

Künstliche Intelligenz selbst gemacht

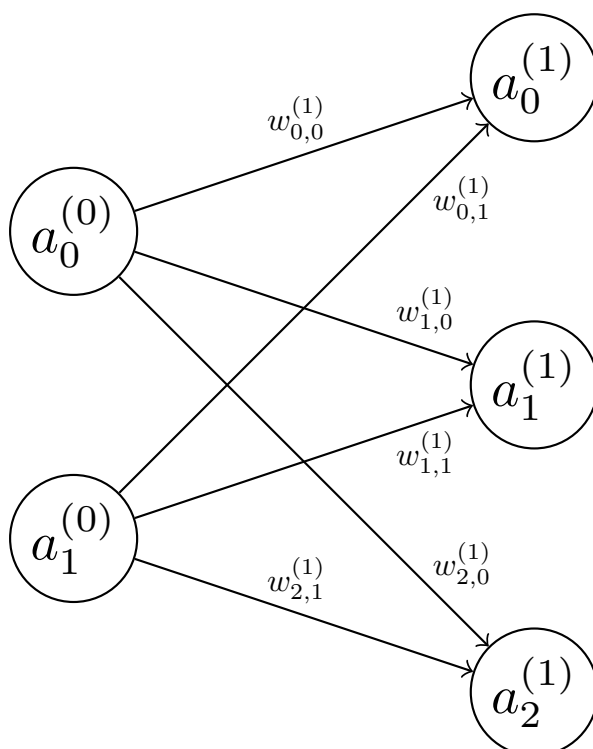
Ein neuronales Netz mit Python entwickeln

Notationen im neuronalen Netz

x_i	i -ter Eintrag im Inputvektor des Trainingsdatensatzes
$a_j^{(l)}$	Aktivierung des j -ten Neurons im Layer l
$b^{(l)}$	Bias im Layer l zum Links-Rechts-Shift der Aktivierungsfunktion
$w_{jk}^{(l)}$	Gewicht vom k -ten Neuron im Layer $l - 1$ zum j -ten Neuron im Layer l
$z_j^{(l)}$	Gewichtete Summe aller eingehenden Neuronenaktivierungen im j -ten Neuron im Layer l
$\sigma^{(l)}$	Aktivierungsfunktion im Layer l
y_i	i -ter Eintrag im erwarteten Outputvektor des Trainingsdatensatzes

Aktivierungsfunktionen

Name	$f(x)$	$f'(x)$ (1. Ableitung)
Rectified Linear Unit	$ReLU(x) = \begin{cases} x & \text{wenn } x > 0 \\ 0 & \text{sonst} \end{cases}$	$ReLU'(x) = \begin{cases} 1 & \text{wenn } x > 0 \\ 0 & \text{sonst} \end{cases}$
Sigmoid	$\sigma(x) = \frac{1}{1 + e^{-x}}$	$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$



$$A^{(1)} = \sigma^{(1)} \left(Z^{(1)} \right)$$

$$= \sigma^{(1)} \left(W^{(1)} A^{(0)} \right)$$

$$= \sigma^{(1)} \left[\begin{pmatrix} w_{0,0}^{(1)} & w_{0,1}^{(1)} \\ w_{1,0}^{(1)} & w_{1,1}^{(1)} \\ w_{2,0}^{(1)} & w_{2,1}^{(1)} \end{pmatrix} \begin{pmatrix} a_0^{(0)} \\ a_1^{(0)} \end{pmatrix} \right]$$

Kostenfunktion bei Regression

Squared Error Loss: $C = (A^{(l)} - Y)^2$

Ableitung:

$$\frac{\partial C}{\partial w_{jk}^{(l)}} = \frac{\partial C}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{jk}^{(l)}}$$

1

$$\frac{\partial z_j^{(l)}}{\partial w_{jk}^{(l)}} = \frac{\partial w_{jk}^{(l)} a_k^{(l-1)}}{\partial w_{jk}^{(l)}} = a_k^{(l-1)}$$

2

$$\frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} = \sigma^{(l)'}(z_j^{(l)})$$

3

$$\frac{\partial C}{\partial a_j^{(l)}} = 2(a_j^{(l)} - y_j)$$

4

$$\frac{\partial C}{\partial a_j^{(l)}} = \sum_{i=0}^{n^{(l+1)}-1} \left(\frac{\partial C}{\partial a_i^{(l+1)}} \frac{\partial a_i^{(l+1)}}{\partial z_i^{(l+1)}} w_{ij}^{(l+1)} \right)$$

5

$$\frac{\partial C}{\partial w_{jk}^{(l)}} = \frac{\partial C}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \cdot a_k^{(l-1)} = \delta_j^{(l)} \cdot a_k^{(l-1)}$$

6

$$\delta_j^{(l)} = \begin{cases} 2(a_j^{(l)} - y_j) \cdot \sigma^{(l)'}(z_j^{(l)}) & \text{für den Output-Layer} \\ \left(\sum_{i=0}^{n^{(l+1)}-1} \delta_i^{(l+1)} w_{ij}^{(l+1)} \right) \cdot \sigma^{(l)'}(z_j^{(l)}) & \text{für innere Layer} \end{cases}$$

7

Anpassung der Gewichte

$$\Delta w_{jk}^{(l)} = -\eta \cdot \frac{\partial C}{\partial w_{jk}^{(l)}} = -\eta \cdot \delta_j^{(l)} \cdot a_k^{(l-1)}$$

8

Softmax und Cross-Entropy Loss

Softmax:

$$\text{softmax} \left(z_i^{(l)} \right) = \frac{e^{z_i^{(l)}}}{\sum_{j=0}^{n^{(l)}-1} e^{z_j^{(l)}}} \quad \text{softmax}^* \left(z_i^{(l)} \right) = \frac{e^{z_i^{(l)} - \max(Z^{(l)})}}{\sum_{j=0}^{n^{(l)}-1} e^{z_j^{(l)} - \max(Z^{(l)})}}$$

Cross-Entropy Loss:

$$C = - \sum_{i=0}^{n^{(l)}-1} y_i \cdot \ln \left(a_i^{(l)} \right)$$

Ableitung:

$$\frac{\partial s_i^{(l)}}{\partial z_j^{(l)}} = s_i^{(l)} \cdot \frac{\partial}{\partial z_j^{(l)}} \ln \left(s_i^{(l)} \right)$$

I

$$\frac{\partial z_i^{(l)}}{\partial z_j^{(l)}} = 1_{\{i=j\}} = \begin{cases} 1 & \text{falls } i = j \\ 0 & \text{sonst} \end{cases}$$

II

$$\frac{\partial s_i^{(l)}}{\partial z_j^{(l)}} = s_i^{(l)} \cdot \left(1_{\{i=j\}} - s_j^{(l)} \right)$$

III

$$\frac{\partial C}{\partial z_j^{(l)}} = s_j^{(l)} - y_j$$

IV

$$\frac{\partial C}{\partial w_{jk}^{(l)}} = \left(s_j^{(l)} - y_j \right) \cdot a_k^{(l-1)}$$

V