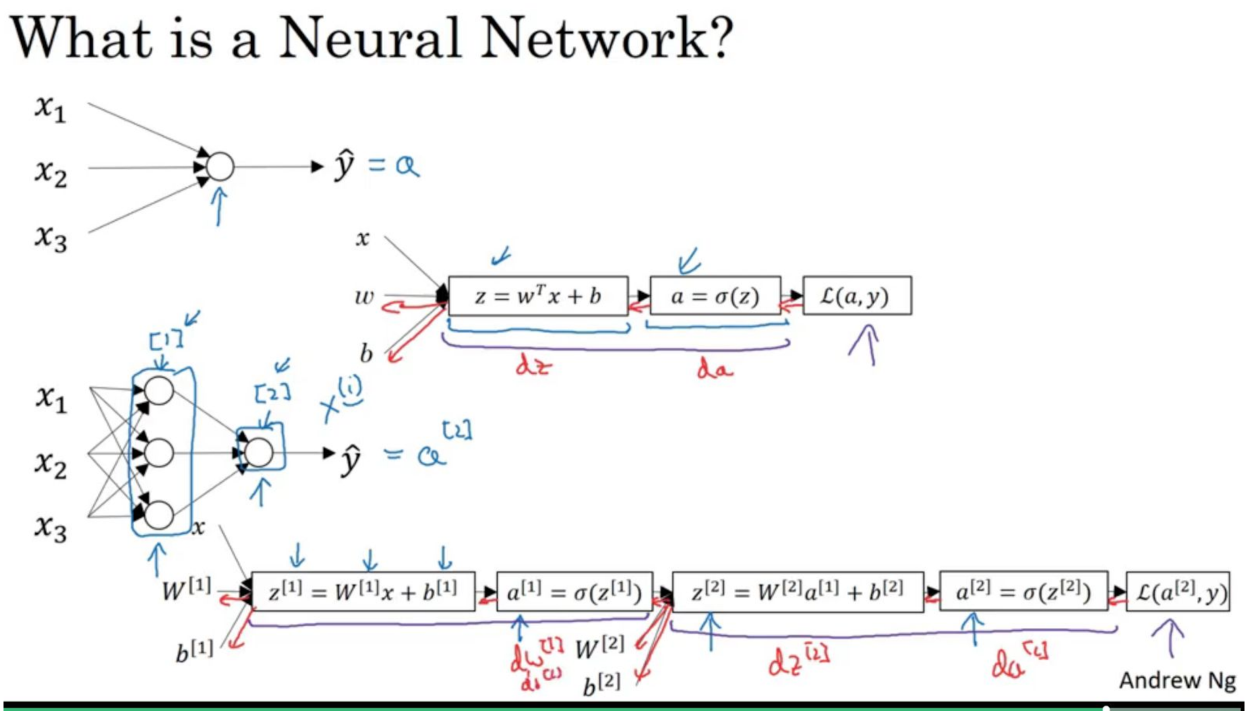
$\boxed{\frac{d}{da} f(a) = 10}$ when $\boxed{a=5}$

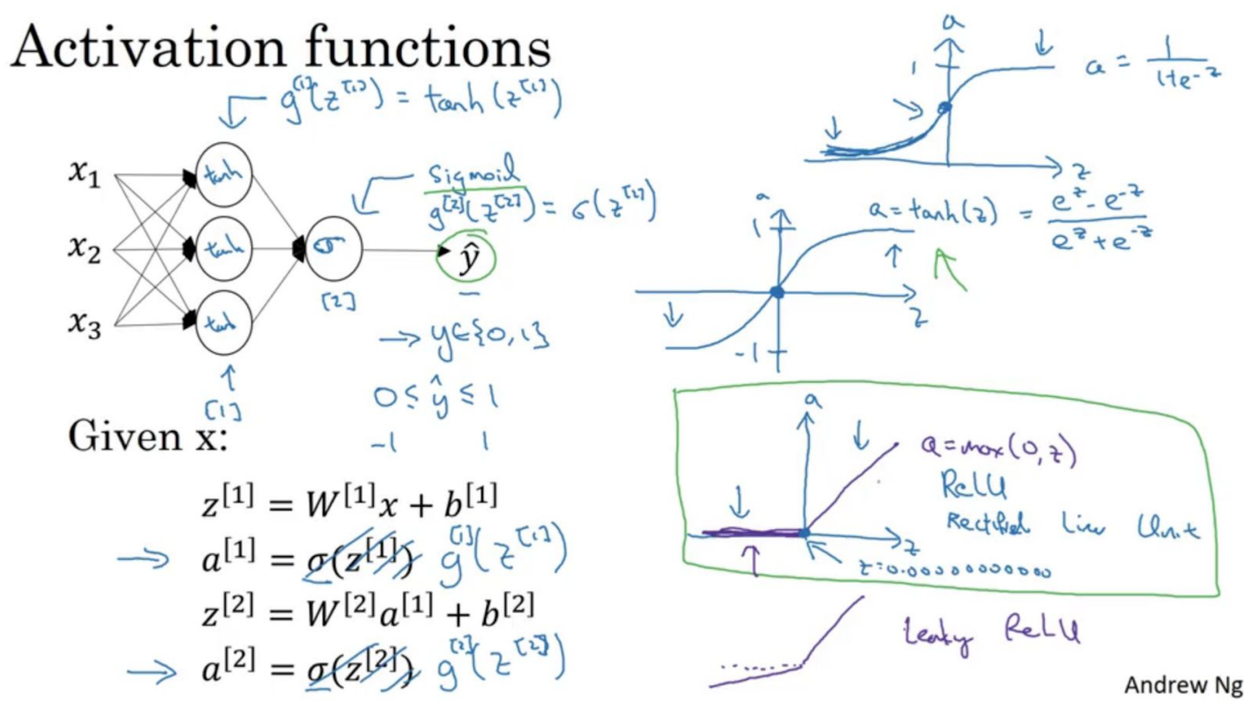
$\frac{d}{da} f(a) = \frac{d}{da} a^2 = \boxed{2a}$



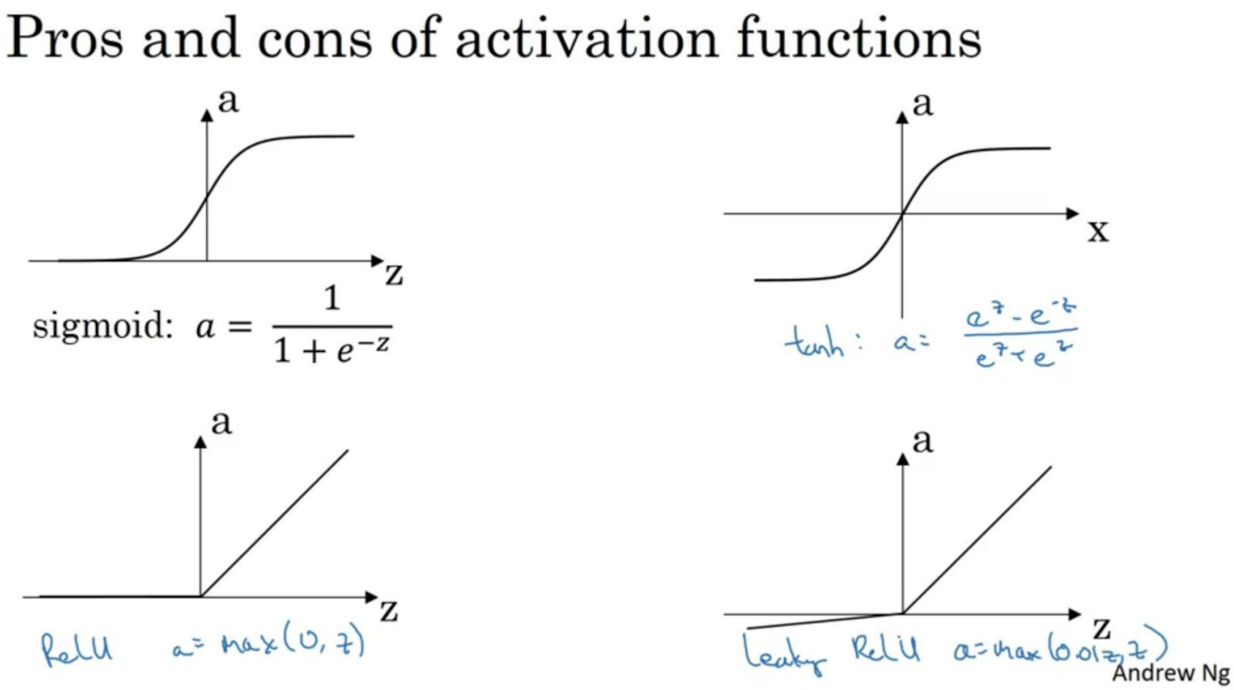
VECTORIZATION:
Avoid for loops and just use dot function -- this will allow better parallel use of MPU power.



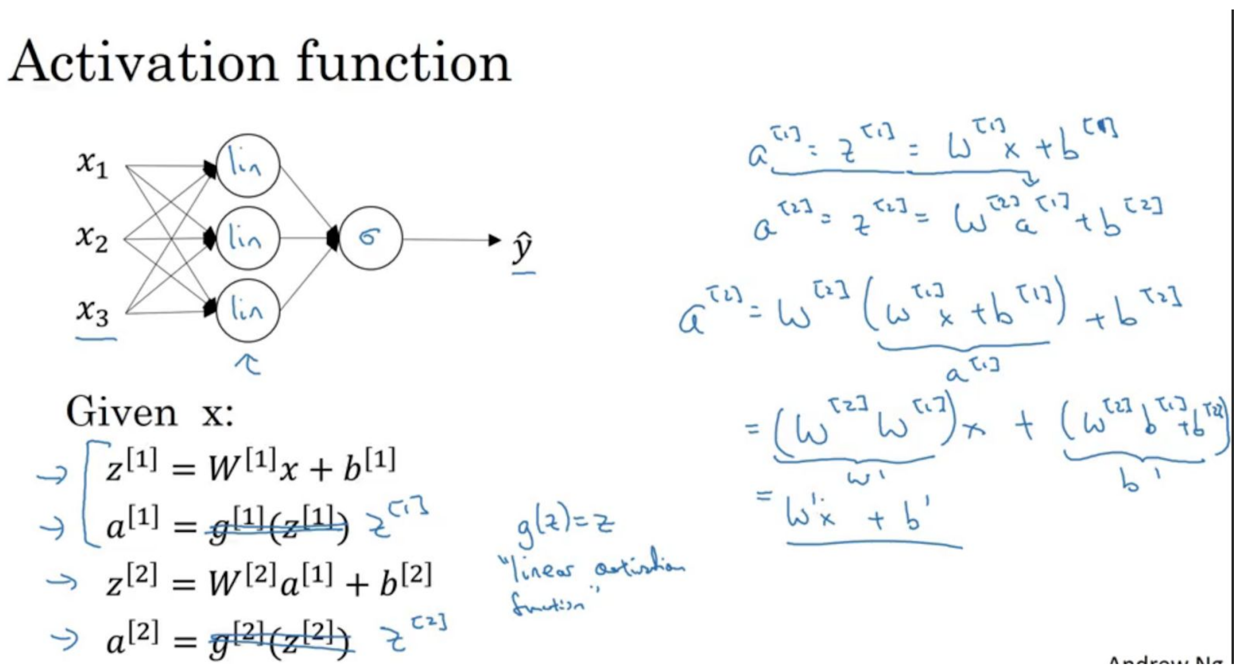


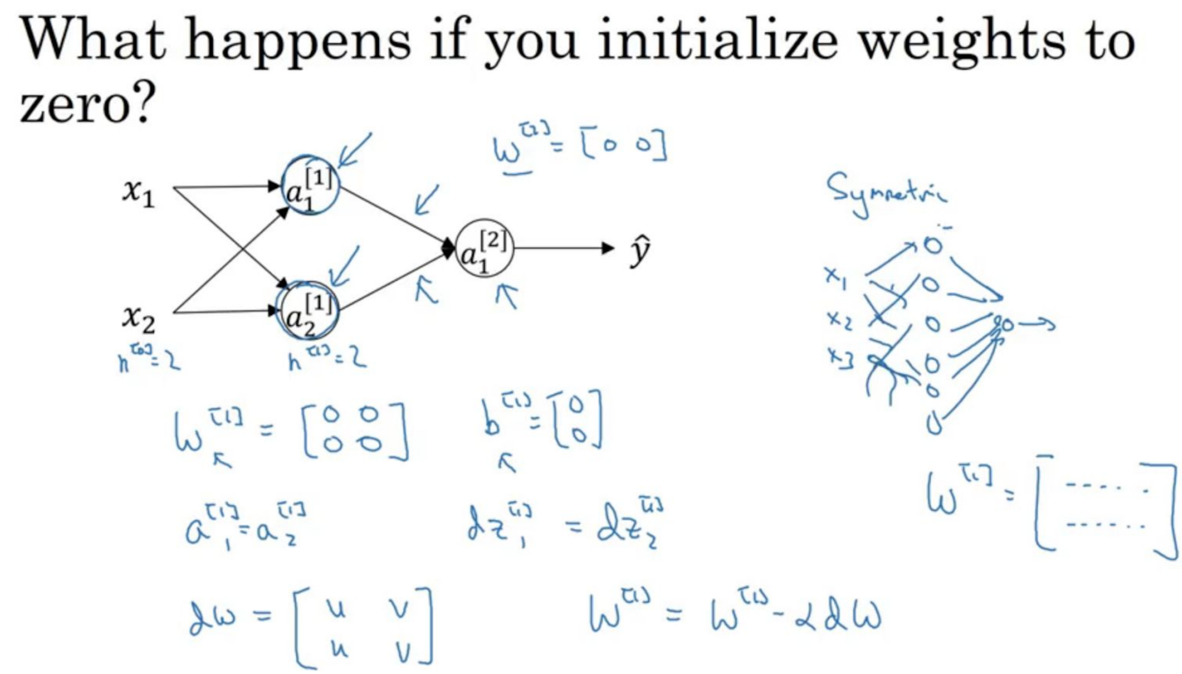


Relu is really good - no vanishing gradient.

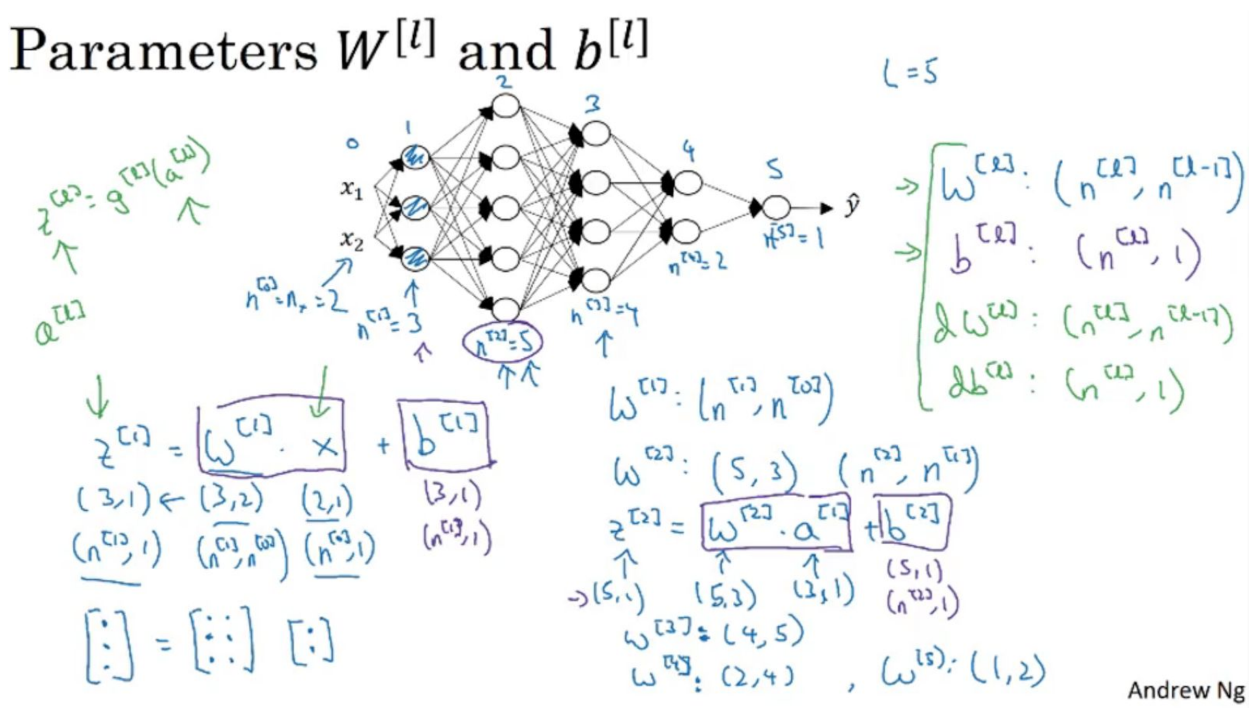


if only linear activations then its no better to use hidden layers, all linear and last sigmoid is just logistic regression. To learn interesting functions you need non linearity.
Exception: Use linear output neuron for regression problems but inner hidden layers are all sigmoid or tanh or relu

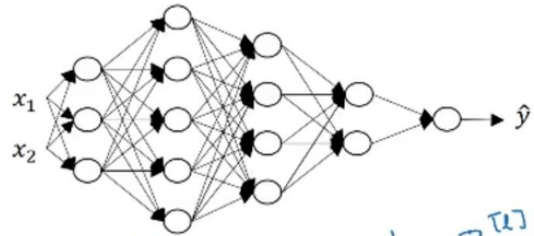




Very good overview of dimensions of W and b.



Vectorized implementation



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

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

$[z^{[1](1)} \; z^{[1](2)} \dots z^{[1](m)}]$

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

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

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

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

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

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

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

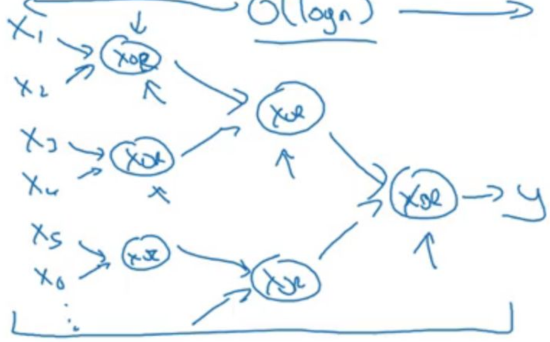
Andrew Ng

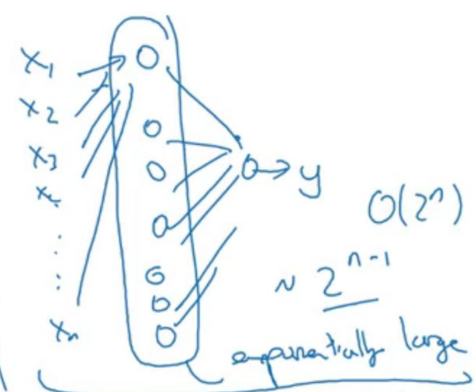
Circuit theory and deep learning

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.

$y = x_1 \text{ XOR } x_2 \text{ XOR } x_3 \text{ XOR} \dots \text{ XOR } x_n$

$O(\log n)$





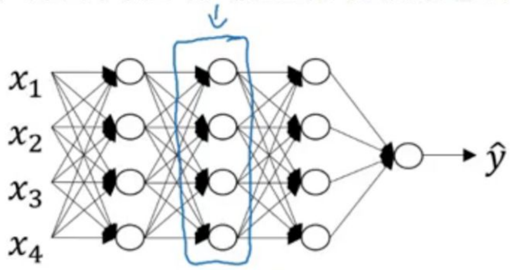
$O(2^n)$

$\sim 2^{n-1}$

exponentially large

Andrew Ng

Forward and backward functions



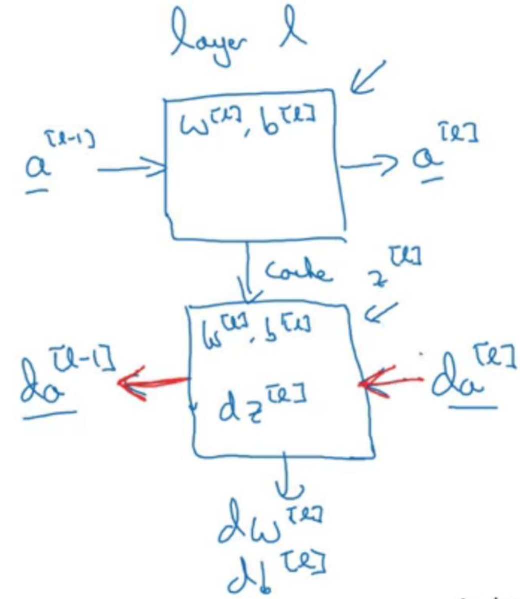
Layer l: $W^{[l]}, b^{[l]}$

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

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

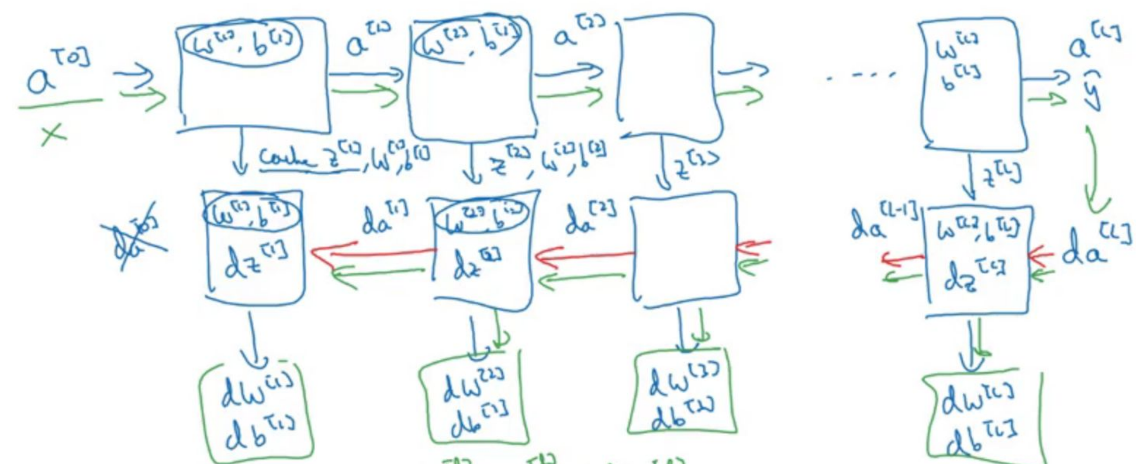
$a^{[l]} = g^{[l]}(z^{[l]})$

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



Andrew N

Forward and backward functions



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

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

Andrew Ng

What are hyperparameters?

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

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, ...

Forward and backward propagation

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = g^{[2]}(Z^{[2]})$$
$$\vdots$$
$$A^{[L]} = g^{[L]}(Z^{[L]}) = \hat{Y}$$

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