Artificial Intelligence and Machine Learning

Neural Networks

Lecture Outline

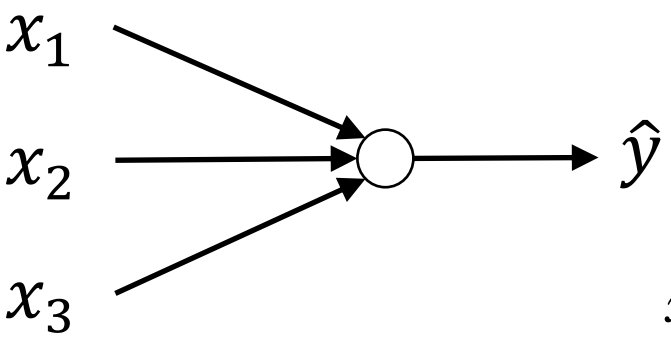
- Logistic Regression Review
- Neural Networks
  - Forward pass
  - Backward pass

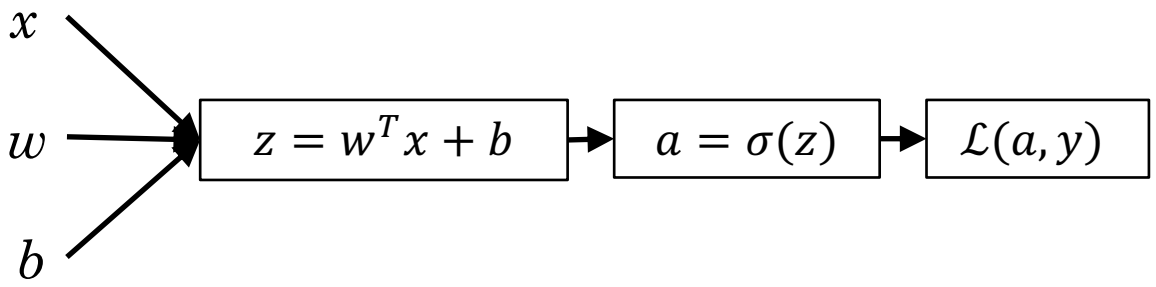
deeplearning.ai

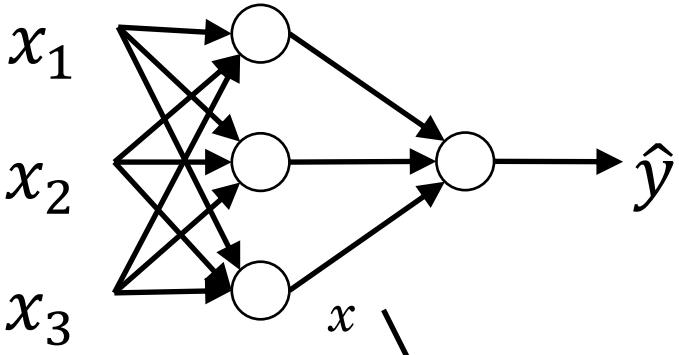
One hidden layer Neural Network

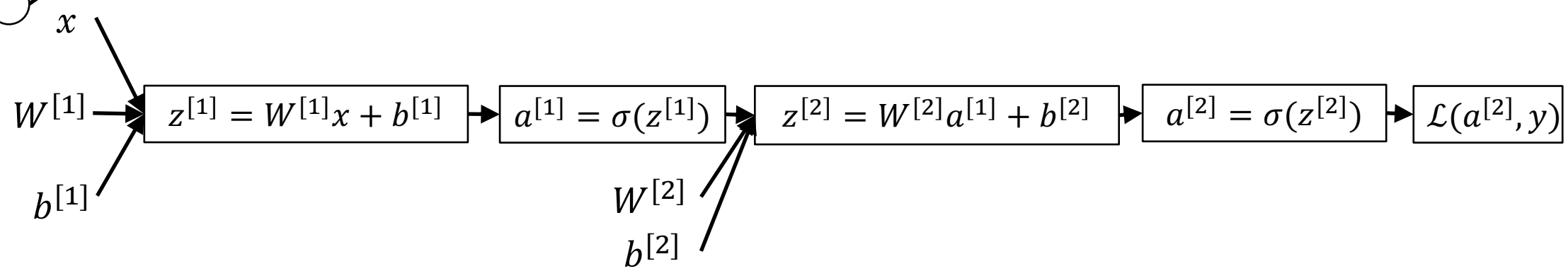
Neural Networks Overview

What is a Neural Network?









deeplearning.ai

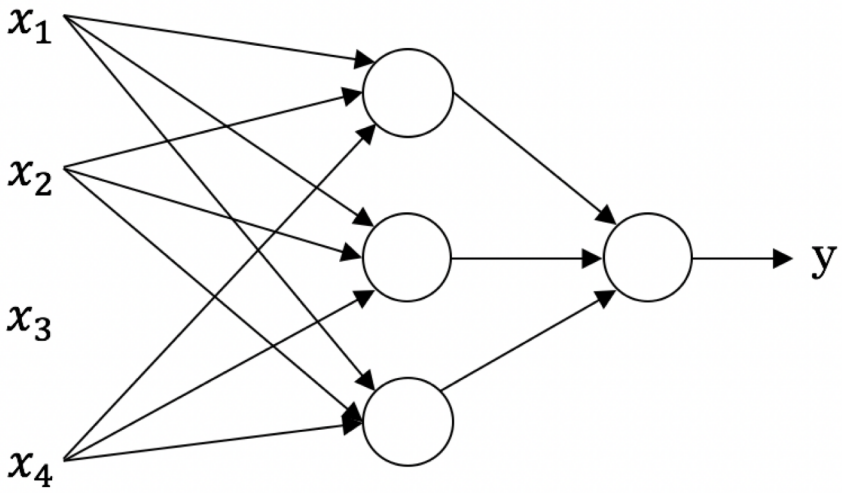
Introduction to Deep Learning

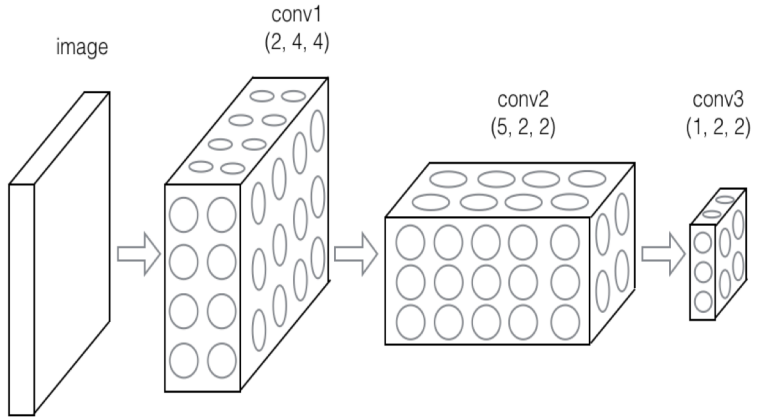
Supervised Learning with Neural Networks

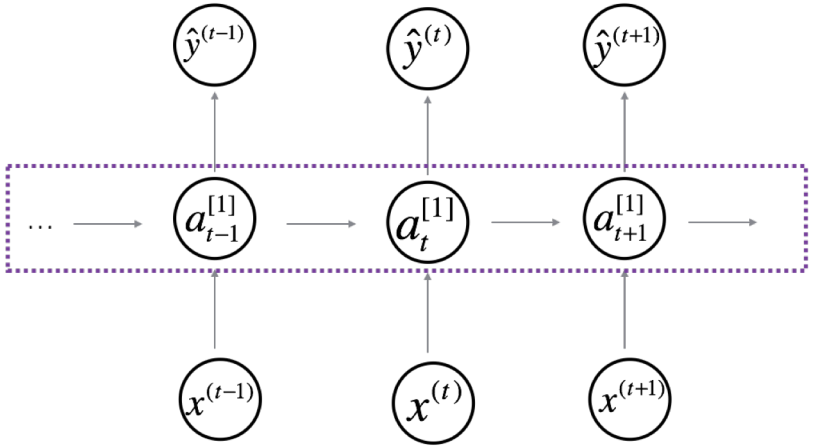
Supervised Learning

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

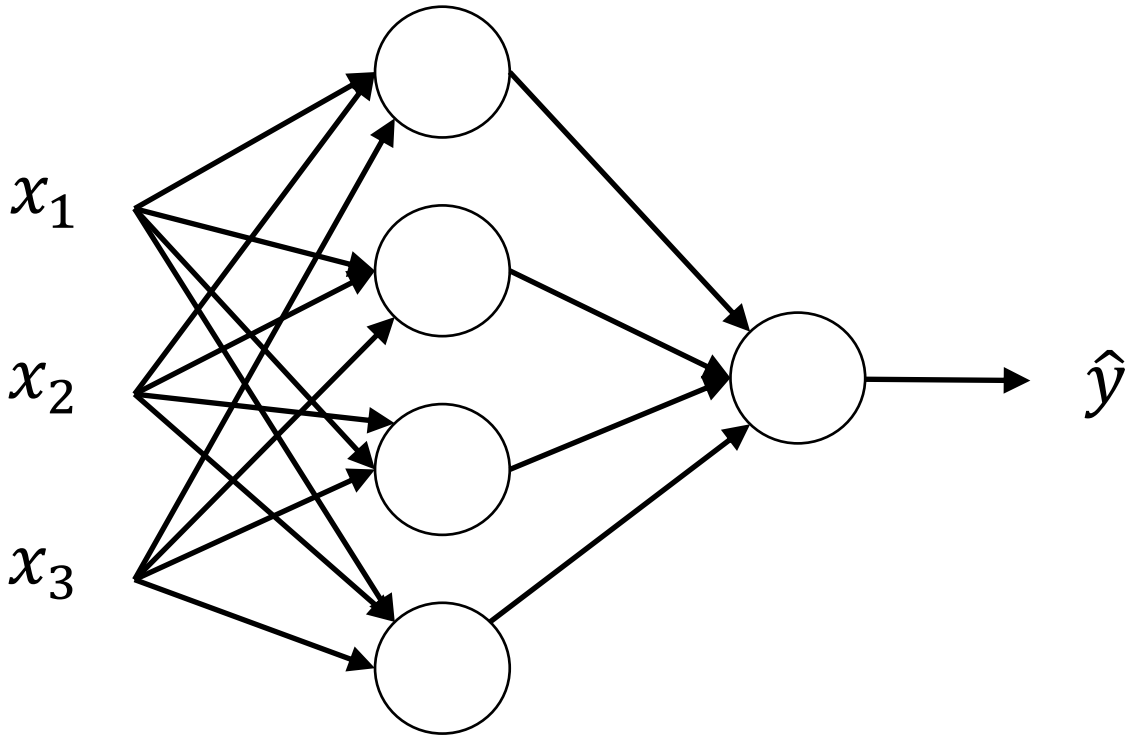
Recurrent NN

deeplearning.ai

One hidden layer Neural Network

Neural Network Representation

Neural Network Representation

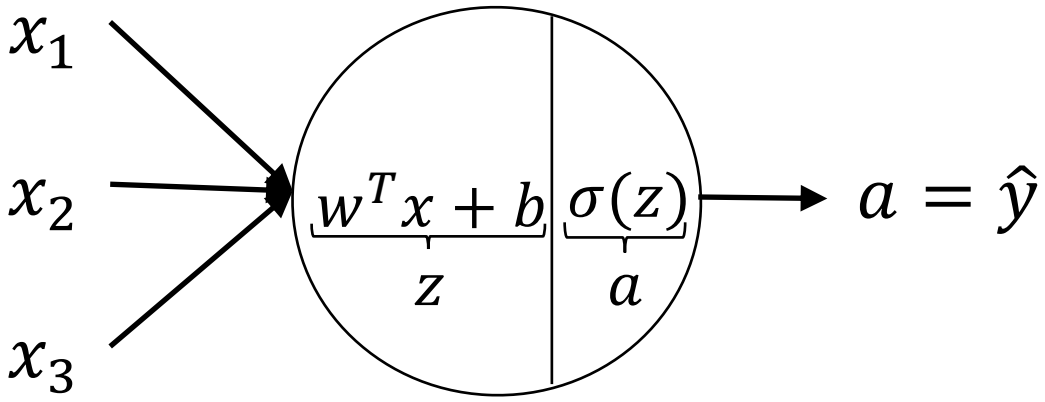


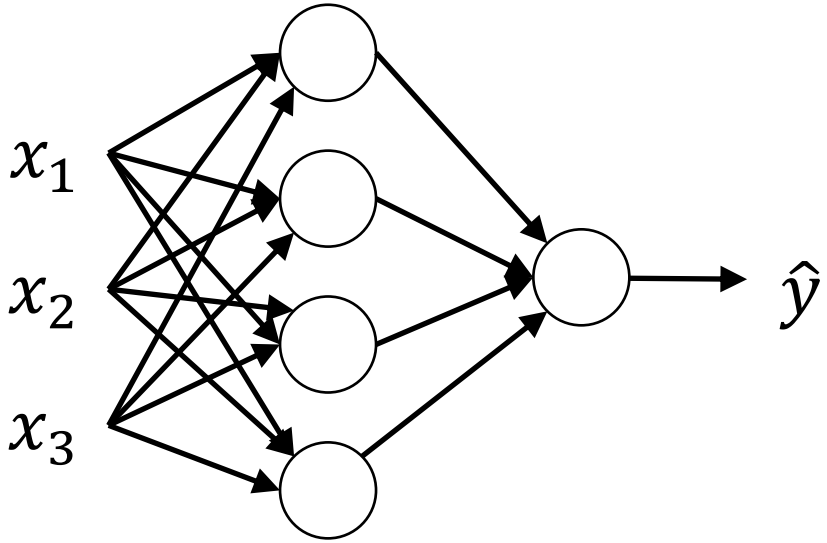
deeplearning.ai

One hidden layer Neural Network

Computing a Neural Network's Output

Neural Network Representation

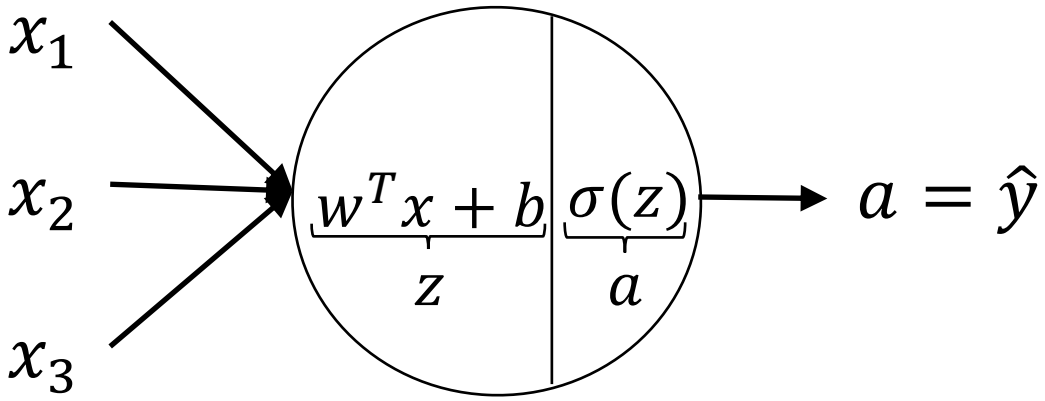




$$z = w^T x + b$$

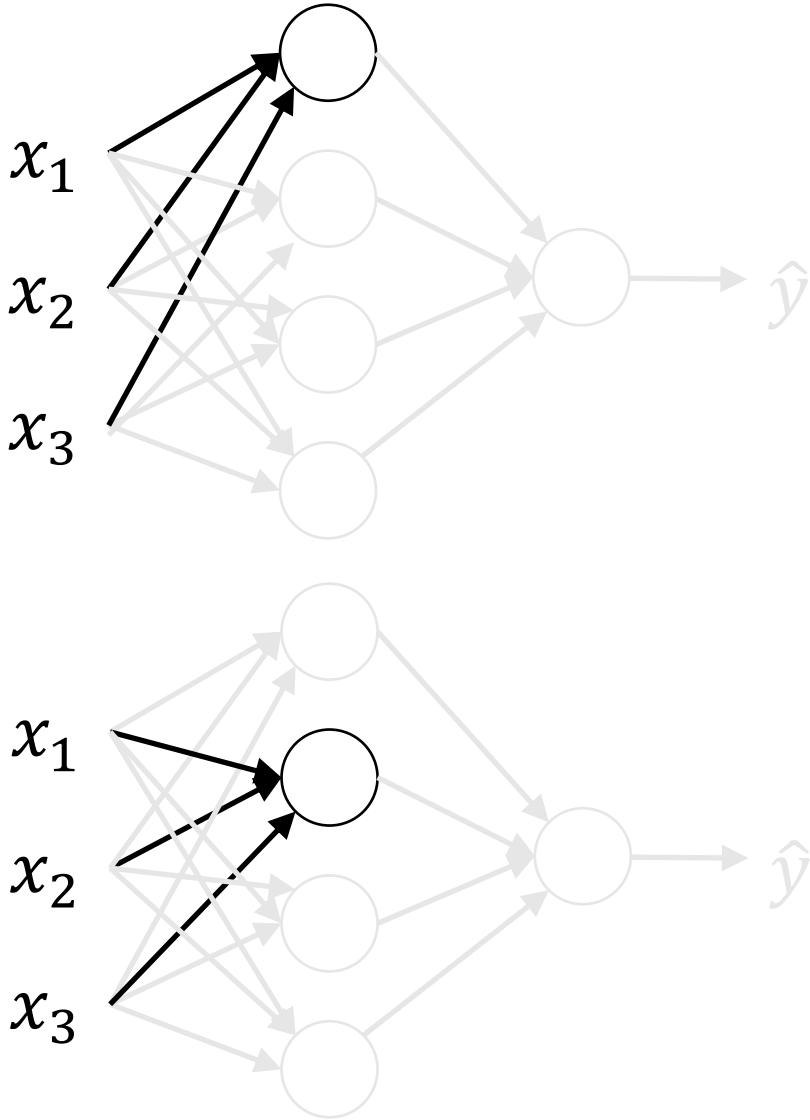
$$a = \sigma(z)$$

Neural Network Representation

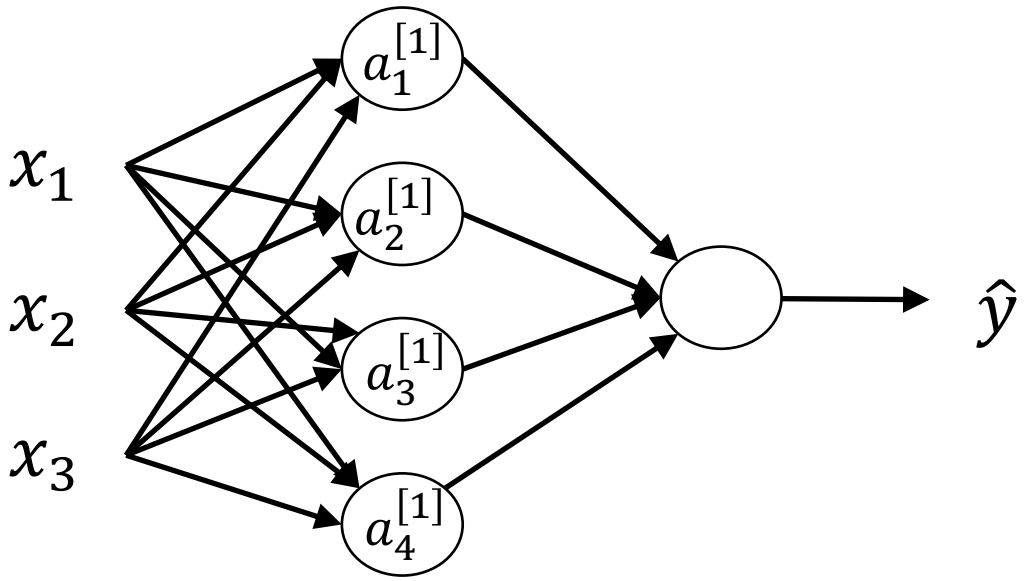


$$z = w^T x + b$$

$$a = \sigma(z)$$



Neural Network Representation



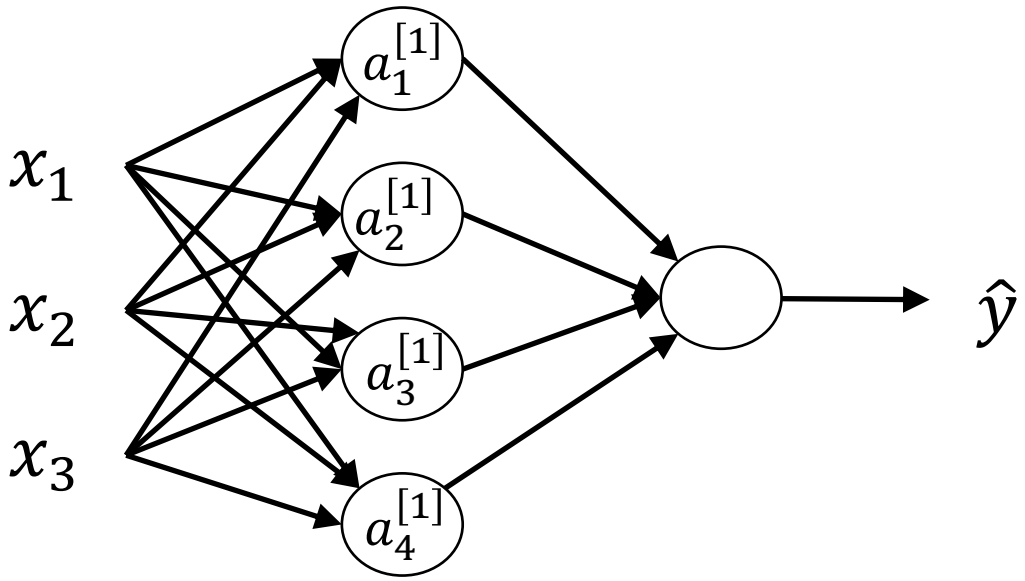
$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]},\ a_1^{[1]} = \sigma(z_1^{[1]})$$

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

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

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

Neural Network Representation learning



Given input x:

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

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

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

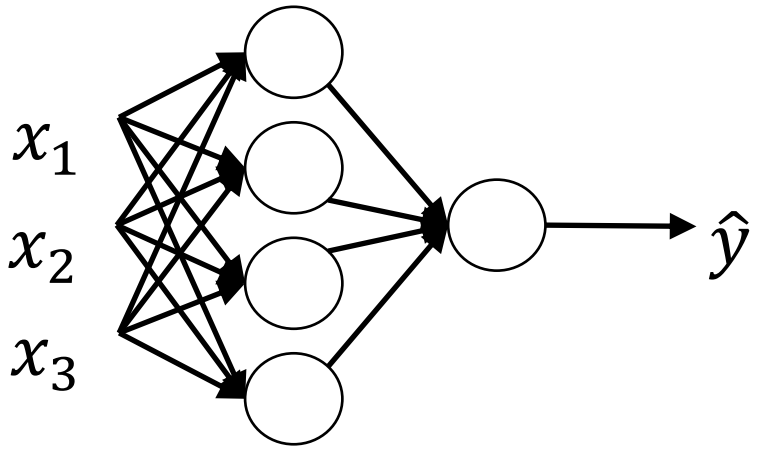
$$a^{[2]} = \sigma(z^{[2]})$$

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:
```

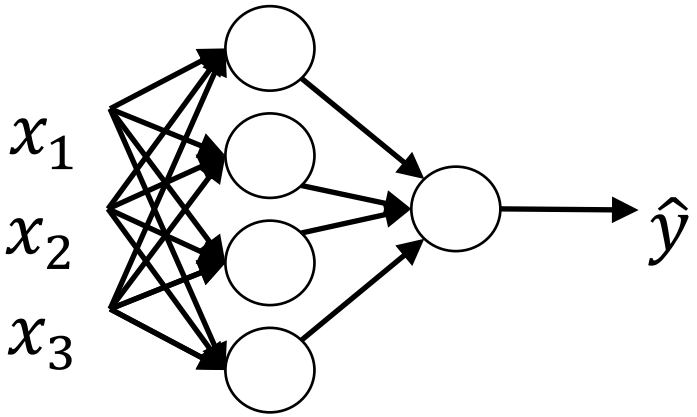
$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$

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

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

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

Recap of vectorizing across multiple examples




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

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

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

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

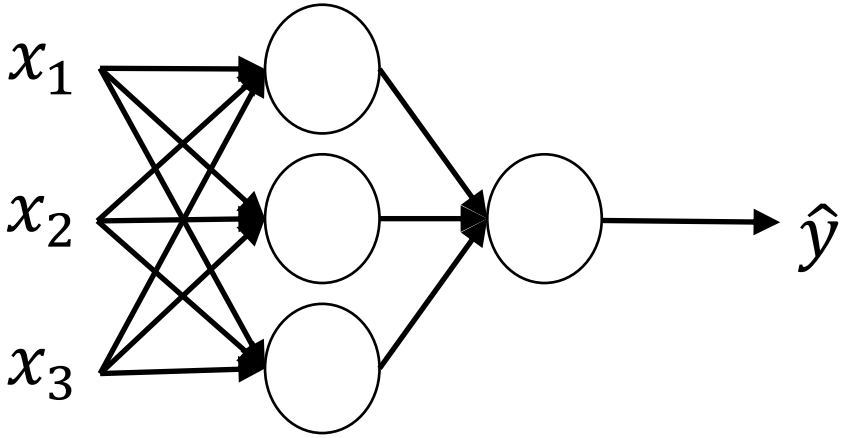
$$\mathrm{A}^{[1]} = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

deeplearning.ai

One hidden layer Neural Network

Activation functions

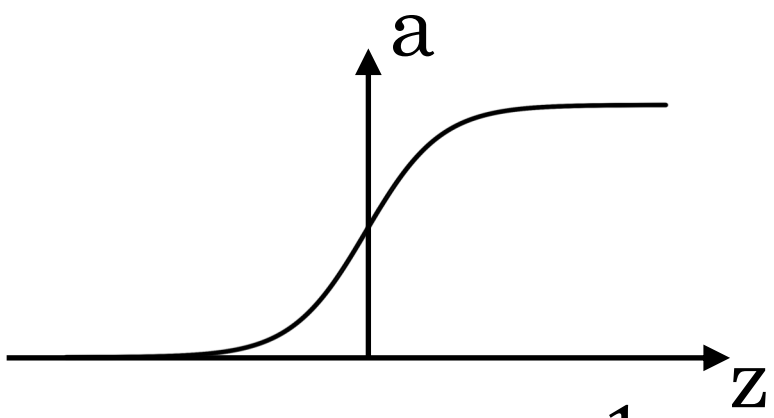
Activation functions



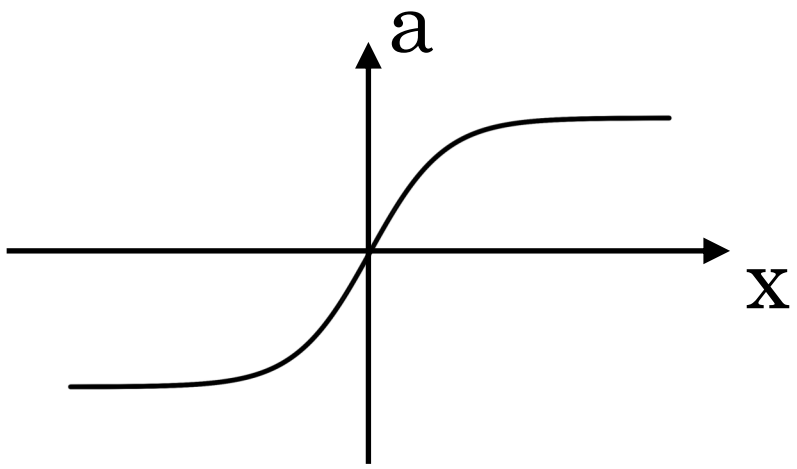
Given x:

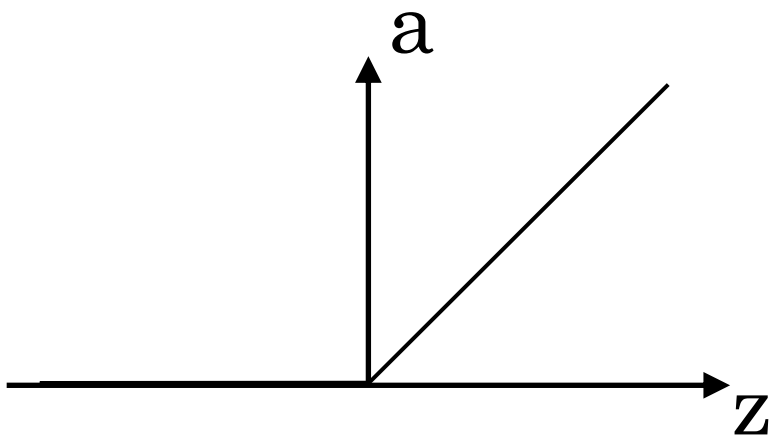
$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = \sigma(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = \sigma(z^{[2]})$$

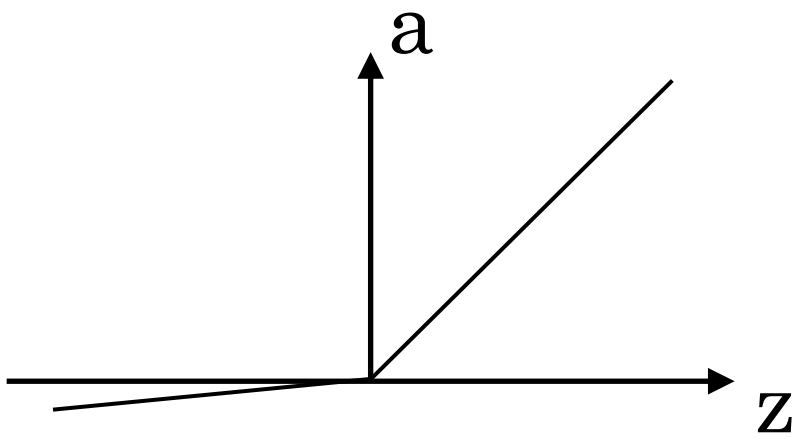
Pros and cons of activation functions



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





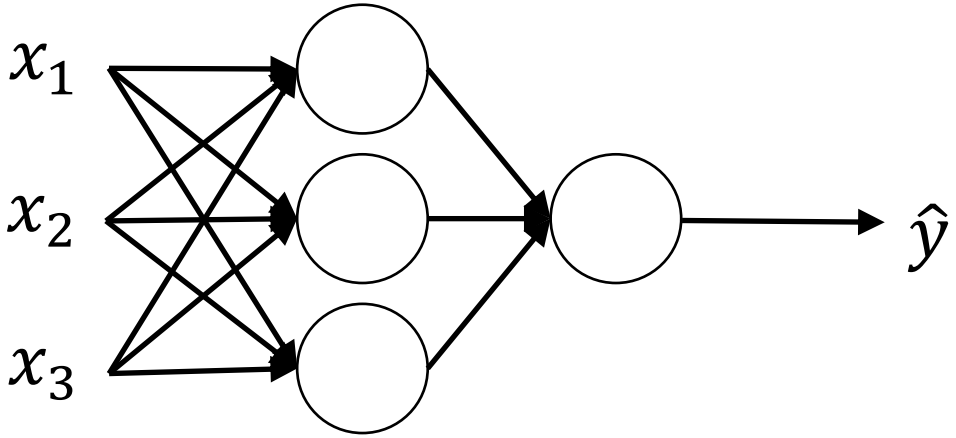


deeplearning.ai

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

Gradient descent for neural networks

Gradient descent for neural networks

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:

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

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

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

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

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

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

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

$*$ ↑ element-wise product

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

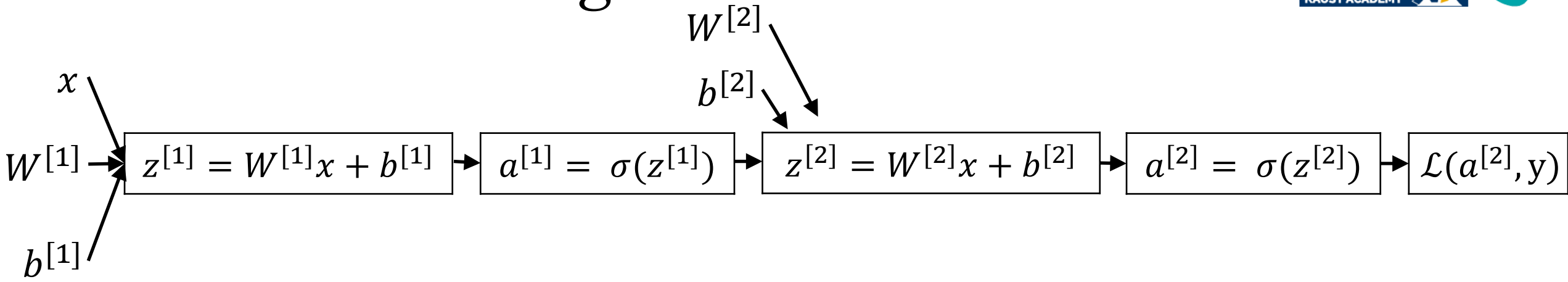
$$db^{[1]} = \frac{1}{m} \text{np.sum}(dz^{[1]}, \text{axis}=1, \text{keepdims}=\text{True})$$

deeplearning.ai

One hidden layer Neural Network

Backpropagation intuition

Neural network gradients



Andrew Ng

Summary of gradient descent

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

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

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

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

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

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

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

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

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

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

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

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

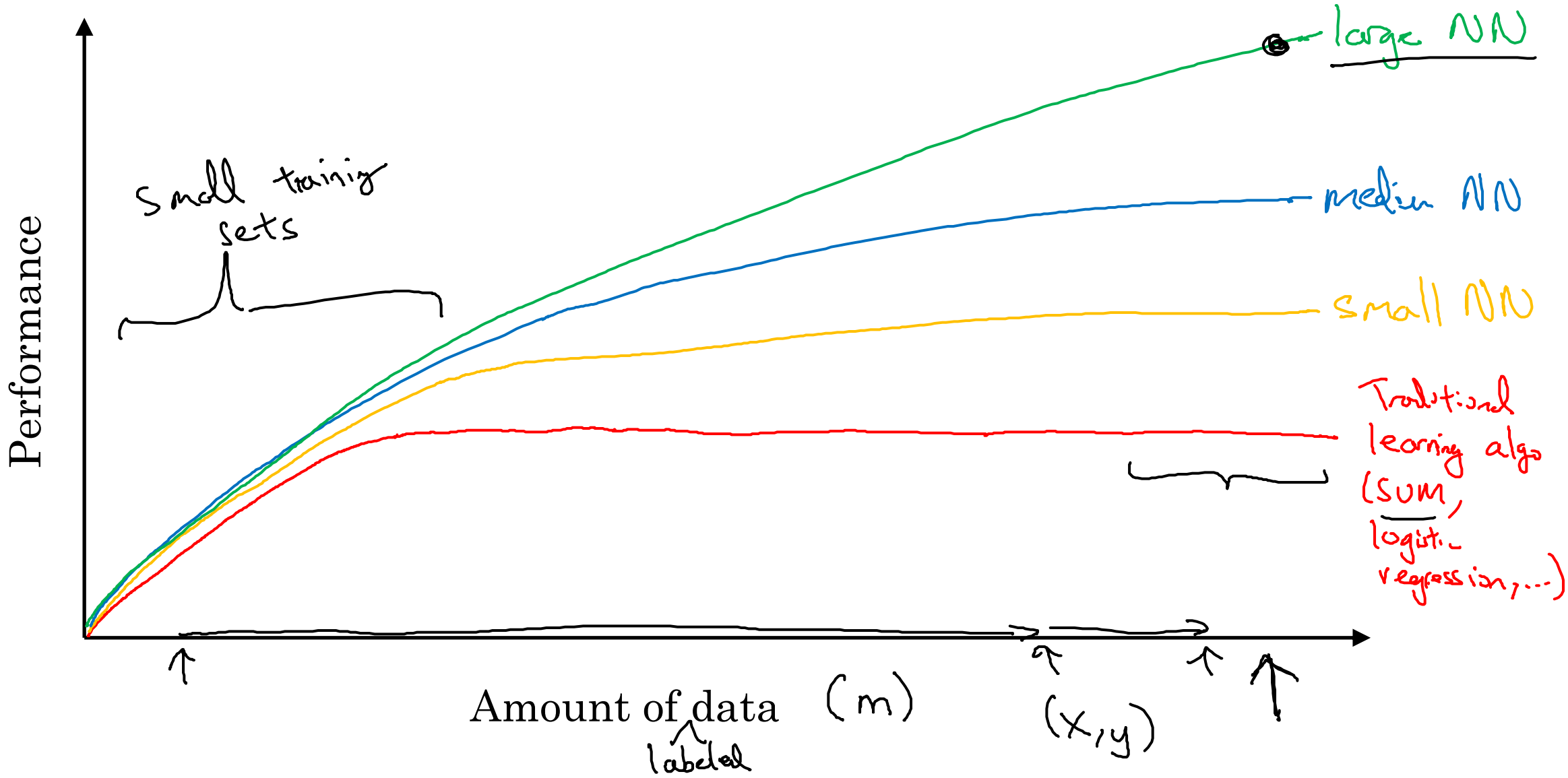
Andrew Ng

deeplearning.ai

Introduction to Neural Networks

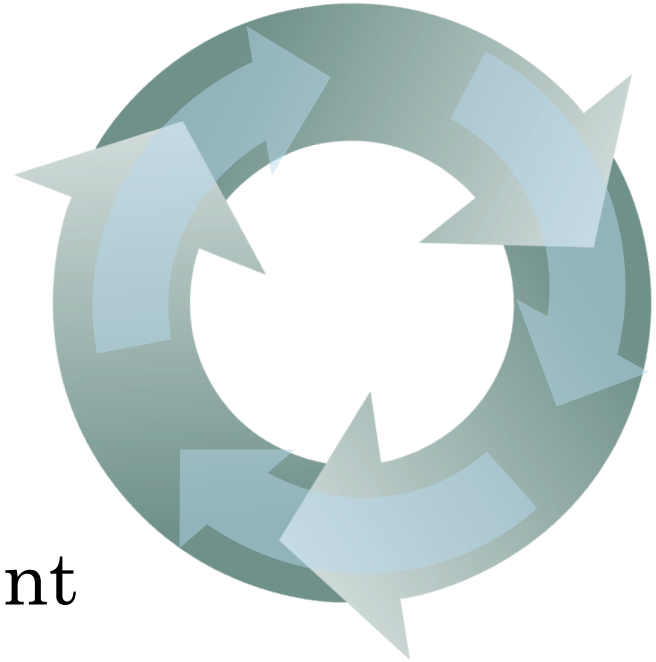
Why is Deep Learning taking off?

Scale drives deep learning progress



Scale drives deep learning progress

- Data
- Computation
- Algorithms

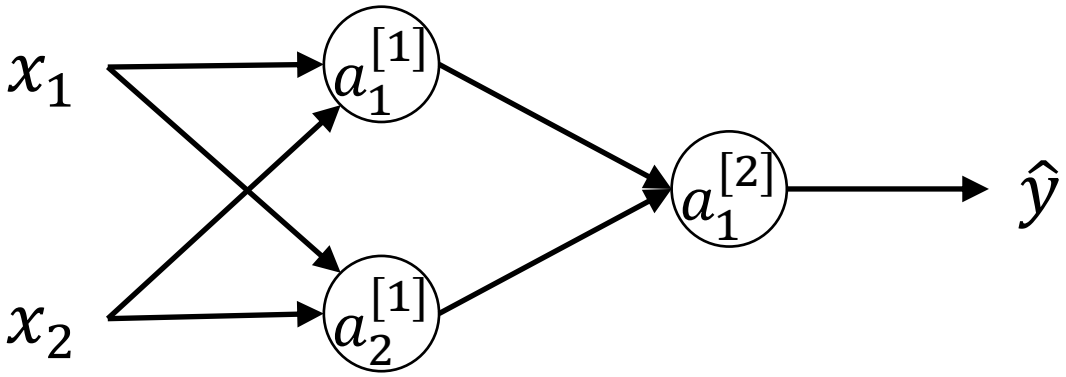


deeplearning.ai

One hidden layer Neural Network

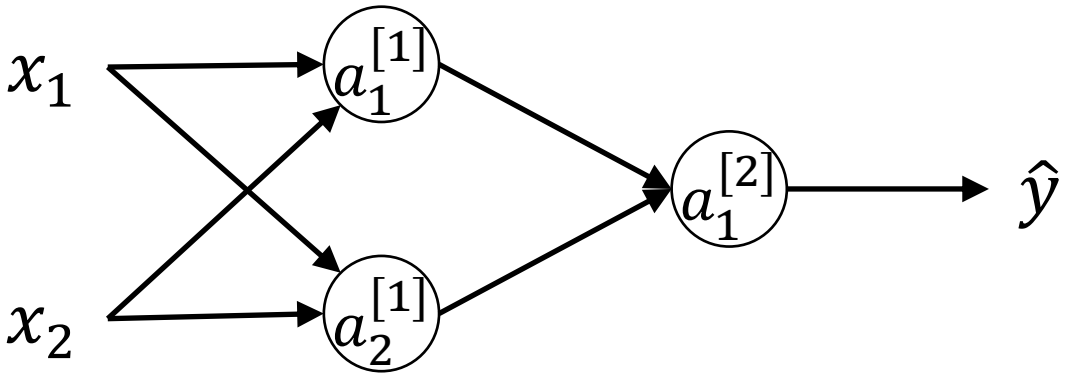
Random Initialization

What happens if you initialize weights to zero?



1. **Symmetry Problem**
2. **Impact on Activation Functions**

Random initialization



Very close to zero, but randomly

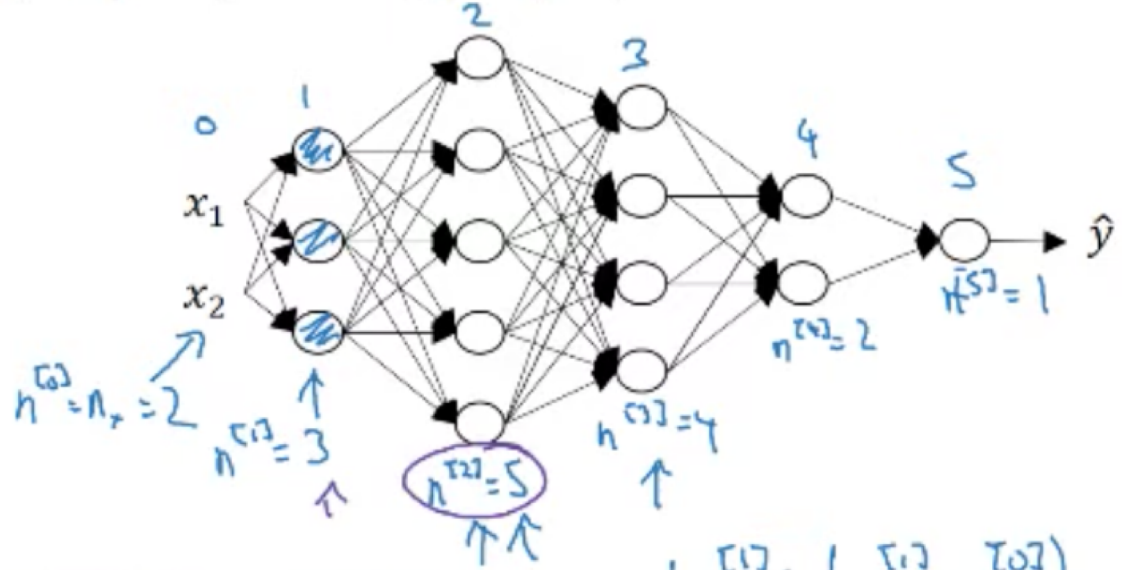
In numpy:
np.random.randn(layer_size[l], layer_size[l-1]), generates a matrix of random numbers drawn from a standard normal distribution (mean = 0, standard deviation = 1). This matrix has a shape determined by the size of the current layer (layer_size[l]) and the size of the previous layer (layer_size[l-1]).

deeplearning.ai

Deep Neural Networks

Getting your matrix dimensions right

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



$L = 5$

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

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

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

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

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

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

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

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

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

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

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

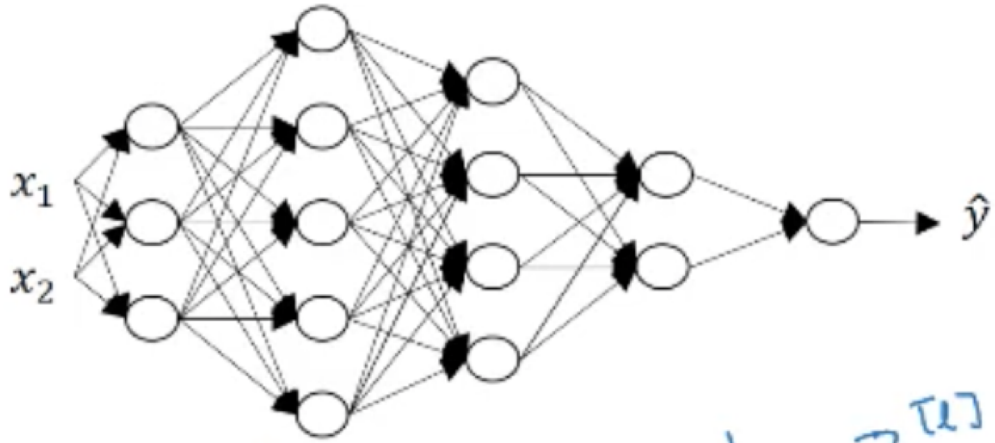
$(n^{[2]},1)$

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

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

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

Vectorized implementation



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

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

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

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

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

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

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

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

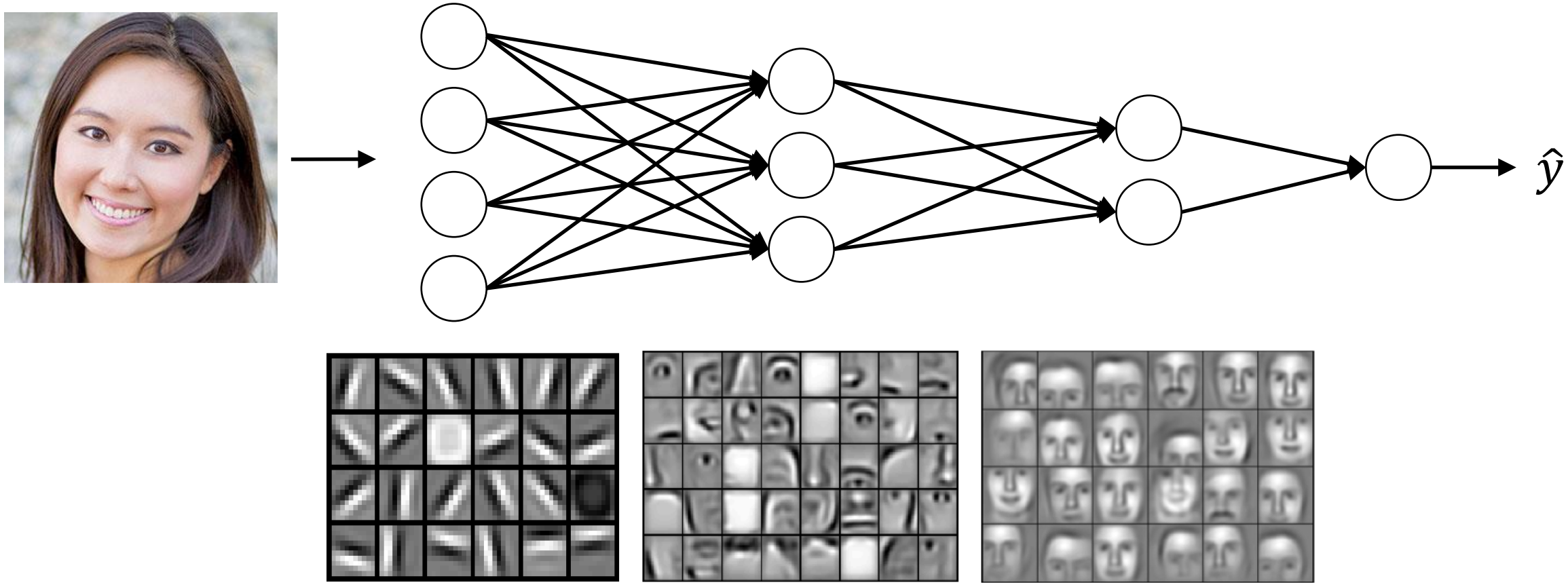
$dZ^{[l]}, dA^{[l]} : (n^{[l]}, m)$

deeplearning.ai

Deep Neural Networks

Why deep representations?

Intuition about deep representation



deeplearning.ai

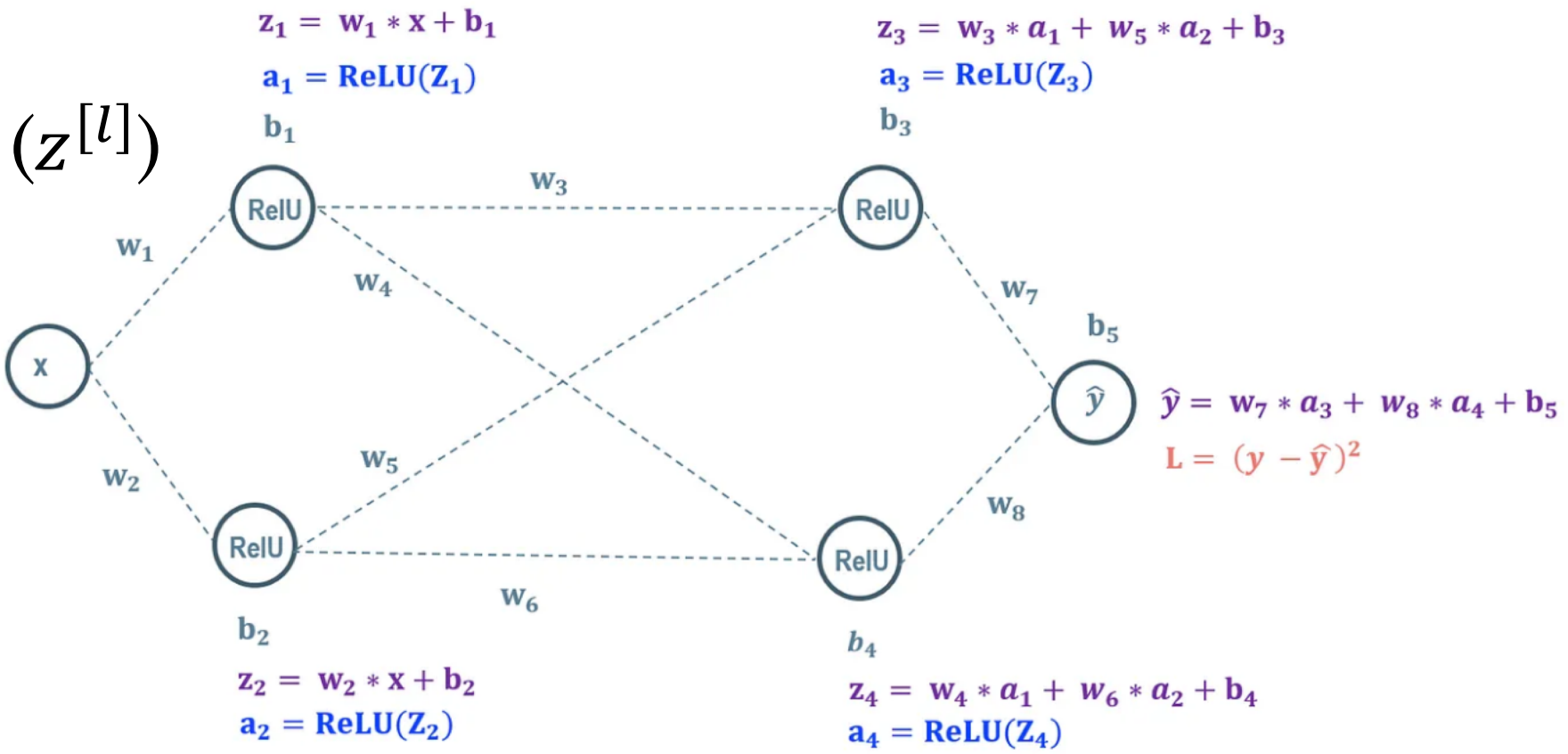
Deep Neural Networks

Forward and backward propagation

Forward and Backpropagation propagation for layer *l*

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

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



Summary