1. (2 Punkte) Ein Netz gibt für drei Bilder die Wahrscheinlichkeit $\hat{y}$ für die Klasse 1 aus. Berechne jeweils den Fehler $L = -\big(y \ln(\hat{y}) + (1-y)\ln(1-\hat{y})\big)$. Bei welchem Bild liegt das Netz am meisten daneben?
   (a) $\hat{y} = 0.8$, $y = 1$
   (b) $\hat{y} = 0.8$, $y = 0$
   (c) $\hat{y} = 0.3$, $y = 0$

Lösung:
(a) $L = -\ln(0.8) = 0.2231$
(b) $L = -\ln(0.2) = 1.6094$
(c) $L = -\ln(0.7) = 0.3567$
Am meisten daneben liegt das Netz beim zweiten Bild: Es gibt der falschen Klasse 1 die Wahrscheinlichkeit 0.8.

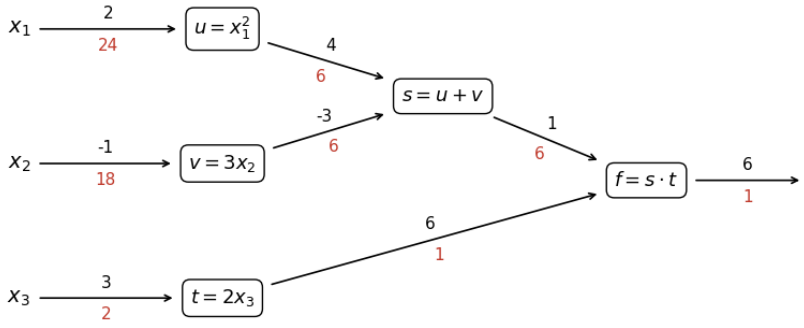
2. (4 Punkte) (a) Zeige, dass für $\sigma(z) = \dfrac{1}{1+e^{-z}}$ gilt: $\sigma'(z) = \sigma(z)\big(1-\sigma(z)\big)$.
   Tipp: Schreibe $\sigma(z) = (1+e^{-z})^{-1}$ und benutze die Kettenregel.
   (b) Für den Fehler gilt $\frac{\partial L}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$ mit $\hat{y} = \sigma(z)$. Zeige, dass $\frac{\partial L}{\partial z} = \hat{y} - y$ ist, getrennt für $y = 1$ und $y = 0$.
   (c) Wo ist $\sigma'(z)$ am größten und wie groß ist es dort?

Lösung:
(a) $\sigma'(z) = -(1+e^{-z})^{-2} \cdot (-e^{-z}) = \dfrac{e^{-z}}{(1+e^{-z})^2} = \dfrac{1}{1+e^{-z}} \cdot \dfrac{e^{-z}}{1+e^{-z}}$.
Der zweite Faktor ist $\dfrac{(1+e^{-z})-1}{1+e^{-z}} = 1 - \dfrac{1}{1+e^{-z}} = 1 - \sigma(z)$. Also $\sigma'(z) = \sigma(z)(1-\sigma(z))$.
(b) Mit $\frac{\partial \hat{y}}{\partial z} = \hat{y}(1-\hat{y})$ und der Kettenregel:
$y = 1$: $\frac{\partial L}{\partial z} = -\frac{1}{\hat{y}} \cdot \hat{y}(1-\hat{y}) = -(1-\hat{y}) = \hat{y} - 1$.
$y = 0$: $\frac{\partial L}{\partial z} = \frac{1}{1-\hat{y}} \cdot \hat{y}(1-\hat{y}) = \hat{y} = \hat{y} - 0$.
(c) Bei $z = 0$, dort ist $\sigma(0) = 0.5$ und $\sigma'(0) = 0.5 \cdot 0.5 = 0.25$. Für große $|z|$ geht $\sigma'(z)$ gegen 0.

3. (3 Punkte) Gegeben ist $f(x_1, x_2, x_3) = (x_1^2 + 3x_2) \cdot 2x_3$ an der Stelle $x_1 = 2$, $x_2 = -1$, $x_3 = 3$.
   (a) Zeichne den Rechengraphen mit den Zwischenergebnissen $u = x_1^2$, $v = 3x_2$, $s = u + v$, $t = 2x_3$ und $f = s \cdot t$. Trage die Werte vorwärts und die Ableitungen von f rückwärts ein.
   (b) Gehe mit $lr = 0.01$ einen Schritt gegen die Ableitungen: $x_i^{neu} = x_i - lr \cdot \frac{\partial f}{\partial x_i}$. Wie groß ist f an der neuen Stelle?

Lösung:

(a) Rechengraph (schwarz die Werte, rot die Ableitungen von f):



f 6 $\partial f/\partial x_1$ 24 $\partial f/\partial x_2$ 18 $\partial f/\partial x_3$ 2

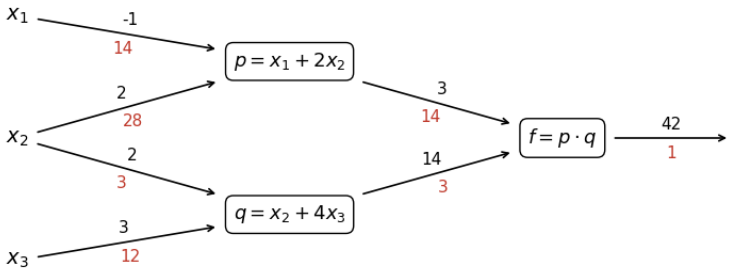
(b) x_1^{neu} 1.76 x_2^{neu} -1.18 x_3^{neu} 2.98 f^{neu} -2.6367

f ist von 6 auf -2.6367 gesunken.

4. (3 Punkte) Gegeben ist $f(x_1, x_2, x_3) = (x_1 + 2x_2)(x_2 + 4x_3)$ an der Stelle $x_1 = -1$, $x_2 = 2$, $x_3 = 3$.
   (a) Zeichne den Rechengraphen mit $p = x_1 + 2x_2$, $q = x_2 + 4x_3$ und $f = p \cdot q$. Trage die Werte vorwärts und die Ableitungen rückwärts ein.
   (b) Bestimme $\frac{\partial f}{\partial x_1}$, $\frac{\partial f}{\partial x_2}$ und $\frac{\partial f}{\partial x_3}$. Was ist bei x_2 anders als bei den anderen beiden?

Lösung:

(a) Rechengraph (schwarz die Werte, rot die Ableitungen von f):



(b) f 42 $\partial f/\partial x_1$ 14 $\partial f/\partial x_2$ 31 $\partial f/\partial x_3$ 12

x_2 kommt auf zwei Wegen zu f, über p und über q. Die Beiträge werden addiert: $14 \cdot 2 + 3 \cdot 1 = 31$.

5. (3 Punkte) Ein Neuron mit einem Input, Sigmoid und Binary Cross-Entropy hat die Parameter $w = 1$, $b = 0.5$. Das Trainingsbeispiel ist $x = -1$, $y = 0$, die Lernrate $lr = 1$.
   (a) Berechne z, $\hat{y}$ und L.
   (b) Berechne $\delta = \hat{y} - y$, $\frac{\partial L}{\partial w}$ und $\frac{\partial L}{\partial b}$.
   (c) Berechne die neuen Parameter und damit noch einmal $\hat{y}$ und L. Ist das Netz besser geworden?

Lösung:

z	-0.5	$\partial L/\partial w$	-0.3775	z^{neu}	-1.2551
$\hat{y}$	0.3775	$\partial L/\partial b$	0.3775	$\hat{y}^{neu}$	0.2218
L	0.4741	w^{neu}	1.3775	L^{neu}	0.2508
δ	0.3775	b^{neu}	0.1225		

Ja, $\hat{y}$ ist näher an $y = 0$, der Fehler ist kleiner geworden.

6. (4 Punkte) Ein Neuron mit zwei Inputs, Sigmoid und Binary Cross-Entropy hat die Parameter $w_1 = 0.5$, $w_2 = -1$, $b = 1$. Das Trainingsbeispiel ist $x_1 = 2$, $x_2 = 1$, $y = 0$, die Lernrate $lr = 0.5$.
   (a) Berechne z, $\hat{y}$ und L.
   (b) Berechne δ und die Ableitungen nach w_1, w_2 und b.
   (c) Berechne die neuen Parameter und damit $\hat{y}$ und L.
   (d) Warum ändert sich w_1 stärker als w_2?

Lösung:

z	1	$\partial L/\partial w_1$	1.4621	w_2^{neu}	-1.3655
$\hat{y}$	0.7311	$\partial L/\partial w_2$	0.7311	b^{neu}	0.6345
L	1.3133	$\partial L/\partial b$	0.7311	$\hat{y}^{neu}$	0.2327
δ	0.7311	w_1^{neu}	-0.2311	L^{neu}	0.2649

Die Ableitung nach einem Gewicht ist δ mal dem Input. $x_1 = 2$ ist doppelt so groß wie $x_2 = 1$, also ist auch die Änderung von w_1 doppelt so groß.

7. (3 Punkte) Ein Neuron mit einem Input und Sigmoid soll aus zwei Trainingsbeispielen lernen: $x = 1$ mit $y = 1$ und $x = -1$ mit $y = 0$. Am Anfang ist $w = 0$, $b = 0$, die Lernrate ist $lr = 1$. Der Fehler ist der Mittelwert der Fehler beider Beispiele.
   (a) Berechne für jedes Beispiel $\hat{y}$, δ, $\frac{\partial L}{\partial w}$ und $\frac{\partial L}{\partial b}$.
   (b) Berechne die gemittelten Ableitungen und die neuen Parameter.
   (c) Berechne den mittleren Fehler vorher und nachher.

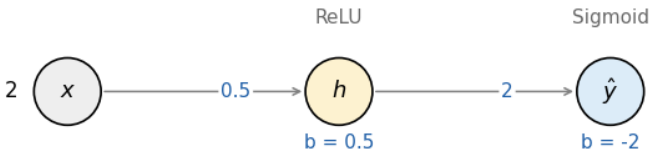
Lösung:

(a) Beispiel 1: $\hat{y} = 0.5$, $\delta = -0.5$, $\frac{\partial L}{\partial w} = -0.5$, $\frac{\partial L}{\partial b} = -0.5$
Beispiel 2: $\hat{y} = 0.5$, $\delta = 0.5$, $\frac{\partial L}{\partial w} = -0.5$, $\frac{\partial L}{\partial b} = 0.5$

(b) $\partial L/\partial w$ -0.5 $\partial L/\partial b$ 0 w^{neu} 0.5 b^{neu} 0
Bei b heben sich die Wünsche der beiden Beispiele auf, bei w zeigen sie in dieselbe Richtung.

(c) L^{vorher} 0.6931 $\hat{y}_1^{neu}$ 0.6225 $\hat{y}_2^{neu}$ 0.3775 $L^{nachher}$ 0.4741

8. (4 Punkte) Ein verstecktes Neuron h mit ReLU und ein Ausgabeneuron mit Sigmoid sind hintereinander geschaltet: $z_1 = w_1 x + b_1$, $h = ReLU(z_1)$, $z_o = w_2 h + b_2$, $\hat{y} = \sigma(z_o)$.

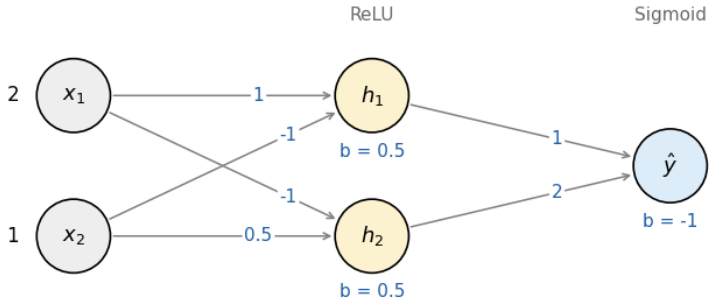


Die Parameter sind $w_1 = 0.5$, $b_1 = 0.5$, $w_2 = 2$, $b_2 = -2$. Das Trainingsbeispiel ist $x = 2$, $y = 0$, die Lernrate $lr = 0.1$.
   (a) Rechne vorwärts: z_1, h, z_o, $\hat{y}$ und L.
   (b) Rechne rückwärts: δ_o, δ_h und die Ableitungen nach allen vier Parametern.
   (c) Berechne die neuen Parameter und damit h, $\hat{y}$ und L.

Lösung:

z_1	1.5	δ_h	1.4621	b_1^{neu}	0.3538
h	1.5	$\partial L/\partial w_2$	1.0966	w_2^{neu}	1.8903
z_o	1	$\partial L/\partial b_2$	0.7311	b_2^{neu}	-2.0731
$\hat{y}$	0.7311	$\partial L/\partial w_1$	2.9242	h^{neu}	0.7689
L	1.3133	$\partial L/\partial b_1$	1.4621	$\hat{y}^{neu}$	0.3499
δ_o	0.7311	w_1^{neu}	0.2076	L^{neu}	0.4306

9. (6 Punkte) Das Netz hat zwei versteckte Neuronen mit ReLU und ein Ausgabeneuron mit Sigmoid.



Die Parameter sind $w_{11} = 1$, $w_{12} = -1$, $b_1 = 0.5$, $w_{21} = -1$, $w_{22} = 0.5$, $b_2 = 0.5$, $v_1 = 1$, $v_2 = 2$, $c = -1$. Dabei ist w_{ji} das Gewicht von x_i zu h_j. Das Trainingsbeispiel ist $x_1 = 2$, $x_2 = 1$, $y = 0$, die Lernrate $lr = 0.1$.

(a) Rechne vorwärts: z_1, h_1, z_2, h_2, z_o, $\hat{y}$ und L.

(b) Rechne rückwärts: δ_o, δ_1, δ_2 und die Ableitungen nach allen neun Parametern.

(c) Berechne die neuen Parameter und damit $\hat{y}$ und L.

(d) Drei Parameter ändern sich nicht. Welche und warum?

Lösung:

(a)

z_1	1.5	z_2	-1	z_o	0.5	L	0.9741
h_1	1.5	h_2	0	$\hat{y}$	0.6225		

(b)

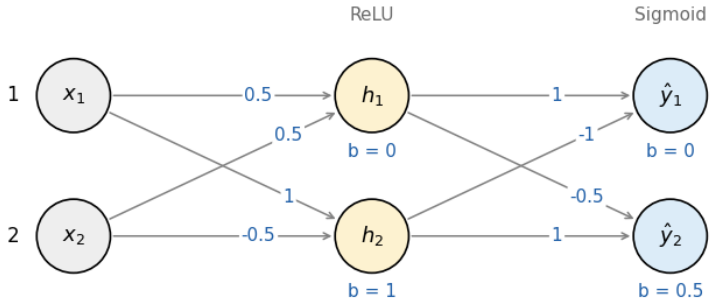
δ_o	0.6225	$\partial L/\partial v_2$	0	$\partial L/\partial b_1$	0.6225
δ_1	0.6225	$\partial L/\partial c$	0.6225	$\partial L/\partial w_{21}$	0
δ_2	0	$\partial L/\partial w_{11}$	1.2449	$\partial L/\partial w_{22}$	0
$\partial L/\partial v_1$	0.9337	$\partial L/\partial w_{12}$	0.6225	$\partial L/\partial b_2$	0

(c)

w_{11}^{neu}	0.8755	v_1^{neu}	0.9066	$\hat{y}^{neu}$	0.4898
w_{12}^{neu}	-1.0622	v_2^{neu}	2	L^{neu}	0.6729
b_1^{neu}	0.4378	c^{neu}	-1.0622		

(d) w_{21}, w_{22} und b_2. Es ist $z_2 < 0$, also $ReLU'(z_2) = 0$ und $\delta_2 = 0$. Damit sind alle Ableitungen nach den Parametern von h_2 gleich 0. (v_2 ändert sich auch nicht, weil $h_2 = 0$ ist.)

10. (5 Punkte) Das Netz hat zwei Ausgaben, der Fehler ist die Summe der Fehler beider Ausgaben.



Die Parameter sind $w_{11} = 0.5$, $w_{12} = 0.5$, $b_1 = 0$, $w_{21} = 1$, $w_{22} = -0.5$, $b_2 = 1$, $v_{11} = 1$, $v_{12} = -1$, $c_1 = 0$, $v_{21} = -0.5$, $v_{22} = 1$, $c_2 = 0.5$. Dabei ist v_{kj} das Gewicht von h_j zu $\hat{y}_k$. Das Trainingsbeispiel ist $x_1 = 1$, $x_2 = 2$, $y_1 = 1$, $y_2 = 1$.

(a) Rechne vorwärts: h_1, h_2, $\hat{y}_1$ und $\hat{y}_2$.

(b) Berechne δ_{o1} und δ_{o2} und die Fehlersignale δ_1 und δ_2 der versteckten Neuronen.

(c) Berechne $\frac{\partial L}{\partial w_{11}}$, $\frac{\partial L}{\partial w_{22}}$ und $\frac{\partial L}{\partial v_{21}}$.

Lösung:

(a) h_1 1.5 h_2 1 $\hat{y}_1$ 0.6225 $\hat{y}_2$ 0.6792

(b) δ_{o1} -0.3775 δ_{o2} -0.3208 δ_1 -0.2171 δ_2 0.0567

$\delta_1 = (\delta_{o1} v_{11} + \delta_{o2} v_{21}) \cdot 1$, $\delta_2 = (\delta_{o1} v_{12} + \delta_{o2} v_{22}) \cdot 1$, weil beide versteckten Neuronen aktiv sind.

(c) $\partial L/\partial w_{11}$ -0.2171 $\partial L/\partial w_{22}$ 0.1134 $\partial L/\partial v_{21}$ -0.4812

11. (4 Punkte) Ein verstecktes Neuron h mit ReLU und ein Ausgabeneuron mit Sigmoid sind hintereinander geschaltet wie in der Aufgabe mit der Kette: $z_1 = w_1 x + b_1$, $h = ReLU(z_1)$, $z_o = w_2 h + b_2$, $\hat{y} = \sigma(z_o)$. Die Parameter sind $w_1 = 1$, $b_1 = 0.5$, $w_2 = 2$, $b_2 = -1$. Das Trainingsbeispiel ist $x = 1$, $y = 0$, die Lernrate $lr = 1$.

    (a) Rechne einen Schritt vorwärts und rückwärts und berechne die neuen Parameter.

    (b) Berechne mit den neuen Parametern z_1, h und $\hat{y}$.

    (c) Warum ändern sich w_1 und b_1 bei diesem Trainingsbeispiel nie mehr?

    (d) Für welche Inputs $x \geq 0$ ist das Neuron jetzt tot? Was bedeutet das für die Ausgabe des Netzes?

Lösung:

(a)

z_1	1.5	$\hat{y}$	0.8808	w_1^{neu}	-0.7616	b_2^{neu}	-1.8808
h	1.5	δ_o	0.8808	b_1^{neu}	-1.2616		
z_o	2	δ_h	1.7616	w_2^{neu}	0.6788		

(b) z_1^{neu} -2.0232 h^{neu} 0 $\hat{y}^{neu}$ 0.1323

(c) Weil $z_1 < 0$ ist, ist $ReLU'(z_1) = 0$ und damit $\delta_h = 0$. Die Ableitungen nach w_1 und b_1 sind 0, das Update ändert sie nicht. Also bleibt z_1 negativ.

(d) w_1 und b_1 sind beide negativ, also ist $z_1 = w_1 x + b_1 < 0$ für alle $x \geq 0$. Für diese Inputs ist $h = 0$ und $\hat{y} = \sigma(b_2)$, die Ausgabe hängt gar nicht mehr vom Input ab. Der Schritt mit $lr = 1$ war zu groß.

12. (3 Punkte) Für die Kette aus verstecktem Neuron (ReLU) und Ausgabeneuron (Sigmoid) mit $w_1 = -1$, $b_1 = 0.5$, $w_2 = 1$, $b_2 = 0.5$. Das Trainingsbeispiel ist $x = 1$, $y = 1$, die Lernrate $lr = 0.5$. hat jemand so gerechnet:

$$z_1 = -1 \cdot 1 + 0.5 = -0.5,\ h = 0 \qquad z_o = 1 \cdot 0 + 0.5 = 0.5,\ \hat{y} = 0.6225$$
$$\delta_o = 0.6225 - 1 = -0.3775 \qquad \delta_h = \delta_o \cdot w_2 = -0.3775$$
$$\frac{\partial L}{\partial b_2} = \delta_o = -0.3775 \qquad b_2^{neu} = 0.5 + 0.5 \cdot (-0.3775) = 0.3112$$

Finde die beiden Fehler und gib die richtigen Werte für δ_h und b_2^{neu} an.

Lösung: Erster Fehler: Bei δ_h fehlt der Faktor $ReLU'(z_1)$. Wegen $z_1 < 0$ ist er 0, richtig ist $\delta_h = 0$.

Zweiter Fehler: Beim Update muss man die Ableitung abziehen, nicht addieren: $b_2^{neu} = 0.5 - 0.5 \cdot (-0.3775) = 0.6888$.

13. (3 Punkte) Wir nehmen das Neuron aus der Aufgabe mit einem Input: $x = -1$, $w = 1$, $b = 0.5$, $y = 0$. Backpropagation liefert dort $\frac{\partial L}{\partial w} = -0.3775$.

    (a) Berechne den Fehler L für $w = 1.01$ und für $w = 0.99$ (die anderen Werte bleiben gleich) und daraus

    $$\frac{L(w + 0.01) - L(w - 0.01)}{0.02}$$

    Vergleiche mit dem Wert aus Backpropagation.

    (b) Ein Netz hat eine Million Parameter. Wie viele Durchgänge vorwärts braucht man, um alle Ableitungen auf diese Weise zu berechnen? Wie viele Durchgänge braucht Backpropagation?

Lösung:

(a) $L(1.01)$ 0.4703 $L(0.99)$ 0.4779 Quotient -0.3775
Der Quotient stimmt bis auf die vierte Nachkommastelle mit der Ableitung aus Backpropagation überein.

(b) Zwei Durchgänge pro Parameter, also zwei Millionen. Backpropagation braucht einen Durchgang vorwärts und einen rückwärts.

14. (4 Punkte) (a) Was gibt das Fehlersignal δ eines Neurons an?

    (b) Warum berechnet man die Ableitungen rückwärts, von der Ausgabe zum Input?

    (c) Was ist eine Epoche? Wie geht man vor, wenn es mehrere Trainingsbeispiele gibt?

    (d) Warum benutzt man in den versteckten Schichten tiefer Netze ReLU statt Sigmoid?

Lösung:

(a) $\delta = \frac{\partial L}{\partial z}$ ist die Ableitung des Fehlers nach der Summe z des Neurons. Es sagt, wie stark und in welche Richtung sich der Fehler ändert, wenn z ein wenig größer wird. Die Ableitung nach einem Gewicht ist δ mal dem Wert, der über das Gewicht hereinkommt.

(b) Das δ eines Neurons wird aus den δ-Werten der Neuronen dahinter berechnet. Rückwärts kann man jedes δ einmal berechnen und für alle Gewichte davor wiederverwenden. Ein Durchgang rückwärts liefert so alle Ableitungen.

(c) Ein Durchgang durch alle Trainingsbeispiele mit anschließendem Update der Parameter. Der Fehler ist der Mittelwert der Fehler der einzelnen Beispiele, deshalb mittelt man auch die Ableitungen.

(d) Die Ableitung der Sigmoidfunktion ist höchstens 0.25. Auf dem Weg zurück wird δ in jeder Schicht damit multipliziert und wird in tiefen Netzen verschwindend klein. Die Ableitung der ReLU ist 1 für aktive Neuronen.

15. (4 Punkte) Erweitere die Klasse `Wert` aus dem Notebook um eine Methode `tanh`. Die Funktion $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ liefert Werte zwischen -1 und 1, ihre Ableitung ist $1 - \tanh(z)^2$. In Python berechnet `math.tanh` den Funktionswert.

    Teste die Methode mit $f = \tanh(w \cdot x + b)$ für $w = 0.5$, $x = 2$, $b = -0.5$: Berechne `w.grad` und vergleiche mit der numerischen Ableitung.

Lösung:

```
def tanh(self):
    t = math.tanh(self.wert)
    erg = Wert(t, (self,))
    def zurueck():
        self.grad += (1 - t ** 2) * erg.grad
    erg._zurueck = zurueck
    return erg

Wert.tanh = tanh

w, x, b = Wert(0.5), Wert(2), Wert(-0.5)
f = (w * x + b).tanh()
f.rueckwaerts()
print(w.grad)

eps = 0.0001
num = (math.tanh((0.5 + eps) * 2 - 0.5) - math.tanh((0.5 - eps) * 2 - 0.5)) / (2 * eps)
print(num)
```

Ergebnis: `w.grad` $= 2 \cdot (1 - \tanh(0.5)^2) = 1.5729$, die numerische Ableitung stimmt damit überein. (In der Klasse kann man die Methode natürlich auch direkt neben `sigmoid` schreiben.)