Artificial Intelligence and Machine Learning

Neural Networks

Lecture Outline

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

Review: Logistic Regression

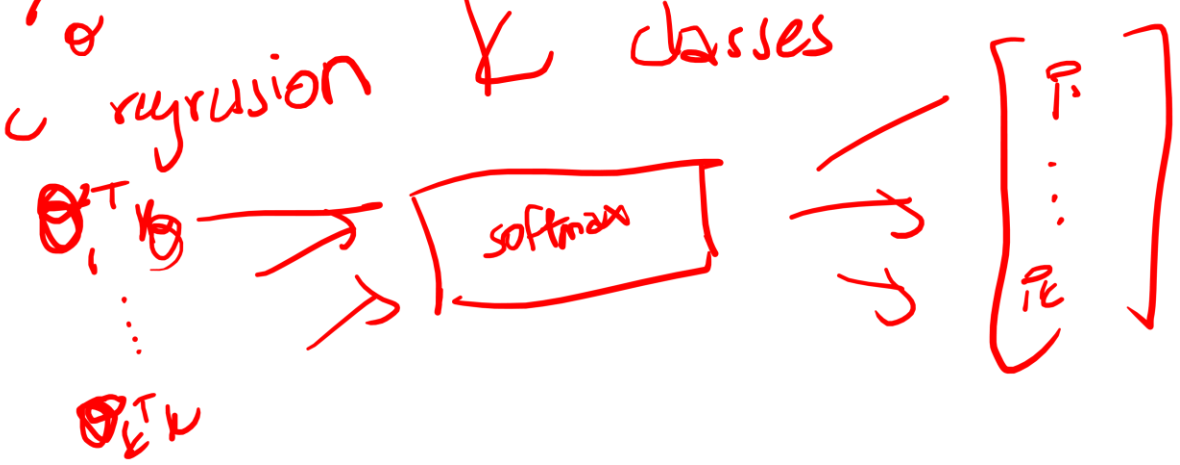
$p_i = \sigma(\theta^T x)$

σ: sigmoid

Loss function: LogLoss $\rightarrow J = \frac{1}{N} \sum_{i=1}^{N} -(y_i \log p_i + (1-y_i) \log(1-p_i))$

$\nabla_\theta J = 0$

Multiclass logistic regression K classes



Gradient descent

$\theta \leftarrow \theta - \eta \nabla_\theta J$

learning

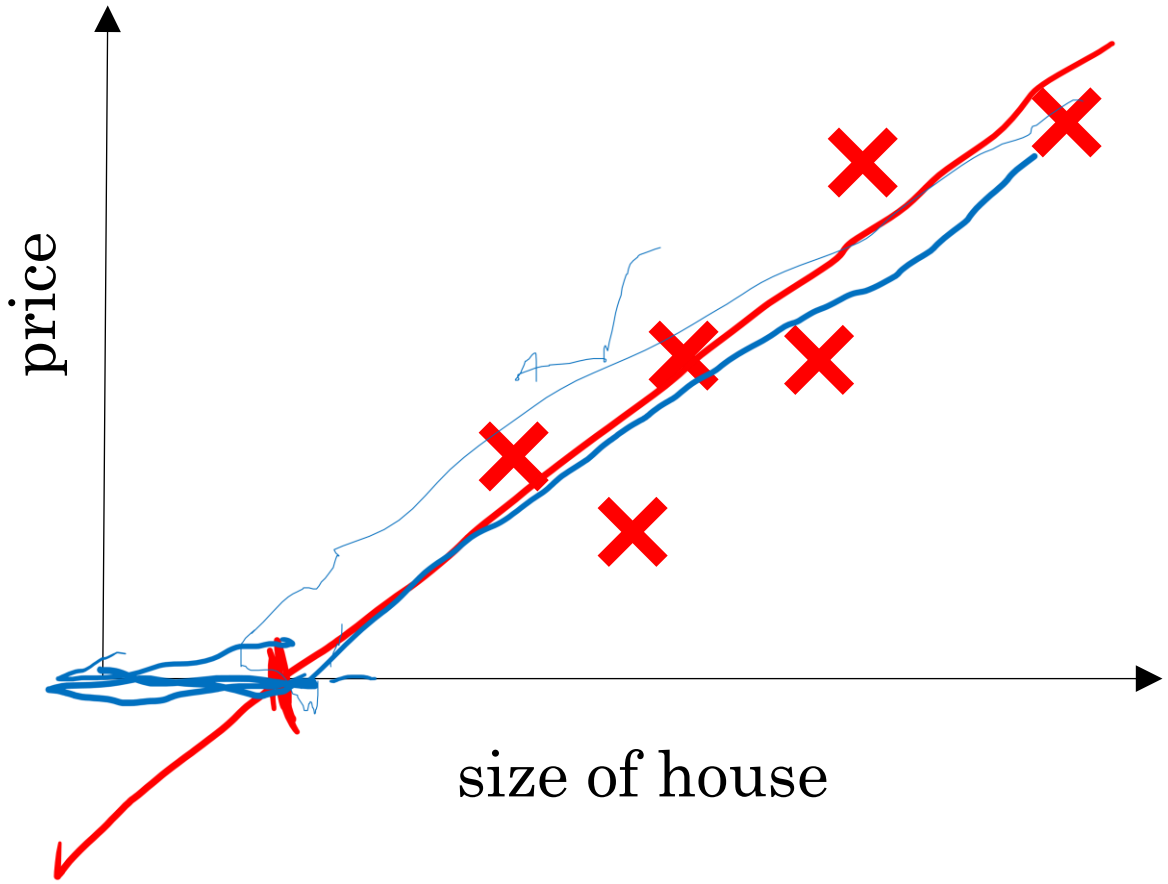
- basic FULL Traing
- minibatch
- Stchastic GD

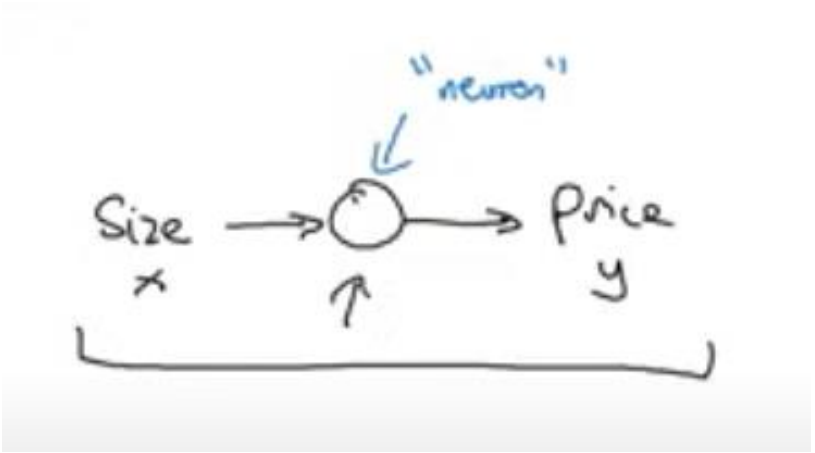
deeplearning.ai

Introduction to Deep Learning

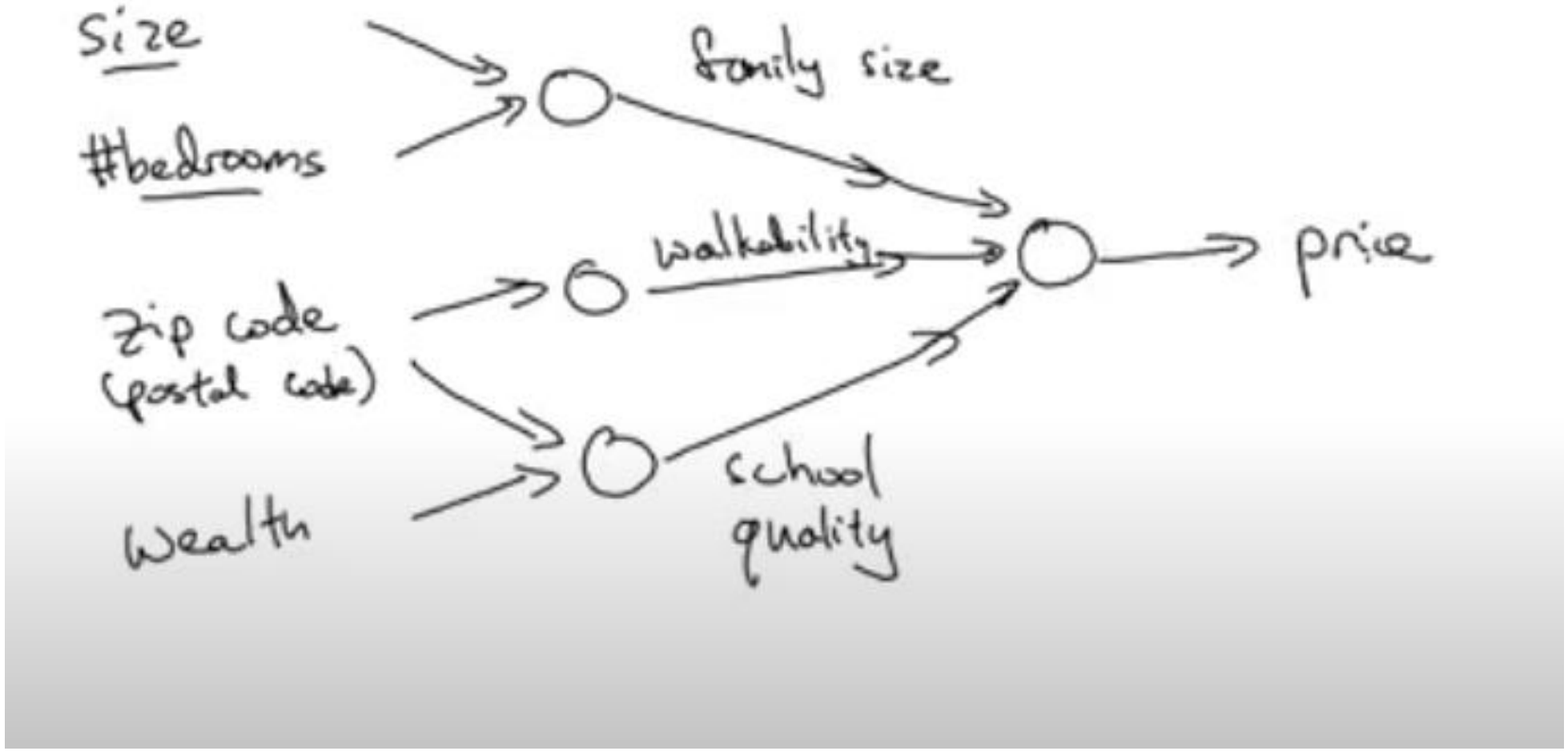
What is a Neural Network?

Housing Price Prediction

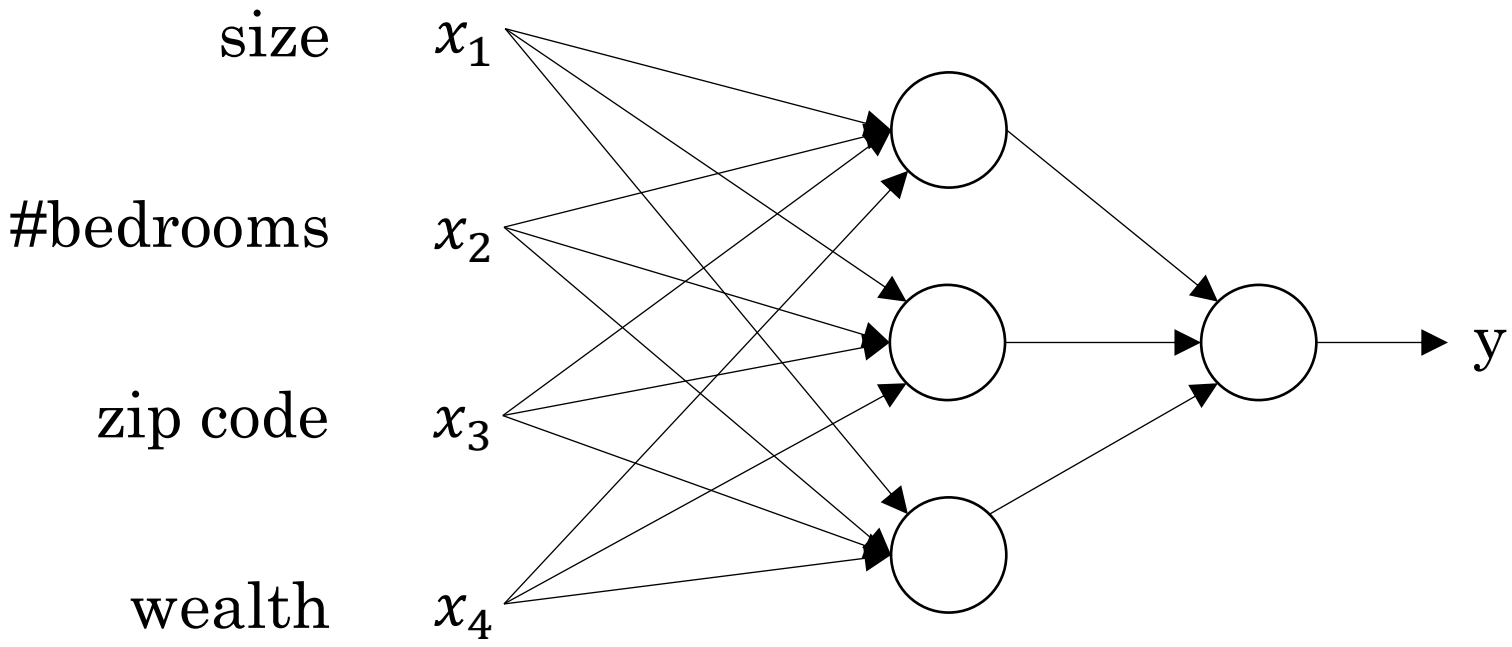




Housing Price Prediction



Housing Price Prediction



deeplearning.ai

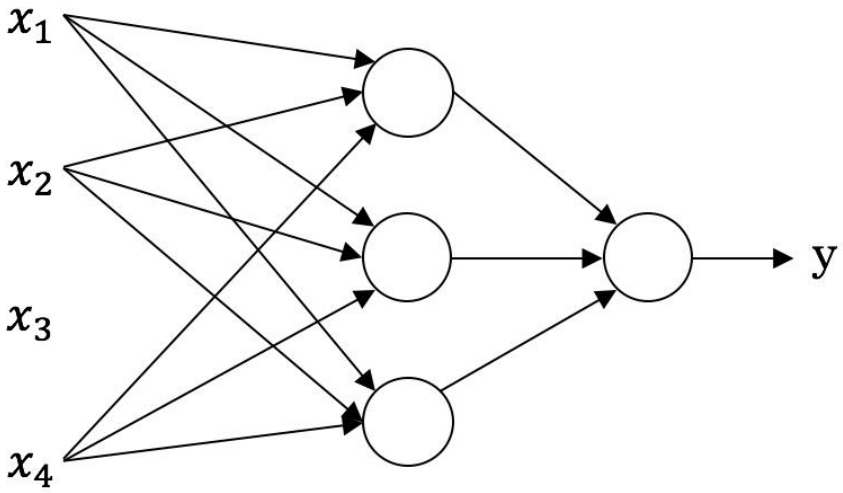
Introduction to Deep Learning

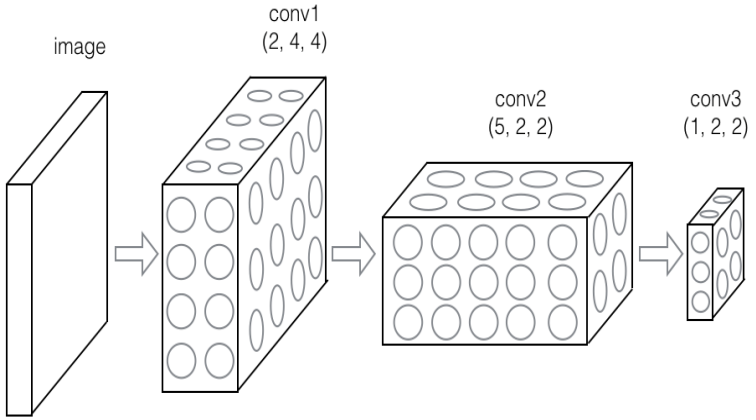
Supervised Learning with Neural Networks

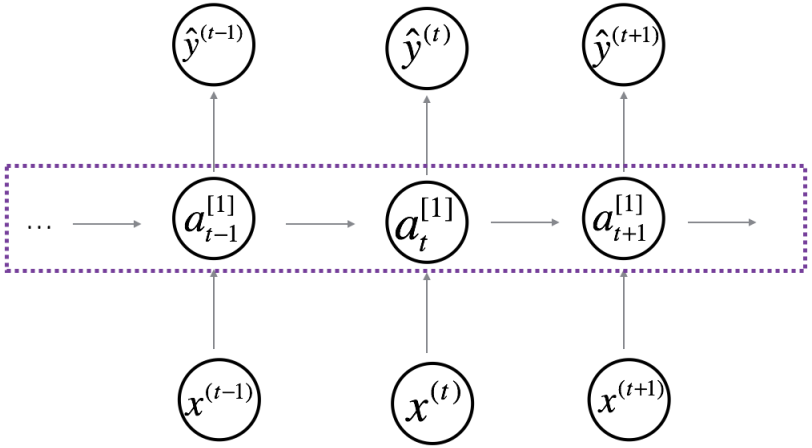
Supervised Learning

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

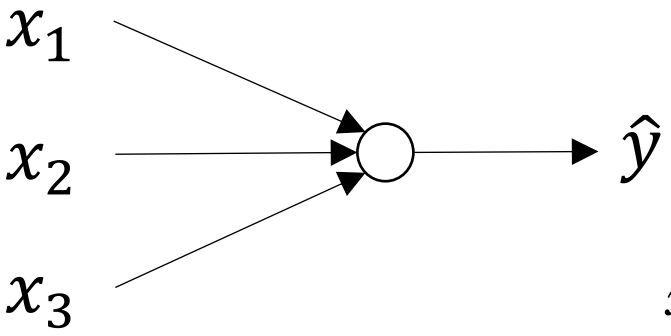
Recurrent NN

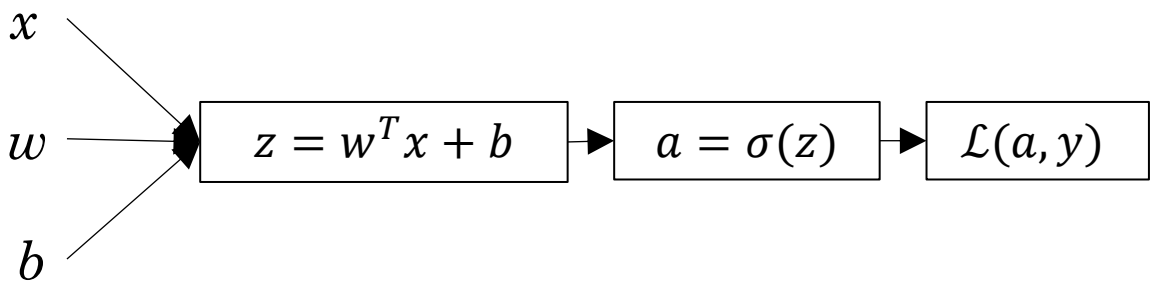
deeplearning.ai

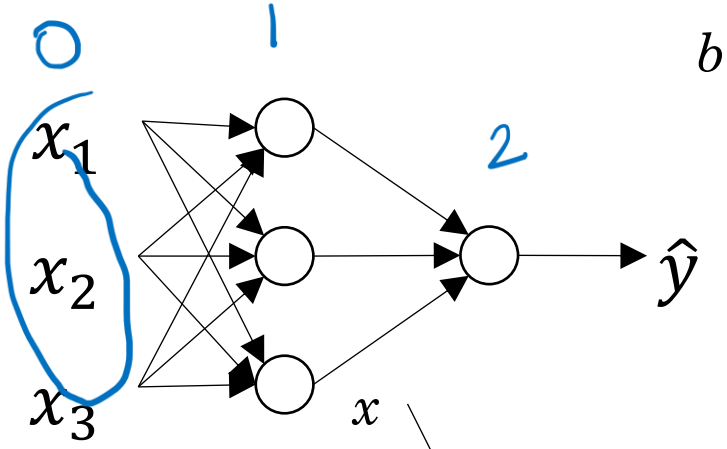
One hidden layer Neural Network

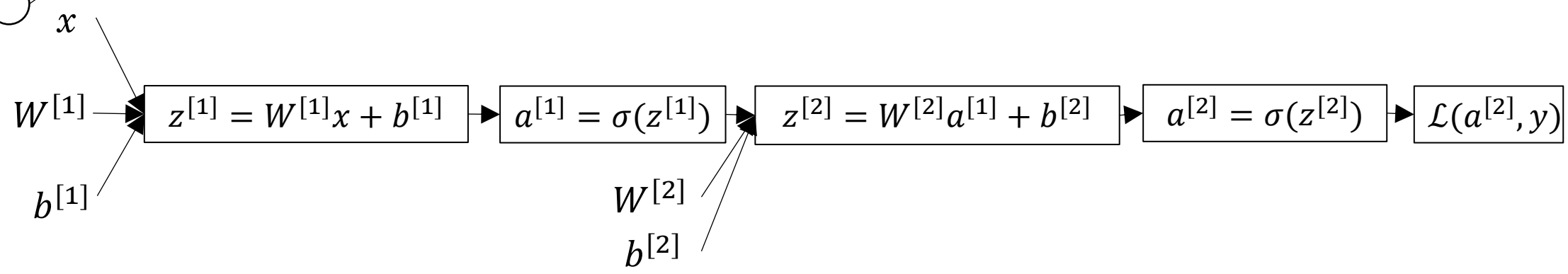
Neural Networks Overview

What is a Neural Network?







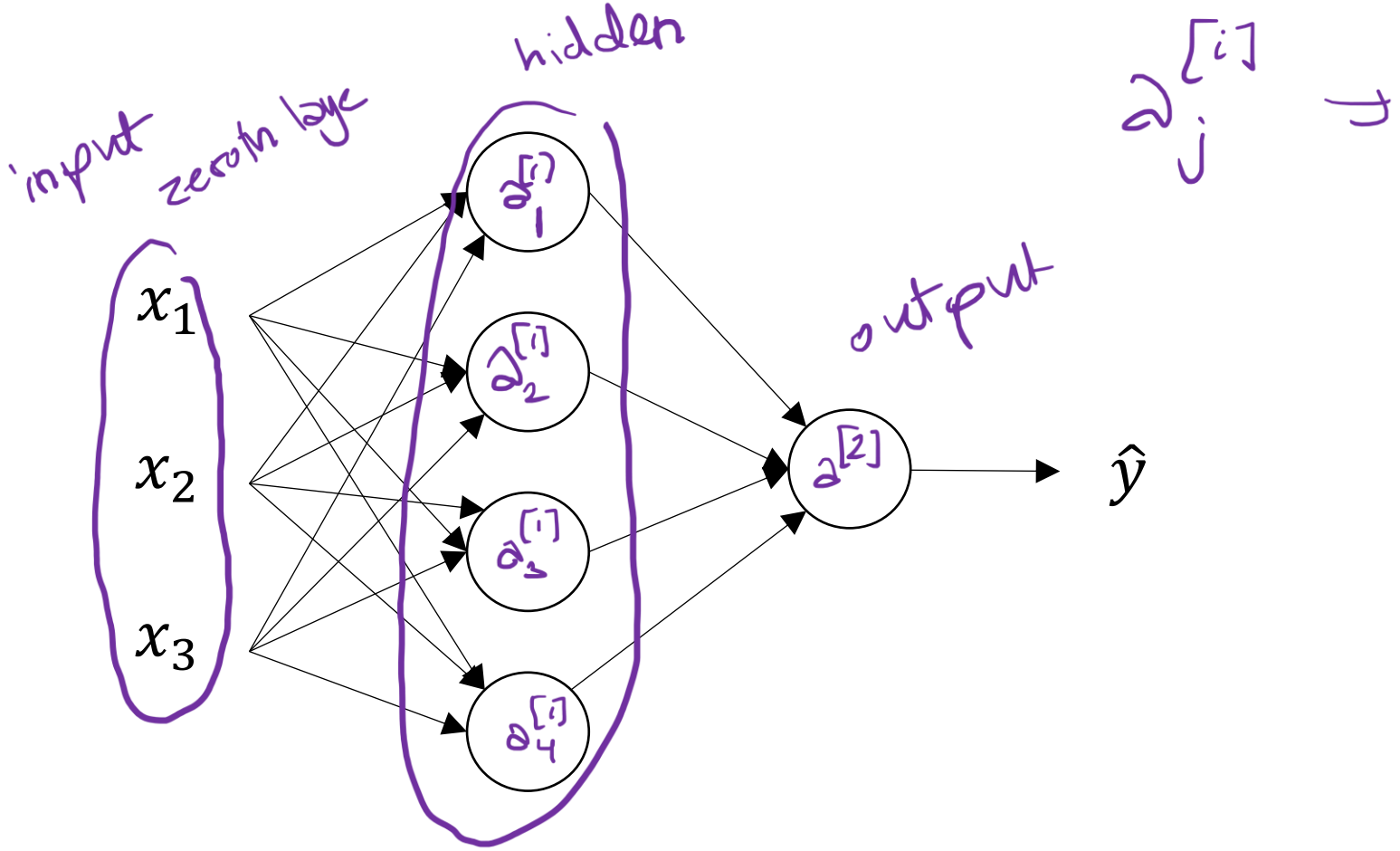


deeplearning.ai

One hidden layer Neural Network

Neural Network Representation

Neural Network Representation

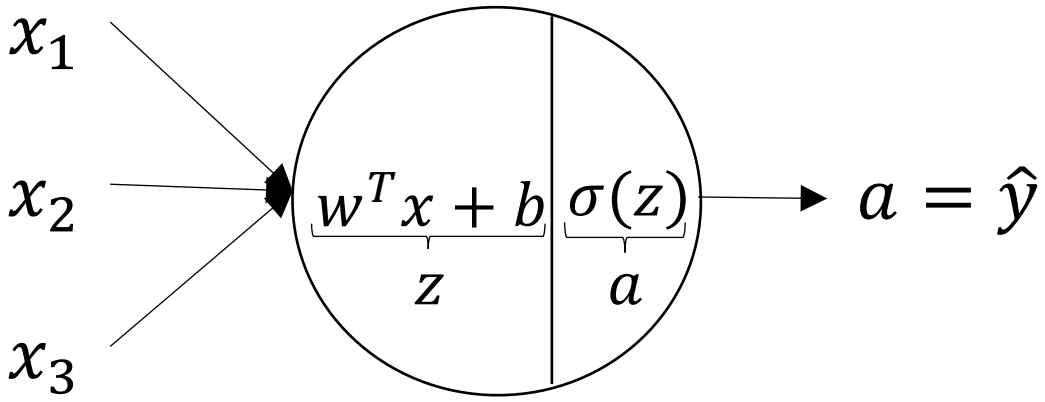


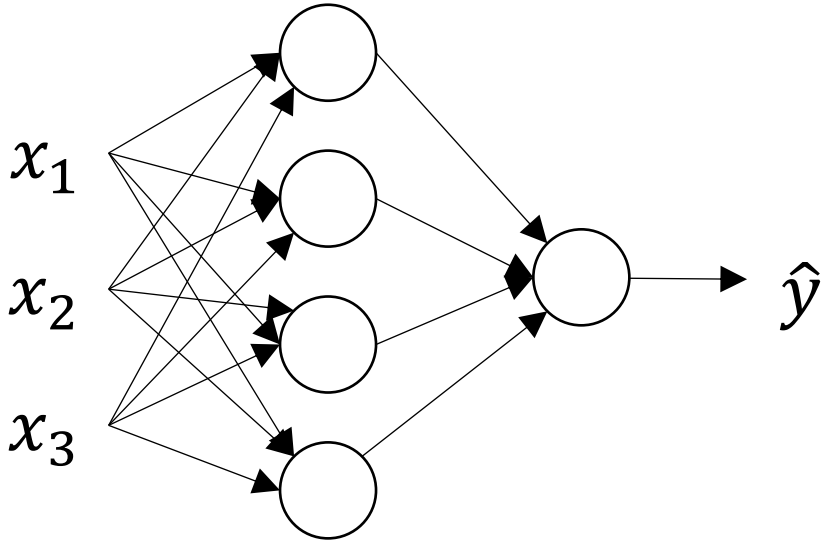
deeplearning.ai

One hidden layer Neural Network

Computing a Neural Network's Output

Neural Network Representation

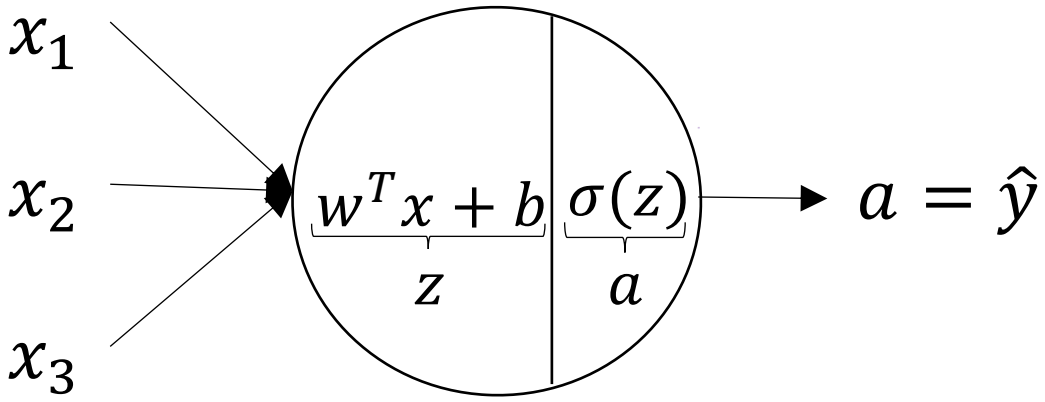




$$z = w^T x + b$$

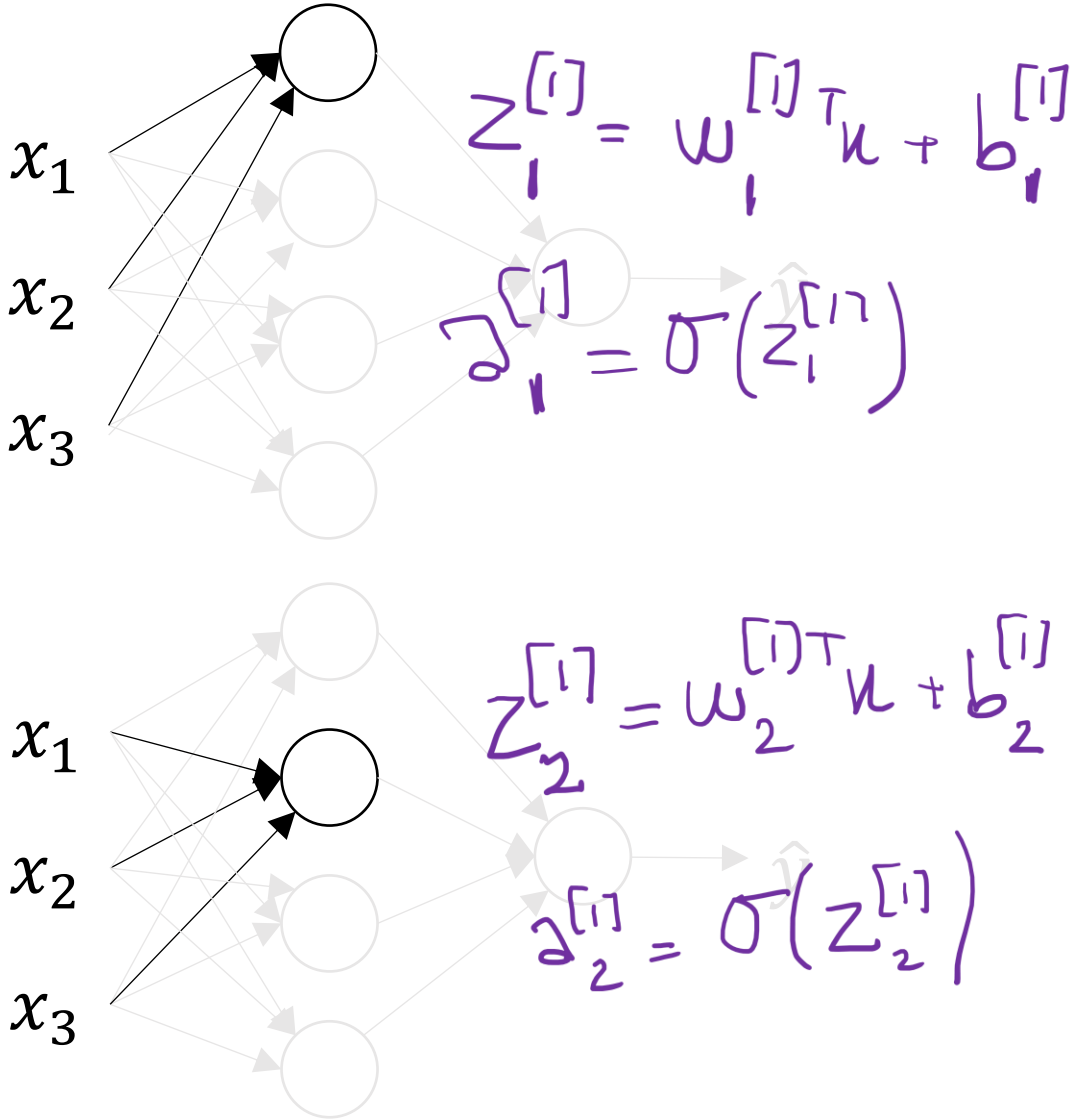
$$a = \sigma(z)$$

Neural Network Representation

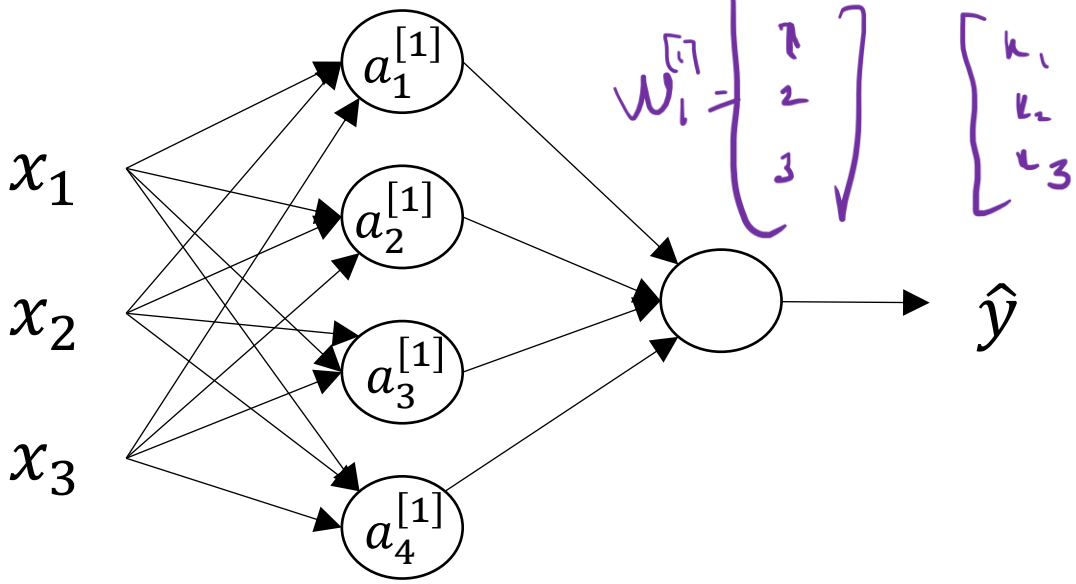


$$z = w^T x + b$$

$$a = \sigma(z)$$



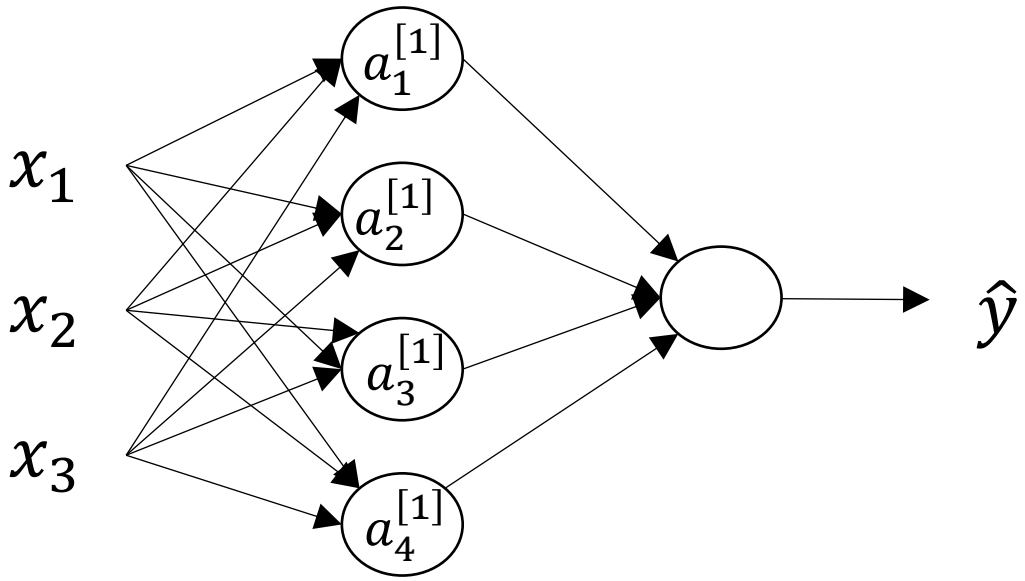
Neural Network Representation



$$
\begin{cases}
z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, & a_1^{[1]} = \sigma(z_1^{[1]}) \\
z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, & a_2^{[1]} = \sigma(z_2^{[1]}) \\
z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, & a_3^{[1]} = \sigma(z_3^{[1]}) \\
z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, & a_4^{[1]} = \sigma(z_4^{[1]})
\end{cases}
$$

$$
z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}
\quad
b^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}
\quad
a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix}
\quad
W^{[1]} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix}
$$

Neural Network Representation learning



Given input x:

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

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

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

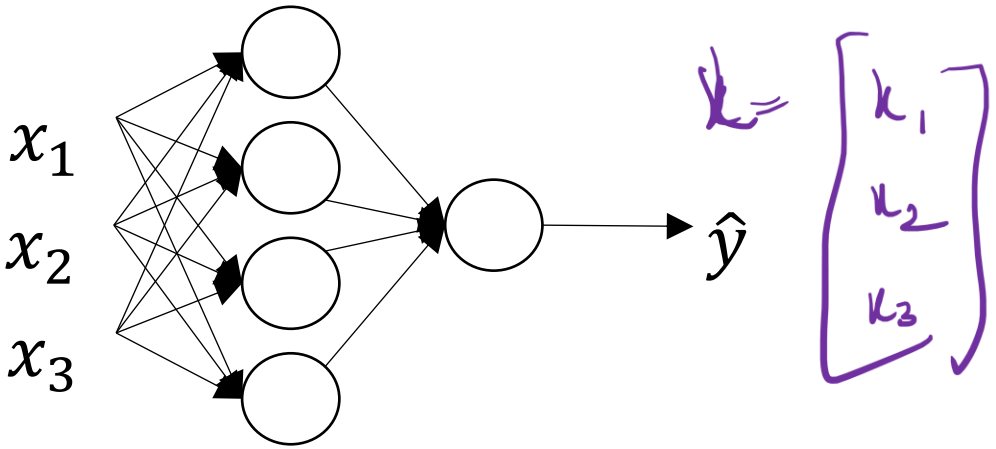
$$a^{[2]} = \sigma(z^{[2]})$$

deeplearning.ai

One hidden layer Neural Network

Vectorizing across multiple examples

Vectorizing across multiple examples



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

$$X = \begin{bmatrix} x^{(1)} & x^{(2)} & \cdots & x^{(m)} \end{bmatrix} \qquad Z^{[1]} = W^{[1]}X + b^{[1]}$$

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

Vectorizing across multiple examples


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

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

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

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

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

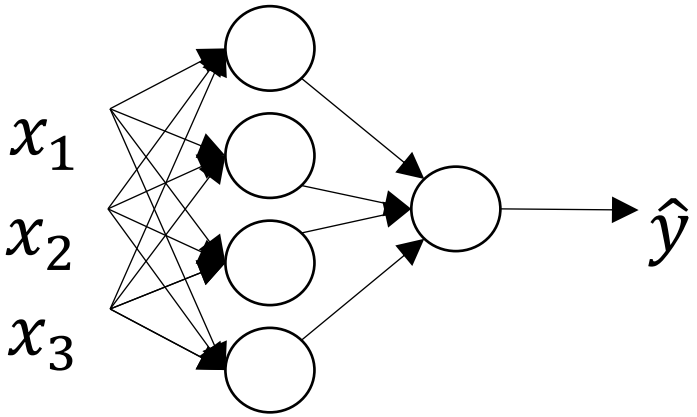
$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

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

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

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

Recap of vectorizing across multiple examples




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

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

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

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

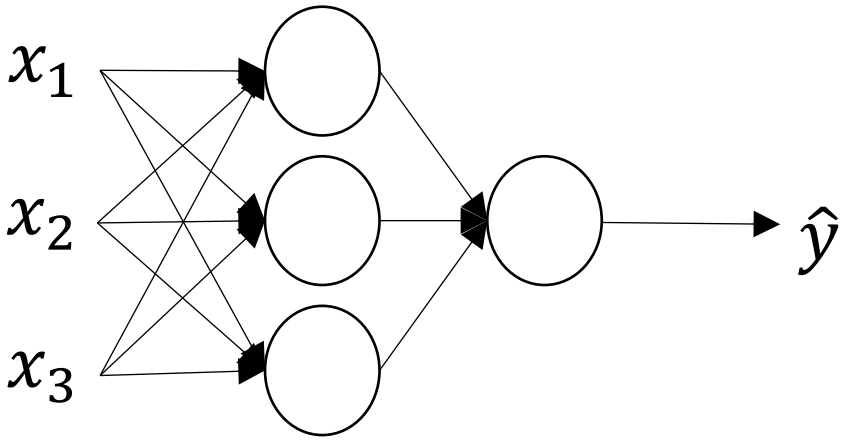
$$\mathrm{A}^{[1]} = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

deeplearning.ai

One hidden layer Neural Network

Activation functions

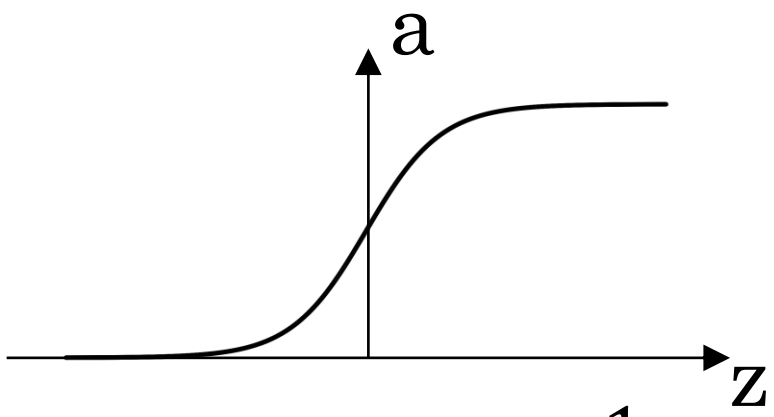
Activation functions



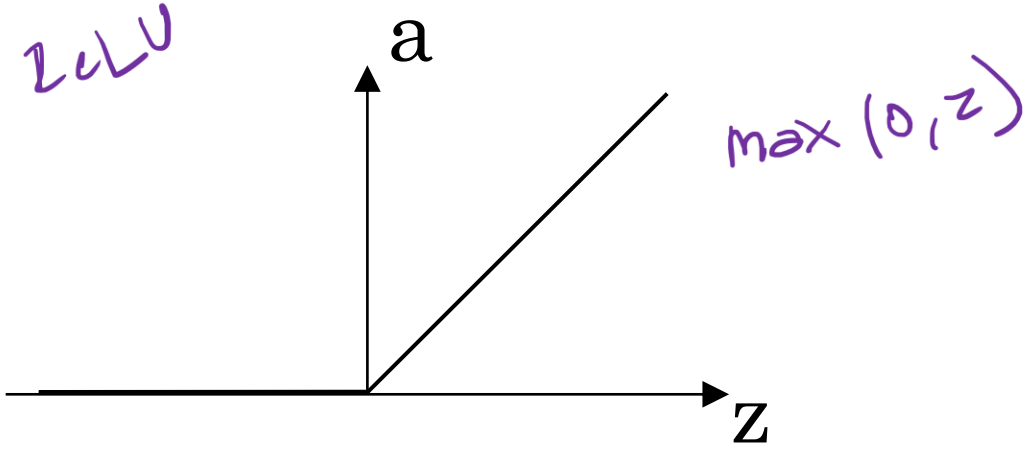
Given x:

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

Pros and cons of activation functions



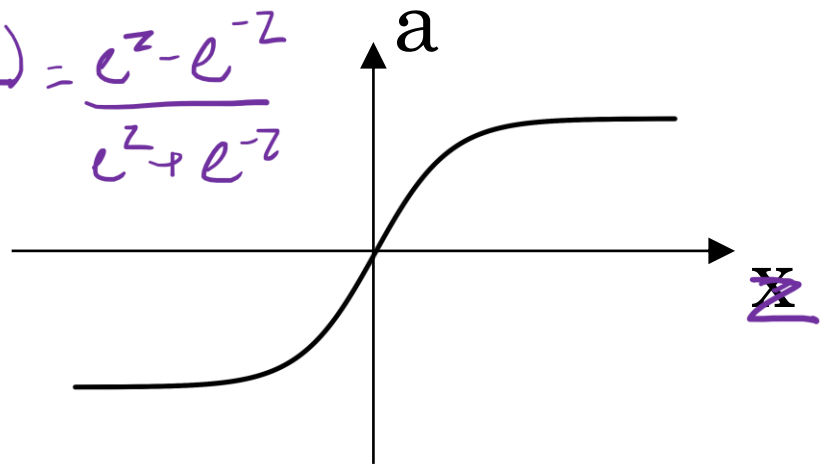
sigmoid: $a = \dfrac{1}{1 + e^{-z}}$



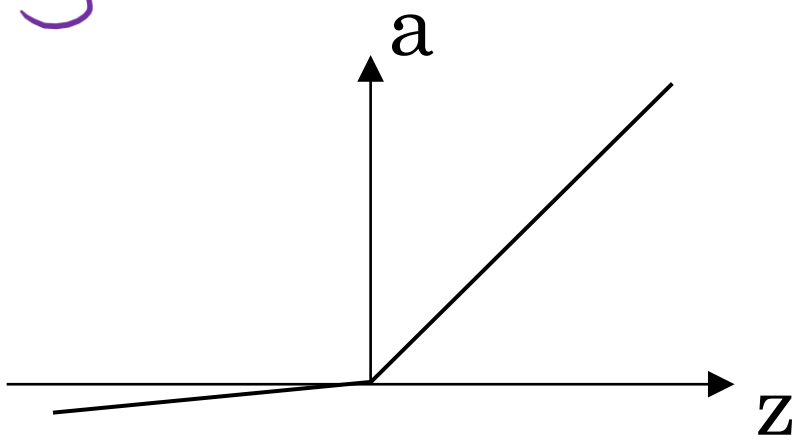
ReLU

max (0, z)

$\tanh(z) = \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$



Leaky ReLU

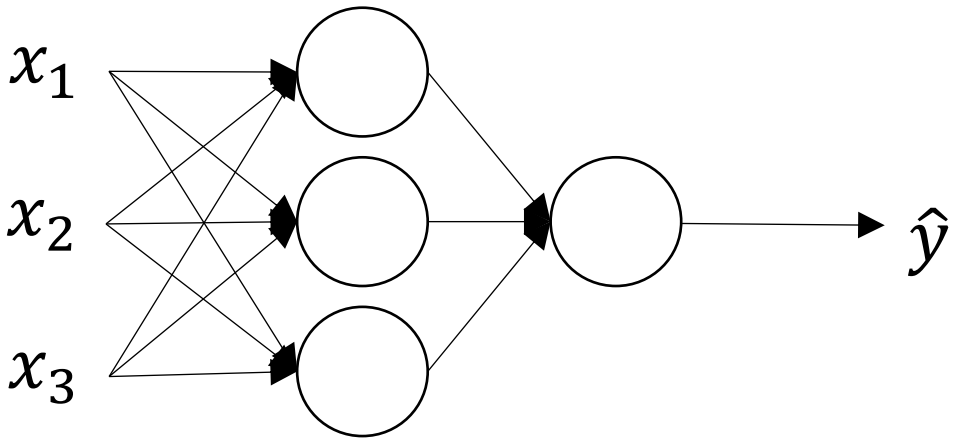


deeplearning.ai

One hidden layer Neural Network

Why do you need non-linear activation functions?

Activation function



Given x:

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

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

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

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

$a^{[2]} = z^{[2]}$

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

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

$= W^{[2]}(W^{[1]}x + b^{[1]}) + b^{[2]}$

$g(z) = z$ linear activation / identity act.

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

W' $\quad b'$

$a^{[2]} = W'x + b'$

deeplearning.ai

One hidden layer Neural Network

Gradient descent for neural networks

Gradient descent for neural networks

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

$$J = \text{LogLoss} = \frac{1}{m}\sum_{i=1}^{m} -\left(y_i \log p_i + (1-y_i)\log(1-p_i)\right)$$

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

$$W^{[1]} \leftarrow W^{[1]} - \alpha \frac{dJ}{dW^{[1]}}, \quad b^{[1]} \leftarrow b^{[1]} - \alpha \frac{dJ}{db^{[1]}}$$

$$W^{[2]} \leftarrow W^{[2]} - \alpha \frac{dJ}{dW^{[2]}}, \quad b^{[2]} \leftarrow b^{[2]} - \alpha \frac{dJ}{db^{[2]}}$$

$dW^{[2]}$ $db^{[2]}$

Repeat

Forward propagation

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

Compute $dW^{[1]}, db^{[1]}, dW^{[2]}, db^{[2]}$

$\downarrow$

J

Backward propagation

$$W^{[1]} \leftarrow W^{[1]} - \alpha dW^{[1]}, \quad b^{[1]} \leftarrow b^{[1]} - \alpha db^{[1]}$$

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

Forward propogation

g

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

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

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

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

Formulas for computing derivatives

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

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

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

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

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

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

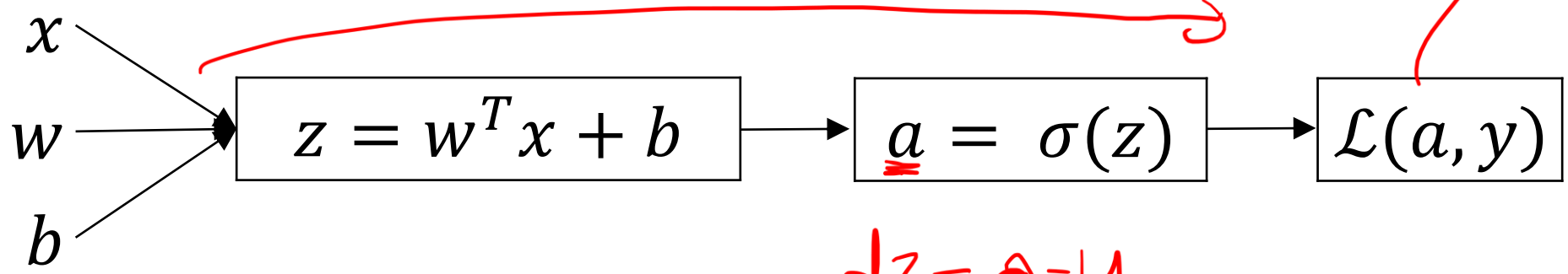
deeplearning.ai

One hidden layer Neural Network

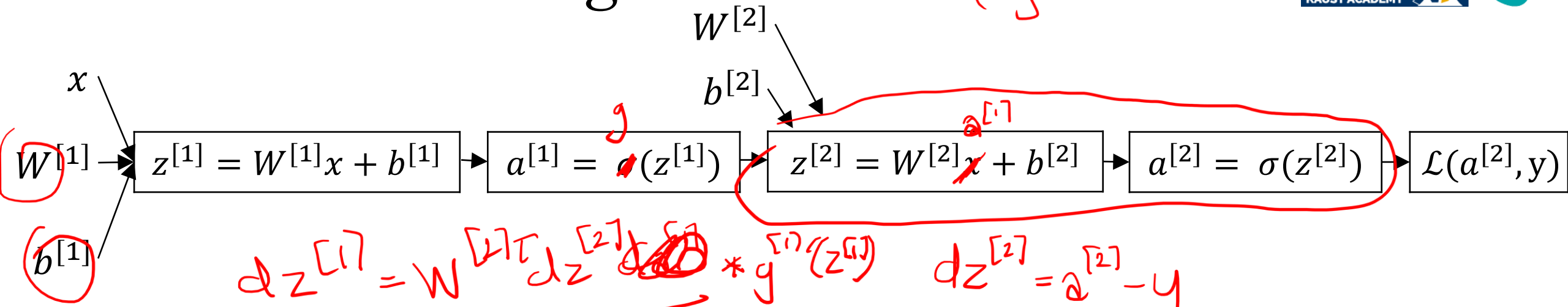
Backpropagation intuition

Computing gradients

Logistic regression



Neural network gradients



logistic

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

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

$$\begin{cases} dW^{[1]} = dz^{[1]} x^T \\ db^{[1]} = dz^{[1]} \end{cases}$$

$$\begin{cases} dW^{[2]} = dz^{[2]} a^{[1]T} \\ db^{[2]} = dz^{[2]} \end{cases}$$

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

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

$W^{[2]} \leftarrow W^{[2]} - \alpha\, dW^{[2]}$

$b^{[2]} \leftarrow b^{[2]} - \alpha\, db^{[2]}$

Andrew Ng

Summary of gradient descent

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

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

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

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

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

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

Andrew Ng

Summary of gradient descent

one training example

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

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

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

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

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

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

vectorized implementation of m training examples

$$X = [x^{(1)} \dots x^{(m)}]$$

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

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

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

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

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T \quad \frac{1}{m} \left[dz^{[1](1)} \dots dz^{[1](m)} \right] \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(m)} \end{bmatrix}$$

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

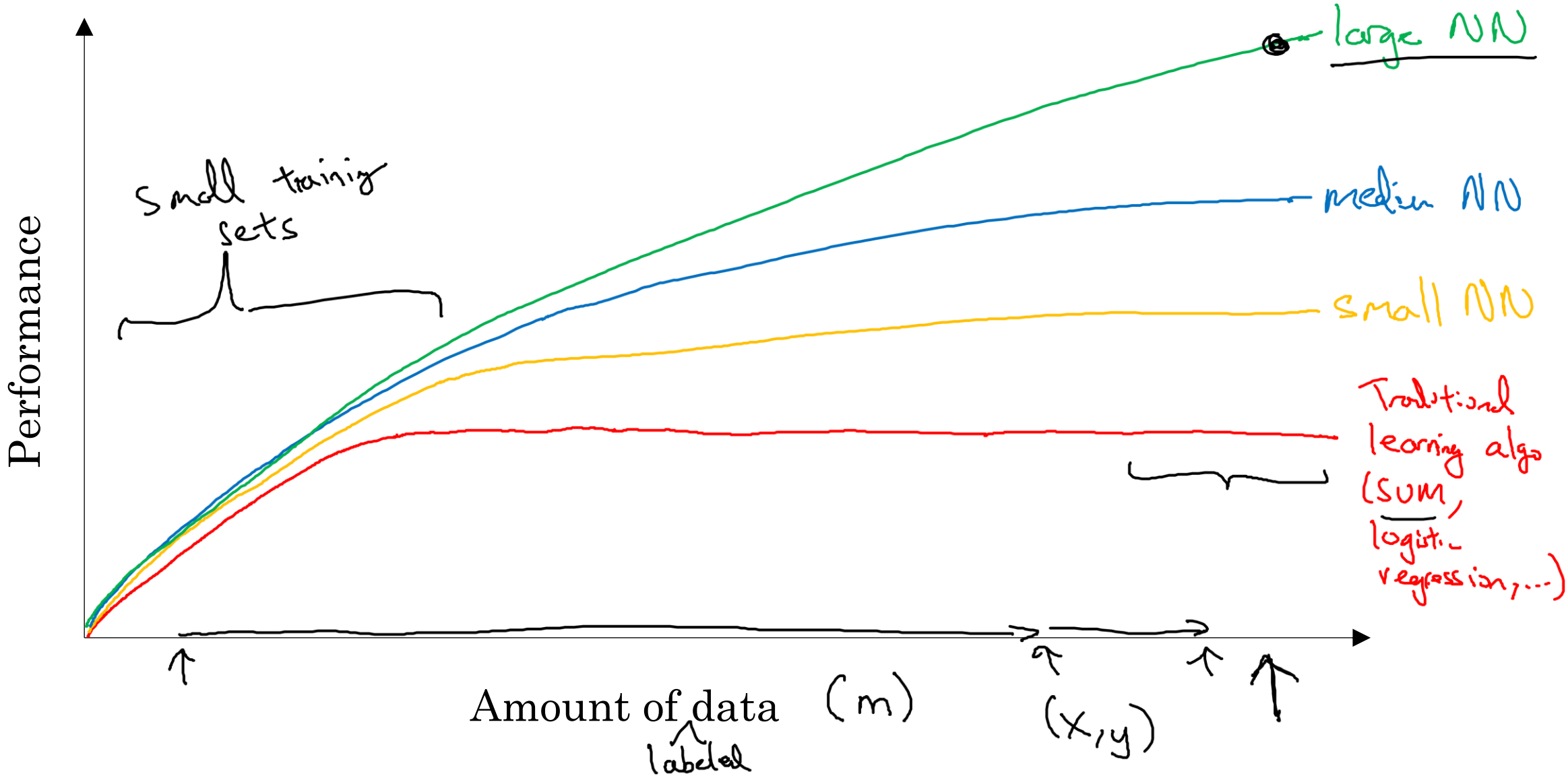
Andrew Ng

deeplearning.ai

Introduction to Neural Networks

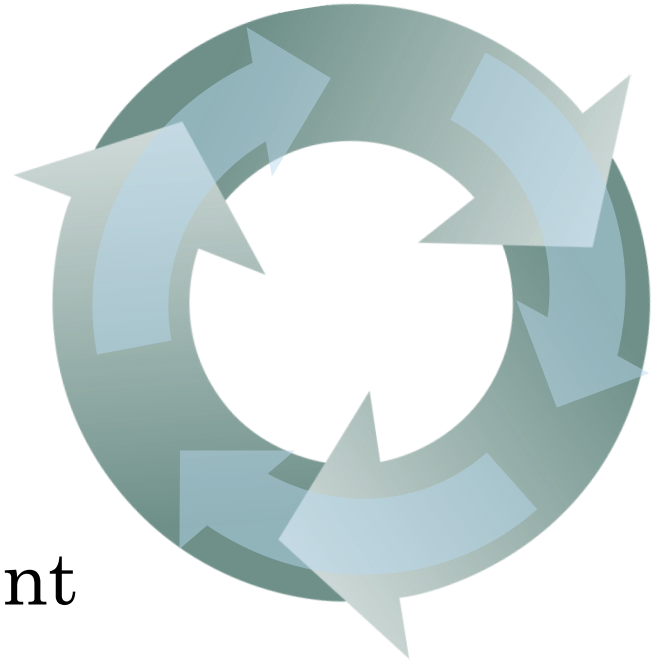
Why is Deep Learning taking off?

Scale drives deep learning progress



Scale drives deep learning progress

- Data
- Computation
- Algorithms

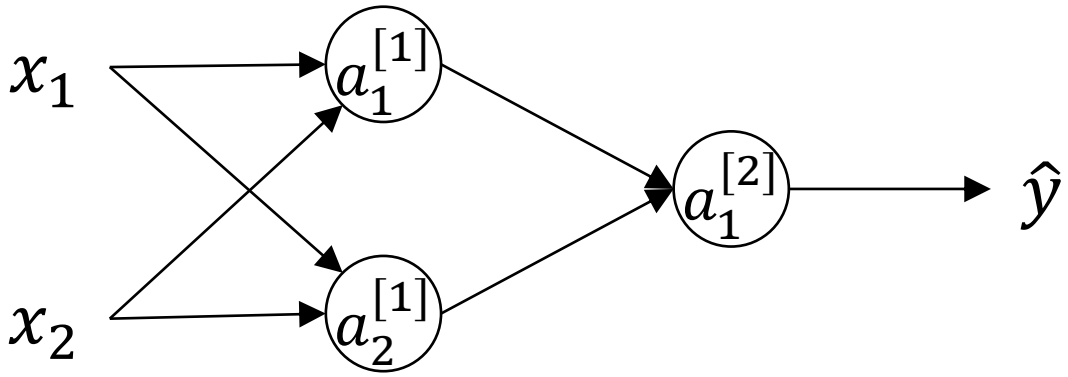


deeplearning.ai

One hidden layer Neural Network

Random Initialization

What happens if you initialize weights to zero?



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

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

$$W^{[2]} = [0 \ 0] \quad b^{[2]} = [0]$$

$a_1^{[1]} = a_2^{[1]} = g^{[1]}(0)$

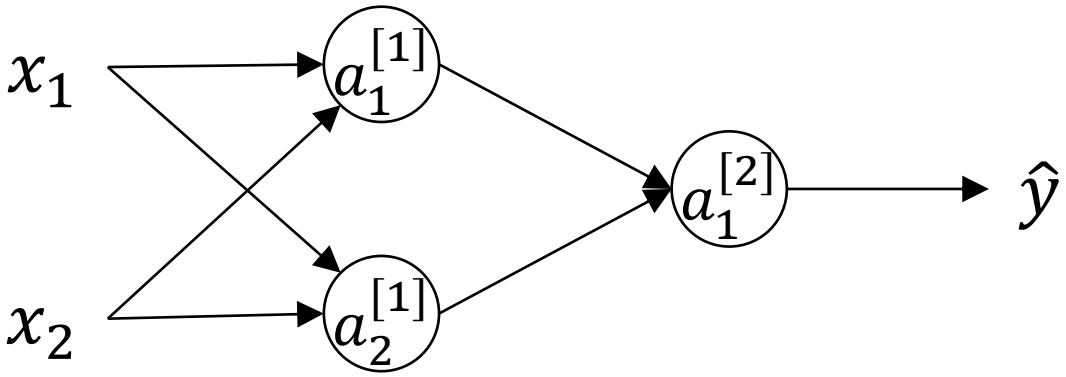
$\downarrow$

$dz_1^{[1]} = dz_2^{[1]} = 0$

$$dW = \begin{bmatrix} u & v \\ u & v \end{bmatrix}$$

$W^{[1]} \leftarrow W^{[1]} - dW$

Random initialization



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

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

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

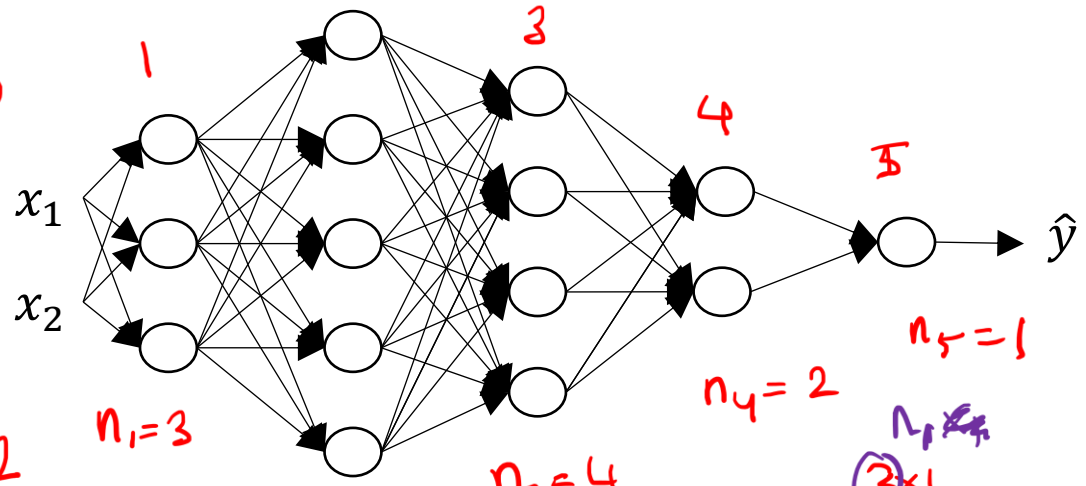
$b^{[2]} = np.zeros((1,1))$

deeplearning.ai

Deep Neural Networks

Getting your matrix dimensions right

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



$a^{[0]} = x$

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$n_0 = n_x = 2$

$a^{[0]}$

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

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

$n_1 \quad n_0 \quad n_0 \quad n_1$

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

$(3,1)$

n_1

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

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

$n_2 \quad n_2 \quad n_1 \quad n_2$

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

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

$n_2 \quad n_2$

ith layer

$z^{[i]} \rightarrow n_i \times 1$

$a^{[i]} \rightarrow n_i \times 1$

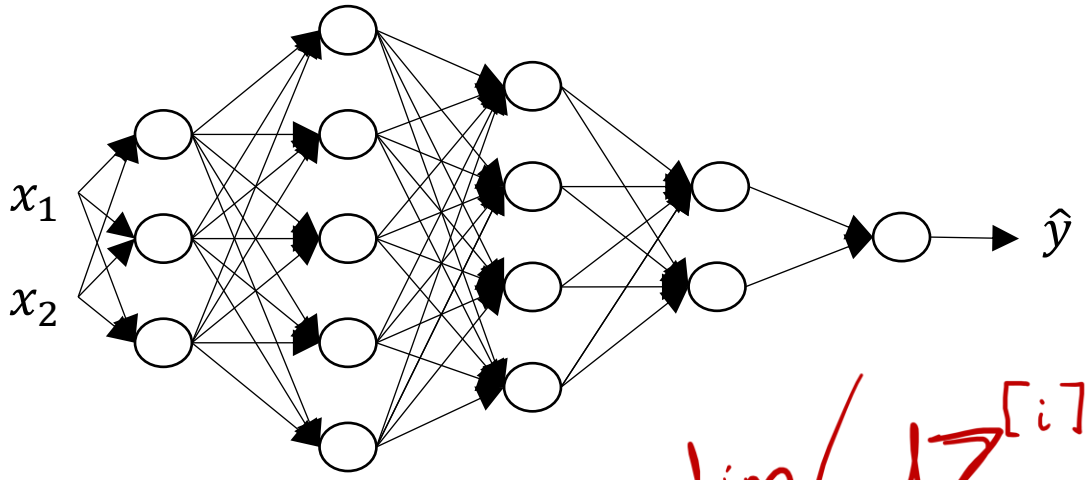
$b^{[i]} \rightarrow n_i \times 1$

$W^{[i]} \rightarrow n_i \times n_{i-1}$

Vectorized implementation

$\{x_i, y_i\}_{i=1}^{m}$

$n_i \times 1$



ith layer

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

$n_i \times m$

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

$n_i \times m$ $\quad$ $n_i \times 1$

$dim(dZ^{[i]}) = dim(Z^{[i]}) = n_i \times m$

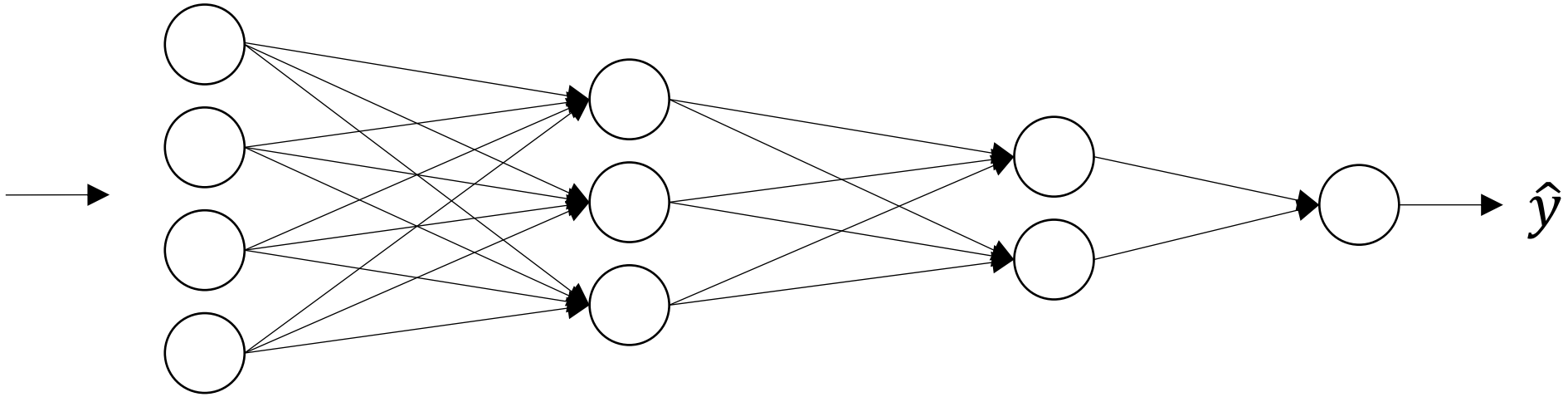
$dim(db^{[i]}) = dim(b^{[i]}) = n_i \times 1$

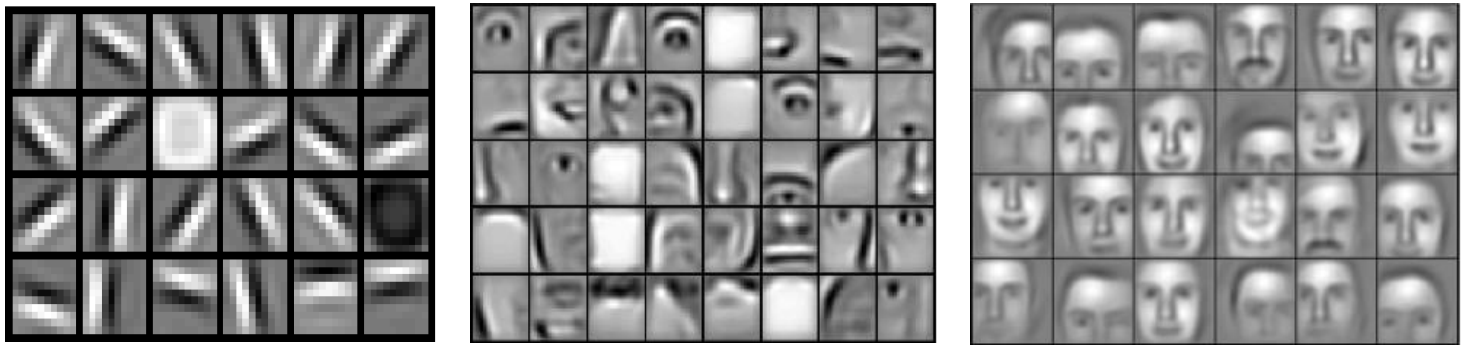
deeplearning.ai

Deep Neural Networks

Why deep representations?

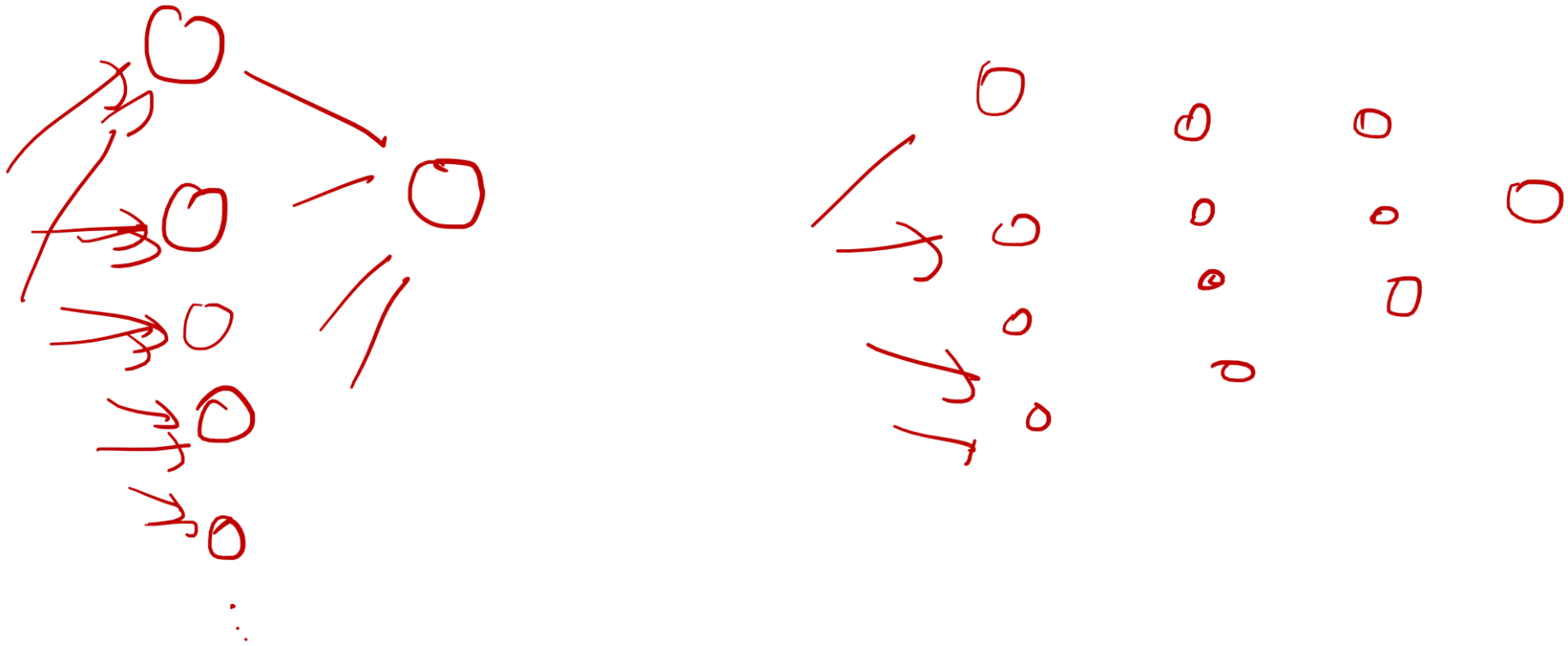
Intuition about deep representation





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.

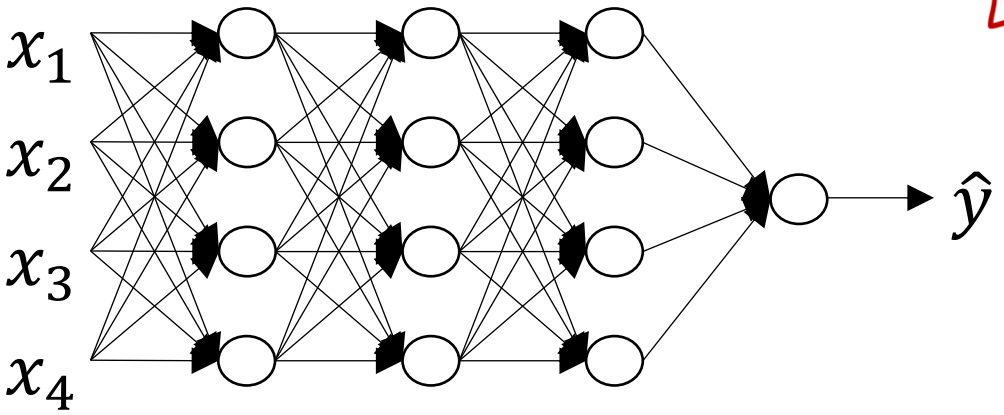


deeplearning.ai

Deep Neural Networks

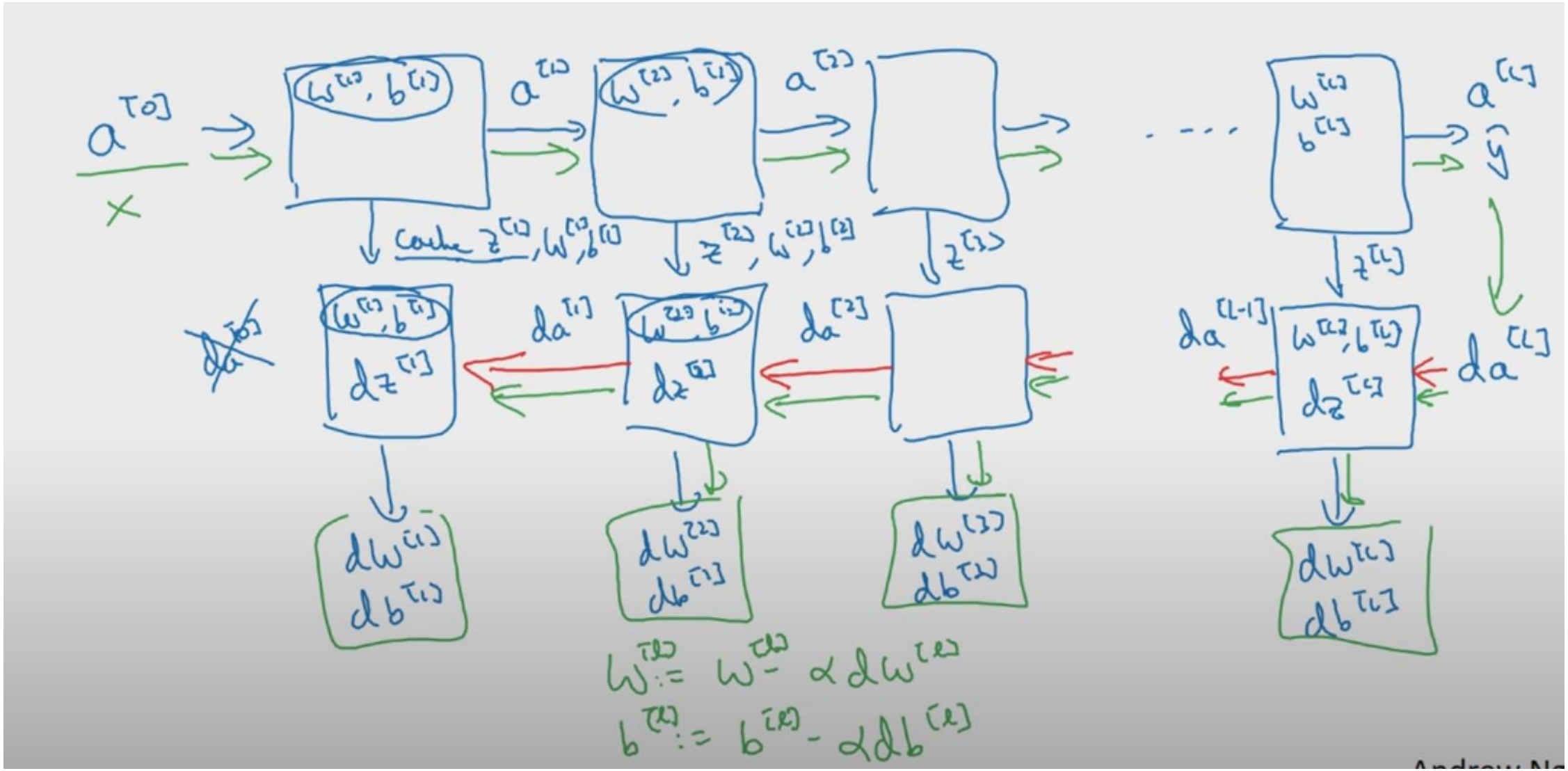
Building blocks of deep neural networks

Forward and backward functions



L layer

Forward and backward functions



deeplearning.ai

Deep Neural Networks

Forward and backward propagation

Forward propagation for layer l

L

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

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

one sample

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

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

Vectorized

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

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

Backward propagation for layer *l*

Input $da^{[l]}$

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

single sample

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

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

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

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

Vectorized

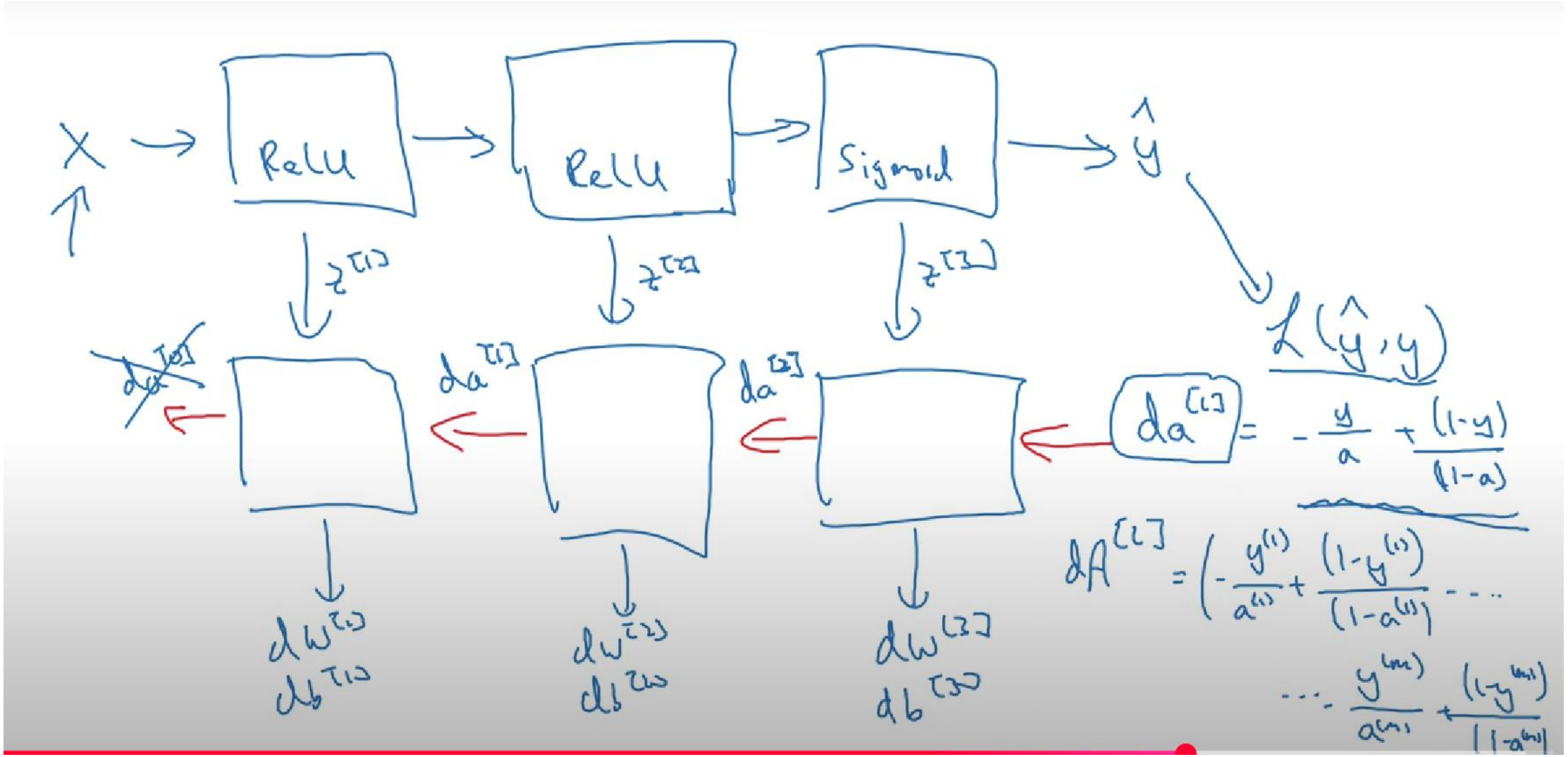
$$dZ^{[l]} = dA^{[l]} * g^{[l]\prime}(Z^{[l]})$$

$$dW^{[l]} = \frac{1}{m} dZ^{[l]} \cdot A^{[l-1]T}$$

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

$$dA^{[l-1]} = W^{[l]T} \cdot dZ^{[l]}$$

Summary


X
ReLU
ReLU
Sigmoid
ŷ
$z^{[1]}$
$z^{[2]}$
$z^{[3]}$
$\mathcal{L}(\hat{y}, y)$
$da^{[0]}$
$da^{[1]}$
$da^{[2]}$
$da^{[3]} = -\frac{y}{a} + \frac{(1-y)}{(1-a)}$
$dW^{[1]}$ $db^{[1]}$
$dW^{[2]}$ $db^{[2]}$
$dW^{[3]}$ $db^{[3]}$
$dA^{[L]} = \left(-\frac{y^{(1)}}{a^{(1)}} + \frac{(1-y^{(1)})}{(1-a^{(1)})} \cdots \right.$
$\left. \cdots - \frac{y^{(m)}}{a^{(m)}} + \frac{(1-y^{(m)})}{(1-a^{(m)})} \right)$