Einführung in die Künstliche Intelligenz

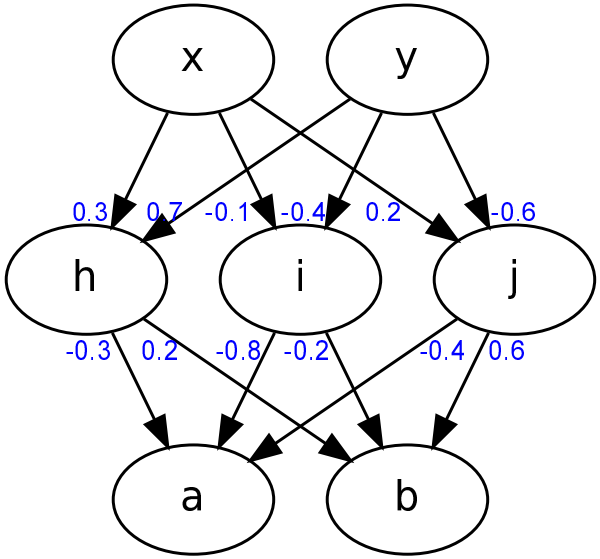
WS14/15 - Eneldo Loza Mencia

TECHNISCHE UNIVERSITÄT DARMSTADT

Beispiellösung für das 6. Übungsblatt

Aufgabe 1 Neuronale Netze

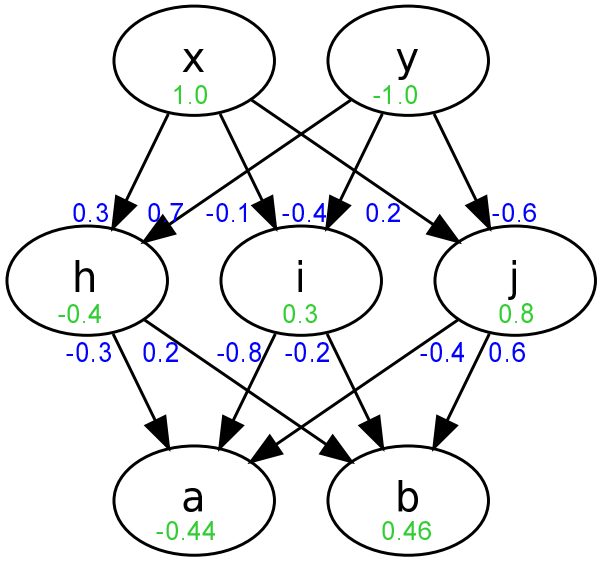
Gegeben sei folgendes Neuronales Netz mit der Identität als Aktivierungsfunktion, d.h. $g(x) = x$.



$W_{x,h} = 0.3$ $W_{h,a} = -0.3$
$W_{x,i} = -0.1$ $W_{i,a} = -0.8$
$W_{x,j} = 0.2$ $W_{j,a} = -0.4$
$W_{y,h} = 0.7$ $W_{h,b} = 0.2$
$W_{y,i} = -0.4$ $W_{i,b} = 0.2$
$W_{y,j} = -0.6$ $W_{j,b} = 0.6$

a) Berechnen Sie die Outputs (a, b) für die Eingabe $x = 1$ und $y = -1$. Geben Sie auch alle relevanten Zwischenresultate an (z.B. die Aktivierung der Zwischenknoten).

Berechnen Sie die Outputs (a, b) für die Eingabe $x = 1$ und $y = -1$. Geben Sie auch alle relevanten Zwischenresultate an (z.B. die Aktivierung der Zwischenknoten).



$$in_h = W_{x,h} \cdot x + W_{y,h} \cdot y = 0.3 \cdot 1 + 0.7 \cdot (-1) = -0.4$$
$$in_i = W_{x,i} \cdot x + W_{y,i} \cdot y = -0.1 \cdot 1 + (-0.4) \cdot (-1) = 0.3$$
$$in_j = W_{x,j} \cdot x + W_{y,j} \cdot y = 0.2 \cdot 1 + (-0.6) \cdot (-1) = 0.8$$

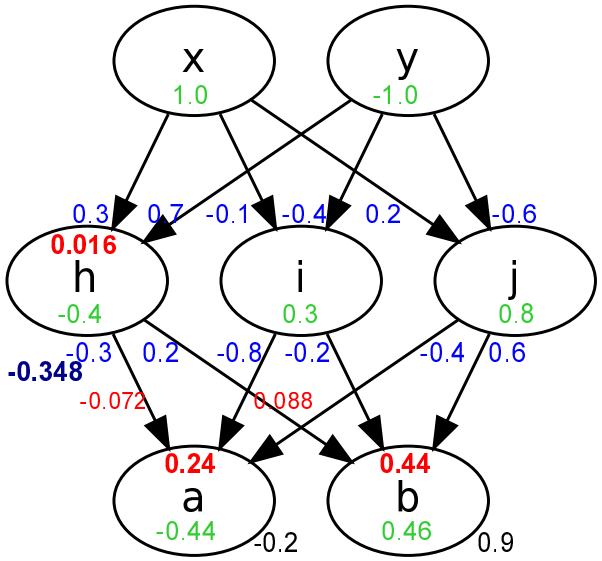
Die angegebene Aktivierungsfunktion $g(x) = x$ gibt die Aktivierungswerte unverändert weiter, d.h. $out_x = in_x$.

$$\begin{aligned} in_a &= W_{h,a} \cdot out_h + W_{i,a} \cdot out_i + W_{j,a} \cdot out_j \\ &= (-0.3) \cdot (-0.4) + (-0.8) \cdot 0.3 + (-0.4) \cdot 0.8 = -0.44 \\ in_b &= W_{h,b} \cdot out_h + W_{i,b} \cdot out_i + W_{j,b} \cdot out_j \\ &= 0.2 \cdot (-0.4) + 0.2 \cdot 0.3 + 0.6 \cdot 0.8 = 0.46 \end{aligned}$$

Die Ausgabewerte bleiben wiederum unverändert.

b) Nehmen Sie nun an, dass das Netzwerk für obigen Input $(x, y) = (1, -1)$ die Ausgabe $(a, b) = (-0.2, 0.9)$ liefern soll. Die Lernrate sei $\alpha = 0.5$.

Nehmen Sie nun an, dass das Netzwerk für obigen Input $(x, y) = (1, -1)$ die Ausgabe $(a, b) = (-0.2, 0.9)$ liefern soll. Die Lernrate sei $\alpha = 0.5$.



1. Berechnen Sie die Fehlerterme Δ_a und Δ_b

$$g'(x) = 1$$
$$\Delta_a = Err_a \cdot g'(in_a) = (-0.2 - (-0.44)) \cdot 1 = 0.24$$
$$\Delta_b = Err_b \cdot g'(in_b) = (0.9 - 0.46) \cdot 1 = 0.44$$

2. Berechnen Sie die Fehlerrate Δ_h

$$\Delta_h = W_{h,a} \cdot \Delta_a \cdot g'(in_h) + W_{h,b} \cdot \Delta_b \cdot g'(in_h)$$
$$= (-0.3) \cdot 0.24 + 0.2 \cdot 0.44 = -0.072 + 0.088 = 0.016$$

3. Berechnen Sie die Gewichtsänderung für das Gewicht $W_{h,a}$

   $W_{h,a} \leftarrow W_{h,a} + \alpha \cdot \Delta_a \cdot out_h = -0.3 + 0.5 \cdot 0.24 \cdot (-0.4) = -0.348$

   Diese würde beim gleichen Eingangsbeispiel das Ausgangssignal $out_a (= in_a)$ um $(-0.348 \cdot -0.4) - (-0.3 \cdot -0.4) = 0.1392 - 0.12 = 0.0192$ zum gewünschten Signal -0.2 verschieben.

c) Angenommen, Sie können den Hidden Layer dieses Netzes beliebig vergrößern. Welche Art von Funktionen könnten Sie dann in den Outputs a und b zumindest lernen? Was ändert sich, wenn beliebige Aktivierungsfunktionen verwendet werden können?

Angenommen, Sie können den Hidden Layer dieses Netzes beliebig vergrößern. Welche Art von Funktionen könnten Sie dann in den Outputs a und b zumindest lernen? Was ändert sich, wenn beliebige Aktivierungsfunktionen verwendet werden können?

Mit der Identität als Aktivierungsfunktion stellt jedes Neuron eine Linearkombination ihrer Eingaben dar. Da Linearkombinationen von Linearkombinationen wiederum Linearkombinationen sind, können im ersten Fall nur lineare Funktionen gelernt werden. Ist jede beliebige Aktivierungsfunktion erlaubt, können alle stetigen Funktionen gelernt werden.

Aufgabe 2 Logische Funktionen

Geben Sie für die folgenden Funktionen jeweils ein neuronales Netz an (Struktur, Vernetzung und Gewichte), welches die Funktion umsetzt. Gehen Sie dabei von einem neuronalen Netz mit der Schwellwertfunktion $g(x) = 1$ für $x > 0$ und sonst $g(x) = 0$, und den Eingangs- und Ausgangssignalen 0 und 1 für logisch *false* und *true* aus. Geben Sie auch jeweils die Wahrheitstabellen mit den Eingangssignalen jedes einzelnen Neurons an (vor Anwendung der Schwellwertfunktion).

Es gibt jeweils verschiedene Lösungen. Versuchen Sie eine möglichst kompakte Lösung zu finden, d.h., ein Netzwerk mit möglichst wenigen Neuronen und Layern.

a) x AND y

b) x OR y

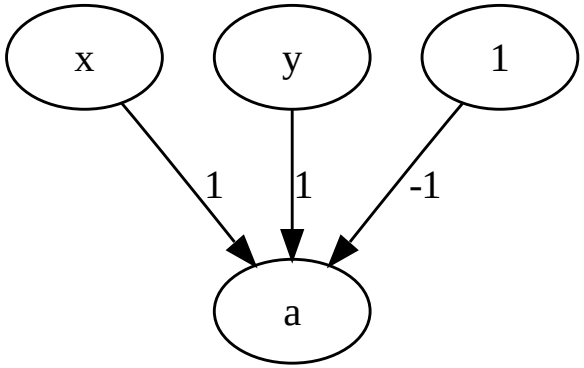
c) $(x$ OR $y)$ AND z

d) x XOR y

a) x AND y

Wir nutzen hierfür aus, dass Neuronen im Prinzip nichts anderes machen als zu addieren. Damit x AND y wahr ist, muss die Summe von x und y 2 sein. Bei jeder anderen Kombination wäre die Summe höchstens 1. Um den Schwellwert demnach z.B. auf 1 statt 0 zu setzen (jeder Schwellwert $1 \leq t < 2$ wäre für AND möglich) benutzen wir einen Bias-Eingang (Eingang mit konstant Signal 1) und setzen das Gewicht auf das Negative des Schwellwerts, also $-t$.

Das resultierende Netzwerk sieht folgendermaßen aus:

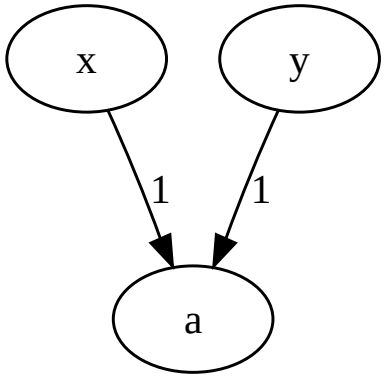


Wir überprüfen die Korrektheit des Netwerkes anhand der Wahrheitstabelle:

x	y	in_a	out_a
0	0	-1	0
0	1	0	0
1	0	0	0
1	1	1	1

b) x OR y

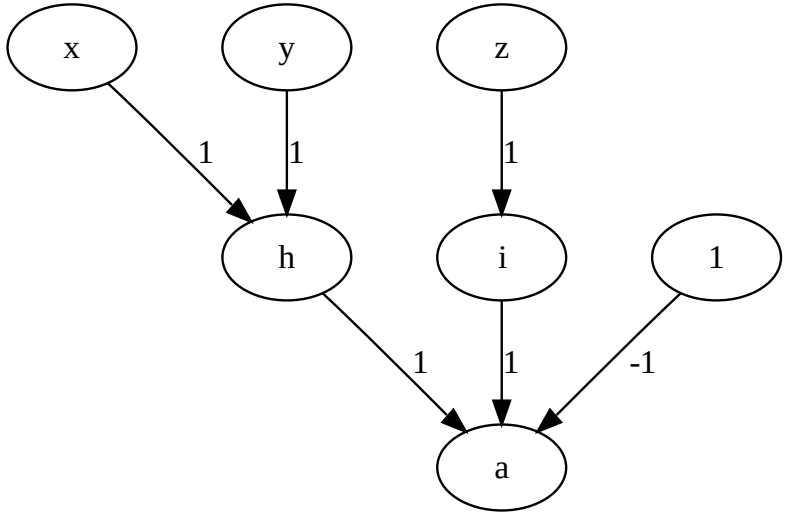
Wir sehen an der Wahrheitstabelle für AND recht einfach, daß wir für die OR-Funktion lediglich den Schwellwert anpassen müssen. Dieser ist nun 0, also benötigen wir auch nicht den Bias-Eingang:



x	y	in_a	out_a
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	1

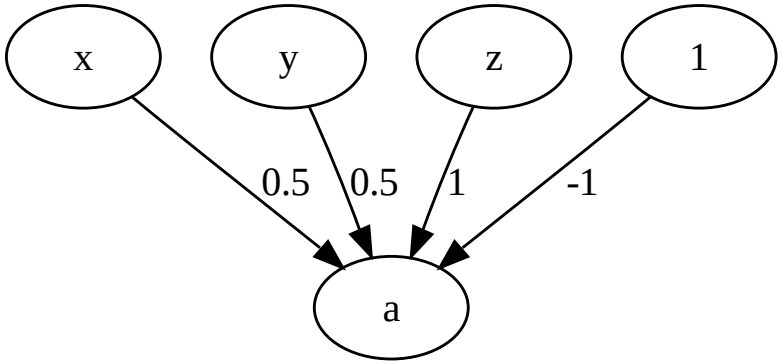
c) $(x \text{ OR } y) \text{ AND } z$

Diese Funktion können wir anhand einfacher Verkettung der vorherigen Netwerke erhalten:



Jedoch benötigen wir hierfür eine zusätzliche Schicht.

Um ein kompakteres neuronales Netz zu erhalten, können wir auf die Additionsfähigkeit zurückgreifen und lassen das Netzwerk auf $z + \frac{1}{2}x + \frac{1}{2}y > 1$ testen:



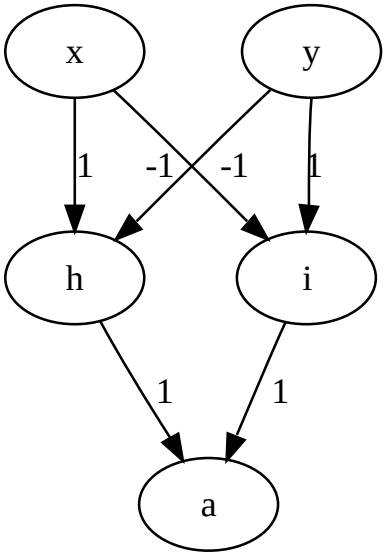
Die Wahrheitstabelle bestätigt die korrekte Funktionsweise:

x	y	z	in_a	out_a
0	0	0	-1	0
0	0	1	0	0
0	1	0	-0.5	0
0	1	1	0.5	1
1	0	0	-0.5	0
1	0	1	0.5	1
1	1	0	0	0
1	1	1	1	1

d) x XOR y

Die XOR-Funktion lässt sich ohne Hidden Layer mit einem normalen Netzwerk nicht erlernen oder berechnen. Der Grund ist, daß sich keine lineare Trennebene zwischen positiven und negativen Beispielen finden lässt. Anders ausgedrückt, x XOR y lässt sich nicht als Summe von x und y und Schwellwert t angeben, d.h. $x + y > t$.

Wir benötigen demnach ein Hidden Layer zwischen Eingangs- und Ausgangsschicht. Das im Folgenden gewählte Modell entspricht der Auflösung x XOR $y = (x$ AND $\neg\, y)$ OR$(\neg\, x$ AND $y)$:



Die Wahrheitstabelle ist nun:

x	y	in_h	in_i	out_h	out_i	in_a	out_a
0	0	0	0	0	0	0	0
0	1	-1	1	0	1	1	1
1	0	1	-1	1	0	1	1
1	1	0	0	0	0	0	0