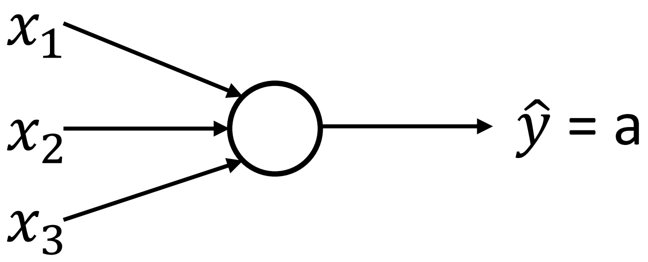
Deep learning

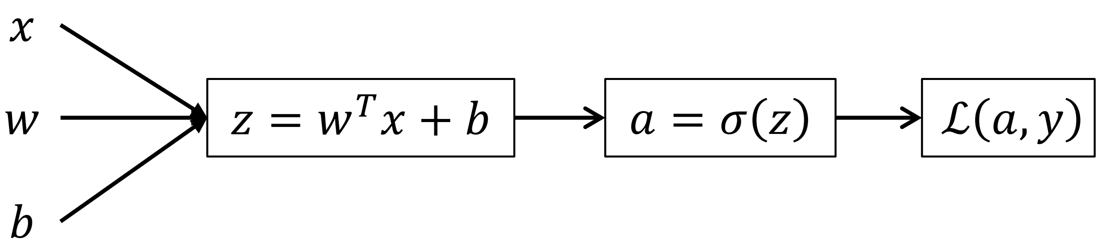
Dr. Aissa Boulmerka
a.boulmerka@centre-univ-mila.dz

2023-2024

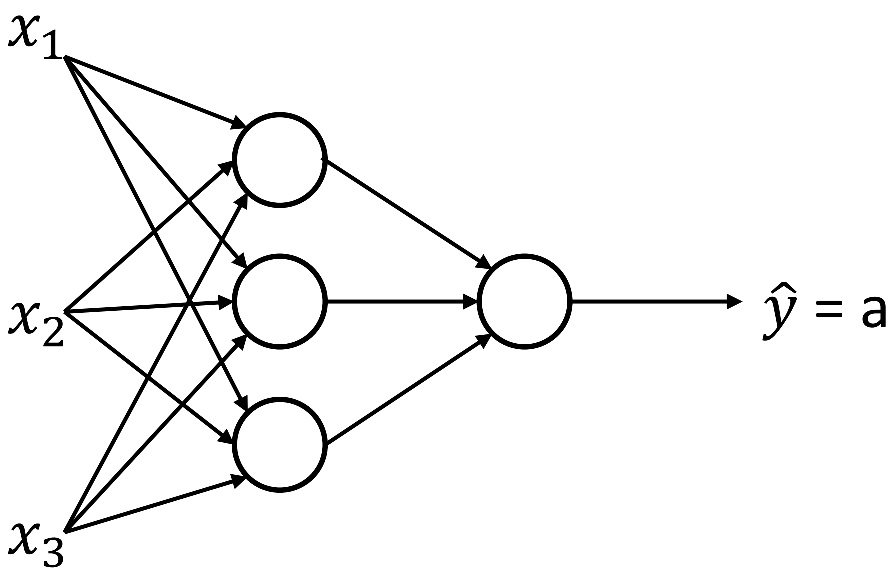
CHAPTER 2
SHALLOW NEURAL NETWORKS

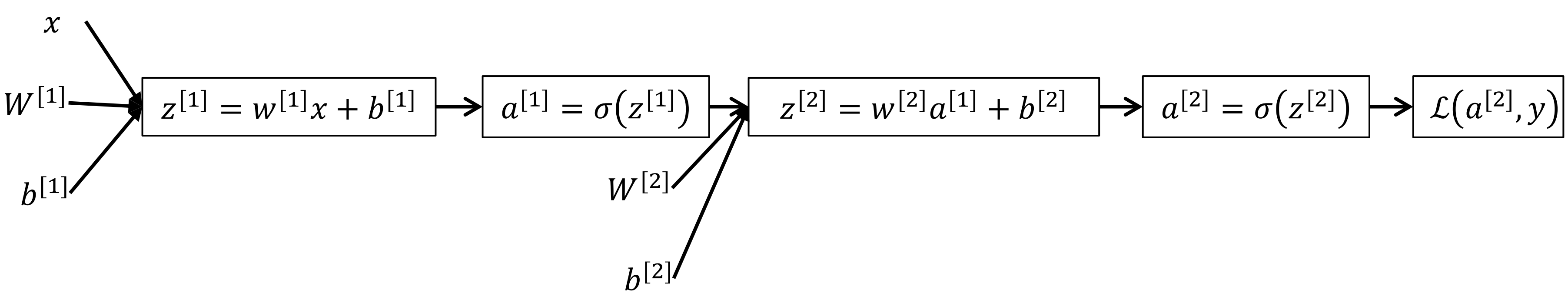
What is a Neural Network?





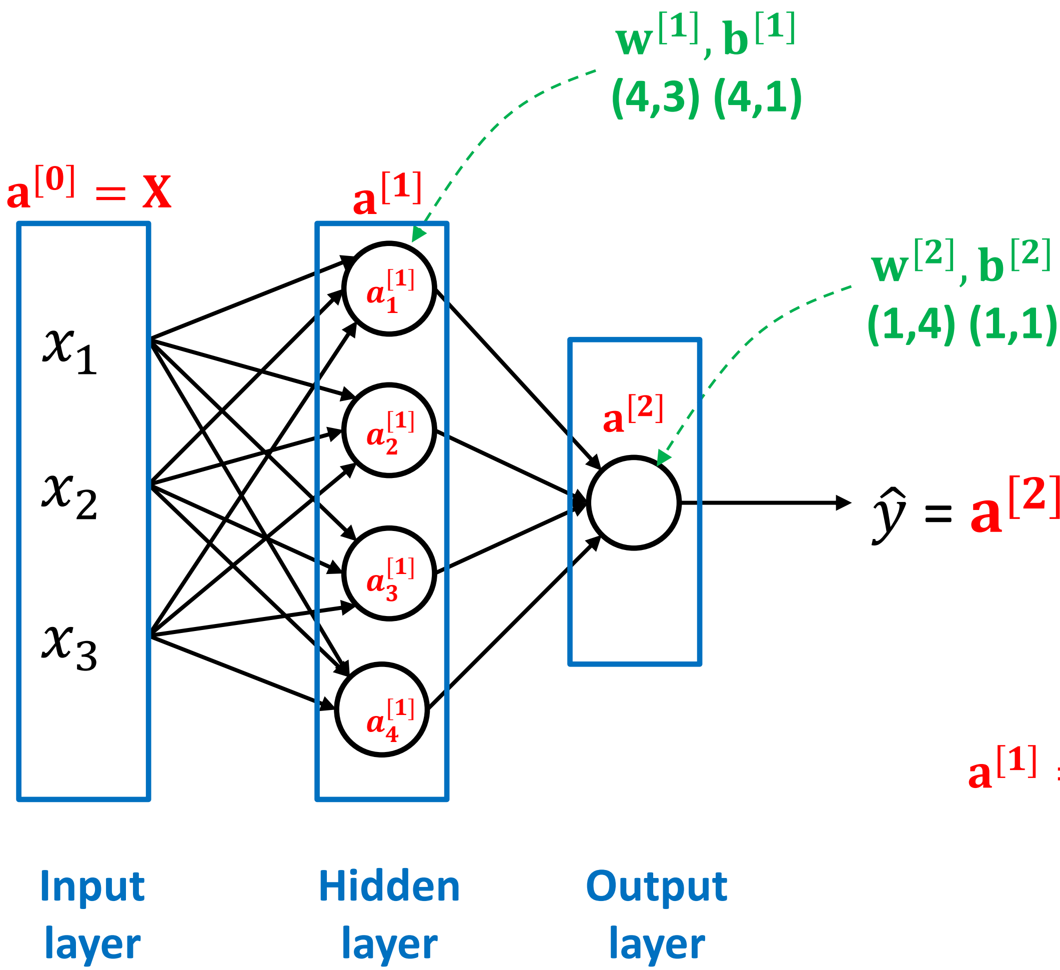
What is a Neural Network?





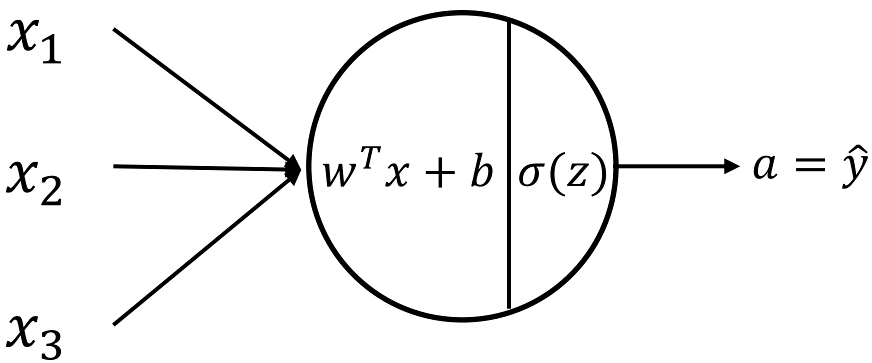
Neural Network Representation

2 layers neural network

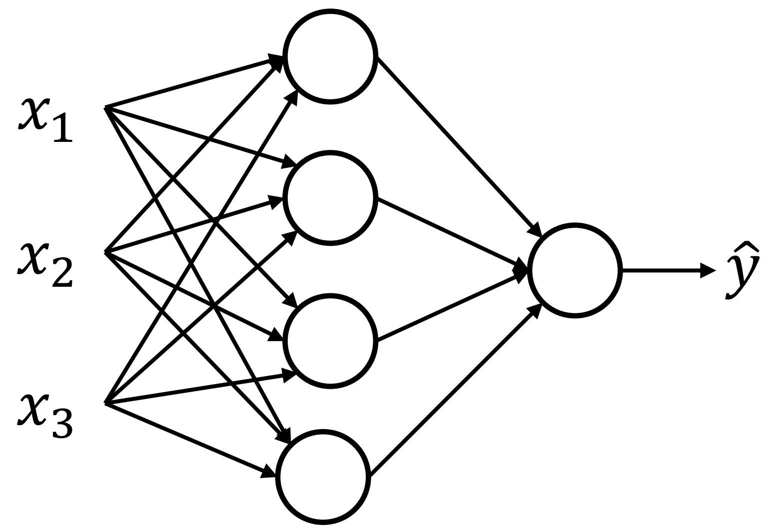


$$\mathbf{a}^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix}$$

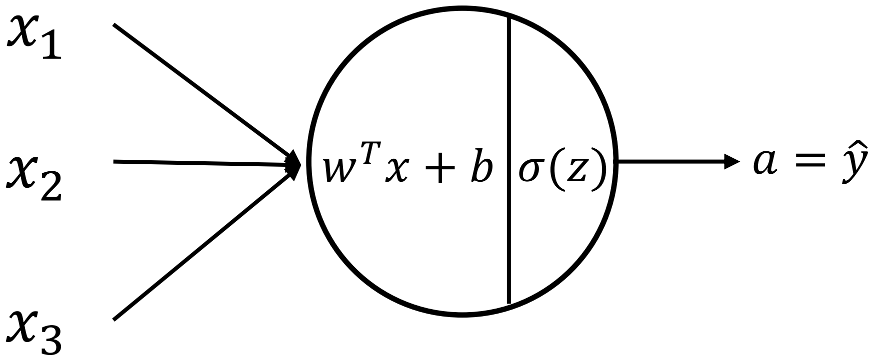
Neural Network Representation



$$z = w^T x + b$$
$$a = \sigma(z)$$

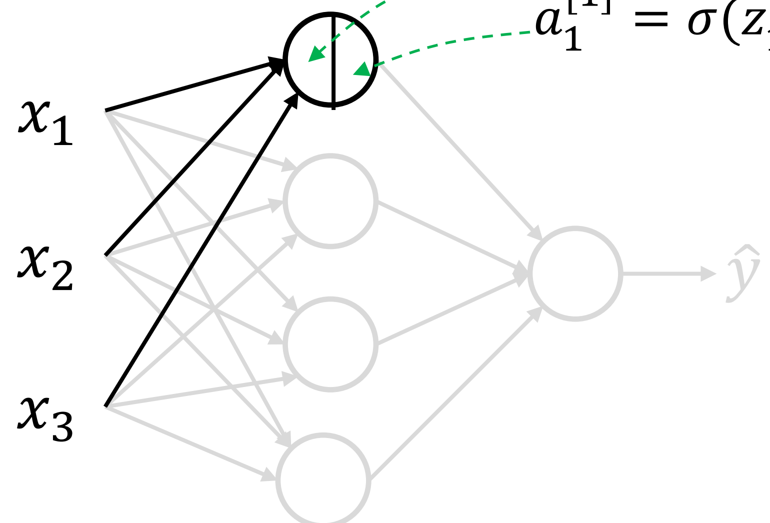


Neural Network Representation

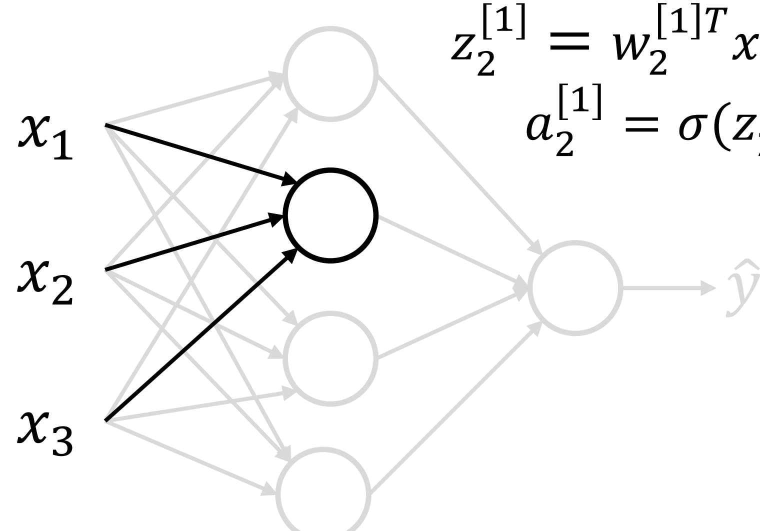


$$z = w^T x + b$$
$$a = \sigma(z)$$

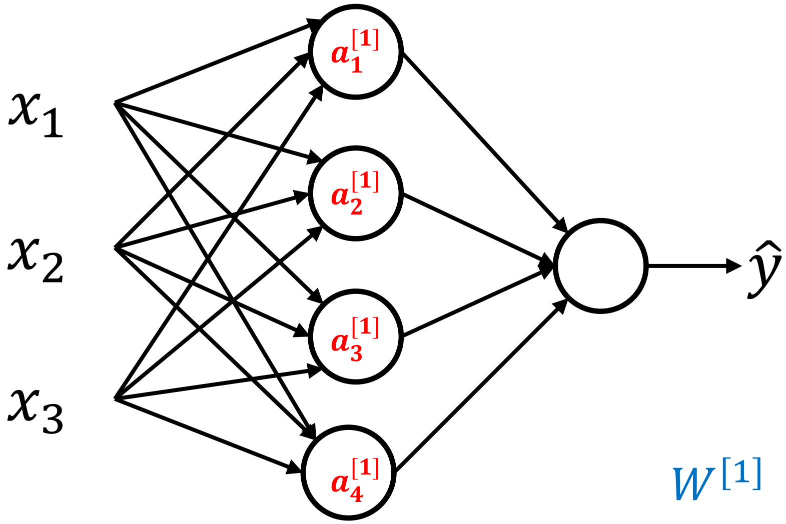
$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}$$
$$a_1^{[1]} = \sigma(z_1^{[1]})$$



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



Neural Network Representation

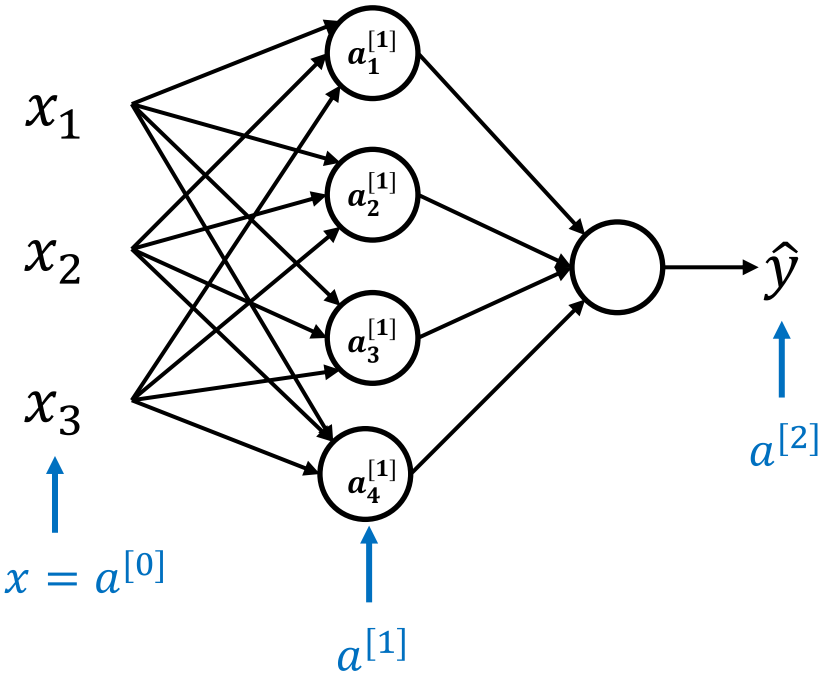


$$z_1^{[1]} = w_1^{[1]T}x + b_1^{[1]}, \quad a_1^{[1]} = \sigma\left(z_1^{[1]}\right)$$
$$z_2^{[1]} = w_2^{[1]T}x + b_2^{[1]}, \quad a_2^{[1]} = \sigma\left(z_2^{[1]}\right)$$
$$z_3^{[1]} = w_3^{[1]T}x + b_3^{[1]}, \quad a_3^{[1]} = \sigma\left(z_3^{[1]}\right)$$
$$z_4^{[1]} = w_4^{[1]T}x + b_4^{[1]},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

$$\underset{(4,1)}{z^{[1]}} = \underset{(4,3)}{\overset{W^{[1]}}{\begin{bmatrix} — w_1^{[1]T} — \\ — w_2^{[1]T} — \\ — w_2^{[1]T} — \\ — w_2^{[1]T} — \end{bmatrix}}} \underset{(3,1)}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}} + \underset{(4,1)}{\overset{b^{[1]}}{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}}} = \begin{bmatrix} w_1^{[1]T}x + b_1^{[1]} \\ w_2^{[1]T}x + b_2^{[1]} \\ w_3^{[1]T}x + b_3^{[1]} \\ w_4^{[1]T}x + b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \sigma\left(Z^{[1]}\right)$$

Neural Network Representation Learning



Given input x:

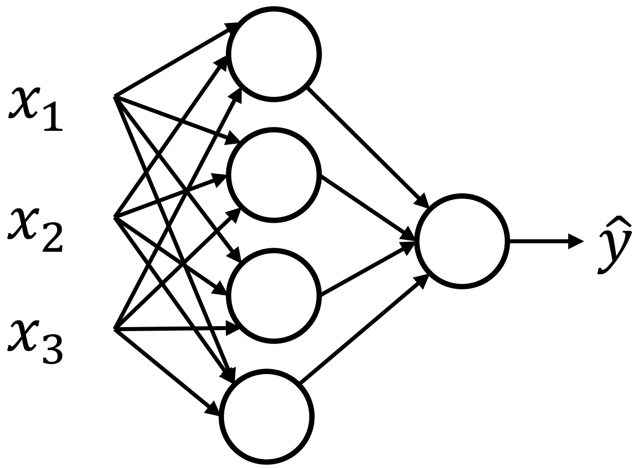
$$z^{[1]} = W^{[1]}x + b^{[1]}$$
(4, 1) (4, 3)(3, 1) (4, 1)

$$a^{[1]} = \sigma\left(z^{[1]}\right)$$
(4, 1) (4, 1)

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
(1, 1) (1, 4)(4, 1) (1, 1)

$$a^{[2]} = \sigma\left(z^{[2]}\right)$$
(1, 1) (1, 1)

For loop 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 -------------> $a^{[2]} = \hat{y}$

$x^{(1)}$ -------------> $a^{[2](1)} = \hat{y}^{(1)}$

$x^{(2)}$ -------------> $a^{[2](2)} = \hat{y}^{(2)}$

… … …

$x^{(m)}$ -------------> $a^{[2](m)} = \hat{y}^{(m)}$

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

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

$a^{[2](i)}$

Layer 2 ------> [2]

Example (i) ------> (i)

Vectorizing across multiple examples

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

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

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

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

$(\boldsymbol{n_x}, \boldsymbol{m})$

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

training examples

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

hidden units

Justification for vectorized implementation

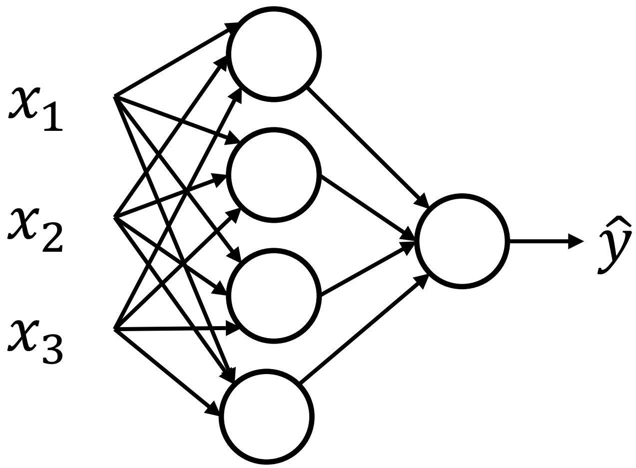
$$Z^{[1](1)} = W^{[1]}x^{(1)} + b^{[1]} \quad , \quad Z^{[1](2)} = W^{[1]}x^{(2)} + b^{[1]} \quad , \quad Z^{[1](3)} = W^{[1]}x^{(3)} + b^{[1]}$$

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

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

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

Recap of vectorizing across multiple examples




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

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

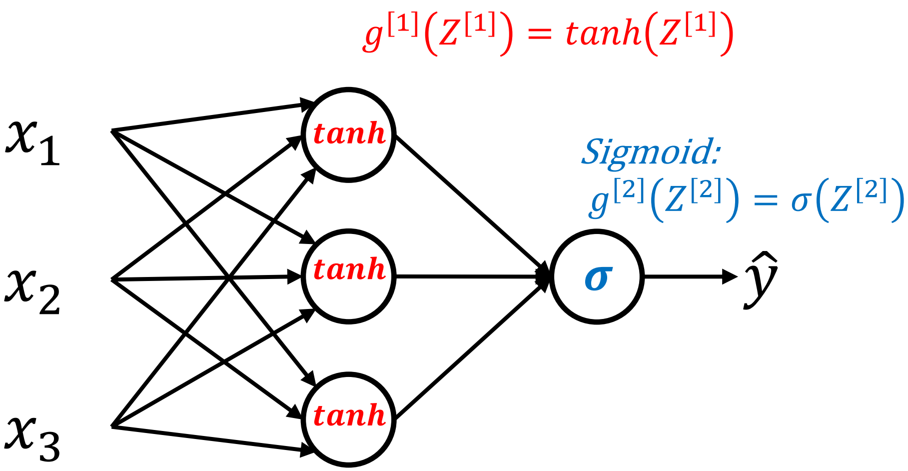
$$X = \begin{bmatrix} | & | & \cdots & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & \cdots & | \end{bmatrix}$$

$A^{[0]}$

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

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

Activation functions

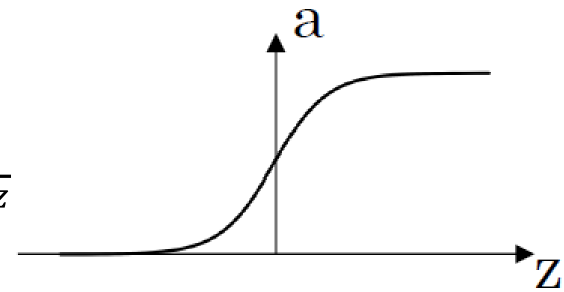


Given $\boldsymbol{X}$**:**

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

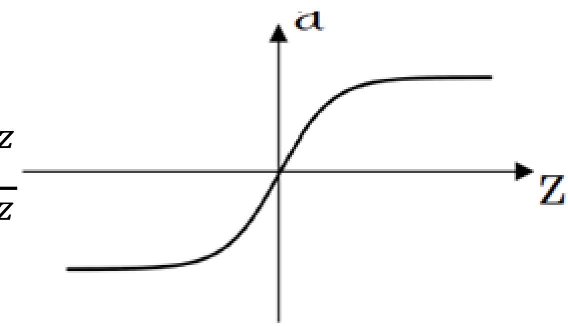
Sigmoid:

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



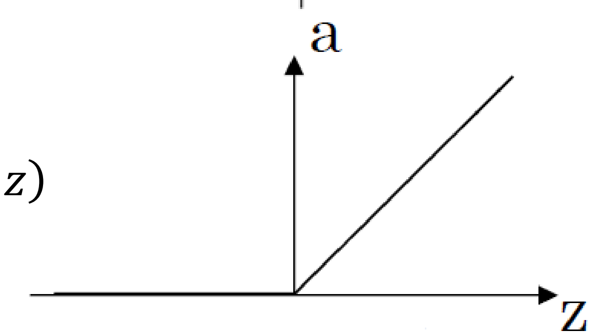
tanh:

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



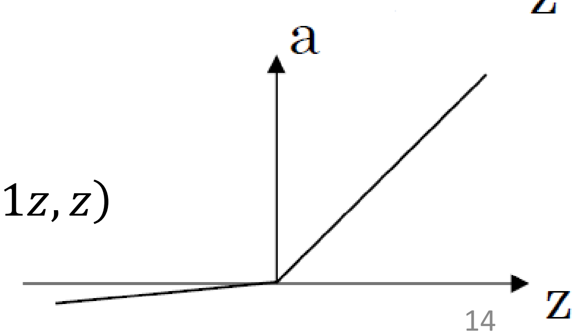
ReLU:

$a = \max(0, z)$



Leaky ReLU:

$a = \max(0.01z, z)$

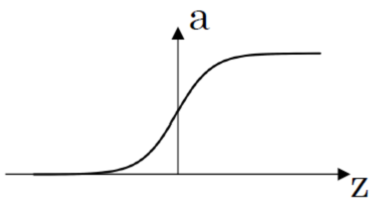


Pros and cons of activation functions

Sigmoid activation function:

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

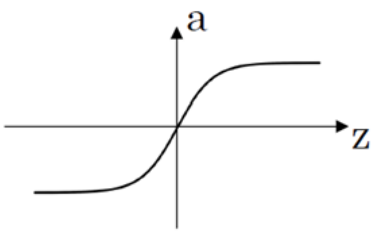
- Never use this, except for the output layer.
- if you are doing binary classification, or maybe almost never use this.



tanh activation function:

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

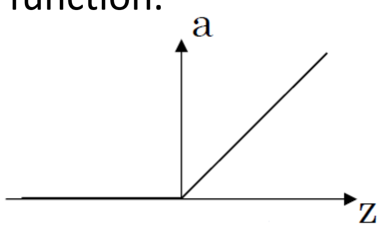
- The tanh is much strictly superior.



ReLU activation function:

$$a = \max(0, z)$$

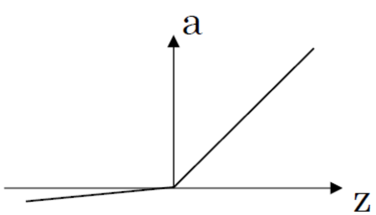
- The default and the most commonly used activation function is the ReLU.
- So if you're not sure what else to use, use the ReLU function.



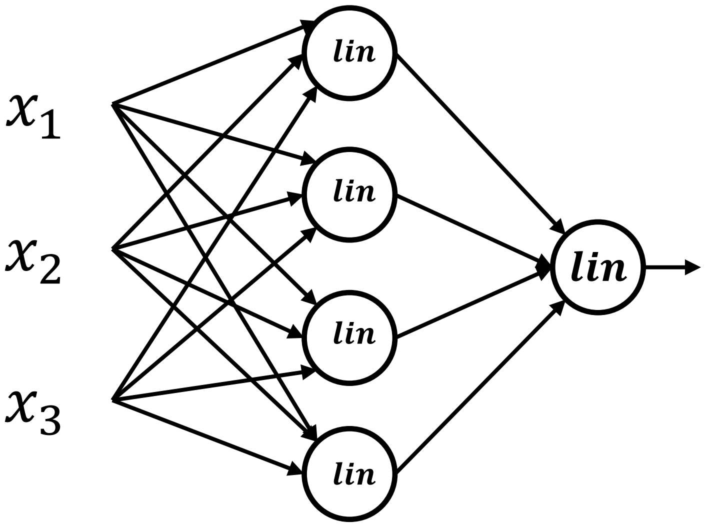
Leaky ReLU activation function:

$$a = \max(0.01z, z)$$

- You can also try the leaky ReLU function.



Why do you need non linear activation functions?



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

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

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

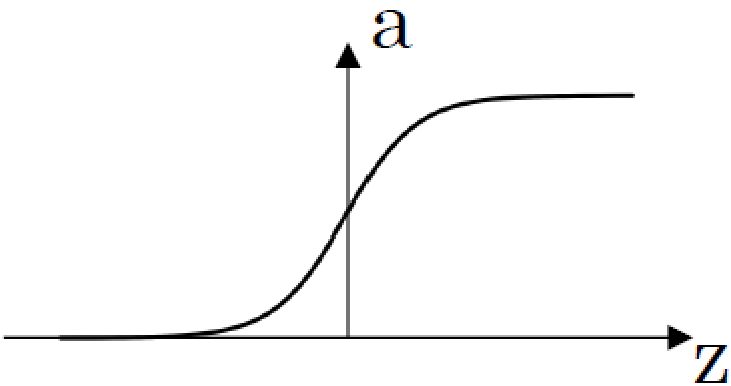
Given X:

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

Linear activation function: $g(z) = z$

Derivatives of activation functions

Sigmoid activation function:



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

$$g'(z) = \frac{d}{dz}g(z) = \text{slope of } g(z) \text{ at } z$$

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

$$= g(z)\big(1 - g(z)\big)$$

$$a = g(z), \qquad g'(z) = a(1-a)$$

- $z = 10 \implies g(z) \approx 1$

$$\implies \frac{d}{dz}g(z) \approx 1(1-1) \approx 0$$

- $z = -10 \implies g(z) \approx 0$

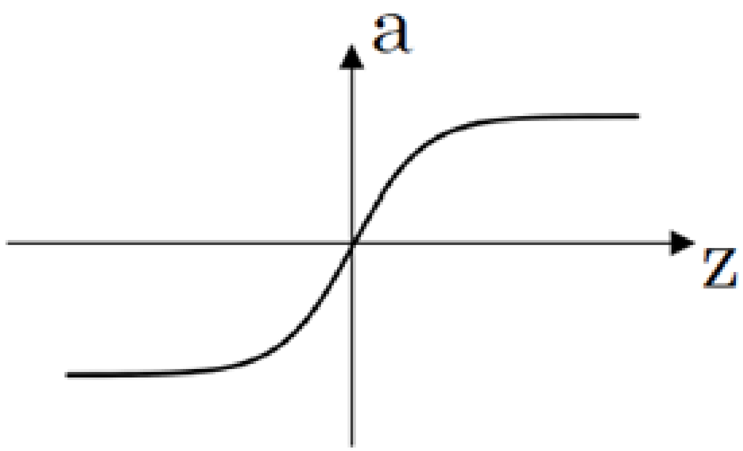
$$\implies \frac{d}{dz}g(z) \approx 0(1-0) \approx 0$$

- $z = 0 \implies g(z) = 1/2$

$$\implies \frac{d}{dz}g(z) = \frac{1}{2}\left(1 - \frac{1}{2}\right) = 1/4$$

Derivatives of activation functions

Tanh activation function:



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

- $z = 10 \implies tanh(z) \approx 1$
  $\implies g'(z) \approx 0$
- $z = -10 \implies g(z) \approx -1$
  $\implies g'(z) \approx 0$
- $z = 0 \implies g(z) = 0$
  $\implies g'(z) = 1$

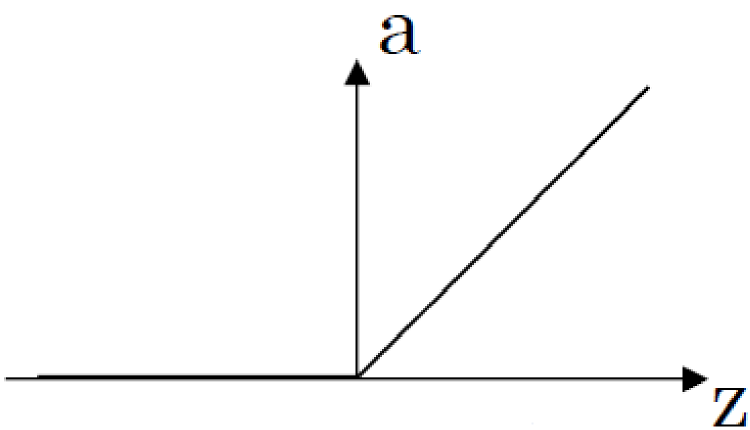
$$g'(z) = \frac{d}{dz} g(z) = \text{slope of } g(z) \text{ at } z$$

$$= 1 - (\tanh(z))^2$$

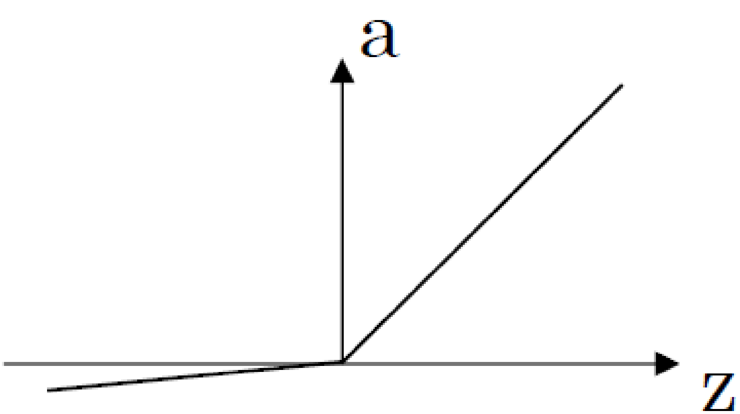
$$a = g(z), \qquad g'(z) = 1 - a^2$$

Derivatives of activation functions

ReLU and Leaky ReLU :



$$g(z) = max(0, z)$$



$$a = max(0.01z, z)$$

$$g'(z) = \begin{cases} 1 \ si \ z \geq 0 \\ 0 \ si \ z < 0 \end{cases}$$

$$g'(z) = \begin{cases} 1 \ si \ z \geq 0 \\ 0.01 \ si \ z < 0 \end{cases}$$

Gradient descent for neural networks

Parameters :

$W^{[1]}$	$b^{[1]}$	$W^{[2]}$	$b^{[2]}$
$\left(n^{[1]}, n^{[0]}\right)$	$\left(n^{[1]}, 1\right)$	$\left(n^{[2]}, n^{[1]}\right)$	$\left(n^{[2]}, 1\right)$

with $n_x = n^{[0]}, n^{[1]}, n^{[2]} = 1$

Cost function:

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

Gradient descent:

Repeat {

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

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

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

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

$$dW^{[2]} = \frac{\partial J}{\partial W^{[2]}}, db^{[2]} = \frac{\partial J}{\partial b^{[2]}}$$

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

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

}

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:

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

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

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

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

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

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

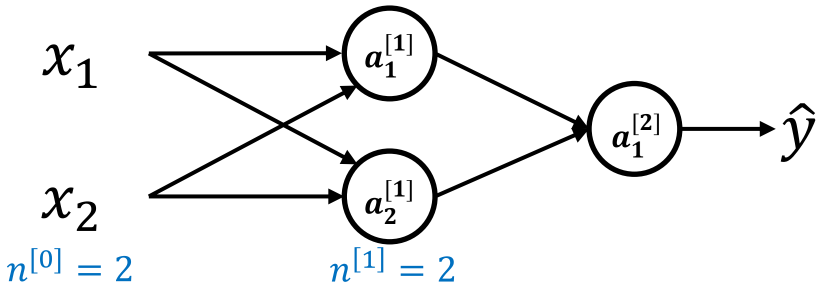
Element wise product

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

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

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

What happens if you initialize weights to zero?



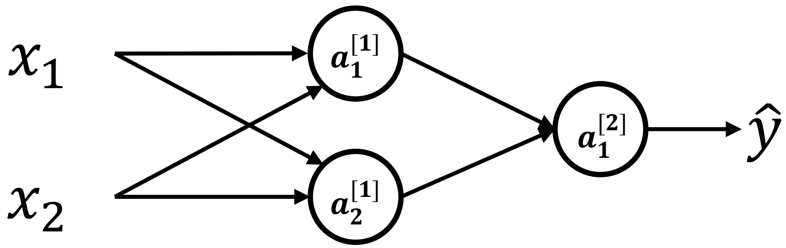
$$W^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$\boldsymbol{a_1^{[1]}} = \boldsymbol{a_2^{[1]}}$(symmetric) $\quad \boldsymbol{dz_1^{[1]}} = \boldsymbol{dz_2^{[1]}}$

$$dw = \begin{bmatrix} u & v \\ u & v \end{bmatrix} \qquad W^{[1]} = W^{[1]} - \alpha dw$$

- The bias terms b can be initialized by 0, but initializing W to all 0s is a problem:
  - The two activations $\boldsymbol{a_1^{[1]}}$ and $\boldsymbol{a_2^{[1]}}$ will be the same, because both of these hidden units are computing exactly the same function.
  - After every single iteration of training the two hidden units are still computing exactly the same function.

Random initialization



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

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

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

$$b^{[1]} = 0$$

Vectorization demo

Entrée [3]:

```python
import time

a = np.random.rand(10000000)
b = np.random.rand(10000000)

tic = time.time()
c = np.dot(a,b)
toc = time.time()

print("{:.6f}".format(c))
print("Vectorized version:" + str(1000*(toc-tic))+"ms")

tic = time.time()
c = 0
for i in range(10000000):
    c = c + a[i]*b[i]
toc = time.time()

print("{:.6f}".format(c))
print("For loop:" + str(1000*(toc-tic)) + "ms")
```

```
2499584.436480
Vectorized version:4.6520233154296875ms
2499584.436480
For loop:4441.97678565979ms
```

References

- Andrew Ng. Deep learning. Coursera.
- Geoffrey Hinton. Neural Networks for Machine Learning.
- Kevin P. Murphy. Probabilistic Machine Learning An Introduction. MIT Press, 2022.