A B C D E F G H I J K L M N O P Q R S T U V W X Y Z AA AB AC AD AE AF AG AH
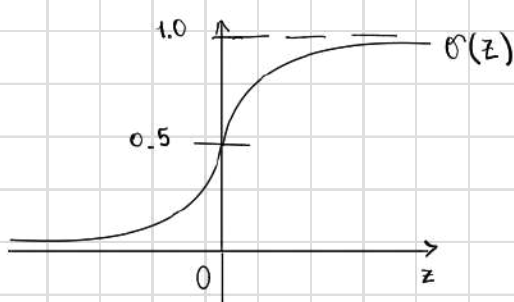
Logistic Regression

- Input: $X \in \mathbb{R}^{n_x}$ ("n" dimensional vector).   • Parameters: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$.
- Output: $\hat{y} = P(y = 1 | x)$, $0 \le \hat{y} \le 1$.

$\hat{y} = w^T x + b$ doesn't make sense for logistic regression because $\hat{y}$ can't be a large number.
So we use the sigmoid function in this case: $\hat{y} = \sigma(w^T x + b) \longrightarrow w^T x + b = z \longrightarrow \hat{y} = \sigma(z)$

$\sigma(z) = \dfrac{1}{1 + e^{-z}}$; $\lim\limits_{z \to \infty} \sigma(z) = 1$; $\lim\limits_{z \to -\infty} \sigma(z) = 0$.

Given $\{ (x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)}) \}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

Loss Function For Logistic Regression (single error)

We don't use $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ in L.R. because the optimization will not converge

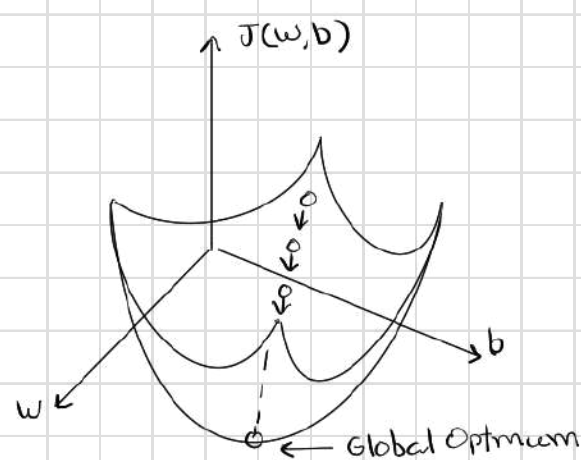$L(\hat{y}, y) = -(y \log(\hat{y}) + (1-y) \cdot \log(1 - \hat{y}))$

- If $y = 0$: $L(\hat{y}, y) = -\log \hat{y}$ ← want $\log(\hat{y})$ large, so want $\hat{y}$ large

- If $y = 1$: $L(\hat{y}, y) = -\log(1 - \hat{y})$ ← want $\log(1-\hat{y})$ large, so want $\hat{y}$ small
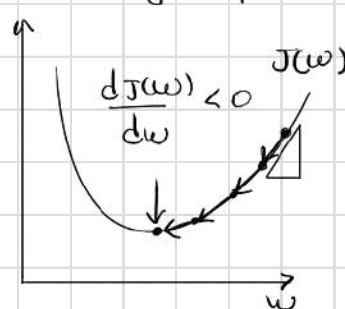
Cost Function (average error)

$$J(w,b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \cdot \log \hat{y}^{(i)} + (1 - y^{(i)}) \cdot \log(1 - \hat{y}^{(i)}) \right]$$

Gradient Descent

Want to find w, b to minimize $J(w,b)$.

Considering only w

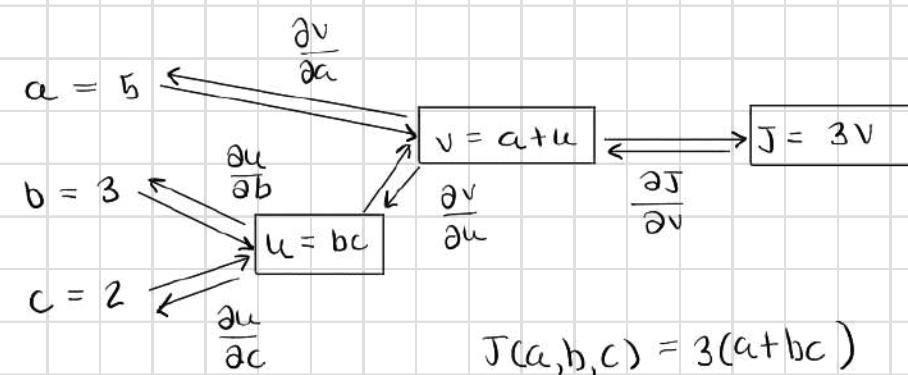Considering multiple parameters, we use "partial derivative".

$$w := w - \alpha \frac{\partial J(w,b)}{\partial w}$$
$$b := b - \alpha \frac{\partial J(w,b)}{\partial b}$$

$\frac{dJ(w)}{dw} < 0$

Repeat until converged:
$\{ w := w - \alpha \frac{dJ(w)}{w} \}$

Computation Graph

To find partial derivatives we're going to consider the propagation throght the computational graph.

$a = 5$
$b = 3$
$u = bc$
$c = 2$
$v = a + u$
$J = 3v$

$J(a,b,c) = 3(a + bc)$

$\frac{\partial J}{\partial u} = \frac{\partial J}{\partial v} \cdot \frac{\partial v}{\partial u} = 3 \times 1 = 3$ $\frac{\partial J}{\partial v} = 3$

$\frac{\partial J}{\partial a} = \frac{\partial J}{\partial v} \cdot \frac{\partial v}{\partial a} = 3 \times 1 = 3$

$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial v} \cdot \frac{\partial v}{\partial u} \cdot \frac{\partial u}{\partial b} = 3 \times c = 6$

$\frac{\partial J}{\partial c} = \frac{\partial J}{\partial v} \cdot \frac{\partial v}{\partial u} \cdot \frac{\partial u}{\partial c} = 3 \times b = 9$

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z AA AB AC AD AE AF AG AH

Gradient Descent on "m" Example

$$z = w^T x + b; \quad \hat{y} = a = \sigma(z); \quad L(a,y) = -(y \log(a) + (1-y)\log(1-a))$$

$$\frac{\partial a}{\partial z} = a(1-a) \qquad \frac{\partial L(a,y)}{\partial a} = \frac{-y}{a} + \frac{1-y}{1-a}$$

$$\frac{\partial L(a,y)}{\partial w_1} = x_1 \cdot \frac{\partial L(a,y)}{\partial z}$$

x_1, w_1, x_2, w_2, b

$$\boxed{z = w_1 x_1 + w_2 x_2 + b} \longleftarrow \boxed{a = \sigma(z)} \longleftarrow \boxed{L(a,y)}$$

$$\frac{\partial L(a,y)}{\partial w_2} = x_2 \cdot \frac{\partial L(a,y)}{\partial z}$$

$$\frac{\partial L(a,y)}{\partial z} = \frac{\partial L(a,y)}{\partial a} \cdot \frac{\partial a}{\partial z} = \left(\frac{-y}{a} + \frac{1-y}{1-a}\right) a(1-a) = a - y$$

$$\frac{\partial L(a,y)}{\partial b} = \frac{\partial L(a,y)}{\partial z}$$

$$w_1 := w_1 - \alpha \frac{\partial L(a,y)}{\partial w_1} \qquad w_2 := w_2 - \alpha \frac{\partial L(a,y)}{\partial w_2} \qquad b := b - \alpha \frac{\partial L(a,y)}{\partial b}$$

Algorithm :

$$J(w,b) = \frac{1}{m} \sum_{i=1}^{m} L(a^{(i)}, y^{(i)})$$

```
J=0; dw₁=0; dw₂=0; db=0
For i=1 to M:
```
$$z^{(i)} = w^T x^{(i)} + b$$
$$a^{(i)} = \sigma(z^{(i)})$$
$$J += -[y^{(i)} \log(a^{(i)}) + (1-a^{(i)})\log(1-a^{(i)})]$$
$$dz = a^{(i)} - y^{(i)}$$
$$dw_1 += x_1^{(i)} \cdot dz^{(i)}$$
$$dw_2 += x_2^{(i)} \cdot dz^{(i)}$$
$$db += dz^{(i)}$$
$$J = J/M; \quad dw_1 = dw_1/M; \quad dw_2 = dw_2/M; \quad db = db/M;$$

Vectorized Implementation
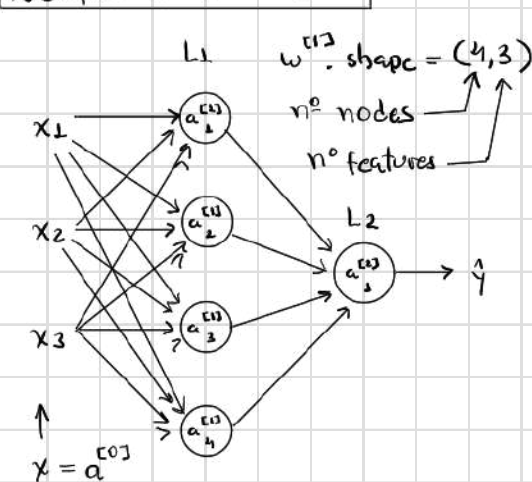
$$Z = w^T X + b$$
$$A = \sigma(Z)$$
$$dZ = A - Y$$
$$dw = \frac{1}{m} X \, dZ^T$$
$$db = \frac{1}{m} dZ$$
$$w := w - \alpha \, dw$$
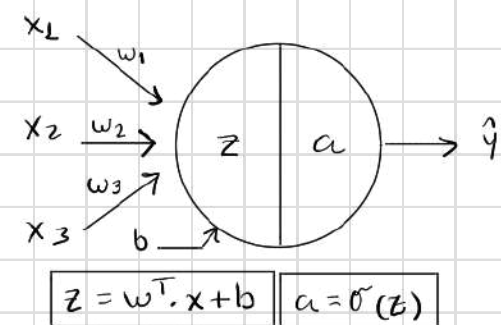$$b := b - \alpha \, db$$

Neural Network

L_1

$w^{[1]}$: shape = (4,3)

n° nodes
n° features

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

L_2

$\hat{y}$

$x = a^{[0]}$

$$z_1^{[2]} = w_1^{[2]T} X + b_1^{[2]}, \quad a_1^{[2]} = \sigma(z_1^{[2]})$$

x_1, w_1, x_2, w_2, x_3, w_3, b

$$\boxed{z} \, \boxed{a} \rightarrow \hat{y}$$

$$\boxed{z = w^T \cdot x + b} \quad \boxed{a = \sigma(z)}$$

vectorizing across multiple examples

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

$$z^{[1]} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} \cdot x + b_1^{[1]} \\ w_2^{[1]T} \cdot x + b_2^{[1]} \\ w_3^{[1]T} \cdot x + b_3^{[1]} \\ w_4^{[1]T} \cdot x + b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

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

$$z^{[2]} = \begin{bmatrix} w_1^{[2]T} \end{bmatrix} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} + \begin{bmatrix} b_1^{[2]} \end{bmatrix} = \begin{bmatrix} w_1^{[2]} \cdot a^{[1]} + b_1^{[2]} \end{bmatrix} = \begin{bmatrix} z_1^{[2]} \end{bmatrix}$$
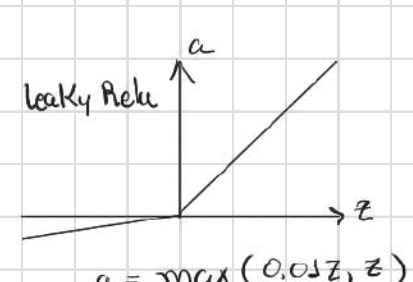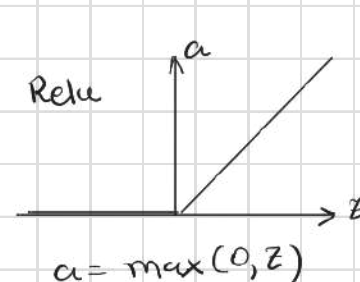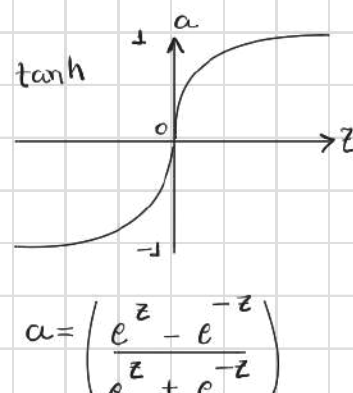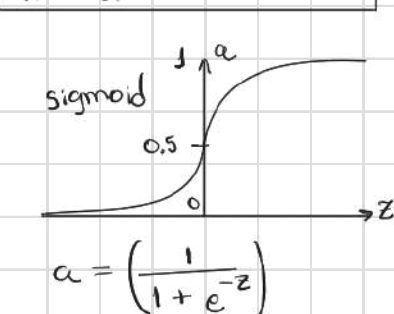
Activation Functions

tanh

sigmoid

0.5

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

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

Relu

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

leaky Relu

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

Gradient Descent For Neural Networks

In this case, $g(z)$ represents a generic activation function.

Forward propagation

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

Back propagation

$$dZ^{[2]} = A^{[2]} - Y$$
$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$$
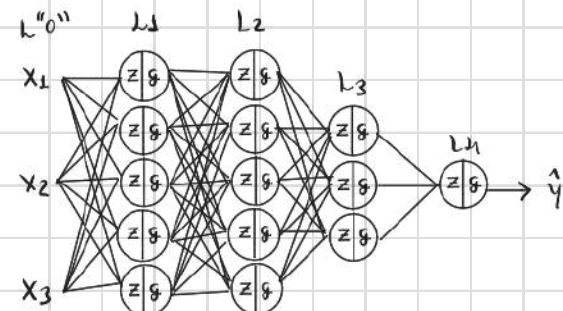$$db^{[2]} = \frac{1}{m} \cdot np.sum(dZ^{[2]}, axis = 1, keepdims = True)$$
$$dZ^{[1]} = W^{[2]T} dZ^{[2]} * g^{[1]}(Z^{[1]})$$
$$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^{T}$$
$$db^{[1]} = \frac{1}{m} \cdot np.sum(dZ^{[1]}, axis = 1, keepdims = True)$$

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

$\longrightarrow$ element wise product

Deep Learning Notation

L "0" L_1 L_2

x_1 L_3 L_4 $\hat{y}$

x_2

x_3

$x = a^{[0]}$

L = number of layers
$n^{[\ell]}$ = number of units in layer "ℓ"
$a^{[\ell]}$ = activations in layer "ℓ"
$a^{[\ell]} = g^{[\ell]}(z^{[\ell]})$,
$W^{[\ell]}$
$b^{[\ell]}$ = weights for $z^{[\ell]}$

Forward propagation

$$z^{[\ell]} = W^{[\ell]} a^{[\ell-1]} + b^{[\ell]}$$
$$a^{[\ell]} = g^{[\ell]}(z^{[\ell]})$$

Vectorized vectorization

$$Z^{[1]} = W^{[1]} A^{[0]} + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$\vdots$$
$$\hat{Y} = g(Z^{[4]}) = A^{[4]}$$

Parameters Dimensions

$Z^{[\ell]} \longrightarrow (n^{[\ell]}, 1)$
$W^{[\ell]} \longrightarrow (n^{[\ell]}, n^{[\ell-1]})$
$A^{[\ell]} \longrightarrow (n^{[\ell]}, 1)$
$b^{[\ell]} \longrightarrow (n^{[\ell]}, 1)$
$dW^{[\ell]} \longrightarrow (n^{[\ell]}, n^{[\ell-1]})$
$db^{[\ell]} \longrightarrow (n^{[\ell]}, 1)$

Notes that:

$z^{[\ell]}, a^{[\ell]} \longrightarrow (n^{[\ell]}, 1)$
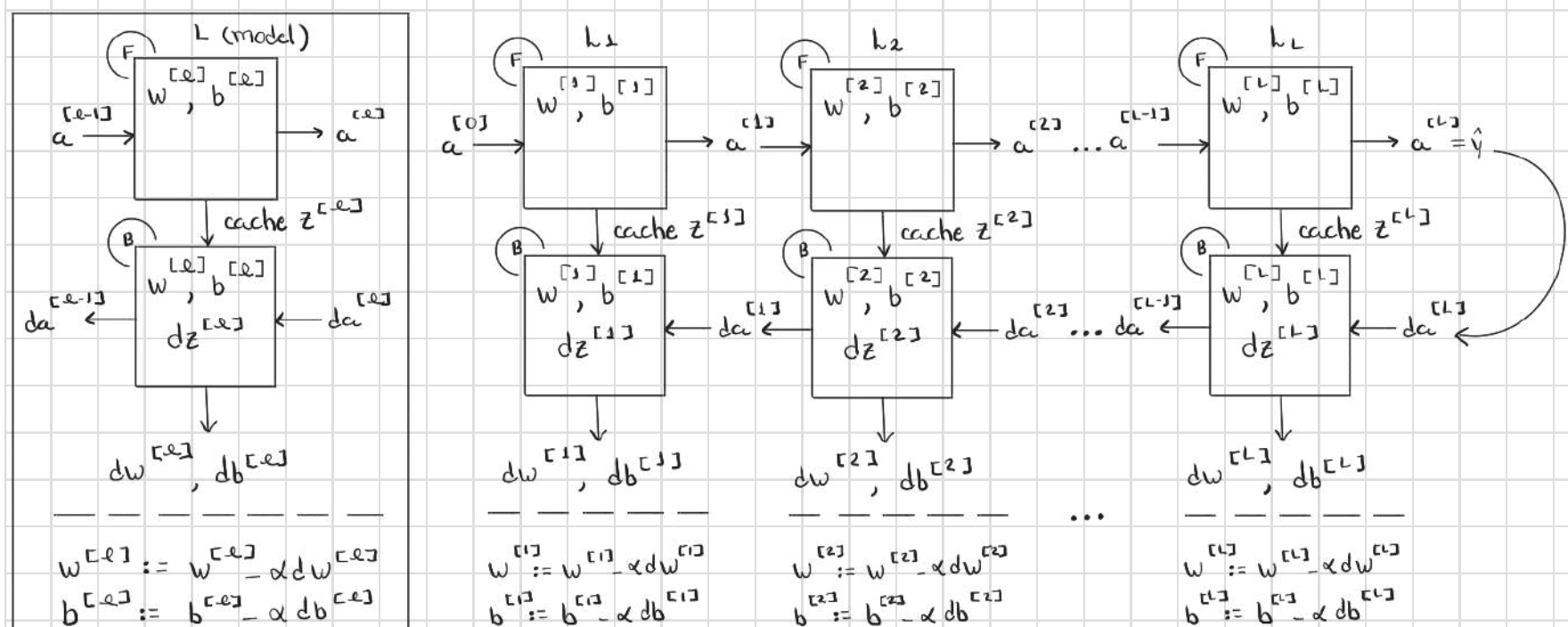$Z^{[\ell]}, A^{[\ell]} \longrightarrow (n^{[\ell]}, m)$
for $\ell = 0$, $A^{[0]} = X \longrightarrow (n^{[0]}, m)$
$dZ^{[\ell]}, dA^{[\ell]} \longrightarrow (n^{[\ell]}, m)$

$$Z^{[1]} = W^{[1]} \cdot X + b^{[1]}$$
$\downarrow \qquad \downarrow \qquad \downarrow \qquad (n^{[1]}, 1)$ python broadcast
$(n^{[1]}, m) \quad (n^{[1]}, n^{[0]}) \quad (n^{[0]}, m) \quad (n^{[1]}, m)$

Forward and Backward function blocks

Forward: Input $a^{[\ell-1]}$, output $a^{[\ell]}$, cache $z^{[\ell]}$.   Backward: Input $da^{[\ell]}$, output $da^{[\ell-1]}$, $dw^{[\ell]}$, $db^{[\ell]}$.

L (model)

F $\quad w^{[\ell]}, b^{[\ell]}$
$a^{[\ell-1]} \rightarrow \qquad \rightarrow a^{[\ell]}$
cache $z^{[\ell]}$
B $\quad w^{[\ell]}, b^{[\ell]}$
$da^{[\ell-1]} \leftarrow \quad dz^{[\ell]} \quad \leftarrow da^{[\ell]}$
$dw^{[\ell]}, db^{[\ell]}$
$$w^{[\ell]} := w^{[\ell]} - \alpha dw^{[\ell]}$$
$$b^{[\ell]} := b^{[\ell]} - \alpha db^{[\ell]}$$

L_1
F $\quad w^{[1]}, b^{[1]}$
$a^{[0]} \rightarrow \qquad \rightarrow a^{[1]}$
cache $z^{[1]}$
B $\quad w^{[1]}, b^{[1]}$
$da^{[0]} \leftarrow \quad dz^{[1]} \quad \leftarrow da^{[1]}$
$dw^{[1]}, db^{[1]}$
$$w^{[1]} := w^{[1]} - \alpha dw^{[1]}$$
$$b^{[1]} := b^{[1]} - \alpha db^{[1]}$$

L_2
F $\quad w^{[2]}, b^{[2]}$
$\rightarrow a^{[2]}$
cache $z^{[2]}$
B $\quad w^{[2]}, b^{[2]}$
$da^{[1]} \leftarrow \quad dz^{[2]} \quad \leftarrow da^{[2]}$
$dw^{[2]}, db^{[2]}$
$$w^{[2]} := w^{[2]} - \alpha dw^{[2]}$$
$$b^{[2]} := b^{[2]} - \alpha db^{[2]}$$

... $a^{[L-1]} \rightarrow$

L_L
F $\quad w^{[L]}, b^{[L]}$
$\rightarrow a^{[L]} = \hat{y}$
cache $z^{[L]}$
B $\quad w^{[L]}, b^{[L]}$
... $da^{[L-1]} \leftarrow \quad dz^{[L]} \quad \leftarrow da^{[L]}$
$dw^{[L]}, db^{[L]}$
$$w^{[L]} := w^{[L]} - \alpha dw^{[L]}$$
$$b^{[L]} := b^{[L]} - \alpha db^{[L]}$$

...

ANNs and the human neuron

x_1 $w_1^{[1]}$
x_2 $w_2^{[1]}$ $z | \sigma$ $a^{[1]} = \hat{y}$
x_3 $w_3^{[1]}$
$b^{[1]}$