8. Self-Organizing Maps

In dieser Lerneinheit wollen wir uns zwei Typen von sogenannten selbstorganisierenden Netzen ansehen: Kohonenkarten und Neuronengas. Beide sind in der Lage, selbständig (d.h. ohne das Vorhandensein eines externen "Lehrers") bestimmte Strukturen mit recht interessanten Eigenschaften herauszubilden, die wir gleich noch kennenlernen werden.

8.1 Kohonenkarten

Kohonennetze, oder auch Kohonenkarten (nach ihrem Entwickler, Teuvo Kohonen), sind eine spezielle Art selbst-organisierender Netze. DIe Neuronen befinden sich in einer festen, meist gitterförmigen Anordnung (wie einem Quadrat oder einem Rechteck) zueinander. Prinzipiell sind aber auch kompliziertere Anordnungen wie etwa eine Kugeloberfläche, das innere eines Kubus oder ein Torus denkbar, werden jedoch seltener verwendet. Jedem Neuron ist dabei eine bestimmte Position in Form ganzzahliger Koordinaten innerhalb dieses Gitters zugeordnet, siehe Abbildung 8.1 links. Diese Position ist unveränderlich und wird am Anfang zusammen mit der Topologie (der dimensionalen Ausdehnung und dem Nachbarschaftsverhältnis) des Gitters festgelegt.

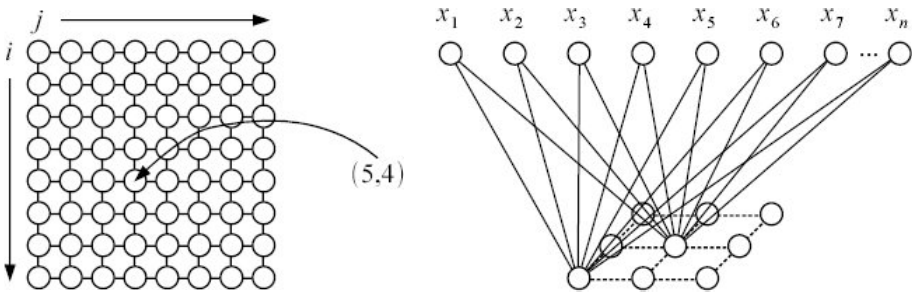

Abbildung 8.1 *Links:* Gitterartige Anordnung von Neuronen. Jedem Neuron sind Koordinaten zugewiesen, die dessen Position innerhalb des Gitters bestimmen. *Rechts:* Jedes Neuron des Gitters besitzt Verbindungen zur gesamten Eingabe des Netzes (schematisch für 2 Neuronen dargestellt). Jede dieser Verbindungen besitzt ein modifizierbares Gewicht. Die gestrichelten Verbindungen deuten die Nachbarschaftsbeziehungen der Neuronen an.

Die Verbindungen zwischen den einzelnen Neuronen des Gitters sind hier allerdings keine Verbindungen mit anpassbaren Gewichten, so wie wir das etwa beim Perzeptron gesehen haben. Man kann sie sich eher als eine Art "Definition der Nachbarschaft" vorstellen. Allerdings ist auch eine Interpretation möglich, bei der es sich um Neuronenverbindungen handelt, über die ein aktives Neuron alle Neuronen seiner Umgebung, mit denen es verbunden ist, in deren Aktivität beeinflussen kann.

Die Beeinflussung zwischen zwei Neuronen kann dabei entfernungsabhängig entweder die Aktivität anregend oder hemmend sein, ist aber in jedem Fall *nur auf eine bestimmte nähere Umgebung* eines Neurons beschränkt.

Diese Eigenschaft ist biologisch motiviert, da es zum Beispiel in der Netzhaut von Tieren (und der des Menschen) ähnliche Vorgänge gibt, bei der gleichmäßig angeordnete Neuronen mehrere auch weiter entfernte Nachbarn beeinflussen können und dadurch die Wahrnehmung von Kanten verstärken oder es uns überhaupt ermöglichen, Details in unterschiedlichen Helligkeitsbereichen wahrzunehmen. Abbildung 8.2 zeigt ein Beispiel dazu aus der Retina eines Pferdes.

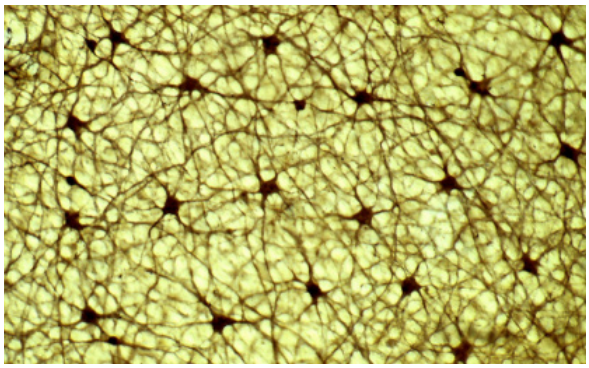

Abbildung 8.2 Lichtmikroskopische Aufnahme von Horizontalzellen eines Pferdes. Sie sind u.A. mit den Photorezeptoren (auf dem Foto nicht zu erkennen) der Netzhaut verbunden und bewirken, daß die Aktivität eines Photorezeptors die Aktivität benachbarter Photorezeptoren hemmen kann. Dieser Vorgang ist der Grund, warum wir bei unterschiedlich starken äußeren Helligkeitsbedingungen überhaupt einzelne Unterschiede in der Farb- und Helligkeitsverteilung wahrnehmen können.

Neben den Nachbarschaftsverbindungen innerhalb des Kohonennetzes sind alle Neuronen darüberhinaus noch mit anpassbaren Verbindungen zu einer Netzeingabe versehen, siehe Abbildung 8.1, rechts. Jede dieser Verbindungen besitzt ein modifizierbares Gewicht, so daß sich jedem Neuron ein Gewichtsvektor $\underline{w} \in \mathbb{R}^n$ zuordnen läßt.

8.2 Aktivierung

Wir haben zuvor Neuronen betrachtet, deren Ausgabe $y \in \mathbb{R}$ bei Eingabe eines $\underline{x} \in \mathbb{R}^n$ durch $y = \underline{w}^T \underline{x}$ gebildet wird. Je größer die Erregung, umso größer auch die Ausgabe des Neurons.

Bei den Neuronen des Kohonennetzes wird die Ausgabe aber etwas anders interpretiert. Wir sagen, daß ein Neuron des Kohonennetzes *aktiviert* wird oder *aktiv* ist, falls der euklidische Abstand zwischen seinem Gewichtsvektor $\underline{w}$ und dem Eingabevektor $\underline{x}$ minimal ist unter allen übrigen Neuronen. Dies soll dabei allerdings für *beliebige* (nicht notwendigerweise normierte) Gewichtsvektoren gelten.

8.3 Lernen

In einem Kohonennetz lernen die Neuronen in einem Verbund, d.h. wird an ein Kohonennetz ein Eingabevektor $\underline{x}$ angelegt, dann wird eine ganze Gruppe von Neuronen gleichzeitig an diesen einen Vektor angepaßt. Die Anpassung erfolgt dabei durch Veränderung der Gewichtsvektoren, so daß diese dem Vektor $\underline{x}$ nach der Gewichtsanpassung etwas mehr ähneln als vorher.
Welche Gewichte angepaßt werden, wird dabei durch dasjenige Neuron bestimmt, das durch den Eingabevektor

$\underline{x}$ aktiviert wurde. Es werden genau die Neuronen am Lernprozess beteiligt, die sich in einer Umgebung des aktivierten Neurons befinden. Die Umgebung wird dabei anhand der festen Nachbarschaftsverhältnisse bestimmt, die den Neuronen zu Anfang vorgegeben wurde.

Bezeichnen wir die Position (die Koordinaten innerhalb des Gitters) eines Neurons k in dieser Nachbarschaft als $\underline{p}_k$ und die Position des aktivierten Neurons (das Neuron, dessen Gewichtsvektor sich am nächsten an $\underline{x}$ befindet) mit $\underline{p}_*$, dann lautet die Lernregel für den Gewichtsvektor $\underline{w}_k$ des Neurons *k*:

$$\underline{w}_k \leftarrow \underline{w}_k + \eta(t) \cdot h(\underline{p}_k, \underline{p}_*, t) \cdot (\underline{x} - \underline{w}_k)$$

$\eta(t)$ ist dabei die Lernrate für diese Lernregel und $h(\cdot)$ eine Funktion (die sogenannte Nachbarschaftsfunktion), die angibt, wie stark Neuronen, die sich in einer bestimmten Entfernung zu dem Neuron befinden, das aktiviert wurde, am Lernprozess beteiligt werden. Dabei ist t ein Zeitparameter, der andeuten soll, daß sich η und $h(\cdot)$ zeitlich verändern. Eine typische Wahl für η und $h(\cdot)$ ist etwa:

$$\begin{aligned} h(\underline{p}_k, \underline{p}_*, t) &= \exp\left(-\frac{||\underline{p}_k - \underline{p}_*||_2^2}{\sigma(t)^2}\right) \\ \sigma(t) &= \sigma_0 \cdot \exp(-t/\tau_1) \\ \eta(t) &= \eta_0 \cdot \exp(-t/\tau_2) \\ t &= 0, 1, 2... \end{aligned}$$

$\sigma(t)$ bestimmt dabei den *Nachbarschaftsradius*, innerhalb dessen die Neuronen ihre Gewichte am stärksten an $\underline{x}$ anpassen dürfen. $\sigma_0 > 0$ gibt den Nachbarschaftsradius zu Beginn des Lernprozesses an und $\eta_0 < 1$ bezeichnet die anfängliche Lernrate. Die Konstanten $\tau_1 >> 0$ und $\tau_2 >> 0$ legen fest, wie schnell diese Werte im Laufe der Zeit abnehmen. Aktivität und Lernen sind zur Übersicht nochmals zusammenfasst:

Lernen im Kohonennetz

A. Finde für $\underline{x}$ das beste Neuron

1. Finde p^* mit $||\underline{x} - \underline{w}_p^*|| = \texttt{min}(||\underline{x} - \underline{w}_i||)$

B. Verbessere die Gewichte der Nachbarschaftsneuronen

2. $\sigma \leftarrow \sigma_0 \cdot \exp(-t/\tau_\sigma)$
3. $\eta \leftarrow \eta_0 \cdot \exp(-t/\tau_\eta)$
4. für $i = 1, ..., AnzahlNeuronen$
5. $\quad h_i \leftarrow \exp(-||\underline{p}^* - \underline{p}_i||_2^2/\sigma^2)$
6. $\quad \underline{w}_i \leftarrow \underline{w}_i + \eta \cdot h_i \cdot (\underline{x} - \underline{w}_i)$
7. end

Dabei ist t = *0, 1, 2, ..., T* ein Zeitindex, der die aktuelle Anzahl der bisher durchgeführten Iterationen bezeichnet. σ_0 ist der anfängliche Nachbarschaftsradius und η_0 die anfängliche Lernrate des Kohonennetzes in der ersten Iteration. τ_σ und τ_η sind Parameter, die angeben, wie schnell der Nachbarschaftsradius und die Lernrate im Laufe der Iterationen am Absinken
sind. Sie werden üblicherweise als $\tau_\sigma = -\frac{T}{\ln(\sigma_1/\sigma_0)}$ und $\tau_\eta = -\frac{T}{\ln(\eta_1/\eta_0)}$ gewählt, wobei σ_1 bzw. η_1 den Nachbarschaftsradius bzw. die Lernrate angibt, die in der letzten Iteration T angenommen werden soll. Dabei ist $\mathbf{w}_i$ der Gewichtsvektor des *i*-ten Neurons und $\mathbf{p}_i$ die zugehörige Gitterkoordinate von Neuron *i*. Mit $\mathbf{p}^*$ ist die Gitterkoordinate desjenigen Neurons bezeichnet, dessen Gewichtsvektor den geringsten Abstand zur Eingabe **x** hat. Man kann es als das "aktivierte Neuron" ansehen. Der Koeffizient h_i ist der Wert der Nachbarschaftsfunktion, die die Stärke der Anpassung eines Neurons in Abhängigkeit seines Abstandes zum aktivierten Neuron bestimmt. Sie kann willkürlich als Tabelle festgelegt werden, oder eine Funktion bilden, etwa in Form einer Gaussglocke über der Nachbarschaft mit Mittelpunkt im Index des aktivierten Neurons.

Nachdem wir nun gesehen haben, wie ein Kohonennetz formal funktioniert, fragen wir: worin besteht der tiefere Sinn der obigen Lernregel und was kann man genau damit anfangen? Nun, Daten liegen normalerweise nicht als zufällige Verteilung innerhalb eines Raumes vor, sondern bilden oftmals kompliziertere Strukturen innerhalb dieses Raumes. Vor allem in höheren Dimensionen ist es oft sehr schwierig, diese Strukturen zu erfassen.

Die festgelegte Nachbarschaft der Neuronen und die Anpassung der Gewichte anhand dieser Nachbarschaft führen nun dazu, daß die Gewichtsvektoren im Eingaberaum ebenfalls dazu neigen, benachbart zu sein. Da der Nachbarschaftsradius während des Lernprozesses mit der Zeit verringert wird, werden nach und nach immer kleinere Gruppen von Neuronen ihre Gewichte in Richtung eines Eingabevektors ändern und sich daher immer feiner der Verteilung der Eingabedaten anpassen. Werden nun verschiedene benachbarte Punkte aus dem Eingaberaum gezogen, dann werden auch die durch diese Eingaben aktivierten Neuronen dazu neigen, benachbart (innerhalb der vorgegebenen Gitterstruktur) zu sein. Man hat auf diese Weise eine *topologieerhaltende Abbildung* konstruiert, die einen (möglicherweise hochdimensionalen) Eingaberaum auf einen fest gewählten (und niedrigdimensionaleren) Ausgaberaum (definiert durch das Gitter der Neuronen) abbildet. Ein Beispiel dazu ist in Abbildung 8.3 dargestellt.

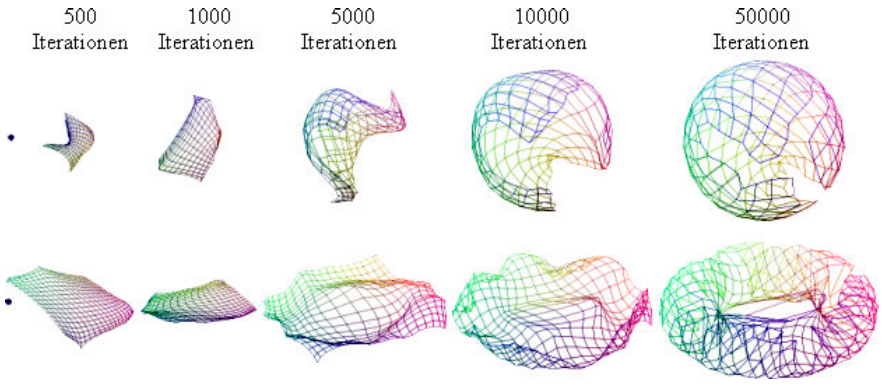

Abbildung 8.3 Abbildung einer Kugel- und einer Torusoberfläche auf ein 2-dimensionales Rechteck. Dargestellt sind die

```
Gewichtsvektoren der Neuronen im Eingaberaum. Die einzelnen
Gewichtsvektoren wurden so miteinander verbunden, wie es die
feste Nachbarschaft der Neuronen innerhalb ihrer Gitteranordnung
vorgibt, dadurch entsteht die dargestellte netzartige Struktur.
Das Netz wurde jeweils mit gleichverteilten Punkten der
Oberfläche der einzelnen Objekte trainiert und lernte dabei
selbständig die Anordnung der Gewichtsvektoren im Eingaberaum.
Werden dem trainierten Netz nun Punkte der Oberfläche
präsentiert, so erhält man 2-dimensionale Gitterkoordinaten (der
aktivierten Neuronen), wobei die Nachbarschaftsverhältnisse im
Wesentlichen erhalten bleiben. Auf diese Weise wurde eine
(weitestgehend die Topologie erhaltende) Abbildung der Oberfläche
auf eine 2-dimensionale Gitterstruktur realisiert.
```

Kohonenkarten können aus diesem Grund dazu eingesetzt werden, die Verteilung von Datenpunkten innerhalb eines höherdimensioanlen Raumes gewissermaßen zu "kartographieren", indem man sich die aktivierten Neuronen des Gitters in Abhängigkeit von der Eingabe des Netzes betrachtet.

nach oben

8.4 Neuronengas

Kohonennetze eignen sich gut zum Lernen topologieerhaltender Abbildungen in einen *festen* Zielraum. Will man jedoch die Nachbarschaftsverhältnisse des Eingaberaums *selbst* lernen, so bekommt man Schwierigkeiten, da die Nachbarschaftsverhältnisse der Neuronen zu Anfang fest vorgegeben wurden und damit unveränderlich sind. Abhilfe schafft hier das von Thomas Martinetz und Klaus Schulten in einem Paper von 1991 vorgestellte *Neuronengas*.

Die Funktionsweise des Neuronengases ist ähnlich wie die des Kohonennetzes. Auch hier wird das aktivste Neuron bei einer Eingabe **x** zuerst ermittelt, und dann in der Nachbarschaft darum gelernt. Dazu werden alle Neuronen der Gruppe an einen Eingabevektor **x** angepasst, so daß sich die Ähnlichkeit der Gewichtsvektoren erhöht und damit der euklidische Abstand im Eingaberaum verringert. Der einzige Unterschied zum Kohonen-Netz besteht allerdings darin, daß die Neuronen nun nicht mehr in einer festen Struktur angeordnet, sondern "frei" beweglich sind (daher der Ausdruck "Gas") und eine *flexible* Nachbarschaft besitzen, die sich Laufe der Zeit leicht verändern kann. Entsprechend ist auch die Lerngleichung darauf ausgelegt. Sie lautet für den Gewichtsvektor $\underline{w}_k$ des k-ten Neurons:

$$\underline{w_k} \leftarrow \underline{w_k} + \eta(t) \cdot h(k,t) \cdot (\underline{x} - \underline{w_k})$$

Sie sieht ähnlich aus, wie die des Kohonennetzes, jedoch ist die Nachbarschaftsfunktion $h(\cdot)$ aufgrund der flexiblen Nachbarschaft der Neuronen für das *k*-te Neuron definiert als:

$$\begin{aligned} h(k,t) &= \exp(-\tfrac{r(k)}{\lambda(t)}) \\ r(k) &= ||\{i : ||\underline{x} - \underline{w_i}||_2 < ||\underline{x} - \underline{w_k}||_2\}|| \\ \lambda(t) &= \lambda_0 \cdot (\lambda_1/\lambda_0)^{t/t_{max}} \end{aligned}$$

$r(k)$ gibt die Anzahl der Neuronen an, deren Gewichtsvektor sich näher an der Eingabe $\underline{x}$ befindet, als der des gerade betrachteten *k*-ten Neurons. $\lambda(t)$ ist wieder ein Parameter, der den zeitabhängigen Nachbarschaftsradius angibt, innerhalb dessen Neuronen signifikant am Lernprozess beteiligt werden. In diesem Fall entspricht er etwa

der Anzahl der Neuronen, die in der Nähe des Gewinnerneurons (dem Neuron, dessen Gewichtsvektor sich am nächsten an $\underline{x}$ befindet) lernen sollen. Damit lernen alle Neuronen, die weiter weg vom Gewinnerneuron sind, wenig, da sich viele andere Neuronen zwischen ihnen und dem Gewinnerneuron befinden. λ_0 ist der Radius zu Beginn des Lernprozesses und λ_1 der Nachbarschaftsradius, der am Ende des Lernprozesses angenommen wird, wobei $\lambda_1 << \lambda_0$ gewählt werden sollte. t_{max} bezeichnet dabei die Zahl der insgesamt durchzuführenden Iterationen und $t = 0, 1, 2, ..., t_{max}$ gibt die Iteration an, in der man sich zur Zeit befindet.

Was nun noch fehlt, ist die zeitabhängige Lernrate $\eta(t)$. Sie hat die gleiche Form, wie die Formel für den Nachbarschaftradius und ist definiert als:

$$\eta(t) = \eta_0 \cdot (\eta_1/\eta_0)^{t/t_{max}}$$

η_0 und η_1 geben dabei wieder die Lernrate zu Beginn und zum Ende des Lernprozesses vor. Dabei sollte die Beziehung $\eta_1 < \eta_0 << 1$ eingehalten werden.

Zusätzlich zu den Lernregeln für die Gewichtsvektoren gibt es beim Neuronengas (im Unterschied zu Kohonen-Netzen) noch eine weitere Regel: die Lernregel für die Nachbarschaftsbeziehungen. Welche Neuronen benachbart sind, oder nicht, wird dabei durch eine Adjazenzmatrix A bestimmt, die für jedes Paar von Neuronen einen Eintrag von +1 enthält, wenn es sich um topologische Nachbarn handelt und ansonsten den Wert 0 besitzt. Im Laufe des Lernpozesses wird diese Matrix nach und nach aufgebaut, bis sie schließlich gegen Ende im Optimalfall die korrekten Nachbarschaftsbeziehungen widerspiegelt.

Ist *k* der Index des Neurons, dessen Gewichtsvektor sich am nächsten an $\underline{x}$ befindet und *j* der Index des zweitnächsten Neurons, dann lauten die Lernregeln für die Adjazenzmatrix A:

$$\begin{array}{lllll}
1. & \forall(i, A_{ki} = 1): & T_{ki} & \leftarrow & T_{ki} + 1 \\
2. & & A_{kj} & \leftarrow & 1 \\
3. & & T_{kj} & \leftarrow & 0 \\
4. & \forall(i, T_{ki} \geq \tau): & A_{ki} & \leftarrow & 0
\end{array}$$

Dabei ist **T** eine Matrix, die für jede Kante eine Art "Alter" mitführt. Sinn davon ist, während den einzelnen Iterationen Kanten in der Adjazenzmatrix zu identifizieren, die nicht mehr benötigt werden. Die Bedeutung von Zeile 1 der obigen Lernregel besteht dann darin, das Alter aller ausgehenden Kanten des *k*-ten Neurons um eins zu erhöhen. Zeile 2 besagt, daß eine Kante zwischen Neuron *k* und Neuron *j* (den beiden nächsten an der Eingabe $\underline{x}$ befindlichen Neuronen) hinzugefügt werden soll (sofern noch nicht vorhanden). Zeile 3 setzt das Alter dieser Kante auf Null und gibt dadurch an, daß es sich um eine "junge" Kante handelt. Existiert bereits eine Verbindung zwischen *k* und *j*, dann wird diese dadurch gewissermaßen "aufgefrischt". In Zeile 4 wird dann überprüft, ob eine der ausgehenden Kanten ein vorgegebenes "maximales Alter" τ überschritten hat. Ist dies der Fall, wird die Kante durch Nullsetzen in der Adjazenzmatrix entfernt.

Da "Nachbarschaft" immer gegenseitig ist, sind die in der Adjazenzmatrix definierten Kanten ungerichtet und die Änderungen an den Matrizen müssen durch die obige Lernregel stets symmetrisch sein, d.h. nach Anwenden der Lernregeln muß sichergestellt werden, daß $A_{ij} = A_{ji}$ und $T_{ij} = T_{ji}$ für alle i, j gilt. Eleganterweise wird man beim Anwenden der Lernregeln dann eine obere oder untere Dreicksmatrix als Adjazenzmatrix verwenden.

Zusammengefaßt lassen sich die Lernoperationen folgendermaßen formulieren:

Der Lernalgorithmus für Neuronengas

A. Lernen der Gewichtsvektoren

1. $\eta \leftarrow \eta_0 \cdot (\frac{\eta_1}{\eta_0})^{t/T}$
2. $\lambda \leftarrow \lambda_0 \cdot (\frac{\lambda_1}{\lambda_0})^{t/T}$
3. für $i = 1, ..., AnzahlLNeuronen$
4. $\quad r_i \leftarrow \left|\{j : ||\underline{x} - \underline{w}_j||_2 < ||\underline{x} - \underline{w}_i||_2\}\right|$
5. $\quad h_i \leftarrow \exp(-r_i/\lambda)$
6. $\quad \underline{w}_i \leftarrow \underline{w}_i + \eta \cdot h_i \cdot (\underline{x} - \underline{w}_i)$
7. end

B. Lernen der Nachbarschaftsbeziehungen

8. $k \leftarrow$ Index des am nächsten an $\underline{x}$ liegenden Neurons
9. $j \leftarrow$ Index des zweitnächsten an $\underline{x}$ liegenden Neurons
10. für alle i mit $A_{k,i} = 1$:
11. $\quad T_{k,i} \leftarrow T_{k,i} + 1$
12. end
13. $A_{k,j} \leftarrow 1$
14. $T_{k,j} \leftarrow 0$
15. für alle i mit $T_{k,i} \geq \tau$:
16. $\quad A_{k,i} \leftarrow 0$
17. end
18. für alle i:
19. $\quad A_{i,k} \leftarrow A_{k,i}$
20. end

Abbildung 8.4 enthält noch ein Beispiel dazu, wie der Lernprozess von Neuronengas im Laufe mehrerer Iterationen im Eingaberaum aussieht. Als Parameter wurden jeweils $\eta_0 = 0.2, \eta_1 = 0.1$ für die Lernraten, $\lambda_0 = 30, \lambda_1 = 0.01$ für die Nachbarschaftsradien, $\tau = 50$ für das maximale Alter einer Kante, $t_{max} = 30000$ Iterationen und 500 Neuronen verwendet.

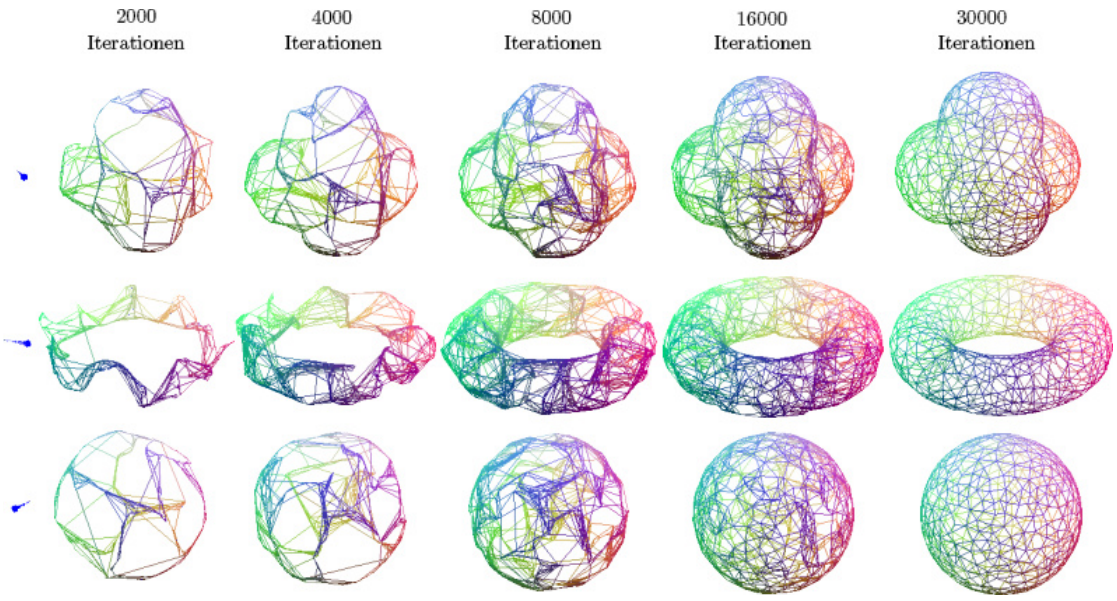

Abbildung 8.4 Neuronengas im 3-dimensionalen Raum, trainiert mit zufälligen Punkten aus der Oberfläche vier miteinander verschmolzener Kugeln (oben), einer Torusoberfläche (mitte) und einer Kugelschale (unten). Die Gewichtsvektoren der Neuronen (in der Abbildung als Eckpunkte der Verbindungslinien zu erkennen) waren anfänglich zufällig innerhalb eines kleinen Raumbereichs im Zentrum des Koordinatensystems verteilt und haben sich im Verlauf der Iterationen selbständig auf den zu lernenden Oberflächen der Objekte angeordnet. Die dargestellten Verbindungen zwischen den einzelnen Neuronen wurden entsprechend der momentan gelernten Adjazenzmatrix eingezeichnet.

nach oben