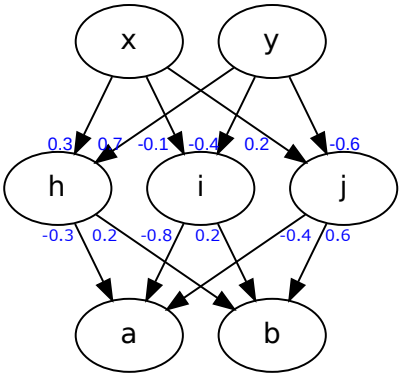
Einführung in die
Künstliche Intelligenz
SS17 - Christian Wirth

TECHNISCHE UNIVERSITÄT DARMSTADT

Beispiellösung für das 6. Übungsblatt

Aufgabe 1 Neuronale Netze

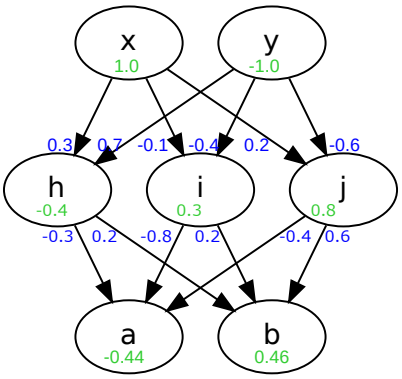
Gegeben sei folgendes Neuronales Netz mit der Identität als Aktivierungsfunktion, d.h. $g(x) = x$.



$$
\begin{array}{ll}
W_{x,h} = 0.3 & W_{h,a} = -0.3 \\
W_{x,i} = -0.1 & W_{i,a} = -0.8 \\
W_{x,j} = 0.2 & W_{j,a} = -0.4 \\
W_{y,h} = 0.7 & W_{h,b} = 0.2 \\
W_{y,i} = -0.4 & W_{i,b} = 0.2 \\
W_{y,j} = -0.6 & W_{j,b} = 0.6
\end{array}
$$

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).



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

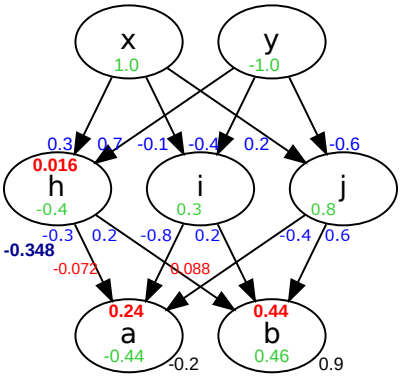
Die angegebene Aktivierungsfunktion $g(x) = x$ gibt die Aktivierungswerte unverändert weiter, d.h. $out_x = in_x$.

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

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

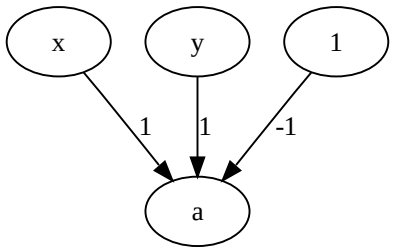
c) $(x$ OR $y)$ AND z

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:

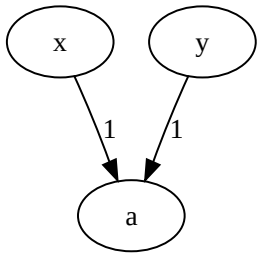


Wir überprüfen die Korrektheit des Netwerkes anhand der Wahrheitstabelle:

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

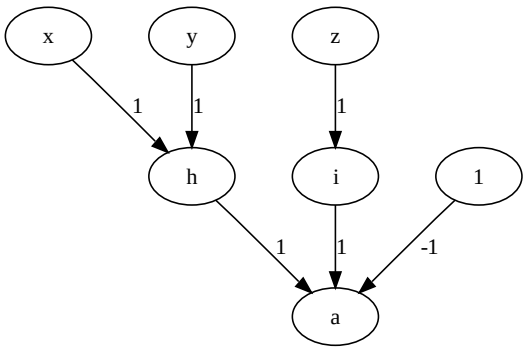
Wir sehen an der Wahrheitstabelle für AND recht einfach, daß wir für die OR-Funktion lediglich den Schwellwert anpassen müssen. Dieser ist nun 0, also benötigen wir auch nicht den Bias-Eingang:



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

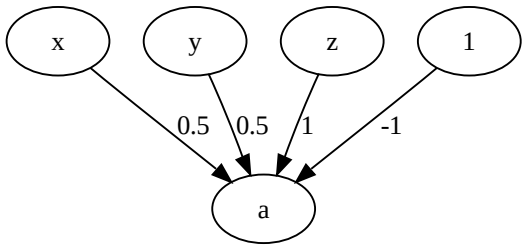
c) $(x$ OR $y)$ AND z

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:



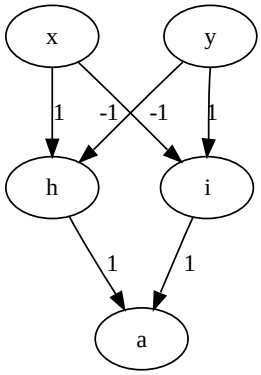
Die Wahrheitstabelle bestätigt die korrekte Funktionsweise:

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 $y = (x \text{ AND } \neg\, y) \text{ OR} (\neg\, x \text{ AND } y)$:



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

Aufgabe 3 Reinforcement Learning

1. Laut Aufgabenstellung erhalten nur (unmittelbare) Aktionen einen Reward (1), die zur Folge haben, dass der Agent sich daraufhin im Feld f befindet. Diese Aktionen sind im Zustand c nach unten zu gehen und im Zustand e sich nach rechts zu bewegen. Alle anderen Aktionen erhalten einen Reward von 0.

$$\begin{array}{lllll} r(a,r)=0 & r(b,r)=0 & r(c,u)=1 & r(d,o)=0 & r(e,o)=0 \\ r(a,u)=0 & r(b,u)=0 & r(c,l)=0 & r(d,r)=0 & r(e,r)=1 \\ & r(b,l)=0 & & & r(e,l)=0 \end{array}$$

2. Der akkumulierte erwartete Reward eines Zustandes s wird folgendermaßen berechnet: $V^{\pi}(s) = \sum_{k=0}^{\infty} \gamma^k \cdot r_k$, wobei die Rewards r_k, den Rewards entsprechen, die man erhält, wenn man vom Anfangszustand s aus Aktionen gemäß der Policy π ausführt. Als Beispiel wird die Berechnung von $V^{\pi}(d)$ dargestellt: Laut Policy bewegt sich der Agent ausgehend von d wie folgt: $\rightarrow a \rightarrow b \rightarrow c \rightarrow f$, wobei im Feld f keine Aktion mehr möglich ist.

Die Bewertung $V^\pi(d)$ ergibt sich also als:

$$\begin{aligned} V^\pi(d) &= \gamma^0 \cdot r(d,o) + \gamma^1 \cdot r(a,r) + \gamma^2 \cdot r(b,r) + \gamma^3 \cdot r(c,u) \\ &= 1 \cdot 0 + 0.8 \cdot 0 + 0.8^2 \cdot 0 + 0.8^3 \cdot 1 \\ &= 0.512 \end{aligned}$$

Analog berechnet man die Bewertungen der restlichen Felder:

$V^\pi(a) = 0.64$	$V^\pi(b) = 0.8$	$V^\pi(c) = 1$
$V^\pi(d) = 0.512$	$V^\pi(e) = 0.64$	

3. POLICYIMPROVEMENT ändert die aktuelle Policy π für einen Zustand s um, indem sie die Aktion a selektiert, die folgendes maximiert:

$$\max_a r(s,a) + \gamma \cdot V^\pi(s') \qquad \text{wobei } s' = \delta(s,a)$$

Die aktuelle Policy $\pi(e)$ für den Zustand e würde einen Schritt nach **oben** vorgeben, mit der Gesamt-Bewertung 0.64. Für die anderen Aktionen ergibt sich:

links: $r(e,l) + \gamma \cdot V^\pi(d) = 0 + 0.8 \cdot 0.512 = 0.4096$
rechts: $r(e,r) = 1$

Da die Aktion *rechts* im Zustand e die beste Bewertung hat, würde die aktuelle Policy für den Zustand e mit der Anweisung $\pi'(e) = r$ überschrieben werden.

4. Wir überlegen uns für jedes Feld, welches ein optimaler Weg zu f wäre. Beispielsweise würde für das Feld a der Weg $\rightarrow b \rightarrow c \rightarrow f$ einen optimalen Reward erzielen, genauso wie $\rightarrow d \rightarrow e \rightarrow f$, nämlich:

$$= 0.64$$

Analog berechnet man die Bewertungen der restlichen Felder und erhält:

$V^*(a) = 0.64$	$V^*(b) = 0.8$	$V^*(c) = 1$
$V^*(d) = 0.8$	$V^*(e) = 1$	

Die optimale Q-Funktion für alle Zustandspaare lässt sich nun mit den berechneten optimalen Bewertungsfunktionen recht einfach berechnen. Wie aus der Vorlesung bekannt, gilt für die optimale Q-Funktion :

$$Q(s,a) = r(s,a) + \gamma \cdot V^*(s')$$

Im Feld a erhalten wir beispielsweise :
$Q(a,r) = r(a,r) + \gamma \cdot V^*(b) = 0 + 0.8 \cdot 0.8 = 0.64$
$Q(a,u) = r(a,u) + \gamma \cdot V^*(d) = 0 + 0.8 \cdot 0.8 = 0.64$
Insgesamt ergibt dies:

$Q(a,r) = 0.64$	$Q(b,r) = 0.8$	$Q(c,u) = 1$	$Q(d,o) = 0.512$	$Q(e,o) = 0.64$
$Q(a,u) = 0.64$	$Q(b,u) = 0.8$	$Q(c,l) = 0.64$	$Q(d,r) = 0.8$	$Q(e,r) = 1$
	$Q(b,l) = 0.512$			$Q(e,l) = 0.64$

5. Die optimale Policy wählt in jedem Feld diejenige Aktion aus, die den höchsten akkumulierten erwarteten Reward verspricht:

$$\begin{aligned} \pi^*(s) &= \text{argmax}_a \, r(s,a) + \gamma \cdot V^*(s') \\ &= \text{argmax}_a \, Q(s,a) \end{aligned}$$

Mithilfe der vorigen Teilaufgaben lässt sich einfach die optimale Policy ablesen, indem man für jeden Zustand die Aktion wählt, die den höchsten Q-Wert aufweist. Insgesamt ergibt das folgende graphisch dargestellte Policy:

$\downarrow\rightarrow$	$\downarrow\rightarrow$	$\downarrow$
$\rightarrow$	$\rightarrow$	

6. Alle Werte $\hat{Q}(s,a)$ werden zunächst auf null gesetzt. Wir verwenden folgende graphische Ansicht der $\hat{Q}$-Werte:

a	$\hat{Q}(a,r)=0$	$\hat{Q}(b,l)=0$	b	$\hat{Q}(b,r)=0$	$\hat{Q}(c,l)=0$	c
$\hat{Q}(a,u)=0$			$\hat{Q}(b,u)=0$			$\hat{Q}(c,u)=0$
$\hat{Q}(d,o)=0$			$\hat{Q}(e,o)=0$			$\hat{Q}(f,o)=0$
d	$\hat{Q}(d,r)=0$	$\hat{Q}(e,l)=0$	e	$\hat{Q}(e,r)=0$	$\hat{Q}(f,l)=0$	f

Wir wählen zufällig ein Feld aus, sagen wir d. Da die beiden Aktionen o und r gleich bewertet sind, wählen wir erneut zufällig die Aktion r.

Nun ergibt sich der neue Wert $\hat{Q}(d,r) = \hat{Q}(d,r) + \alpha[r(d,r) + \gamma \cdot \max_a \hat{Q}(e,a) - \hat{Q}(d,r)]$. Da α auf 1 gesetzt wurde, wird der alte Wert nicht berücksichtigt, d.h. als Update-Regel wird $\hat{Q}(d,r) = r(d,r) + \gamma \cdot \max_a \hat{Q}(e,a)$ verwendet.

Darüber hinaus sind alle $\hat{Q}$-Werte von e auf 0 gesetzt, so dass $\hat{Q}(d,r) = 0 + 0.8 \cdot 0 = 0$.

Im Feld e wählen wir zufällig die Aktion r : $\hat{Q}(e,r) = 1$
Die momentanen $\hat{Q}$-Werte sehen dann wie folgt aus:

a	0.0	0.0	b	0.0	0.0	c
0.0			0.0			0.0
0.0			0.0			0.0
d	0.0	0.0	e	1.0	0.0	f

In einer weiteren Iteration starten wir von b und wählen die Aktion u. Es ergibt sich nun $\hat{Q}(b,u) = r(b,u) + \gamma \cdot \max_a \hat{Q}(e,a)$. Laut unserer aktuellen Q-Funktion ist im Feld e die optimale Aktion mit 1.0 bewertet, deshalb erhalten wir $\hat{Q}(b,u) = 0 + 0.8 \cdot 1 = 0.8$. Im Feld e wird daraufhin die Aktion r gewählt und die Q-Werte ändern sich nicht.

a	0.0	0.0	b	0.0	0.0	c
0.0			0.8			0.0
0.0			0.0			0.0
d	0.0	0.0	e	1.0	0.0	f

Ein weiterer Durchlauf sei folgenden Weg gegangen : $d \to e \to f$ (wobei die Wahl der nächsten Aktion im Feld e eindeutig war).

a	0.0	0.0	b	0.0	0.0	c
0.0			0.8			0.0
0.0			0.0			0.0
d	0.8	0.0	e	1.0	0.0	f

Weitere Durchläufe ergaben folgende Wege : $c \to f$ und $a \to b \to e \to f$

a	0.64	0.0	b	0.0	0.0	c
0.0			0.8			1.0
0.0			0.0			0.0
d	0.8	0.0	e	1.0	0.0	f

In weiteren Durchläufen finden keine weiteren Änderungen an den Q-Werten mehr statt. Insgesamt wurde eine (pseudo-)optimale Policy gefunden, die unten zu sehen ist. Beachten Sie, dass die ermittelten Q-Werte nicht immer optimal sein müssen. Die Konvergenz von Q-LEARNING an die optimale Q-Funktion gilt im Allgemeinen nur, wenn jedes Zustands-Aktions Paar beliebig oft besucht wird.

$\to$	$\downarrow$	$\downarrow$
$\to$	$\to$	

Ein Beispiel für Simulationssequenzen, so dass Q-LEARNING mit einer minimalen Anzahl an Updates konvergiert: $c \to f, e \to f, b \to c \to f, d \to e \to f, a \to b \to c \to f$

7. Analog zur Q-Funktion für nicht-deterministische Übergänge muss nun die Bewertungsfunktion $V^\pi(s)$ modifiziert werden. Das heißt, nun müssen die Wahrscheinlichkeiten für die Übergänge mit Betrachtet werden. $V^\pi(s_t) = r(s_t,a_t) + \gamma V^\pi(\delta(s_t,a_t))$ wird daher zu $V^\pi(s_t) = r(s_t,a_t) + \gamma \sum_{s'} P(s'|s,a) V^\pi(s')$

   Wir haben nun Wahrscheinlichkeiten $P(s_{t+1}|s_t,a_t) = 0.9$ und $P(s_{\text{dead}}|s_t,a_t) = 0.1$. Da in s_{dead} keine Aktionen mehr möglich sind können wir von $V^\pi(s_{\text{dead}}) = 0$ ausgehen. Somit vereinfacht sicht in diesem Fall die Formel zu $V^\pi(s_t) = r(s_t,a_t) + \gamma \cdot 0.9 \cdot V^\pi(\delta(s_t,a_t))$

   Es ergibt sich daher:

$V^\pi(a) = 0.52$	$V^\pi(b) = 0.72$	$V^\pi(c) = 1$
$V^\pi(d) = 0.37$	$V^\pi(e) = 0.52$	

8. Da unser Reward nun vom Folgezustand abhängt, müssen wir mit $r(s, a, s')$ arbeiten. Somit wird die Bewertungsfunktion zu $V^{\pi}(s) = \sum_{s'} P(s'|s, a)(r(s, a, s') + \gamma V^{\pi}(s'))$