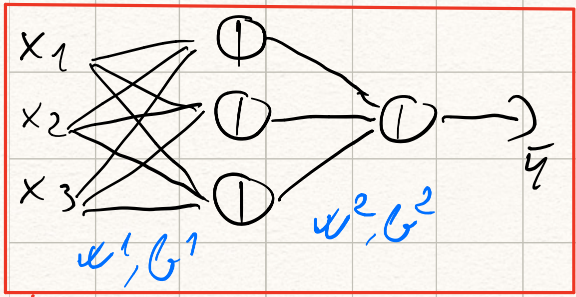


$\mathcal{L}(\bar{y}, y)$: loss

- Training a neural network is an optimization problem. The goal is to minimize the loss by update the values of the weight matrices and biases.

w^2, b^2 $\quad$ dw^2, db^2

$x, w^1, b^1 \rightarrow \boxed{z^1 = w^1 x + b^1} \rightarrow \boxed{a^1 = f(z^1)} \rightarrow \boxed{z^2 = w^2 a^1 + b^2} \rightarrow \boxed{a^2 = f(z^2)} \rightarrow \boxed{\mathcal{L}(a^2, y)}$

dw^1, db^1 $\quad$ dz^1 $\quad$ da^1 $\quad$ dz^2 $\quad$ da^2

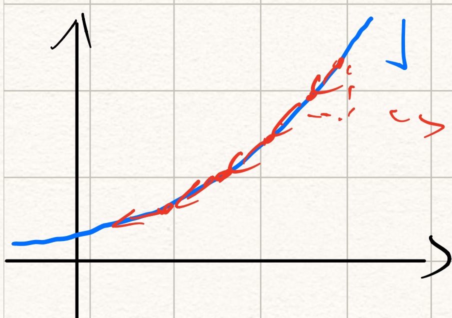
$\longrightarrow$ Forward propagation → compute a prediction and calculate the loss

$\longleftarrow$ Back propagation

Back propagate the error through the gradients to update the weights and biases.

Cost function:

$$J(w^1, b^1, w^2, b^2) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\bar{y}, y)$$



→ derivatives

$\Rightarrow \min(J(w, b))$

gradient descent

Repeat until convergence {

compute forward predictions $(\hat{y}^{(i)}, i = 1 \dots n)$

$dw^1 = \frac{\partial J}{\partial w^1}$, $db^1 = \frac{\partial J}{\partial b^1}$ / w^2, b^2 is the same

$w^1 := w^1 - h \cdot dw^1$

$b^1 := b^1 - h \, db^1$ / w^2, b^2 is the same

}

→ learning rate

Compute these derivative terms with

Backpropagation

$dz^2 = a^2 - Y$, $Y = [y^1, y^2 \dots y^n]$ labeled data

$dw^2 = \frac{1}{m} dz^2 \cdot a^{1T}$

$db^2 = \frac{1}{m} \text{sum}(dz^2, \text{axis} = 1)$ $(n, 1)$ array

$dz^1 = w^{2T} \cdot dz^2 * f^{1\prime}(z^1)$ derivative of the activation function

→ element wise

$dw^1 = \frac{1}{m} dz^1 \cdot x^T$ → a^{0T}

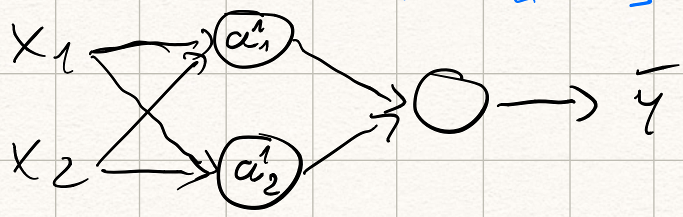
$db^1 = \frac{1}{m} \text{sum}(dz^1, \text{axis} = 1)$

It is based on the chain rule:

$$F(x) = f(g(x)) \Rightarrow F'(x) = f'(g(x)) \cdot g'(x)$$

Random initialization

$w^2 = [0 \ 0]$



$w^1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \Rightarrow a_1^1 = a_2^1 \Rightarrow dz_1^1 = dz_2^1$

Both hidden units compute the same function.

$\Rightarrow$ Random weight matrices initialization