Aufgabe: Erläutern Sie anhand eines Beispiels, wie Sie ein neuronales Netz auslegen.

Lösung: Ein neuronales Netz besitzt eine sogenannte Black box, um die wir uns nicht eingehender zu kümmern brauchen. In dieser befinden sich die Algorithmen, die wir an sich nicht ändern brauchen, wenn sie einmal vorliegen. Im Anhang ist der MATLAB-Quellcode für einen solchen Algorithmus angegeben. Es gibt aber trotzdem Dinge, die uns das Programm nicht abnimmt, das ist die Zahl der Eingabe- und Ausgabeneuronen festzulegen und sich geeignete Testmuster auszudenken. Ferner kann man ein Abbruchkriterium für die Iterationen festlegen, wenn sich am Ergebnis signifikant nichts mehr ändert. In der Regel wird man dafür den Fehler des Verfahrens verwenden. Man sieht sehr schnell, wie vieler Iterationen es bedarf und nach wie vielen der Meßfehler kaum noch besser wird.

Im ersten Schritt legen wir die Zahl $i_{\max}$ des Ausgangneuronen fest. Wenn jedes Neuron einen anderen Vorgang auslösen soll, brauchen wir genau so viele Ausgangsneuronen wie Vorgänge. Diese Vorgänge können z.B. Klassifikationen sein. Man hat jedoch in einem binären Muster $2^{i_{\max}}$ Unterscheidungsmöglichkeiten, wenn man die Zahl der Ausgangsneuronen beschränken möchte.

Die nächstwichtige Frage ist die Zahl $p_{\max}$ der Trainingsmuster, deren es bedarf, um das System auf eine gewisse Anzahl von Mustervarianten anzulernen. So sind etwa $j_{\max}$ Eingangsneuronen erforderlich, um damit $p_{\max} = 2^{j_{\max}}$ binäre Muster trainieren zu können. Für 3 Trainingsmuster benötigt man also mindestens 2 Eingangsneuronen.

Es bleibt uns leider nicht erspart, für jeden Input-Vektor und jedes der $p_{\max}$ Trainingsmuster die einzelnen $p_{\max} \times j_{\max}$ Komponenten x_{pj} festzulegen.

Gleichermaßen müssen wir auch alle $p_{\max} \times i_{\max}$ Komponenten η_{pi} des Soll-Ausgabevektors manuell festlegen. Das ist aber eher die einfachere Aufgabe, weil hierzu keinerlei Überlegungen zu den Inhalten angestellt werden müssen, außer daß die Zählweise festgelegt werden muß.

Das eigentliche Problem bei neuronalen Netzen ist, geeignete Kenngrößen oder Merkmale für die Eingangsneuronen zu finden, von denen die Optimierung abhängig ist. Das menschliche Gehirn trifft über diese Kenngrößen lediglich Abschätzungen, weil dem Menschen eine schnelle Berechnung im Kopf nicht gegeben ist.[1] Trotzdem ist das neuronale Netz in der Lage, rein aufgrund dieser Schätzgrößen und aufgrund von in Erinnerung gerufenen, trainierten Erfolgen die richtige Reaktion auszulösen. Auf Schätzgrößen greift man immer erst zurück, wenn man nichts Besseres hat. Man kann natürlich auch konkrete Meßergebnisse in neuronale Netze einspeisen und mit Schätzgrößen, für die man keine exakten Werte besitzt, verknüpfen.

Die Gewichtung ist nicht das Problem. In unserem Beispiel belegen wir alle Gewichte mit dem Anfangswert eins, weil das Netz die korrekten Gewichte von sich aus findet.

[1] Die Natur an sich kennt keine Mathematik.

Als Abbruchkriterium für die Iterationen haben wir $k_{\max} = 10$ gewählt, womit der Anfangsfehler von 8,3819 auf 0,719 absinkt, also um mehr als einen Faktor 10. Viel besser wird man auch nach 100 Iterationen nicht, so daß es keinen großen Nährwert hat, wertvolle Rechenzeit zu vergeuden.

Mit dem sogenannten Lernparameter muß man spielen, um seinen Einfluß kennenzulernen. Wir haben ihn auf einen Zahlenwert von 5 festgelegt.

Nach dem Training werden die Gewichte „eingefroren“. Tauchen nun während der Anwendung des neuronalen Netzes an konkreten Szenarien bekannte Muster als Eingangsgrößen auf, werden diese anhand des Erlernten klassifiziert. Stimmen sie mit einem der Testmuster weitgehend überein, so ist höchstwahrscheinlich auch das reale Ergebnis im Einklang mit der geschätzten Art des Ziels.

In unserem Beispiel wird etwa das zweite Muster gar nicht erkannt. Das haben wir unten im Ausdruck in Rot markiert. Der Grund ist auch klar: Ein Muster, das sich in nichts von anderen unterscheidet und sozusagen alles und nichts ist, also nur aus Einsen besteht, kann alles sein. Das ist kein Fehler des Programms.

Die erkannten Werte liegen alle über 0,8. Legen wir also eine Aktivierungsschwelle über einen Wert von 0,8, so würden die entsprechenden Neuronen feuern.

Quellcode

```
% Programm Neuronale Netze

% Fehlerberechnung für ein zweistufiges neuronales Netz ohne verdeckte
% Schicht mit beliebig vielen Ein- und Ausgabeneuronen und einer
% Sigmoidkurve als Aktivierungsfunktion
clear all

% Der Lernparameter sei lambda
lambda = 5;

% p ist die Laufvariable des jeweiligen Trainingsmusters
% pmax ist die maximale Zahl vorliegender Trainingsmuster
pmax = 5;

% Eingangs-, Soll-Ausgangs- und Ist-Ausgangsvektor hängen jeweils von der
% Laufvariablen p des Musters ab.

% Der Soll-Ausgangsvektor eta(p,i) hat imax Komponenten, wobei i die
% Zeilennummer der Gewichtsmatrix angibt
imax = 5;

% Die Spaltennummer der Gewichtsmatrix sei j
jmax = 5;

% Die 5 Komponenten des Soll-Ausgangsvektors für die 5 Muster für den
% Trainingslauf sind gegeben durch
eta(1,1)=1; eta(1,2)=0; eta(1,3)=0; eta(1,4)=0; eta(1,5)=0;
eta(2,1)=0; eta(2,2)=1; eta(2,3)=0; eta(2,4)=0; eta(2,5)=0;
eta(3,1)=0; eta(3,2)=0; eta(3,3)=1; eta(3,4)=0; eta(3,5)=0;
```



```
eta(4,1)=0; eta(4,2)=0; eta(4,3)=0; eta(4,4)=1; eta(4,5)=0;
eta(5,1)=0; eta(5,2)=0; eta(5,3)=0; eta(5,4)=0; eta(5,5)=1;

% Die 5 Komponenten der 5 Trainingsmuster x(p,j) des Eingangsvektors sind
% gegeben durch
x(1,1) = 1; x(1,2) = 0; x(1,3) = 1; x(1,4) = 0; x(1,5) = 1;
x(2,1) = 1; x(2,2) = 1; x(2,3) = 1; x(2,4) = 1; x(2,5) = 1;
x(3,1) = 0; x(3,2) = 1; x(3,3) = 0; x(3,4) = 1; x(3,5) = 0;
x(4,1) = 1; x(4,2) = 1; x(4,3) = 0; x(4,4) = 1; x(4,5) = 1;
x(5,1) = 0; x(5,2) = 0; x(5,3) = 1; x(5,4) = 0; x(5,5) = 0;

% Die Laufvariable der kmax Iterationsschritte sei k
kmax = 10;

% Für den Trainingslauf verwenden wir als Startgewichte eine Matrix
% aus lauter Einsen:
for i=1:imax
    for j=1:jmax
        w(i,j,1) = 1;
    end
end

for k=1:kmax
    E = 0;
    for i=1:imax
        % Berechnung der Aktivierungsfunktion
        for p=1:pmax
            ypsilon = 0;
            for j=1:jmax
                y(p,i,k) = ypsilon + w(i,j,k)*x(p,j);
                ypsilon = y(p,i,k);
            end
            f(p,i,k)=1/(1+exp(-y(p,i,k)));
        end
        % Änderung des Verbindungsgewichts der iten Zeile und jten Spalte
        for j=1:jmax
            Deltawij = 0;
            for p=1:pmax
                Deltaw(i,j,k) = Deltawij + lambda*(eta(p,i)-
f(p,i,k))*f(p,i,k)*(1-f(p,i,k))*x(p,j);
                Deltawij = Deltaw(i,j,k);
            end
            w(i,j,k+1) = w(i,j,k) + Deltaw(i,j,k);
        end
    end
end

% Fehlerrechnung
for k=1:kmax
    for i=1:imax
        for p=1:pmax
            EM(p,k) = 0.5*(eta(p,i)-f(p,i,k))^2;
        end
        EM0 = 0;
        for p=1:pmax
            Error(i,k) = EM0 + EM(p,k);
            EM0 = Error(i,k);
        end
    end
    E0 = 0;
```



```
    for i=1:imax
        E(k) = E0 + Error(i,k);
        E0 = E(k);
    end
end

% Ausdruck
for p=1:pmax
    for i=1:imax
        disp(['y(',num2str(p),',',num2str(i),') = ' num2str(y(p,i,kmax))])
    end
end
disp(' ');
for i=1:imax
    for j=1:jmax
        disp(['w(',num2str(i),',',num2str(j),') = ' num2str(w(i,j,kmax))])
    end
end
disp(' ');
for i=1:imax
    for j=1:jmax
        disp(['Deltaw(',num2str(i),',',num2str(j),') = ' num2str(Del-
taw(i,j,kmax))])
    end
end
disp(' ');
for p=1:pmax
    for i=1:imax
        disp(['f(',num2str(p),',',num2str(i),') = ' num2str(f(p,i,kmax))])
    end
end
disp(' ');
disp(['E = ' num2str(E)])
```

```
>> neuronalenetze
y(1,1) = 1.4081
y(1,2) = -2.1119
y(1,3) = -6.0603
y(1,4) = -1.192
y(1,5) = -1.9282
y(2,1) = -3.0353
y(2,2) = -3.9548
y(2,3) = -4.3126
y(2,4) = -1.9573
y(2,5) = -4.5667
y(3,1) = -4.4435
y(3,2) = -1.8429
y(3,3) = 1.7477
y(3,4) = -0.76539
y(3,5) = -2.6385
y(4,1) = -1.8676
y(4,2) = -2.2423
y(4,3) = -2.0176
y(4,4) = 1.6811
y(4,5) = -6.0095
y(5,1) = -1.1678
y(5,2) = -1.7125
```



```
y(5,3) = -2.295
y(5,4) = -3.6384
y(5,5) = 1.4428

w(1,1) = 1.288
w(1,2) = -2.2217
w(1,3) = -1.1678
w(1,4) = -2.2217
w(1,5) = 1.288
w(2,1) = -0.19971
w(2,2) = -0.92143
w(2,3) = -1.7125
w(2,4) = -0.92143
w(2,5) = -0.19971
w(3,1) = -1.8826
w(3,2) = 0.87383
w(3,3) = -2.295
w(3,4) = 0.87383
w(3,5) = -1.8826
w(4,1) = 1.2232
w(4,2) = -0.3827
w(4,3) = -3.6384
w(4,4) = -0.3827
w(4,5) = 1.2232
w(5,1) = -1.6855
w(5,2) = -1.3193
w(5,3) = 1.4428
w(5,4) = -1.3193
w(5,5) = -1.6855

Deltaw(1,1) = 0.067569
Deltaw(1,2) = -0.088261
Deltaw(1,3) = -0.069542
Deltaw(1,4) = -0.088261
Deltaw(1,5) = 0.067569
Deltaw(2,1) = -0.0031309
Deltaw(2,2) = -0.031838
Deltaw(2,3) = -0.060404
Deltaw(2,4) = -0.031838
Deltaw(2,5) = -0.0031309
Deltaw(3,1) = -0.061679
Deltaw(3,2) = 0.032052
Deltaw(3,3) = -0.038951
Deltaw(3,4) = 0.032052
Deltaw(3,5) = -0.061679
Deltaw(4,1) = -0.17132
Deltaw(4,2) = -0.30722
Deltaw(4,3) = -0.27836
Deltaw(4,4) = -0.30722
Deltaw(4,5) = -0.17132
Deltaw(5,1) = -0.070907
Deltaw(5,2) = -0.021314
Deltaw(5,3) = 0.076844
```



```
Deltaw(5,4) = -0.021314
Deltaw(5,5) = -0.070907

f(1,1) = 0.80347
f(1,2) = 0.10794
f(1,3) = 0.0023283
f(1,4) = 0.23291
f(1,5) = 0.12695
f(2,1) = 0.045855
f(2,2) = 0.018802
f(2,3) = 0.013221
f(2,4) = 0.12375
f(2,5) = 0.010285
f(3,1) = 0.011618
f(3,2) = 0.13671
f(3,3) = 0.85166
f(3,4) = 0.31748
f(3,5) = 0.066699
f(4,1) = 0.13383
f(4,2) = 0.096018
f(4,3) = 0.11737
f(4,4) = 0.84305
f(4,5) = 0.0024493
f(5,1) = 0.23725
f(5,2) = 0.15284
f(5,3) = 0.091539
f(5,4) = 0.02562
f(5,5) = 0.80889

E = 8.3819      6.5336       1.665      1.2263      1.0042      0.94229
0.97871     0.77392     0.74416     0.71896
```

