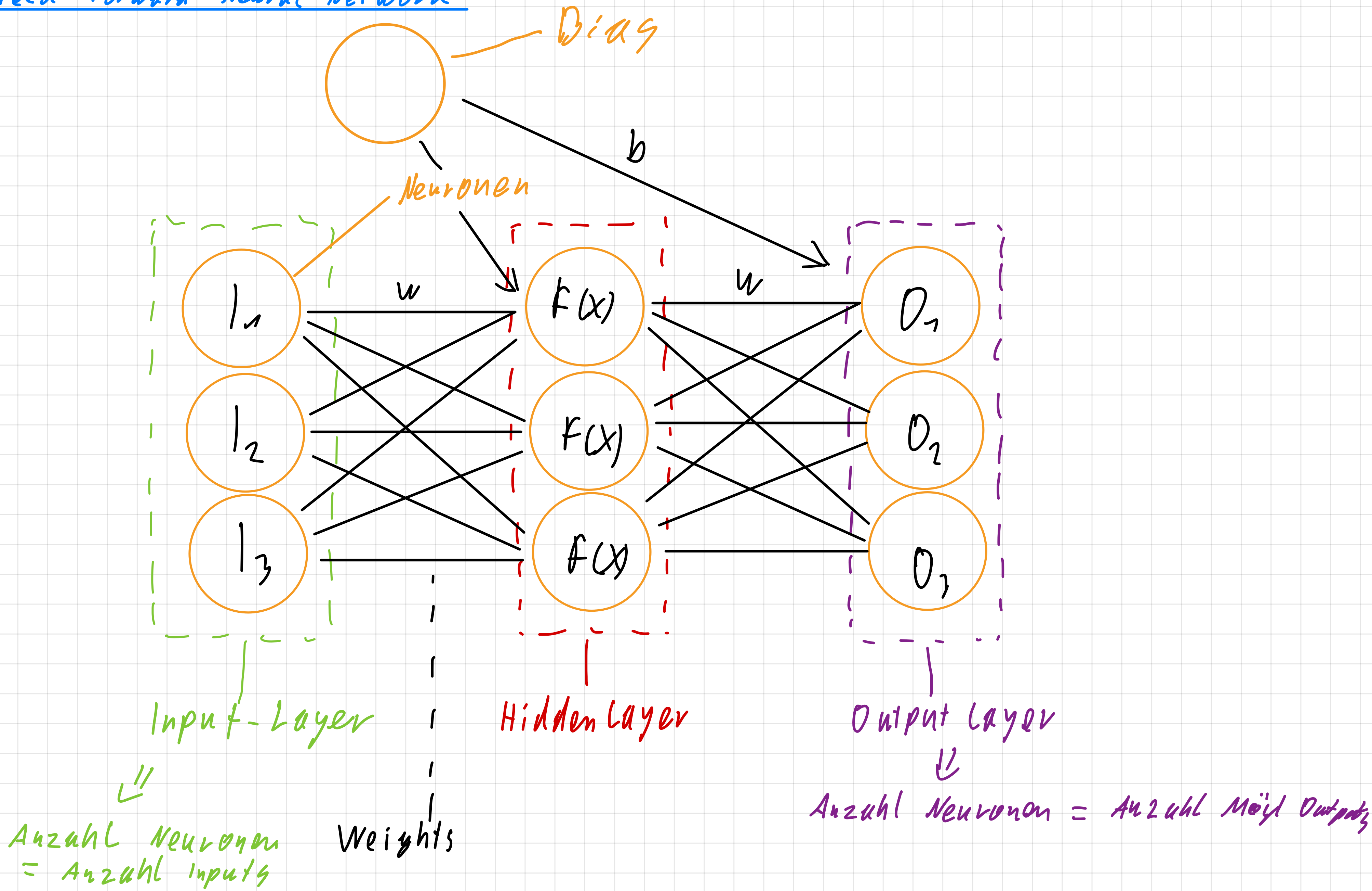
Feed - Forward - Neural Network



Neuron

$0 < x_n < 1$

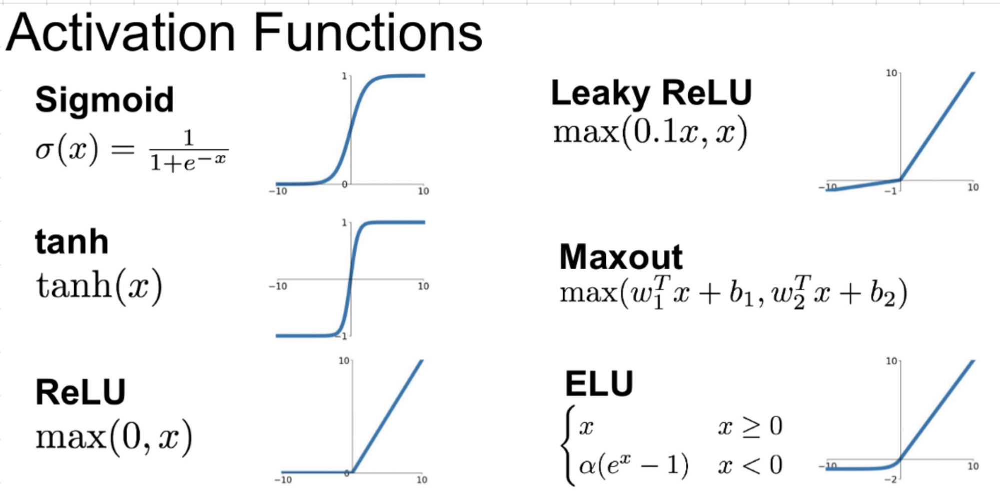
$\vee$

$f(x)$

Beinhaltet meist eine Zahl zwischen 0 und 1. Kann auch als mathematische Funktion f(x) ergänzt werden.

$\Updownarrow$

„Aktivierungsfunktion"



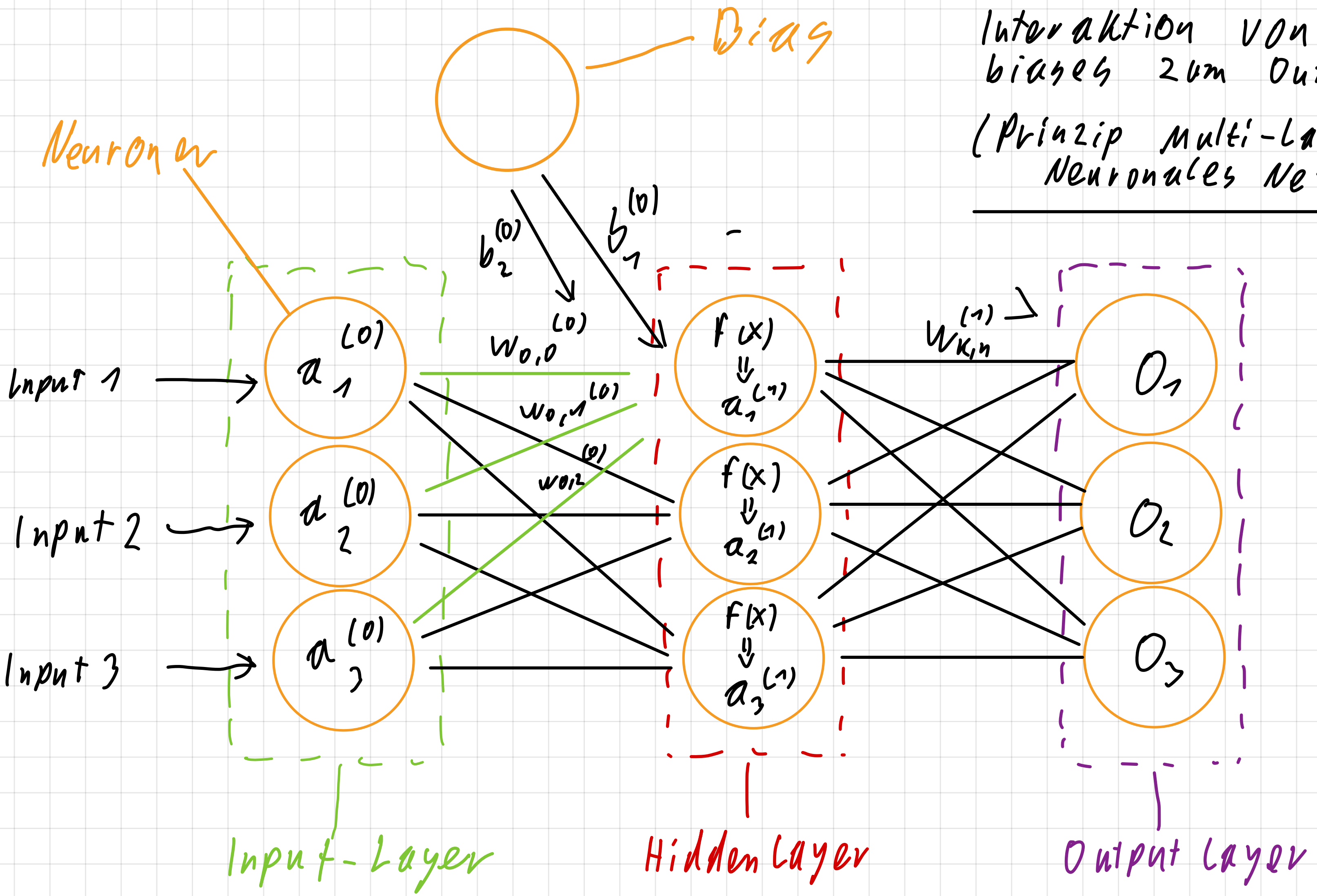
Wie wird das „Learning" gemacht?

- Weights und Biases der Aktivierungsfunktion können angepasst werden
- Das „Lernen" ist das Finden der passenden weights und biases

Mathematik

Interaktion von Neuronen, weights, biases zum Output Signal!

(Prinzip Multi-Layer - Perceptron) Neuronales Netzwerk



$$\Rightarrow \quad \underset{\text{Inputs}}{I} = \begin{pmatrix} I_1^{(0)} \\ I_2^{(0)} \\ I_3^{(0)} \\ \vdots \\ I_n^{(0)} \end{pmatrix} ; \quad W = \begin{pmatrix} \boxed{w_{0,0} \quad w_{0,1} \quad \dots \quad w_{0,n}} \\ w_{1,0} \quad w_{1,1} \quad \dots \quad w_{1,n} \\ \vdots \qquad\qquad \ddots \qquad \vdots \\ w_{k,0} \quad w_{k,1} \quad \dots \quad w_{k,n} \end{pmatrix} ; \quad B = \begin{pmatrix} b_1^{(n)} \\ b_2^{(n)} \\ b_3^{(n)} \\ \vdots \\ b_k^{(n)} \end{pmatrix}$$

$$\Rightarrow \boxed{\begin{aligned} a_n^{(0)} &= f\left(W^{(0)} \cdot 1 + B\right) \\ a_n^{(k)} &= \underbrace{f}_{\text{Aktivierungsfunktion (siehe S. 2)}}\left(W^{(k)} \cdot a_n^{(k-1)} + B\right) \end{aligned}}$$

Erste „berechnung" mit ersten Weights

Weitere Berechnung, $K > 0 \in \mathbb{N}$

Die Aktivierungsfunktion erzeugt die Non-Lineare Klassifizierung.

Siehe Coding bsp.

Kosten Funktion

Die Kostenfunktion ist ein Indikator dafür, wie gut / schlecht das Neuronale Netzwerk den Input erkennt. Je niedriger die Kosten, desto „besser" ist es.

Sei O = Outputs, G = Ziel Outputs, C = Kosten, dann gilt:

$$\boxed{C = \sum_{i=0}^{|O|} d(O_i, G_i)^2}$$

$(\text{Output} - \text{Zieloutput})^2$

Da wir daran interessiert sind wie die Kosten über mehrere Daten sind, gilt:

$$\boxed{\varnothing C = \frac{\sum_{i=0}^{|\text{Daten}|} C_i}{|\text{Daten}|}}$$

Summe aller Kosten : Anzahl der Daten

Siehe Coding bsp.

Gradiant Descent

Da, wir nun eine Kostenfunktion haben, interressieren wir uns für das Minimum der Kostenfunktion in abhängigkeit von weights und biases

$\Rightarrow \quad C(w,b) \rightarrow$ Kosten

$\Rightarrow \quad C'(w,b) = \left(\frac{\partial C}{\partial w}, \frac{\partial C}{\partial b} \right)$ „Partielle Ableitung" „∇C" ← Gradient

Theoretisch würden wir bekannterweise

$C'(w,b) = (0,0)$ setzen um das Minima zu bekommen.

In der Praxis wird das nicht gemacht, da dies zu „schwer", unpraktisch und manchmal nicht durchführbar ist besonders für sehr große Dimensionen.

Deshalb kommt das Prinzip vom Gradiant Descent „Gradienten Abstieg".

Eine andere Betrachtung des Gradianten, ist die lokale Steigung der Funktion.

$\nabla C(w,b)$ = lokale Steigung

Somit können wir $-\nabla C$-schrittige Iterationen durchführen um ein lokales Minimum zu erhalten, welches abhängig von den weights und biases ist.

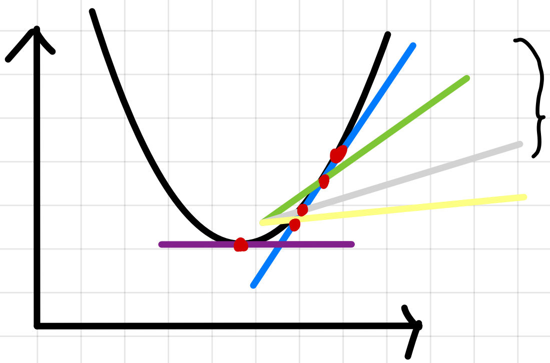
$$\boxed{C'(w_{k,n}, b_n) = \left(\frac{\partial C}{\partial w_{k,n}} \quad \frac{\partial C}{\partial b_n}\right)}$$

$f(x) = x^3$

$f'(x) = 3x^2$

w_i

Skizze:



Tangentensteigung (m) => (−m) Schritte gehen bis $m = 0 = -m$

Gradiant weights der Kosten:

$$\boxed{\frac{\partial C}{\partial w_{k,n}} = \lim_{h \to 0} \frac{C(w_{k,n} + h, b_n) - C(w_{k,n}, b_n)}{h}}$$

Gradiant biases der Kosten:

$$\boxed{\frac{\partial C}{\partial b_n} = \lim_{h \to 0} \frac{C(w_{k,n}, b_n + h) - C(w_{k,n}, b_n)}{h}}$$

Nun müssen alle weights und biases „angepasst" werden.

Dabei gilt:

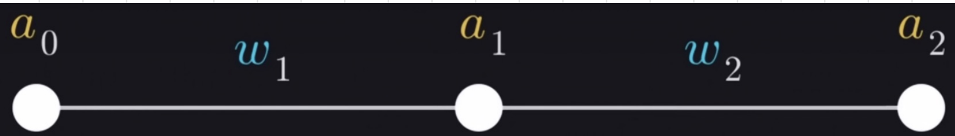
$$\boxed{\begin{aligned} w_{k,n} &= w_{k,n} - \frac{\partial C}{\partial w_{k,n}} \cdot L \\ b_n &= b_n - \frac{\partial C}{\partial b_n} \cdot L \end{aligned}}$$

, L = Learnrate $\in (0,1)$

$\Downarrow$

Learnrate ist sozusagen die Lerngeschwindigkeit bzw. Schrittweite.

Backpropagation



(Für Output-Layer)

$$z_1 = a_0 w_1 + b_1$$

$$a_1 = A(z_1)$$

$$z_2 = a_1 w_2 + b_2$$

$$a_2 = A(z_2)$$

$$c = C(a_2, y)$$

$$\frac{\partial c}{\partial w_2} = \frac{\partial z_2}{\partial w_2} \times \frac{\partial a_2}{\partial z_2} \times \frac{\partial c}{\partial a_2}$$

$$\frac{\partial c}{\partial b_2} = \frac{\partial z_2}{\partial w_2} \times \frac{\partial a_2}{\partial z_2} \times \frac{\partial c}{\partial a_2}$$

$$\boxed{\frac{\partial C}{\partial a_2} = 2 \cdot d(O_i, G_i)}$$ Ableitung Kostenfunktion.

$$\boxed{\frac{\partial a_2}{\partial z_2} = A'(z_2) = \begin{cases} 0, & z_2 < 0 \\ 1, & z_2 > 0 \end{cases} \quad \text{(für ReLu)}}$$ Ableitung Aktivierungsfunktion.

$$\boxed{\frac{\partial z_2}{\partial w_2} = a_1} \quad \boxed{\frac{\partial z_2}{\partial b_1} = 1}$$ Ableitung „Signal" bevor Aktivierungsfunktion.

$$\Rightarrow \underline{\underline{\frac{\partial C}{\partial W_2} = a_1 \cdot \begin{pmatrix} 0, & z_2 < 0 \\ 1, & z_2 > 0 \end{pmatrix} \cdot 2 \cdot (O_i - G_i)}}$$

$$\Rightarrow \frac{\partial C}{\partial b_1} = 1 \cdot \begin{pmatrix} 0, & z_2 < 0 \\ 1, & z_2 > 0 \end{pmatrix} \cdot 2 \cdot (O_i - G_i)$$

(Für Hiddenlayer)

ähnlich wie oben.

wird fortgeführt beim Ankommen zum Backpropagation.