Aufgabe: Erläutern Sie das Prinzip einer selbstorganisierenden Karte anhand eines zweidimensionalen Kohonen-Netzes mit eindimensionaler Eingabeschicht (Kette) und wählen Sie als Nachbarschaftsfunktion eine normalisierte Gaußverteilung.

Lösung: Selbstorganisierende Karten sind einschichtige neuronale Netzwerke, die mit unüberwachten Lernverfahren trainiert werden. Es gibt also keine erwünschten Ausgabemuster. Betrachten wir zur Erläuterung des Prinzips Abb. 1.[1] Jedes Neuron der Eingabeschicht $(n=2)$ ist mit jedem Neuron der (9×9)-Kohonenschicht[2] $(m=2)$ verbunden. Ein „elastisches“ Gitternetz aus 81 Gewichtsvektoren vermittelt nach Art einer nichtlinearen Regression eine topologieerhaltende Abbildung *n*-dimensionaler Eingabevektoren auf das *m*-dimensionale Neuronengitter. Bei den Neuronen, die neben dem Erregungszentrum ihre Gewichtsvektoren anpassen dürfen, handelt sich um solche, deren Entfernung auf der Karte nicht größer ist als ein zeitabhängiger Schwellenwert, der als Entfernungsreichweite bezeichnet wird.

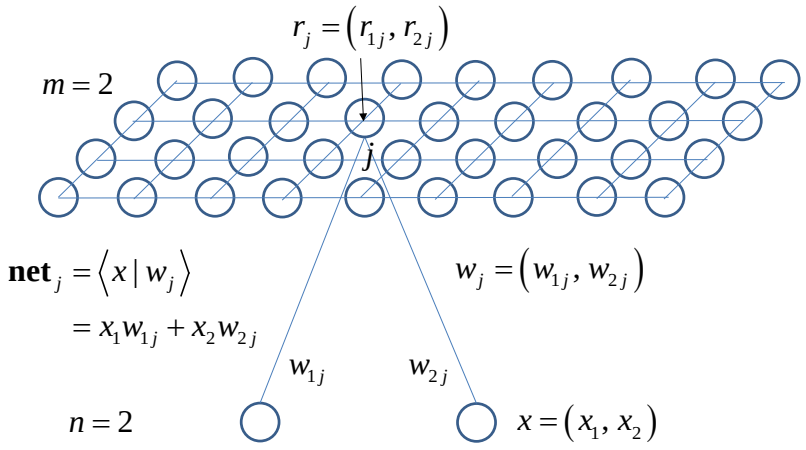

Abbildung 1. Netzstruktur eines zweidimensionalen Kohonen-Netzes mit eindimensionaler Eingabeschicht

Beim Lernverfahren wird ein *n*-dimensionaler Eingabevektor $x=(x_1, x_2, \ldots, x_n)$ parallel mit allen Gewichtsvektoren $w_j=(w_{1j}, w_{2j}, \ldots, w_{nj})$ in einer euklidischen Norm verglichen. Das Neuron *s*, dessen Gewichtsvektor w_s am ähnlichsten zum Eingabevektor *x* ist, gewinnt:

$$\|x-w_s\| = \min_j \left(\|x-w_j\|\right).$$

Unter Verwendung der Funktion arg, die den Index der Minimumsuche liefert,

$$s = \arg\min_j \left(\|x-w_j\|\right) = \arg\min_j \left(\sqrt{\sum_{i=1}^{n}\left(x_i - w_{ij}\right)^2}\right),$$

[1] Im Fall eines eindimensionalen Netzes mit zweidimensionalen Gewichtsvektoren läßt sich das graphisch gut veranschaulichen.

[2] Benannt nach dem finnischen Ingenieur Teuvo Kalevi Kohonen (* 11.07.1934 in Lauritsala)

gewinnt das Neuron mit dem größten Skalarprodukt:

$$\left\langle x \mid w_j \right\rangle = \sum_{i=1}^{n} x_i w_{ij} = \text{net}_j .$$

Jedes der *j* Neuronen entspricht also einer Netzeingabe. Während des Lernens ändern sich die Gewichtsvektoren aller Neuronen des Neuronengitters in der Nachbarschaft des Neurons *s* um

$$\Delta w_j = \lambda(t) h_{sj}(t)\left(x - w_j\right),$$

d.h.

$$w_j(t+1) = w_j(t) + \lambda(t) h_{sj}(t)\left(x(t) - w_j(t)\right).$$

Dabei ist die zeitabhängige Lernrate

$$\lambda(t) = \lambda(0) \cdot \left(\frac{\lambda(t_{\max})}{\lambda(0)} \right)^{\frac{t}{t_{\max}}}$$

eine monoton fallende Funktion mit $0 < \lambda(t) < 1$. Um den Lernfortschritt der Karte im Laufe des Trainings zu verringern, wird sie nach jeder Epoche *t* verkleinert. Die Größe

$$h_{sj}(t) = h\left(\left\| r_s - r_j \right\|, t\right) = \exp\left(-\frac{\left\| r_s - r_j \right\|^2}{d^2(t)} \right)$$

ist die sogenannte Nachbarschaftsfunktion, wobei r_s und r_j die Ortsvektoren des Neurons *s* und des Neurons *j* im Neuronengitter angeben. Die monoton fallende Distanzfunktion

$$d(t) = d(0) \cdot \left(\frac{d(t_{\max})}{d(0)} \right)^{\frac{t}{t_{\max}}}$$

entspricht dabei der Varianz einer Gaußfunktion. Mit dem Adaptionsradius $d(0)$ zum Start des Verfahrens und $d(t_{\max})$ als dem Adaptionsradius am Ende des Verfahrens, definiert durch $t_{\max}$, werden die Ergebnisse der Radius- und Distanzfunktion der Nachbarschaftsfunktion als Parameter übergeben:

$$h\left(\left\| r_s - r_j \right\|, t\right) = h(z, d) = \exp\left(-\frac{z^2}{d^2} \right),$$

wobei die Differenz *z* der Ortvektoren gegeben ist durch

$$z \equiv \left\| r_s - r_j \right\| = \sqrt{\sum_{i=1}^{n} \left(r_{s,i} - r_{j,i} \right)^2}.$$

Jedes Neuron im zweidimensionalen Nachbarschaftsgitter wird dabei durch eines der folgenden Koordinatenpaare beschrieben:

$$\begin{aligned}
r_{j,1} &= -2 + \frac{j-1}{8}4, & r_{j,2} &= -2 && \text{für} \quad j = 1, \ldots, 9, \\
r_{j+9,1} &= -2 + \frac{j-1}{8}4, & r_{j+9,2} &= -2 + \frac{1}{8}4 && \text{für} \quad j = 1, \ldots, 9, \\
&\vdots & &\vdots && \vdots \quad \vdots \\
r_{j+64,1} &= -2 + \frac{j-1}{8}4, & r_{j+64,2} &= -2 + \frac{9}{8}4 && \text{für} \quad j = 1, \ldots, 9, \\
r_{j+72,1} &= -2 + \frac{j-1}{8}4, & r_{j+172,2} &= 2 && \text{für} \quad j = 1, \ldots, 9.
\end{aligned}$$

Wurden sämtliche Eingabemuster der Karte einmal präsentiert, beginnt die neue Epoche, in der erneut alle Muster eingegeben werden. In Abb. 2 sind ausgehend von willkürlichen Zufallswerten 81 Gitterpunkte einer Kohonenkarte dargestellt, die alle die gleiche Tendenz verfolgen. Die Gewichtsvektoren mit der größten Ähnlichkeit zum Eingabevektor werden zum Eingabevektor hingezogen, der hier in Bildmitte liegt. Vektoren außerhalb des Nachbarschaftsradius werden nicht adaptiert.

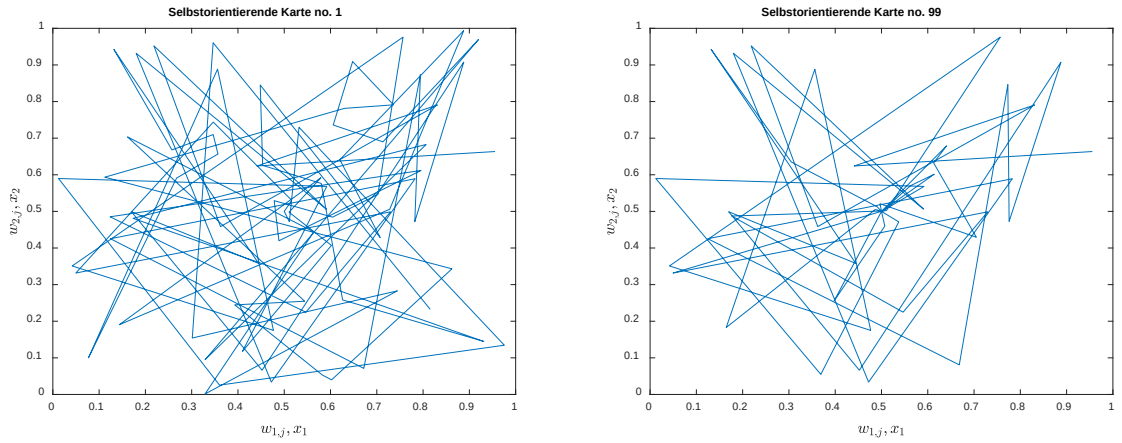

Abbildung 2. Um den Eingabevektor (0.5, 0.5) verteilte Zufallsgewichte nach 1 und 100 Epochen

Es hat den Anschein, als habe nach Durchlaufen von 100 Epochen die Dichte der Gitterpunkte abgenommen, man kann aber dennoch eindeutig identifizieren, welche noch da sind und welche nicht. Die Länge der die Punkte verbindenden Trajektorien hat eindeutig abgenommen. Deutet man die Trajektorien als Weglängen eines Teilchens, welches der Brownschen Molekularbewegung unterliegt, so hätte die mittlere freie Weglänge, also die Zeit zwischen zwei Stößen, eindeutig zugenommen. Die „Sogwirkung“ des Zentrums erfaßt aber nicht alle Gitterpunkte, da die Distanzfunktion irgendwann auf den Wert Null abgesunken ist und damit auch der Nachbarschaftsradius gegen Null geht, so daß sich außerhalb von diesem nichts mehr ändert.

Man kann z.B. anhand der Epochenfolge in Abb. 3 klar erkennen, wie aus einer ungeordneten[3] Normalverteilung im Laufe vieler Epochen Ordnung entsteht. Zum Teil sind die Unterschiede zwischen Karte 499 und 999 beträchtlich, doch irgendwann werden die Verhältnisse stationär, und dann führt auch jede zusätzliche Iteration zu keiner weiteren Verbesserung mehr.

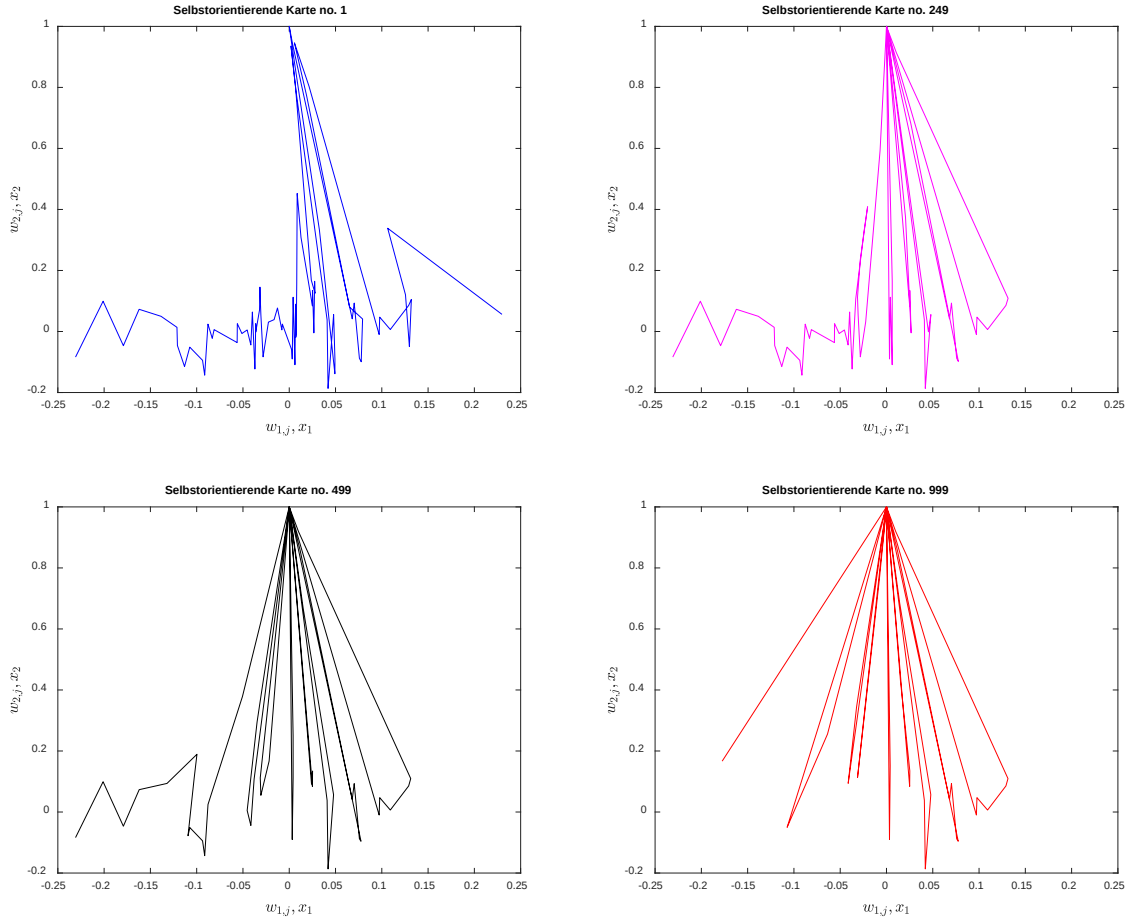

Abbildung 3. 4 Beispiele aus den ersten 1000 Epochen einer SOM um den Eingabevektor (0, 1) der Eingabeschicht

Das Lernverfahren der selbstorganisierenden Karten kann sehr gut zur Visualisierung und zum Auffinden von Ähnlichkeitsbeziehungen in einem hochdimensionalen Eingaberaum verwendet werden. Selbstorganisierende Karten sind eine technische Umsetzung des Prinzips der lateralen Hemmung in der Neurobiologie, indem etwa eine aktive Nervenzelle die Aktivität der benachbarten Zellen hemmt. Ein Ergebnis der lateralen Hemmung ist beispielsweise die Kontrastverstärkung bei der Herausbildung von Gestaltgrenzen zur räumlichen Orientierung.

Anhang

```
% Programm SOM
% Selbstorganisierende Karte
clear all
% Intervallgrenzen der Karte
a = -2;
```

[3] verrauschten


```
b = 2;
% Matrixgröße p in einer Dimension
p = 9;
q=1;
% Radiusfunktion
for n = 1:9
    for j = 1:9
        r(j+(n-1)*p,1) = a + (j-1)/(p-1)*(b-a);
        r(j+(n-1)*p,2) = a + (q-1)/(p-1)*(b-a);
        X    =    ['r(',num2str(j+(n-1)*p),',1)    =    ',num2str(r(j,1)),';
r(',num2str(j+(n-1)*p),',2) = ',num2str(r(j+(n-1)*p,2))];
        disp(X)
    end
    disp(' ');
    q=q+1;
end
disp(' ');
% Initialisierung der Verbindungsgewichte
% Zufallszahlen einer Gaußverteilung mit Mittelwert und Standardabweichung
for j = 1:p^2
    w1 = random('Normal',0,0.6,1,81);
    w2 = random('Normal',-0.5,0.6,1,81);
    X = ['w1(',num2str(j),') = ',num2str(w1(j)),'    w2(',num2str(j),') =
',num2str(w2(j))];
    disp(X)
end
disp(' ');
w1 = sort(w1);
for j = 1:p^2
    X = ['w1(',num2str(j),') = ',num2str(w1(j)),'    w2(',num2str(j),') =
',num2str(w2(j))];
    disp(X)
end
% Neuronen der Eingabeschicht
x1 = 0;
x2 = 1.5;
% Initialisierung der Epoche
t = 1;
% Initialisierung der Distanzfunktion
d(1) = 1;
% Initialisierung der Lernratenfunktion
lambda(1) = 1;
% Zahl der zu durchlaufenden Epochen
tmax = 1000;
% Maximalwert der Distanzfunktion
d(tmax) = d(1)/10;
% Maximalwert der Lernratenfunktion
lambda(tmax) = lambda(1)/10;
while t < tmax/100
    % Bestimmung des Siegerneurons
    s = 1;
    min(1) = sqrt((x1-w1(1))^2+(x2-w2(1))^2);
    for j = 1:p^2
        rms(j) = sqrt((x1-w1(j))^2+(x2-w2(j))^2);
        X = ['rms(',num2str(j),') = ',num2str(rms(j))];
        disp(X)
        if rms(j) < min(s)
            s = j;
            min(s) = rms(j);
        end
    end
```



```
    disp(' ');
    X = ['min(',num2str(s),') = ',num2str(min(s))];
    disp(X)
    disp(' ');
    for j = 1:p^2
        % Bestimmung der Differenzen z(j) für das Siegerneuron
        z(j) = ((r(s,1)-r(j,1))^2 + (r(s,2)-r(j,2))^2);
        % Distanzfunktion
        d(t) = d(1)*(d(tmax)/d(1))^((t-1)/tmax);
        X = ['z(',num2str(j),') = ',num2str(z(j)),'   d(',num2str(t),') =
',num2str(d(t))];
        disp(X)
        % Nachbarschaftsfunktion h(j)
        h(j) = exp(-z(j)^2/d(t)^2);
        % Bestimmung der Gewichte
        w1(j) = w1(j) + lambda(t)*h(j)*(x1-w1(j));
        w2(j) = w2(j) + lambda(t)*h(j)*(x2-w2(j));
        X = ['h(',num2str(j),') = ',num2str(h(j)),';     w1(',num2str(j),')
= ',num2str(w1(j)),';    w2(',num2str(j),') = ',num2str(w2(j))];
        disp(X)
    end
    disp(' ');
    X = ['t = ',num2str(t)];
    disp(X)
    figure(1)
    if t==1
        plot(w1,w2,'y')
    end
    if t==2
        plot(w1,w2,'m')
    end
    if t==3
        plot(w1,w2,'c')
    end
    if t==4
        plot(w1,w2,'r')
    end
    if t==5
        plot(w1,w2,'g')
    end
    if t==6
        plot(w1,w2,'k')
    end
    if t==7
        plot(w1,w2,'b')
    end
    hold on
    xlim([-1.5 1.5])
    ylim([-2 2])
    title('Selbstorientierende Karte no. 9')
    legend('t = 1','t = 2','t = 3','t = 4','t = 5','t = 6','t = 7');
    xlabel('$w_{1,j},x_{1}$','interpreter','latex')
    ylabel('$w_{2,j},x_{2}$','interpreter','latex')
    disp(' ');
    t = t+1;
    % Lernratenfunktion
    lambda(t) = lambda(1)*(lambda(tmax)/lambda(1))^((t-1)/tmax);
end
```

