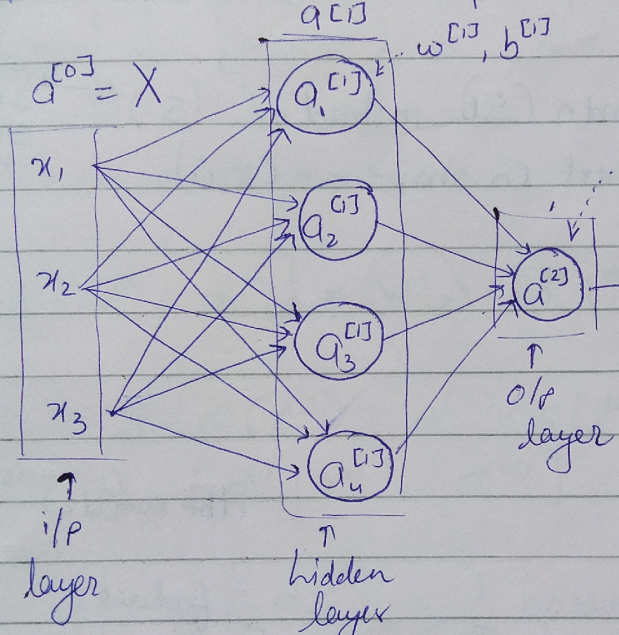
Week-3 Neural Network Representation

2 layer NN



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

$[1]$ ← layer in n/w

a_i ← node in layer

For hidden layer 1

$$\begin{bmatrix} z_1^{[1]} \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} \\ \\ \\ \end{bmatrix} x + \begin{bmatrix} b_1^{[1]} \\ \\ \\ \end{bmatrix}, \quad \begin{bmatrix} a_1^{[1]} \\ \\ \\ \end{bmatrix} = g\left(\begin{bmatrix} z_1^{[1]} \\ \\ \\ \end{bmatrix}\right)$$

$$Z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}_{4\times 1} \qquad W^{[1]} = \begin{bmatrix} x_1 & x_2 & x_3 \\ w_1^{[1]} & & \\ w_2^{[1]} & & \\ w_3^{[1]} & & \\ w_4^{[1]} & & \end{bmatrix}_{4\times 3} \qquad b^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}_{4\times 1} \qquad a_0^{[1]} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}_{4\times 1}$$

finally vectorized form is:

$$z^{[1]} = \underset{(4\times 3)}{W^{[1]}} \underset{(3\times 1)}{x} + \underset{(4\times 1)}{b^{[1]}} \quad \text{check!}$$

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

$$z^{[2]} = \underset{(4\times 1)}{W^{[2]T}} \underset{(4\times 1)}{a^{[2]}} + \underset{(4\times 1)}{b^{[2]}}$$

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

We can say that

$$x \longrightarrow a^{[2]} = \hat{y}$$

from this,

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

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

$$x^{(3)} \longrightarrow a^{[2](3)} = \hat{y}^{(3)}$$

$$\vdots \longrightarrow$$

$$x^{(m)} \longrightarrow a^{[2](m)} = \hat{y}^{(m)}$$ ← m^{th} training example

So, actually what happens is

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

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

and so on...

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

$$Z^{[1]} = \begin{bmatrix} | & & | \\ z^{[1](1)} & \cdots & z^{[1](m)} \\ | & & | \end{bmatrix}_{(n_1, m)}$$

no. of neurons in layer 1

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

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

n: no. of neurons in that layer; m: no. of training examples

(n_x, m), $n_x = 3$

testing samples →

hidden units (nodes) ↓

$$\begin{bmatrix} z_1^{[1](1)} & z_1^{[1](2)} & \cdots & z_1^{[1](m)} \\ z_2^{[1](1)} & \cdots & \cdots & z_2^{[1](m)} \\ \vdots & & & \vdots \\ z_n^{[1](1)} & \cdots & \cdots & z_n^{[1](m)} \end{bmatrix}$$

$$W^{T[1]} = \begin{bmatrix} w_1^{[1](1)} & \cdots & w_1^{[1](m)} \\ w_2^{[1](1)} & & | \\ w_3^{[1](1)} & & | \\ w_4^{[1](1)} & & w_4^{[1](m)} \end{bmatrix}$$

(consider transpose of this as $W^{[1]}$)

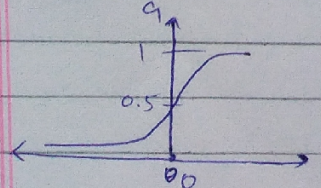
$(4, m)$ ~~$(4, n_x)$~~

$$Z^{[1](1)} = w^{[1](1)} x^{(1)} + b^{[1]}, \quad Z^{[2](1)} = w_2^{[1](2)}$$

we get $w^{[1]} x^{(1)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$ $w^{[1]} x^{(2)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$ $w^{[1]} x^{(3)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$

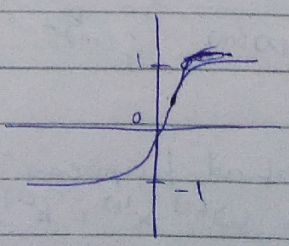
$$W^{[1]} \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix} = Z^{[1]}$$

Various Activat^n functions



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

Sigmoid f^n $\quad 0 \le \hat{y} \le 1$

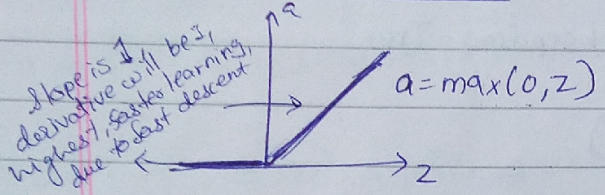


$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

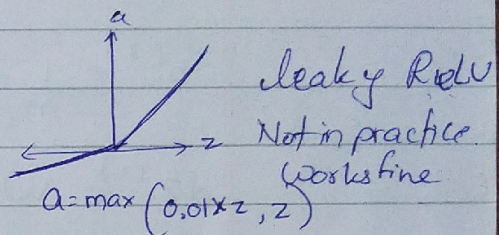
$-1 \le \hat{y} \le 1$

It may be possible that different functions are used for different layers in such a case, instead of $a^{[2]} = g(z^{[1]})$ we use $a^{[1]} = g^{[1]}(z^{[1]})$ ← indicates activation fn of layer 1

Rectified Linear Unit



Use it when you don't know what to use



It is very critical to have a non-linear activation function, or else your model would work similar to a normal machine learning problem

g(z)	derivative
Sigmoid(z)	$g(z)(1-g(z))$
tanh(z)	$1-(g(z))^2$
ReLU	$\begin{cases} 0 & \text{if } z<0 \\ 1 & \text{if } z>0 \\ \text{undef} & z=0 \end{cases}$ $z \geq 0$ ✗
leaky ReLU	$\begin{cases} 0.01 & \text{if } z<0 \\ 1 & \text{if } z \geq 0 \end{cases}$

Parameters and their sizes: $w^{[1]}$ $(n^{[1]}, n^{[0]})$, $b^{[1]}$ $(n^{[1]}, 1)$, $w^{[2]}$ $(n^{[2]}, n^{[1]})$, $b^{[2]}$ $(n^{[2]}, 1)$ $\quad n_x = n^{[0]}, n^{[1]}, n^{[2]} = 1$

Gradients All formulas

Forward Propagation

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

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

$Z^{[2]} = w^{[2]} X + b^{[2]}$

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

Back Propagation

$dZ^{[2]} = A^{[2]} - Y \qquad Y = [y^{(1)}\ y^{(2)} \dots y^{(m)}]$

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

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

$dZ^{[1]} = A^{[1]} - Y \rightarrow w^{[2]T} dZ^{[2]} * g^{[1]'}(Z^{[1]})$ (* = elementwise product)

$dw^{[1]} = \frac{1}{m} dZ^{[1]} A^{[0]T}$

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

Vectorized Backpropagation

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

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

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

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

$$dw^{[1]} = \frac{1}{m} dz^{[1]} x^{T}$$

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