

Kapitel 2

Neuromathematische Aspekte

„Wie sieht das Gleichungssystem für das Gehirn aus? Es hat offenbar einen gewaltigen Umfang; denn da man wenigstens eine Gleichung pro Neuron braucht, muss es mindestens 10 Milliarden Gleichungen umfassen.“ (Günther Palm, 1988)

„Wenn wir über Mathematik sprechen, sprechen wir möglicherweise über eine sekundäre, auf die im Zentralnervensystem verwendete Primärsprache aufbauende Sprache.“ (John von Neumann, 1957)

Die Funktionsweise einer Nervenzelle¹ mathematisch zu beschreiben und auf diesem Wege Nachbildungen der Leistungen eines Gehirns zu erhalten, übt seit Jahrzehnten seinen vielfältigen Reiz auf die Forschung aus. Die eingangs zitierten Gedanken von Günther Palm und von John von Neumann belegen zwei der zahlreichen Fragestellungen, die der Neuromathematik zugerechnet werden können. Die Frage von Günther Palm verweist auf das in der Mathematik geläufige Verhalten, den Zusammenhang zwischen Ein- und Ausgaben eines Objektes durch eine Gleichung auszudrücken.² Darauf wird in diesem Kapitel näher eingegangen werden.

Fragestellungen der Neuromathematik

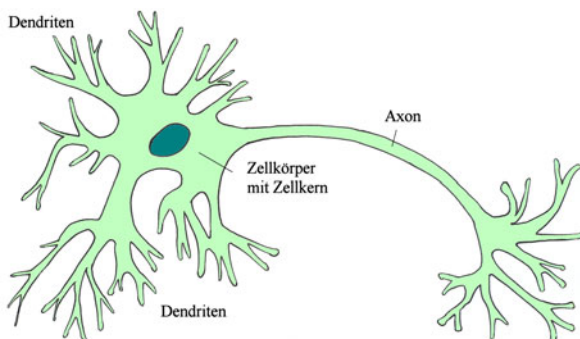


Abb. 2.1: Nervenzelle mit Dendriten, Zellkörper und Axon

John von Neumann hebt in seiner Betrachtung auf eine Unterscheidung zwischen der Mathematik ab, die die Vorgänge im Gehirn beschreibt, und der Mathematik, über die mit einem solchen Gehirn dann nachgedacht werden kann.³ Er nimmt damit einerseits an, dass sich die Vorgänge im Gehirn mathematisch modellieren lassen, und deutet andererseits eine Abhängigkeit unserer mathematischen Erkenntnisse von der Funktionsweise unseres Gehirns an.^{4, 5} Das mag zum einen Anlass für Überlegungen sein, ob sich unseren mathematischen Erkenntnissen prinzipielle Grenzen setzen, und zum anderen unsere Neugierde darauf lenken, welche mathematischen Probleme mit Hilfe mathematischer Modelle von Nervenzellstrukturen lösbar sind und werden.⁶ Schon die Aufgabe, die Grundrechenarten durch einen Verbund von Nervenzellen ausführen zu lassen, stellt eine interessante Herausforderung dar, wie das Kapitel 6 zum Assoziativen Rechnen in diesem Buch zeigen wird.

neuronale Struktur,
neuronales Netz

Ein Verbund von Nervenzellen wird im Weiteren **neuronale Struktur** oder **neuronales Netz** genannt. Es sind unter diesen Begriffen nicht nur einzelne Bündel untereinander wechselwirkender Neuronen zu verstehen, sondern auch umfangreichere Strukturen, die sich aus solchen Bündeln zusammensetzen lassen und die zum Bau von Assoziativmaschinen (s. Kapitel 4) benötigt werden. Die Einordnung neuronaler Netze in das Gebiet der Assoziativspeicher wird in Kapitel 3 angesprochen.

biologische Vorgaben
für mathematisches
Handeln

Das *Neuromath Network* erweiterte zu Beginn der 2000er-Jahre den Begriff von Neuromathematik um die Bemühungen für das Verständnis derjenigen Hirntätigkeiten, die einerseits mathematische Erkenntnis ermöglichen und die andererseits mathematische Unfertigkeiten verursachen.⁷ Das Kapitel 6 dieses Buches wird zu diesem Aspekt beitragen, indem es ein Rechnen mit Assoziierwerken im Einzelnen beschreibt.

Angewandte Mathe-
matik mit neuronalen
Netzen vs.
Mathematik der
Nervenzelle

Während von einigen Forschern die Neuromathematik als derjenige Teil der Angewandten Mathematik aufgefasst wird, der die Verfahren zur Lösung mathematischer Aufgaben mit Hilfe künstlicher neuronaler Netze entwickelt⁸ wird die Neuromathematik andernorts als eine Mathematik aufgefasst, die die Vorgänge in und zwischen Nervenzellen beschreiben hilft.⁹

Modellierung der
Eigenschaften natür-
licher neuronaler
Strukturen

Die Neuromathematik erhält in diesem Buch nicht die Aufgabe, neurobiologische Phänomene wie etwa die Art der Signalweiterleitung durch die oder zwischen den Nervenzellen möglichst genau zu beschreiben, sondern die Zuordnungseigenschaften von Nervenzellstrukturen zu modellieren und sie für mathematische Prozesse zu nutzen. Die biologischen Einzelheiten der Nervenzelle (s. Abb. 2.1) rücken dabei in den Hintergrund. Stattdessen werden die Dendriten als Orte der Eingabe von Werten $\mathbf{x} = (x_1, x_2, \dots, x_n)$ für eine Zuordnung f verstanden, der Zellkörper versinnbildlicht den Zuordnungsterm $f(\mathbf{x})$ und das Axon mit seinen Verästelungen leitet den Ausgabewert y der Zuordnung weiter, kurz: $\mathbf{x} \xrightarrow{f} y$.

keine Nachbildung
natürlicher
Spannungsverläufe

Geleitet von der Feststellung, dass sich in neuronalen Strukturen alle Situationen und Aktionen durch Folgen zweier Werte darstellen lassen,¹⁰ genügen uns einfache schaltalgebraische beziehungsweise binäre schaltungstechnische Mittel zur Erzeugung arbeitsfähiger Systeme. Die Nachbildung natürlicher Spannungsverläufe wie zum Beispiel nach dem Hodgkin-Huxley-Modell¹¹ (s. Abb. 2.2 und Abb. 2.3) zum Aufbau neuromorpher Strukturen aus Schaltkreisen der Analogtechnik wird nicht beabsichtigt.¹² Wie im Streben um den

Bau eines neuromorphen Speichers¹³ bietet es sich auch für den Bau von Assoziativmaschinen an, bei einem Fertigungsprozess auf die technischen Möglichkeiten hoher Bauteildichte zuzugreifen.¹⁴

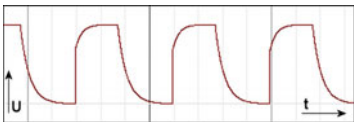
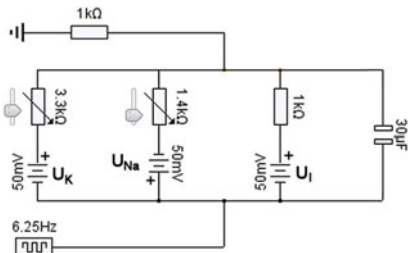


Abb. 2.2: Modellneuron nach Hodgkin-Huxley mit einstellbaren Stromstärken für den Kalium- und Natriumionentransport

Abb. 2.3: Spannungsverlauf am Modellneuron nach Hodgkin-Huxley bei regelmäßiger Anregung durch einen Taktgeber (s. Abb. 2.2)

Die zum Aufbau von Modellneuronen eingesetzten Bauteile entnehmen wir in unseren Simulationen, wie es die Ausschnitte in den Abbildungen 2.4 und 2.5 erkennen lassen, aus der Digitalelektronik. Es kommen die üblichen UND-Gatter und geläufige bistabile Kippstufen zum Einsatz.

einfache digitale Bauteile

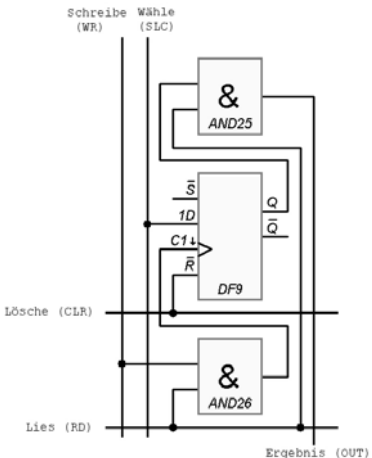
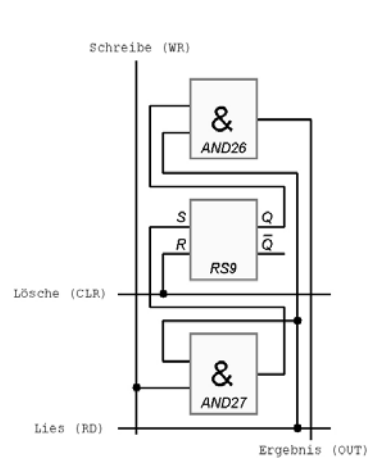


Abb. 2.4: Schaltung für eine synaptische Verbindung des **Langzeit**gedächtnisses einer Assoziativmaschine

Abb. 2.5: Schaltung für eine synaptische Verbindung des **Kurzzeit**gedächtnisses einer Assoziativmaschine

Diese kurze Gegenüberstellung technischer Einzelheiten könnte weiter ausgearbeitet werden, um die Unterschiede von Konzepten der Neuroinformatik zu unserem Ansatz, der zur Neuromathematik führt, sichtbar zu machen. Es ist das Ziel, binäre Daten zu verarbeiten, alle Vorgänge durch Folgen von „Nullen“ und „Einsen“, also durch zwei Zustände darzustellen und auszulösen. Auf die sich daraus ergebenden schaltungstechnischen Möglichkeiten wird im kommenden Text nicht weiter eingegangen. Für den technisch interessierten Leser finden sich jedoch weiterführende Darlegungen bei Ulrich Rückert¹⁵ und erläuterte Schaltpläne in [Dierks 2005].

binärer Ansatz

In den weiteren Abschnitten dieses Kapitels werden zur Einordnung des binären Ansatzes in das Thema Neuronale Netze einige grundlegende Ideen vorgestellt. Darunter sind dann auch solche, die sich mit dem eben genannten binären Ansatz kaum vereinbaren lassen, aber dem Überblick auf dem Gebiet der Modellierung neuronaler Strukturen dienen sollen.

2.1 Modellneuronen

Die nebenstehende Abb. 2.6 veranschaulicht die Vorstellung von einem ersten, einfachen Modellneuron. Die an den waagrecht eingezeichneten Leitungen (Zeilenleitungen, Dendriten) angelegten Werte x_i werden über die mit punktförmigen Markierungen gekennzeichneten Verbindungsstellen auf die senkrechte Leitung (Spaltenleitung, Axon) übertragen. Mit Hilfe einer Zuordnung f wird der Wert von y ermittelt. Für jedes Neuron wird dieselbe Zuordnung f angenommen. Diese Darstellung lässt sich in einfacher Weise für mehrere, gemeinsam mit Werten x_i zu versorgende Modellneuronen nutzen wie Abb. 2.7 zeigt. Doch würden hier alle Ausgänge y_i den gleichen Wert annehmen, so dass diese Modellierung ungenügend ist.



Abb. 2.6: Einfaches Modellneuron

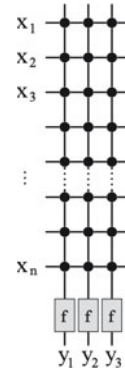


Abb. 2.7: Drei einfache, gemeinsam versorgte Modellneuronen

gemeinsam versorgte Modellneuronen

wahlweise gemeinsam versorgte Modellneuronen

Verbindungsmatrizen V

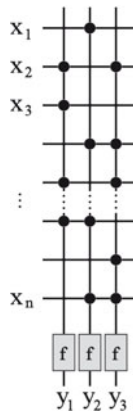


Abb. 2.8: Drei einfache, wahlweise gemeinsam versorgte Modellneuronen

Eine Verbesserung des ersten Modells erhält man, indem Verbindungsstellen durch Weglassen der punktförmigen Markierung als unwirksam eingetragen werden können. In der nebenstehenden Abb. 2.8 hat beispielsweise der Wert x_1 nur Einfluss auf den Wert von y_2 . Das Verbindungsmuster lässt sich dann wie folgt in Gestalt einer $n \times 3$ -Matrix V schreiben:

$$V = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

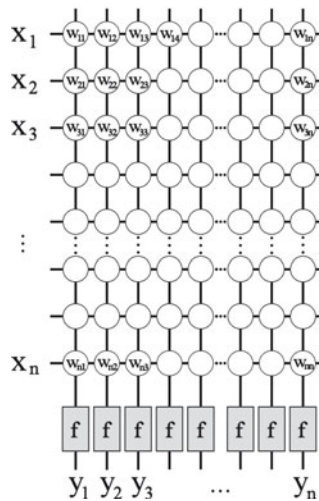
Als Zuordnungsgleichung für f käme hier zum Beispiel

$$f(s) = \begin{cases} s, & \text{falls } s \geq 3 \\ 0, & \text{sonst} \end{cases}$$

Zuordnungsgleichung

in Betracht, wobei mit $s \in \mathbb{R}$ die jeweilige Spaltensumme des Neurons beziehungsweise der Matrix bezeichnet sei. Das Modellneuron leitet in diesem Fall also nur dann einen Wert weiter, wenn wenigstens drei Verbindungsstellen wirksam wurden.

Die jetzt erreichte Modellierung wäre für die späteren Inhalte dieses Buches nahezu ausreichend. Dennoch setzen wir noch fort, um auch die Möglichkeit nachzubilden, dass jede Eingabe x_i einen verschieden großen Einfluss auf das jeweilige Neuron nehmen kann. Dazu wird an den Verbindungsstellen in die Markierung hineingeschrieben, mit welcher Zahl w_{ij} die Eingabe x_i multipliziert werden soll.¹⁶ Auf diese Weise soll die Stärke der Verbindung zwischen der i . Zeilen- und j . Spaltenleitung und damit der Einfluss auf den Wert von y_j angegeben werden.



gewichtete
Verbindungen

Abb. 2.9: Mehrere Modellneuronen mit den Verbindungsgewichten w_{ij}

Sei zum Beispiel der Wert von $x_3 = 0,7$ und das Verbindungsgewicht $w_{31} = 5,2$, dann würde die dritte Zeilenleitung einen Wert von $3,64$ zur Spaltensumme für y_1 beisteuern.

B1

Die Verbindungsgewichte lassen sich wie oben bei den wahlweise gemeinsam mit Werten x_i zu versorgenden Modellneuronen ebenfalls zu Verbindungsmatrizen $V = (w_{ij})$ anordnen.

Verbindungsmatrizen
enthalten
Verbindungsgewichte

Dieser Stand der Modellierung gibt die biologischen Verhältnisse auch hinsichtlich sich **ändernder** Verbindungsgewichte wieder, wie Donald Hebb sie 1949 bei der Entdeckung der synaptischen Plastizität vorfand.¹⁷ Also hat man sich die Verbindungsgewichte nicht als feste Größen vorzustellen, sondern als Werte, die sich bei jedem Zuordnungsschritt verändern können.

Verbindungsgewichte
können sich ändern

Zuletzt berücksichtige man bei der Nachbildung einer Nervenzellstruktur, dass jedes Neuron über sein Axon Verbindung zu den Dendriten jedes anderen Neurons herstellen könnte. Abb. 2.10 zeigt, wie dazu die Spaltenleitungen den Zeilenleitungen zugeordnet sein sollen: die Spaltenleitung zu y_i reicht seinen Wert an die Zeilenleitung für x_i weiter. Sollte es zwischen einer Spalte

Verbindung von
Modellneuronen
untereinander

(dem Axon eines Neurons) und einer Zeile (einem Dendriten eines anderen Neurons) keine Verbindung geben, so werden in diesem Modell die betroffenen Verbindungsgewichte w_{ij} auf null gesetzt.

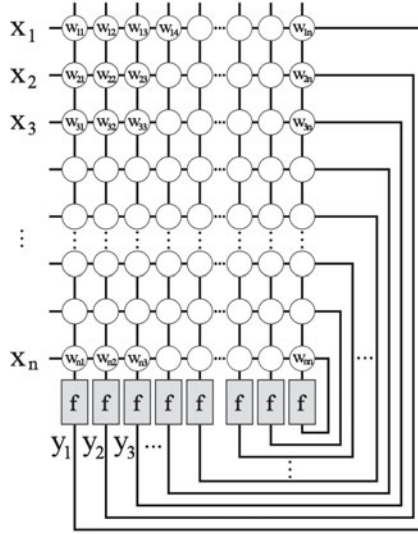


Abb. 2.10: Modellneuronen und ihre gegenseitige Verbindung über ihre Axone

Nunmehr fehlt noch eine geeignete Zuordnungsgleichung für f , um eine Struktur wie in Abb. 2.10 zu betreiben. Dazu wird erst einmal angesetzt, dass alle mit den Verbindungsgewichten multiplizierten Eingaben x_i aufsummiert werden, um eine Spaltensumme s_j zu bestimmen ($i, j = 1, 2, \dots, n$)

$$s_j = \sum_{i=1}^n w_{ij} \cdot x_i ,$$

so dass sich das Ergebnis für y_j mit der Zuordnung f danach ergibt durch

$$y_j = f(s_j) .$$

Kurzschreibweise für
die Neuronenabfrage

Allgemein lässt sich das gesamte Ergebnis η auch kurz in der Form

$$\eta = f(\mathfrak{x} \cdot V)$$

notieren und berechnen.¹⁸ Nehmen wir an, die gesuchte Zuordnungsvorschrift sei einfach nur $f(s) = s$. Wird dann beispielsweise folgende durch die Matrix V festgelegte Verbindung zweier Nervenzellen

B2

$$V = \begin{pmatrix} 0,5 & 0 \\ -0,1 & 7 \end{pmatrix}$$

mit den Eingaben $\mathfrak{x} = (3 \quad -2)$ versorgt, dann ergibt sich $y_1 = 1,7$ und $y_2 = -14$. Setzt man dieses $\eta = (1,7 \quad -14)$ gemäß Abb. 2.10 nun wieder als Eingabe $\mathfrak{x}$ ein, dann liefert

$$\eta = f(f(\mathfrak{x} \cdot V) \cdot V)$$

die neuen Werte $y_1 = 2,25$ und $y_2 = -98$. Wiederholt man diesen Rechenschritt, dann erhält man $y_1 = 10,925$ und $y_2 = -686$. Bei jeder Wiedervorlage der Werte werden die Ergebnisse betragsmäßig größer, was auf einen Mangel in der Modellierung hindeutet, denn eine Nervenzelle wird keine beliebig großen Werte weiterleiten oder empfangen können.

Tatsächlich liefert eine Nervenzelle über ihr Axon nur dann einen Wert y_j größer als null, wenn die Summe der Eingangswerte x_i einen gewissen **Schwellwert** Θ erreicht oder überschreitet.¹⁹ Folglich kommt die Zuordnungsgleichung

$$f(s) = \begin{cases} 1, & \text{falls } s \geq \Theta \\ 0, & \text{sonst} \end{cases}$$

verbesserte
Zuordnungsgleichung
mit Schwellwert Θ

mit der Spaltensumme $s = \sum_{i=1}^n w_{ij} \cdot x_i$ den natürlichen Gegebenheiten näher.

Mit dieser verbesserten Zuordnungsgleichung und $\Theta = 1$ und

B3

$$V = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

erhält man beispielsweise für die in der Tabelle 2.1 genannten Eingabewerte x_i die angegebenen Ergebnisse, die zeigen, dass dieses einzelne Modellneuron die aussagenlogische UND-Verknüpfung nachbildet:

x_1	x_2	y_1
0	0	0
0	1	0
1	0	0
1	1	1

Tabelle 2.1: Aussagenlogische UND-Verknüpfung

Für $\Theta = 0,5$ und derselben Matrix V ergibt sich die aussagenlogische ODER-Verknüpfung und mit $\Theta = -0,5$ und $V = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$ erhält man die NICHT-Verknüpfung. Mit den Kurzschreibweisen

UND-, ODER- und
NICHT-Verknüpfung
durch einzelne
Modellneuronen

$$V_{\wedge} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}_{1,5}, \quad V_{\vee} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}_{0,5}, \quad V_{\neg} = \begin{pmatrix} -1 \end{pmatrix}_{-0,5}$$

werden die UND-, ODER- und NICHT-Verknüpfung notiert. Der Schwellwert Θ wird als Index an die Matrix gesetzt.

Schwellwert Θ als
Matrixindex

Da alle anderen aussagenlogischen Verknüpfungen durch die drei genannten ausgedrückt werden können, zeigt sich, dass jeder aussagenlogische Term durch eine Kombination dieser drei Arten von Modellneuronen nachgebildet werden kann. Wenn man wie [Palm 1982], S. 26, ein Verhalten als Zuordnung einer Situation zu einer Aktion begreift, und sich sowohl die Situation als auch die Aktion durch Folgen der Werte 0 und 1 darstellen lassen, dann ließe sich jedes Verhalten durch eine Nervenzellstruktur aus den oben beschriebenen Modellneuronen für die NICHT-, UND- und ODER-Verknüpfung erreichen.²⁰

B4

Soll beispielsweise auf die Situation $\mathfrak{x} = (1 \ 0 \ 1)$ mit der Aktion $\mathfrak{y} = (0 \ 1 \ 1 \ 0 \ 1)$ reagiert werden, so lässt sich für y_1 und y_4 , also den beiden Nullen, ein aussagenlogischer Term wie folgt aufstellen

$$\overline{x_1 \wedge \overline{x_2} \wedge x_3} = \overline{x_1} \vee x_2 \vee \overline{x_3}$$

und y_2, y_3 und y_5 erhalten als Wert das dazu aussagenlogische Gegenteil. Der Wert für y_1, y_4 wird dann durch folgende neuronale Anordnung geliefert

$$y_1 = y_4 = (((x_1 \cdot V_-) \cdot x_2) \cdot V_-) \cdot (x_3 \cdot V_-)) \cdot V_- ,$$

wobei die mit dem $\cdot$ bezeichnete Multiplikation die Schwellwertoperation mit einschließt. Mit der Eingabe $\mathfrak{x} = (1 \ 0 \ 1)$ wird nun rechnerisch überprüft:

$$\begin{aligned} y_1 &= (((1 \cdot V_-) \cdot 0) \cdot V_-) \cdot (1 \cdot V_-)) \cdot V_- \\ &= (((0 \cdot V_-) \cdot 0) \cdot V_-) \cdot V_- \\ &= (0 \cdot 0) \cdot V_- \\ &= 0 . \end{aligned}$$

B5

Erweitert man diese Anordnung, so dass es auf weitere Situation-Aktion-Paare, zum Beispiel auf $\mathfrak{x} = (1 \ 1 \ 0)$ mit $\mathfrak{y} = (1 \ 0 \ 0 \ 1 \ 1)$, richtig reagiert, werden die nachzubildenden Terme meist umfangreicher. Hier ist allein schon für den aussagenlogischen Term zu y_1

$$(\overline{x_1} \vee x_2 \vee \overline{x_3}) \vee (x_1 \wedge x_2 \wedge \overline{x_3})$$

die neuronale Anordnung

$$\begin{aligned} y_1 &= (((((x_1 \cdot V_-) \cdot x_2) \cdot V_-) \cdot (x_3 \cdot V_-)) \cdot V_-) \\ &\quad (((x_1 \cdot x_2) \cdot V_-) \cdot (x_3 \cdot V_-)) \cdot V_-) \cdot V_- \end{aligned}$$

zu schaffen und auszuwerten.

zu aufwändige
Konstruktion

Lernverfahren

Frage-Antwort-Paare

Nicht nur, dass auf diese Weise das „Lernen“ zusätzlicher Situation-Aktion-Paare in der Regel einen immer aufwändiger werdenden Aufbau der zugehörigen neuronalen Struktur nach sich zieht, auch Fehlertoleranz und Fähigkeiten zur Mustervervollständigung oder Musterextraktion fehlen dem so entstehenden Nervenzellverbund. Daher werden andere Möglichkeiten gesucht, Modellneuronen so zusammenzufügen und zu betreiben, dass ihr Verbund zusätzliche nützliche Eigenschaften gewinnt. Dazu werden im Folgenden verschiedene Wege beschrieben, die als **Lernverfahren** bezeichnet werden, um Verbindungsmatrizen für neuronale Netze so mit geeigneten Gewichten zu füllen, dass auf eine vorgelegte Situation $\mathfrak{x}$ mit der gewünschten Aktion $\mathfrak{y}$ reagiert wird. Situation-Aktion-Paare werden auch **Frage-Antwort-Paare** genannt.

2.2 Aufbau neuronaler Strukturen ohne Lernüberwachung

Durch die Beispiele B4 und B5 wird zudem deutlich, dass Matrixoperationen, also Abfragen der Modellneuronen, nicht immer in nur einem Schritt,

sondern oft auch nacheinander ausgeführt werden müssen, um zum Ergebnis zu gelangen. So könnten zwei Verbindungsmatrizen V_1 und V_2 mit

$$V_1 = \begin{pmatrix} 0,2 & 0 & 0 & 0 & 0 \\ 0 & 0,7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}_1, \quad V_2 = \begin{pmatrix} 0 & 0 & -1 & 5 & 3 \\ 0 & 0 & -2 & 3 & 0,6 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}_1 \quad \boxed{\text{B6}}$$

für einen Verbund von fünf, vollständig miteinander verknüpften Modellneuronen eine zweistufige Abfrage bestimmen, indem zum Beispiel das Ergebnis einer Abfrage mit $\mathfrak{x} = (4 \ 7 \ 0 \ 0 \ 0)$ durch

$$\mathfrak{y} = ((4 \ 7 \ 0 \ 0 \ 0) \cdot V_1) \cdot V_2$$

berechnet wird. Hier wird also mit $x_1 = 4$ und $x_2 = 7$ angefragt und das Resultat befindet sich zum Schluss in $y_3 = 0, y_4 = 1, y_5 = 0$. Die beiden Modellneuronen, die im ersten Schritt die Eingaben für den Verbund erhalten und verarbeiten, sind von den drei Modellneuronen, die im zweiten Schritt die Ausgabewerte liefern, zeitlich so getrennt, dass man dieses Nacheinander durch zwei kleinere Strukturen mit

$$V_1 = \begin{pmatrix} 0,2 & 0 \\ 0 & 0,7 \end{pmatrix}_1, \quad V_2 = \begin{pmatrix} -1 & 5 & 3 \\ -2 & 3 & 0,6 \end{pmatrix}_1$$

erzeugen könnte. Die ursprüngliche Struktur lässt sich hier also in zwei **Schichten** zerlegen. Die Abfrage

$$\mathfrak{y} = ((4 \ 7) \cdot V_1) \cdot V_2$$

erbringt das gleiche Ergebnis wie oben.

Ein Verbund aus Modellneuronen, welcher sich in Schichten einteilen lässt, heißt **Perzeptron**.²¹ Diejenigen Modellneuronen, die die Eingabewerte erhalten, werden **Eingabeschicht** genannt,²² diejenigen, über die die Ausgabewerte geliefert werden, heißen **Ausgabeschicht**. Ein Perzeptron besteht mindestens aus einer Eingabe- und einer Ausgabeschicht, kann aber auch **Zwischenschichten** besitzen, wie das Beispiel B7 des durch nachstehende Verbindungsmatrizen V_1, V_2, V_3 gegebenen Perzeptrons zeigt:

$$V_1 = \begin{pmatrix} -1 & 0 \\ 0 & 2 \end{pmatrix}_1, \quad V_2 = \begin{pmatrix} 4 & 1 & -2 \\ 2 & -3 & 6 \end{pmatrix}_1, \quad V_3 = \begin{pmatrix} 3 & -5 \\ -1 & 2 \\ 9 & 2 \end{pmatrix}_1. \quad \boxed{\text{B7}}$$

Fragt man das dieses Perzeptron beispielsweise mit $\mathfrak{x} = (-2 \ 7)$ ab, so errechnet sich $\mathfrak{y} = ((\mathfrak{x} \cdot V_1) \cdot V_2) \cdot V_3 = (1 \ 0)$.²³

Die Darstellung eines Perzeptrons durch Verbindungsmatrizen V_i hat gegenüber der an anderer Stelle üblichen Darstellung neuronaler Strukturen durch Graphen wie in Abb. 2.11 den Vorteil, dass der Leser mit Hilfe der Matrizenrechnung und bei komplexeren Aufgaben gegebenenfalls durch den Einsatz eines Computeralgebrasystems (CAS) die Beispiele dieses Buches nachrechnen als auch eigene Versuche zur Funktionsweise neuronaler Netze vornehmen kann. Das schließt Versuche zu den in den kommenden Abschnitten vorgestellten Lernverfahren und zu Assoziativmatrizen ein. In den Anmerkungen

Schichten

Perzeptron mit
Eingabe-, Ausgabe-
schicht

Zwischenschichten

Matrizendarstellung
und CAS

zu diesem Kapitel werden entsprechende Hinweise für das Computeralgebra-system *Maxima* gegeben.²⁴

Die Eingabeschicht besteht in [B7] aus den Modellneuronen für die Ausgaben y_1, y_2 , die Zwischenschicht aus denjenigen für y_3, y_4, y_5 und die Ausgabeschicht aus denen für y_6, y_7 . Die zugehörige graphische Darstellung (s. Abb. 2.11) des Perzeptrons macht diese Schichtung deutlicher. Ein Perzeptron, welches Zwischenschichten besitzt, wird **mehrlagig** genannt.²⁵

mehrlagiges
Perzeptron

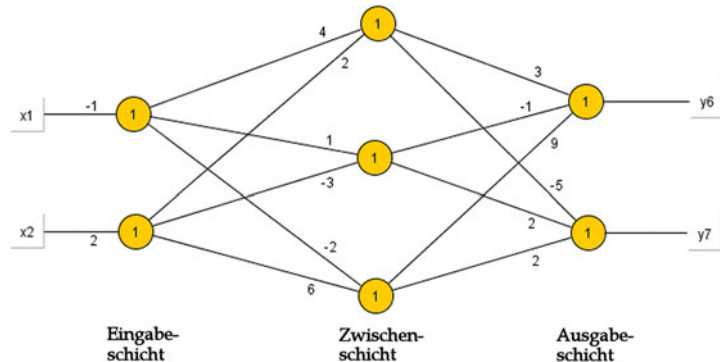


Abb. 2.11: Graphische Darstellung eines Perzeptrons

Ein Perzeptron kann vom Anwender für eine gegebene Aufgabe **konstruiert** werden, was bedeutet, dass er die Anzahl an Schichten und die Einträge in die zugehörigen Matrizen vorgibt.²⁶ Eine anfängliche Festlegung auf die Matrixeinträge muss jedoch im Allgemeinen für neuronale Netze nicht unbedingt vorgenommen werden. Es ist zu Beginn auch eine Belegung der Verbindungsmatrizen mit rein zufälligen Werten möglich, wie das nächste Kapitel 2.3 zu neuronalen Netzen mit **trainierenden** Lernverfahren zeigen wird.

konstruktives
Lernverfahren

Zur Erläuterung des konstruktiven Lernverfahrens bei einem Perzeptron sei die Aufgabe [B8] vorgelegt, in einem Foto dunkle Pixel²⁷ zu erkennen, die wie in Abb. 2.12 sich genau in der Mitte des beobachteten Fotoausschnitts befinden. Tabelle 2.13 zeigt die zugehörigen Helligkeitswerte der Pixel in Zahlen zwischen 0 und 255 an.

[B8]



161	205	161	162	160
166	163	164	164	159
156	156	51	155	157
155	149	159	157	160
153	158	161	157	159

Abb. 2.12: Fotoausschnitt 5 x 5 Pixel

Abb. 2.13: Helligkeitswerte Fotoausschnitt

Zur Lösung der Aufgabe wird ein Perzeptron aufgebaut, dessen Eingabeschicht aus neun Modellneuronen $y_1, y_2, \dots, y_9$ und dessen Ausgabeschicht aus einem einzelnen Modellneuron y_{10} besteht. Die Eingabeschicht wird mit den neun Helligkeitswerten versorgt, die sich in einer beliebigen Umgebung von 3×3 Pixeln auf dem Foto befinden. Die Eingabeschicht hat festzustellen, welche der Pixel als hell und welche als dunkel gelten sollen. Die Ausgabeschicht entscheidet dann, ob sich in der Mitte der jeweils betrachteten 3×3 Pixel ein dunkles Pixel befindet.

Die Matrizen V_1 , V_2 der Eingabe- und Ausgabeschicht, die hier diese Aufgabe erfüllen, haben eine recht einfache Gestalt:

$$V_1 = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}_{-128} \quad V_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}_1.$$

Fragt man nun mit den Helligkeitswerten der linken, oberen 3×3 Pixel ab, so liefert

$$\eta = ((161 \ 205 \ 161 \ 166 \ 163 \ 164 \ 156 \ 156 \ 51) \cdot V_1) \cdot V_2$$

für das Ausgabeneuron den Wert $y_{10} = 0$. Dort befindet sich also kein dunkles Pixel in der Mitte. Fragt man hingegen mit den Werten des mittleren 3×3 -Pixelfeldes ab, so ergibt sich für das Ausgabeneuron

$$\eta = ((163 \ 164 \ 164 \ 156 \ 51 \ 155 \ 149 \ 159 \ 157) \cdot V_1) \cdot V_2$$

der Wert $y_{10} = 1$. Das Ausgabeneuron teilt also mit, dass sich in der Mitte der betrachteten 3×3 Pixel eines der gesuchten dunklen Pixel befindet, womit die gestellte Aufgabe gelöst ist. Der Schwellwert von V_1 legt dabei fest, was als „dunkel“ gilt.

Erweitert man die Aufgabe in B8 und verlangt von dem Perzeptron zusätzlich, ein dunkles Pixel nur zu melden, wenn es von hellen Pixeln umgeben ist, so wird man eine Schicht verändern oder eine neue einfügen müssen. In B8 genügt es, V_2 wie nebenstehend zu ändern. Doch 1969 untersuchten Marvin Minsky und Seymour Papert Aufgabenstellungen ausführlich, an denen ein Perzeptron prinzipiell scheitert. Eines der Probleme, das XOR-Problem, welches ein einlagiges Perzeptron nicht bewältigen kann, löste Frank Rosenblatt durch Einfügen einer Zwischenschicht.²⁸ Minsky und Papert vermuteten, dass die Leistung eines Gehirns nicht durch ein einförmiges Netzwerk gebildet wird, sondern durch den Verbund kleinerer, spezialisierter Netzwerke.²⁹

$$V_2 = \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}_1 \quad \text{XOR-Problem}$$

Das Beispiel [B8] mag einerseits deutlich machen, dass man die nötigen Einfälle haben muss, um mit seinem Perzeptron zum Ziel zu kommen, aber auch, dass hier jedes der beteiligten Modellneuronen seinen Zweck exakt erfüllen muss, was dem biologischen Vorbild kaum entspricht.

Verfahren für binär
vorliegende Daten

Für den Fall, dass die vom Perzeptron zu verarbeitenden Daten in binärer Form wie in Beispiel [B4] vorliegen, lässt sich ein Verfahren angeben, um eine geeignete Matrix V aufzustellen. Die beiden einzutragenden Paare aus [B4] sind $\mathbf{r}_1 = (1 \ 0 \ 1) \rightarrow \boldsymbol{\eta}_1 = (0 \ 1 \ 1 \ 0 \ 1)$ und $\mathbf{r}_2 = (1 \ 1 \ 0) \rightarrow \boldsymbol{\eta}_2 = (1 \ 0 \ 0 \ 1 \ 1)$. Da die Daten binär vorliegen und drei Modellneuronen für die Eingaben und fünf für die Ausgaben vorzusehen sind, wird für die Eingabeschicht als Verbindungsmatrix V_1 eine 3x3-Einheitsmatrix mit dem Schwellwert $\Theta = 1$ gewählt. Für die Ausgabeschicht werden in eine 3x5-Nullmatrix zuerst die Werte von $\boldsymbol{\eta}_1$ in diejenigen Zeilen hineinkopiert, für die $\mathbf{r}_1$ eine Eins enthält, hier also in die erste und dritte Zeile. Nach dem Eintragen des ersten Frage-Antwort-Paars ergibt sich damit für V_2 die Matrix

$$V_2 = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Danach wird $\boldsymbol{\eta}_2$ entsprechend eingetragen, wobei eine bereits vorhandene Eins beim Hineinkopieren erhalten bleibt, also nicht etwa durch eine Zwei oder Null ersetzt wird. Als Schwellwert wird hier $\Theta = 2$ gewählt, da die Fragen je zwei Einsen enthalten. Dadurch wird V_2 zu

$$V_2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}_2.$$

Die Rechnungen

$$\begin{aligned} (1 \ 0 \ 1) \cdot V_2 &= (0 \ 1 \ 1 \ 0 \ 1) \\ (1 \ 1 \ 0) \cdot V_2 &= (1 \ 0 \ 0 \ 1 \ 1) \end{aligned}$$

bestätigen, dass das Perzeptron wie gewünscht arbeitet. Die Verbindungsmatrix V_1 der Eingabeschicht kann bei dieser Abfrage ausgelassen werden, da sie als Einheitsmatrix angesetzt wurde. Trägt man weitere Frage-Antwort-Paare in V_2 ein, so muss man zunehmend damit rechnen, dass unerwartete Antworten gegeben werden, da die Matrix zu viele Einsen eingetragen bekommt, um noch auf jede Frage korrekt reagieren zu können.

Problem der „falschen“
Einsen

L-Lernregel

Das soeben vorgestellte Verfahren zum Füllen einer Matrix V mit geeigneten Verbindungsgewichten, die **L-Lernregel**, geht auf [Willshaw et al. 1969] zurück³⁰ und wird im Kapitel 2.5 als eine der Lernregeln für Assoziativmatrizen verwendet werden.³¹

Hopfield-Netz

Die Gliederung des Perzeptrons in Schichten findet wie oben gezeigt seinen Ausdruck in der Beschreibung der Struktur und der Verbindungsstärken durch mehrere, nacheinander anzuwendende Matrizen V_i . Die Modellneuronen einer Schicht nehmen auf diejenigen der vorangehenden Schicht keinen Einfluss. Bei **Hopfield-Netzen** lässt man das nun aber zu, stellt folglich nur eine einzige Matrix $V = (w_{ij})_{i,j=1,\dots,n}$ für n vollständig miteinander verbundene Modellneuronen auf, schränkt jedoch ein, dass kein Modellneuron mit

sich selbst verbunden und dass ein Modellneuron a mit einem Modellneuron b genau so stark verknüpft wird, wie das Modellneuron b mit dem Modellneuron a . Für die Verbindungsgewichte in Zeile i und Spalte j gilt also $w_{ii} = 0$ und $w_{ij} = w_{ji}$.

Bei den in ein Hopfield-Netz einzutragenden Frage-Antwort-Paaren ist die Frage gleich der Antwort. In solchen Fällen werden im Folgenden die Fragen und Antworten **Muster** genannt. Entspricht beim Abfragen eine gestellte Frage nicht exakt dem gelernten Muster, bezeichnet man diese Frage als **gestörtes** Muster.

Muster

John J. Hopfield³² gibt in [Hopfield 1982] zur Bestimmung der Verbindungsgewichte für n abzuspeichernde Muster $m_1, m_2, m_3, \dots, m_n$ mit $i \neq j$ an:³³

Hopfield-Lernregel

$$w_{ij} = \sum_{k=1}^n (2 \cdot (m_k)_i - 1) \cdot (2 \cdot (m_k)_j - 1)$$

Sollten zum Beispiel folgende drei Muster m_1, m_2, m_3 von einem Hopfield-Netz zu speichern sein

B9

$$m_1 = (0 \ 0 \ 1 \ 1 \ 0), \quad m_2 = (0 \ 1 \ 1 \ 0 \ 0), \quad m_3 = (1 \ 0 \ 0 \ 0 \ 1),$$

dann ergibt sich durch Anwendung der von Hopfield genannten Regel³⁴ die Verbindungsmatrix V mit

$$V = \begin{pmatrix} 0 & -1 & -3 & -1 & 3 \\ -1 & 0 & 1 & -1 & -1 \\ -3 & 1 & 0 & 1 & -3 \\ -1 & -1 & 1 & 0 & -1 \\ 3 & -1 & -3 & -1 & 0 \end{pmatrix}_0.$$

Die Rechnungen $m_1 \cdot V = m_1$, $m_2 \cdot V = m_2$ und $m_3 \cdot V = m_3$ bestätigen, dass das in B9 aufgestellte Hopfield-Netz die drei gespeicherten Muster erkennt.³⁵ Fragt man mit einem nicht gelernten Muster ab, beispielsweise mit $(0 \ 0 \ 0 \ 0 \ 1)$, dann antwortet das Netz mit $(1 \ 0 \ 0 \ 0 \ 0)$, fragt man mit $(0 \ 0 \ 0 \ 1 \ 0)$ erhält man die Antwort $(0 \ 0 \ 1 \ 0 \ 0)$. Legt man diese Antworten wiederum als Fragen vor, lässt sich erkennen, dass das Netz im Unterschied zum Verhalten bei gelernten Mustern hier nicht zu einer festen Antwort gelangt. Dass ein Hopfield-Netz jedoch auch in der Lage ist, bei Anfragen mit nicht gelernten Mustern zu einer festen Antwort zu kommen, soll Beispiel B10 verdeutlichen.

MAX-Beispiel

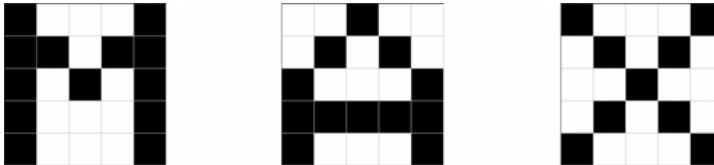


Abb. 2.14: Drei zu lernende Pixelmuster 'M', 'A', 'X' für ein Hopfield-Netz

Ein Hopfield-Netz soll die drei Pixelmuster aus Abb. 2.14 lernen. Dazu werden die dunklen Pixel mit dem Wert 1 und die hellen Pixel mit dem Wert 0 in die Muster m_1, m_2 und m_3 eingetragen.

B10

Zeile für Zeile werden die Pixelwerte nacheinander notiert und es ergeben sich somit als Werte für die drei Muster

$$m_1 = (1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1)$$

$$m_2 = (0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1)$$

$$m_3 = (1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1)$$

und mit der Hopfield-Lernregel bestimmen sich für die Gewichte der Verbindungsmatrix V die in Abb. 2.15 gezeigten Werte. Der Schwellwert wird auf $\Theta = 0$ gesetzt.

0	-1	-3	-1	3	1	1	-1	1	1	-1	-1	3	-1	-1	-1	-3	-1	-1	1	-1	-1	-1	1
-1	0	1	3	-1	1	-3	3	-3	1	-1	3	-1	3	-1	-1	-1	1	-1	-1	-3	3	3	-3
-3	1	0	1	-3	-1	1	1	-1	-1	1	1	-3	1	1	1	1	3	1	1	-1	1	1	-1
-1	3	1	0	-1	1	-3	3	-3	1	-1	3	-1	3	-1	-1	-1	1	-1	-1	-3	3	3	-3
3	-1	-3	-1	0	1	1	-1	1	1	-1	-1	3	-1	-1	-1	-1	-3	-1	-1	1	-1	-1	1
1	1	-1	1	1	0	-1	1	-1	3	1	1	1	1	1	1	-3	-1	-3	1	-1	1	1	-1
1	-3	-1	-3	1	-1	0	-3	3	-1	1	-3	1	-3	1	1	1	-1	1	1	3	-3	-3	3
-1	3	1	3	-1	1	-3	0	-3	1	-1	3	-1	3	-1	-1	-1	1	-1	-1	-3	3	3	-3
1	-3	-1	-3	1	-1	3	-3	0	-1	1	-3	1	-3	1	1	1	-1	1	1	3	-3	-3	3
1	1	-1	1	1	3	-1	1	-1	0	1	1	1	1	1	1	-3	-1	-3	1	-1	1	1	-1
-1	-1	1	-1	-1	1	1	-1	1	1	0	-1	-1	-1	3	3	-1	1	-1	3	1	-1	-1	1
-1	3	1	3	-1	1	-3	3	-3	1	-1	0	-1	3	-1	-1	-1	1	-1	-1	-3	3	3	-3
3	-1	-3	-1	3	1	1	-1	1	1	-1	-1	0	-1	-1	-1	-1	-3	-1	-1	1	-1	-1	1
-1	3	1	3	-1	1	-3	3	-3	1	-1	3	-1	0	-1	-1	-1	1	-1	-1	-3	3	3	-3
-1	-1	1	-1	-1	1	1	-1	1	1	3	-1	-1	-1	0	3	-1	1	-1	3	1	-1	-1	1
-1	-1	1	-1	-1	1	1	-1	1	1	3	-1	-1	-1	3	0	-1	1	-1	3	1	-1	-1	1
-1	-1	1	-1	-1	-3	1	-1	1	-3	-1	-1	-1	-1	-1	-1	0	1	3	-1	1	-1	-1	1
-3	1	3	1	-3	-1	-1	1	-1	-1	1	1	-3	1	1	1	1	0	1	1	-1	1	1	-1
-1	-1	1	-1	-1	-3	1	-1	1	-3	-1	-1	-1	-1	-1	-1	3	1	0	-1	1	-1	-1	1
-1	-1	1	-1	-1	1	1	-1	1	1	3	-1	-1	-1	3	3	-1	1	-1	0	1	-1	-1	1
1	-3	-1	-3	1	-1	3	-3	3	-1	1	-3	1	-3	1	1	1	-1	1	1	0	-3	-3	3
-1	3	1	3	-1	1	-3	3	-3	1	-1	3	-1	3	-1	-1	-1	1	-1	-1	-3	0	3	-3
-1	3	1	3	-1	1	-3	3	-3	1	-1	3	-1	3	-1	-1	-1	1	-1	-1	-3	3	0	3
-1	3	1	3	-1	1	-3	3	-3	1	-1	3	-1	3	-1	-1	-1	1	-1	-1	-3	3	3	0
1	-3	-1	-3	1	-1	3	-3	3	-1	1	-3	1	-3	1	1	1	-1	1	1	3	-3	-3	0

Abb. 2.15: Gewichte der Verbindungsmatrix V zum MAX-Beispiel

Die drei Muster aus Abb. 2.14 werden vom durch V gegebenen Hopfield-Netz erkannt, wie die Rechnungen $m_1 \cdot V = m_1$, $m_2 \cdot V = m_2$ und $m_3 \cdot V = M_3$ zeigen.

Am Beispiel B10 lässt sich zudem die Eigenschaft von Hopfield-Netzen sichtbar machen, Muster vervollständigen zu können. Man frage V dazu mit den in Teilen veränderten Mustern aus Abb. 2.16 ab.

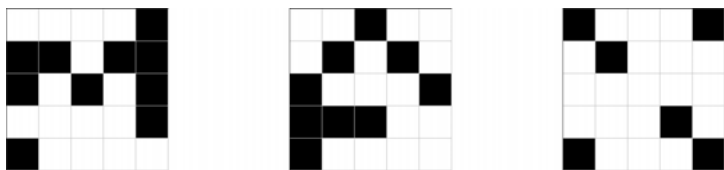


Abb. 2.16: Drei gestörte Pixelmuster zur Abfrage des Hopfield-Netzes

Die drei vorgelegten Muster aus Abb. 2.16, in denen im Unterschied zu den gelernten Mustern m_1 , m_2 und m_3 einige dunkle Pixel fehlen, werden bei einer Abfrage korrekt zu den gelernten Mustern ergänzt.

Doch treten beim vorgestellten Verfahren für Hopfield-Netze auch Fälle auf, in denen keine Mustererkennung gelingt. Die Abb. 2.17 zeigt drei Muster, die wie oben in ein Hopfield-Netz eingetragen werden.

ABC-Beispiel

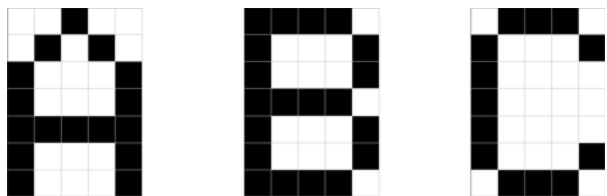


Abb. 2.17: Drei zu lernende Pixelmuster 'A', 'B', 'C' für ein Hopfield-Netz

Fragt man mit den ersten beiden Mustern aus Abb. 2.17 das eben erzeugte Hopfield-Netz ab, so erhält man die erwarteten Ergebnisse für 'A' und 'B'. Doch legt man dem Hopfield-Netz das dritte Muster vor, so lässt die Antwort ein Zeichen erscheinen, welches einem 'B' ähnelt. Fragt man mit dieser Antwort nochmals ab, erscheint endgültig das 'B'-Muster, wie in Abb. 2.18 veranschaulicht. Alle dunklen Pixel des 'C'-Musters sind auch im 'B'-Muster enthalten, so dass es zu dieser unerwünschten Antwort kommt.

B11

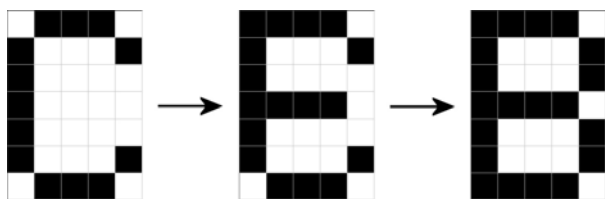


Abb. 2.18: Die Abfrage mit dem 'C'-Muster führt letztlich zu einem 'B'-Muster.

Lernt man nun aber im Beispiel B11 außer den drei Mustern für 'A', 'B' und 'C' als viertes noch ein 'D'-Muster wie in Abb. 2.19, dann endet die Abfrage des Hopfield-Netzes mit dem 'C'-Muster zuletzt bei einem Muster, welches nicht gelernt wurde und eine Mischung aus dem 'C'-Muster und dem 'D'-Muster zu sein scheint.

Für weitere Versuche mit dem Hopfield-Netz bieten sich die Übungen 2.4 an, in denen unter anderem geprüft wird, wie sich das Netz bei Abfragen mit Mustern verhält, die sich von den gelernten Mustern durch hinzugefügte dunkle Pixel unterscheiden oder in denen sowohl dunkle Pixel des ursprünglichen Musters gelöscht als auch neue hinzugefügt wurden. Damit wird auf die Fähigkeiten der Hopfield-Netze zur Musterergänzung und Musterextraktion eingegangen.

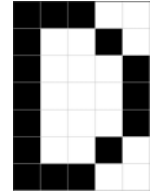


Abb. 2.19:
'D'-Muster

Kohonenkarten, SOM

Weitere Verfahren zum Aufbau einer neuronalen Struktur ohne Lernüberwachung wurden von Teuvo Kohonen³⁶ zum Bilden selbstorganisierender Karten, auch Kohonenkarten genannt,³⁷ eingeführt. In ihnen werden linear oder flächig angeordnete Neuronen, die jeweils durch ein n -Tupel an Werten modelliert sind, wiederholt Eingabereizen ausgesetzt, die ebenfalls als n -Tupel angelegt werden. Mit Hilfe einer Kohonen-Lernregel³⁸ werden die Werte der Neuronen so verändert, dass Reize, die ähnlich einem der Eingabereize sind, in dessen Nähe zu liegen kommen. Es ist anzumerken, dass die Lernalgorithmen für Kohonenkarten weder überprüfen noch sich daran orientieren, ob das Lernen erfolgreich ist. Daher wird es zu den Verfahren ohne Lernüberwachung gerechnet. Als solches eignet es sich für Anwendungen, bei denen die gewünschten Antworten nicht von vorneherein feststehen oder bekannt sind, sondern sich erst im Laufe des Lernprozesses ergeben. Zur Lernzeit kann sich dieses neuronale Netz dank seiner Fähigkeit zur Selbstorganisation an geänderte Lernziele oder neue von der Sensorik gelieferte Werte anpassen.³⁹

B12

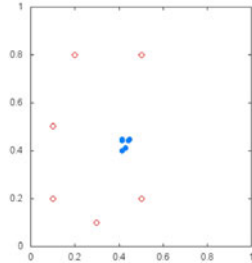


Abb. 2.20: Startsituation

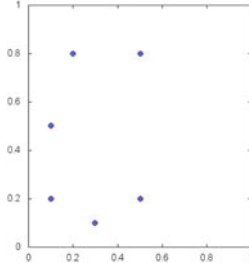


Abb. 2.21: Endsituation

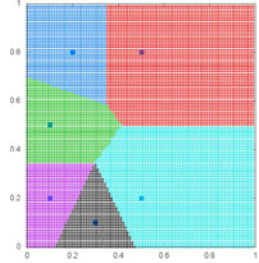


Abb. 2.22: Abfrage

In [B12] wurden einem neuronalen Netz aus 3×2 Modellneuronen sechs Eingabereize wiederholt in zufälliger Abfolge vorgelegt. Jedes dieser Modellneuronen und jeder Eingabereiz besteht aus zwei Werten, nämlich aus den Koordinaten eines Ortes der Ebene. Die Abb. 2.20 gibt die in den 3×2 Neuronen und in den Eingabereizen abgelegten Orte wieder. Anfangs „befinden“ sich die Neuronen an zufällig gewählten Orten in der Mitte des Feldes, markiert als dunkle Flecken in Abb. 2.20. Die Eingabereize „liegen“ hier, als kleine Kreise dargestellt, weiter außen. Nach einigen tausend Lernschritten haben sich die Werte der Neuronen den Eingabereizen angepasst (s. Abb. 2.21). Die Flecken liegen jetzt innerhalb der Kreise. Fragt man das Netz nun mit Koordinaten von den Orten der Ebene ab, ergibt sich die in Abb. 2.22 gezeigte Karte. Ein im Detail erläutertes *Maxima*-Skript für [B12] ist in den Anmerkungen am Ende dieses Kapitels nachzulesen.⁴⁰

2.3 Aufbau neuronaler Strukturen mit Lernüberwachung

Bei den weiter oben vorgestellten Verfahren wird die Verbindungsmatrix V durch einmaliges Eintragen der Datenpaare oder Muster in festgelegter Weise gefüllt. Bei den selbstorganisierenden Karten nach Kohonen diene ein wiederholtes Vorlegen von Eingabereizen dem Entstehen der Einträge von V . Es gibt demgegenüber jedoch auch Verfahren, bei denen die Verbindungsgewichte nach einem anfänglichen Eintragen der Datenpaare in weiteren Durchgängen anhand der auftretenden Fehler solange verändert werden, bis die durch diese Schritte angepasste Matrix die gewünschten Ausgaben liefert. Dieses Vorgehen wird **Training** genannt.

Trainingsverfahren für neuronale Netze

Mit einer auf den Erkenntnissen von Donald Hebb fußenden Regel lässt sich eine schrittweise Veränderung der Verbindungsgewichte mit dem Ziel vornehmen, dass das neuronale Netz letztlich auf jede der gelernten Frage-Antwort-Paare korrekt reagiert. Die Änderung ΔV der Verbindungsmatrix V bestimmt sich mit dieser Regel durch

Hebbsche Lernregel

$$\Delta V = \mathfrak{f}^T \cdot \eta (\mathbf{a}_{soll} - \mathbf{a}_{ist}) \quad ,$$

wobei mit η eine geeignet zu wählende, sogenannte **Lernrate** bezeichnet wird, mit $\mathfrak{f}$ eine Frage und mit $\mathbf{a}_{soll}$ die gewünschte Antwort gemeint ist. Die Antwort $\mathbf{a}_{ist}$, die das Netz aktuell liefert, wird mit der Antwort $\mathbf{a}_{soll}$ verglichen. Sollte sich $\mathbf{a}_{ist}$ von $\mathbf{a}_{soll}$ nicht unterscheiden, dann ist ΔV die Nullmatrix und folglich wäre V nicht zu verändern. Im anderen Falle liefert ΔV diejenigen Werte Δw_{ij} , um die die derzeitigen Einträge der Verbindungsmatrix V zu ändern sind: $V_{neu} = V_{alt} + \Delta V$.

Lernrate η

Seien beispielsweise folgende drei Frage-Antwort-Paare von einem neuronalen Netz zu lernen:

B13

$$\begin{aligned} \mathfrak{f}_1 &= (1 \ 0 \ 1 \ 0) \rightarrow \mathbf{a}_1 = (1 \ 1 \ 0 \ 1) \\ \mathfrak{f}_2 &= (1 \ 1 \ 0 \ 0) \rightarrow \mathbf{a}_2 = (1 \ 0 \ 1 \ 1) \\ \mathfrak{f}_3 &= (0 \ 1 \ 1 \ 1) \rightarrow \mathbf{a}_3 = (0 \ 0 \ 1 \ 1) \quad . \end{aligned}$$

Dann lässt sich die Verbindungsmatrix V gemäß der Hebbschen Einsicht, dass sich neuronale Verbindungen umso mehr verstärken, je öfter sie genutzt werden, bei n Frage-Antwort-Paaren zu Beginn wie folgt ansetzen:

$$V = \sum_{k=1}^n \mathfrak{f}_k^T \cdot \mathbf{a}_k \quad .$$

Für B13 ergibt sich dadurch für V die Rechnung⁴¹

$$\begin{aligned} V &= \mathfrak{f}_1^T \cdot \mathbf{a}_1 + \mathfrak{f}_2^T \cdot \mathbf{a}_2 + \mathfrak{f}_3^T \cdot \mathbf{a}_3 \\ &= \begin{pmatrix} 2 & 1 & 1 & 2 \\ 1 & 0 & 2 & 2 \\ 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{pmatrix}_0 \quad . \end{aligned}$$

Da nun aber alle Abfragen nicht das gewünschte Ergebnis erzielen, es ergibt sich nämlich für alle Fragen $\mathbf{f}_i \cdot V = (1 \ 1 \ 1 \ 1)$, werden die Einträge der Verbindungsmatrix geändert und ΔV wie oben beschrieben berechnet. Für das erste Frage-Antwort-Paar $(\mathbf{f}_1, \mathbf{a}_1)$ folgt mit der Lernrate $\eta = \frac{1}{2}$

$$\begin{aligned}\Delta V &= \mathbf{f}_1^T \cdot \eta (\mathbf{a}_1 - (1 \ 1 \ 1 \ 1)) \\ &= \begin{pmatrix} 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}_0\end{aligned}$$

und

$$\begin{aligned}V_{neu} &= V_{alt} + \Delta V \\ &= \begin{pmatrix} 2 & 1 & \frac{1}{2} & 2 \\ 1 & 0 & 2 & 2 \\ 1 & 1 & \frac{1}{2} & 2 \\ 0 & 0 & 1 & 1 \end{pmatrix}_0\end{aligned}$$

Führt man diese Schritte anschließend noch mit den anderen beiden Frage-Antwort-Paaren durch,⁴² so entsteht

$$V = \begin{pmatrix} 2 & \frac{1}{2} & \frac{1}{2} & 2 \\ \frac{1}{2} & -1 & 2 & 2 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 2 \\ -\frac{1}{2} & -\frac{1}{2} & 1 & 1 \end{pmatrix}_0$$

und das Netz antwortet immerhin schon auf die Frage $\mathbf{f}_2$ richtig. Nach einem weiteren Korrekturlauf mit den drei Frage-Antwort-Paaren erhält man

$$V = \begin{pmatrix} 2 & \frac{1}{2} & 0 & 2 \\ 0 & -1 & 2 & 2 \\ 0 & \frac{1}{2} & 0 & 2 \\ -1 & -\frac{1}{2} & 1 & 1 \end{pmatrix}_0$$

und das Netz reagiert wie erhofft auf alle Fragen $\mathbf{f}_i$ mit der zugehörigen Antwort $\mathbf{a}_i$. Wäre eine andere Lernrate η gewählt worden, so hätte das Einfluss auf das Änderungsverfahren genommen. Für $\eta = 1$ käme man beispielsweise in [B13] mit weniger Schritten zum Ziel.

Backpropagation-
Lernregel

Eine weitere, sehr bekannte Lernregel für das Training einer neuronalen Struktur ist die **Backpropagation-Lernregel**. Auch wenn das Verfahren im Kern wie oben in diesem Kapitel durch $V_{neu} = V_{alt} + \Delta V$ beschrieben werden kann, so unterscheidet es sich jedoch unter anderem von der Hebbischen Lernregel wesentlich darin, dass es stets auf mehrschichtige Netze angewandt wird und dass kein Schwellwert angegeben wird, bezüglich dessen ein Neuron entweder eine 1 oder eine 0 ausgibt, sondern dass eine **Ausgangsfunktion** f den Ausgabewert der Neuronen bestimmt. Der Einfachheit halber nehmen wir dazu im Folgenden an, dass f für alle Neuronen dieselbe Gleichung

Ausgangsfunktion f

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

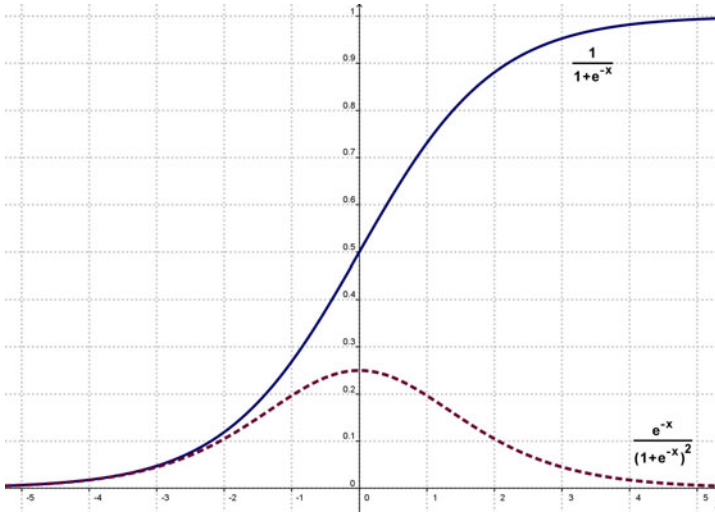


Abb. 2.23: Graphen der Ausgangsfunktion f und ihrer Ableitungsfunktion f' für die Backpropagation-Lernregel

besitzt. Die Neuronen antworten also mit Werten zwischen 0 und 1. Der Graph der Ausgangsfunktion f hat die in Abb. 2.23 gezeigte Gestalt.

Wie bei der Hebbischen Lernregel bildet beim Backpropagation-Verfahren die Abweichung $\mathbf{a}_{soll} - \mathbf{a}_{ist}$ die Grundlage für die Korrektur der Verbindungsmatrizen. Hier wird sie aber zusätzlich mit Hilfe von Werten der ersten Ableitung von f gewichtet, wobei sich die Gleichung der Ableitungsfunktion f' in die Form

$$f'(x) = f(x) \cdot (1 - f(x))$$

bringen lässt und f' folglich bei 0 ihren größten Wert nämlich $\frac{1}{4}$ aufweist. Den Graphen von f' gibt Abb. 2.23 gestrichelt wieder.

Die Backpropagation-Lernregel wird zum Training mehrschichtiger neuronaler Strukturen eingesetzt. Es müssen eine Eingabe- und eine Ausgabeschicht und mindestens eine Zwischenschicht vorhanden sein. Nehmen wir an, dass für ein dreischichtiges Netz die Verbindungsmatrizen V_1 und V_2 mit zufällig ermittelten Werten gefüllt wurden.⁴³ Nach Abfrage mit diesen Matrizen über $\mathbf{a}_{ist} = f(f(\mathbf{x} \cdot V_1) \cdot V_2)$ werden aufgrund der Abweichung von $\mathbf{a}_{ist}$ zu $\mathbf{a}_{soll}$ die Verbindungsmatrizen korrigiert. Die Änderung ΔV_2 der Verbindungsmatrix V_2 wird berechnet durch

$$\Delta V_2 = \eta \odot f(\mathbf{x} \cdot V_1)^T \cdot [f'(f(\mathbf{x} \cdot V_1) \cdot V_2) \odot (\mathbf{a}_{soll} - \mathbf{a}_{ist})]$$

mit $(\mathbf{x}, \mathbf{a}_{soll})$ als zu lernendes Frage-Antwort-Paar und η als Lernrate. Die Ausgangsfunktion f beziehungsweise deren Ableitungsfunktion f' wird jeweils auf alle Komponenten der berechneten Tupel angewandt. Das Operatorzeichen $\odot$ steht für die komponentenweise Multiplikation.⁴⁴ Bei der Korrektur der Werte in V_1 greift man wiederum auf die mit Werten der Ableitungsfunktion f' gewichtete Differenz $\mathbf{a}_{soll} - \mathbf{a}_{ist}$ zu:

$$\Delta V_1 = \eta \odot \mathbf{x}^T \cdot (f'(\mathbf{x} \cdot V_1) \odot [f'(f(\mathbf{x} \cdot V_1) \cdot V_2) \odot (\mathbf{a}_{soll} - \mathbf{a}_{ist})] \cdot V_2^T)$$

Abweichung $\mathbf{a}_{soll} - \mathbf{a}_{ist}$

Ableitungsfunktion f'

dreischichtiges Netz

vierschichtiges Netz

Der Tatsache, dass man bei der Korrektur weiter vorn liegender Schichten auf weiter hinten liegende Schichten zugreift, verdankt das Backpropagation-Verfahren seinen Namen.⁴⁵ Die Bildungsweise der Änderungsmatrizen für die Backpropagation-Lernregel wird erkennbar, wenn man die Änderungsmatrizen ΔV_1 , ΔV_2 , ΔV_3 für ein vierschichtiges Netz untersucht:

$$\begin{aligned}\Delta V_3 &= \eta \odot f(f(\mathbf{x} \cdot V_1) \cdot V_2)^T \cdot \\ &\quad [f'(f(f(\mathbf{x} \cdot V_1) \cdot V_2) \cdot V_3) \odot (\mathbf{a}_{soll} - \mathbf{a}_{ist})] \\ \Delta V_2 &= \eta \odot f(\mathbf{x}^T \cdot V_1) \cdot (f'(f(\mathbf{x} \cdot V_1) \cdot V_2) \odot \\ &\quad [f'(f(f(\mathbf{x} \cdot V_1) \cdot V_2) \cdot V_3) \odot (\mathbf{a}_{soll} - \mathbf{a}_{ist})]) \cdot V_3^T \\ \Delta V_1 &= \eta \odot \mathbf{x}^T \cdot (f'(\mathbf{x} \cdot V_1) \odot (f'(f(\mathbf{x} \cdot V_1) \cdot V_2) \odot \\ &\quad [f'(f(f(\mathbf{x} \cdot V_1) \cdot V_2) \cdot V_3) \odot (\mathbf{a}_{soll} - \mathbf{a}_{ist})]) \cdot V_3^T) \cdot V_2^T)\end{aligned}$$

In einem ersten Beispiel sollen folgende Frage-Antwort-Paare in ein dreischichtiges Netz eingetragen werden, welches eine Zwischenschicht mit fünf Neuronen erhält (4-5-3-Netz):

B14

$$\begin{aligned}\mathbf{f}_1 &= (1 \ 0 \ 0 \ 1) \rightarrow \mathbf{a}_1 = (1 \ 0 \ 1) \\ \mathbf{f}_2 &= (1 \ 0 \ 1 \ 0) \rightarrow \mathbf{a}_2 = (0 \ 1 \ 0) \\ \mathbf{f}_3 &= (0 \ 1 \ 0 \ 1) \rightarrow \mathbf{a}_3 = (0 \ 0 \ 1)\end{aligned}$$

Die 4x5-Matrix V_1 und die 5x3-Matrix V_2 werden zu Beginn mit Zufallswerten gefüllt:

$$V_1 = \begin{pmatrix} 0,2 & -0,3 & 0,4 & 0,1 & -0,5 \\ -0,1 & -0,2 & 0,6 & 0,4 & 0,0 \\ 0,2 & -0,3 & 0,3 & 0,1 & -0,1 \\ 0,5 & -0,9 & 0,8 & 0,3 & 0,2 \end{pmatrix} \quad V_2 = \begin{pmatrix} -0,2 & 0,1 & -0,9 \\ 0,6 & 0,0 & 0,7 \\ -0,2 & 0,1 & -0,9 \\ -0,3 & -0,5 & 0,2 \\ 0,8 & 0,4 & -0,7 \end{pmatrix}$$

zufällige Auswahl
eines Frage-Antwort-
Paares bei jedem
Trainingsschritt

Zunächst soll hier im Beispiel derjenige Term berechnet werden, der in den oben genannten Backpropagation-Regeln in eckige Klammern gesetzt wurde, da er wiederholt auftritt: $[f'(f(\mathbf{x} \cdot V_1) \cdot V_2) \odot (\mathbf{a}_{soll} - \mathbf{a}_{ist})]$. Da drei Frage-Antwort-Paare zu lernen sind, wird bei jedem Trainingsschritt eines davon zufällig ausgewählt und für $\mathbf{x}$ und bei $\mathbf{a}_{soll}$ und $\mathbf{a}_{ist}$ eingesetzt. Sei zufällig das erste Frage-Antwort-Paar $(\mathbf{f}_1, \mathbf{a}_1)$ gewählt. Damit ergibt sich:

$$\begin{aligned}\mathbf{a}_{soll} &= \mathbf{a}_1 = (1 \ 0 \ 1) \\ \mathbf{a}_{ist} &= f(f(\mathbf{f}_1 \cdot V_1) \cdot V_2) \approx (0,503 \ 0,504 \ 0,213) \\ \mathbf{a}_{soll} - \mathbf{a}_{ist} &\approx (0,497 \ -0,504 \ 0,787) \\ f(\mathbf{f}_1 \cdot V_1) &\approx (0,668 \ 0,231 \ 0,769 \ 0,599 \ 0,426) \\ f'(f(\mathbf{f}_1 \cdot V_1) \cdot V_2) &\approx (0,250 \ 0,250 \ 0,167) \\ f'(f(\mathbf{f}_1 \cdot V_1) \cdot V_2) \odot (\mathbf{a}_{soll} - \mathbf{a}_{ist}) &\approx (0,124 \ -0,126 \ 0,132)\end{aligned}$$

Dieses Ergebnis wird nunmehr sowohl zur Berechnung von ΔV_1 als auch ΔV_2 genutzt, wobei die Lernrate $\eta = 1$ gesetzt ist. Für ΔV_2 errechnet sich:

$$\eta \odot f(\mathbf{f}_1 \cdot V_1)^T = \begin{pmatrix} 0,668 \\ 0,231 \\ 0,769 \\ 0,599 \\ 0,426 \end{pmatrix}$$

$$\begin{aligned}\Delta V_2 &= \begin{pmatrix} 0,668 \\ 0,231 \\ 0,769 \\ 0,599 \\ 0,426 \end{pmatrix} \cdot (0,124 \quad -0,126 \quad 0,132) \\ &= \begin{pmatrix} 0,0830 & -0,0841 & 0,0881 \\ 0,0288 & -0,0291 & 0,0305 \\ 0,0955 & -0,0968 & 0,101 \\ 0,0744 & -0,0754 & 0,0789 \\ 0,0529 & -0,0536 & 0,0561 \end{pmatrix}\end{aligned}$$

Für ΔV_1 berechnet man nach der oben genannten Regel:

$$\begin{aligned}(0,124 \quad -0,126 \quad 0,132) \cdot V_2^T \\ \approx (-0,156 \quad 0,167 \quad -0,156 \quad 0,0520 \quad -0,0433)\end{aligned}$$

$$\begin{aligned}\Delta V_1 &\approx \eta \odot \mathbf{f}_1^T \cdot (f'(\mathbf{f}_1 \cdot V_1) \odot (-0,156 \quad 0,167 \quad -0,156 \quad 0,0520 \quad -0,0433)) \\ &\approx \begin{pmatrix} -0,0346 & 0,0297 & -0,0278 & 0,0125 & -0,0106 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -0,0346 & 0,0297 & -0,0278 & 0,0125 & -0,0106 \end{pmatrix}\end{aligned}$$

Nun erhält man durch $V_1 = V_1 + \Delta V_1$ und $V_2 = V_2 + \Delta V_2$ die korrigierten Verbindungsmatrizen.

$$\begin{aligned}V_1 &\approx \begin{pmatrix} 0,165 & -0,270 & 0,372 & 0,113 & -0,511 \\ -0,1 & -0,2 & 0,6 & 0,4 & 0,0 \\ 0,2 & -0,3 & 0,3 & 0,1 & -0,1 \\ 0,465 & -0,870 & 0,772 & 0,313 & 0,189 \end{pmatrix} \\ V_2 &\approx \begin{pmatrix} -0,117 & 0,0159 & -0,812 \\ 0,629 & -0,0291 & 0,731 \\ -0,105 & 0,0032 & -0,799 \\ -0,226 & -0,575 & 0,279 \\ 0,853 & 0,346 & -0,644 \end{pmatrix}\end{aligned}$$

Wird nach diesem ersten Trainingsschritt das aktuelle $\mathbf{a}_{ist}^{neu}$ bestimmt und mit dem letzten $\mathbf{a}_{ist}$ verglichen, so erkennt man bereits eine Verbesserung hinsichtlich $\mathbf{a}_{soll}$:

$$\begin{aligned}\mathbf{a}_{ist} &\approx (0,503 \quad 0,504 \quad 0,213) \\ \mathbf{a}_{ist}^{neu} &\approx (0,554 \quad 0,451 \quad 0,257) \\ \mathbf{a}_{soll} &= (1 \quad 0 \quad 1)\end{aligned}$$

Weitere Trainingsschritte mit den drei Frage-Antwort-Paaren führen in befriedigender Näherung zum gewünschten Verhalten. Ein Ergebnis, das nach einigen zehntausend Schritten entstand, lautet:

Trainingsergebnis

$$\begin{aligned}\mathbf{a}_{1ist} &\approx (0,995 \quad 0,00306 \quad 0,997) \\ \mathbf{a}_{1soll} &= (1 \quad 0 \quad 1) \\ \mathbf{a}_{2ist} &\approx (0,00396 \quad 0,996 \quad 0,00445)\end{aligned}$$

$$\begin{aligned}
\mathbf{a}_{2\text{ soll}} &= (0 \ 1 \ 0) \\
\mathbf{a}_{3\text{ ist}} &\approx (0,00404 \ 0,00310 \ 0,997) \\
\mathbf{a}_{3\text{ soll}} &= (0 \ 0 \ 1)
\end{aligned}$$

Da die Werte 0 und 1, die in den Frage-Antwort-Paaren in [B14] stehen, als Funktionswerte der Ausgangsfunktion f nicht auftreten können, sind derartige Trainingsziele dem Backpropagation-Verfahren nicht angemessen. Man hat sich beim Lernen binärer Frage-Antwort-Paare auf diesen Umstand einzulassen und legt gegebenenfalls einen Schwellwert fest, ab dem nach dem Training ein Wert der Ausgangsschicht als 0 oder 1 aufgefasst werden soll.

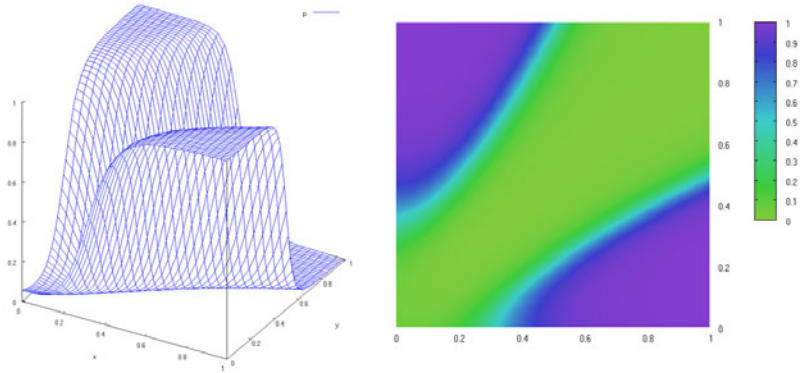


Abb. 2.24: Reaktion der Ausgangsschicht des 2x5x1-Backpropagationnetzes

Das folgende Beispiel [B15] veranschaulicht diese Eigenschaft der Backpropagation-Lernregel, indem zwar in den vier Fragen keine Werte 0 oder 1 auftauchen, wohl aber in den zugeordneten Antworten. Das Netz soll eine Warnlampe einschalten (Wert 1), wenn sich an einem Punkt, deren Koordinaten als Frage dienen, etwas befindet, und sonst nicht (Wert 0). Dazu sollen folgende Frage-Antwort-Paare durch ein dreischichtiges Backpropagation-Netz trainiert werden:

[B15]

$$\begin{aligned}
\mathbf{f}_1 &= (0,2 \ 0,2) \rightarrow \mathbf{a}_1 = (0) \\
\mathbf{f}_2 &= (0,2 \ 0,8) \rightarrow \mathbf{a}_2 = (1) \\
\mathbf{f}_3 &= (0,8 \ 0,2) \rightarrow \mathbf{a}_3 = (1) \\
\mathbf{f}_4 &= (0,8 \ 0,8) \rightarrow \mathbf{a}_4 = (0)
\end{aligned}$$

Die mittlere Schicht soll aus fünf Modellneuronen bestehen, so dass zunächst eine 2x5-Matrix V_1 und eine 5x1-Matrix V_2 mit zufälligen Werten gefüllt werden.

Nach Anwenden der Backpropagation-Lernregel ergeben sich Verbindungsmatrizen V_1 und V_2 , mit denen sich nun auch das Verhalten des Netzes für andere als die gelernten vier Fragen untersuchen lässt. Die Abbildungen in 2.24 zeigen, wie das Netz auf beliebige Fragen $(x \ y)$ reagiert.⁴⁶

Welche guten Eigenschaften diese Backpropagation-Netze für Klassifizierungsprobleme aufweisen, wird in [B15] daran deutlich, dass zu allen Fragen der

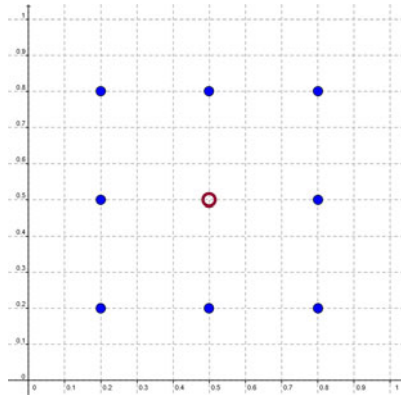


Abb. 2.25: Zu lernende Trainingspunkte für ein 2x15x1-Backpropagationnetz

Abfragefläche eine Antwort gegeben wird, die zu den in der Nähe befindlichen Trainingspunkten passt. Wenn sich etwas in der Nähe eines Trainingspunktes befindet, wird hier darauf mit der Warnlampe so reagiert, wie für diesen Trainingspunkt vorgesehen. Der Abb. 2.24 entnimmt man, dass sich in diesem Fall zwischen den Trainingspunkten links unten und rechts oben ein „Korridor“ ausgebildet hat, in welchem das Netz gleiches Verhalten zeigt. Durch weitere, zu lernende Trainingspunkte lässt sich das Verhalten den jeweiligen Wünschen anpassen.

Fehlertoleranz

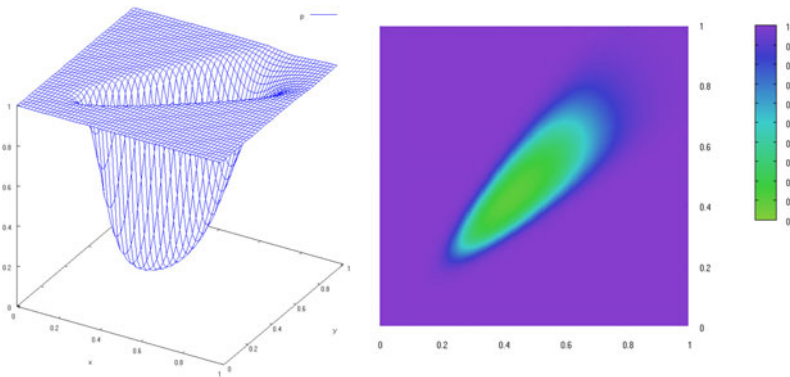


Abb. 2.26: Reaktion der Ausgabeschicht des 2x15x1-Backpropagationnetzes

Für das folgende Beispiel B16 wurden die in Abb. 2.25 angegebenen neun Trainingspunkte gewählt, um ein Backpropagation-Netz mit einer Zwischenschicht aus 15 Neuronen entscheiden lassen zu können, ob sich etwas in der Mitte der Abfragefläche befindet.

Das Ergebnis nach einem Training von gut 150.000 Schritten zeigt Abb. 2.26. Ein einziger Trainingspunkt für eine 0 in der Mitte der Abfragefläche umgeben von acht Trainingspunkten für eine 1 hat genügt, das Netz sicher über ein Drinnen und Draußen oder eine Randlage entscheiden lassen zu können.

B16

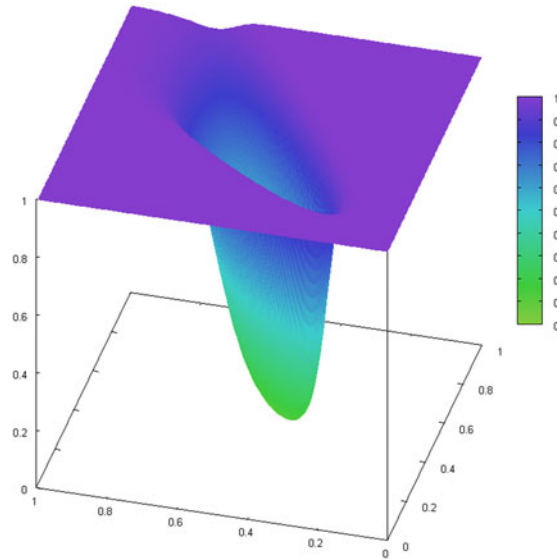


Abb. 2.27: Seitenansicht der Reaktion der Ausgabeschicht des 2x15x1-Netzes

Veranschaulichung
durch Funktionen-
plotter

Besondere Grafikprogramme erlauben es, die Reaktion der Ausgabeschicht von Beispiel B16 nicht nur in der Draufsicht, sondern auch interaktiv und von allen Seiten vor Augen zu führen (s. Abb. 2.26 und Abb. 2.27),⁴⁷ was für die Wahl des Netzaufbaus oder die Auswahl von geeigneten Trainingsdaten hilfreich sein kann.

2.4 Übungen 1

Ü 1.1 (Verbindungsmatrizen)

Es soll auf die Situation $\mathfrak{x} = (0 \ 0 \ 1 \ 1)$ mit der Aktion $\mathfrak{y} = (1 \ 0 \ 0 \ 0)$ reagiert werden. Man gebe wie in Beispiel B4 den dazu nötigen Term mit den Verbindungsmatrizen $V_{\wedge}$, $V_{\vee}$ und V_{-} an.

Ü 1.2 (Perzeptron)

Man konstruiere ein Perzeptron, welches auf folgende Fragen f_i mit den zugehörigen Antworten a_i reagiert.

$$\begin{aligned} f_1 &= (1 \ 0 \ 1 \ 0 \ 0) \rightarrow a_1 = (1 \ 1 \ 0 \ 0) \\ f_2 &= (0 \ 1 \ 0 \ 1 \ 0) \rightarrow a_2 = (0 \ 1 \ 0 \ 1) \\ f_3 &= (0 \ 0 \ 1 \ 0 \ 1) \rightarrow a_3 = (1 \ 0 \ 1 \ 0) \end{aligned}$$

Ü 1.3 (Perzeptron)

Ein mehrlagiges Perzeptron sei durch folgende vier Verbindungsmatrizen beschrieben.

$$\begin{aligned} V_1 &= \begin{pmatrix} -4 & 0 \\ 0 & -3 \end{pmatrix}_0, \quad V_2 = \begin{pmatrix} 4 & 1 \\ 2 & -3 \end{pmatrix}_3, \\ V_3 &= \begin{pmatrix} 3 & -5 & 7 & 1 \\ -1 & 2 & 0 & 9 \end{pmatrix}_{-2}, \quad V_4 = \begin{pmatrix} -3 \\ 6 \\ 7 \\ 1 \end{pmatrix}_5. \end{aligned}$$

- Man berechne, was dieses Perzeptron ausgibt, wenn man es mit $\mathfrak{x} = (-8 \ 5)$ abfragt.
- Man stelle das Perzeptron grafisch dar (s. Abb. 2.11).

Ü 1.4 (Hopfield-Netz)

Man gebe eine Verbindungsmatrix V für ein Hopfield-Netz an, in welche folgende Muster m_i eingetragen werden.

$$\begin{aligned} m_1 &= (1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1) \\ m_2 &= (0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0) \\ m_3 &= (0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0) \end{aligned}$$

- Man prüfe, ob das Netz auf alle drei gelernten Muster m_i korrekt antwortet.
- Nun werden dem Netz gestörte Muster vorgelegt. Zunächst sind es solche, die verglichen mit dem gelernten Muster zusätzliche Einsen enthalten. Man frage V mit $m_1^+ = (1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1)$, mit $m_1^{++} = (1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1)$ und mit $m_2^+ = (0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1)$ ab und prüfe, ob diese Muster dennoch als gelernte Muster erkannt werden.

- c) Es sind Einsen gelöscht worden. Man frage V mit $m_1^- = (1\ 0\ 0\ 1\ 0\ 0\ 0\ 0)$ und mit $m_2^- = (0\ 1\ 1\ 0\ 0\ 0\ 0\ 0)$ ab.
- d) Es sind sowohl Einsen eingefügt als auch gelöscht worden. Man frage V mit $m_1^{+-} = (1\ 0\ 1\ 1\ 0\ 0\ 0\ 0)$ und mit $m_2^{+-} = (0\ 1\ 1\ 0\ 0\ 0\ 0\ 1)$ ab. Sollte nicht sofort die erhoffte Antwort erfolgen, dann lege man dem Netz die jeweilige Antwort wiederholt solange vor, bis es sich für eine feste Antwort entschieden hat.

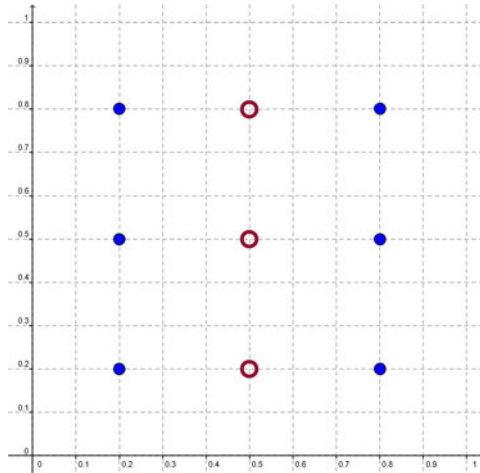
Ü 1.5 (Hebbsche Lernregel)

Eine Verbindungsmatrix soll mit Hilfe der Hebbschen Lernregel und der Lernrate $\eta = 1$ so mit Werten gefüllt werden, dass sie auf die Fragen f_i mit den Antworten a_i reagiert. Es sind die folgenden beiden Frage-Antwort-Paare gegeben:

$$\begin{aligned} f_1 &= (1\ 0\ 1\ 0\ 0) \rightarrow a_1 = (0\ 1\ 1\ 0\ 1) \\ f_2 &= (1\ 1\ 0\ 0\ 0) \rightarrow a_2 = (1\ 0\ 0\ 1\ 1) \end{aligned}$$

Ü 1.6 (Backpropagation-Lernregel)

- a) Man trainiere wie in Beispiel B16 ein neuronales Netz mit der Backpropagation-Lernregel und folgenden Trainingspunkten.



- b) Man trage in ein dreischichtiges Netz folgende Frage-Antwort-Paare ein. Die Zwischenschicht soll aus fünf Neuronen bestehen.

$$\begin{aligned} f_1 &= (1\ 0\ 1) \rightarrow a_1 = (1\ 0\ 0) \\ f_2 &= (0\ 1\ 1) \rightarrow a_2 = (0\ 1\ 0) \\ f_3 &= (1\ 1\ 0) \rightarrow a_3 = (0\ 0\ 1) \end{aligned}$$

2.5 Assoziativmatrizen

2.5.1 Darstellung von Assoziativmatrizen

Assoziativmatrizen sind Verbindungsmatrizen, in denen durch die zugehörigen Lernregeln nur die Werte 0 und 1 als Verbindungsgewichte eingetragen werden. Sie lassen sich je nach Absicht des Erklärenden in mehreren Weisen darstellen. Wird eine neuromathematische Beschreibung wie in den Kapiteln ab 2.1 gewünscht, so rückt man die Matrizendarstellung in den Vordergrund. Nähert man sich dem Thema hingegen aus historischer Sicht über die Steinbuchsche Lernmatrix wie in Kapitel 3.2, wird man auf eine gitterförmig angeordnete Leitungsstruktur hinweisen wollen. Dazu beschreibt man das Lernen über das Herstellen von Verbindungen zwischen horizontalen und vertikalen, zwischen Zeilen- und Spalten-Leitungen⁴⁸ und das Abfragen über das Weiterleiten von elektrischen Strömen oder Zählimpulsen über die hergestellten Verbindungen.⁴⁹

Assoziativmatrizen sind Verbindungsmatrizen aus Nullen und Einsen

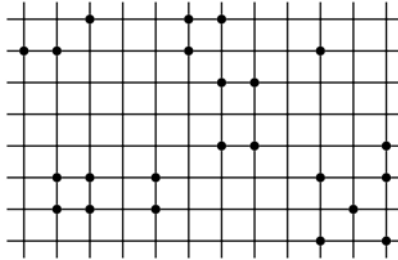


Abb. 2.28: Gitterdarstellung einer Assoziativmatrix

Die in Abb. 2.28 gewählte Darstellung von Assoziativmatrizen heißt **Gitterdarstellung**. Sie ist auch beim Schaltungsentwurf nützlich, wenn es darum geht, Assoziativmatrizen durch elektronische Bauteile nachzubilden (vgl. [Dierks 2005], S. 33 ff.).

Gitterdarstellung

Steht die Absicht im Vordergrund, eine mathematische Beschreibung des Lern- und Abfragevorgangs zu liefern, wird man die **Matrixdarstellung** einer Assoziativmatrix bevorzugen. Diese Darstellung erweist sich auch für den Anwender eines Computeralgebrasystems als nützlich, der auf diesem Wege Assoziativmatrizen untersucht, oder für den Programmierenden, der für ein Anwendungsprogramm einen geeigneten Datentyp zum Umgang mit Assoziativmatrizen sucht. Die folgende Darstellung gibt die gleiche Assoziativmatrix A wie die Gitterdarstellung in Abb. 2.28 an.

Matrixdarstellung

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}_{\Theta}$$

VIDAS-Darstellung

Das später ausführlich vorzustellende Simulationsprogramm VIDAS, mit dem die Vorgänge in und zwischen Assoziativmatrizen geplant und veranschaulicht werden können, nutzt ebenfalls die Matrixdarstellung. Die Assoziativmatrix A hätte dort eine Gestalt wie in Abb. 2.29. Eine weitere Darstellung derselben Matrix zeigt Abb. 2.30. Sie wird gelegentlich zur Darstellung auf kariertem Papier eingesetzt.

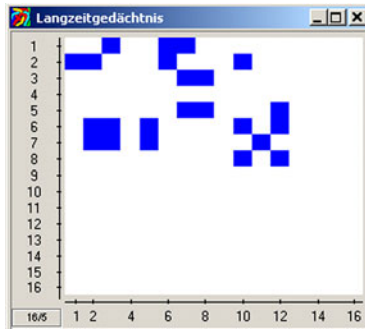


Abb. 2.29: VIDAS-Matrixdarstellung

			X			X	X								
X	X					X				X					
							X	X							
							X	X							X
		X	X		X						X			X	
		X	X		X							X			
												X			X

Abb. 2.30: Alternative Matrixdarstellung

2.5.2 Lernregeln von Assoziativmatrizen

Lernregeln bestimmen, wie die Verbindungen zwischen Spalten- und Zeilenleitungen in der Assoziativmatrix hergestellt werden, beziehungsweise wohin die Einsen in die Assoziativmatrix gesetzt werden sollen. Von den möglichen Lernregeln für Assoziativmatrizen⁵⁰ kommen in den in diesem Buch vorgestellten Assoziativmaschinen zwei zur Anwendung. Die **erste**, herkömmliche **Lernregel** ist Hebb-artig⁵¹ und wird für alle Matrizen genutzt außer für die Matrizen der sogenannten Kurzzeitgedächtnisse, deren Lernregel eine Art „Vergessen“ ermöglicht.⁵² Die erste Lernregel ist im Folgenden stets gemeint, wenn nicht ausdrücklich auf die Lernregel für das Kurzzeitgedächtnis einer Assoziativmaschine Bezug genommen wird. Diese andere Lernregel wird kurz **K-Lernregel** genannt.

aussagenlogische
Darstellung der ersten
Lernregel

Die mathematische Beschreibung der ersten Lernregel erfolgt durch die aussagenlogische Gleichung

$$a'_{i,j} = a_{i,j} \vee f_i a_j \quad ,$$

wobei $A' = (a'_{i,j})$ die Matrix nach dem Lernen, $A = (a_{i,j})$ die Matrix vor dem Lernen, f die Frage und a die Antwort bezeichnet. In den Fragen f und Antworten a werde dabei eine auftretende 0 als Wahrheitswert *falsch* und eine 1 als Wahrheitswert *wahr* gelesen.

Ein Frage-Antwort-Paar wird also gelernt, indem die Antwort in jede Zeile hineingeschrieben wird, in der die senkrecht neben der Matrix notierte Frage eine 1 besitzt. Kommt eine 1 auf eine Stelle, die bereits mit 1 belegt ist, dann ändert sich dort nichts, falls dort allerdings noch 0 steht, dann wird diese zu einer 1 geändert. An allen anderen Stellen ändert sich nichts. Dieses Vorgehen wurde in Kapitel 2.2 bereits als Regel zum Aufbau eines Perzeptrons für binär vorliegende Daten beschrieben und als **L-Lernregel** bezeichnet.

L-Lernregel

Die aussagenlogische Gleichung für die zweite Lernregel, die K-Lernregel, lautet:

$$a'_{i,j} = \overline{f_i} \, a_{i,j} \vee f_i \, a_j \quad .$$

aussagenlogische
Darstellung der
Lernregel für das
Kurzzeitgedächtnis
(K-Lernregel)

Ein Frage-Antwort-Paar wird hier so gelernt, dass alle Zeilen in der Matrix, in der die senkrecht neben der Matrix notierte Frage eine 1 hat, auf 0 gesetzt werden und danach in diese Zeilen die Antwort eingetragen wird.

Der mathematischen Beschreibung der Lernregeln sollen nun Beispiele mit Abbildungen folgen, um die genannten Lernregeln zu veranschaulichen.

In eine Assoziativmatrix mit acht Zeilen und zwölf Spalten sollen die folgenden drei Frage-Antwort-Paare (f_i, a_i) gemäß der ersten Lernregel eingetragen werden.

Beispiel für die
L-Lernregel

$$\begin{aligned} f_1 &= (0, 0, 1, 1, 0, 0, 0, 1) \, , \, a_1 = (1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0) \\ f_2 &= (0, 0, 0, 1, 1, 0, 1, 0) \, , \, a_2 = (0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1) \\ f_3 &= (1, 0, 1, 0, 0, 1, 1, 0) \, , \, a_3 = (0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0) \end{aligned}$$

B17

Dazu stelle man sich die Frage senkrecht neben die Zeilenleitungen geschrieben vor und denke sich diejenigen Leitungen aktiviert, vor denen eine 1 steht. Mit der Antwort und den Spaltenleitungen mache man es entsprechend. An den Kreuzungspunkten der aktivierten Leitungen wird die Verbindung hergestellt, was in der Gitterdarstellung wie in Abb. 2.31 als „Lötfleck“ und in der Matrixdarstellung als 1 eingetragen wird.

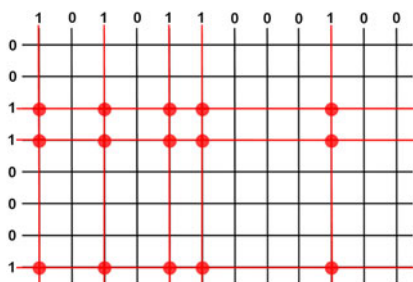


Abb. 2.31: Eintrag von Frage-Antwort-Paar (f_1, a_1) mit der L-Lernregel

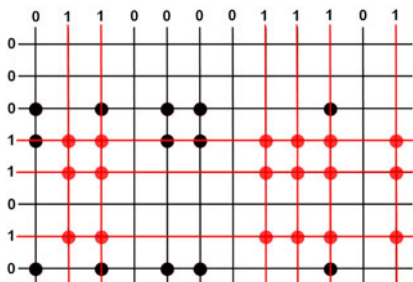


Abb. 2.32: Eintrag von Frage-Antwort-Paar (f_2, a_2) mit der L-Lernregel

Für das nächste einzutragende Frage-Antwort-Paar (f_2, a_2) geht man genauso vor und erhält den Zustand wie in Abb. 2.32 dargestellt.

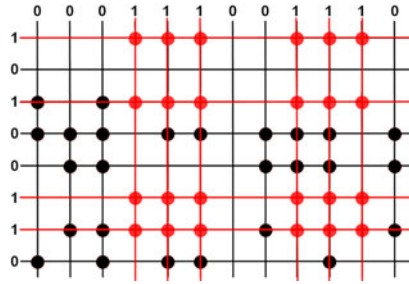


Abb. 2.33: Eintrag von Frage-Antwort-Paar (f_3, a_3) mit der L-Lernregel

Das dritte in diesem Beispiel einzutragenden Frage-Antwort-Paar (f_3, a_3) lässt nochmals deutlich werden, dass herzustellende Verbindungen an Kreuzungspunkten, die bereits verbunden sind, einfach im verbundenen Zustand verbleiben. Abb. 2.33 zeigt das Ergebnis nach dem Lernen aller drei Frage-Antwort-Paare nach der L-Lernregel.

Beispiel für die Lernregel für das Kurzzeitgedächtnis (K-Lernregel)

Die Lernregel für das Kurzzeitgedächtnis K bringt mit den Daten von B17 ein anderes Ergebnis. Nach dem Lernen der Frage-Antwort-Paare (f_1, a_1) und (f_2, a_2) enthält die Matrix die in Abb. 2.34 gezeigten Einträge.

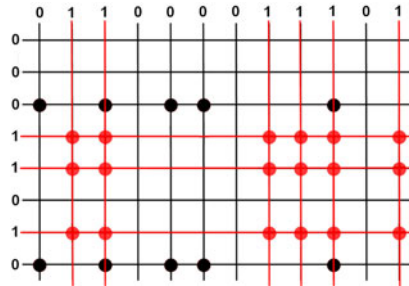


Abb. 2.34: Eintrag von Frage-Antwort-Paar (f_2, a_2) mit der K-Lernregel

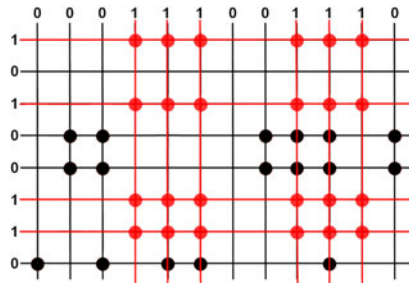


Abb. 2.35: Eintrag von Frage-Antwort-Paar (f_3, a_3) mit der K-Lernregel

Kommt nun noch das Frage-Antwort-Paar (f_3, a_3) hinzu, erkennt man im Vergleich mit Abb. 2.33 die Unterschiede zur ersten Lernregel deutlich.

Das Eintragen von n Frage-Antwort-Paaren in eine Assoziativmatrix A mit der L-Lernregel lässt sich durch Matrizenrechnung auch über den Zwischenschritt

Matrizenrechnung für L-Lernregel

$$A^* = \sum_{i=1}^n f_i^T \cdot a_i$$

erreichen, indem man auf die Elemente von Matrix A^* zum Schluss eine Schwelloperation mit dem Schwellwert $\Theta = 1$ anwendet.⁵³ Für die K-Lernregel sind hingegen in mehreren Schritten auszuführende, auf die einzelnen Matricelemente bezogene Operationen anzusetzen, da es auf die Reihenfolge der einzutragenden Frage-Antwort-Paare ankommt.⁵⁴

2.5.3 Abfragerregel von Assoziativmatrizen

Abgefragt werden Assoziativmatrizen immer nach der gleichen Regel, unabhängig davon, mit welcher Lernregel die Matrizen gefüllt wurden. Die Multiplikation der Frage f mit der Assoziativmatrix A

mathematische Darstellung der Abfragerregel

$$s = f \cdot A$$

berechnet das Zwischenergebnis s .⁵⁵ Als Schwellwert Θ wird ein Wert des Zwischenergebnisses s ausgewählt und die anschließende Schwelloperation

$$a_i = \begin{cases} 1, & \text{falls } s_i = \Theta \\ 0, & \text{sonst} \end{cases}$$

liefert die Antwort a .⁵⁶ Als Schwellwert Θ wird zumeist der größte Wert gewählt, der sich im Zwischenergebnis s finden lässt. Er wird als $\hat{\Theta}$ notiert. Doch gibt es Anwendungsfälle, in denen aus s auch kleinere Werte für die Schwelloperation genommen werden. Wenn im Folgenden bei Abfragen nicht anders angegeben, ist $\hat{\Theta}$ als Schwellwert anzunehmen.

größte Spaltensumme als Schwellwert $\hat{\Theta}$

Für das folgende Beispiel werden die Daten aus [B17] wieder aufgegriffen.

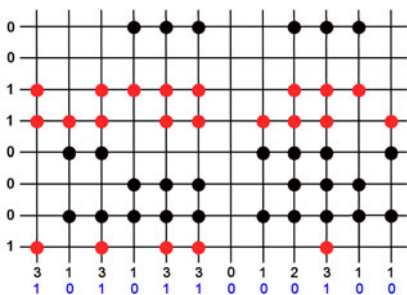


Abb. 2.36: Abfrage der Assoziativmatrix mit der Frage f_1

Zur Abfrage einer Assoziativmatrix stellt man sich die Frage senkrecht neben die Zeilenleitungen geschrieben vor und denkt sich diejenigen Leitungen

Beispiel für die Abfragerregel

aktiviert, vor denen eine 1 steht. An den Verbindungspunkten („Lötflecken“) wird die Aktivierung in die Spaltenleitungen weitergeleitet und aufsummiert. In Abb. 2.36 und 2.37 stehen diese Summen unterhalb der Spaltenleitungen.

Auf die zur Frage gehörende Antwort kommt man, wenn man die Positionen mit dem größten Wert, hier also dem Wert 3, auf 1 setzt und die übrigen auf 0. In Abb. 2.36 ist die entstehende Antwort unterhalb der Summen notiert. Auf die Frage f_1 wird also korrekt mit a_1 geantwortet.

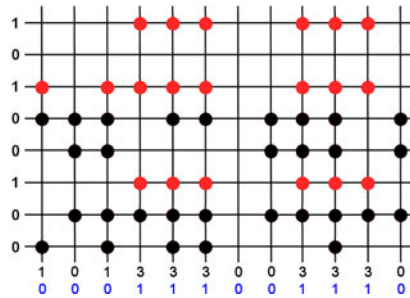


Abb. 2.37: Abfrage der Assoziativmatrix mit der Frage f_4

Im Beispiel aus Abb. 2.37 wird die Matrix mit einer Frage f_4 abgefragt, welche sich nur in der 7. Position von f_3 unterscheidet: $f_4 = (1, 0, 1, 0, 0, 1, 0, 0)$. Trotzdem liefert die Assoziativmatrix die Antwort a_3 , reagiert also fehlertolerant.

Schwellwertdiagramm

In der Matrixdarstellung von VIDAS werden die Spaltensummen wie in Abb. 2.38 unterhalb der Matrix angezeigt. Da man den Schwellwert aus diesen Summen ausgewählt, wird das Diagramm **Schwellwertdiagramm** genannt.

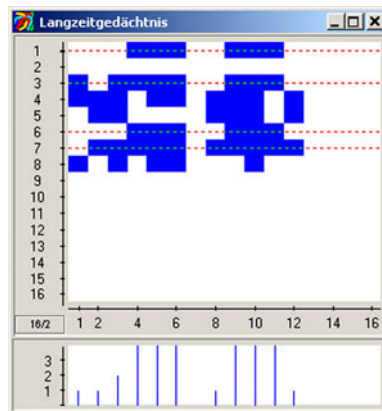


Abb. 2.38: Darstellung der Spaltensummen in VIDAS

In Abb. 2.38 ist die Abfrage der Matrix A mit der Frage f_3 aus **B17** wiedergegeben. Die aktivierten Zeilenleitungen werden markiert und die Größe der Spaltensummen entspricht der Höhe der Stäbe im Schwellwertdiagramm, hier liest man also $(1, 1, 2, 4, 4, 4, 0, 1, 4, 4, 4, 1)$ ab.

2.5.4 Besondere Eigenschaften der Assoziativmatrizen

Assoziativmatrizen sind Speicher mit besonderen Eigenschaften. Dazu gehören Eigenschaften aus den Bereichen Mustervervollständigung (pattern completion), Musterextraktion (pattern extraction) und Mustererkennung (pattern recognition). Das folgende Beispiel macht dies deutlich.

Gegeben seien die vier Buchstabenpaare $H - W$, $A - O$, $R - R$ und $D - K$, die gespeichert werden sollen. Die binäre Darstellung wählen wir hier nicht zufällig, sondern nehmen den ASCII-Code⁵⁷ wie in Tabelle 2.2 notiert.

Beispiel
HARD-WORK

H		1	0	0	1	0	0	0
	W	1	0	1	0	1	1	1
A		1	0	0	0	0	0	1
	O	1	0	0	1	1	1	1
R		1	0	1	0	0	1	0
	R	1	0	1	0	0	1	0
D		1	0	0	0	1	0	0
	K	1	0	0	1	0	1	1

B18

Tabelle 2.2: Kodierung HARD-WORK

Mit Hilfe der L-Lernregel aus Kapitel 2.5 werden in eine Matrix M die vier Buchstabenpaare $H - W$, $A - O$, $R - R$ und $D - K$ eingetragen, indem jeweils die Kodierung für den ersten Buchstaben die Frage und die Kodierung für den zweiten Buchstaben die Antwort bilden. Die Matrix M bekommt dann folgenden Inhalt:

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} .$$

Gemäß der Abfragerregel aus Kapitel 2.5 ergibt die Abfrage von M mit dem Buchstaben H und der größten auftretenden Summe als Schwellwert $\hat{\Theta}$:

$$(1\ 0\ 0\ 1\ 0\ 0\ 0) \cdot M = (2\ 0\ 2\ 1\ 2\ 2\ 2) \xrightarrow{\hat{\Theta}} (1\ 0\ 1\ 0\ 1\ 1\ 1) \hat{=} W$$

Wie gewünscht erhält man als Antwort also den Buchstaben W . Die Spaltensummen könnte man ohne Multiplikation hier auch schnell durch Addition der Zeilen 1 und 4 von Matrix M berechnen. Analoges gilt für alle vier gespeicherten Paare, das heißt alle Antworten werden durch M korrekt erzeugt.

Es sei nun angenommen, dass bei einer gewünschten Abfrage mit R die erste Eins bei einer Übertragung verloren gegangen sei. Die Matrix M leistet die angestrebte **Mustervervollständigung**

Mustervervollständigung

$$(0\ 0\ 1\ 0\ 0\ 1\ 0) \cdot M = (2\ 0\ 2\ 0\ 0\ 2\ 0) \xrightarrow{\hat{\Theta}} (1\ 0\ 1\ 0\ 0\ 1\ 0) \hat{=} R ,$$

indem sie mit dem Buchstaben R wie erwartet antwortet.

Musterextraktion

Jetzt sei demgegenüber bei einer Abfrage mit dem Buchstaben R versehentlich eine Eins hinzugekommen, dann erbringt die Matrix M die erhoffte **Musterextraktion**

$$(1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0) \cdot M = (3 \ 0 \ 3 \ 1 \ 1 \ 3 \ 1) \xrightarrow{\hat{Q}} (1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0) \hat{=} R \ .$$

Hierzu beachte man, dass die Abfrage mit der zusätzlichen Eins eine maximale Summe von 4 im Zwischenergebnis erwarten lassen könnte, da in der Abfrage durch den Fehler nunmehr vier Einsen enthalten sind. Dennoch ergibt die Abfrageregeln hier als größte Summe den Wert 3. Eine Assoziativmaschine, die mit diesen Assoziativmatrizen umgeht, muss folglich bei der Hardwarerealisierung ein Schaltwerk für die Schwellwertsteuerung besitzen, welches nicht automatisch nur die maximal zu erwartende Summe als Schwellwert einsetzt, sondern auf die tatsächlich größte Summe achtet.

Schwellwertsteuerung

Mustererkennung

Angenommen, es kam bei einer Übertragung für eine Abfrage mit dem Buchstaben D zu zwei Fehlern, eine Null wurde zur Eins und eine Eins wurde zu Null. Jetzt liefert die Matrix M die gewünschte **Mustererkennung**

$$(0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0) \cdot M = (1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1) \xrightarrow{\hat{Q}} (1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1) \hat{=} K \ .$$

Beispiel Ortsnamen



Abb. 2.39: 525 zufällig ausgewählte Ortsnamen

Obgleich dieses Beispiel [B18] eigens konstruiert wurde, um auf kleinem Platz alle drei betrachteten Fälle exemplarisch zu zeigen, bleiben die gesehenen günstigen Eigenschaften der Assoziativmatrix auch in konkreten Anwendungen erhalten. Dazu hat man eine passende Matrixgröße und eine geeignete, spärliche Kodierung zu finden, die möglichst ähnlichkeitserhaltend ist.⁵⁸ Es ist nicht immer möglich (und auch nicht nötig), die Anzahl der Einsen für alle Fragen und Antworten konstant zu halten, man kann aber die Kenntnisse darüber bei der Schwellwertsteuerung mit Vorteil verwenden.

Die eben beschriebenen günstigen Eigenschaften seien durch ein weiteres Beispiel belegt. Es werden dazu in eine Assoziativmatrix über 500 zufällig ausgewählte Ortsnamen eingetragen. Anschließend lässt sich die Matrix mit fehlerhaften Schreibungen oder Namensumstellungen oder irrtümlichen Zusätzen abfragen, um ihr Antwortverhalten zu untersuchen.⁵⁹

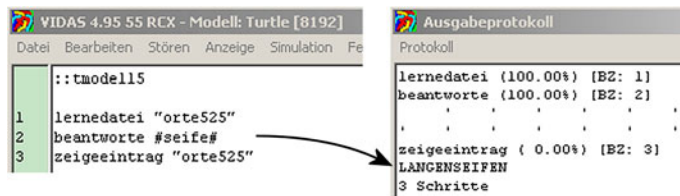


Abb. 2.40: Mustervervollständigung: Nur noch Teile eines Ortsnamens sind bekannt.

Die Abfrage in Abb. 2.40 veranschaulicht, wie sich Assoziativmatrizen zur Mustervervollständigung einsetzen lassen. Ein Namensteil genügt, um den vollständigen Ortsnamen zu finden. Das Bruchstück 'seife' wird mit dem Ortsnamen Langenseifen beantwortet. Sollte dieser Namensteil in mehreren der gelernten Ortsnamen auftauchen, werden alle diese Ortsnamen ausgegeben.



Abb. 2.41: Mustervervollständigung: Was ist hier abgebildet?

Die Fähigkeit zur Mustervervollständigung wird sich in vielen Bereichen nutzen lassen, in denen nur noch Fragmente von gelernten Daten zur Abfrage vorliegen. Was man auf diesem Feld zu leisten selbst im Stande ist, studiere man beim Betrachten von Abb. 2.41.

Mustervervollständigung

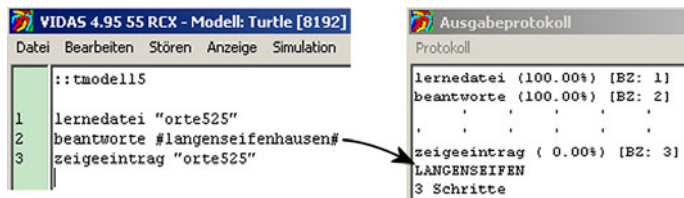


Abb. 2.42: Musterextraktion: Der Ortsname erhält irrtümlich Zusätze.

Sollte man auf der Suche nach einem Ortsnamen irrtümlich Zusätze anfügen, die der gesuchte Name gar nicht enthält, so führt das hier bei der Matrixabfrage trotzdem zum gesuchten Ortsnamen wie Abb. 2.42 zeigt. Der Ortsname Langenseifen wird aus der Anfrage 'langenseifenhausen' extrahiert.

Musterextraktion

GARASCHALEM

Abb. 2.43: Musterextraktion: GARASCHALEM

Auch aus der durch Hinzufügen von drei Zeichen entstandenen Buchstabenfolge in Abb. 2.43 lässt sich das ursprüngliche Wort extrahieren.⁶⁰

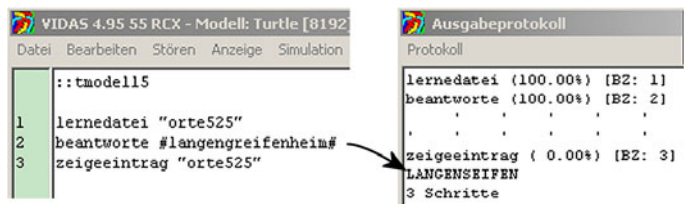


Abb. 2.44: Mustererkennung: Der Ortsname enthält Fehlschreibungen und Zusätze.

Ferner kann der gesuchte Ortsname auch aus fehlgeschriebenen oder irrtümlich ergänzten Anfragen ermittelt werden. Zur Anfrage 'langengreifenheim'

Mustererkennung

wird der gesuchte Name erkannt. Abb. 2.44 verdeutlicht dieses günstige Verhalten der Assoziativmatrix bei der Mustererkennung.



Abb. 2.45: Mustererkennung: Was ist hier abgebildet?

Es fällt uns nicht schwer, zugrundeliegende Figuren in Abb. 2.45 zu erkennen, obwohl sie von den erwarteten und gelernten Mustern in mehreren Einzelheiten abweichen. Dabei kann es dazu kommen, dass sich gelernte Muster so überlagern, dass man beispielsweise rechts unten im Bild entweder vornehmlich eine Katze oder vornehmlich einen Fisch erkennt.

Speicherkapazität

Mit dem HARD-WORK-Beispiel B18 lässt sich außerdem auf die hohe Speicherkapazität einer Assoziativmatrix hinweisen, indem zusätzlich zu den vier Buchstabenpaaren noch das Buchstabenpaar $K - B$ in die Matrix mit $B = (1\ 0\ 0\ 0\ 1\ 0)$ eingetragen wird. Alle fünf Paare lassen sich korrekt auslesen. Die diskutierten Erkennungsbeispiele bleiben unberührt, da sich die Belegung der Matrix M nicht geändert hat.

Wegen der sich beim Eintragen der Antworten in die Matrix ergebenden Redundanz (die Antworten werden gemäß der Lernregeln jeweils in mehrere Zeilen eingetragen) und der resultierenden Robustheit könnte man annehmen, der durch die Matrix benötigte Platz werde relativ schlecht zum Speichern der Information genutzt („Redundanz kostet Platz“). Das ist aber nicht der Fall. Man erinnere sich daran, dass ein Teil der redundant eingetragenen Einsen oft auch von den Einsen anderer Einträge genutzt werden, wodurch das Speichervermögen insgesamt wieder ansteigt. Mit mathematischen Methoden lässt sich das Zusammenwirken der Einfluss nehmenden Größen beschreiben und im Hinblick auf unterschiedliche Fragestellungen (Zeit, Platz u.a.) optimieren. Es gibt Beispiele, in denen der Platz so gut ausgenutzt wird, dass geradezu eine Kompression der einzutragenden Daten geschieht – bei unverminderter, also sehr schneller Auslesegeschwindigkeit. Es muss nicht dekomprimiert werden.

Beispiel mit 13 Frage-Antwort-Paaren in einer 6x6-Matrix

B19

Es sei dazu ein Beispiel angefügt, bei welchem wir berechnen, wie viel Information in der Matrix gespeichert und wie groß die Ausnutzung der Matrix dadurch wird. Die eingesetzte Assoziativmatrix habe nur je sechs Zeilen- und Spaltenleitungen. In ihr werden über die herkömmliche Lernregel die 13 Frage- und Antwort-Paare aus Tabelle 2.46 abgelegt. Jede Frage und jede Antwort besteht aus zwei Einsen und vier Nullen. Nach dem Lernen sind in der Matrix 21 von 36 möglichen Plätzen belegt, wie in Abb. 2.47 zu erkennen, die Dichte der Belegung beträgt also über 58 Prozent.

Frage	Antwort
000110	010100
010100	100100
100100	110000
110000	100010
100010	010010
010010	000110
000011	010100
000101	010100
011000	101000
101000	100001
100001	010001
010001	001100
001001	001001

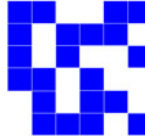


Abb. 2.47: 6x6-Assoziativmatrix mit 13 gelernten Frage-Antwort-Paaren

Abb. 2.46: 13 Frage- und Antwort-Paare für eine 6x6-Assoziativmatrix

Zunächst vergewissere man sich, dass die 6x6-Matrix auf alle Fragen aus Tabelle 2.46 die zugehörigen Antworten korrekt liefert. Das trifft zu. In jeder der 13 Antworten steckt Information, deren Betrag sich danach bemessen sollte, wie gewiss oder ungewiss das Eintreten der Antwort ist. Wenn beispielsweise sicher ist, dass eine Matrix stets mit (1 1 0 1 0 1) antworten wird, dann ist der Informationsgehalt der Antwort gleich null, denn man weiß die Antwort schon vorher. In unserem Beispiel weiß man von der Antwort im Voraus jedoch nur, dass sie zwei Einsen enthalten wird. Diese zwei Einsen können in der Antwort an sechs verschiedenen Positionen stehen, daher beträgt die Anzahl an Möglichkeiten $\binom{6}{2} = 15$. Die Bestimmung des Informationsgehalts einer Antwort orientiert sich daran, wie viele Ja-Nein-Fragen nötig sind, um diese Antwort zu erraten.⁶¹ Seien alle Möglichkeiten gleichwahrscheinlich. Man denke sich alle 15 Möglichkeiten notiert in einer Liste. Dann führen Ja-Nein-Fragen wie „Liegt die Antwort in der ersten Hälfte der Liste?“ immer wieder angewandt auf die Listenhälfte, in der die Antwort liegt, zu einer Anzahl von höchstens vier Ja-Nein-Fragen. Diese Fragestrategie ist womöglich nicht optimal.⁶² Die Informationstheorie⁶³ bestimmt den Gehalt an Information I einer Antwort⁶⁴ durch den Zweierlogarithmus ld des Kehrwerts der Wahrscheinlichkeit p , mit dem die Antwort eintritt.⁶⁵

$$I(p) = ld\left(\frac{1}{p}\right) = -ld(p)$$

Die Maßeinheit für den Informationsgehalt ist das Informationsbit, kurz *bit*.⁶⁶

Folgt man der Definition von Shannon, so käme man in unserem Beispiel auf $I = ld(15) \approx 3,907 \text{ bit}$, was zum obigen Ergebnis von höchstens vier Ja-Nein-Fragen pro Antwort passt. Damit enthielten die 13 Antworten in der 6x6-Matrix $13 \cdot ld(15) \text{ bit} \approx 50,79 \text{ bit}$ Information. Jeder der 36 Plätze in der Matrix trägt $\frac{50,79}{36} \text{ bit} \approx 1,41 \text{ bit}$ Information. Der **Ausnutzungsgrad** der Matrix dieses konstruierten Beispiels betrüge dann etwa 141 Prozent, ein Ergebnis, das nicht im Einklang mit dem normalen Verständnis von Informationsgehalt und Repräsentanten für Information steht. Es legt nahe, die Grundbegriffe beziehungsweise die Grundkonfiguration zu überdenken und neu zu analysieren.⁶⁷ Unserer Meinung nach lässt sich in solchen konstruierten Matrizen wie in Beispiel [B19] Information hochgradig komprimiert

Ausnutzungsgrad
einer Assoziativmatrix

speichern, mit der Besonderheit, dass das Auslesen ohne extra Aufwand für die Dekompression möglich ist.

Im **allgemeinen Fall** liegt der asymptotisch maximale Ausnutzungsgrad für größer werdende Matrizen bei 69 Prozent, weil man im Unterschied zum konstruierten Beispiel einzurechnen hat, wie viel Information durch das Auftreten „falscher Einsen“ wieder verloren geht.⁶⁸ Lernt man zu den 13 Frage-Antwort-Paaren aus Abb. 2.46 zum Beispiel als 14. Paar (0 0 1 1 0 0, 0 1 0 1 0 1) hinzu, dann werden in fünf Antworten eine falsche Eins und in einer Antwort zwei falsche Einsen erscheinen, was den Informationsgehalt ansenkt. Trägt man nun als 15. Paar noch (0 0 1 0 1 0, 1 0 1 0 1 0) ein, dann gäbe es zwar auf alle kombinatorisch möglichen Anfragen eine Antwort, allerdings auf Kosten eines weiteren Verlustes an Auslesegüte.

falsche Einsen

2.5.5 Assoziativmatrizen für besondere Aufgaben

Im vorangegangenen Kapitel wurden die Eigenschaften von Assoziativmatrizen bei der Erfüllung von Aufgaben der Mustererkennung erläutert. Dabei ergab das Lernen der Frage-Antwort-Paare die Belegung der Matrix. Das soll in diesem Kapitel anders sein. Die Belegung der folgenden Assoziativmatrizen erfolgt durch Konstruktion, damit aussagenlogische Verknüpfungen wie in Kapitel 2.1 möglich werden oder damit in den Matrizen Sequenzen abgelegt werden können.⁶⁹ Weiter unten kommen dann zudem Rechenoperationen hinzu, die ebenfalls mit recht kleinen Assoziativmatrizen geleistet werden können.

konstruierte Matrizen

Zunächst wird wie in Beispiel B18 vorgegangen. Es wird festgelegt, wie die Wahrheitswerte binär zu kodieren sind (s. Tabelle 2.3).

B20

<i>wahr</i>	1	1
<i>falsch</i>	0	1

Tabelle 2.3: Kodierung *wahr-falsch*

Matrizen, die einen Wahrheitswert negieren (NICHT-Verknüpfung), entstehen nun nicht durch Lernen, sondern werden passend überlegt. Hier könnten die Matrizen M_{NOT} oder M_{NOT}^* diesen Zweck erfüllen, was wir anschließend überprüfen werden.

$$M_{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad M_{NOT}^* = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

Wir verabreden, dass der Schwellwert immer gleich der Anzahl der Einsen in der Frage sein soll, und rechnen nach:

$$\begin{aligned} (1 \ 1) \cdot M_{NOT} &= (1 \ 1) \xrightarrow{2} (0 \ 0) \hat{=} \textit{undefiniert} \\ (1 \ 1) \cdot M_{NOT}^* &= (1 \ 2) \xrightarrow{2} (0 \ 1) \hat{=} \textit{falsch} \\ (0 \ 1) \cdot M_{NOT} &= (1 \ 0) \xrightarrow{1} (1 \ 0) \hat{=} \textit{undefiniert} \\ (0 \ 1) \cdot M_{NOT}^* &= (1 \ 1) \xrightarrow{1} (1 \ 1) \hat{=} \textit{wahr} \end{aligned}$$

Vorschnell ließe sich als Ergebnis dieser Rechnungen die Matrix M_{NOT} beiseite legen, denn sie liefert keine definierten Antworten gemäß Tabelle 2.3.

Doch wenn man nur auf das erste Bit der Antworten achtet oder wenn man auch (1 0) als *wahr* und (0 0) als *falsch* in Tabelle 2.3 nachträgt, dann erfüllt auch M_{NOT} seine Aufgabe.

Lediglich einzelne Bits der Antwort auf eine Matrixabfrage als Ergebnis zu betrachten, ist eine Herangehensweise, die es erlaubt, mehrere aussagenlogische Verknüpfungen gleichzeitig auszuführen. Dazu betrachte man die Kodierung in Tabelle 2.48 und Matrix $M_{\#}$.

<i>wahr</i> – <i>wahr</i>	1	1	0	0
<i>wahr</i> – <i>falsch</i>	1	0	1	0
<i>falsch</i> – <i>wahr</i>	0	1	1	0
<i>falsch</i> – <i>falsch</i>	0	0	1	1

$$M_{\#} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

B21

Abb. 2.48: Kodierung der Wahrheitswertepaare

Die Abfrage mit allen Wahrheitswertepaaren ergibt:

$$\begin{aligned} (1\ 1\ 0\ 0) \cdot M_{\#} &= (1\ 1\ 2\ 2) \xrightarrow{2} (0\ 0\ 1\ 1) \\ (1\ 0\ 1\ 0) \cdot M_{\#} &= (2\ 1\ 1\ 2) \xrightarrow{2} (1\ 0\ 0\ 1) \\ (0\ 1\ 1\ 0) \cdot M_{\#} &= (1\ 2\ 1\ 2) \xrightarrow{2} (0\ 1\ 0\ 1) \\ (0\ 0\ 1\ 1) \cdot M_{\#} &= (2\ 2\ 1\ 1) \xrightarrow{2} (1\ 1\ 0\ 0) \end{aligned}$$

Man beachte nun, dass durch die Konstruktion der Matrix in den dritten Bits der Antworten stets die aussagenlogische Konjunktion und in den vierten Bits die Disjunktion der Wahrheitswerte des abgefragten Wahrheitswertepaares zu finden ist. Und auch die ersten beiden Bits haben eine Bedeutung: hier lässt sich am zweiten Bit die Negation des ersten Operanden und am ersten Bit diejenige des zweiten Operanden ablesen. Somit sind durch $M_{\#}$ die AND-, OR- und NOT-Verknüpfung der Aussagenlogik nachgebildet worden.

aussagenlogische
Verknüpfungen

Die Beispiele B20 und B21 machen nochmals deutlich, dass für die Operanden jeweils eine eindeutige Zuordnung wie in Tabelle 2.48 gegeben sein muss, damit die angegebenen Matrizen ihre gewünschte Leistung erbringen können. Und auch die Weise, wie das Ergebnis abzulesen ist, wird genau festgelegt. Damit sind die Schnittstellen zur Außenwelt, sei es beispielsweise zum Anschluss einer Tastatur und einer Digitalanzeige, klar beschrieben. Zur Nachbildung eines Rechnens durch ein Gehirn wird man hier anders vorgehen müssen, da Tabellen wie 2.48 fehlen (s. Kapitel 6).

Besonders angenehm ist in diesen Beispielen, dass die Matrizen klein sind. Dies ist ein günstiger Nebenaspekt der Konstruktion. Im Kontrast zur Verwendung der Assoziativmatrizen als Informationsspeicher, wenn sich die Kodierungen also an der Anwendungsaufgabe orientieren und Fehler möglichst klein gehalten werden müssen, spielen hier stochastische Auflagen praktisch keine Rolle. Insofern erzwingt die gewünschte Leistung keine langen, weil spärlich zu kodierende, Bitfolgen. Dafür sind diese besonderen Verknüpfungsmatrizen jedoch nicht ohne Weiteres fehlertolerant und die beteiligten Bitfolgen nicht unbedingt spärlich mit Einsen besetzt. Wir werden später im Kapitel 6 auf das Assoziative Rechnen eingehen, bei dem andere Ansätze zum

Rechnen durch Assoziativmatrizen verfolgt werden. Dabei werden wir dann nicht auf spezielle Optimierungen im Hinblick auf konkrete Implementierungen achten, sondern das Hauptaugenmerk auf den systematischen Aufbau legen. Die Erkenntnisse, Begriffe und Werkzeuge, beispielsweise die sogenannten Tilde-Variablen, lassen sich dann auch für andere Aufgaben verwenden.

Sequenzen speichern

Bevor wir nun auf weitere besonders konstruierte Assoziativmatrizen eingehen, die zum Rechnen dienen können, sei nochmals auf Matrizen wie in Abbildung 2.47 eingegangen, die in der Lage sind, Sequenzen von Bitfolgen zu speichern. Diese Sequenzen von Bitfolgen werden im Weiteren auch Assoziationsketten genannt. Setzt man beginnend mit der Frage (0 0 0 1 1 0) in die gezeigte 6x6-Matrix die jeweilige Antwort wieder als Frage ein, landet man nach 12 Schritten bei der Antwort (0 0 1 1 0 0). Dieses ist erstaunlich, denn alle Glieder der Assoziationskette besitzen genau zwei Einsen, es sind also nur maximal 15 verschiedene dieser Glieder überhaupt möglich. Man kann mit einigem Probieren viele ähnliche Beispiele finden. Besonders schön finden wir das Beispiel [B22] einer 7x7-Matrix M_{Fab} mit drei separaten Assoziationskreisen der Länge 7, wobei sämtliche $\binom{7}{2} = 21$ der möglichen Glieder mit genau zwei Einsen genutzt werden.⁷⁰

[B22]

$$M_{Fab} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Mit der Startfrage (1 1 0 0 0 0) erhält man den ersten Assoziationskreis:

$$\begin{aligned} (1 \ 1 \ 0 \ 0 \ 0 \ 0) \cdot M_{Fab} &= (1 \ 2 \ 2 \ 1 \ 1 \ 1 \ 0) \xrightarrow{2} (0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0) \\ (0 \ 1 \ 1 \ 0 \ 0 \ 0) \cdot M_{Fab} &= (0 \ 1 \ 2 \ 2 \ 1 \ 1 \ 1) \xrightarrow{2} (0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0) \\ &\vdots \\ (1 \ 0 \ 0 \ 0 \ 0 \ 1) \cdot M_{Fab} &= (2 \ 2 \ 1 \ 1 \ 1 \ 0 \ 1) \xrightarrow{2} (1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0) \end{aligned}$$

Fragt man aber mit (1 0 1 0 0 0) ab, dann ergibt sich der zweite Assoziationskreis:

$$\begin{aligned} (1 \ 0 \ 1 \ 0 \ 0 \ 0) \cdot M_{Fab} &= (1 \ 1 \ 2 \ 1 \ 2 \ 0 \ 1) \xrightarrow{2} (0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0) \\ (0 \ 0 \ 1 \ 0 \ 1 \ 0) \cdot M_{Fab} &= (0 \ 1 \ 1 \ 1 \ 2 \ 1 \ 2) \xrightarrow{2} (0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1) \\ &\vdots \\ (1 \ 0 \ 0 \ 0 \ 1 \ 0) \cdot M_{Fab} &= (2 \ 1 \ 2 \ 0 \ 1 \ 1 \ 1) \xrightarrow{2} (1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0) \end{aligned}$$

Und mit (1 0 0 1 0 0) lässt sich der dritte Assoziationskreis der Länge 7 starten, wovon sich der Leser durch eigene Rechnung überzeugen mag.

In Assoziativmatrizen lassen sich auch Sequenzen speichern, die mit unterschiedlichen Fragen gestartet werden, sich bei einem gemeinsamen Glied der

Assoziationskette treffen und letztlich bei einer festen Antworten verharren, wie folgendes Beispiel B23 mit der Matrix M_{Fix} zeigt.

$$M_{Fix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \text{B23}$$

Startet man mit (1 1 0 0 0 0) landet man letztlich bei (0 0 0 0 1 1).

$$\begin{aligned} (1 \ 1 \ 0 \ 0 \ 0 \ 0) \cdot M_{Fix} &= (2 \ 1 \ 2 \ 1 \ 1 \ 1) \xrightarrow{2} (1 \ 0 \ 1 \ 0 \ 0 \ 0) \\ (1 \ 0 \ 1 \ 0 \ 0 \ 0) \cdot M_{Fix} &= (1 \ 2 \ 1 \ 2 \ 0 \ 0) \xrightarrow{2} (0 \ 1 \ 0 \ 1 \ 0 \ 0) \\ (0 \ 1 \ 0 \ 1 \ 0 \ 0) \cdot M_{Fix} &= (1 \ 0 \ 1 \ 0 \ 2 \ 2) \xrightarrow{2} (0 \ 0 \ 0 \ 0 \ 1 \ 1) \end{aligned}$$

Wird hingegen zuerst mit (0 0 1 0 0 1) abgefragt, gelangt man nach einem Schritt zu einem gemeinsamen Glied der beiden Ketten, so dass man ebenfalls bei (0 0 0 0 1 1) anlangt.

$$\begin{aligned} (0 \ 0 \ 1 \ 0 \ 0 \ 1) \cdot M_{Fix} &= (0 \ 2 \ 0 \ 2 \ 1 \ 1) \xrightarrow{2} (0 \ 1 \ 0 \ 1 \ 0 \ 0) \\ (0 \ 1 \ 0 \ 1 \ 0 \ 0) \cdot M_{Fix} &= (1 \ 0 \ 1 \ 0 \ 2 \ 2) \xrightarrow{2} (0 \ 0 \ 0 \ 0 \ 1 \ 1) \end{aligned}$$

Für die elementare Aufgabe, bestimmte Abschnitte einer Bitfolge, beispielsweise die ersten drei Bits von (a b c 0 0 0) mit $a, b, c \in \{0, 1\}$, an eine andere Stelle zu verschieben, eignen sich Assoziativmatrizen $M_{\rightarrow}$ folgender Gestalt:

Abschnitte verschieben

$$M_{\rightarrow} = \begin{pmatrix} 0 & 0 & 0 & a & b & c \\ 0 & 0 & 0 & a & b & c \\ 0 & 0 & 0 & a & b & c \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{B24}$$

Fragt man $M_{\rightarrow}$ mit Bitfolgen wie nachstehend an, so stehen die ersten drei Bits der Frage anschließend in der Antwort an den letzten drei Positionen.

$$\begin{aligned} (1 \ 0 \ 1 \ 0 \ 0 \ 0) \cdot M_{\rightarrow} &= (0 \ 0 \ 0 \ 2 \ 0 \ 2) \xrightarrow{\hat{=}} (0 \ 0 \ 0 \ 1 \ 0 \ 1) \\ (1 \ 1 \ 0 \ 0 \ 0 \ 0) \cdot M_{\rightarrow} &= (0 \ 0 \ 0 \ 2 \ 2 \ 0) \xrightarrow{\hat{=}} (0 \ 0 \ 0 \ 1 \ 1 \ 0) \end{aligned}$$

Den zahlreichen Möglichkeiten zwei natürliche Zahlen zu addieren oder subtrahieren fügen wir nun eine weitere hinzu, die eine besondere Assoziativmatrix zu Hilfe nimmt. Damit dieses übersichtlich bleibt, rechnen wir im 3er-System, stellen also alle Zahlen mit nur drei Ziffern dar, die hier 0, 1 und 2 sein sollen. Für die Addition werden nunmehr zwei Verknüpfungstafeln aufgestellt (s. Abbildungen 2.49 und 2.50), wovon die linke gewählt wird, wenn die vorangegangene Addition einer Stelle keinen Übertrag geliefert hat, und die rechte sonst.

Rechnen mit natürlichen Zahlen

+	0	1	2
0	00	01	02
1	01	02	10
2	02	10	11

Abb. 2.49: Addition ohne Übertrag

+	0	1	2
0	01	02	10
1	02	10	11
2	10	11	12

Abb. 2.50: Addition mit Übertrag

Möchte man also beispielsweise die Ziffern 2 und 1 addieren und liegt kein Übertrag vor, dann entnimmt man der linken Tabelle in die dritte Zeile und zweiten Spalte das Ergebnis 10, wobei die 0 die Ergebnisziffer bedeutet und die 1 den Übertrag, der für die nächste Stelle zu berücksichtigen ist. Die hintere Ziffer des Ergebnisses ist also stets als Ergebnisziffer und die vordere als Übertragsanzeige zu verstehen.

Durch die Verknüpfungstabellen 2.49 und 2.50 wird ersichtlich, dass wir mit unserer zu konstruierenden Rechen-Assoziativmatrix auf neun verschiedene Fragen mit sechs Antworten reagieren können müssen und dabei den Übertrag zu berücksichtigen haben. Zunächst wird dazu eine Kodierung der neun Fragen angegeben (s. Tabelle 2.4). Das letzte Bit der Kodierung ist mit $\ddot{u}$ bezeichnet, denn es soll den Wert 1 erhalten, wenn **kein** Übertrag zu verrechnen ist, und den Wert 0 sonst.

B25

0+0	1	0	0	0	1	0	$\ddot{u}$
0+1	0	0	0	1	1	0	$\ddot{u}$
0+2	1	0	1	0	0	0	$\ddot{u}$
1+0	0	0	1	0	1	0	$\ddot{u}$
1+1	1	1	0	0	0	0	$\ddot{u}$
1+2	0	0	0	0	1	1	$\ddot{u}$
2+0	0	1	1	0	0	0	$\ddot{u}$
2+1	0	1	0	0	0	1	$\ddot{u}$
2+2	1	0	0	0	0	1	$\ddot{u}$

Tabelle 2.4: Kodierung der neun Summen

Die nachstehende Matrix M_3 soll die gewünschten Summen liefern.

$$M_3 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

ohne Übertrag

Ohne einem zu verrechnenden Übertrag ergibt die Abfrage von M_3 mit allen neun möglichen Summen aus Tabelle 2.4:

$$\begin{aligned} 0+0: & (1\ 0\ 0\ 0\ 1\ 0\ 1) \cdot M_3 \xrightarrow{3} (1\ \mathbf{1\ 0\ 0\ 0\ 0\ 0}\ 1\ 0\ 0\ 0\ 0\ 0) \\ 0+1: & (0\ 0\ 0\ 1\ 1\ 0\ 1) \cdot M_3 \xrightarrow{3} (1\ \mathbf{0\ 1\ 0\ 0\ 0\ 0}\ 1\ 0\ 0\ 0\ 0\ 0) \\ 0+2: & (1\ 0\ 1\ 0\ 0\ 0\ 1) \cdot M_3 \xrightarrow{3} (1\ \mathbf{0\ 0\ 1\ 0\ 0\ 0}\ 1\ 0\ 0\ 0\ 0\ 0) \\ 1+0: & (0\ 0\ 1\ 0\ 1\ 0\ 1) \cdot M_3 \xrightarrow{3} (1\ \mathbf{0\ 1\ 0\ 0\ 0\ 0}\ 1\ 0\ 0\ 0\ 0\ 0) \end{aligned}$$

$$\begin{aligned}
1+1: (1\ 1\ 0\ 0\ 0\ 0\ 1) \cdot M_3 &\xrightarrow{3} (1\ \mathbf{0\ 0\ 1\ 0\ 0}\ 1\ 0\ 0\ 0\ 0\ 0) \\
1+2: (0\ 0\ 0\ 0\ 1\ 1\ 1) \cdot M_3 &\xrightarrow{3} (1\ \mathbf{0\ 0\ 0\ 1\ 0}\ 0\ 0\ 0\ 0\ 0\ 0) \\
2+0: (0\ 1\ 1\ 0\ 0\ 0\ 1) \cdot M_3 &\xrightarrow{3} (1\ \mathbf{0\ 0\ 1\ 0\ 0}\ 1\ 0\ 0\ 0\ 0\ 0) \\
2+1: (0\ 1\ 0\ 0\ 0\ 1\ 1) \cdot M_3 &\xrightarrow{3} (1\ \mathbf{0\ 0\ 0\ 1\ 0}\ 0\ 0\ 0\ 0\ 0\ 0) \\
2+2: (1\ 0\ 0\ 0\ 0\ 1\ 1) \cdot M_3 &\xrightarrow{3} (1\ \mathbf{0\ 0\ 0\ 0\ 1}\ 0\ 0\ 0\ 0\ 0\ 0) \\
&\quad 00\ 01\ 02\ 10\ 11
\end{aligned}$$

Die Ergebnisse der Rechnungen ohne Übertrag sind an den Bits an den Positionen 2 bis 6 abzulesen. Steht eine 1 an Position 2, dann ist das Ergebnis 00. Steht an der dritten Position eine 1, dann lautet das Ergebnis 01 und so fort. Die jeweilige Bedeutung wurde unterhalb der Spalten notiert.

Mit einem zu verrechnenden Übertrag ergibt die Abfrage von M_3 mit allen neun möglichen Summen aus Tabelle 2.4:

mit Übertrag

$$\begin{aligned}
0+0: (1\ 0\ 0\ 0\ 1\ 0\ 0) \cdot M_3 &\xrightarrow{2} (1\ 1\ 0\ 0\ 0\ 0\ 1\ \mathbf{1\ 0\ 0\ 0\ 0}) \\
0+1: (0\ 0\ 0\ 1\ 1\ 0\ 0) \cdot M_3 &\xrightarrow{2} (1\ 0\ 1\ 0\ 0\ 0\ 1\ \mathbf{0\ 1\ 0\ 0\ 0}) \\
0+2: (1\ 0\ 1\ 0\ 0\ 0\ 0) \cdot M_3 &\xrightarrow{2} (1\ 0\ 0\ 1\ 0\ 0\ 1\ \mathbf{0\ 0\ 1\ 0\ 0}) \\
1+0: (0\ 0\ 1\ 0\ 1\ 0\ 0) \cdot M_3 &\xrightarrow{2} (1\ 0\ 1\ 0\ 0\ 0\ 1\ \mathbf{0\ 1\ 0\ 0\ 0}) \\
1+1: (1\ 1\ 0\ 0\ 0\ 0\ 0) \cdot M_3 &\xrightarrow{2} (1\ 0\ 0\ 1\ 0\ 0\ 1\ \mathbf{0\ 0\ 1\ 0\ 0}) \\
1+2: (0\ 0\ 0\ 0\ 1\ 1\ 0) \cdot M_3 &\xrightarrow{2} (1\ 0\ 0\ 0\ 1\ 0\ 0\ \mathbf{0\ 0\ 0\ 1\ 0}) \\
2+0: (0\ 1\ 1\ 0\ 0\ 0\ 0) \cdot M_3 &\xrightarrow{2} (1\ 0\ 0\ 1\ 0\ 0\ 1\ \mathbf{0\ 0\ 1\ 0\ 0}) \\
2+1: (0\ 1\ 0\ 0\ 0\ 1\ 0) \cdot M_3 &\xrightarrow{2} (1\ 0\ 0\ 0\ 1\ 0\ 0\ \mathbf{0\ 0\ 0\ 1\ 0}) \\
2+2: (1\ 0\ 0\ 0\ 0\ 1\ 0) \cdot M_3 &\xrightarrow{2} (1\ 0\ 0\ 0\ 0\ 1\ 0\ \mathbf{0\ 0\ 0\ 0\ 1}) \\
&\quad 01\ 02\ 10\ 11\ 12
\end{aligned}$$

Die Ergebnisse der Rechnungen mit Übertrag sind an den Bits an den Positionen 8 bis 12 abzulesen. Steht eine 1 an Position 8, dann ist das Ergebnis 01. Steht an der neunten Position eine 1, dann lautet das Ergebnis 02 und so fort. Die jeweilige Bedeutung wurde unterhalb der Spalten notiert.

Mit Hilfe von M_3 soll zum Beispiel die Summe von 112 und 201 bestimmt werden. Mit den Einern wird begonnen, ein Übertrag liegt noch nicht vor, also wird wie folgt abgefragt:

Beispielrechnung zur Addition

$$2+1: (0\ 1\ 0\ 0\ 0\ 1\ 1) \cdot M_3 \xrightarrow{3} (1\ \mathbf{0\ 0\ 0\ 1\ 0}\ 0\ 0\ 0\ 0\ 0\ 0) \hat{=} 10$$

Der Antwort entnimmt man, dass die letzte Stelle des Ergebnisses gleich 0 ist und sich ein Übertrag ergeben hat. Zum Addieren der Dreier, also der vorletzten Stelle, ist folglich anzusetzen mit:

$$1+0: (0\ 0\ 1\ 0\ 1\ 0\ 0) \cdot M_2 \xrightarrow{2} (1\ 0\ 1\ 0\ 0\ 0\ 1\ \mathbf{0\ 1\ 0\ 0\ 0}) \hat{=} 02$$

Zuletzt sind demzufolge die Neuner ohne Übertrag zu verrechnen:

$$1+2: (0\ 0\ 0\ 0\ 1\ 1\ 1) \cdot M_3 \xrightarrow{3} (1\ \mathbf{0\ 0\ 0\ 1\ 0}\ 0\ 0\ 0\ 0\ 0\ 0) \hat{=} 10$$

Da diese letzte Matrixabfrage einen Übertrag liefert, der für einen 27er steht, setzt sich die Lösung aus den Ergebnisziern und dem letzten Übertrag korrekt zu 1020 zusammen.

Die Matrix M_3 wurde nicht durch einen Lernvorgang gewonnen, sondern eigens für ihren Zweck konstruiert. Algebraische Eigenschaften wie die Kommutativität spielten dabei keine Rolle. Vielmehr gelingt durch die Konstruktion ein zusätzlicher Nutzen. Mit der Matrix M_3 lässt sich auch subtrahieren, indem man lediglich die Differenzen den Summen wie in Tabelle 2.5 zuordnet und dann die Matrix M_3 mit der Kodierung dieser Summen aus Tabelle 2.4 abfragt. Diese Möglichkeit ist dem isomorphen Aufbau der Verknüpfungstafeln von Addition und Subtraktion zu verdanken, was man erkennt, wenn man diese wie in den Abbildungen 2.51 und 2.52 anordnet.

Addition und
Subtraktion mit
derselben Matrix

-	0	1	2
2	02	01	00
1	01	00	12
0	00	12	11

Abb. 2.51: Subtraktion ohne Übertrag

-	0	1	2
2	01	00	12
1	00	12	11
0	12	11	10

Abb. 2.52: Subtraktion mit Übertrag

2 - 0	2 - 1	2 - 2	1 - 0	1 - 1	1 - 2	0 - 0	0 - 1	0 - 2
0+0	0+1	0+2	1+0	1+1	1+2	2+0	2+1	2+2

Tabelle 2.5: Zuordnung der Differenzen zu den Summen

Beispielrechnung zur
Subtraktion

Als Beispiel soll die Differenz von 1020 und 201 gebildet werden. Mit den Einern wird wieder begonnen, ein Übertrag liegt nicht vor, also wird laut Tabelle 2.5 wie bei 2+1 abgefragt:

$$0 - 1: (0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1) \cdot M_3 \xrightarrow{3} (1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0) \hat{=} 12$$

Zum Ergebnis 12 kommt es, weil in der Verknüpfungstabelle der Subtraktion an der Stelle, an der bei der Addition eine 10 steht, nun eine 12 abzulesen ist. Die nächste Abfrage ist nun mit Übertrag auszuführen:

$$2 - 0: (1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0) \cdot M_3 \xrightarrow{2} (1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0) \hat{=} 01$$

Die Differenz 0-2 ist danach mit der Kodierung für 2+2 ohne Übertrag abzufragen:

$$0 - 2: (1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1) \cdot M_3 \xrightarrow{3} (1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0) \hat{=} 11$$

Im letzten Schritt wird wegen des gelieferten Übertrags nun noch 1-1 bestimmt:

$$1 - 1: (1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0) \cdot M_3 \xrightarrow{2} (1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0) \hat{=} 00$$

Damit liegt das Ergebnis 112 korrekt vor.

Mit Assoziativmatrizen ist die Ausführung weiterer mathematischer Operationen möglich. Die Spanne reicht von der Auswertung aussagenlogischer Terme bis zur Darstellung von und Operationen auf Bäumen und Graphen.⁷¹

2.6 Übungen 2

Ü 2.1 (Assoziativmatrix)

Man betrachte die Assoziativmatrix in Abb. 2.28. Schwellwert aller Abfragen sei stets $\hat{\Theta}$.

- Welche Antwort ergibt sich, wenn man bei einer Abfrage die zweite und sechste Zeile aktiviert?
- Die Abfrage der Assoziativmatrix mit einer Frage, die zwei Einsen enthält, brachte die Antwort $(0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$. Welche Zeilen sind aktiviert worden?
- Es wurden die 3. bis 5. Zeile aktiviert, so dass sich folglich die Antwort $(0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0)$ ergab. Ändert sich die Antwort, wenn man auch noch die zweite oder siebte Zeile aktiviert?
- Man gebe die Assoziativmatrix als Verbindungsmatrix V an und multipliziere diese mit der Frage $(1\ 1\ 0\ 0\ 0\ 0\ 1\ 1)$.

Ü 2.2 (Assoziativmatrix)

Gegeben seien folgende Frage-Antwort-Paare (f_i, a_i) :

$$f_1 = (0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0)$$

$$a_1 = (0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0)$$

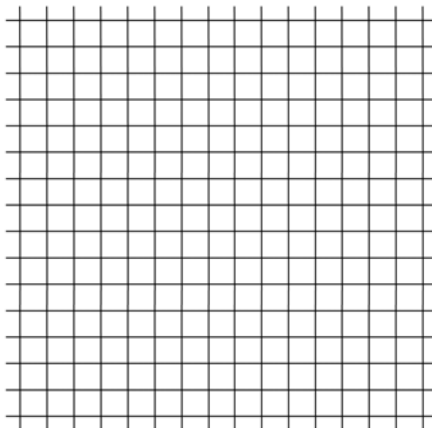
$$f_2 = (0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0)$$

$$a_2 = (1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$$

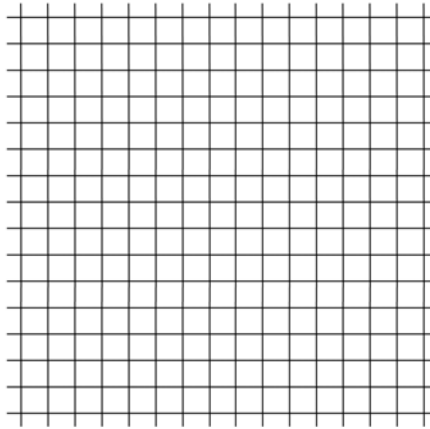
$$f_3 = (0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0)$$

$$a_3 = (0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0)$$

- Nach dem Eintragen der Frage-Antwort-Paare (f_i, a_i) in eine Assoziativmatrix L mit der L-Lernregel trage man deren Belegung in folgende Gitter-Darstellung der Matrix L ein.



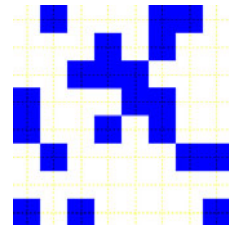
- b) Man trage die Frage-Antwort-Paare mit der K-Lernregel in folgende Assoziativmatrix K ein. Anschließend gebe man die Spaltensummen an, die bei Abfrage von K mit den Fragen f_i entstehen.



Ü 2.3 (Assoziativmatrix)

Eine Assoziativmatrix A sei durch nebenstehende Abbildung gegeben. Man berechne und vergleiche:

- $a_1 = (1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0) \cdot A$
- $a_2 = ((1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0) \cdot A) \cdot A$
- $a_3 = (((1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0) \cdot A) \cdot A) \cdot A$
- $a_4 = (1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0) \cdot A^2$
- $a_5 = (1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0) \cdot A^3$



Ü 2.4 (Assoziativmatrix)

Wenn eine Schokolade süß, sahnig und nach Joghurt schmeckt, dann wurde sie von der Firma *Jogurama* produziert. Ist sie krümelig und schmeckt etwas bitter und nach Kaffee, stammt sie von *Caffètti*. Und wenn ihr Geschmack an frische Erdbeeren erinnert, sie süß ist und ebenfalls krümelig, so wurde sie in einer Fabrik von *Fruttello* hergestellt.

- Man überlege eine Kodierung, durch die die Sinneseindrücke auf die Schokoladenmarke abgebildet werden.
- Man trage die Frage-Antwort-Paare mit der L-Lernregel in eine Assoziativmatrix L ein.
- Man überprüfe, ob bei einer Anfrage nach einer sahnig-süßen Schokolade mit *Jogurama* geantwortet wird.
- Man notiere, wie die Matrix L auf folgende Anfragen antwortet:
 - frische Erdbeeren, süß
 - sahnig, frische Erdbeeren, süß
 - bitter, Joghurt, Kaffee
 - sahnig, krümelig

2.7 Assoziative Quader

Fügt man mehrere Assoziativmatrizen zu einem mehrschichtigen Gebilde zusammen, dann gewinnt man eine zusätzliche Antwortebene. In Schichten angeordnete Assoziativmatrizen werden **assoziative Quader** genannt. In Abbildung 2.53 befindet sich vorn in der ersten Schicht die erste Assoziativmatrix, bestehend aus drei Zeilen und vier Spalten, die weiteren Matrizen in vier Schichten dahinter.

Schichten aus
Assoziativmatrizen

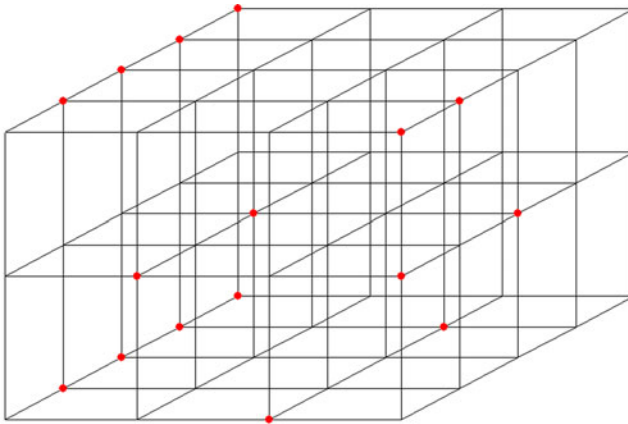
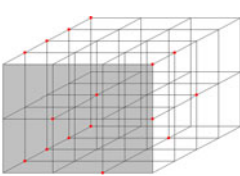


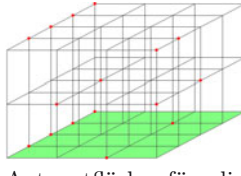
Abb. 2.53: Ein assoziativer Quader mit drei Zeilen, vier Spalten und fünf Schichten

Beim Abfragen eines assoziativen Quaders wird eine Fragefläche, also eine Matrix aus Nullen und Einsen, an die erste Schicht des Quaders angelegt. Alle Einsen wirken nun nicht nur auf die erste Schicht, sondern auch auf alle Schichten dahinter. Der assoziative Quader lässt sich dann senkrecht oder waagrecht abfragen, wobei die Schwellwerte zum Boden oder zur Seite hin gebildet werden, wie nachstehende Abbildung veranschaulicht.

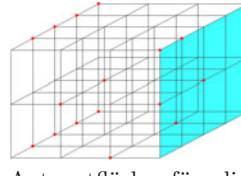
senkrechte und
waagerechte Abfrage



Fragefläche

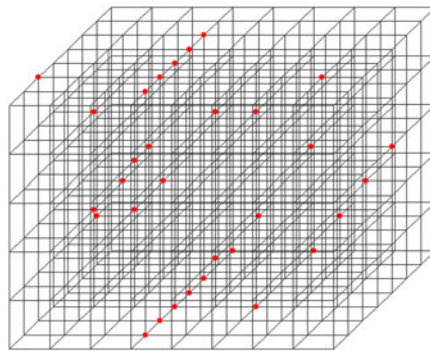


Antwortfläche für die
senkrechte Abfrage

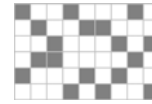


Antwortfläche für die
waagerechte Abfrage

Im folgenden Beispiel wurde eine Fragefläche mit einer Antwortfläche für die senkrechte und eine weitere für die waagerechte Abfrage in den Quader jeweils mit Hilfe der L-Lernregel eingetragen (s. Kapitel 2.5). Ein Computeralgebrasystem wie *Maxima* unterstützt bei der rechnerischen Bestimmung der Schwellwerte der Antwortflächen.⁷² In die nachstehend abgebildeten Antwortflächen wurden die Schwellwerte mit eingetragen. Ob man bei der Abfrage den in einer Antwortfläche größten aufgetretenen Schwellwert benutzt oder den jeweils in einer Spalte oder Zeile größten oder auch einen in einer gewissen Umgebung aufgetretenen größten Schwellwert, das bleibt dem Anwender und Anwendungsfall überlassen.



Über die L-Lernregel gefüllter assoziativer Quader



Fragefläche

0	0	0	1	0	1	0	0	1
1	0	0	0	1	0	1	0	0
0	1	1	2	0	0	0	1	0
0	0	1	2	0	0	1	0	1
0	2	2	2	2	2	0	0	0
1	0	0	2	0	0	0	1	0
0	0	0	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Antwortfläche nach der senkrechten Abfrage

0	1	3	1	1	1	0	0
0	0	0	3	0	0	0	0
0	0	0	1	3	0	0	0
0	0	0	2	0	3	0	0
0	0	0	1	0	0	3	0
0	1	1	2	1	1	0	3

Antwortfläche nach der waagerechten Abfrage

Lage der
Antwortflächen

Die Abbildung 2.54 macht am vorstehenden Beispiel die Lage der Antwortflächen sichtbar. Am Boden des Quaders erkennt man die Antwortfläche für die senkrechte Abfrage und rechts diejenige für die waagerechte Abfrage.

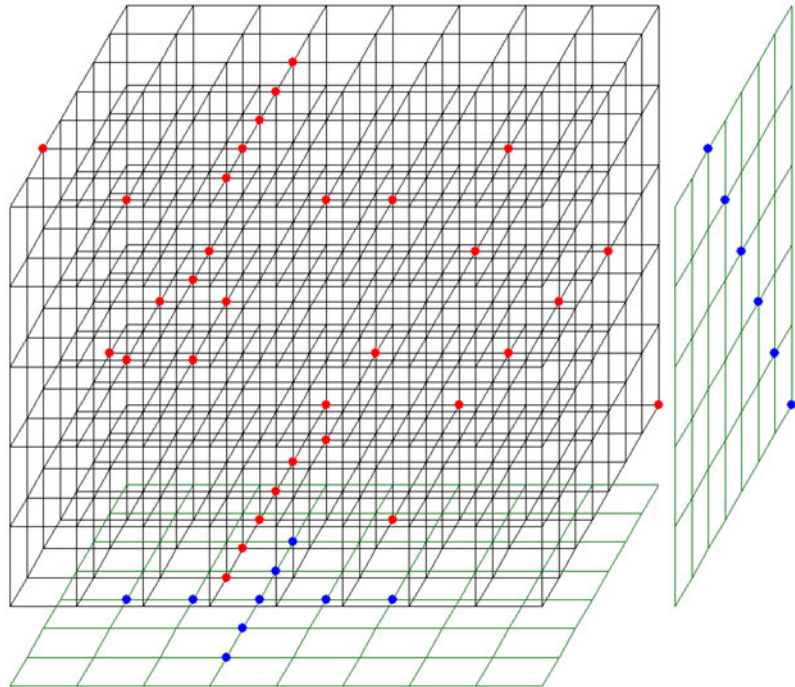


Abb. 2.54: Lage der Antwortflächen

Auch beim assoziativen Quader finden sich die von den Assoziativmatrizen her bekannten Eigenschaften der Mustererkennung, Mustervervollständigung und Musterextraktion, was man sich bei größeren Quadern gut sichtbar machen kann, denen man Fotografien auf die Frageflächen aufträgt. Dazu ist bei-

spielsweise ein Quader mit 150 Spalten und ebenso vielen Zeilen und Schichten gewählt worden. Folgende schwarzweiße Bilder aus B26 wurden in den Quader als Frage- und Antwortflächen eingetragen.



Fragefläche mit einem schwarzweißen Bild



Antwortfläche für die senkrechte Abfrage



Antwortfläche für die waagerechte Abfrage

B26

Beim Abfragen des assoziativen Quaders mit dem Bild auf der vorstehenden Fragefläche ergeben sich folgende Antworten, die wegen weniger „falscher Einsen“ (s. Kapitel 2.5.4) nur geringe Abweichungen von den gelernten Flächen aufzeigen:



Antwortfläche nach der senkrechten Abfrage



Antwortfläche nach der waagerechten Abfrage

Folgende gelochte oder halbierte Bilder werden trotz ihrer Unvollständigkeit mit den gleichen beiden Antwortflächen wie vorstehend beantwortet.

störunanfällige Abfragen



Abfrage mit zweifach gelochtem Bild



Abfrage mit senkrecht halbiertem Bild



Abfrage mit waagerecht halbiertem Bild

Auch gegen ein Verschieben eines Bildes auf der Fragefläche lässt der assoziative Quader hier ein tolerantes Verhalten erkennen, wenngleich die Abweichungen zu den gelernten Antworten größer werden, wie folgende Abbildungen des Beispiels B27 vom Antwortverhalten des Quaders zeigen. In den

Bildern rechts sind die jeweiligen Abweichungen als Unterschiede zur gelernten Antwort eingezeichnet.

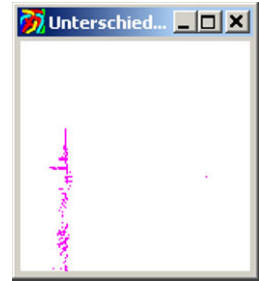
B27



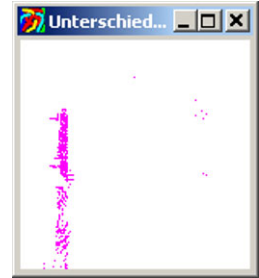
Frage



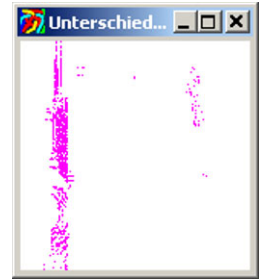
Antwort

Unterschied zur ge-
lernten AntwortFrage um 5 Pixel nach
links oben verschoben

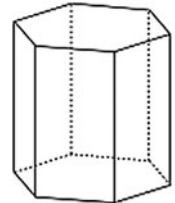
Antwort

Unterschied zur ge-
lernten AntwortFrage um 10 Pixel nach
links oben verschoben

Antwort

Unterschied zur ge-
lernten Antwort

Vorteil assoziativer Quader ist, dass bei einer Abfrage gleichzeitig in zwei Richtungen Antworten entstehen, jedes Neuron innerhalb dieser Struktur also mehrfach wirksam wird, was vermuten lässt, dass sich im Quader mehr Information unterbringen lässt als in einschichtig hintereinander gelegten oder gerollten Matrizen. Bei einem Sechseckprisma ergebe sich eine weitere Abfrage-richtung, bei einem Achteckprisma wären es schon vier und so fort.



Die Prismen bieten sich hier als assoziative Gebilde an, da als Fragefläche die Bodenfläche dienen kann, auf der die Abfrage-Seitenflächen senkrecht stehen.

Durch die Nutzung assoziativer Gebilde in mehreren Richtungen entstehen beim Lernen mehr „falsche Einsen“, was sich in folgendem Beispiel B28 augenfällig bemerkbar macht. In der oberen Zeile sind die beiden mit der Frage verbundenen Antworten wiedergegeben, in der Zeile darunter erscheint die senkrechte Antwort demgegenüber fehlerfrei, da als waagerechte Antwort eine leere Fläche gelernt wurde, so dass sich die Anzahl falscher Einsen verminderte.

falsche Einsen



Frage



senkrechte Antwort



waagerechte Antwort

B28



Frage



senkrechte Antwort



waagerechte Antwort

In Fortführung der Konstruktion eines assoziativen Quaders als Matrix von Matrizen gelangt man zu höherdimensionalen assoziativen Quadern, die zu einer Frage ebenfalls mehr als zwei Antworten liefern können.⁷³ Bei der Nutzung dieser assoziativen Quader werden die Kapazitätsüberlegungen, die für verschiedene Matrixmodelle vorliegen (s. [Knoblauch et al. 2010]) vermutlich zu modifizieren sein, eine Arbeit, die noch geleistet werden muss. Weil mit dem Lernen in den senkrechten Schichten unweigerlich auch Einträge für die waagerechte Abfrage definiert werden (und umgekehrt), könnte man diese Verknüpfung auch als Komposition oder Verschränkung von Nutzinformation mit Metainformation interpretieren und damit ein Modell für Bewusstsein gewinnen.⁷⁴

höherdimensionale
assoziative Quader

Anmerkungen

¹Das menschliche Gehirn besteht aus mindestens zehn Milliarden Nervenzellen (Neuronen) (vgl. [Palm 1988], S. 54, oder [Abdi 1994], S. 7). Jede Nervenzelle leitet ein Ausgangssignal über ihr Axon an mehrere zehntausend andere Nervenzellen weiter (s. ebenda).

zehn Milliarden
Nervenzellen

²In [Palm 1988], S. 55 f., schätzt der Autor des Weiteren ab, dass man zudem etwa 10²⁰ Werte für die „Anfangsverknüpfungen zwischen den Nervenzellen“ festzulegen, also den möglichen Einfluss jeder Nervenzelle auf jede andere zu bemessen habe.

hundert Trillionen
Startwerte

³Das Zitat stammt aus [Neumann 1970], S. 77.

⁴Leider brechen die Aufzeichnungen von John von Neumann (1903 – 1957) an dieser Stelle wegen seiner schweren Erkrankung ab.

⁵In [Dehaene 1999], S. 17, schreibt Stanislas Dehaene dazu: „Man kann nicht umhin, die Flexibilität des menschlichen Gehirns zu bewundern, die es je nach Umwelt und Epoche erlaubt, eine Jagd auf Mammut zu planen oder sich einen Beweis für die Fermatsche Vermutung auszudenken. Diese Flexibilität sollte dennoch nicht überschätzt werden. Ich behaupte, dass es genau die Vor- und Nachteile unserer zerebralen Schaltkreise sind, die die Stärken und Grenzen unserer mathematischen Fähigkeiten bestimmen. Unser Gehirn wurde wie das der Ratte vor undenklichen Zeiten mit der intuitiven Repräsentation für Anzahlen ausgestattet. Deshalb sind wir so begabt für Näherungen, und deshalb erscheint es uns so offensichtlich, dass 10 größer ist als 5.“

Aufgaben für die
Neuromathematik

⁶In [Galushkin et al. 2003] zählen die Autoren als grundlegende mathematische Aufgaben, die Gegenstand der Neuromathematik seien, die folgenden auf: Systeme linearer und nichtlinearer Gleichungen, Integralgleichungen, gewöhnliche und partielle Differentialgleichungen, Systeme linearer Ungleichungen, Matrizenoperationen, Optimierungsaufgaben, Approximation, Extrapolation u.a.m.

⁷Im *Neuromath Network* tauschten Mitarbeiter von sechs europäischen Instituten ihre Ergebnisse aus, darunter Stanislas Dehaene („Der Zahlensinn“, s. [Dehaene 1999]).

⁸Der Beitrag [Galushkin et al. 2003] wird beschrieben als „summary of a set of Russian scientists’ works [...] in the field of neuromathematics — neural network algorithms for solving mathematical tasks.“

Neuromathematik zur
Beschreibung von
Vorgängen im Gehirn

⁹R. Moreno-Diaz jr. und K. N. Leibovic nutzen beispielsweise in [Moreno-Diaz and Leibovic 1995] den Begriff Neuromathematik im angegebenen Sinne und zeigen auf, wie sich von der beobachteten Funktionsweise eines Zellverbunds auf dessen Struktur schließen lässt. Nach einer Buchbeschreibung des Verlags zu [Scott 2002] lassen sich mathematische Verfahren in folgenden sechs Punkten in die Neurowissenschaften einbringen: (i) zur exakten Beschreibung gut verstandener Prozesse in und zwischen den Nervenzellen, (ii) zur bildhaften Beschreibung komplexerer Phänomene der Hirntätigkeit, (iii) für informationstheoretische Überlegungen zu Speicherkapazität und Datenübertragung, (iv) für aussagenlogische Betrachtungen zur Analyse der Informationsverarbeitung, (v) für zahlen-theoretische Überlegungen zu großen Anzahlen an Kombinationsmöglichkeiten und (vi) für statistische Verfahren zur Organisation und Auswertung von Messwerten.

Maschine für Abbildun-
gen zwischen 0-1-Folgen

¹⁰Bei [Palm 1982], S. 26, heißt es dazu: „A behavior can be defined as a mapping associating outputs (actions) to inputs (situations). The input-output-signals can be coded into 0,1-sequences. In this way the problem of constructing a machine that displays a pre-described behavior is reduced to the problem of constructing a machine that performs a pre-described mapping between 0,1-sequences.“

Hodgkin-Huxley-Modell

¹¹Das Hodgkin-Huxley-Modell des Neurons wurde 1952 von Alan Lloyd Hodgkin (1914-1998) und Andrew Fielding Huxley (*1917) entwickelt. Für ihre Entdeckungen zu den verschiedenen elektrischen Potentialen an der Nervenzelle erhielten sie zusammen mit John Carew Eccles 1963 den Nobelpreis für Physiologie oder Medizin. In der Abb. 2.2 befindet sich die Nachbildung des durch Kaliumionen entstehenden Ionenstroms ganz links, daneben der gegenläufige, welcher durch Natriumionen entsteht. Als dritter Strom trägt der Leckstrom (dritter Leitungsweg) und als vierter der durch die außen angelegte Erregungsspannung verursachte Strom zum Gesamtstrom bei, der durch die Nervenzelle fließt. Der Kondensator ganz rechts bildet durch sein Auf- und Entladen die typischen zeitlichen Abläufe nach.

KTechLab

¹²Die Simulation des Modellneurons nach Hodgkin-Huxley ist hier mit dem Programm *KTechLab* unter Linux vorgenommen worden.

Integrationsgrad

¹³Solche Projekte sind in Anmerkung 22 von Kapitel 1 vorgestellt worden.

¹⁴Mit dem Integrationsgrad „Very Large Scale Integration (VLSI)“ ist zum Beispiel meist eine sechsstellige Anzahl von Transistoren pro integriertem Schaltkreis gemeint. In modernen Prozessoren werden über 500 Millionen Transistoren eingesetzt. Ulrich Rückert geht zum Beispiel in [Rückert 1991] auf den Bau von Assoziativmatrizen in VLSI-Technik ein.

¹⁵Ulrich Rückert beschreibt unter anderem in [Rückert 1990], [Rückert 1991], [Rückert 1993] und [Rückert 1994] technische Möglichkeiten zum Bau von Assoziativmatrizen und neuronalen Netzen mit digitalen und analogen elektronischen Bauteilen.

¹⁶Die Bezeichnung w für die Verbindungsstärke leitet sich vom englischen Wort *weight* (Gewicht, Einfluss) ab. Daher wird die Verbindungsstärke auch Verbindungsgewicht genannt.

Verbindungsgewicht

¹⁷Bei wiederholten Übertragungen von einem Neuron zu einem anderen werden die betroffenen Verbindungsgewichte größer (s. [Hebb 1949]).

synaptische Plastizität

¹⁸Zum Rechnen mit Matrizen lese man z. B. in [Sperner 1951], §9, nach.

¹⁹Dazu schreibt [Palm 1988], S. 57: „Wie weit diese Fähigkeit [der Informationsweitergabe] bei einzelnen Neuronen reicht ist noch nicht vollständig erforscht, aber es kann als sicher gelten, dass Nervenzellen wenigstens zu einer (nichtlinearen) Schwellenoperation fähig sind: Sie reagieren erst dann, wenn die von anderen Neuronen eingehende Erregung insgesamt einen gewissen Wert übersteigt“. Im grundlegenden Artikel [McCulloch and Pitts 1943], S. 115, halten die Autoren fest: „The nervous system is a net of neurons, each having a soma and an axon. Their adjunctions, or synapses, are always between the axon of one neuron and the soma of another. At any instant a neuron has some threshold, which excitation must exceed to initiate an impulse.“

Schwellenoperation

²⁰In [Palm 1982], S. 26, wird unter der Überschrift *How to build well-behaving machines* dazu geführt: „Such a machine can indeed be constructed from 'logical machines' that correspond to the logical operations of 'not', 'and' and 'or'. It can also be constructed from 'threshold neurons', which are neuron-like machines.“

UND-, ODER-,
NICHT-Neuronen für
Logikmaschinen

²¹Die Beschreibung des Perzeptrons geht auf Frank Rosenblatt in [Rosenblatt 1958] zurück. Dem Psychologen und Informatiker Frank Rosenblatt (1928-1971) gelang 1960 auf der Grundlage seines Perzeptron-Modells an der Cornell University der Bau eines ersten lernfähigen Rechners.

Frank Rosenblatt

²²Alle Neuronen der Eingabeschicht erhalten hier genau einen Eingabewert. Daher enthält die zugehörige Verbindungsmatrix nur in der Hauptdiagonalen Werte ungleich null. In der Abb. 2.55 aus [Shaw and Palm 1988], S. 522, entspricht das der Weiterleitung der Signale von den „S-Units“ der Retina zu den „A-Units“ der „association area“.

Eingabeschicht eines
Perzeptrons

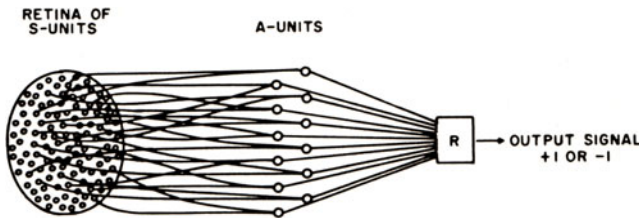


Abb. 2.55: Typical Elementary Perceptron (Rosenblatt 1962)

²³Mit dem unter der „GNU General Public License“ entwickelten, frei verfügbaren Computeralgebrasystem (CAS) *Maxima* lässt sich die Eingabe der Matrizen V_1, V_2, V_3 , die Eingabe der Frage x und die Rechnung wie folgt ausführen:

Computeralgebrasystem
Maxima

```
v1: matrix(      v2: matrix(      v3: matrix(      x: matrix(
  [-1,0],        [4,1,-2],        [3,-5],        [-2,7]
  [0,2]          [2,-3,6]         [-1,2],         );
);              );              );
theta: 1;
f(x):= if x >= theta then 1 else 0;
tmp: matrixmap(f,x.v1);
tmp: matrixmap(f,tmp.v2);
matrixmap(f,tmp.v3);
```

In der Variablen `tmp` werden im Verlauf der Rechnung die Zwischenergebnisse abgelegt. Die Befehle zur Eingabe der Werte für die Matrizen `v1`, `v2`, `v3` und `x` schreibt man nicht wie hier im Druck nebeneinander sondern untereinander. Die Zuordnung f wird in diesem Beispiel durch den Befehl `matrixmap` auf alle Einträge der zu berechnenden Matrixprodukte angewandt.

wxMaxima

²⁴*Maxima* wurde für die Beispiele dieses Buches unter der grafischen Oberfläche *wxMaxima* aufgerufen.

Zählung der Lagen eines Perzeptrons

²⁵Die Zählung der Schichten oder Lagen des Perzeptrons wird unterschiedlich vorgenommen. So wird gelegentlich von einem **einlagigen** Perzeptron gesprochen, wenn dieses nur aus Eingabe- und Ausgabeschicht besteht, ein **zweilagiges** Perzeptron besitzt eine Zwischenschicht („hidden layer“), ein dreilagiges dann zwei und so fort. Rosenblatt schreibt demgegenüber in [Shaw and Palm 1988], S. 522: „The simplest perceptron which is capable of full generality in any discrimination experiment, calling for an arbitrary dichotomy of the set of all possible visual input patterns, is a **three-layer** network called a *simple perceptron*. Such a network is illustrated in [Abb. 2.55]. It consists of a layer of sensory points, a single layer of A-units, and a single R-unit, which emits an output of +1 if the total input signal is strictly positive, or -1 if the total input signal is negative.“ Rosenblatt zählt hier also sogar die sensorische Schicht mit. Mit „S-units“ bezeichnet er „sensory units“ (der Netzhaut), mit „A-units“ benennt er „association cells“ und mit „R-units“ die „response“-Zellen (vgl. [Rosenblatt 1958], S. 389 f.).

Feedback-Verfahren beim Perzeptron

²⁶Schon in [Rosenblatt 1958], S.390, beschreibt Frank Rosenblatt jedoch Feedback-Verfahren, mit denen Einfluss auf die „association area“ genommen wird. Die von ihm dort zur Illustration eingefügte Abb. 2.56 deutet die Rückkopplung durch einen Doppelpfeil an.

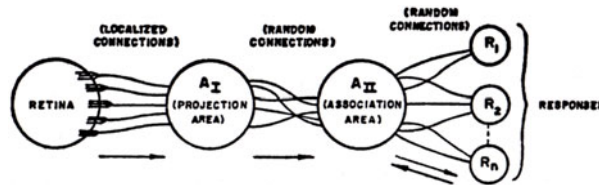


Abb. 2.56: Organization of a perceptron (Rosenblatt 1958)

Er erläutert das Rückkopplungsverfahren mit: „Note that up to A_{II} all connections are forward, and there is no feedback. When we come to the last set of connections, between A_{II} and the R-units, connections are established in both directions. The rule governing feedback connections, in most models of the perceptron, can be either of the following alternatives: (a) Each response has excitatory feedback connections to the cells in its own source-set, or (b) Each response has inhibitory feedback connections to the complement of its own source-set (i.e., it tends to prohibit activity in any association cells which do not transmit to it)“. Mit „its own source-set“ sind diejenigen A-units gemeint, die Signale an die jeweilige Antwortzelle senden.

Pixel

²⁷das Pixel — engl. Abk. für picture element, Bildpunkt

Minsky, Papert, XOR-Problem

²⁸Marvin Minsky und Seymour Papert zeigten 1969 (vgl. [Minsky and Papert 1988]), dass ein einlagiges Perzeptron beispielsweise nicht in der Lage ist, die aussagenlogische XOR-Verknüpfung darzustellen. Für die UND-, ODER- und NICHT-Verknüpfung wurden auf Seite 19 Verknüpfungsmatrizen angegeben, für die XOR-Verknüpfung (exklusives ODER, ENTWEDER-ODER-Verknüpfung) wäre eine Lösung durch folgende Matrizen V_1 und V_2 gegeben:

$$V_1 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}_1, \quad V_2 = \begin{pmatrix} -1 \\ 2 \\ -1 \end{pmatrix}_1$$

Verbunde kleiner, spezialisierter Netzwerke

²⁹In [Minsky and Papert 1988], Vorwort S. xv, heißt es dazu: „But, as we explain in our epilogue, the marvelous powers of the brain emerge not from any single, uniformly structured connectionist network but from highly evolved arrangements of smaller, specialized networks which are interconnected in very specific ways.“ Und im Nachwort folgt auf S. 270: „We recognize that the idea of distributed, cooperative processing has a powerful appeal of common sense as well to computational and biological science.“

Assoziatives Netz von Willshaw et al.

³⁰David J. Willshaw, O. Peter Bunemann und H. C. Longuet-Higgins erläutern in [Willshaw et al. 1969] die Lernregel für ihr Modell durch Abb. 2.57. Die darin eingetragenen Muster zeigt Abb. 2.58. Diese Lernregel wurde tiefergehend von Günther Palm untersucht, speziell die Auswirkungen auf die Informationskapazität des Speichers (s. [Palm 1980]).

Dabei wies er auf die besondere Bedeutung der Spärlichkeit der Kodierung hin, also die Verwendung von sehr wenigen Einsen und vielen Nullen in den Frage-Antwort-Paaren.

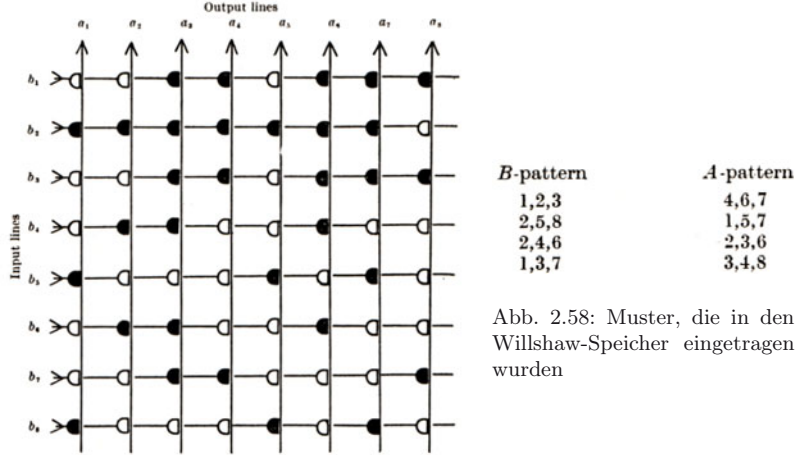
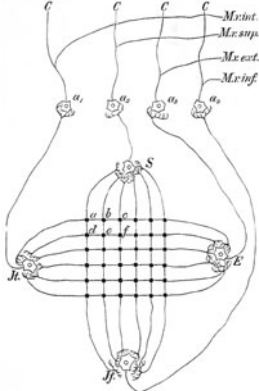


Abb. 2.57: Nicht-holografischer Speicher von Willshaw, Bunemann und Longuet-Higgins

Ein Zusammenwirken von Neuronen durch einen matrixförmigen Anschluss an „empfindende Netzhauptelemente“ beschreibt der Physiologe Sigmund Exner (1846-1926) bereits 1894. Er nimmt dabei an, dass „exquisite Summationszellen“ (Zellen E und S) der Bewegungswahrnehmung dienen, indem diese die von der Netzhaut kommenden Signale zeitlich auswerten (s. [Exner 1894], S. 192 ff.).



Matrixförmige Zusammenarbeit von Neuronen (Sigmund Exner, 1894)

³¹Die **L**-Lernregel wurde hier so benannt, weil sie zum Füllen des **L**angzeitgedächtnisses und des Programmspeichers der Assoziativmaschine **SYSTEM 9** eingesetzt wird.

L-Lernregel

³²Der Physiker und Biologe John Joseph Hopfield (geb. 1933 in Chicago) veröffentlichte 1982 in [Hopfield 1982] seine Idee eines inhaltsadressierbaren Speichers aus rückgekoppelten Modellneuronen. Laut [Hecht-Nielsen 1991], S. 19, erreichte Hopfield damit eine starke Wiederbelebung der wissenschaftlichen Beschäftigung mit neuronalen Netzen: „Hopfield wrote two highly readable papers on neural networks in 1982 (s. [Hopfield 1982]) and 1984 (s. [Hopfield 1984]) and these, together with his many lectures all over the world, persuaded hundreds of highly qualified scientists, mathematicians, and technologists to join the emerging field of neural networks. In fact, by the beginning of 1986, approximately one-third of the people in the field had been brought in directly by Hopfield or by one of his earlier converts. Hopfield’s work as a recruiter was perhaps the single most important contribution to the early growth of the revitalized field.“

John Joseph Hopfield

³³In [Hopfield 1982] schreibt der Autor auf S. 2555: „Suppose we wish to store the set of states $V^s, s = 1 \dots n$. We use the storage prescription [...] $T_{ij} = \sum_s (2V_i^s - 1)(2V_j^s - 1)$ but with $T_{ii} = 0$.“

Hopfield-Lernregel

³⁴Mit Hilfe des CAS *Maxima* lässt sich die Verbindungsmatrix V durch nachstehende Befehlsfolge erzeugen:

```
m_anz: 3; m[1]: [0,0,1,1,0]; m[2]: [0,1,1,0,0]; m[3]: [1,0,0,0,1];
gewicht(i,j) := sum((2*m[k][i]-1)*(2*m[k][j]-1),k,1,m_anz);
w[i,j] := if i = j then 0 else gewicht(i,j);
N: length(m[1]);
v: genmatrix(w,N,N);
```

³⁵Mit dem CAS *Maxima* erhält man zum Beispiel nach Abfrage mit dem Muster m_1 durch `matrix([0,0,1,1,0]).v;` das Ergebnis `[-4 0 1 1 -4]` und nach Berücksichtigung des Schwellwerts wird `[0 0 1 1 0]` geliefert, also nach Abfrage mit m_1 ergibt sich wiederum m_1 .

³⁶Der finnische Ingenieur Teuvo Kohonen (*1934) forscht unter anderem zu den Themen Selbstorganisation, Künstliche Neuronale Netze und Assoziativspeicher (s. [Kohonen 1987],

Teuvo Kohonen

[Kohonen 1988], [Kohonen 2001]).

SOM

³⁷Selbstorganisierende Karten werden abgekürzt als SOM (self-organizing **map**) oder SOFM (self-organizing feature map) geschrieben und auch Kohonennetze genannt.

Kohonenlernregel,
Lernrate η

³⁸Eine einfache Kohonenlernregel ist für ein Neuron w und einen Eingabereiz e beispielsweise $w_{neu} = w_{alt} + \eta \cdot (e - w_{alt})$. Dabei ist $0 < \eta < 1$ eine geeignete Lernrate und w dasjenige Neuron, welches den kleinsten euklidischen Abstand zum Eingabereiz e zeigt. Durch die Kohonenlernregel wird also das Neuron w ein Stück zum Eingabereiz e „hinbewegt“. Die Eingabereize werden dem neuronalen Netz in zufälliger Reihenfolge vorgelegt. Um nicht nur dasjenige Neuron w zu „bewegen“, welches zum Eingabereiz den kleinsten Abstand hat, wird eine Nachbarschaftsfunktion ϕ genutzt, durch die auch die in der Nähe von w liegenden Neuronen v etwas in Richtung auf e mitbewegt werden. Die zugehörige Kohonenlernregel wäre damit $w_{neu} = w_{alt} + \eta \cdot \phi(v, w) \cdot (e - w_{alt})$. Eine weitere Verfeinerung bieten Kohonenlernregeln, mit denen die Lernrate η oder die Nachbarschaftsfunktion ϕ von der Laufzeit t des Lernens abhängig gemacht wird: $w_{neu} = w_{alt} + \eta(t) \cdot \phi(t, w) \cdot (e - w_{alt})$.

³⁹Eine ausführlichere Darstellung zu den Themen Selbstorganisation und Kohonenkarten liefern [Kohonen 2001] und [Ritter et al. 1991].

Maxima-Skript für eine
Kohonenkarte mit
grafischen Ausgaben

⁴⁰Das CAS *Maxima* bietet für das gegebene Beispiel zum Aufbau eines Kohonennetzes aus 3x2 Neuronen einen mehr als ausreichenden Befehlssatz an. Zunächst werden hier die Laufzeit t_{max} , also die Anzahl der Lernschritte, dann die von der Laufzeit abhängige Lernrate η , die Eingabereize e , die Anzahl der Eingabereize e_{anz} festgelegt und für die spätere grafische Aufbereitung eine Liste der Eingabereize $eliste$ erzeugt:

```
t_max: 10000;
eta(t):= 0.75*(1-t/t_max);
e: matrix([0.2,0.8],[0.5,0.8],[0.1,0.5],[0.1,0.2],[0.3,0.1],[0.5,0.2]);
e_anz: length(e); eliste: [];
for m: 1 step 1 thru e_anz do eliste: append(eliste,[e[m]]);
```

Anschließend wird das Netz aus 3x2 Neuronen, die jeweils aus zwei Zufallswerten bestehen, die als Koordinaten für Orte in Feldmitte dienen, erzeugt. Dabei wird außer der *breite* und *hoehe* des Netzes auch die Anzahl der Neuronen v_anz und die Liste $uliste$ für die spätere grafische Ausgabe bestimmt:

```
zufall() := random(0.05)+0.45;
breite: 2; hoehe: 3; v_anz: breite*hoehe;
tmp: matrix([matrix([zufall(),zufall()])]);
for m: 2 step 1 thru breite do
  tmp: addcol(tmp,[matrix([zufall(),zufall()])]);
k: tmp;
for n: 2 step 1 thru hoehe do (
  tmp: matrix([matrix([zufall(),zufall()])]),
  for m: 2 step 1 thru breite do
    tmp: addcol(tmp,[matrix([zufall(),zufall()])]),
    k: addrow(k,tmp);
);
uliste: [];
for n: 1 step 1 thru hoehe do
  for m: 1 step 1 thru breite do
    uliste: append(uliste,[k[n,m][1]]);
```

Dann werden Funktionen und Konstanten für die Nachbarschaftsfunktion ϕ ins *Maxima*-Skript eingetragen. Für den zeitlich veränderlichen Nachbarschaftsradius, auch Adaptionsradius genannt, wird $radius(t) = r_{max} \cdot \left(\frac{r_{min}}{r_{max}}\right)^{\frac{t}{t_{max}}}$ gewählt, als Nachbarschaftsfunktion

Nachbarschaftsfunktion
 ϕ und Adaptionsradius

für die Neuronen u und v wird $\phi(t, u, v) = e^{-\left(\frac{d(u,v)}{radius(t)}\right)^2}$ angesetzt, wobei $d(u, v)$ den Abstand der Plätze der beiden Neuronen u und v im neuronalen Netz angibt. Die Funktion *euklid* berechnet den euklidischen Abstand zweier Orte in der Ebene.

```
euklid(x,y):= sqrt((x[1]-y[1])^2 + (x[2]-y[2])^2);
r_max: 3.0; r_min: 1.0;
radius(t):= r_max*(r_min/r_max)^(t/t_max);
d(x1,y1,x2,y2):= abs(x1-x2)+abs(y1-y2);
phi(t,x1,y1,x2,y2):= if d(x1,y1,x2,y2) <= radius(t) then
  exp(-(d(x1,y1,x2,y2)/radius(t))^2)
else 0;
```

Die Kohonenlernregel wird dann t_{max} Mal aufgerufen, wobei jeweils einer der Eingaberei-

ze e zufällig ausgewählt wird. Im ersten Teil des Lernalgorithmus wird dasjenige Neuron gesucht, welches zum Eingabereiz den kleinsten euklidischen Abstand hat. Danach werden die Werte dieses Neurons und die seiner Nachbarn in Richtung auf den Ausgabereiz hin „verschoben“:

```

for t: 1 step 1 thru t_max do (
  zuf: random(e_anz)+1,
  abst: 1, idx: 1, idy: 1,
  for n: 1 step 1 thru hoehe do (
    for m: 1 step 1 thru breite do (
      tmp: euklid(k[n,m][1],e[zuf]),
      if tmp < abst then (abst: tmp, idx: n, idy: m)
    )
  ),
  for n: 1 step 1 thru hoehe do
    for m: 1 step 1 thru breite do
      k[n,m][1]: k[n,m][1] +
        eta(t)*phi(t,n,m,idx,idy)*(e[zuf]-k[idx,idy][1])
);

```

Die folgenden *Maxima*-Befehle dienen zur grafischen Ausgabe. Zunächst werden in der Liste *kliste* die von den 3x2 Neuronen gelernten Orte eingetragen. Danach zeigt der Befehl *wxplot2d* die Listen *uliste* der anfänglichen Orte, *eliste* der Eingabereize und *kliste* der gelernten Orte an.

grafische Ausgabe

```

kliste: [];
for n: 1 step 1 thru hoehe do
  for m: 1 step 1 thru breite do
    kliste: append(kliste,[k[n,m][1]]);
wxplot2d([[discrete,uliste],[discrete,eliste]],
  [style,points],[legend,false],[x,0,1],[y,0,1]),
  wxplot_size=[500,500];
wxplot2d([[discrete,kliste],[discrete,eliste]],
  [style,points],[legend,false],[x,0,1],[y,0,1]),
  wxplot_size=[500,500];

```

Zuletzt wird im vorgelegten Beispiel überprüft, auf welche Reize das Kohonennetz in welcher Weise reagiert, indem ihm die Orte der Fläche in Hundertstelschritten vorgelegt werden. Dazu liefern Funktionen f und p die Koordinaten, die durch das Kohonennetz mit demjenigen Ort der Fläche assoziiert sind, mit dem abgefragt wird.

```

f(x):= ( abst: 1, idx: 1, idy: 1,
  for n: 1 step 1 thru hoehe do
    for m: 1 step 1 thru breite do (
      tmp: euklid(k[n,m][1],x),
      if tmp < abst then (abst: tmp, idx: n, idy: m)
    ),
  k[idx,idy][1]
);
p(x,y):= f([x,y]);

```

Danach wird in Zehntelschritten ermittelt, welche Anzahl an unterschiedlichen Antworten das Kohonennetz liefert. Die verschiedenen Antworten stehen anschließend in der Liste *pliste*.

```

pliste: [];
for x: 0 step 0.1 thru 1 do (
  for y: 0 step 0.1 thru 1 do (
    pliste: append(pliste,[p(x,y)]),
  )
);

```

Nun werden die Orte der Fläche zusammengefasst, die zu einem der 3x2 Neuronen gehören. Die Koordinaten dieser Orte werden in einer zum jeweiligen Neuron i gehörenden Liste *liste[i]* festgehalten.

```

for n: 1 step 1 thru e_anz do (
  liste[n]: [],
  for x: 0 step 0.01 thru 1 do
    for y: 0 step 0.01 thru 1 do
      if euklid(p(x,y),e[n]) < 0.01 then
        liste[n]: append(liste[n],[[x,y]])
);

```

Die Listen *pliste* und *liste*[1] bis *liste*[*e_{anz}*] lassen sich dann mit dem *Maxima*-Befehl *wxplot2d* wie oben gezeigt grafisch darstellen.

⁴¹Mit dem CAS *Maxima* lässt sich die Startmatrix *V* aus den Frage-Antwort-Paaren (*x_i*, *y_i*) wie folgt erzeugen:

```
v: transpose(x[1]).y[1] + transpose(x[2]).y[2] + transpose(x[3]).y[3];
```

⁴²Die zugehörigen *Maxima*-Befehle könnten sein:

```
f(x,theta):= if x > theta then 1 else 0;
eta: 1/2;
tmp[1]: matrix(makelist(f(y,0),y,list_matrix_entries(x[1].v)));
tmp[2]: matrix(makelist(f(y,0),y,list_matrix_entries(x[2].v)));
tmp[3]: matrix(makelist(f(y,0),y,list_matrix_entries(x[3].v)));
v: transpose(x[1]).(eta*(y[1]-tmp[1])) + v;
v: transpose(x[2]).(eta*(y[2]-tmp[2])) + v;
v: transpose(x[3]).(eta*(y[3]-tmp[3])) + v;
```

⁴³In *Maxima* geschieht das zum Beispiel für eine 3x5-Matrix *V₁* und eine 5x2-Matrix *V₂* durch:

```
h[i,j]:= random(1.0)-0.5;
v1: genmatrix(h,3,5);
v2: genmatrix(h,5,2);
```

⁴⁴In *Maxima* wären mögliche Befehle zur Berechnung von ΔV_1 und ΔV_2 :

```
delta_v2: eta * transpose(f(x.v1)).(f1(f(x.v1).v2)*(a_soll-a_ist));
delta_v1: eta * transpose(x).
(f1(x.v1)*(f1(f(x.v1).v2)*(a_soll-a_ist)).transpose(v2));
```

Die Matrizenmultiplikationen $\cdot$ werden in *Maxima* durch einen Punkt $'.'$ und die komponentenweisen Multiplikationen $\odot$ durch einen Stern $'*'$ angegeben.

⁴⁵back-propagating — rückwärts weitergebend

Maxima-Skript für ein
Backpropagation-Netz
mit grafischen Ausgaben

⁴⁶Das vollständige *Maxima*-Skript für dieses Beispiel lautet:

```
eta: 0.9; m_anz: 4;
x1: matrix([0.2,0.2]); a1: matrix([0]);
x2: matrix([0.2,0.8]); a2: matrix([1]);
x3: matrix([0.8,0.2]); a3: matrix([1]);
x4: matrix([0.8,0.8]); a4: matrix([0]);
m: [x1,x2,x3,x4];
a: [a1,a2,a3,a4];
h[i,j]:= random(2.0)-1.0;
v1: genmatrix(h,2,5); v2: genmatrix(h,5,1);
f(x):= 1/(1+%e^(-x)); f1(x):= f(x)*(1-f(x));
a_ist(k):= f(f(m[k].v1).v2);
err():= sum(sqrt((a[i]-a_ist(i)).(a[i]-a_ist(i))),i,1,m_anz);
durchlaufe: 0;
while err() > .05 and durchlaufe < 1000000 do (
  zuf: random(m_anz) + 1,
  delta_v2: eta * transpose(f(m[zuf].v1)).
    (f1(f(m[zuf].v1).v2)*(a[zuf]-a_ist(zuf))),
  delta_v1: eta * transpose(m[zuf]).
    (f1(m[zuf].v1)*(f1(f(m[zuf].v1).v2)*(a[zuf]-a_ist(zuf))).
      transpose(v2)),
  v2: v2 + delta_v2, v1: v1 + delta_v1,
  durchlaufe: durchlaufe + 1
)$
p(x,y):= f(f(matrix([x,y]).v1).v2);
wxplot3d(p,[x,0,1],[y,0,1],[z,0,1],
  [grid,31,60],[palette,false],[color,blue]),
  wxplot_size=[800,600];
plot3d(p,[x,0,1],[y,0,1],[z,0,1],
  [elevation,0],[azimuth,0],
  [mesh_lines_color,false],
  [colorbox,true],[grid,150,150],
  [legend,"Reaktion des Neuronalen Netzes"],
  [xlabel,""],[ylabel,""],[zlabel,""]];
```

⁴⁷Hier wurde das unter freier Lizenz vertriebene Programm *Gnuplot* von der grafischen Benutzeroberfläche *wxmaxima* aus eingesetzt.

⁴⁸Die Leitungen der Matrizen werden nach neurobiologischem Vorbild auch Axone genannt. Ein Axon ist ein Fortsatz der Nervenzelle, der ihr Ausgangssignal weiterleitet und verteilt.

⁴⁹Die Verbindungen zwischen den Zeilen- und Spaltenleitungen einer Matrix nennt man auch synaptische Verbindungen. Die Synapsen stellen die Schnittstellen zwischen Nervenzellen dar. Eine Nervenzelle im menschlichen Großhirn bildet etwa 10.000 synaptische Verbindungen zu anderen Nervenzellen aus (s. auch Anm. 1).

⁵⁰Palm gibt in [Palm o.J.] als zwei zur Auswahl stehende Lernregeln

$$w_{ij} = \sum_{\mu} x_i^{\mu} y_j^{\mu}$$

und

$$w_{ij} = \max_{\mu} x_i^{\mu} y_j^{\mu}$$

an, wobei mit w_{ij} die Einträge in einer Speichermatrix W bezeichnet sind.

⁵¹Der Kanadier Donald Olding Hebb (1904-1985) befasste sich mit neurobiologischen Prinzipien des Lernens in neuronalen Netzen. Hebb formulierte 1949 eine Regel, derzufolge sich eine synaptische Verbindung verstärkt, wenn die beteiligten Nervenleitungen gemeinsam aktiv sind („what fires together, wires together“). Bereits 1894 schrieb Sigmund Exner: „Nun müssen wir annehmen, dass eine Bahn durch häufige stärkere Erregung selbst stärker wird; das bedeutet in unserer Ausdrucksweise, dass durch häufige Erregung von a_1 m_1 die Zellen a_1 und m_1 in engere Verwandtschaft gesetzt werden“ (s. [Exner 1894], S. 152). Palm untersuchte neben der Hebb-Regel auch andere synaptische Verbindungsregeln (vgl. [Palm 1982], S. 56 und S. 180 ff.).

⁵²Der Speicher für die Werte von Variablen wird in der Assoziativmaschine Kurzzeitgedächtnis K genannt. Da Variablen während eines Programmlaufes im Allgemeinen ihre Werte ändern, wurde eine gesonderte Lernregel, die K-Lernregel, erforderlich, um die Einträge vergangener Werte aus der Assoziativmatrix entfernen zu können.

⁵³Die Assoziativmatrix A lässt sich für die drei Frage-Antwort-Paare (f_i, a_i) durch *Maxima* von folgender Befehlsfolge erzeugen:

```
f[1]: matrix([0,0,1,1,0,0,0,1]); a[1]: matrix([1,0,1,0,1,1,0,0,0,1,0,0]);
f[2]: matrix([0,0,0,1,1,0,1,0,1,0]); a[2]: matrix([0,1,1,0,0,0,0,1,1,1,0,1]);
f[3]: matrix([1,0,1,0,0,1,1,0]); a[3]: matrix([0,0,0,1,1,1,0,0,1,1,1,0]);
s(x):= if x >= 1 then 1 else 0;
tmp: transpose(f[1]).a[1] + transpose(f[2]).a[2] + transpose(f[3]).a[3];
matrixmap(s,tmp);
```

⁵⁴In *Maxima* bietet sich der *genmatrix*-Befehl an, um die gewünschte Matrix elementweise aufzubauen:

```
v1:transpose(f[1]).a[1]; v2:transpose(f[2]).a[2]; v3:transpose(f[3]).a[3];
h1[i,j]:= if (f[2][1])[i] = 1 then v2[i,j] else v1[i,j];
tmp: genmatrix(h1,8,12);
h2[i,j]:= if (f[3][1])[i] = 1 then v3[i,j] else tmp[i,j];
genmatrix(h2,8,12);
```

⁵⁵Die Frage f ist bei der Multiplikation als Zeilenvektor anzusetzen. Bei Nutzung eines Computeralgebrasystems wäre f gegebenenfalls zu transponieren. In *Maxima* lautete der Abfrage-Befehl somit *transpose(f).A*, wobei in f die Nullen und Einsen der Frage und in der Matrix A die Einträge der Assoziativmatrix abgelegt sind.

⁵⁶Die zugehörigen *Maxima*-Befehle für eine Assoziativmatrix v und eine Frage f sind:

```
s(x):= if x = Theta then 1 else 0;
tmp: f.v;
Theta: lmax(list_matrix_entries(tmp));
matrixmap(s,tmp);
```

Als Schwellwert Θ wird hier mit dem Befehl *lmax* der größte Wert ermittelt, der sich im Zwischenergebnis *tmp* befindet.

⁵⁷Der ASCII-Code (American Standard Code for Information Interchange) ist ein 7-Bit-Code, der seit 1967 als Standard zur Zeichendarstellung gilt (s. auch Kapitel 8.2).

Axone

synaptische
Verbindungen

Donald Olding Hebb

Assoziativmatrix:
L-Lernregel mit *Maxima*

Assoziativmatrix:
K-Lernregel mit
Maxima

Assoziativmatrix:
Abfrageregeln mit
Maxima

⁵⁸vgl. mit Ähnlichkeit in Kapitel 3

⁵⁹Die Befehle `lernedatei` und `zeigeeintrag` werden auf S. 246 erklärt.

⁶⁰In der angegebenen Buchstabenfolge das Wort GRASHALM zu erkennen, stellt sich als eine nicht allzu einfache Aufgabe heraus.

⁶¹In [Palm 1982], S. 165, führt der Autor dazu aus: „Roughly, the *amount of information* contained in a message should signify the number of yes/no questions required to guess this message.“ Dann werden dort verschiedene Beispiele diskutiert, auch mit Berechnungen und Überlegungen zum Speicherplatz und dessen Ausnutzung.

⁶²[Palm 1982], S. 165, schreibt dazu: „The actual optimal strategy is called the Huffman Code“.

Informationsgehalt

⁶³Hier ist die Informationstheorie nach Claude Shannon (1916-2001) gemeint, demzufolge sich der Informationsgehalt einer Antwort nach seinem „Überraschungswert“ bemisst.

⁶⁴Richard Hamming zählt in [Hamming 1987], S. 108, die drei Bedingungen auf, die für eine Funktion I gelten sollen, die den Informationsgehalt eines Ereignisses bemisst, das mit der Wahrscheinlichkeit p eintritt:

- i) Es sei $I(p) \geq 0$.
- ii) Für zwei unabhängige Ereignisse, die mit den Wahrscheinlichkeiten p_1 und p_2 eintreten, gelte $I(p_1 \cdot p_2) = I(p_1) + I(p_2)$.
- iii) I sei eine stetige Funktion von p .

Eine Logarithmusfunktion erfüllt die Bedingungen i) bis iii). $I(p) = -\log(p) = \log(\frac{1}{p})$.

⁶⁵Mehr zu den Grundlagen der Informationstheorie findet sich sowohl bei [Ash 1990] als auch bei [Shannon and Weaver 1949] und ganz aktuell bei [Palm 2012].

Informationsbit

⁶⁶Das Informationsbit ist nicht mit dem Bit (binary digit) zur binären Darstellung von Zahlen oder dem Bit als kleinste Speichereinheit eines Digitalspeichers zu verwechseln.

⁶⁷Eine Analyse der Speichereffizienz von Assoziativmatrizen ist in [Knoblauch et al. 2010] vorgenommen worden. Verschiedene Implementierungen assoziativer Speicher werden verglichen mit dem Ziel, die Unterschiede zwischen struktureller und synaptischer Plastizität besser zu verstehen.

falsche Einsen

⁶⁸Palm schreibt dazu in [Palm 1980], S. 24: „Of course all this can only be calculated exactly for a given mapping m . Instead we choose a mapping m at random (i.e., the pairs (x^t, y^t) to be associated are chosen independently) out of all possible lists with the parameters k, l, m, n, z fixed. Then for every pattern (x^t, y^t) the number N^t of wrong „ones“ becomes a random variable and we have to calculate the expectation of the corresponding information. The information stored is

$$I = \sum_{t=1}^z I_t = \sum_{t=1}^z \left[ld \binom{n}{k} - ld \binom{k+N^t}{k} \right]$$

($\binom{n}{k}$) is the number of possible ways to choose k „ones“ from n , ($\binom{k+N^t}{k}$) is the number of possible ways to choose k „ones“ from $k+N^t$.“ Die weitere Umformung liefert:

$$= \sum_{t=1}^z ld \left(\prod_{i=0}^{k-1} \frac{n-i}{k+N^t-i} \right) = \sum_{t=1}^z \sum_{i=0}^{k-1} ld \frac{n-i}{k+N^t-i}$$

Unter den hier von Palm genannten Parametern gibt m die Anzahl der Zeilen- und n die Anzahl der Spaltenleitungen der Assoziativmatrix an. In die Matrix werden z Frage-Antwort-Paare eingetragen. In den Fragen sind stets genau l Einsen gesetzt und in den Antworten genau k Einsen. Beim Auslesen tauchen N^t falsche Einsen auf.

Autoassoziation

Ebenfalls in [Palm 1980], S. 24, leitet der Autor als Informationsgehalt I einer **autoassoziativ** eingesetzten Assoziativmatrix (Fragen werden mit sich selbst beantwortet) im Unterschied zur obigen, nicht-autoassoziativen Nutzung her:

$$I = \sum_{t=1}^z I_t = \sum_{t=1}^z \left[ld \binom{n}{k} - \sum_{l=0}^{k-1} ld \frac{N_l^t + k - l}{k - l} \right]$$

$$\begin{aligned}
&= \sum_{t=1}^z \left[\sum_{l=0}^{k-1} ld \frac{n-l}{k-l} - \sum_{l=0}^{k-1} ld \frac{N_l^t + k - l}{k-l} \right] \\
&= - \sum_{t=1}^z \sum_{l=0}^{k-1} ld \frac{N_l^t + k - l}{n-l}
\end{aligned}$$

mit n als Anzahl der Matrixspalten, k als Anzahl der gesetzten Einsen in den Fragen, z als Anzahl der Frage-Antwort-Paare und N_l^t als Anzahl der „falschen Einsen“.

In [Palm 1993], S. 144, heißt es des Weiteren: „Even with binary synapses a limiting value of [storage efficiency] $E = \ln 2 = 69\%$ can be obtained. In this case the optimal asymptotic formula for the sparseness is $p = \frac{\ln(n)}{n}$. Although this limit can only be reached for very large n , one can achieve efficiencies $E > 50\%$ already for $n = 1.000$. For $p = \frac{1}{1.000}$ this yields an information content of about 70 *bit* per pattern and thus $M = E \cdot \frac{10^6}{70} = 7.000$. For auto-association the theoretical limit with binary synapses is $\frac{\ln(2)}{4} = 17,33\%$.“ Mit M ist hier die Anzahl der Muster bezeichnet, die abgespeichert werden können, und mit n die Anzahl der Matrixspalten. Der Begriff „storage efficiency“ E wird in Kapitel 2.5 als **Ausnutzungsgrad** eingeführt.

Ausnutzungsgrad

⁶⁹Zur **Textrecherche** in großen Datenmengen mit Hilfe von Assoziativmatrizen s. auch [Hagström 1996].

Textrecherche

⁷⁰Das Beispiel mit der 7x7-Matrix wurde den Materialien von Fabian Rosenschein, Universität Hildesheim, entnommen. Rosenschein beschäftigt sich in seiner Dissertation „Assoziative Systeme als Datenspeicher“ (s. [Rosenschein 2012]) mit der Frage, ob sich komplette Texte in den Assoziativmatrizen speichern lassen. Hierbei stellt sich das Problem der Ordnung und Abfolge. Textbausteine, also meist Sätze, liegen in einer Reihenfolge vor, die beachtet werden muss, wenn das Auslesen korrekt erfolgen soll. Gleichlautende Textbausteine dürfen nicht zu Assoziationskreisen führen. In diesem Kontext ist das Erzeugen und Speichern von Sequenzen ausführlich zu untersuchen.

⁷¹Über den Einsatz von Assoziativmatrizen für Grundaufgaben der Informatik schreibt [Heitland 1994].

⁷²Die Abfrage für das abgebildete Beispielquader könnte in *Maxima* wie folgt vorgenommen werden. Zunächst berechnet man die Schwellwerte für eine senkrechte Abfrage von Q mit

Abfrage des assoziativen Quaders mit *Maxima*

```

Q: matrix([
  matrix([0,1,1,1,1],[0,0,0,0,0],[0,0,0,0,0],[1,1,0,0,0]),
  matrix([0,0,0,0,0],[1,0,1,0,0],[0,0,0,0,0],[1,0,1,0,0]),
  matrix([0,1,1,1,1],[0,0,0,0,0],[1,0,0,1,0],[0,0,0,0,0])
]);
F: matrix([1,0,0,1],[0,1,0,1],[1,0,1,0]);
F.Q;

```

und dann für eine waagerechte Abfrage von Q mit derselben, aber für die neue Sicht auf Q passend gedrehten Fragefläche F :

```

Q: matrix([
  matrix([0,1,1,1,1],[0,0,0,0,0],[0,1,1,1,1]),
  matrix([0,0,0,0,0],[1,0,1,0,0],[0,0,0,0,0]),
  matrix([0,0,0,0,0],[0,0,0,0,0],[1,0,0,1,0]),
  matrix([1,1,0,0,0],[1,0,1,0,0],[0,0,0,0,0])
]);
F: matrix([1,0,1],[0,1,0],[0,0,1],[1,1,0]);
F.Q;

```

⁷³Wenn es auch an Anschauung zu diesen höherdimensionalen assoziativen Quadern mangelt, so hilft das Computeralgebrasystem *Maxima* bei der Untersuchung dieser Gebilde, indem man den Aufbau dieser assoziativen Quader durch Matrizen von Matrizen fortschreibt. Beispielsweise baut man einen 3x2x4x5-Quader Q_4 durch folgende Anweisungen auf und fragt ihn mit einem 3x2x4-Fragequader F wie nachstehend ab:

```

Q4: matrix([
  matrix([
    matrix([1,0,1,0,1],[1,1,1,0,0],[1,0,1,0,0],[0,1,1,0,0]),
    matrix([1,0,0,0,1],[0,1,1,1,0],[1,0,0,1,0],[0,0,1,0,0])
  ]),

```

```

matrix([
  matrix([0,0,1,0,0],[0,0,0,1,1],[1,0,1,0,0],[0,0,0,0,0]),
  matrix([1,1,0,1,0],[1,1,0,1,1],[0,1,0,1,0],[0,0,1,1,1])
]),
matrix([
  matrix([0,0,0,0,0],[1,0,0,0,1],[1,0,1,0,0],[1,0,0,0,0]),
  matrix([0,1,0,1,0],[0,0,0,1,0],[0,0,0,1,1],[0,1,1,0,0])
])
]);
F: matrix([
  matrix([[1,0,1,0],[1,1,0,1]]),
  matrix([[0,1,1,0],[1,0,1,1]]),
  matrix([[1,0,1,0],[0,1,1,0]])
]);
F.Q4;

```

Der Fragequader F lässt sich nun in Analogie zum dreidimensionalen Fall ebenfalls drehen und als $2 \times 4 \times 3$ - oder $4 \times 3 \times 2$ -Quader zur Abfrage an $Q4$ nutzen, wodurch wir drei verschiedene dreidimensionale Antwortquader bei der Abfrage dieser vierdimensionalen assoziativen Struktur erhalten. Die Tabelle 2.6 gibt eine Übersicht zu den in diesem Buch vorgestellten assoziativen Strukturen, die mit der L-Lernregel gefüllt werden.

assoziative Strukturen

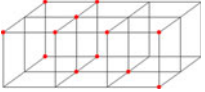
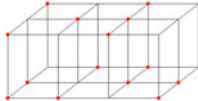
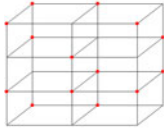
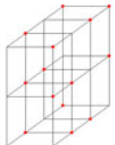
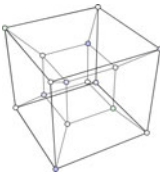
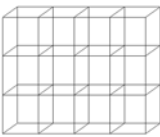
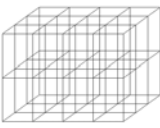
Dimension	Frage	assoziative Struktur	Antwort
2	$\begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 2 & 0 & 1 \end{pmatrix}$
3	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$ $\begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$		$\begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ $\begin{pmatrix} 2 & 0 & 1 \\ 1 & 1 & 2 \end{pmatrix}$
4	  		  

Tabelle 2.6: Fragen und Antworten assoziativer Strukturen

⁷⁴Bei diesem Ansatz zur Beschreibung von Bewusstsein versteht man die „waagerechten“ Einträge als Wahrnehmungs- oder Kontrollinstanz für die „senkrechten“ Einträge (oder umgekehrt). Bewusstsein (lat. *conscientia* — Mitwissen, Gewissen, Bewusstsein) stellt in diesem Sinne eine Mitwahrnehmung, Mitbild, Mitempfindung dar. So lässt sich über assoziative Quader und Prismen nachbilden, dass ein Eindruck nicht nur zu einem einzelnen, mit diesem Eindruck assoziierten Verhalten führt, zum Beispiel dem Fallenlassen eines zu heißen Gegenstands, sondern dass dieser Eindruck auch Mitwahrnehmungen, Mitwirkungen auslöst, zum Beispiel die Zuordnung eines Schmerzindrucks oder einer Benennung. Eine Einführung zum Thema Bewusstsein findet man in [Carter et al. 2010], S. 174-189. Ergebnisse zum Zusammenhang zwischen Bewusstsein und dem synchronen Feuern weit verteilter Neuronenverbände fasst [Wolf 2010] zusammen.

Bewusstsein

Literaturverzeichnis zu Kapitel 2

Hervé Abdi. *Les réseaux de neurones*. Presses universitaires de Grenoble, 1994. ISBN 2-7061-0554-2.

Robert B. Ash. *Information Theory*. Dover Publications, New York, new edition, 1990. ISBN 978-0-486-66521-4.

Rita Carter, Susan Aldridge, Martyn Page, and Steve Parker. *Das Gehirn — Anatomie, Sinneswahrnehmung, Gedächtnis, Bewusstsein, Störungen*. Doring Kindersley Verlag, München, 2010. ISBN 978-3-8310-1730-0.

Stanislas Dehaene. *Der Zahlensinn oder warum wir rechnen können*. Birkhäuser Verlag, Basel Boston Berlin, 1999. ISBN 3-7643-5960-9.

Andreas Dierks. *VidAs — Aufbau einer robusten, frei programmierbaren Maschine aus Assoziativmatrizen. Simulation und Hardware-Lösung*. Universität Hildesheim, 2005. in: „Hildesheimer Informatik-Berichte“, Band 2/2005 (September 2005), ISSN 0941-3014.

Sigmund Exner. *Entwurf zu einer physiologischen Erklärung der psychischen Erscheinungen*. Franz Deuticke, Leipzig und Wien, 1894.

A.I. Galushkin, S.V. Korobkova, and P.A. Kazantsev. *Neuromathematics: Development tendencies*. Baku State University, 2003. in: „Appl. Comput. Math“ Vol. 2 (2003), No. 1, pp. 57-64.

Michael Hagström. *Textrecherche in großen Datenmengen auf der Basis spärlich codierter Assoziativmatrizen*. Universität Hildesheim, 1996. Dissertationsschrift.

Richard Wesley Hamming. *Information und Codierung*. VCH Verlag, Weinheim New York, 1987. ISBN 3-527-26611-9.

Donald Olding Hebb. *The organization of behavior. A neuropsychological theory*. 1949.

Robert Hecht-Nielsen. *Neurocomputing*. Addison-Wesley Publishing, Reading, 1991. ISBN 0-201-09355-3.

Michael Heitland. *Einsatz der SpaCAM-Technik für ausgewählte Grundaufgaben der Informatik*. Universität Hildesheim, 1994. Dissertationsschrift.

John Joseph Hopfield. *Neural networks and physical systems with emergent collective computational abilities*. PNAS, 1982. in: „Proc. Natl. Acad. Sci. USA“, Vol. 79, S. 2554-2558, April 1982.

John Joseph Hopfield. *Neurons with graded response have collective computational properties like those of two-state neurons*. PNAS, 1984. in: „Proc. Natl. Acad. Sci. USA“, Vol. 81, S. 3088-3092, Mai 1984.

Andreas Knoblauch, Günther Palm, and Friedrich T. Sommer. *Memory Capacities for Synaptic and Structural Plasticity*. MIT Press Cambridge, 2010. in „Neural Computation“ Vol. 22 No. 2, S. 289-341.

Teuvo Kohonen. *Content-Adressable Memories*. Springer-Verlag, Berlin Heidelberg New York, 1987. Second Edition, o.J., ISBN 3-540-17625-X.

Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin Heidelberg New York, 1988. Second Edition, o.J., ISBN 3-540-18314-0.

Teuvo Kohonen. *Self-Organizing Maps*. Springer Verlag, Berlin Heidelberg New York, 2001. Third Edition, ISBN 978-3-540-67921-9.

Warren S. McCulloch and Walter Pitts. *A logical calculus of the ideas immanent in nervous activity*. SMB, 1943. in: „Bulletin of Mathematical Biophysics“, Vol. 5, 1943, S. 115-133.

Marvin L. Minsky and Seymour A. Papert. *Perceptrons — An introduction to computational geometry*. The MIT Press, Cambridge (Massachusetts) London, 1988. Expanded edition, ISBN 0-262-63111-3.

R. Moreno-Diaz and K. N. Leibovic. *On some Methods in Neuromathematics (or the Development of mathematical Methods for the Description of Structure and Function in Neurons)*. Springer Verlag, 1995. in: „From Natural to Artificial Neural Computation“, S. 209-214.

John von Neumann. *Die Rechenmaschine und das Gehirn*. Oldenbourg Verlag, München, 1970. 3. Auflage.

Günther Palm. *On Associative Memory*. Springer-Verlag, 1980. in: „Biological Cybernetics“ Nr. 36, S. 19-31.

Günther Palm. *Neural Assemblies — An Alternative Approach to Artificial Intelligence*. Springer-Verlag, Berlin Heidelberg New York, 1982. ISBN 3-540-11366-5.

Günther Palm. *Assoziatives Gedächtnis und Gehirntheorie*. Spektrum der Wissenschaft Verlag, Heidelberg, 1988. in: „Spektrum der Wissenschaft“ Nr. 6/1988, S. 54-64.

Günther Palm. *The PAN System and the WINA Project*. Springer-Verlag, Berlin Heidelberg, 1993. in: „Euro-ARCH '93. Europäischer Informatik Kongreß Architektur von Rechensystemen“ (Spies, P.P, Hrsg.), S. 142-156.

Günther Palm. *Novelty, Information and Surprise*. Springer Verlag, Berlin Heidelberg, 2012. ISBN 978-3-642-29075-6.

Günther Palm. *PAN IV — Ein Massiv Paralleler Neuronaler Assoziativspeicher*. Universität Ulm, o.J.

Ulrich Rückert. *Integrationsgerechte Umsetzung von assoziativen Netzwerken mit verteilter Speicherung*. VDI-Verlag, Düsseldorf, 1990. in: „Reihe 10: Informatik/KommunikationstechnikNr. 130, ISBN 3-18-143010-2.

Ulrich Rückert. *VLSI Design of an Associative Memory based on Distributed Storage of Information*. Kluwer Academic Publishers, Boston, 1991. in: Ramacher, Ulrich; Rückert, Ulrich (Hrsg.) : „VLSI Design of Neural Networks“, S. 153-168.

Ulrich Rückert. *Microelectronic Implementation of Neural Networks*. RWTH Aachen, 1993. in: „Aachenener Beiträge zur Informatik“, Band 3, S. 77-86.

Ulrich Rückert. *Hardwareimplementierung Neuronaler Netze*. GMD, 1994. in: „Konnektionismus und Neuronale Netze“, GMD-Studie Nr. 242 , S. 117-128.

Helge Ritter, Thomas Martinetz, and Klaus Schulten. *Neuronale Netze. Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke*. Addison-Wesley, Bonn, Paris, Reading (u.a.), 1991. 2. unveränderter Nachdruck 1994, ISBN 3-89319-131-3.

Frank Rosenblatt. *The Perceptron: A probabilistic model for information storage and organization in the brain*. APA, 1958. in: „Psychological Review“, Vol. 65, No. 6, S. 386-408.

Fabian Rosenschein. *Assoziative Systeme als Datenspeicher*. Universität Hildesheim, 2012. Dissertationsschrift.

Alwyn Scott. *Neuroscience — A mathematical primer*. Springer Verlag, New York Berlin Heidelberg, 2002. ISBN 0-387-95402-3.

Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, 1949. Taschenbuchausgabe 1998, ISBN 0-252-72548-4.

Gordon L. Shaw and Günther Palm, editors. *Brain theory — Reprint volume*. World Scientific Publishing, Singapore New Jersey Hong Kong, 1988. ISBN 9971-50-483-9.

Emanuel Sperner. *Einführung in die Analytische Geometrie und Algebra*. Vandenhoeck & Ruprecht, Göttingen, 1951. 5. Auflage 1963.

D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins. *Non-Holographic Associative Memory*. Macmillan Magazines, 1969. in: „Nature“ Vol. 222, June 7, 1969, pp. 960.

Christian Wolf. *Dem Bewusstsein auf der Spur*. Spektrum der Wissenschaft Verlag, Heidelberg, 2010. in: „Gehirn & Geist Basiswissen Nr. 2/2010 — Das Gehirn – Aufbau und Funktion“, S. 76-78.

Neuromathematik und Assoziativmaschinen

Bentz, H.-J.; Dierks, A.

2013, X, 383 S. 300 Abb., 77 Abb. in Farbe., Softcover

ISBN: 978-3-642-37937-6