Aufgabe: Konzipieren Sie ein logisches XOR-Gatter mit Hilfe eines neuronalen Netzwerks.

Lösung: Die Wahrheitstafel der Kontravalenz mit zwei Eingangsaktivitäten $x_1^{(1)}$ und $x_2^{(1)}$, zwei verdeckten Aktivitäten $x_1^{(2)}$ und $x_2^{(2)}$ und einer Ausgangsaktivität $x^{(3)}$ lautet:

$x_1^{(1)}$	$x_2^{(1)}$	$x_1^{(2)}$	$x_2^{(2)}$	$x^{(3)}$
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

wobei die logische Verknüpfung gegeben ist durch

$$x^{(3)} = x_1^{(1)} \veebar x_2^{(1)} = \neg x_1^{(2)} \wedge x_2^{(2)} = \left(x_1^{(1)} \barwedge x_2^{(1)}\right) \wedge \left(x_1^{(1)} \vee x_2^{(1)}\right)$$

mit

$$x_1^{(2)} = x_1^{(1)} \wedge x_2^{(1)} \quad \text{und} \quad x_2^{(2)} = x_1^{(1)} \vee x_2^{(1)}.$$

Mit den Eingangsgewichten zu den verdeckten Knoten $\mathbf{w}_i^{(1)} = (w_{i0}^{(1)}, w_{i1}^{(1)}, w_{i2}^{(1)})$ für $i = 1, 2$ und den Ausgangsgewichten $\mathbf{w}^{(2)} = (w_0^{(2)}, w_1^{(2)}, w_2^{(2)})$ wird durch die Vorschrift

$$\begin{aligned} x_1^{(2)} &= g\left(w_{10}^{(1)} x_0^{(1)} + w_{11}^{(1)} x_1^{(1)} + w_{12}^{(1)} x_2^{(1)}\right), \\ x_2^{(2)} &= g\left(w_{20}^{(1)} x_0^{(1)} + w_{21}^{(1)} x_1^{(1)} + w_{22}^{(1)} x_2^{(1)}\right), \\ x^{(3)} &= g\left(w_0^{(2)} x_0^{(2)} + w_1^{(2)} x_1^{(2)} + w_2^{(2)} x_2^{(2)}\right) \end{aligned}$$

jedem Vektor $\mathbf{x}^{(1)} = \left(x_0^{(1)}, x_1^{(1)}, x_2^{(1)}\right)$ ein Mustervektor $\mathbf{X}^{(1)} = \left(x_1^{(1)}, x_2^{(1)}\right)$ mit den Gewichten $\mathbf{W}_i^{(1)} = (w_{i1}^{(1)}, w_{i2}^{(1)})$ so zugeordnet, daß gilt:

$$\mathbf{W}_i^{(1)} \mathbf{X}^{(1)} = -w_{i0}^{(1)} x_0^{(1)}$$

bzw.

$$\mathbf{w}_i^{(1)} \cdot \mathbf{x}^{(1)} = \sum_{j=0}^{2} w_{ij}^{(1)} x_j^{(1)} = 0.$$

Die Schwellen werden darin durch die konstante Eingabe $x_0^{(1)} = 1$ und die Gewichte $w_{i0}^{(1)}$ berücksichtigt.

Analog wird jedem Vektor $\mathbf{x}^{(2)} = \left(x_0^{(2)}, x_1^{(2)}, x_2^{(2)}\right)$ ein Mustervektor $\mathbf{X}^{(2)} = \left(x_1^{(2)}, x_2^{(2)}\right)$ mit den Gewichten $\mathbf{W}^{(2)} = (w_1^{(2)}, w_2^{(2)})$ zugeordnet, derart daß gilt:

$$\mathbf{W}^{(2)}\mathbf{X}^{(2)} = -w_0^{(2)}x_0^{(2)}$$

bzw.

$$\mathbf{w}^{(2)} \cdot \mathbf{x}^{(2)} = \sum_{j=0}^{2} w_j^{(2)}x_j^{(2)} = 0.$$

Die Schwellen werden darin durch die konstante Eingabe $x_0^{(2)} = 1$ und das Gewicht $w_0^{(2)}$ bedient.

Wie man sich leicht überzeugen kann, lösen die Gewichte

$$\begin{aligned}
&\text{UND:} \quad \left(w_{10}^{(1)}, w_{11}^{(1)}, w_{12}^{(1)}\right) = (-1.5, 1.0, 1.0),\\
&\text{ODER:} \quad \left(w_{20}^{(1)}, w_{21}^{(1)}, w_{22}^{(1)}\right) = (-0.5, 1.0, 1.0)\\
&\text{XOR:} \quad \left(w_0^{(2)}, w_1^{(2)}, w_2^{(2)}\right) = (-0.5, -1.0, 1.0)
\end{aligned}$$

die Aufgabenstellung vollständig, denn an der Ebene der verdeckten Knoten gilt

$x_1^{(1)}$	$x_2^{(1)}$	$w_{10}^{(1)}x_0^{(1)} + w_{11}^{(1)}x_1^{(1)} + w_{12}^{(1)}x_2^{(1)}$	$x_1^{(2)}$	$w_{20}^{(1)}x_0^{(1)} + w_{21}^{(1)}x_1^{(1)} + w_{22}^{(1)}x_2^{(1)}$	$x_2^{(2)}$
0	0	-1.5	0	-0.5	0
1	0	-0.5	0	0.5	1
0	1	-0.5	0	0.5	1
1	1	0.5	1	1.5	1

und an der Ebene des Ausgangsknotens haben wir

$x_1^{(2)}$	$x_2^{(2)}$	$w_0^{(2)}x_0^{(2)} + w_1^{(2)}x_1^{(2)} + w_2^{(2)}x_2^{(2)}$	$x^{(3)}$
0	0	-0.5	0
0	1	0.5	1
0	1	0.5	1
1	1	-0.5	0

Nach unserer Vektorgleichung wird der Musterraum $\mathbf{X}^{(1)} = \left(x_1^{(1)}, x_2^{(1)}\right)$ in zwei Klassen unterteilt, die der Bedingung

$$\mathbf{w}_i^{(1)} \cdot \mathbf{x}^{(1)} < 0 \quad \text{bzw.} \quad \mathbf{w}_i^{(1)} \cdot \mathbf{x}^{(1)} > 0$$

genügen. Die Gleichung $\mathbf{w}_i^{(1)} \cdot \mathbf{x}^{(1)} = 0$ trennt diese beiden Klassen und entspricht einer Hyperebene im Musterraum, zu der die Vektoren $\mathbf{w}_i^{(1)}$ jeweils orthogonal sind. Für beide Verknüpfungen sind die Hyperebenen Geraden, die durch die folgenden Gleichungen beschrieben werden:

$$\begin{aligned}
&\text{UND:} \quad x_1 + x_2 = 1.5,\\
&\text{ODER:} \quad x_1 + x_2 = 0.5.
\end{aligned}$$

Verwendet man die Heavyside-Funktion

$$\Theta(x)=\begin{cases}1 & \text{für} \quad x\geq 0,\\ 0 & \text{für} \quad x<0,\end{cases}$$

kann eine scharfe Trennung in die beiden Klassen $x^{(3)}=1$ und $x^{(3)}=0$ vorgenommen werden. Legt man wie im nachfolgenden MATLAB-Beispielprogramm eine Sigmoidfunktion zugrunde, ist die Ausgangsaktivität $x^{(3)}$ ein Maß für den noch nicht in Sättigung befindlichen Abstand von der Hyperebene.

Anhang

```
% Programm XOR-Schaltung
clear all

% XOR-Input für x1 und x2
input = [0 0; 0 1; 1 0; 1 1];
% Erwünschtes XOR-Ergebnis
output = [0;1;1;0];
% Offset-Initialisierung
bias = [-1 -1 -1];
% Lernkoeffizient
coeff = 0.7;
% Zahl der Lernschritte
iterations = 100000;
% Gewichtsberechnung mittels "seed"
rand('state',sum(100*clock));
weights = -1 +2.*rand(3,3);

for i = 1:iterations
   out = zeros(4,1);
   numIn = length (input(:,1));
   for j = 1:numIn
      % Verdeckte Schicht
      H1 = bias(1,1)*weights(1,1) + input(j,1)*weights(1,2) + in-
put(j,2)*weights(1,3);

      % Datentransfer durch die Sigmoidfunktion 1/(1+e^-x)
      x2(1) = 1/(1+exp(-H1));
      H2 = bias(1,2)*weights(2,1) + input(j,1)*weights(2,2) + in-
put(j,2)*weights(2,3);
      x2(2) = 1/(1+exp(-H2));

      % Ausgangsschicht
      x3_1 = bias(1,3)*weights(3,1) + x2(1)*weights(3,2) +
x2(2)*weights(3,3);
      out(j) = 1/(1+exp(-x3_1));

      % Stimmt Delta-Gewichtswerte für die Ausgangsschicht ab:
      % delta(wi) = xi*delta,
      % delta = (1-actual output)*(desired output - actual output)
      delta3_1 = out(j)*(1-out(j))*(output(j)-out(j));

      % Delta-Rückentwicklung in die verdeckten Schichten
      delta2_1 = x2(1)*(1-x2(1))*weights(3,2)*delta3_1;
      delta2_2 = x2(2)*(1-x2(2))*weights(3,3)*delta3_1;
```



```
      % Gewichtsänderungssummationen der ursprünglichen Gewichte
      % zur Verwendung neuer Gewichte, um den Prozeß zu wiederholen
      % delta weight = coeff*x*delta
      for k = 1:3
         if k == 1 % Offset-Fälle
            weights(1,k) = weights(1,k) + coeff*bias(1,1)*delta2_1;
            weights(2,k) = weights(2,k) + coeff*bias(1,2)*delta2_2;
            weights(3,k) = weights(3,k) + coeff*bias(1,3)*delta3_1;
         else % Wenn k=2 oder 3, dann folgende Neuronen-Inputs
            weights(1,k) = weights(1,k) + coeff*input(j,1)*delta2_1;
            weights(2,k) = weights(2,k) + coeff*input(j,2)*delta2_2;
            weights(3,k) = weights(3,k) + coeff*x2(k-1)*delta3_1;
         end
      end
   end
end

out
weights

>> xorschaltung

out =
    0.0039
    0.9961
    0.9960
    0.0049

weights =
   -8.3914   -5.6408   -5.5801
   -3.0168   -7.1320   -7.0584
    6.0144   12.4737  -12.5958
```

