



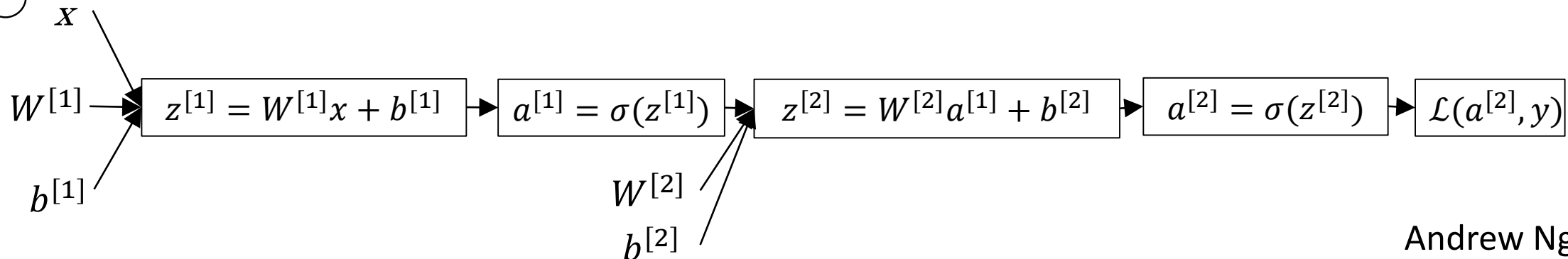
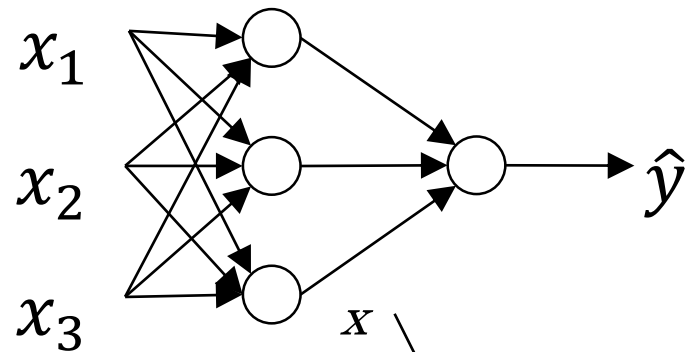
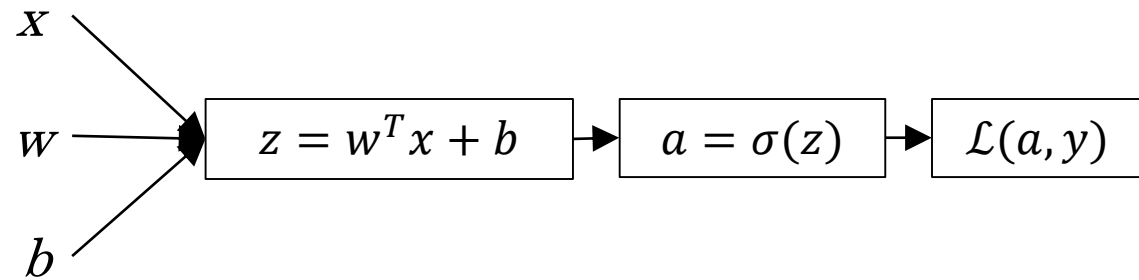
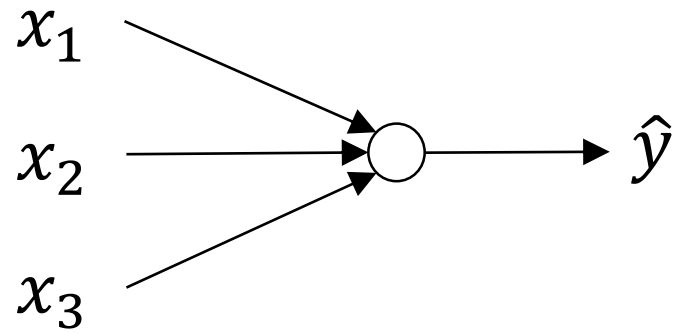
deeplearning.ai

One hidden layer  
Neural Network

---

# Neural Networks Overview

# What is a Neural Network?





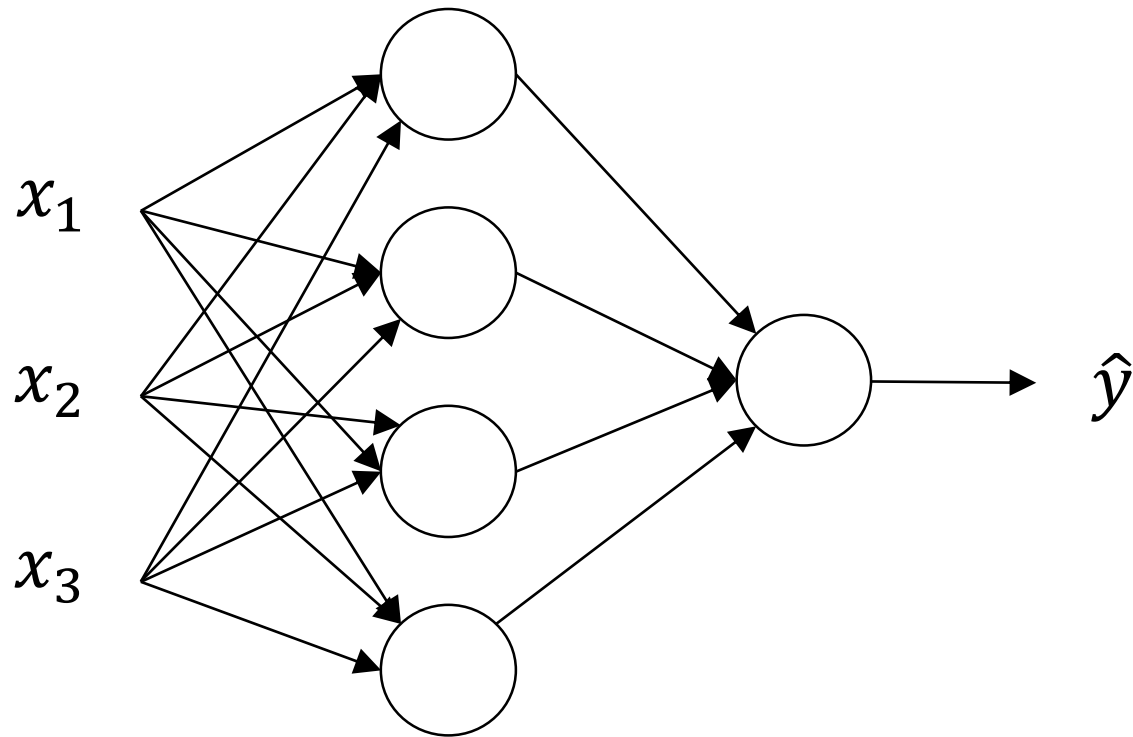
deeplearning.ai

One hidden layer  
Neural Network

---

Neural Network  
Representation

# Neural Network Representation





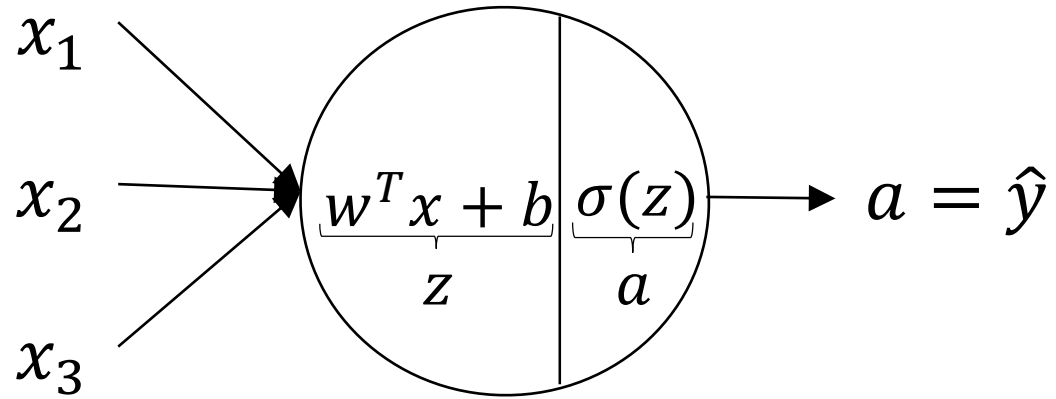
deeplearning.ai

# One hidden layer Neural Network

---

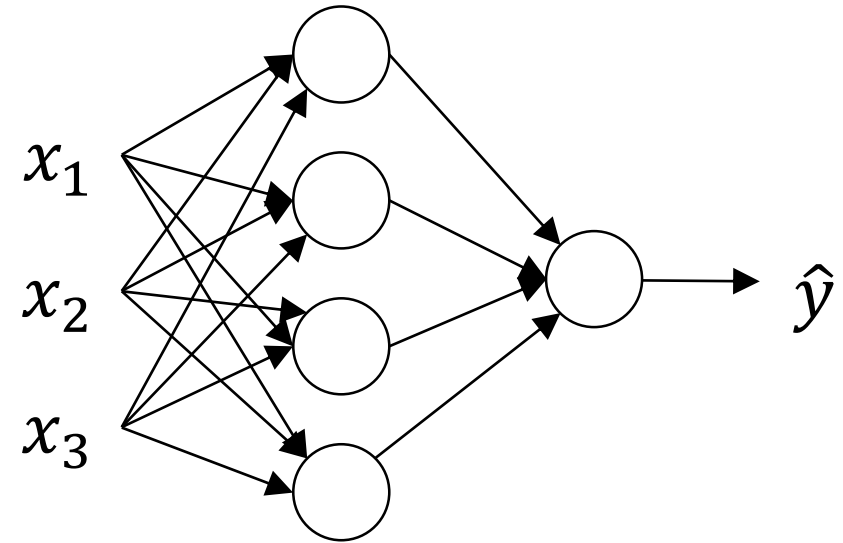
Computing a  
Neural Network's  
Output

# Neural Network Representation

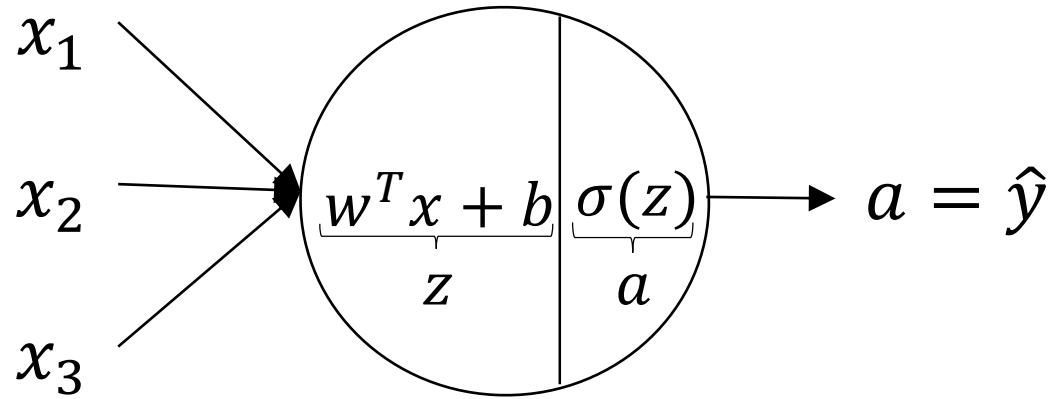


$$z = w^T x + b$$

$$a = \sigma(z)$$

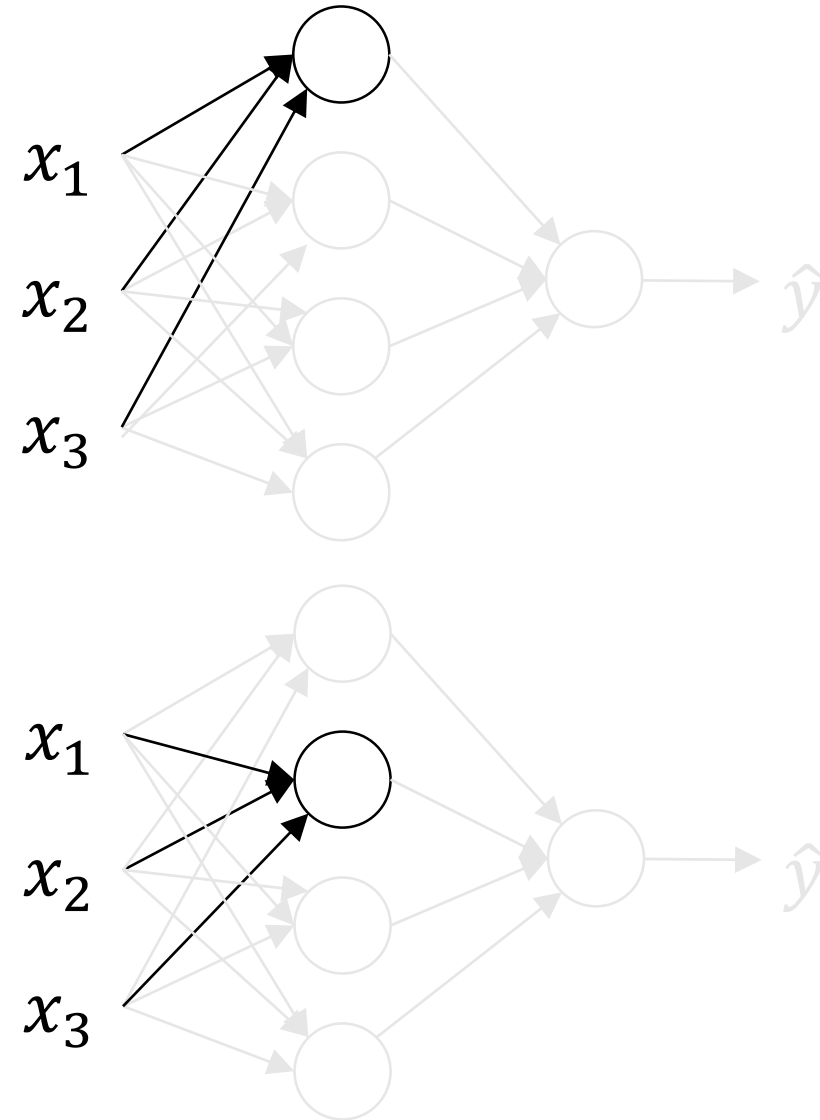


# Neural Network Representation

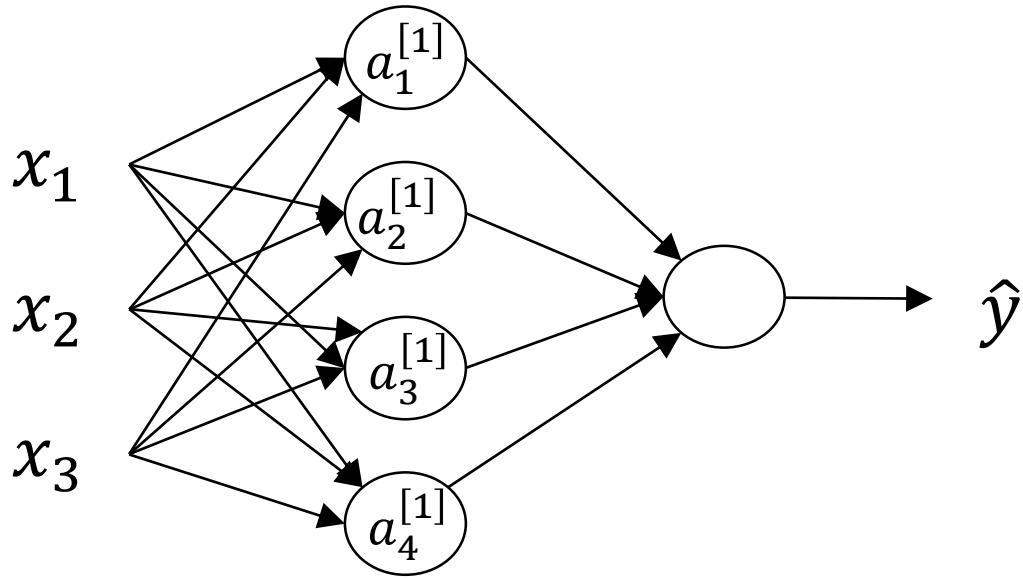


$$z = w^T x + b$$

$$a = \sigma(z)$$



# Neural Network Representation



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

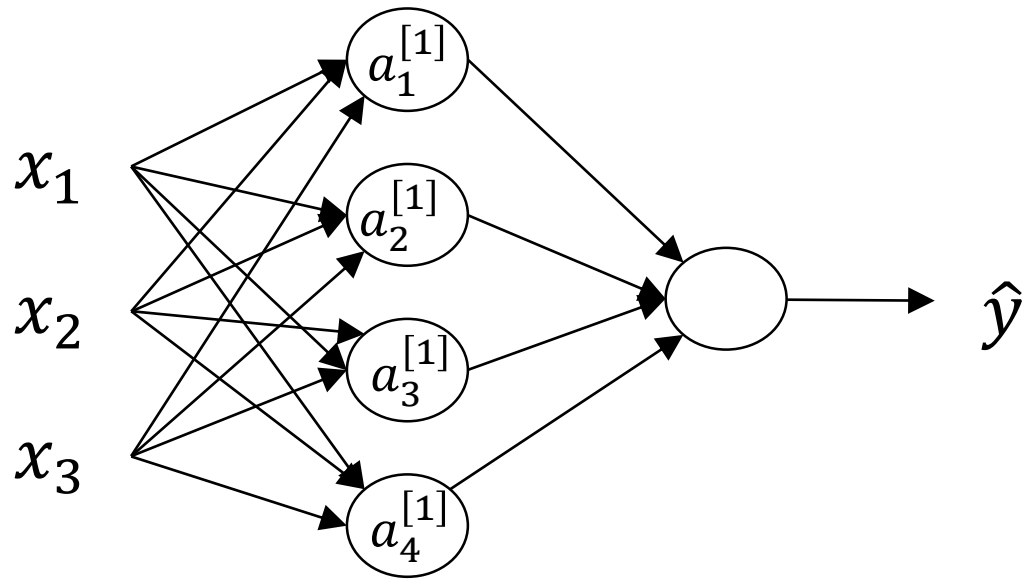
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$

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

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



# Neural Network Representation learning



Given input  $x$ :

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

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

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

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



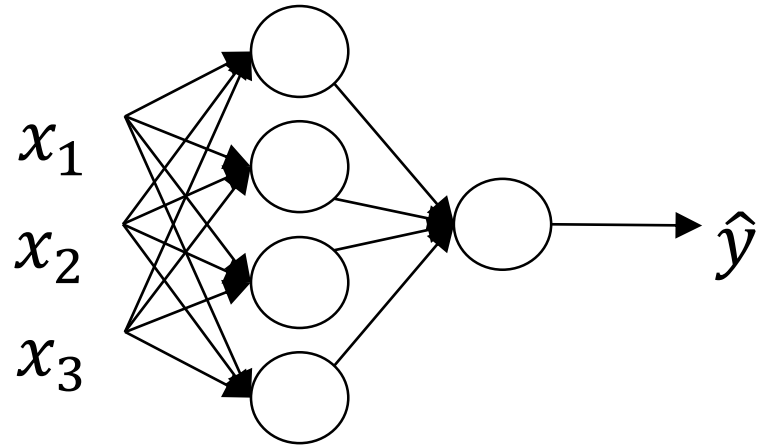
deeplearning.ai

# One hidden layer Neural Network

---

## Vectorizing across multiple examples

# Vectorizing across multiple examples



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

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

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

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

# Vectorizing across multiple examples

for  $i = 1$  to  $m$ :

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

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

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

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



deeplearning.ai

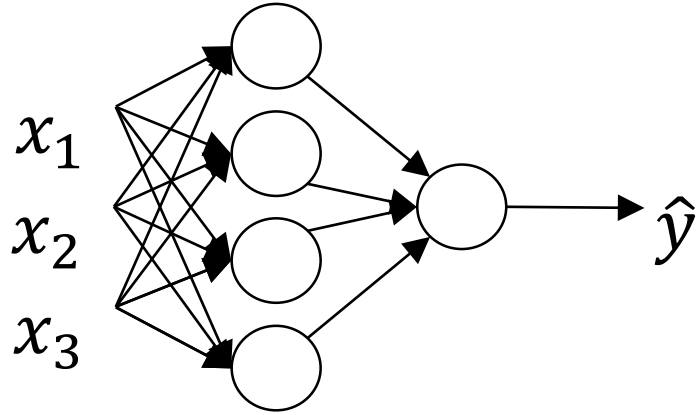
# One hidden layer Neural Network

---

Explanation  
for vectorized  
implementation

# Justification for vectorized implementation

# Recap of vectorizing across multiple examples



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

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

for  $i = 1$  to  $m$

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

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

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

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

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

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

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

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



deeplearning.ai

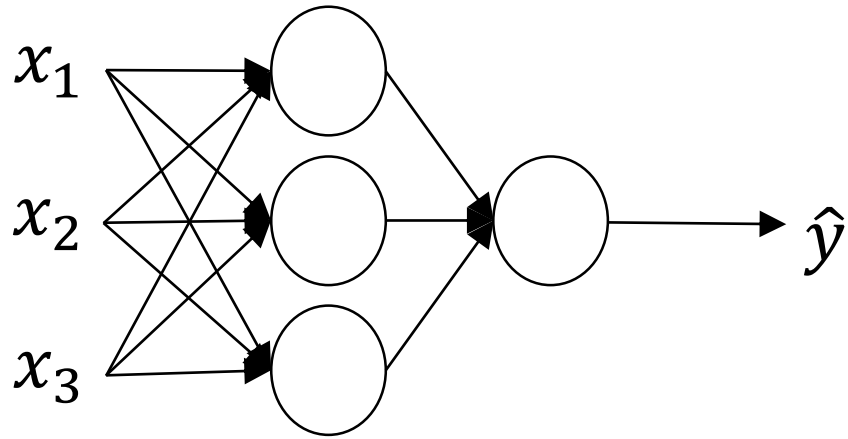
# One hidden layer Neural Network

---

## Activation functions



# Activation functions



Given  $x$ :

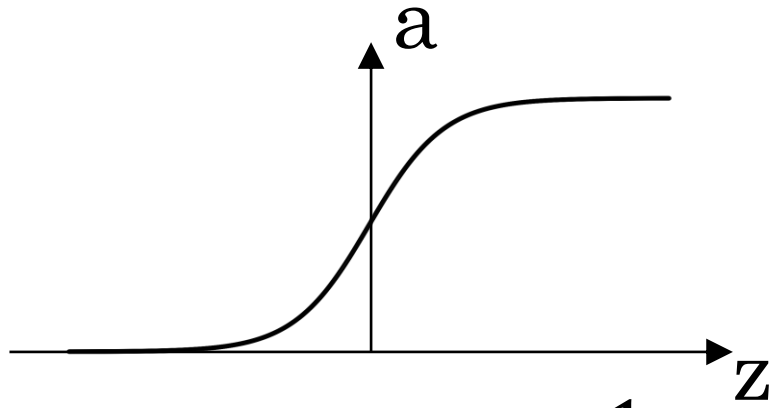
$$z^{[1]} = W^{[1]}x + b^{[1]}$$

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

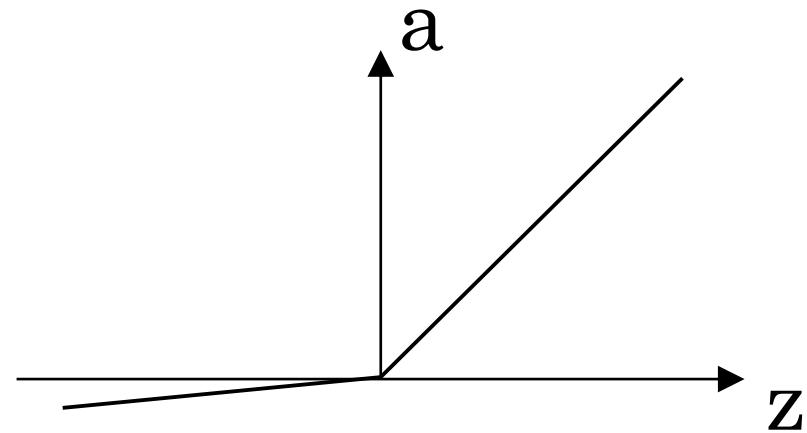
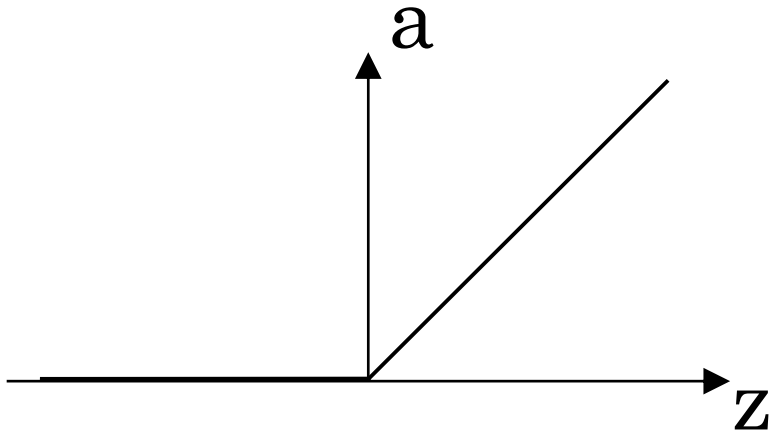
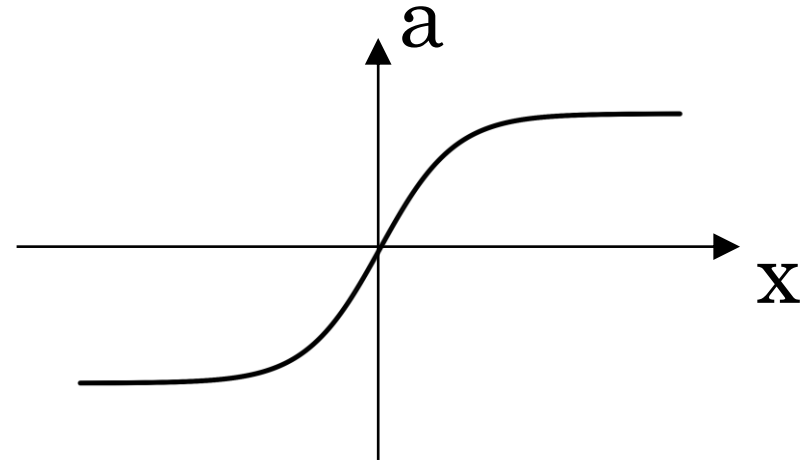
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

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

# Pros and cons of activation functions



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





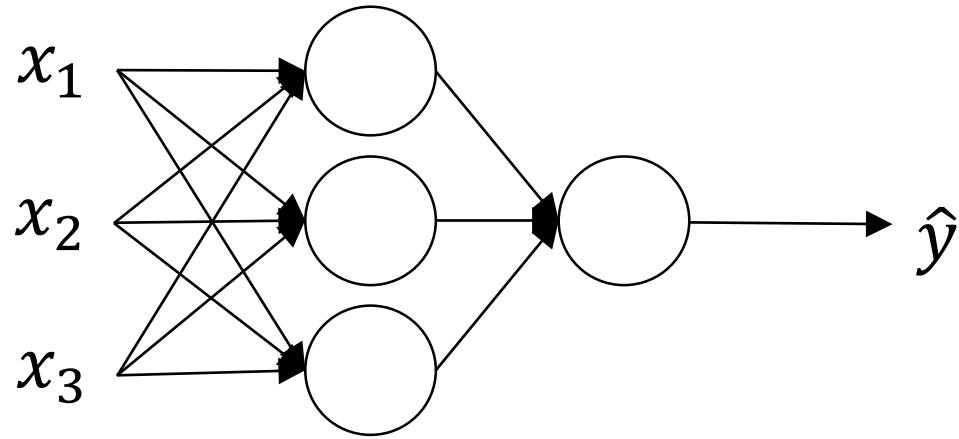
deeplearning.ai

# One hidden layer Neural Network

---

Why do you  
need non-linear  
activation functions?

# Activation function



Given  $x$ :

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

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

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

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



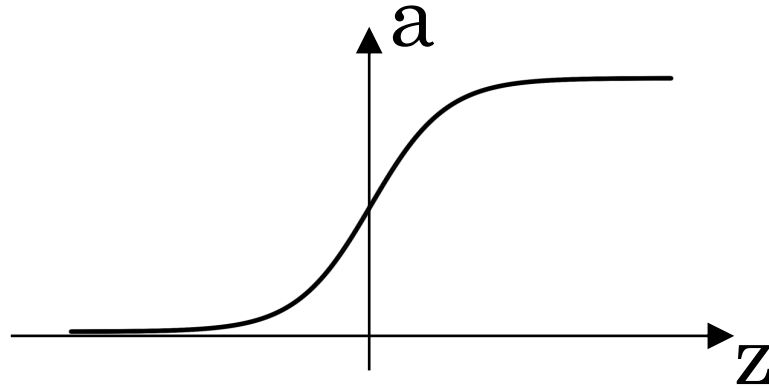
deeplearning.ai

# One hidden layer Neural Network

---

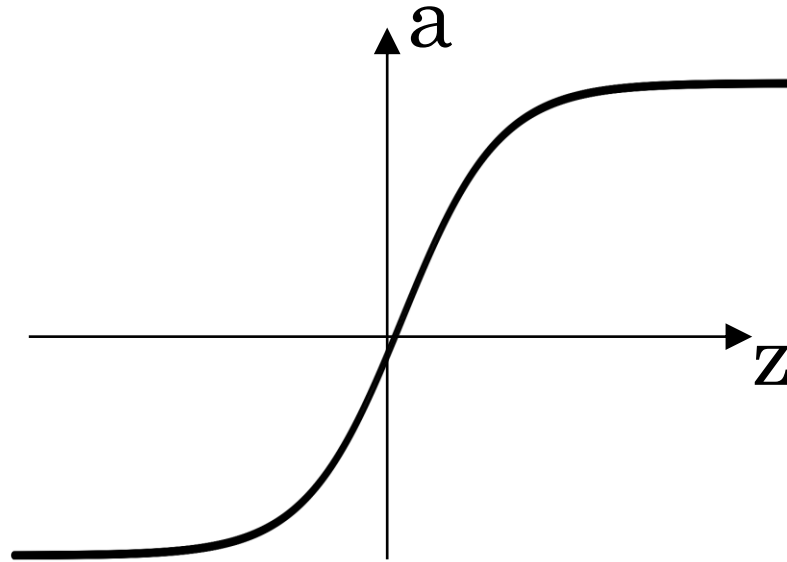
## Derivatives of activation functions

# Sigmoid activation function



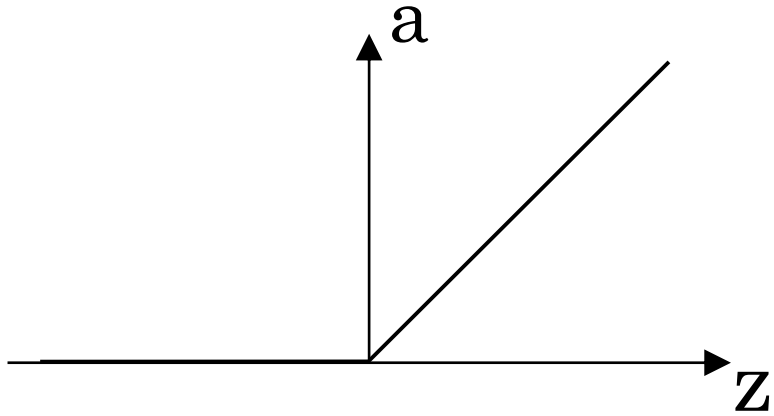
$$g(z) = \frac{1}{1 + e^{-z}}$$

# Tanh activation function

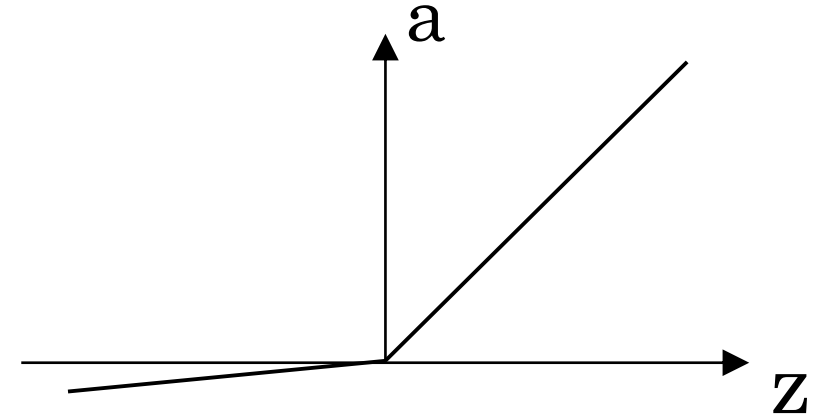


$$g(z) = \tanh(z)$$

# ReLU and Leaky ReLU



ReLU



Leaky ReLU





deeplearning.ai

One hidden layer  
Neural Network

---

Gradient descent for  
neural networks

# Gradient descent for neural networks

# Formulas for computing derivatives



deeplearning.ai

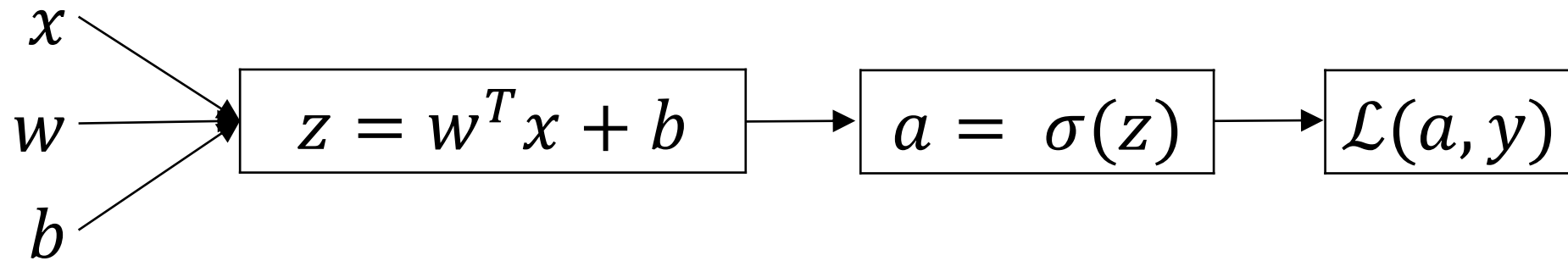
One hidden layer  
Neural Network

---

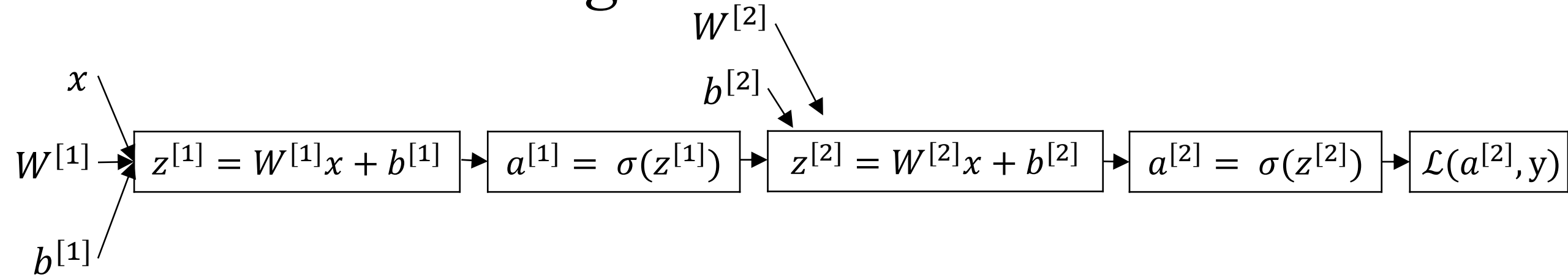
Backpropagation  
intuition (Optional)

# Computing gradients

## Logistic regression



# Neural network gradients



# Summary of gradient descent

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

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

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

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

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

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

# Summary of gradient descent

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

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

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

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

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

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

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

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

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

$$dZ^{[1]} = W^{[2]T} dZ^{[2]} * g^{[1]'}(Z^{[1]})$$

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

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





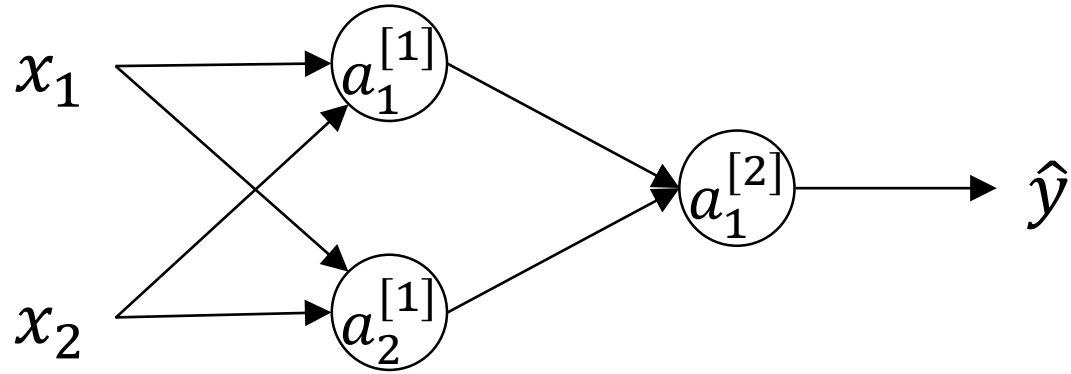
deeplearning.ai

One hidden layer  
Neural Network

---

Random Initialization

# What happens if you initialize weights to zero?



# Random initialization

