Lösungen zum Buch: Wissensverarbeitung

Kapitel 10

Künstliche neuronale Netze

Lösung 10.1 (Maschinelles Lernen)

a) *Ein Computerprogramm lernt aus einer Erfahrung $\mathcal{E}$ bezüglich einer Aufgabenklasse $\mathcal{T}$ und einer Bewertung $\mathcal{P}$, wenn sich durch $\mathcal{E}$ seine durch $\mathcal{P}$ gemessene Leistung bei Aufgaben aus $\mathcal{T}$ verbessert.*

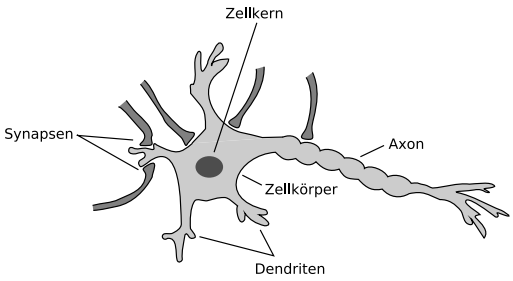
In anderen Worten: es wird das Verhalten eines Programmes bei einer bestimmten Aufgabenklasse $\mathcal{T}$ beobachtet und bewertet ($\mathcal{P}$), die Ergebnisse und in der Regel auch die Bewertung werden dem Programm als Erfahrung $\mathcal{E}$ zugänglich gemacht, welches daraufhin seine internen Parameter und Strategien ändern kann. Führt diese Änderung zu einer Verbesserung der Bewertung bei den nächsten Aufgaben, so hat das Programm gelernt.

b) *Die drei Lernparadigmen überwachtes, bestärkendes und unüberwachtes Lernen unterscheiden sich im Feedback, welches das lernende System vom Lehrer bzw. der Umwelt erhält. Es bekommt die richtige Antwort, eine Bewertung oder keine Rückmeldung.*

c) *Der Konflikt tritt beim bestärkenden Lernen auf. Da die Trainingsmenge vom System selbst durch Auswahl der nächsten Aktion mitbestimmt wird, besteht stets die Wahl zwischen der bis jetzt gefundenen optimalen Aktion für die aktuelle Situation (bzw. einer Suche in deren Nähe) oder dem Erforschen nicht erprobter Aktionen mit unbekanntem Ausgang.*

Lösung 10.2 (Nervenzelle)

a) *Nervenzelle:*



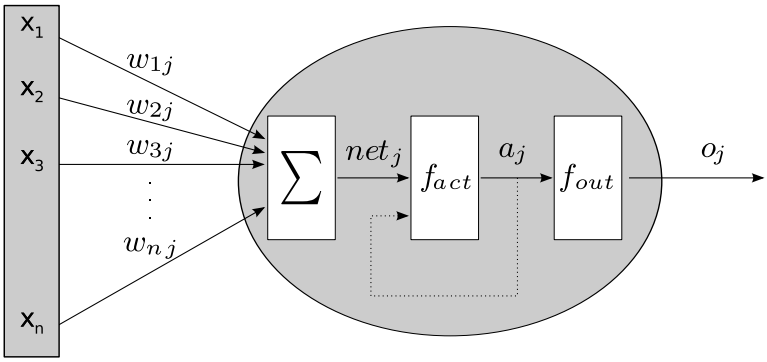
Lernprozesse in Lebewesen werden auf neuronaler Ebene auf das Entstehen und Verändern von Synapsen zurückgeführt.

b) *Hebbsche These:* „Bei gleichzeitiger Aktivität der präsynaptischen und postsynaptischen Zelle wird die Synapse verstärkt."

Werden Zellen, die durch eine Synapse verbunden sind, gleichzeitig aktiviert, so wird die Synapse so verändert, dass die Aktivierung leichter von einer auf die andere Zelle übertragen werden kann.

Lösung 10.3 (Neuronenmodell)

a) *Neuronenmodell:*

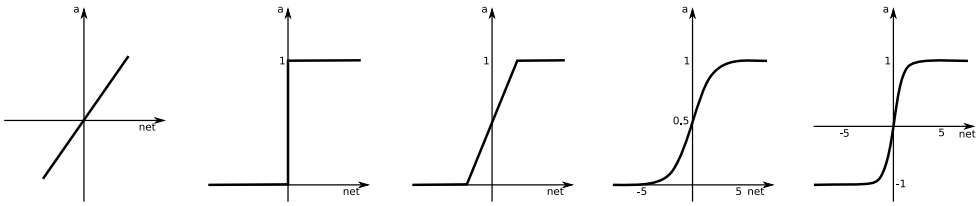


Die Eingabewerte x_i werden mit den jeweiligen Wichtungen w_i multipliziert. Die Summe dieser Produkte wird als Netzaktivität net bezeichnet. Anhand der Netzaktivität (und selten auch der aktuellen Aktivierung) wird durch die Aktivierungsfunktion die neue Aktivierung a berechnet. Die Ausgabefunktion bestimmt aus der Aktivierung die Ausgabe o des Neurons. Hierbei wird oft $o = a$ verwendet.

b) *Skalarprodukt aus Eingabevektor x und Wichtungsvektor w:*

$$net = \sum_i w_i \cdot x_i$$

c) *Aktivierungsfunktionen*



Lösung 10.4 (Perzeptron)

a) *0 und 1.*

b) *Wenn die Netzaktivität den Schwellwert übersteigt.*

c) *Das ON-Neuron ist ein Hilfsmittel, um den Schwellwert eines Perzeptrons als normale Wichtung zu behandeln. Der Schwellwert wird hierbei als negative Wichtung eines ständig mit 1 aktivierten zusätzlichen Neurons $n+1$ interpretiert, also $w_{n+1} = -\Theta$.*

d) *Die Menge der Eingabewerte, für die ein Perzeptron mit n Eingabeneuronen die Ausgabe 1 erzeugt, ist beschrieben durch:*

$$\sum_{i=1}^{n} w_i x_i \ > \Theta,$$

die Trennfläche zwischen 1- und 0-erzeugenden Punkten somit:

$$\sum_{i=1}^{n} w_i x_i \ = \Theta.$$

Die Trennfläche bei $n = 2$ ist die Gerade

$$w_1 x_1 + w_2 x_2 \ = \Theta,$$

die wir als Funktion $x_2 = f(x_1)$ darstellen:

$$x_2 = \frac{-w_1}{w_2} \cdot x_1 + \frac{\Theta}{w_2} \qquad (w_2 \neq 0).$$

e) Wahrheitstafel des NAND:

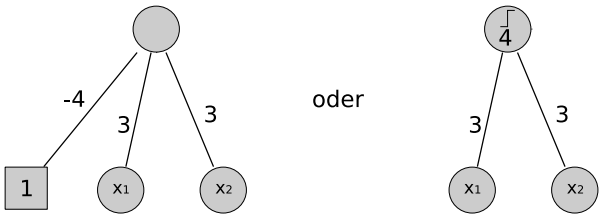
x_1	x_2	a
0	*0*	*1*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

Eine Lösung wäre beispielsweise ein Perzeptron mit $w_1 = w_2 = -2$ und Schwellwert $\Theta = -3$ (bzw. $w_3 = 3$). Die Aktivierungsfunktion des Perzeptrons ist die Sprungfunktion.

x_1	x_2	*net*	*net>-3?*	a
0	*0*	*0*	*ja*	*1*
0	*1*	*-2*	*ja*	*1*
1	*0*	*-2*	*ja*	*1*
1	*1*	*-4*	*nein*	*0*

Lösung 10.5 (Perzeptron mit unbekannter Funktion)

a)



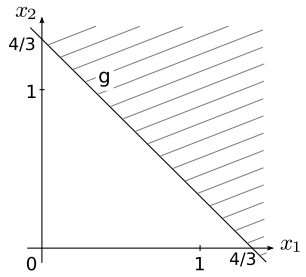
b) Wahrheitstafel:

x_1	x_2	*net*	*net>4?*	a
0	*0*	*0*	*nein*	*0*
0	*1*	*3*	*nein*	*0*
1	*0*	*3*	*nein*	*0*
1	*1*	*6*	*ja*	*1*

c) UND-Funktion

d) $x_2 = f(x_1) = -x_1 + \frac{4}{3}$

e)



Lösung 10.6 (Perzeptron-Lernen)

a) *Überwachtes Lernen*

b)

$$\begin{aligned} w_i(t+1) &= w_i(t) + \eta \cdot x_i \cdot (y - a) \\ \Theta(t+1) &= \Theta(t) - \eta \cdot (y - a) \end{aligned}$$

$w_i(t+1)$:	*Wichtung im nächsten Zeitschritt,*
$w_i(t)$:	*aktuelle Wichtung,*
η:	*Lernrate,*
x_i:	*Eingabewert an Wichtung* w_i,
y:	*Sollwert der Ausgabe bei Eingabe von* x_i,
a:	*Aktivierung und Ausgabe des Perzeptrons,*
$\Theta(t+1)$:	*Schwellwert im nächsten Zeitschritt,*
$\Theta(t)$:	*Schwellwert im aktuellen Zeitschritt*

c) *Das Lernverfahren konvergiert bei linear separierbaren Lernaufgaben.*

d) *Nicht linear separierbare Lernaufgaben können nicht repräsentiert werden.*

Lösung 10.7 (Netztopologien)

a) *Neuronen sind in Schichten angeordnet, wobei die Neuronen einer Schicht ihre Ausgaben nur an Neuronen der nächsten Schicht weitergeben.*

b) *Wichtungen rückwärts zur Hauptrichtung, sogenannte Rückkopplungen.*

c) $10 \cdot 14 + 14 \cdot 5 + 5 \cdot 8 = 250$

Lösung 10.8 (Multilayer-Perzeptron)

a) *Wahrheitstafel:*

x_1	x_2	net_{h1}	$h1$	net_a	a
0	*0*	*-1*	*0*	*2*	*1*
0	*1*	*-3*	*0*	*2*	*1*
1	*0*	*1*	*1*	*-1*	*0*
1	*1*	*-1*	*0*	*2*	*1*

b) $a = \neg x_1 \vee x_2 = x_1 \rightarrow x_2$ *(Implikation)*

Lösung 10.9 (Backpropagation-Algorithmus)

a)
- *Lernphase: Das Netz wird konstruiert und seine Wichtungen durch wiederholtes Präsentieren der Trainingsmenge verändert. Diese Phase ist durch das Einstellen der freien Parameter (Neuronenanzahl,*

Lernrate usw.) und die häufigen Berechnungen beim Lernen zeitaufwendig und rechenintensiv.

- *Einsatz-Phase: Das Netz wird in der Anwendung verwendet und seine Wichtungen nicht mehr verändert. Durch Anlegen eines Eingabemusters und Vorwärtspropagierung der Aktivierung durch das Netz wird die Netzausgabe berechnet. Der Rechenaufwand hierbei ist gering.*

b) *Fehler eines Ausgabeneurons:*

$$\delta_j = f'_{act}(net_j) \cdot (y_j - o_j)$$

Ist f_{act} die logistische Funktion, so gilt:

$$\delta_j = a_j \cdot (1 - a_j) \cdot (y_j - o_j)$$

δ_j:	*Fehler des Ausgabeneurons j,*
$f'_{act}(net_j)$:	*Ableitung der Aktivierungsfunktion des Neurons j an der Stelle der seiner aktuellen Netzaktivität,*
y_j:	*Sollwert der Ausgabe des Neurons j,*
o_j:	*Ausgabe des Neurons j,*
a_j:	*Aktivierung des Neurons j (beim BPV: $a_j = o_j$)*

c) *Fehler eines versteckten Neurons:*

$$\delta_j = f'_{act}(net_j) \cdot \sum_{k \in \mathcal{K}} w_{jk} \cdot \delta_k$$

Ist f_{act} die logistische Funktion, so gilt wieder:

$$\delta_j = a_j \cdot (1 - a_j) \cdot \sum_{k \in \mathcal{K}} w_{jk} \cdot \delta_k$$

δ_j:	*Fehler des versteckten Neurons j,*
$\mathcal{K}$:	*Menge der Neuronen, die vom Ausgang des Neurons j angesteuert werden,*
w_{jk}:	*Wichtung vom Neuron j zum Neuron k,*
δ_k:	*Fehler des Neurons k*

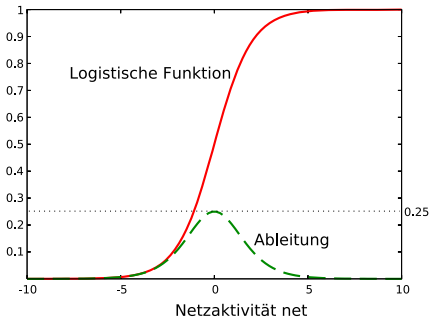
d) *Berechnung der Korrekturwerte der Wichtungen:*

$$\Delta w_{ij} = \eta \cdot o_i \cdot \delta_j$$

Δw_{ij}:	*Wichtungsänderung der Wichtung vom Neuron i zum Neuron j,*
η:	*Lernrate,*
o_i:	*Ausgabe des Neurons i,*
δ_j:	*Fehler des Neurons j, siehe a) und b) dieser Aufgabe*

e) *Logistische Funktion:*

$$a = f(net) = \frac{1}{1 + e^{-net}}$$



Lösung 10.10 (Backpropagation-Anwendung)

a) *Der MSE drückt die Abweichung eines neuronalen Netzes bei der Vorhersage der Klassen einer Musterdatenmenge aus.*

Er dient zur Kontrolle des Lernvorgangs (z. B. Erkennen einer zu hohen Lernrate mit dem MSE der Lerndaten) und zum Abbruch des Trainings bei der Methoden Frühes Stoppen *(Ansteigen des MSE der Testdaten).*

$$MSE = \frac{1}{M} \sum_{m=1}^{M} \sum_{j=1}^{J} (y_j^m - o_j^m)^2.$$

MSE:	*Mean Squared Error,*
M:	*Anzahl der Muster,*
J:	*Anzahl der Ausgabeneuronen,*
y_j^m:	*Sollwert des Ausgabeneurons j beim Muster m,*
o_j^m:	*Ausgabe des Ausgabeneurons j beim Muster m*

b) *Beim Online-Lernen werden die Wichtungen nach jedem Muster verändert. Beim Offline-Lernen werden die Fehler über einen kompletten Zyklus (Anlegen aller Muster) akkumuliert und die Wichtungen nach dem Zyklus verändert.*

c) *1) Kleine Zahlen sind günstig, denn sie erzeugen kleine Netzaktivitäten, bei denen die Ableitung der Aktivierungsfunktion relativ große Werte annimmt. Die Ableitung geht als Faktor in die Wichtungsänderung ein, d. h., Neuronen mit kleinen Eingangswichtungen sind in der Regel lernfreudig.*

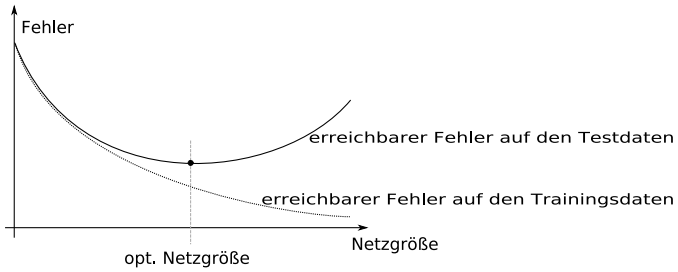
2) Zufallszahlen sind notwendig, denn bei gleicher Initialisierung aller Wichtungen zweier aufeinanderfolgender Wichtungsschichten können sich die Wichtungen nur eingeschränkt entwickeln. Insbesondere bleiben die Wichtungen eines Neurons der ersten Neuronenschicht zu allen Neuronen der zweiten Schicht gleich und ebenso alle eingehenden Wichtungen eines Neurons der dritten Neuronenschicht.

d) *Das Hauptproblem aller Gradientenabstiegsverfahren besteht im Verfangen in lokalen Minima.*

e) *Die Lernschritte sind zu groß, der Trainingsfehler oszilliert unregelmäßig und die Erkennungsleistung wird nicht weiter verbessert.*

f) *Je mehr Wichtungen ein Netz enthält, desto genauer kann es die Trainingsdaten repräsentieren. Allerdings verschlechtert sich ab einer bestimmten*

Größe die Generalisierungsfähigkeit, so dass der erreichbare Fehler auf den Testdaten wieder ansteigt.



g) Die erreichbaren Fehler auf Test- und Trainingsdaten sind annähernd gleich.

h)
- *Mangelnde Generalisierungsfähigkeit*
- *Verlangsamung des Lernens*
- *Mehr Minima im Fehlergebirge*

Lösung 10.11 (Trainingsmenge)

a) Entfernen irrelevanter Attribute und sinnvolle Aggregation von Attributen

b) nominales Attribut, z.B. $Name = \{Paul, Fred, Paula, Sabine\}$
ordinales Attribut, z. B. $Helligkeit = \{schattig, halbschattig, sonnig\}$
numerisches Attribut, z. B. $Alter = \{1, 2, 3, ..., 150\}$

c) Zwei grundsätzliche Möglichkeiten:
- *Transformation auf den Wertebereich der Aktivierung, z. B. auf [0,1] oder [0.1,0.9]*
- *Aufteilung in Intervalle und Zuordnung eines Neurons zu jedem Intervall*

d) Zwei Varianten: 1-aus-n-Kodierung oder Kodierung als Zahl

e) - als fehlend kodieren
- durch häufigen (oder geschätzten) Wert ersetzen

f) Konsistenz: gleiche Eingaben erzeugen gleiche Ausgaben

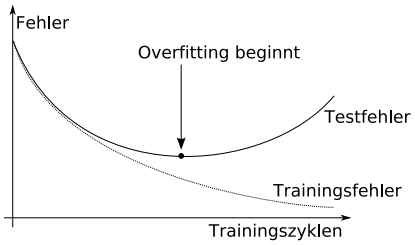
g) Repräsentativität: Häufigkeiten der Klassen der Trainingsdaten sollten mit denen im späteren Einsatzbereich übereinstimmen.

h)

$$\textit{Ausgabe sei } \left\{ \begin{array}{c} 1\ 0\ 0 \\ 0\ 1\ 0 \\ 0\ 0\ 1 \end{array} \right\}, \textit{ wenn für die Eingabe gilt } \left\{ \begin{array}{c} x_3 \neq 0.1 \wedge x_4 \neq 0.8 \\ x_3 \neq 0.1 \wedge x_4 = 0.8 \\ x_3 = 0.1 \end{array} \right\}.$$

Lösung 10.12 (Fehler und Overfitting)

a) *Overfitting:*



Der Fehler des Netzes auf den Testdaten wächst und damit sinkt die Erkennungsleistung in der späteren Anwendung.

b) *Das Lernen wird beim Erkennen des beginnenden Overfittings abgebrochen.*

c) *Auslassungsmethode, Kreuzvalidierung und Bootstrap-Methode*

d) *Bei der n-fachen Kreuzvalidierung teilen wir die Menge der Lerndaten zufällig in n annähernd gleich große, disjunkte Partitionen. Jede einzelne wird als Testmenge gewählt und das Netz mit den anderen $n-1$ trainiert. Der Mittelwert der erreichten Testfehler ist ein Schätzwert für den zu erwartenden Generalisierungsfehler.*

Wird n auf die Anzahl der Lerndatensätze gesetzt, so ergibt sich die Partitionsgröße 1, und damit das Verfahren leave-one-out.

e) *Betroffen ist hier die Auslassungsmethode. Eine Menge von Datensätzen, die nur zum kontrollierten Abbruch der Lernvorganges benutzt wird, darf (theoretisch) nicht als Testdatenmenge zur Schätzung des Generalisierungsfehlers benutzt werden, da sich das Netz durch den Abbruch beim ansteigenden Fehler an diese Datenmenge anpassen konnte. Die Schätzung fiele zu optimistisch aus.*

Lösung 10.13 (Backpropagation-Erweiterungen)

a) *Die Wissensrepräsentation eines neuronalen Netzes besteht aus vielen überlagerten Wichtungsstrukturen verschiedener Stärke, wobei eine Wichtungsstruktur als unscharfer Wenn-dann-Zusammenhang zwischen Ein- und Ausgabe interpretiert werden kann - ähnlich einer Fuzzy-Regel. Das gleichzeitige Aktivieren und Abarbeiten der Strukturen gleicht dem parallelen Anwenden von Fuzzy-Regeln.*

b)
- *Weight Decay vermeidet hohe Wichtungen durch einen proportionalen Abzug in jedem Lernschritt*
- *Quickprop nähert die Fehlerfläche durch eine Parabel an und verändert im Lernschritt die Wichtungen so, dass zum Minimum der angenommenen Parabel gesprungen wird.*
- *Resilient Propagation steuert die Größe und Richtung des Lernschrittes mit Hilfe der Vorzeichen des aktuellen und des letzen Anstiegs der Fehlerfläche.*

Lösung 10.14 (Backpropagation-Anwendungsgebiete)

a) *Anwendungsgebiete:*

Mustererkennung, Diagnose, Beurteilung, Prognose, Regelung u.v.a.

Voraussetzung: es müssen (in der Regel klassifizierte) Lerndaten vorliegen.

b) *Vorteile: Adaptivität, Generalisierungsfähigkeit, Ausdrucksmächtigkeit, Fehlertoleranz, Robustheit, Domänenunabhängigkeit, Kurze Ausführungszeit in der Recall-Phase, Anytime-Algorithmus*

Probleme: Datenqualität und -menge entscheidend für Lernerfolg, hoher Zeitaufwand für Training und Konfiguration, fehlende Transparenz, Vorwissen schwer einzubringen

c) *Klasse, Wertung, Muster, Vorhersage, Aktion u. a.*

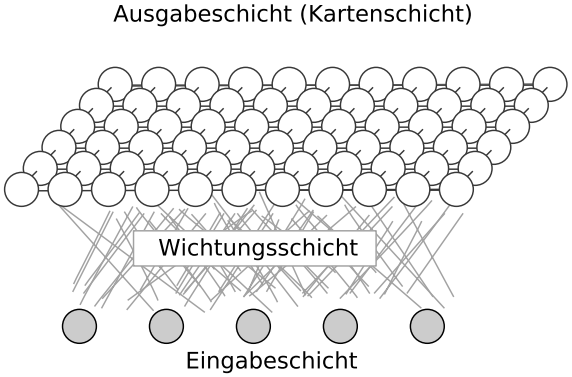
Lösung 10.15 (SOM)

a) *Unüberwachtes Lernen*

b) *Neuronengebiete der Großhirnrinde, die bei Reizung von Sinneszellen aktiviert werden.*

c) *Topologieerhaltende Abbildung auf wenige (bspw. zwei) Dimensionen*

d) *SOM:*



e) *Kette, Gitter (quadratisch oder hexagonal), Quader*

Lösung 10.16 (SOM-Lernvorgang)

a) *Es gewinnt das Neuron, dessen Wichtungsvektor den kleinsten euklidischen Abstand zum Eingangsvektor aufweist, oder bei einer zweiten Form, dessen Wichtungsvektor mit dem Eingangsvektor das größte Skalarprodukt ergibt.*

b) *Wichtungsänderung Δw_c des Gewinnerneurons:*

$$\Delta w_c = \eta \cdot (x - w_c)$$

η: *Lernrate,*
x: *Eingabemuster,*
w_c: *Wichtungsvektor des Gewinnerneurons*

Wichtungsänderung Δw_j eines Neurons j:

$$\Delta w_j = \eta \cdot h_{cj} \cdot (x - w_j)$$

h_{cj}: *Nachbarschaftsfunktion (meist Funktion des Abstandes zwischen Neuron j und dem Gewinnerneuron c*

c) Sie bestimmt, welche Neuronen lernen und wie stark sie lernen.

Neuronen, für die der Wert der Nachbarschaftsfunktion ungleich Null ist, verändern im Lernschritt ihre Wichtungsvektoren, und zwar um so stärker, je näher sie an dem Gewinnerneuron liegen.

d) Lernrate und Lernradius

e) Sie versuchen sich sowohl an die Verteilung der Eingabemuster anzupassen, als auch gleichzeitig in ihrer topologischen Struktur benachbart zu bleiben.

f) Quantisierungsfehler e eines Musters x: $e = ||x - w_c||$

w_c: *Wichtungsvektor des Gewinnerneurons*

Der Quantisierungsfehler eines Musters verdeutlicht, wie gut die Karte das konkrete Muster x repräsentiert. In anderen Worten, wie weit sich die Karte dem Eingabemuster im Eingaberaum genähert hat.

g) Wird eine SOM beim Lernvorgang durch eine U-Matrix visualisiert, so scheint sich der Lernvorgang als Serie von Hammerschlägen auf ein plastisches Material darzustellen: Der Eingabevektor x bestimmt das Zentrum des Schlages, die Nachbarschaftsfunktion h_{cj} die Form des Hammers, der Lernradius d die Ausdehnung des Schlagwerkzeuges und die Lernrate η die Tiefe der Verformung oder Kraft des Schlages.

Lösung 10.17 (SOM-Visualisierung)

a) Musterunabhängig (z. B. Karte im Eingaberaum), abhängig von einem Muster (z. B. Distanzmatrix), abhängig von allen Mustern (z. B. Gewinnhistogramm)

b) Die Wichtungen eines Eingabeneurons

c) Die Wichtungsvektoren der Kartenneuronen haben die Dimensionalität des Eingaberaumes und lassen sich somit als Punkte im Eingaberaum markieren. Diese Punkt werden mit Hilfe der Nachbarschaft der Kartenneuronen mit Linien verbunden.

d) Der U-Wert eines Kartenneurons ist die Summe der Abstände des Wichtungsvektors des Kartenneurons zu den Wichtungsvektoren seiner direkten topologischen Nachbarn.

e) Verfügen die Eingabemuster über einen Namen oder eine Klasse, so kann das Gewinnerneuron eines Eingabemusters damit beschriftet werden.

Lösung 10.18 (SOM-Anwendung)

a) Datenvorverarbeitung für überwachte Lernverfahren, Optimierung, Data Mining (Finden von Clustern und Regeln), Überwachung und Anomaliedetektion, Kontextkarten

b) *Durch Abbildung hochdimensionaler Lerndaten auf wenige Dimensionen kann die Lernaufgabe für das BPV vereinfacht werden.*

c) *Kontextkarten stellen Objekte mit ähnlichem Kontext benachbart dar. Hierzu wird als Trainingsdatensatz nicht die Beschreibung eines Objektes, sondern die Beschreibung des Kontextes des Objektes verwendet, beispielsweise eine Liste zeitlich oder räumlich naheliegender anderer Objekte.*

d) *Eine mit den Merkmalsvektoren normaler Systemzustände trainierte Karte deckt im Merkmalsraum den gelernten Teilraum ab, so dass zu einem Muster, welches den Trainingsdaten ähnelt, ein nicht weit entferntes Gewinnerneuron gefunden werden kann - das Muster hat einen kleinen Quantisierungsfehler. Muster die jedoch untypisch sind, sich also weit entfernt von dem durch die Karte belegten Raum im Merkmalsraum befinden, erzeugen eine hohen Quantisierungsfehler und können so erkannt werden.*

Für Fragen und Hinweise kontaktieren Sie mich unter
boersch@fh-brandenburg.de.