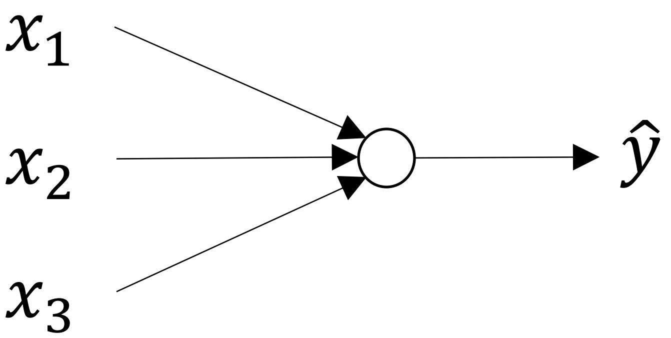
Artificial Intelligence and Machine Learning

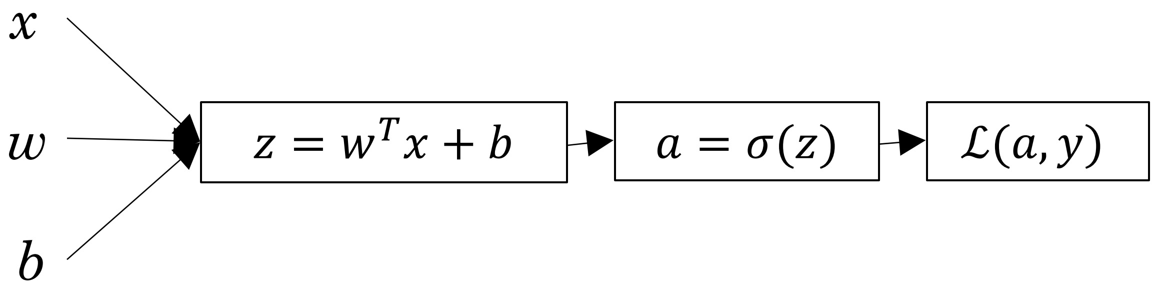
Neural Networks

Lecture Outline

- Logistic Regression Review
- Neural Networks
  - Forward pass
  - Backward pass

Review: Logistic Regression





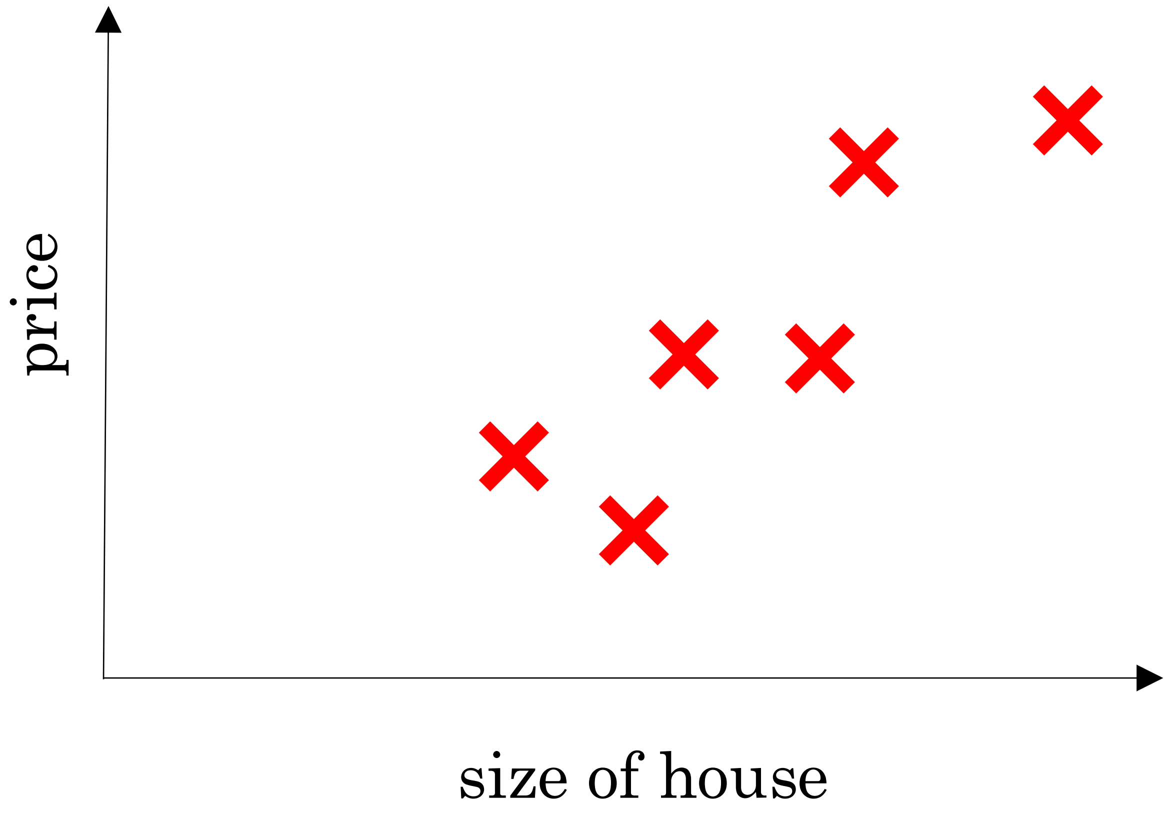
أكاديمية كاوست
KAUST ACADEMY

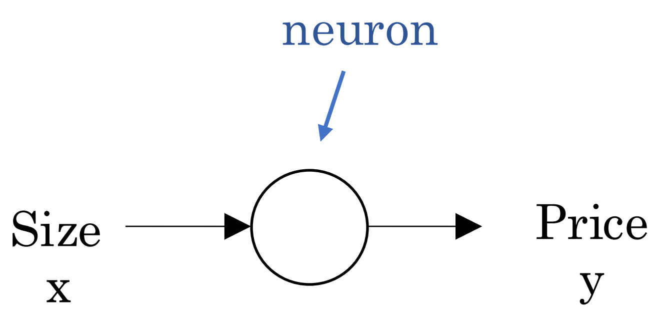
deeplearning.ai

Introduction to Deep Learning

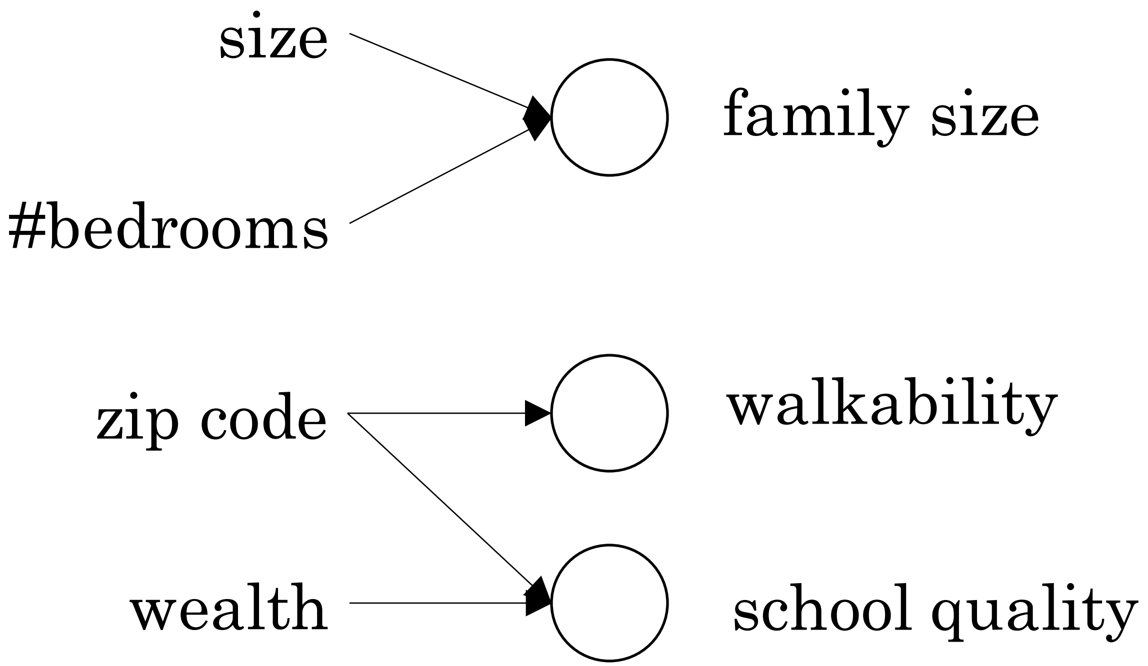
What is a Neural Network?

Housing Price Prediction

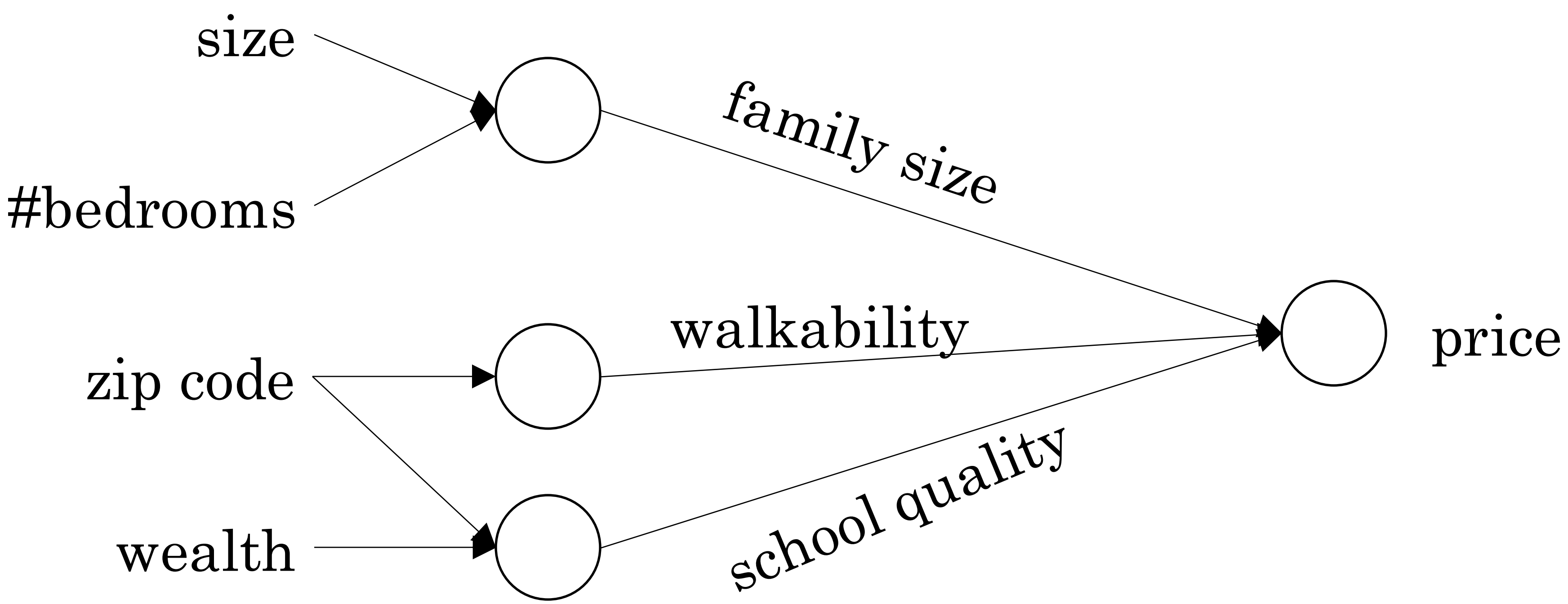




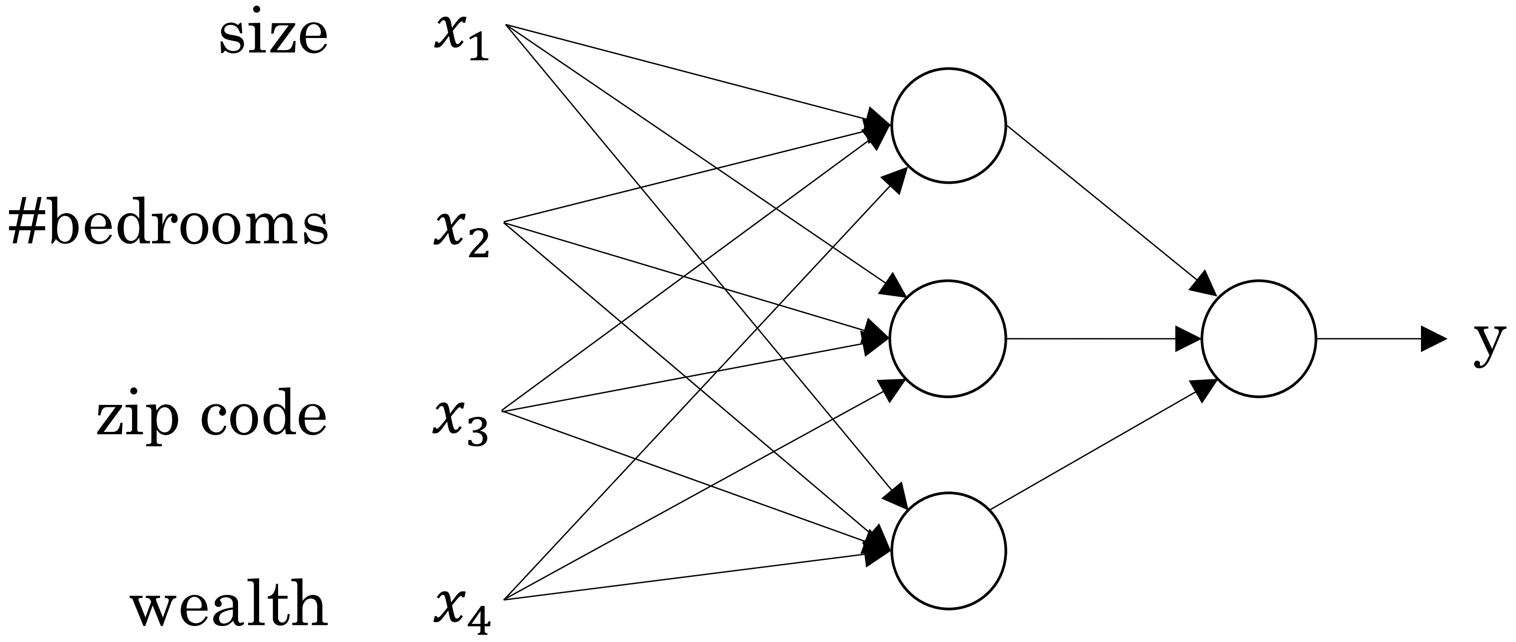
Housing Price Prediction



Housing Price Prediction



Housing Price Prediction



Introduction to Deep Learning

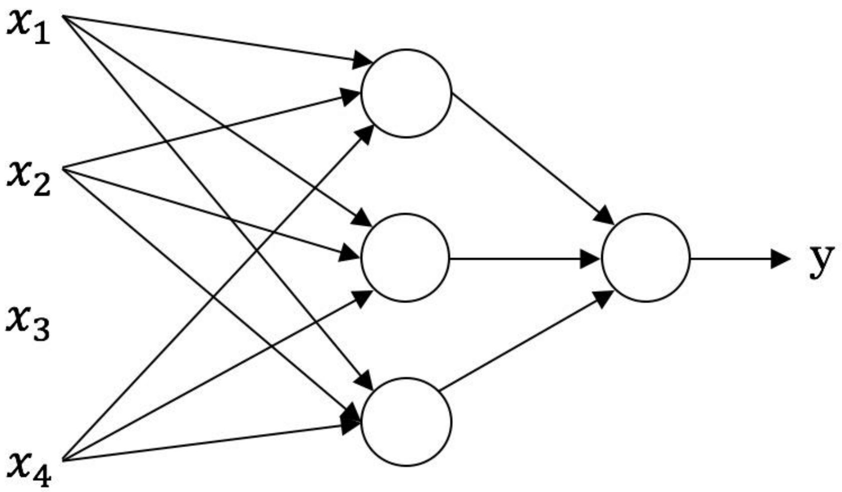
Supervised Learning with Neural Networks

deeplearning.ai

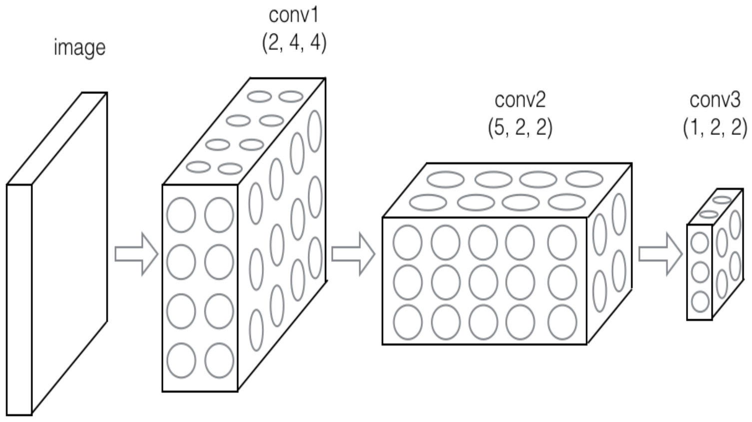
Supervised Learning

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

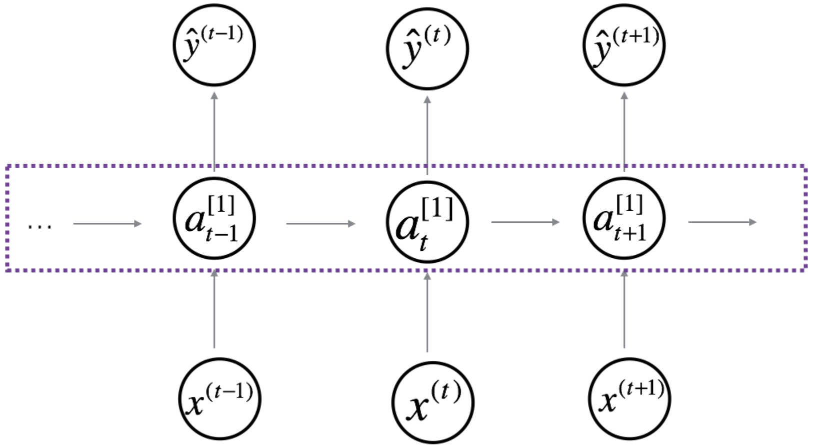
Neural Network examples



Standard NN



Convolutional NN



Recurrent NN

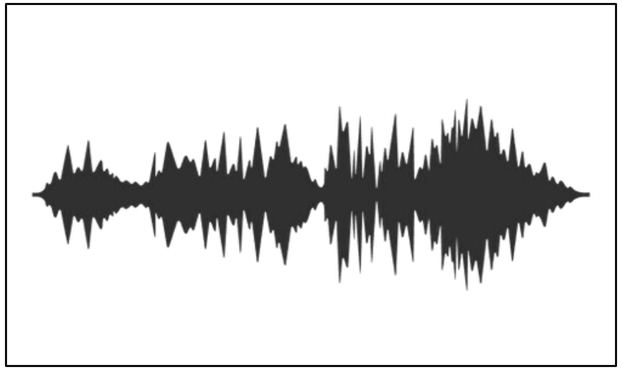
Supervised Learning

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 score and seven years ago

Text

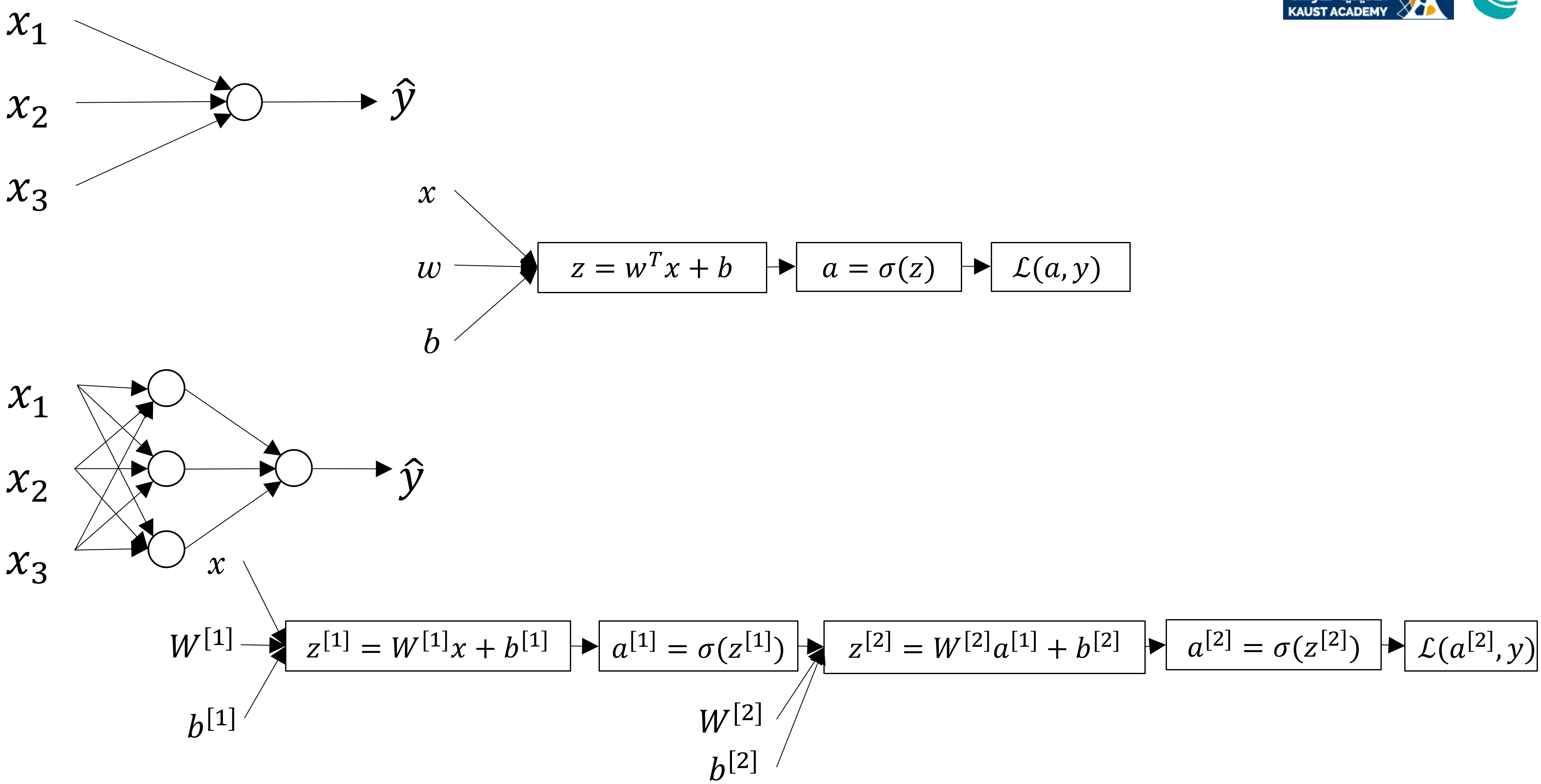
One hidden layer Neural Network

Neural Networks Overview

deeplearning.ai

أكاديمية كاوست
KAUST ACADEMY

What is a Neural Network?



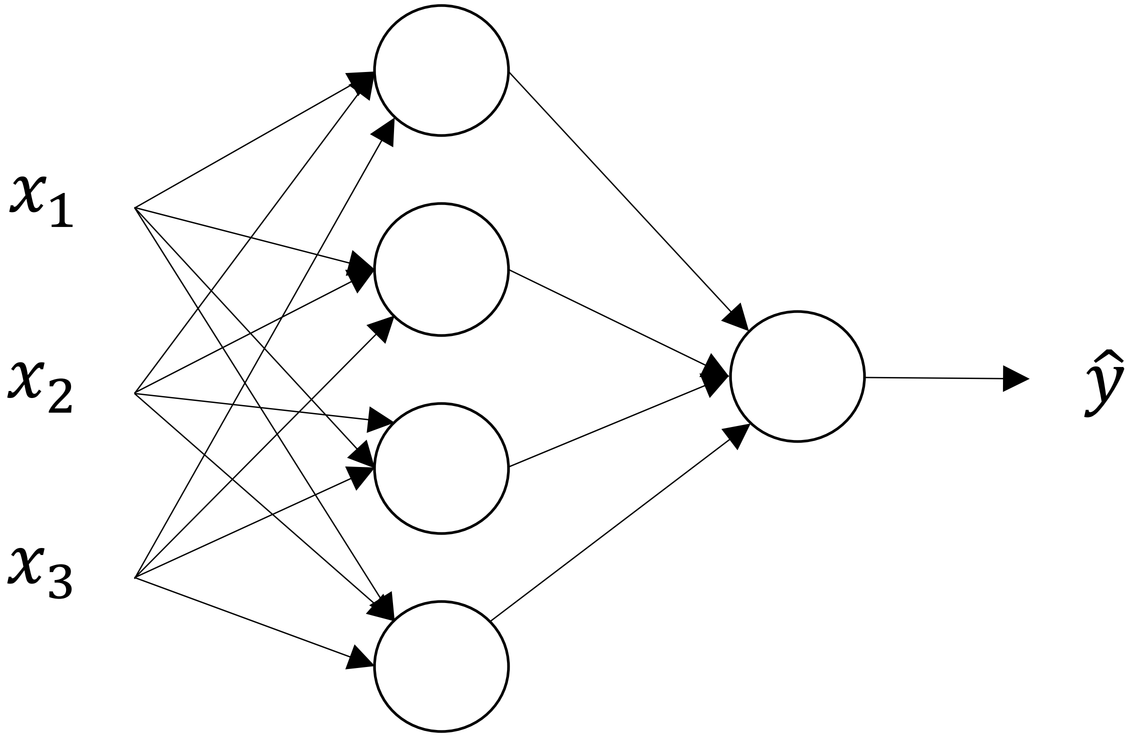
أكاديمية كاوست
KAUST ACADEMY

deeplearning.ai

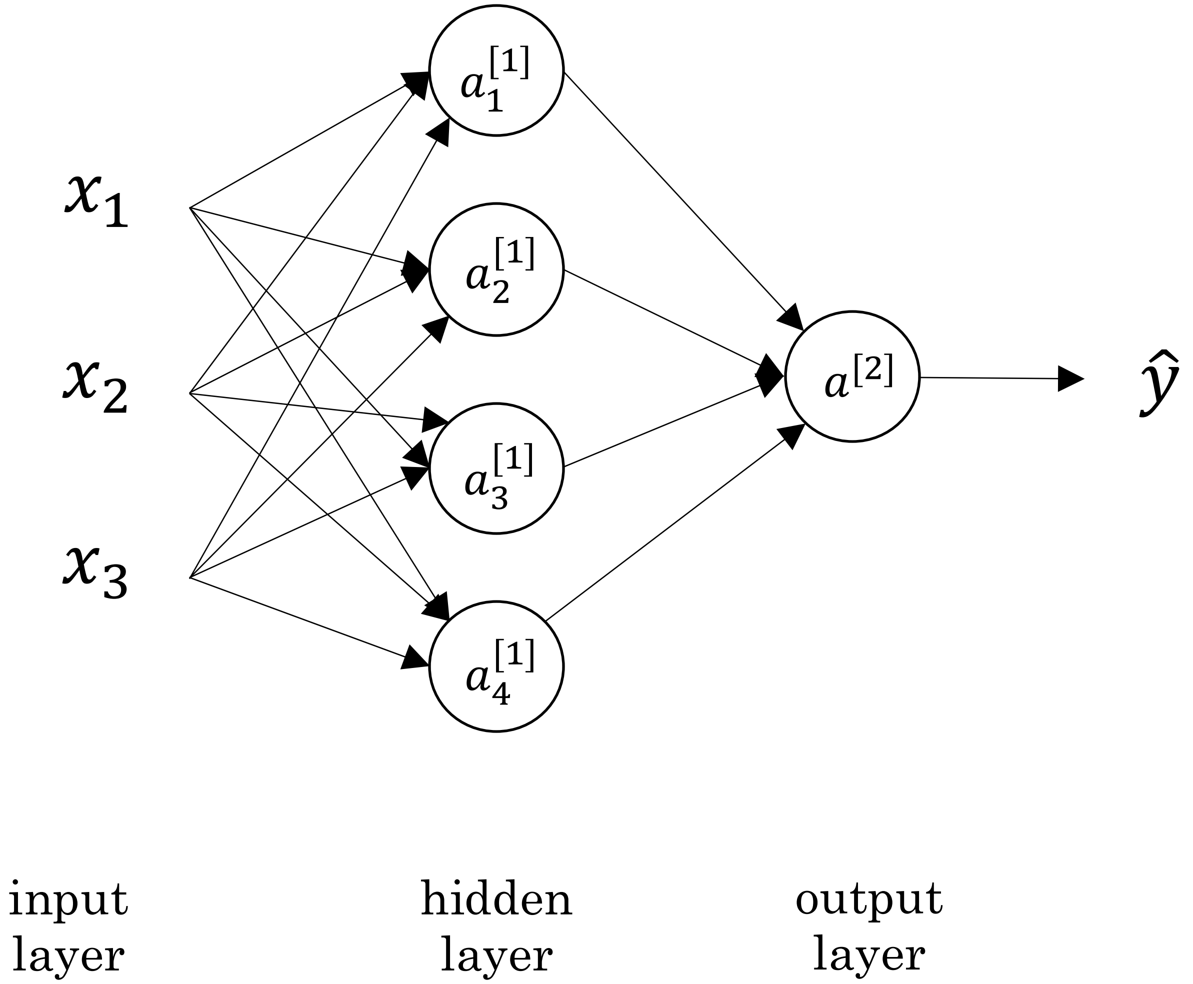
One hidden layer Neural Network

Neural Network Representation

Neural Network Representation



Neural Network Representation



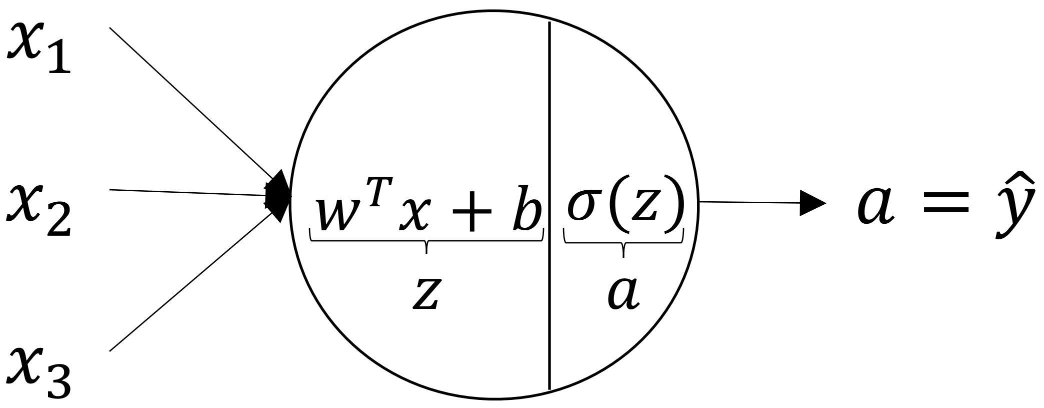
أكاديمية كاوست
KAUST ACADEMY

deeplearning.ai

One hidden layer Neural Network

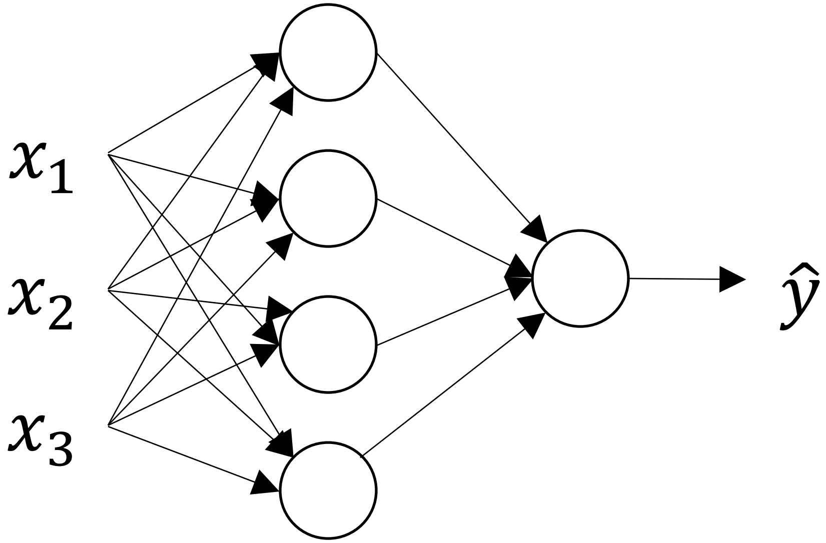
Computing a Neural Network's Output

Neural Network Representation

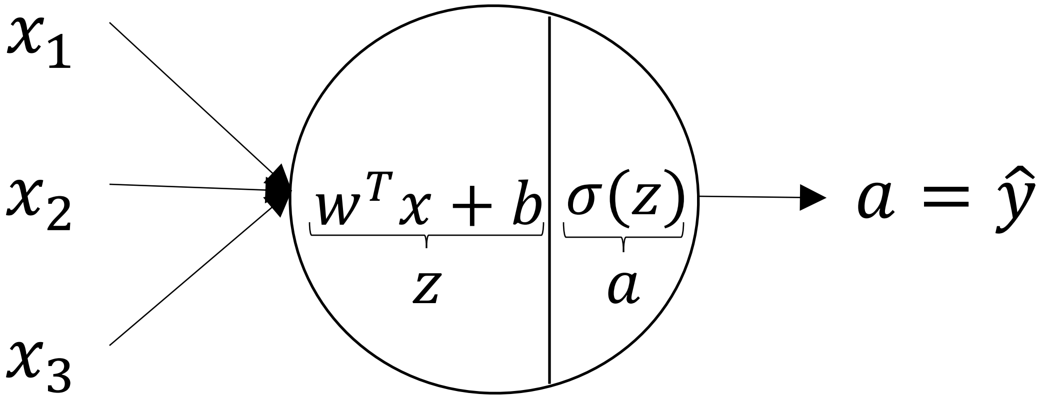


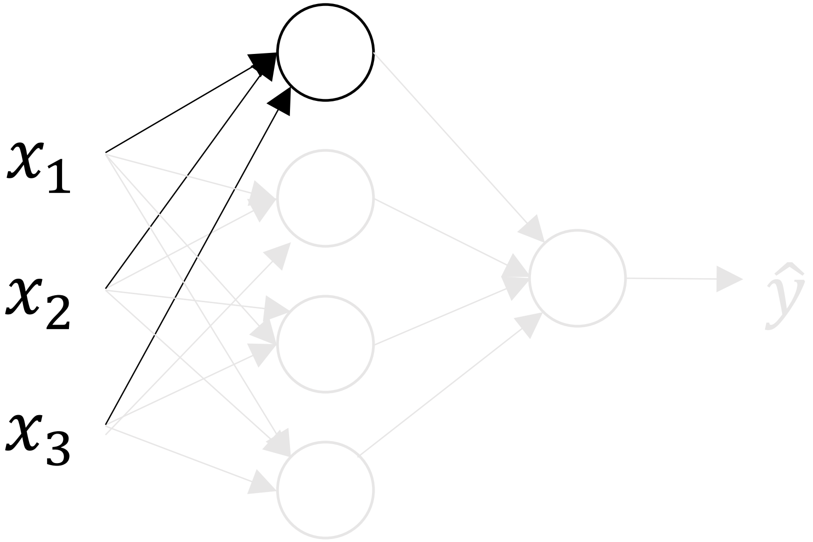
$z = w^T x + b$

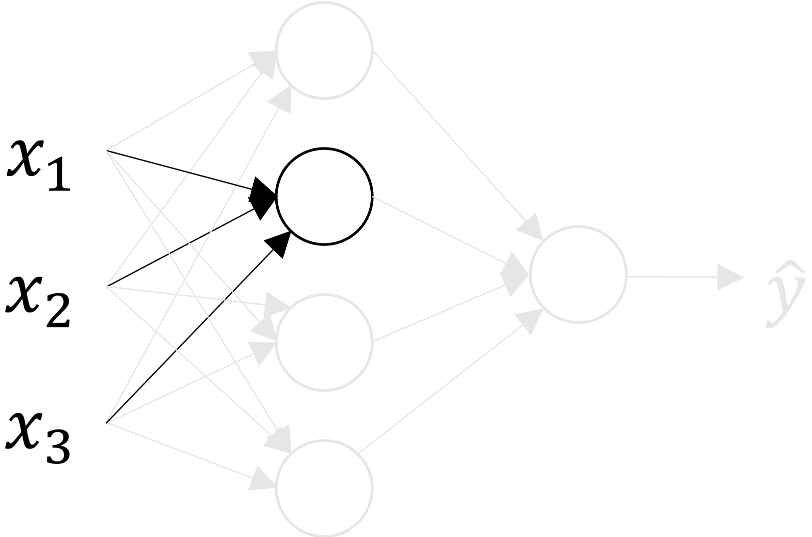
$a = \sigma(z)$



Neural Network Representation



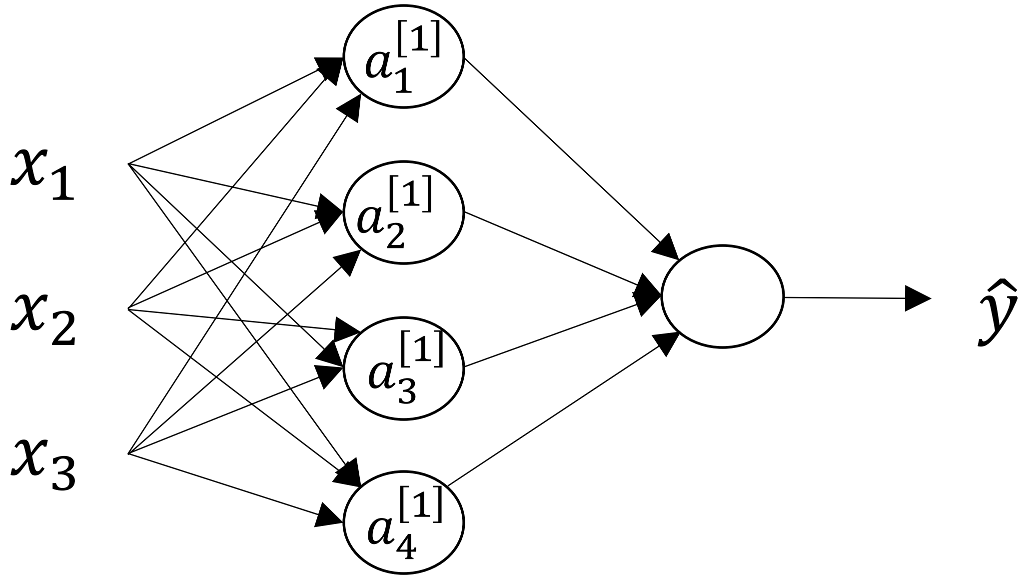




$$z = w^T x + b$$

$$a = \sigma(z)$$

Neural Network Representation



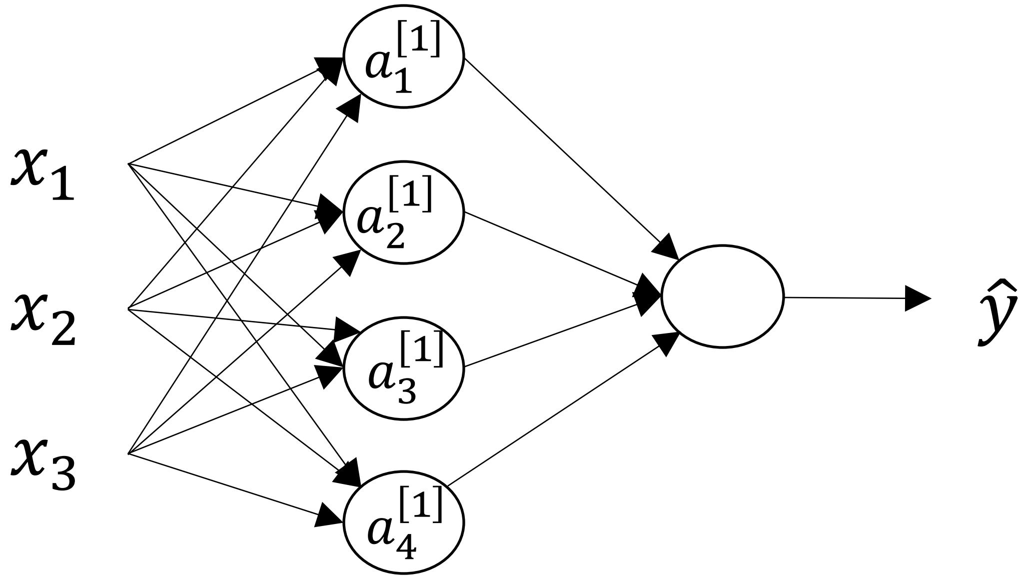
$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, \ a_1^{[1]} = \sigma(z_1^{[1]})$$

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

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

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

Neural Network Representation learning



Given input x:

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

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

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

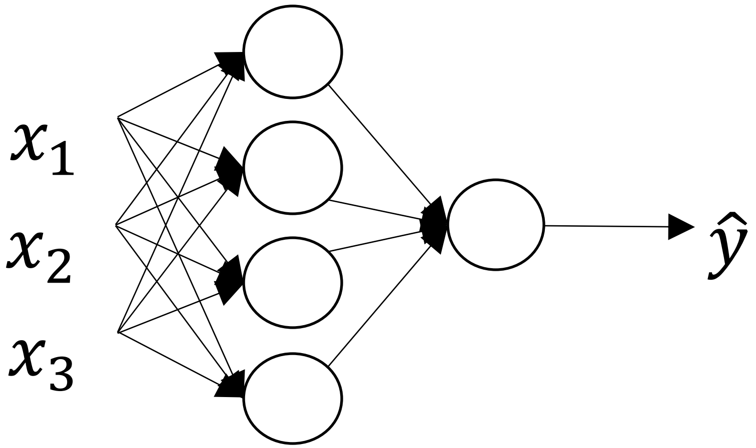
$$a^{[2]} = \sigma(z^{[2]})$$

One hidden layer Neural Network

deeplearning.ai

Vectorizing across multiple examples

Vectorizing across multiple examples

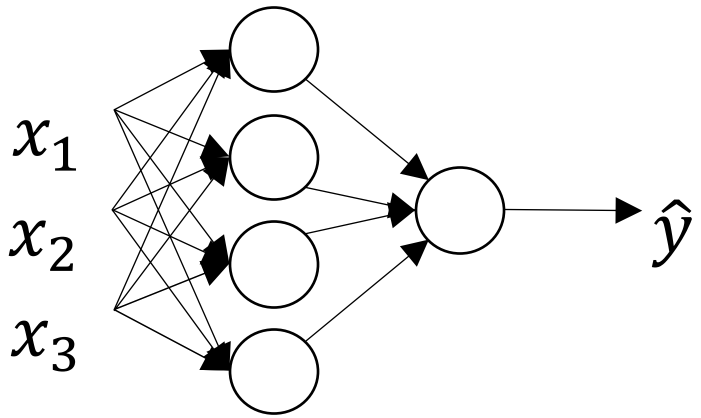


$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = \sigma(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = \sigma(z^{[2]})$$

```
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
```

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

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

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

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

One hidden layer Neural Network

deeplearning.ai

Explanation for vectorized implementation

Justification for vectorized implementation

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

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

Justification for vectorized implementation

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

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

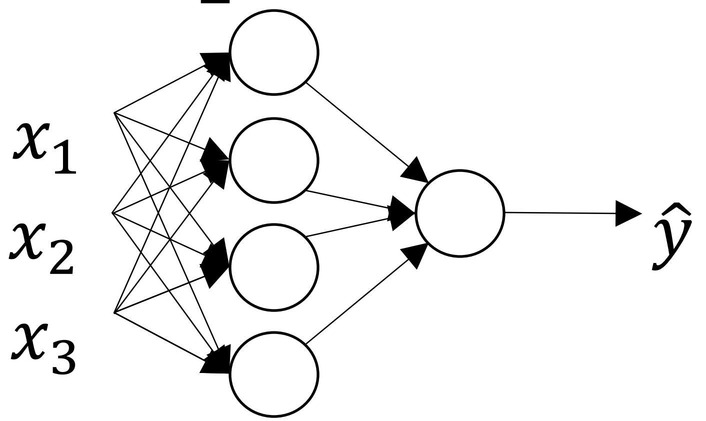
$$W^{[1]}\underbrace{\begin{bmatrix} | & | & | & \\ x^{(1)} & x^{(2)} & x^{(3)} & \cdots \\ | & | & | & \end{bmatrix}}_{X} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} = \begin{bmatrix} | & | & | & \\ z^{[1](1)} & z^{[1](2)}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

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

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

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

Recap of vectorizing across multiple examples




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

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

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

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

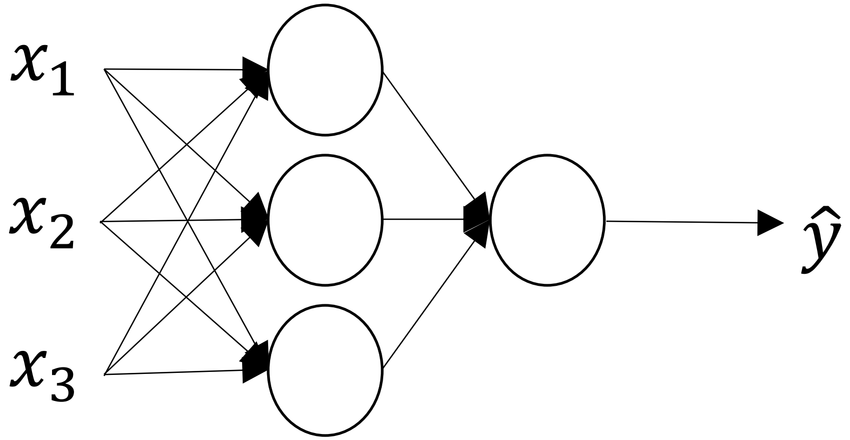
$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = \sigma(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = \sigma(Z^{[2]})$$

One hidden layer Neural Network

Activation functions

deeplearning.ai

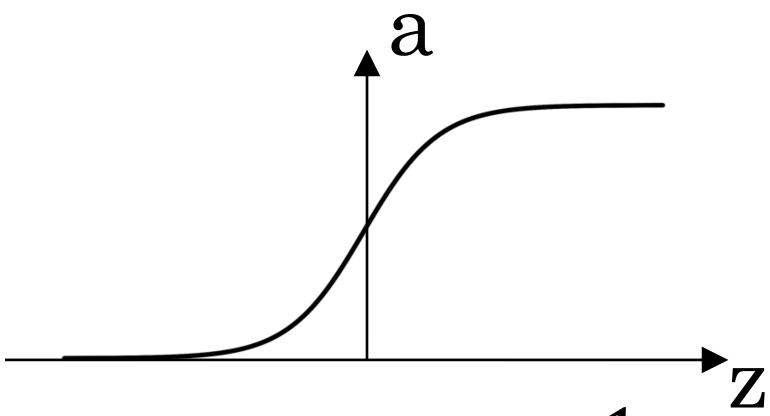
Activation functions



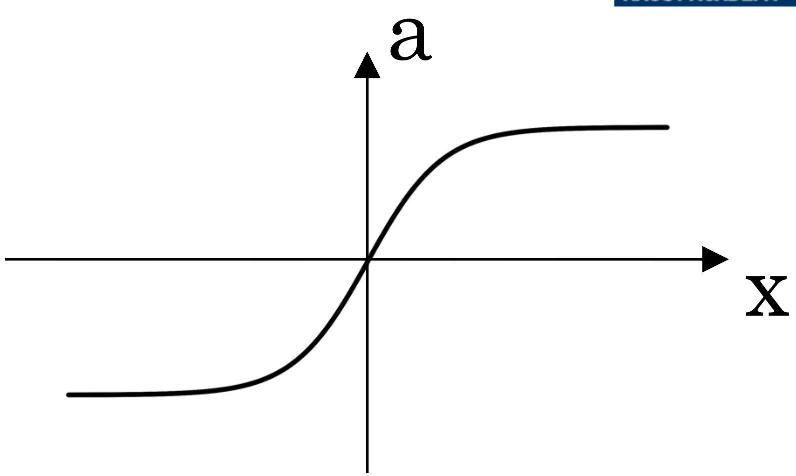
Given x:

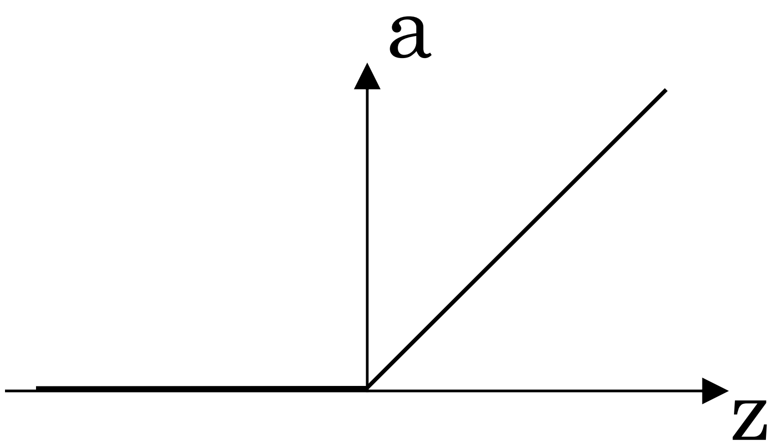
$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = \sigma(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = \sigma(z^{[2]})$$

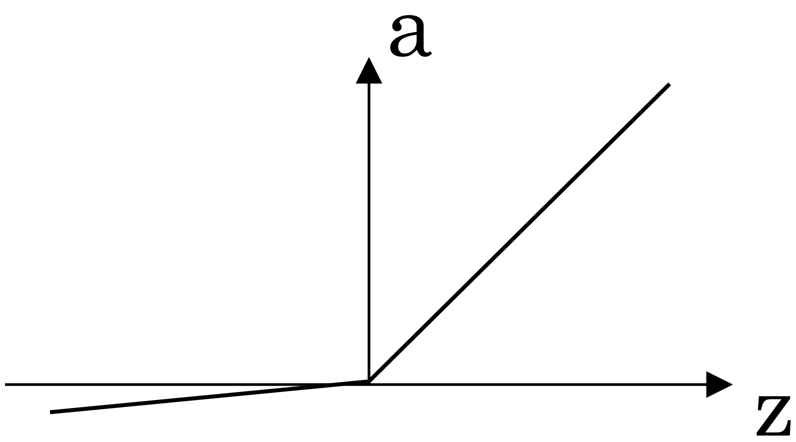
Pros and cons of activation functions



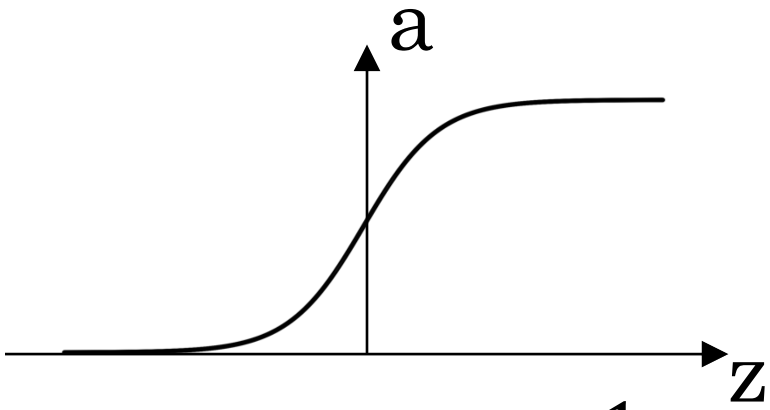
sigmoid: $a = \dfrac{1}{1 + e^{-z}}$



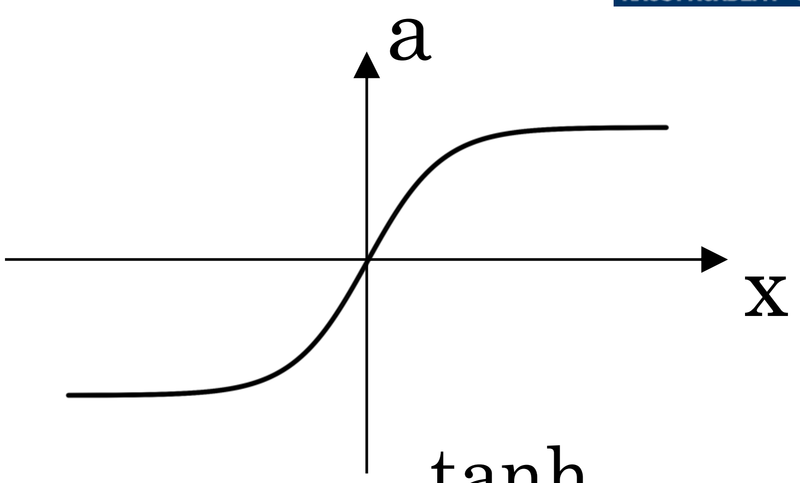




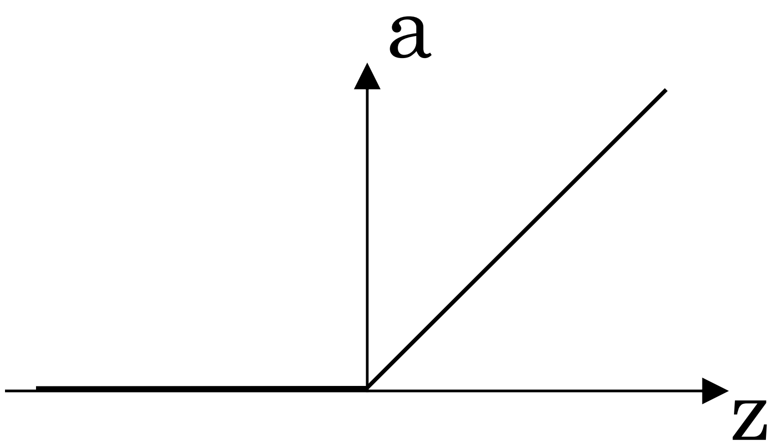
Pros and cons of activation functions



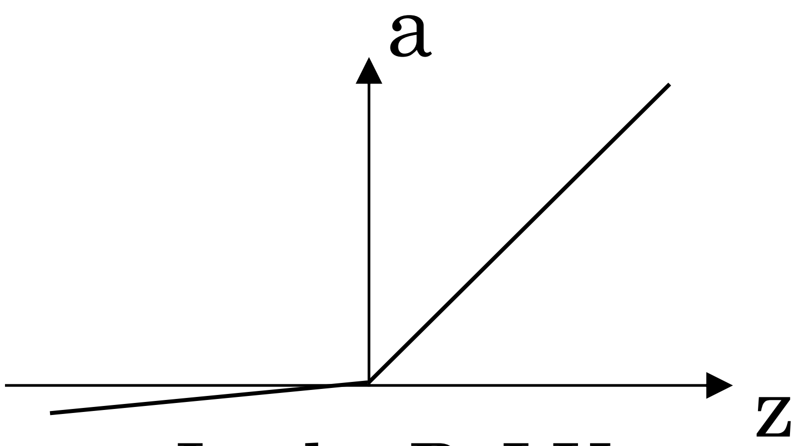
sigmoid: $a = \dfrac{1}{1+e^{-z}}$



tanh



ReLU



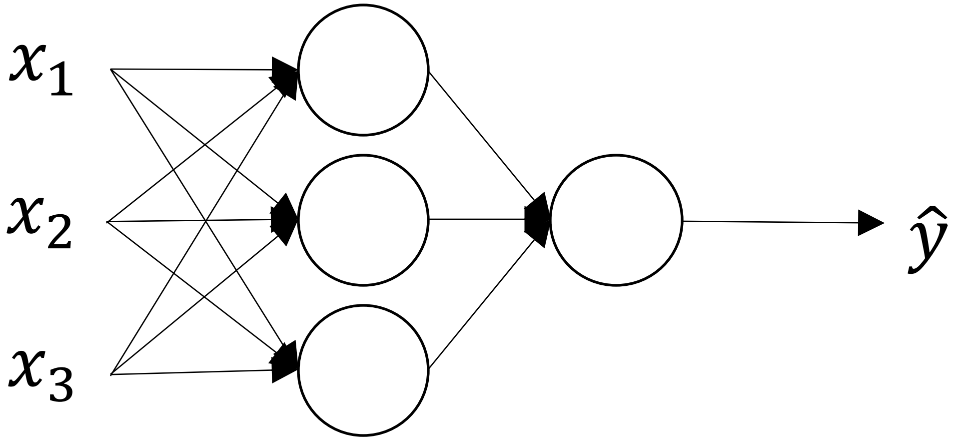
Leaky ReLU

One hidden layer Neural Network

deeplearning.ai

Why do you need non-linear activation functions?

Activation function



Given x:

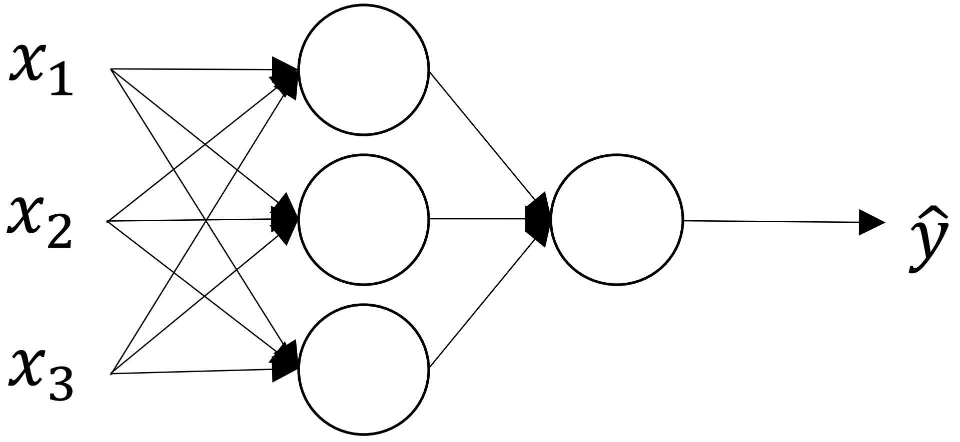
$$z^{[1]} = W^{[1]}x + b^{[1]}$$

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

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

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

Activation function



Given x:

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

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

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

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

One hidden layer Neural Network

Gradient descent for neural networks

deeplearning.ai

Gradient descent for neural networks

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

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

Repeat {

Compute predictions: $\left(\hat{y}^{(i)}, i = 1, \ldots m\right)$

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

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

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

… }

Formulas for computing derivatives

Forward propagation

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

Back propagation

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

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

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

$$dZ^{[1]} = W^{[2]T} dZ^{[2]} * g^{[1]\prime}(\mathrm{Z}^{[1]})$$

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

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

One hidden layer Neural Network

Backpropagation intuition

deeplearning.ai

Computing gradients

Logistic regression

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

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

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

$$dz = da \,.\, g'(z)$$

Computing gradients

Logistic regression

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

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

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

$$dz = da \,.\, g'(z)$$

to do: gradient of $g(z) = \frac{1}{1+e^{-z}}$

Computing gradients

Logistic regression

$$
\begin{tikzcd}
x \arrow[dr] & & & \\
w \arrow[r] & \boxed{z = w^T x + b} \arrow[r] & \boxed{a = \sigma(z)} \arrow[r] & \boxed{\mathcal{L}(a, y)} \\
b \arrow[ur] & & &
\end{tikzcd}
$$

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

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

$$dz = da \,.\, g'(z)$$

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

Andrew Ng

Computing gradients

Logistic regression

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

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

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

$$dz = da \,.\, g'(z) = a - y$$

Computing gradients

Logistic regression

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

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

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

$$dz = da\,.\,g'(z) = a - y$$

$$dw = dz\,.\,x$$

$$db = dz$$

Andrew Ng

Neural network gradients

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

$$W^{[2]},\ b^{[2]} \rightarrow \boxed{z^{[2]} = W^{[2]}x + b^{[2]}}$$

Andrew Ng

Neural network gradients

$$
\begin{tikzcd}
 & & & W^{[2]} \arrow[d] & & \\
 & & & b^{[2]} \arrow[d] & & \\
x \arrow[r] & z^{[1]} = W^{[1]}x + b^{[1]} \arrow[r] & a^{[1]} = \sigma(z^{[1]}) \arrow[r] & z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \arrow[r] & a^{[2]} = \sigma(z^{[2]}) \arrow[r] & \mathcal{L}(a^{[2]}, y) \\
W^{[1]} \arrow[ru] & & & & & \\
b^{[1]} \arrow[ruu] & & & & &
\end{tikzcd}
$$

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

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

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

Neural network gradients

$$\begin{tikzcd}
 & & & W^{[2]},\ b^{[2]} \arrow[d] & & \\
x,\ W^{[1]},\ b^{[1]} \arrow[r] & \boxed{z^{[1]} = W^{[1]}x + b^{[1]}} \arrow[r] & \boxed{a^{[1]} = \sigma(z^{[1]})} \arrow[r] & \boxed{z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}} \arrow[r] & \boxed{a^{[2]} = \sigma(z^{[2]})} \arrow[r] & \boxed{\mathcal{L}(a^{[2]}, y)}
\end{tikzcd}$$

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

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

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

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

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

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

Andrew Ng

Summary of gradient descent

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

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

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

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

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

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

Summary of gradient descent

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

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

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

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

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

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

Vectorization implementation

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

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

Summary of gradient descent

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

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

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

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

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

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

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

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

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

$$dZ^{[1]} = W^{[2]T} dZ^{[2]} * g^{[1]\prime}(\mathrm{Z}^{[1]})$$

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

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

Andrew Ng

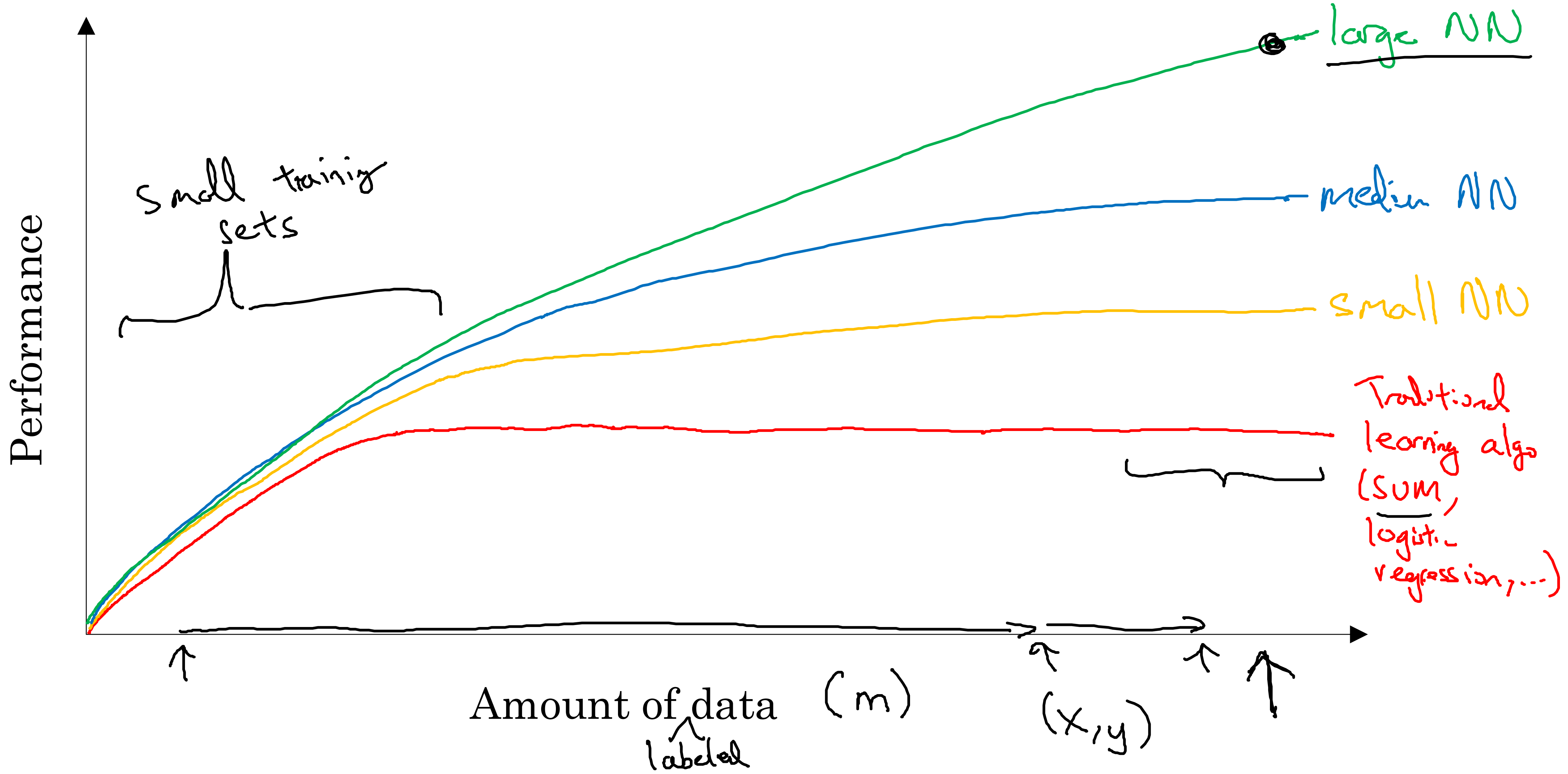
أكاديمية كاوست
KAUST ACADEMY

deeplearning.ai

Introduction to Neural Networks

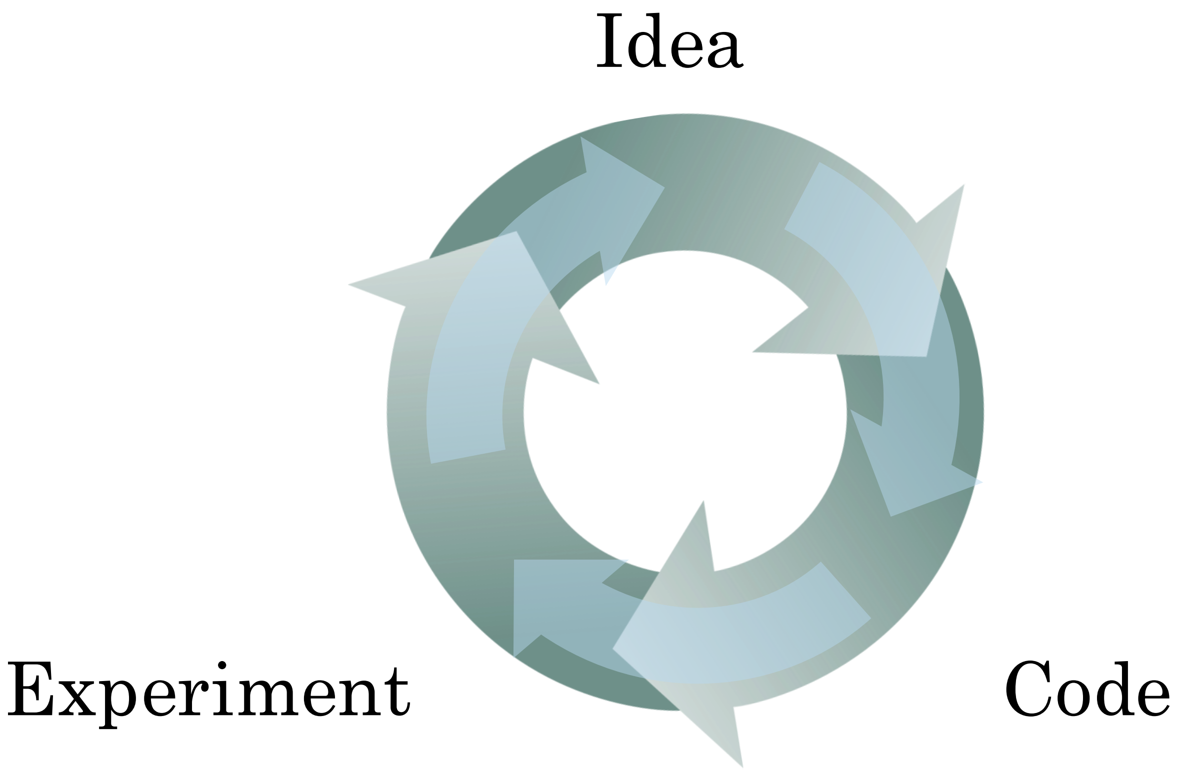
Why is Deep Learning taking off?

Scale drives deep learning progress



Scale drives deep learning progress

- Data
- Computation
- Algorithms

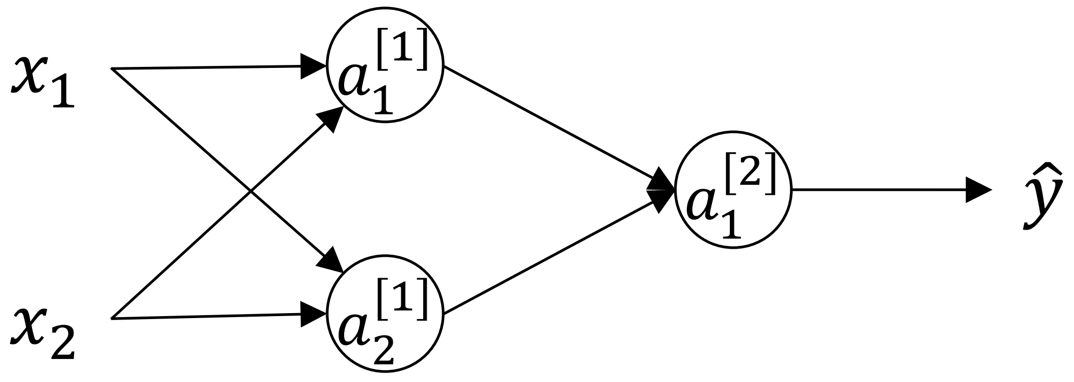


One hidden layer Neural Network

Random Initialization

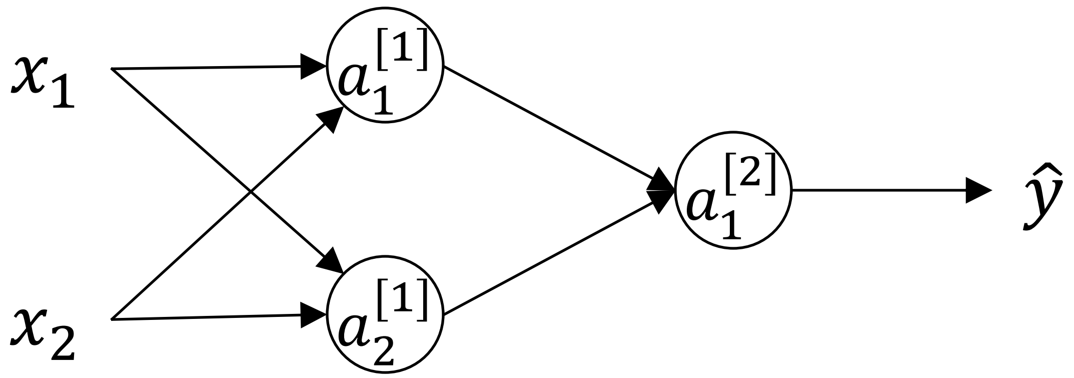
deeplearning.ai

What happens if you initialize weights to zero?



Initial weights = 0 → symmetry → similar updates

Random initialization



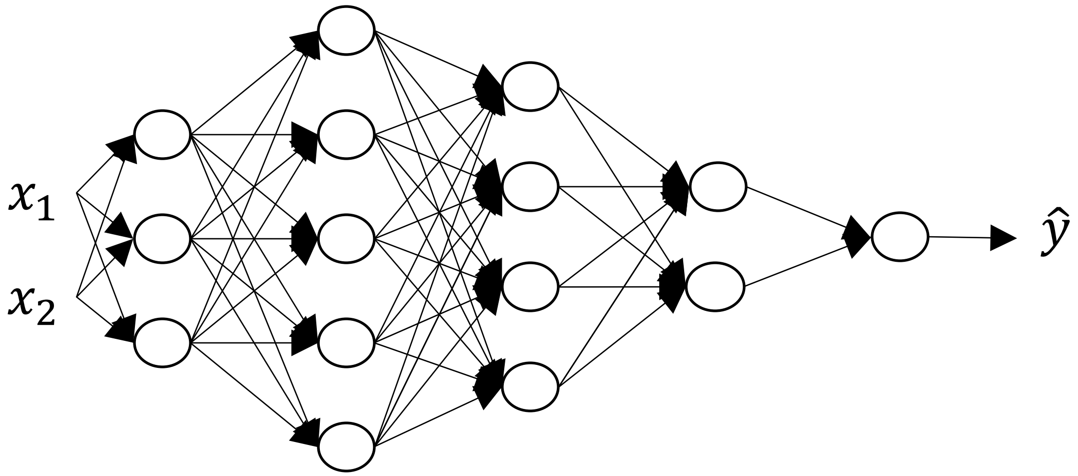
small values for $W^{[1]}$ and $W^{[2]}$

Deep Neural Networks

Getting your matrix dimensions right

deeplearning.ai

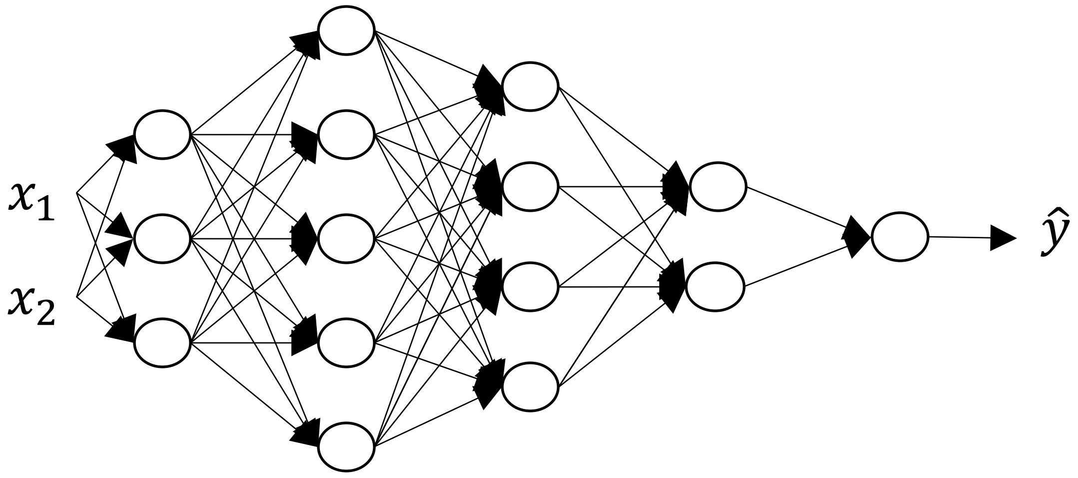
Parameters $W^{[l]}$ and $b^{[l]}$



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

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

Vectorized implementation



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

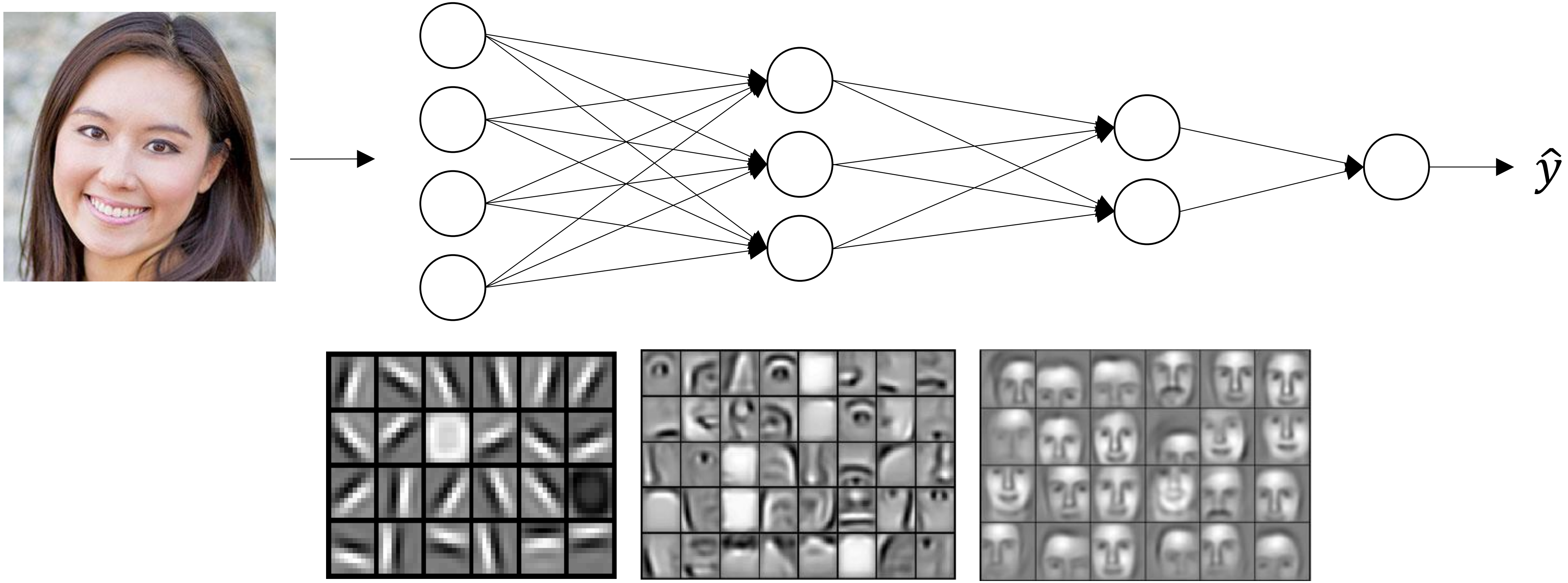
أكاديمية كاوست
KAUST ACADEMY

Deep Neural Networks

deeplearning.ai

Why deep representations?

Intuition about deep representation



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.

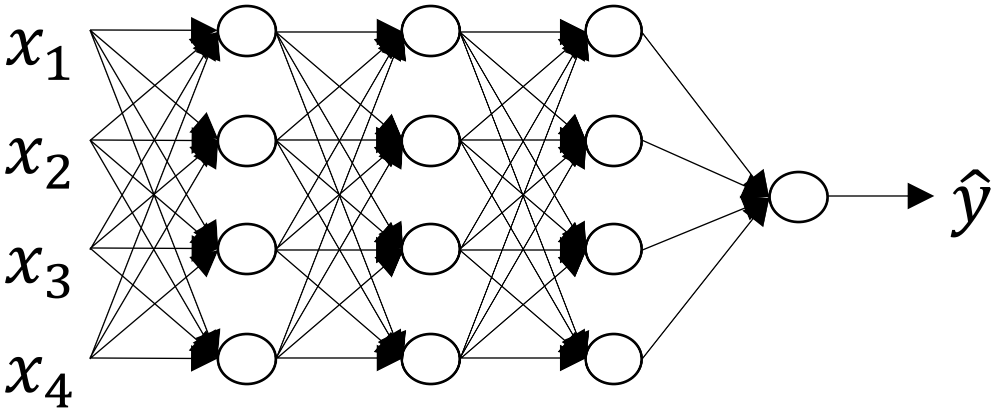
Example: xor

Deep Neural Networks

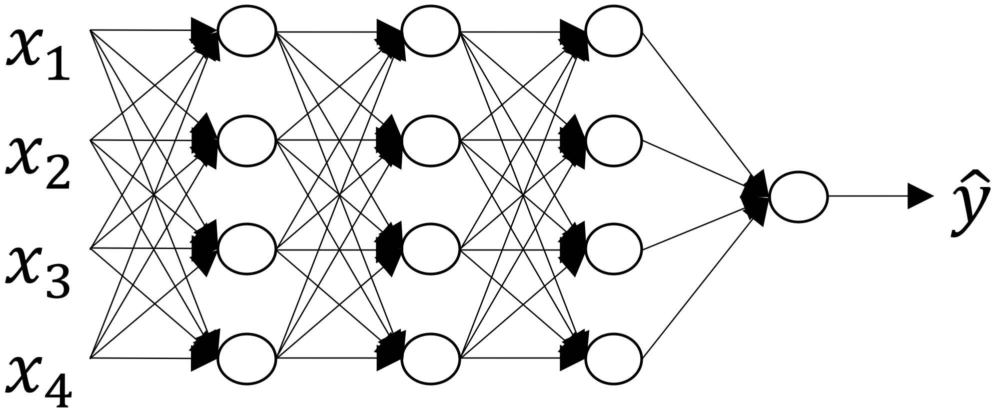
deeplearning.ai

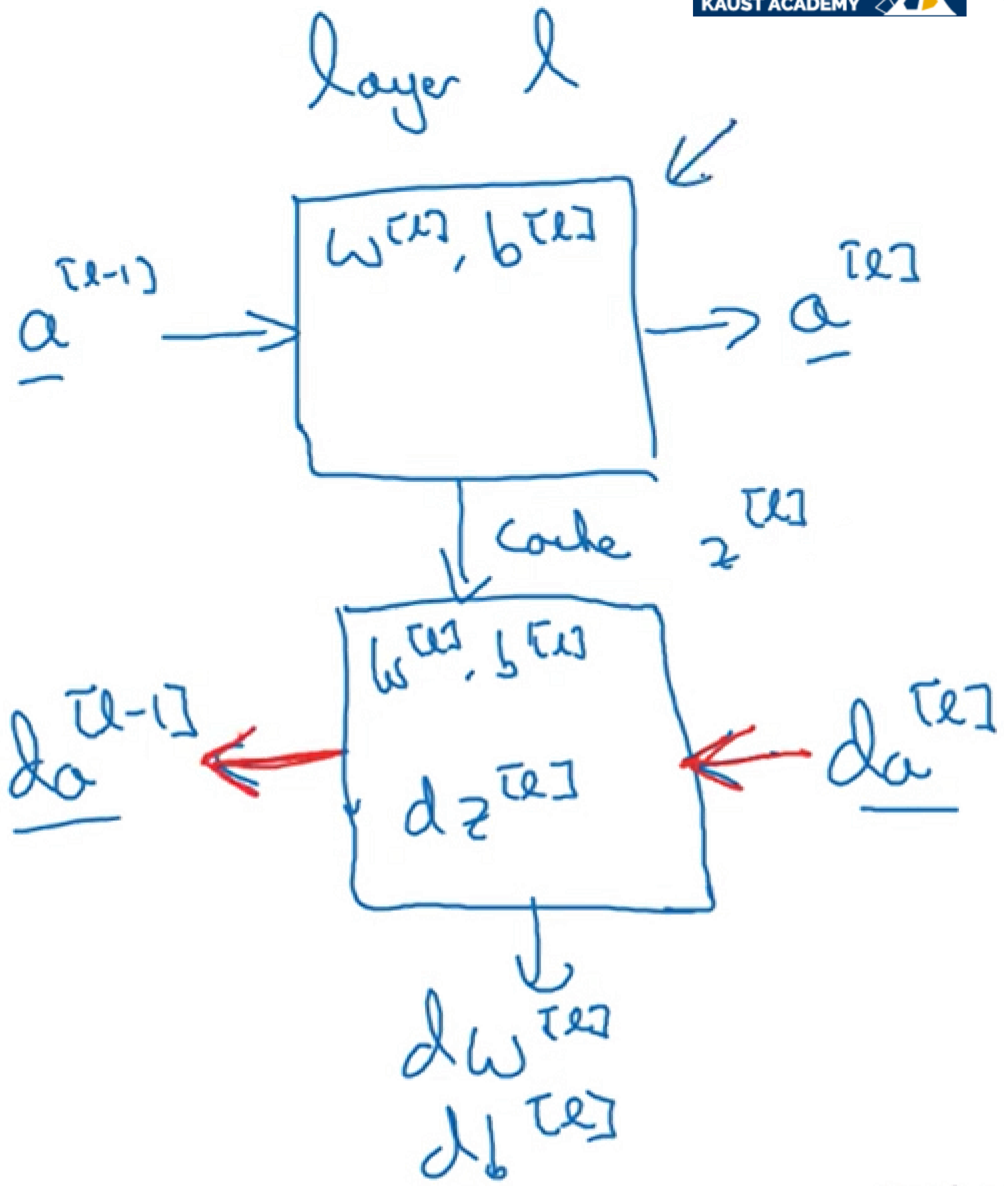
Building blocks of deep neural networks

Forward and backward functions



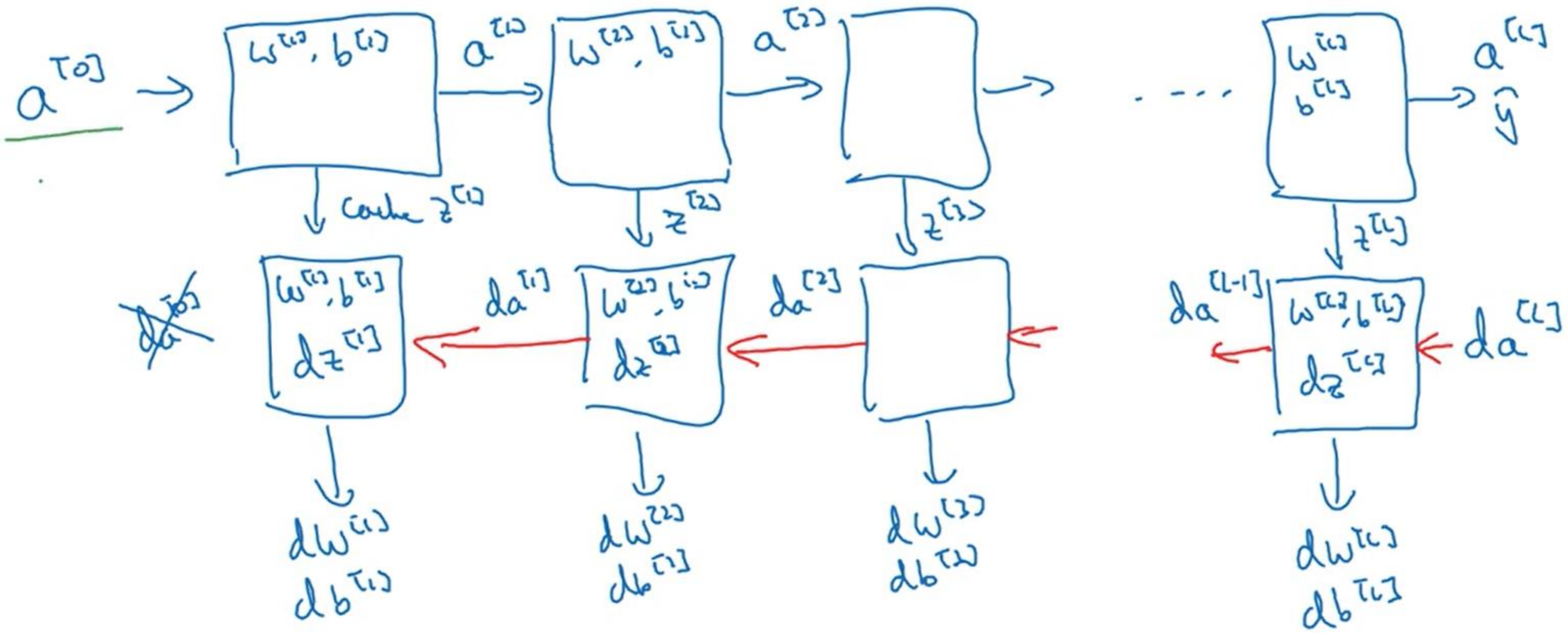
Forward and backward functions





Andrew Ng

Forward and backward functions



Andrew Ng

Deep Neural Networks

Forward and backward propagation

deeplearning.ai

أكاديمية كاوست
KAUST ACADEMY

Forward propagation for layer *l*

Input $a^{[l-1]}$

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

Forward propagation for layer *l*

Input $a^{[l-1]}$

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

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

Backward propagation for layer *l*

Input $da^{[l]}$

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

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

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

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

Previously

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

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

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

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