MATHEMATISCHES INSTITUT
DER UNIVERSITÄT ZU KÖLN

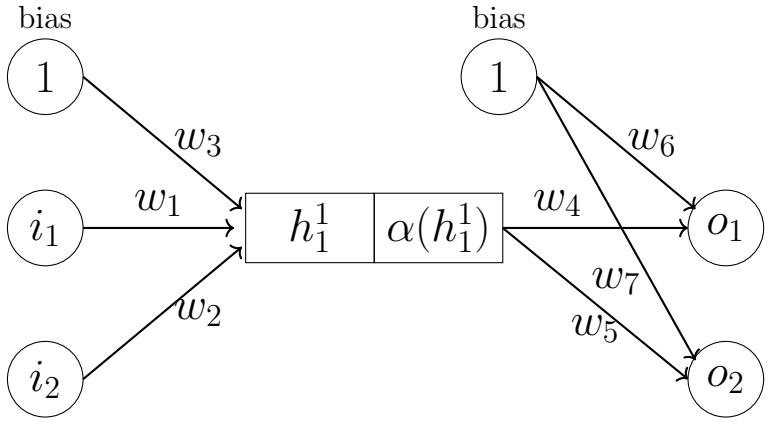
Prof. Dr. A. Klawonn
J. Knepper, M. Sc.
J. Weber, M. Sc.

Wintersemester 2018/2019

16. Januar 2018

13. Übung zu Wissenschaftliches Rechnen I

Aufgabe 1 (Gradienten für ein NN mit einer verdeckten Schicht): (7 Punkte)
Betrachten Sie das folgende dichte feedforward Neuronale Netz mit zwei Eingabe-Neuronen i_1 und i_2, einer verdeckten Schicht mit einem Neuron h_1^1 und zwei Ausgabe-Neuronen o_1 und o_2. Das bedeutet, das Neuronale Netz bekommt als Eingabe pro Datenpunkt den Wert von zwei verschiedenen *features* (da $|I| = 2$; z.B. Punkte in $\mathbb{R}^2$) und klassifiziert die Eingabedaten in zwei unterschiedliche Klassen (da $|O| = 2$).
Wir bezeichnen mit $w_1, \dots, w_7$ die Gewichte zu den jeweiligen Kanten des Neuronalen Netzes. Die Bias-Werte des Vektors b^k werden dabei durch die Gewichte w_3, w_6 und w_7 beschrieben.



$$
\begin{aligned}
h_1^1(i, w) &= w_1 i_1 + w_2 i_2 + w_3 \\
o_1(i, w) &= w_4 \boldsymbol{\alpha}\big(h_1^1(i, w)\big) + w_6 \\
o_2(i, w) &= w_5 \boldsymbol{\alpha}\big(h_1^1(i, w)\big) + w_7
\end{aligned}
$$

Zur Vereinfachung wählen wir in dieser Aufgabe die Identitätsfunktion als Aktivierungsfunktion für die verdeckte Schicht, das heißt, es gilt $\boldsymbol{\alpha}(h_1^1) = h_1^1$ für den Wert h_1^1 des Neurons der verdeckten Schicht in jeder Iteration.
Als Fehlerfunktion $l(\cdot)$ wählen wir die MSE (Mean Squared Error)-Funktion:

$$
l(y, o) = \frac{1}{n} \sum_{i=1}^{n} (y_i - o_i)^2 = \frac{1}{2} \Big[(y_1 - o_1)^2 + (y_2 - o_2)^2 \Big].
$$

Hierbei bezeichnen $n = 2$ die Anzahl der Klassen, o_i die resultierende Ausgabewerte an den Ausgabe-Neuronen und y_i die tatsächlichen Labels der Eingabedaten (*ground truth*). Für den

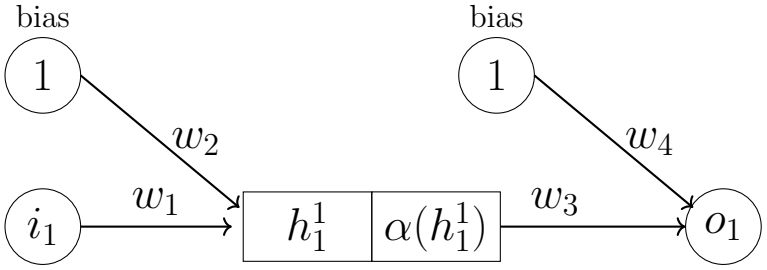
Fall von zwei Klassen $\{0, 1\}$ erhalten wir jeweils als Ausgabewerte für die Ausgabe-Neuronen o_1, o_2 (o_1: Klasse 0, o_2: Klasse 1) die Wahrscheinlichkeit, dass der aktuell betrachtete Datenpunkt zur Klasse 0 bzw. 1 zugeordnet wird. Analog dazu gilt für einen Datenpunkt x_p, dass $y_1 = 1$ und $y_2 = 0$ gilt, falls x_p zur Klasse 0 gehört, bzw. $y_1 = 0$ und $y_2 = 1$, falls x_p zur Klasse 1 gehört.

Leiten Sie für das oben dargestellte Neuronale Netz die allgemeine Formel für den Gradienten aller Gewichte her, wie sie für den Update-Schritt im Gradientenverfahren zum Training des Neuronalen Netzes benötigt wird:

$$\begin{pmatrix} w_1^{k+1} \\ \vdots \\ w_7^{k+1} \end{pmatrix} = \begin{pmatrix} w_1^k \\ \vdots \\ w_7^k \end{pmatrix} - \alpha \begin{pmatrix} \frac{\partial l}{\partial w_1^k} \\ \vdots \\ \frac{\partial l}{\partial w_7^k} \end{pmatrix}.$$

Hierbei bezeichnet α die Lernrate des Verfahrens. Gesucht sind also die Formeln für die einzelnen Ableitungen der Fehlerfunktion l bezüglich der Gewichte $w_1, \ldots, w_7$.

Aufgabe 2 (Trainieren eines NN mit einer verdeckten Schicht): (7+5 = 12 Punkte, 5 Bonuspunkte)
Betrachten Sie eine vereinfachte Version des Neuronalen Netzes aus Aufgabe 1 mit nur einem Eingabe-Neuron i_1 und einem Ausgabe-Neuron o_1.



$$h_1^1(i, w) = w_1 i_1 + w_2$$
$$o_1(i, w) = \boldsymbol{\alpha}\big(h_1^1(i, w)\big) w_3 + w_4$$

Bemerkung: Für den Fall einer Klassifikation in zwei Klassen $\{0, 1\}$ reicht es aus, als Ausgabe für einen Datenpunkt x_p die Wahrscheinlichkeit für die Zugehörigkeit von x_p zu einer der beiden Klassen zu berechnen (Warum?).

Wir wählen wie in Aufgabe 1 die Aktivierungsfunktion $\boldsymbol{\alpha}(x) = x$ und die Fehlerfunktion $l(\cdot)$ als die MSE-Funktion. Im Folgenden sollen Punkte im Intervall $[0, 1] \subset \mathbb{R}$ klassifiziert werden:

$$y_1(x_p) := \begin{cases} 0, & x_p \leq 0.5, \\ 1, & x_p > 0.5. \end{cases}$$

Nutzen Sie dazu das stochastische Gradientenverfahren und wählen Sie in jedem Iterationsschritt genau einen Datenpunkt x_p aus, mit dem Sie aus w^k die neuen Gewichte w^{k+1} berechnen.

(a) Berechnen Sie per Hand die Werte der Gewichte $w_1, \ldots, w_4$ für die ersten 4 Iterationen und wählen Sie

$$w^0 = (w_1^0, w_2^0, w_3^0, w_4^0) := (0.8, 0.0, 0.4, 0.0)$$

als Startvektor sowie $\alpha = 0.1$ für die Lernrate. Wählen Sie in der k-ten Iteration den k-ten Datenpunkt, wobei

$$x_1 = 0.0, \quad x_2 = 0.4, \quad x_3 = 0.6, \quad x_4 = 1.0.$$

Führen Sie dazu in jeder Iteration zunächst einen *Feedforward-Schritt* aus, das heißt, berechnen Sie für die aktuellen Werte der Gewichte w^k den resultierenden Ausgabewert $o_1(i, w)$ für den gegebenen Datenpunkt $i = x_k$. Berechnen Sie dann den Fehler $l = l(y_1(x_k), o_1(i, w^k))$ für die aktuellen Ausgabewerte im Hinblick auf das tatsächliche Label y_1.

Führen Sie danach einen *Backward-Schritt* aus, indem Sie analog zu Aufgabe 1 die Formeln der Gradienten des Fehlers $l(\cdot)$ bzgl. der Gewichte des Neuronalen Netzes berechnen und die Gewichte gemäß der allgemeinen Vorschrift des Gradientenverfahrens aktualisieren.

Überprüfen Sie zum Schluss mit w^4 die Klassifizierung der Trainingsknoten.

(b) Implementieren Sie in Matlab den Trainingsprozess für das oben angegebene Neuronale Netz und überprüfen Sie Ihre Ergebnisse aus Aufgabenteil a).

Wählen Sie nun zufällige Gewichte w_i^0 für die Initialisierung. Erzeugen Sie hierzu zufällige Zahlen aus dem Intervall $[0, 1]$ mit der `rand`-Funktion. Setzen Sie die Lernrate auf $\alpha = 0.01$. Davon abgesehen wählen Sie dieselbe Aktivierungsfunktion, Fehlerfunktion sowie dieselben Eingabedaten x und Labels y wie in Aufgabenteil a). Führen Sie 1 000 Iterationen des Trainingsprozesses durch. Wählen Sie dieses mal in jeder Iteration den Datenpunkt per Zufall aus: `randi([1,4],1)`.

Lassen Sie sich die abschließenden Vorhersagen des Neuronalen Netzes für die gegebenen Trainingsdaten ausgeben und vergleichen Sie diese mit den tatsächlichen Labels. Wie hoch ist die Fehlerrate? Generieren Sie nun die Punkte `linspace(0,1,100)` und klassifizieren Sie diese mit Ihrem Neuronalen Netz. Plotten Sie das Ergebnis.

(c) (Bonusaufgabe) Nutzen Sie nun die Trainingspunkte

```
x = [linspace(0,0.4,10),linspace(0.6,1,10)]
```

mit den Labels

```
y = [zeros(1,10),ones(1,10)].
```

Iterieren Sie 10 000 Schritte und berechnen Sie für jede Iteration die Anzahl Fehlklassifizierungen. Wiederholen Sie dies 100-mal und berechnen Sie damit die durchschnittliche Fehlerrate für jeden Iterationsschritt. Plotten Sie das Ergebnis.

Abgabe: Bis Mittwoch, 23. Januar 2019 , 16:00 Uhr, im entsprechenden Kasten in Raum 3.01 des Mathematischen Instituts.