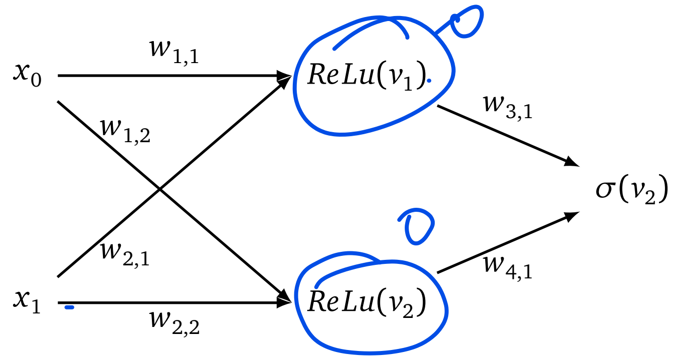
4. Übungsblatt

Aufgabe 4.1 (XOR Theorie) (1 + 1 + 2 Punkte)

In dieser Aufgabe betrachten wir ein neuronales Netz zur Berechnung von XOR, dazu betrachten wir folgendes Modell:



$$W_1 = \begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{pmatrix} = \begin{pmatrix} -1 & 0,5 \\ 1 & -0,5 \end{pmatrix} \qquad W_2 = \begin{pmatrix} w_{3,1} \\ w_{4,1} \end{pmatrix} = \begin{pmatrix} 0,25 \\ -0,25 \end{pmatrix}$$

Zur Erinnerung, die Wahrheitstafel für XOR:

k	x_0	x_1	$XOR(x_0, x_1)$
1	1	1	0
2	1	0	1
3	0	1	1
4	0	0	0

$x^T \cdot W_1 = \overset{ReLu}{(v_1, v_2)} \cdot W_2$

$\frac{1}{1+e^0}$

$\frac{1}{1+e^{\frac{1}{4}}} \approx 0{,}562$

$\frac{1}{1+e^{\frac{1}{8}}} \approx 0{,}469$

Tabelle 4.1: Wahrheitstafel für die logische XOR-Operation

(a) Geben Sie die Funktion $h(x, W_1, W_2)$ mit $x = (x_0 \;\; x_1)^T$ an, die durch dieses neuronale Netz beschreiben wird. Dabei ist $\sigma(v) := \frac{1}{1+e^{-v}}$ und $ReLu(v) := max\{0, v\}$.

(b) Berechnen Sie für jedes Eingabepaar $x^{(k)}$ (siehe Tabelle 4.1) die Modellausgabe.

(c) Als Fehlerfunktion verwenden wir den quadratischen Fehler:

$$QF(y, h(x, W_1, W_2)) := \frac{1}{2} \sum_{k=1}^{4} \left(y^{(k)} - h(x^{(k)}, W_1, W_2)\right)^2$$

Hierbei bezeichnet $x^{(k)}$ das k—te Eingabepaar und $y^{(k)}$ die erwartete Ausgabe aus Tabelle 4.1. Berechnen Sie die partiellen Ableitungen:

$\frac{1}{2}\Big((0-0{,}5)^2 + (1- \quad)^2$
$+ (1- \quad)^2 + (0-0{,}5)^2$

- $\frac{\partial QF(y, h(x, W_1, W_2))}{\partial w_{i,j}}$ für alle $w_{i,j}$ in W_1
- $\frac{\partial QF(y, h(x, W_1, W_2))}{\partial w_{i,j}}$ für alle $w_{i,j}$ in W_2

Aufgabe 4.2 (Perzeptron) (6 Punkte)

Bearbeiten Sie das Notebook `Perzeptron.ipynb`.

Aufgabe 4.3 (MNIST) (8 Punkte)

Bearbeiten Sie das Notebook `MNIST_Pytorch.ipynb`.

Bonusaufgabe 4.4 ((0,5 + 0,5 + 1) + 0,5 + (1 + 0,5 + 1) Punkte)

In dieser Aufgabe betrachten wir ein Klassifizierungsproblem mit K Klassen und m Datenpunkten bzw Samples. Beantworten Sie die folgenden Fragen:

(a) Bei Klassifizierern wird häufig als letzte Aktivierungsfunktion die Softmax-Funktion verwendet. Für einen Vektor $x \in \mathbb{R}^K$ wird Softmax für jeden Vektoreintrag $i \in \{1, 2, \dots, K\}$ wie folgt definiert:

$$\text{softmax}(x)_i := \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{4.1}$$

(i) Berechnen Sie $\sum_{i=1}^{K} \text{softmax}(x)_i$.

(ii) Welche Werte kann $\text{softmax}(x)_i$ annehmen?

(iii) Welche Eigenschaft folgt aus (i) und (ii) für $\text{softmax}(x)$?

(b) Labels werden bei Klassifizierern oft one-hot-encoded. Das heißt, statt der Fehlerfunktion nur die Ground-Truth Klasse $c \in \{1, 2, \dots, K\}$ als Skalar zu übergeben, wird oft ein Vektor $y \in \{0, 1\}^K$ erzeugt, der nur an der Stelle $y_c = 1$ und sonst Null ist. Was haben $\text{softmax}(x)$ und y gemeinsam?

(c) Wir bezeichnen mit $y^{(i)} \in \{1, 2, \dots, K\}$ die Ground-Truth Klasse des i—ten Samples und analog mit $\hat{y}^{(i)} \in \{1, 2, \dots, K\}$ die vorhergesagte Klasse des i—ten Sampels. Betrachten Sie nun die Definitionen folgender bekannter Fehlerfunktionen:

$$\text{Cross-Entropy Loss} := \sum_{i=1}^{m} \left(-\sum_{k=1}^{K} y_k^{(i)} \cdot \log(\hat{y}_k^{(i)}) \right) \tag{4.2}$$

$$\text{MSE} := \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} \left(y_k^{(i)} - \hat{y}_k^{(i)} \right)^2 \tag{4.3}$$

1 Wieso ist es besser, bei einem Klassifizierungsproblem das Cross-Entropy Loss zu verwenden anstatt dem Mean-Squared Error (MSE)? Tipp: Denken Sie an das MNIST Beispiel. Je nach Handschrift können sich die Klassen bzw. Zahlen 0 und 8 ähneln. Angenommen ein Bild zeigt tatsächlich eine 0, unser Modell erkennt auf dem Bild jedoch eine 8. Wie gehen die beiden Fehlerfunktionen damit um und welche Auswirkungen wird das vermutlich in zukünftigen Vorhersagen haben?

2 Warum ist es bei der Verwendung von Cross-Entropy sinnvoll, Softmax als letzte Aktivierungsfunktion zu verwenden?

3 Zeigen sie, dass $\text{softmax}(x)$ nicht idempotent ist. Also dass gilt:

$$\text{softmax(softmax(x))} \neq \text{softmax(x)} \tag{4.4}$$