Institut für Netzwerk- und Systemtheorie
Universität Stuttgart
Prof. Dr-Ing. habil. E. Lüder

Analyse der Eigenschaften von assoziativen Speichern für die Mustererkennung

Diplomarbeit

von

Martin Bässler

Betreuer: Dipl.-Ing. Volker Schmid

Ausgabe der Arbeit: 1. September 1997
Abgabe der Ausarbeitung: 27. Februar 1998

Zusammenfassung

Es werden HOPFIELD- Netzwerke mit bis zu 32 Neuronen und asynchroner Abfrage untersucht.
In einem ersten Teil (Kap. 2) wird die benötigte Theorie aus der Literatur zusammengestellt. Gleichzeitig wird eine neue Erklärung für residuelle Fehler unter Verwendung der Energiefunktion vorgeschlagen.
Als nächstes werden in Kap. 3 acht verschiedene Netzwerke mit bis zu 10 Neuronen vollständig untersucht. Das heißt, es werden auf alle möglichen Anfangszustände alle Permutationen der sequentiellen Reihenfolge angewandt und festgestellt, welchen Zuständen das Netz zustrebt. Dabei ergab sich, daß es drei verschiedene Zustände gibt: (1) Zustände, die einen stabilen Zustand repräsentieren (eingeprägte Muster bzw. durch einen residuellen Fehler leicht verschobene Muster), (2) Zustände, die unabhängig von der Abfragereihenfolge immer zum gleichen stabilen Zustand streben (Zustände in Attraktionsbecken) und (3) Zustände, die je nach Abfragereihenfolge unterschiedliche stabile Zustände erreichen (nicht- zuordenbare Zustände). Der prozentuale Anteil nicht- zuordenbarer Zustände steigt mit wachsender Zahl der stabilen Zustände. Zustände in Attraktionsbecken können über unterschiedliche Trajektorien den stabilen Zustand erreichen. Trotz symmetrischer Gewichtsmatrix können Unsymmetrien in den Netzzuständen auftreten.
In Kap. 4 werden 63 verschiedene Netzwerke aus 32 Neuronen untersucht. Dabei wurden zufällige Abfrageblöcke benutzt. Zunächst wurden möglichst alle stabilen Zustände (Muster und spurious states) ermittelt und dann deren Attraktionsbecken bis zu einem Hamming Abstand von 3 vollständig erfaßt. Dabei wurden die Ergebnisse von Kap. 3 im wesentlichen bestätigt. Darüber hinaus ergab sich, daß die Attraktionsbecken der Muster größer waren als die der spurious states. Die Zahl der spurious states war bei mehr als zwei eingespeicherten Mustern (darunter traten keine spurious states auf) nicht mit der Anzahl der eingespeicherten Muster korreliert.
In Kap. 5 wurden in zwei der vorher untersuchten Netze die gleichen Muster, aber jetzt mit der Projektionsregel eingespeichert. Die Trennung der Muster verbesserte sich, und die Größenunterschiede der Attraktionsbecken von fundamentalen Mustern und spurious states nahmen zu.
In Kap. 6 werden die erhaltenen Ergebnisse im Hinblick auf die Verwendung assoziativer Speicher für die Mustererkennung diskutiert.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Liste der verwendeten Formelzeichen

Formelzeichen	Bedeutung
$\Delta E(t)$	Energiedifferenz der Netzzustände zum Zeitpunkt t
$\Delta x_i(t)$	Differenz der Zustände des Neuorons i zum Zeitpunkt t
α	Verhältnis von Zahl fundamentaler Muster zur Gesamtzahl der Neuronen
ϵ	Residueller Fehler
μ	Laufparameter zur Kennzeichnung fundamentaler Muster
ν	Laufparameter zur Kennzeichnung fundamentaler Muster
ρ	Attraktionsradius
ϑ	Schwellenwert
ξ	fundamentales Muster
$\xi_{\mathbf{dec}}$	Dezimaldarstellung des fundamentalen Musters
ξ_i	$i$tes Element des fundamentalen Musters
$\mathbf{A}$	Abfragereihenfolge
$\mathbf{C}$	Korrelationsmatrix
$E(\mathbf{x})$	Energiefunktion in Abhängigkeit vom Netzzustand
E_r	Rauschenergie des angelegten fundamentalen Musters
E_s	Signalenergie des angelegten fundamentalen Musters
N	Gesamtanzahl der Neuronen
SNR (dB)	Signal-zu-Rausch-Verhältnis (in dB gemessen)
b	Kennzeichnung von fundamentalen Mustern
d	Hamming Abstand
$f\,()$	Nichtlineare Aktivierungsfunktion
$\mathbf{f}$	Stabiler Zustand
i	Laufparameter zur Kennzeichnung von Neuronen

Formelzeichen	Bedeutung
j	Laufparameter zur Kennzeichnung von Neuronen
k	Laufparameter zur Kennzeichnung von Neuronen
$\log()$	Natürlicher Lograrithmus
$\log_{10}()$	Logarithmus zur Basis 10
$p()$	Wahrscheinlichkeitsdichte
p	Gesamtanzahl der eingespeicherten fundamentalen Muster
$\mathrm{sgn}[\,]$	Vorzeichenfunktion
t	Zeitpunkt; Laufparameter zur Kennzeichnung fundamentaler Muster
u_i	Eingangswert des Neurons i
$\mathbf{w}$	Gewichtsmatrix
w_{ij}	Gewichtsfaktor, wobei das Neuron i das Neuron j beeinflußt
x	Zustand des Neurons; Skalarer Wert
$\mathbf{x}(t)$	Netzzustand (Zustandsvektor) zum Zeitpunkt t
$\mathbf{x_{dec}}$	Dezimaldarstellung des Netzzustands
$x_i(t)$	Zustand des Neurons i zum Zeitpunkt t
$x_{i,binär}$	Binäre Darstellung des Zustands des Neurons i
$y_i(t)$	Inneres Argument des Neurons i zum Zeitpunkt t
y_r	Rauschterm des inneren Arguments des angelegten fundamentalen Musters
y_s	Signalterm des inneren Arguments des angelegten fundamentalen Musters

Kapitel 1

Einleitung

1.1 Fragestellungen

Ein mustererkennendes System muß zweierlei leisten: Es muß eine gewisse Anzahl „bedeutsamer“ Muster eindeutig unterscheiden können, auch wenn sie nicht ganz vollständig oder verrauscht sind, und es muß diese bedeutsamen Muster von zufälligen Anordnungen trennen können.
Musterkennung wird in technischen Systemen heute im allgemeinen über Feature- Detektion durchgeführt. Die Mustererkennung selbst einfacher biologischer Systeme läßt sich aber nicht allein durch Feature- Detektion erklären (Cruse, 1972). Es ist also anzunehmen, daß es neben der Feature-Detektion andere Typen von Verfahren gibt, die Mustererkennung leisten können.
Eine denkbare Möglichkeit wäre die Verwendung assoziativer Speicher. Sie wären in der Lage, mehrere Muster gleichzeitig zu speichern, sowie eventuell unvollständige oder verrauschte Muster richtig zuzuordnen. Assoziative Speicher haben aber auch Nachteile für die Mustererkennung. So können assoziative Speicher bei synchroner Abfrage oszillieren. Bei asynchroner Abfrage strebt das System von jedem Anfangszustand immer einem stabilen Zustand zu. Welcher stabile Zustand dies ist, kann von der Abfragereihenfolge abhängen. Es kann also vorkommen, daß eine zufällige Anordnung einem bestimmten Muster zugeordnet wird und diese Zuordnung von der zufällig gewählten Abfragereihenfolge abhängt. Bei der Verwendung assoziativer Speicher für die Mustererkennung muß also auch das Problem der Unterscheidung von unvollständigen Mustern und „Nicht- Mustern“ gelöst werden, neben einer Optimierung der Speicher selbst in Form von Erhöhung der

Speicherkapazität, der Verringerung der „spurious states“ (stabile Zustände, die nicht einem Muster entsprechen, s. dazu aber Abschnitt 6.2) und Vergrößerung der Attraktionsbecken der eingespeicherten Muster.
In der vorliegenden Arbeit werden assoziative Speicher nach Hopfield mit bis zu 32 Neuronen und asynchroner Abfragereihenfolge untersucht. Dabei sollen Kriterien gefunden werden, wie sich unvollständige oder verrauschte Muster von „Nicht- Mustern“ (sogenannte nicht- zuordenbare Zustände) trennen lassen. Es sollen ferner die Größe und Struktur der Attraktionsbecken von Mustern und spurious states vollständig und nicht nur statistisch erfaßt werden. Der Einfluß verschiedener Methoden der Musterspeicherung auf die Möglichkeiten der Trennung von Muster und Nicht- Muster und auf die Größe der Attraktionsbecken soll erfaßt werden. Der Einfluß von der Anzahl der Muster auf die Trennung von Mustern und Nicht- Mustern und die Größe und Struktur der Attraktionsbecken ist zu klären.

1.2 Neuronale Netze und ihre natürlichen Vorbilder

Neuronale Netzwerke finden immer weitere Verbreitung in der Technik. Sie bestehen aus einer meist großen Zahl von einzelnen Neuronen, die in vielfältiger Weise miteinander verbunden sind. Das Vorbild dieser neuronalen Netzwerke ist das Nervensystem der Tiere oder des Menschen. Es besteht aus Nervenzellen, die hochgradig miteinander vernetzt sind. So kann eine einzige Nervenzelle Informationen von bis zu 200 000 anderen Nervenzellen (Purkinjezelle des Kleinhirns) erhalten.
Der Erregungszustand einer normalen Nervenzelle drückt sich in der Frequenz der von ihr erzeugten und weitergeleiteten Aktionspotentiale aus, ist also analog codiert. Ein Aktionspotential ist eine Art Einheitsimpuls von ca. 1 ms Dauer. Andere Nervenzellen werden nur während eines Aktionspotentials beeinflußt, d. h. der Erregungszustand wird von anderen Nervenzellen in einem variablen Zeittakt abgetastet (Reichert, 1990). Die vielen Eingänge, die eine bestimmte Nervenzelle hat, werden in seiner Eingangsregion (Dendritenbaum) zeitlich und räumlich in komplexer Weise überlagert und erzeugen so den Erregungszustand. Die Eingangsregion bildet einen komplizierten Tiefpaß- Filter (Cruse, 1996). Aufgrund der hier nur angedeuteten komplexen Eigenschaften ist es sehr aufwendig, das Verhalten auch nur einer ein-

zigen Nervenzelle zu simulieren (z. B. Nemosys (Eeckman, 1992)). Deshalb nehmen schon Simulationsprogramme, die das Verhalten kleinerer Netzwerke aus Nervenzellen darstellen sollen, deutliche Vereinfachungen vor, z. B. bei der Simulation der Eingangsregion im Programm Biosim (Bergdoll und Koch, 1995).
Neben den bisher erwähnten „normalen“ Nervenzellen gibt es andere, die sich etwas leichter simulieren ließen. Dazu gehören die Nervenzellen ohne Aktionspotentiale wie z. B. viele Nervenzellen der Netzhaut des Auges. Ihre Erregung drückt sich nur in der Höhe ihres Zellpotentials aus, und sie beeinflussen die Folgezellen kontinuierlich (Reichert, 1990). Bei Nervenzellen mit Plateaupotentialen (Reichert, 1990) steigt das Erregungsniveau nach einem ersten, durch irgendwelche Eingänge ausgelösten Aktionspotential durch intrinsische Vorgänge rasch auf sein Maximum an (maximale Aktionspotential-Frequenz). Die Erregung hört erst auf, wenn starke hemmende Eingänge das Zellpotential unter die Schwelle für die Auslösung von Aktionspotentialen drücken (bistabiles Verhalten).
Das komplexe Verhalten einzelner Nervenzellen und die inhomogene Netzwerktopologie macht es unpraktisch, natürliche neuronale Netze als direkte Vorbilder für technische neuronale Netzwerke zu verwenden, selbst wenn die natürlichen Vorbilder gut bekannt sind wie das stomatogastrische Ganglion der Krebse (Selverston und Moulins, 1987).
Die in technischen Netzen verwendeten Neuronen verzichten auf eine spezifische Eingangsregion mit komplexen Eigenschaften. Vielmehr werden alle Eingänge linear aufsummiert (Cruse, 1996). Zur Bestimmung des Ausgangs wird auf die erhaltene Summe eine nichtlineare Funktion angewendet (s. Abb. 1.1).

$$x = f\left(\sum_i u_i + \vartheta\right). \tag{1.1}$$

In der vorliegenden Arbeit wurde nur der Schwellenwert $\vartheta = 0$ benützt. Als Nichtlinearität wurde nur die Signum- Funktion verwendet.
Die in technischen neuronalen Netzwerken verwendeten Topologien sind sehr unterschiedlich. Sie sind aber deutlich regelmäßiger und homogener als die natürlicher neuronaler Netze. In der vorliegenden Arbeit wird nur der Typ des assoziativen Speichers nach Hopfield (1982) verwendet.

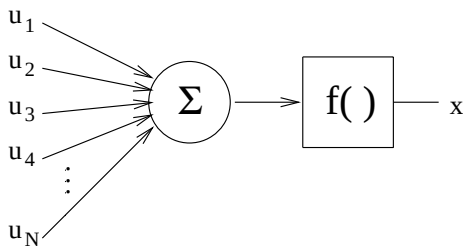


Abbildung 1.1: *Modell eines Neurons*

1.3 Assoziative Speicher

1.3.1 Grundlegender Aufbau

Man kann verschiedene Arten assoziativer Speicher unterscheiden. So kann man Feedforward- Netze zwar im erweiterten Sinne auch als assoziative Speicher bezeichnen, klassischerweise werden aber assoziative Speicher nach Hopfield benutzt. Diese sind einschichtig rückgekoppelte Netze (Abb. 1.2) und bestehen aus einer mehr oder weniger großen Zahl von Neuronen. Jedes Neuron beeinflußt jedes andere Neuron in einer vorher festgelegten Stärke (Kopplungsstärke). Die Einflüsse auf jedes der Neuronen können alle gleichzeitig (synchron) abgefragt werden. Es kann aber auch ein Neuron nach dem anderen (asynchron) abgefragt werden. Die vorliegende Arbeit verwendet nur die asynchrone Abfrage. Für die synchrone Abfrage s. Maier (1995).
Bei der Abfrage eines Neurons ergibt sich dessen Zustand aus der Summe seiner Eingänge und einer überlagerten Nichtlinearität. Unter diesem Einfluß kann sich der Zustand dieses Neurons ändern. Entsprechendes gilt für alle Neuronen. Wenn, wie in den hier untersuchten Netzen, die Kopplungsstärken symmetrisch sind, strebt der Gesamtzustand aller Neuronen zu bestimmten stabilen Zuständen. Um welche stabile Zustände es sich handelt, wird von den Stärken der gegenseitigen Beeinflussungen bestimmt. Je nach Größe des Netzes kann die Zahl der stabilen Zustände unterschiedlich sein. Das Netz strebt also von jedem beliebigen Anfangszustand zu einem dieser stabilen Zustände, weil jeder dieser stabilen Zustände bei einer festgelegten Abfragereihenfolge ein gewisses Attraktionsbecken besitzt. Das Netz ordnet den Anfangszustand damit einem bestimmten stabilen Zustand zu, mit anderen Worten, es assoziiert den Anfangszustand mit einem bestimmten stabilen Zustand.

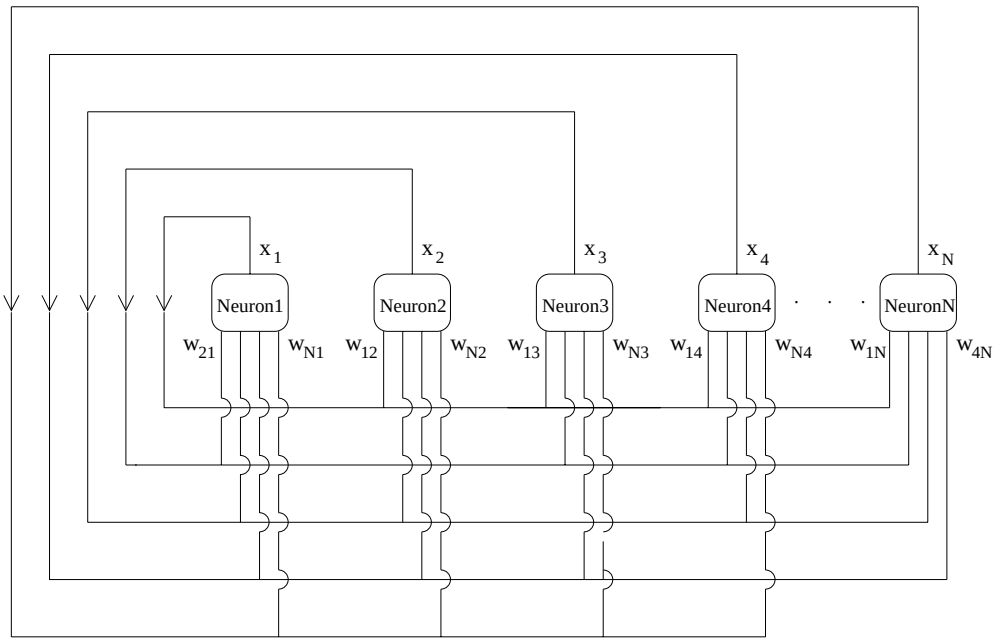


Abbildung 1.2: *Einschichtig rückgekoppeltes Netzwerk nach* HOPFIELD

Aufgrund der beschriebenen Eigenschaft ist ein HOPFIELD- Netz in der Lage, unvollständige oder verrauschte Muster zu erkennen. Dieses Netz kann also als fehlertolerant bezeichnet werden. Man kann auch sagen, das Netz kann einen beliebigen Anfangszustand einem bestimmten Muster zuordnen, es kann klassifizieren.
Die Tatsache, daß das Netz immer einen stabilen Zustand erreicht, kann aber auch ein Nachteil sein. Welcher stabile Zustand erreicht wird, kann nämlich bei asynchroner Abfrage teilweise von der Reihenfolge der abgefragten Neuronen abhängen. Wenn diese Reihenfolge zufällig gewählt wird, kann es also vorkommen, daß bestimmte Anfangszustände zufällig mal dem einen, mal dem anderen Muster zugeordnet werden.

1.3.2 Vergleich mit natürlichen Vorbildern

Mustererkennung kann in technischen wie in biologischen Systemen auf unterschiedliche Weise erfolgen.
Bei den meisten technischen Systemen wird im weitesten Sinne eine Feature-Detektion durchgeführt. Das heißt, an einem vorgelegten Muster wird eine festgelegte Zahl von Parametern ermittelt. Eine bestimmte Wertekombination dieser Parameter (ein bestimmter Bereich des Parameterraumes) wird als Feature eines zu erkennenden Musters definiert.
Das visuelle System der Säugetiere (Reichert, 1990) führt auf seinen ersten Verarbeitungsstufen eine solche Feature- Detektion durch, nicht aber auf den späteren Stufen.
Cruse (1972) konnte dagegen für die Mustererkennung von Honigbienen zeigen, daß keine Feature- Detektion im obigen Sinne vorliegt. Er dressierte die Tiere z. B. auf einen dreistrahligen Stern. Ein unbelohnter vierstrahliger Stern wurde im Versuch in 24% der Anflüge mit dem dreistrahligen Stern verwechselt. Wurde dagegen der vierstrahlige Stern belohnt, flogen die Bienen den dreistrahligen Stern in 37% der Fälle an. Die Unterschiede sind signifikant. Entsprechende Unsymmetrien wurden bei weiteren 10 Figurenpaaren gefunden. Da in einem beliebigen Parameterraum der Abstand von A nach B gleich dem Abstand von B nach A ist, schloß Cruse (1972), daß keine Feature- Detektion vorliegt.
Diese Ergebnisse ließen sich aber durch die Annahme eines assoziativen Speichers erklären. Sofern die Attraktionsbecken zweier Muster unterschiedlich groß sind, müßten genau solche unsymmetrischen Verwechslungen auftreten.

1.4 Spurious States

In vielen Hopfield- Netzwerken treten neben den stabilen Zuständen, die ein eingeprägtes Muster repräsentieren, weitere stabile Zustände auf. Solche unbeabsichtigten (unechten) Zustände bezeichnet man als spurious states. Sie sind „unecht“ (spurious), weil sie nicht den eingeprägten Mustern entsprechen, aber durch die Festsetzung der Kopplungsstärken zum Zwecke der Einprägung der Muster entstehen.

1.5 Vorteile kleiner Netzwerke für die Untersuchungen

Es war das Ziel der vorliegenden Arbeit, das Verhalten assoziativer Speicher zu verstehen, d. h. nachvollziehbar zu machen. Dazu schien es mir wichtig zu sein, das Verhalten möglichst jedes Neurons konkret zu verfolgen und nicht nur statistische Angaben über Gruppen von Neuronen zu machen, wie das in der Literatur vorzugsweise geschieht. Das heißt, ich wollte die Größe der Attraktionsbecken und die Trajektorien, auf denen jeder Anfangszustand zu einem der stabilen Zustände strebt, im Detail verfolgen. Das ließ sich aber nur bei sehr kleinen Netzwerken (bis 10 Neuronen) realisieren.
Um festzustellen, ob sich die an diesen Netzen gefundenen Prinzipien auch auf größere Netze ausdehnen lassen, habe ich auch Netze aus 32 Neuronen verwendet. In diesen ließ sich zwar nicht mehr das Verhalten jedes einzelnen Neurons in allen Fällen verfolgen, aber immerhin noch in einigen ausgewählten Fällen, z. B. in der Nähe stabiler Zustände.
Die beiden Untersuchungsstrategien, nämlich einerseits statistische Untersuchung von Neuronengruppen in großen Netzwerken und Untersuchung einzelner Neuronen in sehr kleinen Netzwerken, finden sich auch in der Biologie. Ein kausales Verständnis der Netzwerkfunktion konnte dort bisher nur durch die zweite Strategie an sehr kleinen Systemen erreicht werden. Ein Beispiel ist das stomatogastrische System der Krebse, ein Netzwerk aus 30 Neuronen, das die komplizierten Bewegungen des Magens steuert (Selverston und Moulins, 1987).

Kapitel 2

Theorie der Assoziativen Speicher

2.1 Verschiedene Arten und Modelle Assoziativer Speicher

Assoziative Speicher erfüllen zwei Bedingungen: das Vorhandensein rückgekoppelter Architekturen und das überwachte Lernen. Eine erste Unterteilung besteht in der Frage, ob binäre oder kontinuierliche Muster verwendet werden sollen. Das HOPFIELD- Netz bildet dabei die Grundlage autoassoziativer, neuronaler Speichermodelle. Es benutzt binäre Eingabe- und Ausgabemuster aufgrund der Vorzeichenfunktion, und es setzt den asynchronen Abfragemodus voraus. Durch Veränderung einzelner seiner Komponenten entstehen verschiedene assoziative Speichermodelle.

Ein Netz mit gleicher Architektur wie das HOPFIELD- Netz aber mit synchroner Abfrage ist das LITTLE- Modell (HAYKIN, 1994).
Andere Aktivierungsfunktionen statt der Vorzeichenfunktion ergeben folgende Modelle: die einfachste, denkbare Aktivierungsfunktion wäre die Identität (Linear Associative Memory, LAM). Werden zusätzlich globale, überwachte Lernregeln angewandt, können diese Netze optimiert werden (Optimal Linear Associative Memory, OLAM). Wird diese Linearität symmetrisch begrenzt, so erhält man das Brain-State-in-a-Box (BSB) Modell. In diesem werden kontinuierliche Eingabewerte zugelassen, doch binäre Ausgabemuster erzeugt. Für dreiwertige Eingabe- und Ausgabemuster benutzt OTT (1997) dreiwertige Aktivierungsfunktionen. Bei der Verwendung nichtmonotoner Aktivie-

rungsfunktionen (Nonmonotonic Activation Dynamic Associative Memory) müssen diskrete und kontinuierliche Systeme getrennt voneinander betrachtet werden. Hysteretic Activation Dynamic Associative Memories besitzen Aktivierungsfunktionen mit Hysterese und benutzen binäre Ein- und Ausgabemuster (Hassoun, 1995).
Man kann auch die Topologie des Hopfield- Netzes (s. Abschnitt 2.3) verändern. Von heteroassoziativen Speichern (s. u.) ausgehend synthetisiert man binäre autoassoziative Speicher (Temporal Associative Memory). Durch diese weit komplexeren Speicherarchitekturen können assoziative Speicher mit exponentiellen Kapazitäten realisiert werden (Exponential- Capacity Dynamic Associative Memory) oder Muster unterschiedlicher Längen gleichzeitig gespeichert werden (Sequence- Generator Dynamic Associative Memory).

In einem heteroassoziativen Speicher (Hetero Dynamic Associative Memory, HDAM) formen zwei assoziative Speicher eine geschlossene Schleife. Dabei werden die jeweiligen Speicher parallel oder sequentiell abgefragt. Sind die beiden assoziativen Speicher äquivalent, so bekommt man einen bidirektionalen, assoziativen Speicher (Bidirectional Associative Memory, BAM). Dieser assoziative Speicher weist sehr ähnliche Eigenschaften wie das Hopfield-Netz auf. (Hassoun, 1995).

2.2 Synchrone und asynchrone Abfrage

Im Little- Modell, das den synchronen Abfragemodus verwendet, benötigt man einen Taktgeber. Innerhalb eines Taktes werden die Zustände der Neuronen verwendet, **bevor** sie einzeln eventuell verändert werden. Dabei können stabile Zustände erreicht werden, es können aber auch Zyklen entstehen. Die Dynamik dieser Netze wird mit folgender Energiefunktion beschrieben:

$$E(\mathbf{x}) = -\sum_{i=1}^{N} \left| \sum_{j=1}^{N} w_{ij} x_j \right| . \tag{2.1}$$

Bei asynchroner Abfrage (Hopfield- Modell) kann das Netz zu jedem beliebigen Zeitpunkt abgefragt werden. Dabei berücksichtigt dieser Modus den zuvor veränderten Zustand einzelner Neuronen. Ferner können in Hopfield-Netzen keine Zyklen entstehen. Der Nachteil dieses Abfragemodus besteht allerdings darin, daß die Dynamik des Systems sehr stark von der Abfragereihenfolge abhängt.
Im folgenden werden asynchrone Abfragereihenfolgen verwendet.

2.3 Topologie eines HOPFIELD-Netzwerkes

Ein neuronaler assoziativer Speicher ist ein vollständig verbundenes neuronales Netzwerk, in dem jedes Neuron mit jedem anderen verbunden ist, außer mit sich selbst. Die Neuronen beeinflussen sich gegenseitig. Die Stärke der gegenseitigen Beeinflussung wird durch Gewichtsfaktoren w_{ij} festgelegt. Dabei bedeuten die Indizes, daß das Neuron i (das „sendende“ Neuron) das Neuron j (das „empfangende“ Neuron) beeinflußt (s. Abb. 1.2). Man kann die einzelnen Gewichtsfaktoren zu einer Gewichtsmatrix $\mathbf{w}$ zusammenfassen, wobei die „sendenden“ Neuronen die Reihe und die „empfangenden“ Neuronen die Spalte der Gewichtsmatrix $\mathbf{w}$ bestimmen.
Die Gewichtsmatrix $\mathbf{w}$ kann bestimmte Eigenschaften haben. Das klassische HOPFIELD-Netz erfüllt folgende Bedingungen. Alle Neuronen haben keine gerichteten Kanten auf sich selber. Daher ist die Gewichtung w_{ii} gleich Null. Die Gewichtung von Neuron i auf Neuron j in w_{ij} ist im symmetrischen Fall gleich der Gewichtung w_{ji} von Neuron j auf Neuron i. Dann gilt folgende Gleichung

$$\begin{aligned} w_{ij} &= w_{ji}, \\ w_{ii} &= 0. \end{aligned} \tag{2.2}$$

Es werden in dieser Arbeit keine andersartigen Netze untersucht.

2.4 Die Dynamik eines HOPFIELD-Netzwerkes mit asynchroner Abfragereihenfolge

2.4.1 Netzzustand

Gegeben sei ein vollständig verbundenes HOPFIELD-Netzwerk mit N Neuronen. Die Neuronen sind jeweils durch gerichtete Kanten miteinander verbunden, die je einen Gewichtungsfaktor w_{ij} haben. Jedem Neuron i mit

$$1 \leq i \leq N, \;\; i \text{ ganz}, \tag{2.3}$$

wird ein Anfangszustand $x_i \in \{\pm 1\}$ fest vorgegeben, so daß zum Zeitpunkt $t = 0$ gilt:

$$x_{i,\text{anf}} = x_i(0). \tag{2.4}$$

Der neue Zustand x_i des Neurons i zu einem Zeitpunkt t wird hier bestimmt durch die Gleichung

$$x_i(t+1) = \operatorname{sgn}[y_i(t)] \tag{2.5}$$

mit dem inneren Argument

$$y_i(t) = \sum_{j=1}^{N} w_{ij} x_j(t) \tag{2.6}$$

und der Vorzeichenfunktion

$$\operatorname{sgn}[x] = \begin{cases} +1 & : \quad x \geq 0 \\ -1 & : \quad x < 0. \end{cases} \tag{2.7}$$

Andere denkbare Vorzeichenfunktionen, z. B.

$$\operatorname{sgn}[x] = \begin{cases} +1 & : \quad x > 0 \\ -1 & : \quad x < 0 \end{cases} \tag{2.8}$$

wie bei Amit (1992) oder bei Haykin (1994) verwendet, sind im Grunde Hysteretic Activation Dynamic Associative Memories (Hassoun, 1995) und wurden nicht weiter verfolgt (Begründung siehe Abschnitt 2.4.5).
Man kann die Zustände der einzelnen Neuronen $x_i(t)$ in einen Zustandsvektor $\mathbf{x}(t)$ der Länge N zusammenfassen. Ein neuer Vektor $\mathbf{A}$ (Abfragereihenfolge, s. u.) bestimmt in seinen Elementen das Neuron i, das nach Gleichung (2.5) seinen neuen Zustand $x_i(t)$ berechnet. Dazu beachte man, daß von einem Zeitschritt zum nächsten nur ein Element des Vektors $\mathbf{x}(t)$ berechnet wird.
Mit der Gewichtsmatrix nach Gleichung (2.2) läßt sich ein absoluter Energieinhalt E zu jedem beliebigen Netzzustand $\mathbf{x}$ zum Zeitpunkt t ermitteln durch die Gleichung

$$E(\mathbf{x}(t)) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} x_i(t) x_j(t) = -\frac{1}{2} \sum_{i=1}^{N} y_i(t) x_i(t). \tag{2.9}$$

Diese Energiefunktion stellt die Lyapunovfunktion des nichtlinearen Systems dar (Pancha und Venkatesh, 1993). Damit kann man eine Aussage über das dynamische Verhalten des Systems machen. Die Lyapunovfunktion nach Gleichung (2.9) gibt an, daß das System asymptotisch stabil ist, das heißt, daß das System einem stabilen Zustand zustrebt.

Errechnet man nach Gleichung (2.5) den neuen Zustand eines Neurons, so gilt für die Energiedifferenz der Netzzustände

$$\Delta E(t) = E(\mathbf{x}(t+1)) - E(\mathbf{x}(t)) = -\frac{1}{2}\sum_{i,j=1}^{N} w_{ij}\Big(x_i(t+1)x_j(t+1) - x_i(t)x_j(t)\Big). \tag{2.10}$$

Man nehme an, daß Neuron k zum Zeitpunkt t seinen neuen Zustand nach Gleichung (2.5) berechnet. Die Differenz $\Delta x_j(t) = x_j(t+1) - x_j(t)$ ist gleich 0 oder ± 2 für $j = k$ und 0 für $j \neq k$. Damit erhält man aus Gleichung (2.10)

$$\Delta E(t) = -\Delta x_k(t)\Big(\sum_{j=1}^{N} w_{kj}x_j(t)\Big) = -\Delta x_k(t)y_k(t). \tag{2.11}$$

Das Neuron k ändert seinen Zustand nach Gleichung (2.5) nicht, wenn gilt:

$$y_k(t)x_k(t) \geq 0. \tag{2.12}$$

Ist diese Bedingung nicht erfüllt, ändert das Neuron k mit Gleichung (2.5) seinen Zustand und $\Delta x_k(t) = 2x_k(t+1)$. Die Energieänderung läßt sich dann wie folgt darstellen

$$\Delta E(t) = -2x_k(t+1)y_k(t). \tag{2.13}$$

Setzt man Gleichung (2.5) in die Gleichung (2.13) ein und beachtet die Identität $|a| = a\ \mathrm{sgn}[a]$, so folgt für die Energieänderung

$$\Delta E(t) = \begin{cases} 0 & : \quad y_k(t)x_k(t) \geq 0 \\ -2|y_k(t)| & : \quad y_k(t)x_k(t) < 0. \end{cases} \tag{2.14}$$

Die Gleichung (2.14) beweist, daß die Energiefunktion E für jeden errechneten Zustand nach Gleichung (2.5) nicht steigen kann. Als eine endliche Summe von endlichen Termen durch die begrenzte Anzahl von Neuronen ist der Wertebereich des Energieinhalts begrenzt. Ferner kann es mit der Vorzeichenfunktion nach Gleichung (2.7) keine Zyklen geben (Komlós und Paturi, 1993). Dies beweist, daß das Hopfield- Netzwerk nur stabile Zustände anstrebt. Übertragen auf die Gleichung (2.5) bedeutet dies, daß die stabilen Zustände die folgende Gleichung erfüllen:

$$x_i = \mathrm{sgn}\left[\sum_{j=1}^{N} w_{ij}x_j\right] \quad \text{für alle } i. \tag{2.15}$$

Die Ergebnisse, die man erhalten hat, kann man wie folgt zusammenfassen (Herz, 1996):

Aussage 2.1 *Wenn die Gewichtsfaktoren w_{ij} die Symmetriebedingung erfüllen, und wenn die Rückführung einzelner Neuronen auf sich selbst verschwindet, (Gleichung (2.2)), dann läßt die Dynamik des* Hopfield- *Netzwerks (Gleichung (2.5)) die Lyapunovfunktion (Gleichung (2.9)) zu und das Netz strebt nur stabile Zustände an.*

2.4.2 Die Abfrage eines einzelnen Neurons i in einem einzigen Zeitschritt

Wird mit der Gleichung (2.5) nur der Zustand eines einzigen Neurons i zum Zeitpunkt t neu berechnet, so gilt für den Vektor $\mathbf{A}$

$$\mathbf{A} = i. \tag{2.16}$$

Die Abfragereihenfolge besteht also nur aus einer Zahl, die das Neuron i bestimmt. Das Neuron i errechnet nach Gleichung (2.5) seinen neuen Zustand. Je nach dem gewählten Neuron i können so unterschiedliche Netzzustände erreicht werden (siehe dazu auch Tabelle 3.1).

2.4.3 Der sequentielle Abfragemechanismus

Es sei $\mathbf{A}$ ein Vektor der Länge N und den Elementen

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & \dots & N \end{pmatrix}. \tag{2.17}$$

Dann liegt ein asynchron sequentieller Abfragemechanismus vor. Zum Zeitpunkt $t = 0$ ist der vorgegebene Anfangszustand $\mathbf{x_{anf}}$ des Netzes vorhanden. Zu einem ersten Zeitpunkt $t = 1$ berechnet das in $A(1)$ vorgegebene Neuron $i = 1$ seinen neuen Zustand $x_1(1)$ nach Gleichung (2.5). Dieser neu berechnete Zustand geht auf der rechten Seite der Gleichung (2.5) für den nächsten Zeitpunkt $t = 2$ in die Berechnung des neuen Zustandes für das in $A(2)$ vorgegebene Neuron $i = 2$ ein. Dieses Verfahren wird so weiterhin bis zum letzten Element von $\mathbf{A}$, das hier N ist, fortgesetzt. Damit ist ein Abfrageblock abgeschlossen.
Zum Zeitpunkt N kann man jedem vorgegebenem Anfangszustand $\mathbf{x_{anf}}$ des Netzes einen Zustand $\mathbf{x}(N)$ zuordnen. Mit dieser Abfragereihenfolge ist auch gegeben, daß innerhalb dieser Zeitspanne jedes Neuron genau einmal abgefragt wird, also jedes Neuron i genau ein Element aus $\mathbf{A}$ ist.

2.4.4 Permutationen des sequentiellen Abfragemechanismus

Man kann für **A** auch andere Abfragereihenfolgen vorgeben. Wenn man die sequentielle Abfragereihenfolge betrachtet, so gibt es daraus insgesamt $N!$ mögliche Permutationen, wie man die N Neuronen anordnen kann. Unter der Voraussetzung, daß jedes Neuron in einem Abfrageblock nur einmal abgefragt wird, gibt es z.B. für $N = 3$ Neuronen neben der sequentiellen Abfragereihenfolge aus Gleichung (2.17) noch 5 weitere Abfragereihenfolgen

$$\begin{aligned} \mathbf{A^{(2)}} &= (2\ 1\ 3) \\ \mathbf{A^{(3)}} &= (3\ 1\ 2) \\ \mathbf{A^{(4)}} &= (3\ 2\ 1) \\ \mathbf{A^{(5)}} &= (2\ 3\ 1) \\ \mathbf{A^{(6)}} &= (1\ 3\ 2)\,. \end{aligned} \tag{2.18}$$

Permutationen der sequentiellen Abfragereihenfolge haben alle die Länge N und innerhalb dieser Zeitspanne ist jedes Neuron genau ein Element von **A**.

2.4.5 Zufälliger Abfragemodus

Dies ist in der Literatur (Hopfield, 1982; McEliece *et al.*, 1987; Amit, 1992; Komlós und Paturi, 1993) der normalerweise verwendete Ansatz eines asynchronen Abfragemechanismus, den ich auch hier benützen will. Ein Zufallsgenerator bestimmt, welches Neuron i zum Zeitpunkt t seinen neuen Zustand x_i nach Gleichung (2.5) berechnet. Dabei besitzt jedes Neuron i jeweils die gleiche Wahrscheinlichkeit, gewählt zu werden, mit der Wahrscheinlichkeitsdichte

$$p(i) = \frac{1}{N}, \quad i = 1 \ldots N, \quad i \text{ ganz.} \tag{2.19}$$

Der Vektor **A** besteht weiterhin aus den Elementen $1 \ldots N$, die jedoch jetzt mehrfach und auch hintereinander stehen können. Demnach können aber im allgemeinen nicht, wie in Abschnitt 2.4.3 und 2.4.4 beschrieben, in der festen Zeitspanne $t = N$ alle Neuronen i in **A** vorhanden sein. Ich habe deshalb die Länge von **A** variabel gemacht. **A** wurde so gewählt, daß jedes Neuron i mindestens einmal in **A** vorhanden ist. Das Ergebnis eines so erhaltenen Abfrageblocks läßt sich mit den Ergebnissen vergleichen, die durch Abfrageblöcke mit sequentieller bzw. Permutationen sequentieller Reihenfolgen entstehen.

Für alle Arten von Abfrageblöcken gilt:

Aussage 2.2 *Ändert sich ein Zustand* $\mathbf{x}(t)$ *nach Berechnung der Gleichung (2.5) mit einem Abfrageblock* $\mathbf{A}$*, in dem jedes Neuron* $i, 1 \leq i \leq N, i$ *ganz, mindestens einmal abgefragt wurde, nicht, so handelt es sich um einen stabilen Zustand (Gleichung (2.15)).*

Begründung: Die Gewichtsmatrix $\mathbf{w}$ sei symmetrisch und mit Null in ihrer Diagonalen nach Gleichung (2.2). Dann läßt sich nach Gleichung (2.9) zu jedem Zustand des Netzes ein Energieinhalt ermitteln. Man beachte, daß das innere Argument y_i in Gleichung (2.6) in der Energiefunktion nach Gleichung (2.9) enthalten ist. Mit der Gleichung (2.5) kann der Zustand eines einzelnen Neurons $x_i(t+1)$ berechnet werden. Eine Zustandsänderung eines einzelnen Neurons bewirkt auch eine Änderung des Gesamtzustands $\mathbf{x}$ in der Art, daß der neue Zustand einen gleichen (bei $y_i = 0$) oder niedrigeren Energieinhalt besitzt. Obwohl die Energiedifferenz für den Fall $y_i = 0$ keinen Einfluß auf den nächsten Zustand hat, gibt die Vorzeichenfunktion in Gleichung (2.7) den Zustand des Neurons $x_i(t+1)$ auch für diesen Fall eindeutig mit $x_i(t+1) = 1$ an. Eine eventuelle Änderung des Zustands des Neurons x_i muß in weiteren Zeitschritten $t > (t+1)$ berücksichtigt werden.
Durch die eindeutige Zuordnung des Falles $y_i = 0$ durch die Gleichung (2.7) wird verhindert, daß ein Netzzustand $\mathbf{x}(t)$ nach einer Veränderung durch die Gleichung (2.5) durch eine spätere Abfrage wieder erreicht werden kann. Es entstehen also keine Zyklen, wie sie bei Verwendung der Gleichung (2.8) auftreten könnten (Komlós und Paturi, 1993). Erst dadurch wird die Verwendung der Energiefunktion nach Gleichung (2.9) möglich.
Der Gesamtzustand wird von den Zuständen aller Neuronen bestimmt. Deshalb müssen alle Neuronen in einem Abfrageblock mindestens einmal abgefragt werden. Ändert sich in einem Abfrageblock, in dem jedes Neuron mindestens einmal abgefragt wurde, der Gesamtzustand nicht, muß dieser Zustand ein lokales Minimum des Energieinhalts repräsentieren, also einen stabilen Zustand darstellen.

2.5 Darstellbarkeit der Zustandsvektoren

Die Elemente $x_i(t) = \pm 1$ des Zustandsvektors $\mathbf{x}(t)$ der Größe N können in Elemente $x_{i,binär} = (0 \wedge 1)$ durch die Gleichung

$$x_{i,binär} = \frac{x_i(t) + 1}{2} \tag{2.20}$$

umgeformt werden. Dabei wird jedem Zustand $x_i = 1$ des Neurons i der Wert $x_{i,binär} = 1$ und jedem Zustand $x_i = -1$ des Neurons i der Wert $x_{i,binär} = 0$ zugeordnet. $x_{1,binär}x_{2,binär}\ldots x_{N,binär}$ bildet somit eine binäre Zahl, die einer dezimalen Zahl zugeordnet werden kann und umgekehrt. Dadurch kann jeder Zustand $\mathbf{x}$ der 2^N möglichen Zustände des Netzes in einer Dezimalzahl ausgedrückt werden und umgekehrt.
Diese Darstellbarkeit der Zustandsvektoren sind nur für Netze mit $N \leq 32$ empfehlenswert, da bereits hier Zustandsvektoren in Dezimaldarstellung 10- stellig sein können.

2.6 Bestimmung der Gewichtsfaktoren

Bisher wurden die Eigenschaften solcher Netze behandelt, deren Gewichtsfaktoren als bekannt vorausgesetzt waren. Eine wichtige Frage ist es natürlich, wie man solche Gewichtsfaktoren bestimmt.

2.6.1 Die Lernregel nach Hebb, Attraktionsbecken, Speicherkapazität

Hopfield (1982) benutzte für die Beschreibung der Gewichtsfaktoren zwischen Neuronen die folgende **Lernregel** (Hebb, 1949). Mit dieser Regel ist es möglich, den Vektor eines stabilen Zustands durch Gewichtsfaktoren festzulegen. Dazu ist erforderlich, daß für den gewünschten stabilen Zustand jede Verbindung zwischen zwei Neuronen die Korrelation der Zustände beider Neuronen abspeichert. Genauer gesagt, man setzt die Gewichtsfaktoren

$$\begin{aligned} w_{ij} &= \xi_i \xi_j, \quad i \neq j \\ w_{ii} &= 0, \end{aligned} \tag{2.21}$$

um einen einzelnen Vektor

$$\xi = (\xi_1, \xi_2, \ldots, \xi_n) \tag{2.22}$$

abzuspeichern. Mit dieser Wahl der Gewichtsfaktoren besitzt das System beim Zustand ξ einen stabilen Zustand.
Ferner hat diese Art der Bestimmung der Gewichtsfaktoren eine weitere Eigenschaft. Dazu betrachte man den Hamming Abstand

$$d(\mathbf{x^{(1)}}, \mathbf{x^{(2)}}) = \frac{1}{2} \sum_{i=1}^{N} |x_i^{(1)} - x_i^{(2)}|. \tag{2.23}$$

Der Hamming Abstand d ist die Anzahl der Elemente, in der sich zwei Netzzustände $\mathbf{x^{(1)}}$ und $\mathbf{x^{(2)}}$ unterscheiden. Wenn ein Netzzustand $\mathbf{x}$ mit einem Hamming-Abstand d von maximal $\frac{N}{2}$ von dem gespeicherten Netzzustand ξ entfernt ist, wird der eingeprägte Zustand ξ in einem synchronen Zeitschritt erreicht. Deshalb berechnet sgn[$\mathbf{Wx}$] für jeden Vektor $\mathbf{x}$, der innerhalb einer Entfernung von $\frac{N}{2}$ von ξ liegt, diesen Vektor ξ. Beim asynchronen Abfragemechanismus konvergiert der Zustand des Netzes monoton zu dem Zustand ξ (Komlós und Paturi, 1993). Die Tatsache, daß das System auf bestimmte Zustände konvergiert, bedeutet, daß eine Fehlerkorrektur möglich ist.
Um mehrere Vektoren $\xi^{(1)}, \ldots, \xi^{(p)}$ in einem System zu speichern, benutzt man die Summe der zugehörigen Gewichtsfaktoren. So wird die Gewichtsmatrix $\mathbf{w}$ durch die Elemente

$$w_{ij} = \sum_{t=1}^{p} \xi_i^t \xi_j^t, \quad (i \neq j),$$

$$w_{ii} = 0 \tag{2.24}$$

definiert. Sollten bei vielen gespeicherten Vektoren die einzelnen Gewichtsfaktoren zu groß werden, können sie durch Normierung (Hopfield, 1982; Amit, 1992; Herz, 1996) oder Begrenzung (Mazza, 1997) verkleinert werden. Dieses Verfahren wurde hier nicht angewandt. Die Hoffnung bei der Addition der Gewichtsfaktoren ist, daß, wenn die gespeicherten Vektoren ausreichend unterschiedlich sind, solch eine lineare Addition der Gewichtsfaktoren keine große Auswirkung auf das Fehlerkorrekturverhalten des Systems hat. Man beachte, daß das System nicht die einzelnen Vektoren ξ speichert, sondern nur die Gewichtsfaktoren, die im wesentlichen die Korrelationen zwischen den Vektoren repräsentieren. Wenn ein neuer Vektor ξ gespeichert werden soll, lernt also das System dadurch, daß zu den bestehenden Gewichtsfaktoren w_{ij} die neuen Faktoren $\xi_i \xi_j$ dazuaddiert werden. Man nennt jeden solchen gespeicherten Vektor ein fundamentales Muster.

Wenn mehrere Vektoren als fundamentale Muster eingespeichert werden, ist das Erkennen eines bestimmten Musters durch das Rauschen, das durch die anderen eingespeicherten Muster erzeugt wurde, in Frage gestellt. Denn, wenn ein fundamentales Muster $\xi^{(\mathbf{b})}, 1 \leq b \leq p$ in das System eingegeben wird, gilt für das innere Argument nach Gleichung (2.6) für das Neuron i

$$y_i = \sum_{j=1}^{N} w_{ij}\xi_j^{(b)} = \sum_{\substack{j=1,\\ j\neq i}}^{N} \sum_{t=1}^{p} \xi_i^{(t)}\xi_j^{(t)}\xi_j^{(b)} \tag{2.25}$$

Vertauscht man die Summenzeichen und vervollständigt die Summe, so daß die Summe über alle j gültig ist, so muß man den Term für $j = i$ wieder abziehen.

$$y_i = \sum_{t=1}^{p} \left(\sum_{j=1}^{N} \xi_i^{(t)}\xi^{(j)}\xi_j^{(b)} - \xi_i^{(t)}\xi_i^{(t)}\xi_i^{(b)} \right) \tag{2.26}$$

Da $\xi_i^{(t)}\xi_i^{(t)} = 1$ ist und über t summiert wird, folgt

$$y_i = \sum_{j=1}^{N} \sum_{t=1}^{p} \xi_i^{(t)}\xi_j^{(t)}\xi_j^{(b)} - p\xi_i^{(b)}. \tag{2.27}$$

Berechnet man daraus den Energieinhalt E für das einzuspeichernde Muster $\xi^{(\mathbf{b})}$ nach Gleichung (2.9), so folgt

$$E = -\frac{1}{2} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{t=1}^{p} \xi_i^{(t)}\xi_i^{(b)}\xi_j^{(t)}\xi_j^{(b)} - \sum_{i=1}^{N} p\xi_i^{(b)}\xi_i^{(b)} \right). \tag{2.28}$$

Nun kann man die Summen vertauschen, so daß man Skalarprodukte entnehmen kann

$$E = -\frac{1}{2} \left(\sum_{t=1}^{p} \left(\sum_{i=1}^{N} \xi_i^{(t)}\xi_i^{(b)} \right) \left(\sum_{j=1}^{N} \xi_j^{(t)}\xi_j^{(b)} \right) - Np \right). \tag{2.29}$$

Geht man von orthogonalen fundamentalen Mustern aus (s. unten), so ist das Skalarprodukt zwischen den einzelnen Mustern $1 \leq t \leq p,\ \ t \neq b$ gleich Null. Das Skalarprodukt des eingegebenen fundamentalen Musters mit sich selbst in dem Fall für $t = b$ ergibt N. So folgt für diesen Fall für den Energieinhalt aller fundamentalen Muster $\xi^{(\mathbf{b})}, 1 \leq b \leq p$

$$E = -\frac{1}{2}(N^2 - Np). \tag{2.30}$$

Wird nur ein fundamentales Muster in das Netz gespeichert, so beträgt der Energieinhalt dieses Zustands mit $p = 1$ nach Gleichung (2.29)

$$E_s = -\frac{1}{2}(N^2 - N) = -\frac{1}{2}N(N-1). \tag{2.31}$$

Dabei ist dieser Wert die Energie des fundamentalen Musters, wenn dieses fundamentale Muster als Anfangszustand in das Netzwerk eingegeben wird. Wenn mehrere Muster p nach HEBB in das Netz nach Gleichung (2.24) eingespeichert wurden, dann wird die Energie E des fundamentalen Musters nach Gleichung (2.29), verglichen mit der Energie nach Gleichung (2.31), durch einen Rauschterm gestört mit

$$E_r = E - E_s. \tag{2.32}$$

Im Falle von orthogonalen fundamentalen Mustern hieße das

$$E_r = -\frac{1}{2}\Big((N^2 - Np) - N(N-1)\Big) = -\frac{1}{2}N(1-p). \tag{2.33}$$

Aus Gleichung (2.31) und (2.32) läßt sich ein Signal-zu-Rausch-Verhältnis

$$\text{SNR (dB)} = 10 \cdot \log_{10}\left(\frac{E_s}{E_r}\right)^2, \tag{2.34}$$

bilden.
Für den Fall orthogonaler fundamentaler Muster ergibt sich daraus

$$\text{SNR (dB)} = 10 \cdot \log_{10}\left(\frac{N-1}{1-p}\right)^2. \tag{2.35}$$

Dieses Ergebnis ist plausibel. Für ein fundamentales Muster ($p = 1$) wird mit einer konstanten Anzahl von Neuronen N das Signal-zu-Rausch-Verhältnis unendlich, d. h. es ist kein Rauschen vorhanden. Ferner beeinflussen sich orthogonale fundamentale Muster immer weniger gegenseitig, je größer die Zahl der Neuronen N im Netzwerk ist.
Man kann auch auf eine andere Weise auf den Signalenergieinhalt in Gleichung (2.31) kommen, wenn man aus Gleichung (2.25) den Term für $t = b$ extrahiert.

$$y_i = y_s + y_r = (N-1)\xi_i^{(b)} + \sum_{\substack{j=1,\\ j\neq i}}^{N} \sum_{\substack{t=1,\\ t\neq b}}^{p} \xi_i^{(t)}\xi_j^{(t)}\xi_j^{(b)} \tag{2.36}$$

Wenn man aus y_s den Energieinhalt E_s bestimmen will, so folgt daraus Gleichung (2.31). y_r ist allerdings nicht geschlossen darstellbar. Da aber das innere Argument von der Energiefunktion abhängt und umgekehrt, bedeutet dies, daß man das innere Argument in einen Signalterm y_s und in einen Rauschterm y_r aufteilen kann. *Obwohl der eigene Zustand des Neurons i nicht in y_i mit eingeht, kann man für das innere Argument in Analogie zur Energiefunktion von einem Signal- und Rauschanteil reden.*

Man kann aber hoffen, daß der Rauschterm y_r in Gleichung (2.36) nicht zu groß wird, wenn die Anzahl der gespeicherten Muster p nicht zu groß ist. Dies ist dann der Fall, wenn jedes der gespeicherten Muster nicht nur stabil ist, sondern auch andere Vektoren innerhalb einer gewissen Entfernung $\rho > 0$ anzieht.
Man kann den Radius ρ eines solchen **Attraktionsbecken** auf unterschiedliche Weise definieren. Beide Definitionen gehen allerdings davon aus, daß ein solches Attraktionsbecken im Mittel rotationssymmetrisch ist, also überall etwa gleich hohe „Ränder“hat. Wie die vorliegende Arbeit zeigt, ist diese Annahme allerdings teilweise eine grobe Vereinfachung.

1. Wenn man einen Attraktionsradius $\rho > 0$ mit statistischen Termen ausdrücken will, so wird folgendes Verfahren vorgeschlagen (Pancha und Venkatesh, 1993). Es werden Anfangszustände gebildet, die einen bestimmten Hamming Abstand d vom stabilen Zustand $\mathbf{f}$ entfernt sind. Dies geschieht dadurch, daß d zufällig ausgewählte Komponenten ihr Vorzeichen wechseln. Von dort aus wird ein stabiler Zustand angestrebt. Ist dieser stabile Zustand der stabile Zustand $\mathbf{f}$, so war eine Fehlerkorrektur möglich. In einem Durchgang wird d von eins bis zu einem maximalen Wert d_{max} gesteigert, bei dem gerade noch eine Fehlerkorrektur möglich ist. Es werden mehrere solche Durchgänge durchgeführt und am jeweils erhaltenem d_{max} der mittlere Attraktionsradius $\rho = d_{max}/N$ errechnet.
   Doch diese Art der Bestimmung der Attraktionsradii um stabile Zustände führt nicht zu einem Attraktionsbecken in dem Sinne, daß alle Zustände innerhalb des Hamming Abstands von ρN angezogen werden, d.h. auf $\mathbf{f}$ zustreben. Es kann nur eine Wahrscheinlichkeit angegeben werden, daß ein bestimmter Zustand innerhalb eines Attraktionsradius ρN vom stabilen Zustand $\mathbf{f}$ von diesem stabilen Zustand $\mathbf{f}$ angezogen wird. Diese Wahrscheinlichkeit hängt vom Rauschterm in Gleichung

(2.36), von der zufälligen Wahl der Anfangszustände und der zufälligen Wahl der Abfragereihenfolge ab.

2. Ein Gleichgewichtszustand mit einem Attraktionsbecken, oder äquivalent, ein asymptotisch stabiler Gleichgewichtszustand besitzt nach Golden (1993) die Eigenschaft, daß ein Anfangszustand, der nahe genug an solch einem Gleichgewichtszustand beginnt, immer (und nicht nur mit einer gewissen Wahrscheinlichkeit) in diesen Gleichgewichtszustand konvergiert. Da meist eine große Anzahl p von fundamentalen Mustern ξ in das Netz eingegeben werden, beeinflußt der Rauschterm in Gleichung (2.36) das Verhalten des Systems außerordentlich. Ferner werden die fundamentalen Muster zufällig gewählt (siehe unten). Wenn man zufällige Anfangszustände wählt, (also nicht nur solche in der Nähe von stabilen Zuständen wie oben) können sehr große Unterschiede in den Attraktionsbecken entstehen. McEliece *et al.* (1987) zeigten, daß im günstigsten Fall der Attraktionsradius ρ nahe $\frac{1}{2}$ sein kann. Auf der anderen Seite war im ungünstigsten Fall, wie Montgomery und Vijaya Kumar (1986) zeigten, schon ein Attraktionsradius von $\rho > \frac{1}{8}$ unmöglich, sogar wenn die Anzahl der Muster p nur 3 ist.

Wenn man sich auf asynchrone Abfrage beschränkt, kann man das Verhalten des Systems als eine Funktion der Zeit betrachten, in der eine geordnete Sequenz von Zuständen (Trajektorien) verfolgt wird (Golden, 1993). Dabei repräsentiert ein Netzzustand einen Punkt in einem einheitlich (nur Werte von ± 1) ausgebreiteten, N- dimensionalen Zustandsraum. Diese Sequenz (Trajektorie) kann von der Abfragereihenfolge abhängen (Personnaz *et al.*, 1986; McEliece *et al.*, 1987; Kamp und Hasler, 1990). Damit könnte die Abfragereihenfolge auch einen Einfluß auf die Größe der Attraktionsbecken haben.
Die erheblichen Unterschiede in der Größe der Attraktionsbecken waren bei der Wahl zufälliger Anfangszustände sichtbar. Für die Wahl zufälliger Abfragereihenfolgen erwartete ich einen ähnlichen Effekt. Deshalb kam ich zunächst auf die Idee, alle Zustände und alle Abfragereihenfolgen in Form aller Permutationen der sequentiellen Reihenfolge (s. Abschnitt 2.4.4) zu verwenden. So wird es möglich, die Attraktionsradii stabiler Zustände direkt und ohne Mittelung einzelner Durchläufe zu erhalten, da alle abfragebedingten Trajektorien bekannt sind. Dadurch war auch eine statistische Untersuchung

nicht weiter erforderlich. Allerdings war dieses Verfahren aus Gründen der Rechenintensität nur bis zu einer Netzgröße von $N = 10$ Neuronen möglich. Bei größeren Netzen wurde die zufällige Abfragereihenfolge (Abschnitt 2.4.5) gewählt.

Man kann das System nicht nur dann als fehlerkorrigierend betrachten, wenn alle Anfangszustände, die innerhalb eines Attraktionsbeckens ρ liegen, auf ein eingespeichertes Muster konvergieren, sondern auch dann, wenn jeder Vektor, der innerhalb einer Entfernung ρ von einem gespeicherten Muster liegt, innerhalb einer Entfernung ϵ, für $\epsilon < \rho$, vom Muster endet. Dabei nennt man ϵ den **residuellen Fehler**. Dies gilt für beide obige Definitionen von ρ. Es stellt sich heraus, daß der residuelle Fehler ϵ und der Attraktionsradius ρ eine Funktion der Anzahl der eingespeicherten Muster p und der Netzgröße N ist (siehe unten). Es ist eine Frage der Definition, ob man diesen Fall unter spurious states einordnet oder nicht. Hier wird er nicht darunter eingeordnet (Begründung s. Abschnitt 3.2).

Wahl der Muster. Man bekommt kein gutes Fehlerkorrekturverhalten für Muster, die sehr ähnlich sind. Deshalb werden die fundamentalen Muster **zufällig** gewählt. Man erwartet, daß das System jedes dieser fundamentalen Muster mit einer Wahrscheinlichkeit nahe 1 speichert. Diese zufällige Wahl der Muster wird meistens dadurch erreicht, daß die Eingabe zuvor codiert wird.
McEliece *et al.* (1987) berechneten die maximale Anzahl p der einzuspeichernden Muster. Sie setzten dabei voraus, daß die einzuspeichernden Muster zufällig gewählt wurden, daß alle Muster stabile Zustände ergeben und daß eine Fehlerkorrektur möglich ist:

1. Wenn
$$p < N\frac{(1-2\rho)^2}{4\log N} \tag{2.37}$$
ist, dann sind, mit einer Wahrscheinlichkeit nahe 1, alle fundamentalen Muster, die eingespeichert werden, stabile Zustände des Netzes. Ferner kann das System für jedes fundamentale Muster die meisten Zustände, die mit dem Hamming Abstand von weniger als ρN vom fundamentalen Muster entfernt sind, in einem einzigen synchronen Zeitschritt anziehen.

2. Wenn
$$\frac{(1-2\rho)N}{(4\log N)} < p < \frac{(1-2\rho)^2 N}{2\log N} \tag{2.38}$$
ist, dann werden immer noch die meisten fundamentalen Muster stabile Zustände ergeben und die meisten Zustände mit dem Hamming Abstand von weniger als ρN in einem synchronen Schritt anziehen.

Newman (1988) deutet auf die von ihm erhaltenen Ergebnisse der Energielandschaft aus der Energiefunktion nach Gleichung (2.9) hin und sieht den Begriff der Speicherung anders:

3. Wenn die fundamentalen Muster nicht exakt im Netz gespeichert sind, kann man noch stabile Zustände in ihrer Nachbarschaft finden (residueller Fehler).

Komlós und Paturi (1988) analysierten die Frage des ungünstigsten Falls und bewiesen folgende Ergebnisse. Es gibt Konstanten $\alpha, \rho < \rho_{\text{rand}}$ in der Weise, daß die folgenden Eigenschaften mit einer Wahrscheinlichkeit nahe 1 für eine zufällige Wahl der fundamentalen Muster und asynchroner Abfrage gelten:

4. Wenn $p \leq \alpha N$ ist (mit α klein) und wenn ein Anfangszustand zufällig mit dem Hamming Abstand von ρN von einem fundamentalen Muster $\mathbf{f}$ gewählt wurde, dann wird das Netz zu einem stabilen Zustand innerhalb eines Hamming Abstands von
$$Ne^{-\frac{N}{4p}} \tag{2.39}$$
von $\mathbf{f}$ konvergieren. Das heißt, wenn
$$p < \frac{N}{4\log N} \tag{2.40}$$
ist, so wird das System zum fundamentalen Muster $\mathbf{f}$ konvergieren.

5. Es gilt für jedes fundamentale Muster $\mathbf{f}$: Der maximale Energieinhalt jedes Zustandes innerhalb des Hamming Abstands ρN von $\mathbf{f}$ ist geringer als der minimale Energieinhalt jedes Zustandes mit dem Hamming Abstand von $\rho_{\text{rand}} N$ von $\mathbf{f}$. Ferner gibt es keine lokalen Minima der Energiefunktion, die mit einem Radius von $\rho_{\text{rand}} N$ und $Ne^{(-n/(4p))}$ von $\mathbf{f}$ entfernt sind. Die Existenz solcher hoher „Energiebarrieren“ um fundamentale Muster herum ist notwendig, um mit asynchroner Abfrage das Konvergieren auf die stabilen Zustände zu etablieren.

Komlós und Paturi (1993) zeigten damit:

6. Wenn p (die Anzahl der eingespeicherten fundamentalen Muster) kleiner als $N/(4 \log N)$ ist, so haben die fundamentalen Muster ein Attraktionsbecken von mindestens einem Radius von ρN mit $\rho = 0.0024$, und sowohl die synchrone, als auch die asynchrone Abfrage konvergieren sehr schnell.

7. Wenn $p = \alpha N$ ist (mit α klein), dann streben alle Anfangszustände innerhalb des Hamming Abstands ρN vom fundamentalen Muster in einen stabilen Zustand, der innerhalb einer Entfernung mit dem Hamming Abstand ϵN vom fundamentalen Muster $\mathbf{f}$ liegt, wobei ϵ ungefähr $e^{(-1/(4\alpha))}$ ist.

8. Die vielen spurious states bei größeren Netzen und hoher Anzahl der fundamentalen Muster besitzen dieselben Konvergenzeigenschaften wie die fundamentalen Speicher nach Punkten 5 bis 7. Der Begriff „spurious states“, wie er hier verwendet wird, umfaßt nicht die Verschiebung der Muster (residueller Fehler) nach den Punkten 3 und 7.

Kanter und Sompolinsky (1987) befaßten sich mit der Lage der spurious states und stellten fest:

9. Durch die Bestimmung der Selbstkopplungen zu $w_{ii} = 0$ können sich spurious states in der Nachbarschaft eines gewünschten Attraktors bilden.

Die geschilderten Untersuchungen behandeln wegen des komplexen dynamischen Verhaltens der Netze nur die Umgebung von stabilen Zuständen. Andere Verfahren beschränken sich auf dieselbe Umgebung (Amari, 1993). Für die Verwendung von assoziativen Speichern für die Mustererkennung wären aber zusätzlich Aussagen über die genaue Struktur der Ränder der Attraktionsbecken notwendig. Darüber gibt es aber bisher keine Befunde.

Speicherkapazität. Hopfield (1982) führte das Konzept einer Energiefunktion (Lyapunovfunktion) in assoziativen Speichern mit einer symmetrischen Verbindungsmatrix ein. Dies wurde in Analogie zu der „spin-glass“ Theorie der statistischen Physik gezeigt. Hopfield gab die Kapazität p an (maximale Anzahl der fundamentalen Mustern in einem Netz aus N Neuronen). Durch Computersimulationen ergab sich, daß die Kapazität ungefähr

$$p = 0.15N \tag{2.41}$$

betrage, wenn N sehr groß ist. Diese Kapazität wird die relative Kapazität genannt. Die theoretische Berechnung der relativen Kapazität mit

$$p = 0.14N \tag{2.42}$$

wurde von Amit *et al.* (1985a,b) durchgeführt. Sie benutzen die Replika-Methode, welche zwar noch nicht bewiesen ist, von der man aber annimmt, daß sie eine gute Approximation liefert.

2.6.2 Projektionsregel und iterative Lernregeln

Für das Einspeichern fundamentaler Muster, die beliebige Korrelationen besitzen, bietet sich auch die **Projektionsregel** an (Kohonen, 1988). Dabei werden die Gewichtsfaktoren über die Korrelationsmatrix

$$C_{\mu\nu} = \sum_{i=1}^{N} \xi_i^{(\mu)} \xi_i^{(\nu)}, \ \mu, \nu = 1, \ldots, p. \tag{2.43}$$

wie folgt bestimmt:

$$w_{ij} = \sum_{\mu,\nu=1}^{p} (C^{-1})_{\mu\nu} \, \xi_i^{(\mu)} \xi_j^{(\nu)}. \tag{2.44}$$

Die Muster stellen nun Eigenfunktionen der Gewichtsmatrix dar (zunächst seien hier Selbstkopplungen erlaubt). Da die Energiefunktion (2.9) am Muster ein Minimum hat, können so bis zu $p = N$ (für $p > N$ existiert $(C^{(-1)})_{\mu\nu}$ nicht) linear unabhängige Muster eingespeichert werden.
Obwohl sich Selbstkopplungen zur Trennung stark korrelierter Muster eignen, führen sie im allgemeinen zu einer starken Einschränkung der Attraktionsbecken (Kanter und Sympolinsky, 1988).
Will man die so erhaltenen Attraktionsbecken mit Attraktionsbecken vergleichen, die durch die Lernregel nach Hebb entstehen, müssen folgende Bedingungen erfüllt sein: Die Diagonalelemente in der Gewichtsmatrix $\mathbf{w}$ werden auf Null gestzt

$$w_{ii} = 0. \tag{2.45}$$

Die Muster müssen linear unabhängig sein. Dies ist bei allen in dieser Arbeit verwendeten Mustern der Fall.
Werden $p = \alpha N$ zufällig generierte fundamentale Muster in größeren Netzwerken mit $\alpha \leq 0.14$ gespeichert, so ergibt sich für die Projektionsregel wieder die Lernregel nach Hebb (Schnelle, 1991).

Weitere Lernregeln zur Speicherung korrelierter Muster verwenden **iterative Methoden** (Gardner, 1988; Krauth *et al.*, 1988; Diederich und Opper, 1987). Sie nutzen neben den lokalen Informationen über die Muster zusätzlich die beim Anlegen der Muster erzeugten Attraktionsbecken. Ihr Ziel ist nicht nur, die Muster einzuspeichern, sondern gleichzeitig und gleichberechtigt in mehreren Lernschritten auch die lokalen Attraktionsbecken zu optimieren. Die Gewichtsmatrix ist an folgende Bedingungen anzupassen:

$$\xi_i^\mu \sum_{i \neq j} w_{ij} \xi_j^\mu \geq \kappa, \text{ für alle } i, \mu, \tag{2.46}$$

wobei der Stabilitätsparameter $\kappa \geq 0$ zu wählen ist.
Durch die iterative Bestimmung der Gewichtsfaktoren kann für sie im allgemeinen kein geschlossener Ausdruck angegeben werden. Zudem werden oft nichtsymmetrische Gewichtsmatrizen erzeugt. Es erscheint jedoch plausibel, daß große Werte von κ auch große Attraktionsbecken sichern (Schnelle, 1991).
Der Vorteil iterativer Prozeduren besteht darin, daß oft Konvergenzbeweise existieren oder sogar die Zahl der notwendigen Lernzyklen analytisch in Abhängigkeit von der Musteranzahl p und κ bestimmt werden kann. Weitere Vorteile zeigt zum Beispiel folgende iterative Lernregel, die sich als äquivalent zur Projektionsregel erweist (Diederich und Opper, 1987)

$$w_{ij}(t+1) = w_{ij}(t) + \gamma/N \sum_\nu e_i^\nu(t) \xi_j^\nu, \text{ wobei}$$

$$e_i^\nu(t) = \xi_i^\nu - \sum_k w_{ik}(t) \xi_k^\nu \text{ und } w_{ij}(0) = 0. \tag{2.47}$$

Erstens wird die bei der Projektionsregel bei jedem neu zu lernenden Muster notwendige Matrixinversion, die in großen Systemen sehr aufwendig wird, aufgehoben. Ferner wird jedes neue Muster, wie bei der Lernregel nach Hebb, in einem bereits vorhandenen Netzwerk gespeichert. Dadurch ist diese iterative Lernregel wie die Lernregel nach Hebb eine lokale Lernregel (Amit, 1992).
Zusammenfassungen weiterer iterativer Ansätze, insbesondere der Eigenstruktur Methode findet sich bei Liu und Lu (1997). Weitere Vorschläge sind die Implementierung eines „Designer“ Netzwerks (Perfetti, 1991) und der duale Spektralalgorithmus (Pancha und Venkatesh, 1993). Sie benutzen zum Teil zur Lösung linearer Ungleichungen die Methode der linearen Programmierung.

Kapitel 3

Untersuchungen an kleinen Netzwerken

3.1 Netzwerke aus 5 oder 6 Neuronen

3.1.1 Grundlegende Eigenschaften an einem Beispiel

Um die Charakteristika assoziativer Speicher anhand eines Beispiels klar zu machen, benutzt Amit (1992) folgendes sehr kleines Hopfield-Netzwerk mit $N = 5$ Neuronen. Dem Netzwerk werden zwei Muster eingeprägt, obwohl das Netzwerk damit nach Abschnitt 2.6.1 eigentlich überladen ist.

$$\begin{aligned} \xi^{(\mathbf{1})} &= \begin{pmatrix} +1 & -1 & +1 & +1 & -1 \end{pmatrix}, \\ \xi^{(\mathbf{2})} &= \begin{pmatrix} +1 & +1 & +1 & -1 & -1 \end{pmatrix} \end{aligned} \tag{3.1}$$

Wenn man nach Abschnitt 2.5 die Muster $\xi^{(\mathbf{1})}$ und $\xi^{(\mathbf{2})}$ nach Gleichung (3.1) in die Dezimaldarstellung nach Gleichung (2.20) umformt, so steht für $\xi^{(\mathbf{1})}$ der Zustand $\xi^{(\mathbf{1})}_{\mathbf{dec}} = 22$ und für $\xi^{(\mathbf{2})}$ der Zustand $\xi^{(\mathbf{2})}_{\mathbf{dec}} = 28$.
Nach Hebb in Gleichung (2.24) ergibt sich die folgende Gewichtsmatrix $\mathbf{w}$

$$\mathbf{w} = \begin{pmatrix} 0 & 0 & +2 & 0 & -2 \\ 0 & 0 & 0 & -2 & 0 \\ +2 & 0 & 0 & 0 & -2 \\ 0 & -2 & 0 & 0 & 0 \\ -2 & 0 & -2 & 0 & 0 \end{pmatrix}. \tag{3.2}$$

Da sich die Zustände $\mathbf{x_{dec}}$ und $(2^N - 1) - \mathbf{x_{dec}}$ nur im Vorzeichen unterscheiden, sind die Antimuster $-\xi^{(\mathbf{1})}_{\mathbf{dec}} = 31 - (\xi^{(\mathbf{1})}_{\mathbf{dec}}) = 9$ und $-\xi^{(\mathbf{2})}_{\mathbf{dec}} = 31 - (\xi^{(\mathbf{2})}_{\mathbf{dec}}) = 3$

mit

$$\begin{aligned} -\xi^{(1)} &= \begin{pmatrix} -1 & +1 & -1 & -1 & +1 \end{pmatrix}, \\ -\xi^{(2)} &= \begin{pmatrix} -1 & -1 & -1 & +1 & +1 \end{pmatrix}. \end{aligned} \tag{3.3}$$

Wie sich im folgenden zeigen wird, sind beide Muster und beide jeweiligen Antimuster trotz der Überladung des Netzes auch stabile Zustände. Dieses Netz habe ich als erstes im Detail untersucht.
In einem ersten Analyseschritt wird **jeweils nur ein einzelnes Neuron abgefragt**. Dies erfolgt von allen möglichen 2^5 Zuständen des Netzes aus und jeweils getrennt für jedes der 5 Neurone. Die Abfrage erfolgt gemäß Gleichung (2.5). Tab. 3.1 zeigt in der ersten Spalte alle 2^N Zustände des HOPFIELD-Netzes als Anfangszustände. In den folgenden Spalten sind die Netzzustände angegeben, die entstehen, wenn das am Kopf der Spalte angegebene Neuron als einziges abgefragt wird. Ändert sich der Zustand nicht, so wird dieser neue Zustand grau unterlegt. Stabile Zustände sind fett gedruckt. Man erkennt an Tab. 3.1:

1. Wenn man die Muster oder Antimuster als Anfangszustand eingibt, dann wird deutlich, daß es mit keiner Abfrage eines Neurons i nach Gleichung (2.5) möglich ist, einen anderen Zustand zu berechnen als den Anfangszustand selbst. Sie werden als stabile Zustände des Netzwerks bezeichnet.

2. Mit der Gewichtsmatrix $\mathbf{w}$ in Gleichung (3.2) sind also die zwei Muster $\xi^{(1)}$ und $\xi^{(2)}$ aus Gleichung (3.1) assoziativ gespeichert. Da der Ansatz nach HEBB in Gleichung (2.24) ein rein symmetrischer Ansatz ist, gibt es zu jedem Muster ein Antimuster mit exakt umgekehrtem Vorzeichen. Demnach sind $\mathbf{x_{anf}} = (-\xi^{(1)} \wedge -\xi^{(2)})$ auch stabile Zustände des Netzes, und sie sind den jeweiligen Mustern $\xi^{(1)}$ und $\xi^{(2)}$ gleichwertig in ihren absoluten Energieinhalten (Tab. 3.2). Muster und Antimuster sind symmetrisch zueinander.

3. Von einem Anfangszustand aus können verschiedene stabile Zustände erreicht werden, je nachdem welches Neuron abgefragt wird (z. B. Zustände 1 und 20).

4. Es gibt Zustände, die sich nur ändern, wenn ein bestimmtes Neuron abgefragt wird und der neue Zustand dann einem stabilen Zustand

Anfangszustand $\mathbf{x_{anf}}$	Neuron 1	Neuron 2	Neuron 3	Neuron 4	Neuron 5
0	16 *	8	4 *	2	1
1	1	**9**	1	**3**	1
2	18 *	2	6 *	2	**3**
3	**3**	**3**	**3**	**3**	**3**
4	20	12	4 *	6	5 *
5	21 *	13	1	7	5 *
6	**22**	6	6 *	6	7 *
7	23 *	7	**3**	7	7 *
8	24 *	8	12 *	8	**9**
9	**9**	**9**	**9**	**9**	**9**
10	26 *	2	14 *	8	11
11	11	**3**	11	**9**	11
12	**28**	12	12 *	12	13 *
13	29 *	13	**9**	13	13 *
14	30	6	14 *	12	15 *
15	31 *	7	11	13	15 *
16	16 *	24	20	18	17 *
17	1	25	21 *	19	17 *
18	18 *	18	**22**	18	19 *
19	**3**	19	23 *	19	19 *
20	20	**28**	20	**22**	20
21	21 *	29	21 *	23	20
22	**22**	**22**	**22**	**22**	**22**
23	23 *	23	23 *	23	**22**
24	24 *	24	**28**	24	25 *
25	**9**	25	29 *	25	25 *
26	26 *	18	30	24	27 *
27	11	19	31 *	25	27 *
28	**28**	**28**	**28**	**28**	**28**
29	29 *	29	29 *	29	**28**
30	30	**22**	30	**28**	30
31	31 *	23	31 *	29	30

Tabelle 3.1: *Netzwerk aus 5 Neuronen. Gewichtsmatrix (3.2). Von jedem möglichen Zustand aus (Anfangszustand $\mathbf{x_{anf}}$) wird jeweils 1 Neuron nach Gleichung (2.5) abgefragt. Der jeweils daraus entstehende Netzzustand ist in den Spalten 2 bis 6 angegeben. Grau unterlegt: Keine Änderung des Netzzustandes. Fett: Stabile Zustände. $*$: Bei einer betreffenden Abfrage war das innere Argument $y_i = 0$*

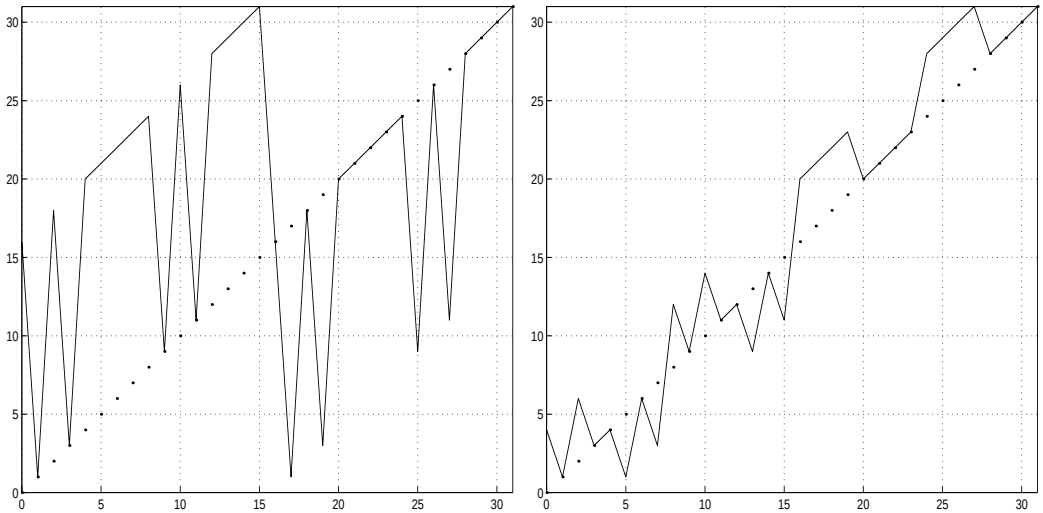


Abbildung 3.1: *A: Die in Spalte 2 (Neuron 1) von Tab. 3.1 stehenden Netzzustände. Abszisse: Anfangszustand* $\mathbf{x_{anf}}$*. Ordinate: den nach Gleichung (2.5) berechneten Zustand, wenn nur Neuron 1 abgefragt wurde. Die jeweiligen Anfangszustände liegen auf der Winkelhalbierenden. B: Wie A; aber Neuron 3 abgefragt (Spalte 4 von Tab. 3.1). Man beachte die Unsymmetrie von 1. und 2. Hälfte der Kurven*

entspricht (Zustand 23 und 29). Diese Zustände liegen offensichtlich im Attraktionsbecken der jeweiligen stabilen Zustände.

5. Solche Attraktionsbecken liegen für die jeweiligen Antimuster nicht vor. In dieser Hinsicht ist also keine Symmetrie vorhanden.

6. Abfragen, in denen das innere Argument nach Gleichung (2.6) $y_i = 0$ ist (mit $*$ bezeichnet), sind ebenfalls symmetrisch verteilt. Trotzdem sind die aus solchen Abfragen entstehenden Zustände nicht symmetrisch angeordnet. So sind z. B. die Anfangszustände 0 und 31 symmetrisch, weil sie sich nur im Vorzeichen unterscheiden. Sie haben deshalb auch den gleichen Energieinhalt. Aus ihnen entstehen aber Zustände, die nicht symmetrisch zueinander sind. Diese Unsymmetrie entsteht durch die Verwendung der Vorzeichenfunktion (2.7). Siehe dazu auch Abb. 3.1.

Im nächsten Analyseschritt werden **Abfrageblöcke** verwendet, und zwar die sequentielle Reihenfolge nach Abschnitt 2.4.3 und alle möglichen Permuta-

Anfangszustand $\mathbf{x_{anf}}$	Zustand nach sequentiller Abfragereihenfolge	Zustand nach sequentiell rücklaufender Abfragereihenfolge	erreichbare Zustände nach Betrachtung aller Permutationen der sequentiellen Reihenfolge	erreichbare stabile Zustände von Anfangszustand $\mathbf{x_{anf}}$	Hammingabstand zum stabilen Zustand 3,9,22,28	absoluter Energieinhalt des Anfangszustands	Anfangszustand $\mathbf{x_{anf}}$ gleich $\xi^{(?)}$?
0	**28**	**3**	**3**,**9**,**22**,23,**28**,29	3,9,22,28	2,2,3,3	4	
1	**9**	**3**	**3**,**9**	3,9	1,1,4,4	-4	
2	**22**	**3**	**3**,**22**,23	3,22	1,3,2,4	0	
3	**3**	**3**	**3**	3	0,2,3,5	-8	$-\xi^{(2)}$
4	**28**	**3**	**3**,**9**,**22**,23,**28**,29	3,9,22,28	3,3,2,2	4	
5	**28**	**3**	**3**,**9**,**22**,23,**28**,29	3,9,22,28	2,2,3,3	4	
6	**22**	**3**	**3**,**22**,23	3,22	2,4,1,3	0	
7	**22**	**3**	**3**,**22**,23	3,22	1,3,2,4	0	
8	**28**	**9**	**9**,**28**,29	9,28	3,1,4,2	0	
9	**9**	**9**	**9**	9	2,0,5,3	-8	$-\xi^{(1)}$
10	**22**	**9**	**3**,**9**,**22**,23,**28**,29	3,9,22,28	2,2,3,3	4	
11	**3**	**9**	**3**,**9**	3,9	1,1,4,4	-4	
12	**28**	**9**	**9**,**28**,29	9,28	4,2,3,1	0	
13	**28**	**9**	**9**,**28**,29	9,28	3,1,4,2	0	
14	**22**	**9**	**3**,**9**,**22**,23,**28**,29	3,9,22,28	3,3,2,2	4	
15	**22**	**9**	**3**,**9**,**22**,23,**28**,29	3,9,22,28	2,2,3,3	4	
16	**28**	23	**3**,**9**,**22**,23,**28**,29	3,9,22,28	3,3,2,2	4	
17	**9**	**23**	**3**,**9**,**22**,23,**28**,29	3,9,22,28	2,2,3,3	4	
18	**22**	23	**3**,**22**,23	3,22	2,4,1,3	0	
19	**3**	23	**3**,**22**,23	3,22	1,3,2,4	0	
20	**28**	**22**	**22**,**28**	22,28	4,4,1,1	-4	
21	**28**	**22**	**22**,**28**	22,28	3,3,2,2	4	
22	**22**	**22**	**22**	22	3,5,0,2	-8	$\xi^{(1)}$
23	**22**	**22**	**22**	22	2,4,1,3	0	
24	**28**	29	**9**,**28**,29	9,28	4,2,3,1	0	
25	**9**	**29**	**9**,**28**,29	9,28	3,1,4,2	0	
26	**22**	29	**3**,**9**,**22**,23,**28**,29	3,9,22,28	3,3,2,2	4	
27	**3**	29	**3**,**9**,**22**,23,**28**,29	3,9,22,28	2,2,3,3	4	
28	**28**	**28**	**28**	28	5,3,2,0	-8	$\xi^{(2)}$
29	**28**	**28**	**28**	28	4,2,3,1	0	
30	**22**	**28**	**22**, **28**	22,28	4,4,1,1	-4	
31	**22**	**28**	**22**,**28**	22,28	3,3,2,2	4	

Tabelle 3.2: HOPFIELD-*Netz aus 5 Neuronen, Gewichtsmatrix (3.2). Netzzustände, die durch Abfrage von allen möglichen Anfangszuständen (Spalte 1), asynchron sequentiell (Spalte 2), asynchron sequentiell rücklaufend (Spalte 3) und mit allen möglichen Permutationen der sequentiellen Reihenfolge (Spalte 4) entstehen. Stabile Zustände fett. Spalte 5: Alle erreichbaren stabilen Zustände (evtl. erst durch weitere Abfrageschritte). Hammingabstände der Anfangszustände zum jeweiligen stabilen Zustand (Spalte 6). Energieinhalt des Anfangzustands (Spalte 7)*

tionen davon nach Abschnitt 2.4.4 ($N! = 120$). Die Ergebnisse sind in Tab. 3.2 dargestellt. Die Abbildungen 3.2 und 3.3 zeigen in Histogrammen die Anzahlen der einzelnen Zustände, die von jedem Anfangszustand aus erreicht werden, wenn alle 120 verschiedenen Abfrageblöcke je einmal verwendet werden.
Aus Tab. 3.2 und den Abb. 3.2 und 3.3 erkennt man:

1. Es gibt Zustände, von denen aus 2 oder gar alle 4 stabilen Zustände erreicht werden können.

2. Von zwei Zuständen (23 und 29) wird auch jetzt nur 1 stabiler Zustand erreicht, d.h. diese Zustände liegen im Attraktionsbecken des jeweiligen stabilen Zustands.

3. Aus 2 ergibt sich, daß es stabile Zustände mit (stabile Zustände 22 und 28) und solche ohne Attraktionsbecken (stabile Zustände 3 und 9) gibt. Daraus ergibt sich

Aussage 3.1 *Es gibt 3 verschiedene Arten von Zuständen:*

1. *Der Zustand ist ein stabiler Zustand (hier: 12,5% der Zustände).*

2. *Der Zustand liegt im Attraktionsbecken eines stabilen Zustands. Unabhängig von der Abfragereihenfolge kann nur dieser Zustand erreicht werden (hier 6,25% der Zustände).*

3. *Es gibt Zustände, die je nach Abfragereihenfolge unterschiedliche stabile Zustände erreichen (hier: 81,25% der Zustände). Ich bezeichne sie im folgenden als nicht- zuordenbare Zustände.*

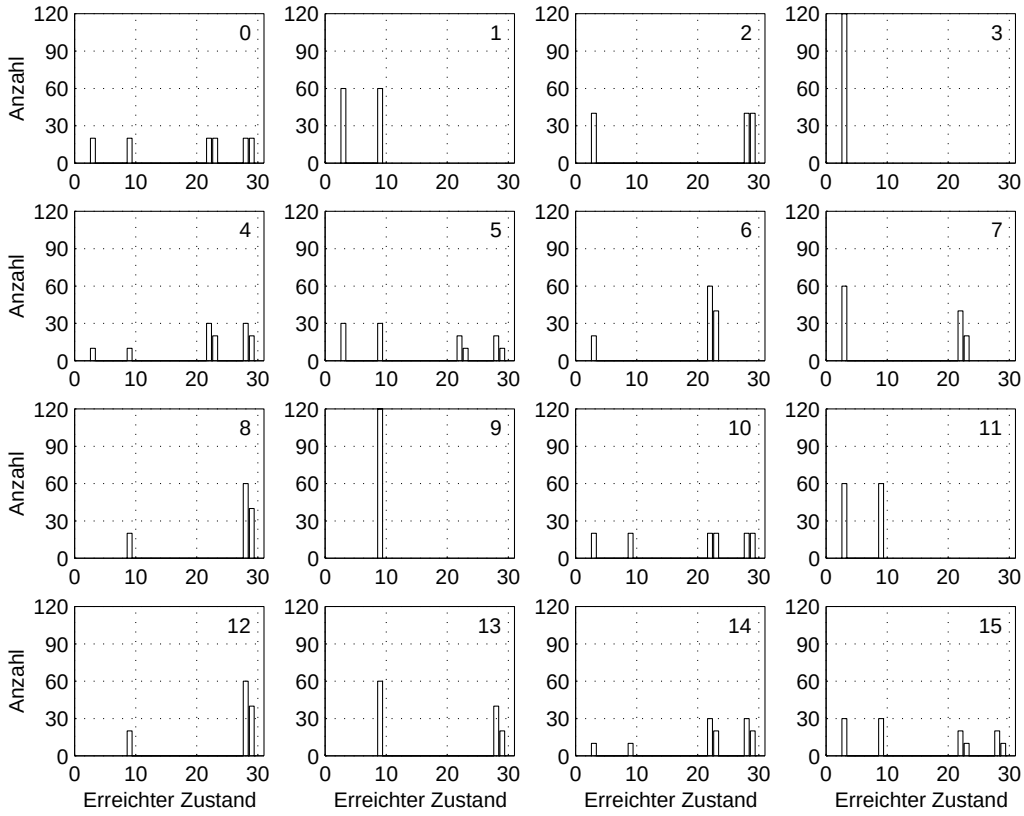


Abbildung 3.2: *Anzahl der Zustände (siehe 4. Spalte von Tab. 3.2), die von den ersten 16 Anfangszuständen bei den 120 Permutationen der sequentiell asynchronen Abfragereihenfolge erreicht wurden. Der jeweilige Anfangszustand ist in der rechten oberen Ecke angegeben*

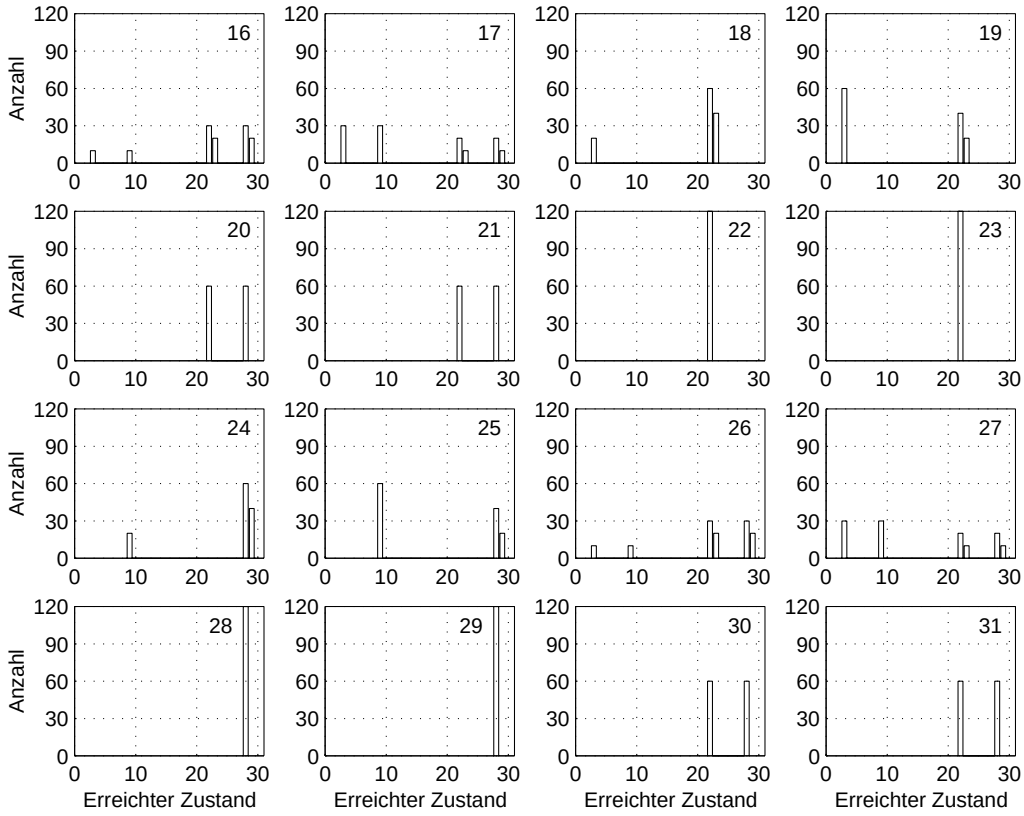


Abbildung 3.3: *Anzahl der Zustände (siehe 4. Spalte von Tab. 3.2), die von den letzten 16 Anfangszuständen bei den 120 Permutationen der sequentiell asynchronen Abfragereihenfolge erreicht wurden. Der jeweilige Anfangszustand ist in der rechten oberen Ecke angegeben*

3.1.2 Bestätigung der gefundenen Eigenschaften an weiteren Beispielen

Um sicher zu sein, daß das zuerst untersuchte Netz kein Ausnahmefall war, wurden weitere Netze aus 5 oder 6 Neuronen auf die gleiche Weise wie in Abschnitt 3.1.1 untersucht. In einem Beispiel wurde nur ein einziges Muster eingeprägt, in den drei anderen Beispielen wurden die Netze stark überladen.

1. Beispiel. Folgendes Muster soll in einem Netz mit $N = 5$ Neuronen gespeichert werden:

$$\xi = \begin{pmatrix} +1 & +1 & -1 & +1 & +1 \end{pmatrix}. \tag{3.4}$$

Nach Hebb in Gleichung (2.24) ergibt sich folgende Gewichtsmatrix

$$\mathbf{w} = \begin{pmatrix} 0 & +1 & -1 & +1 & +1 \\ +1 & 0 & -1 & +1 & +1 \\ -1 & -1 & 0 & -1 & -1 \\ +1 & +1 & -1 & 0 & +1 \\ +1 & +1 & -1 & +1 & 0 \end{pmatrix}. \tag{3.5}$$

Die Ergebnisse sind in Tab. 3.3 zusammengefaßt. Von den 32 möglichen Zuständen waren 2 (6,25%) stabile Zustände, 14 (43,75%) Zustände in Attraktionsbecken und 16 (50%) nicht- zuordenbare Zustände.
Die Ergebnisse bestätigen auch, daß Gleichung (2.31) für die Energiefunktion des fundamentalen Musters erfüllt ist.

Anfangszustand $\mathbf{x_{anf}}$	Zustand nach sequentieller Abfragereihenfolge	Zustand nach sequentiell rücklaufender Abfragereihenfolge	erreichbare Zustände nach Betrachtung aller Permutationen der sequentiellen Reihenfolge	erreichbare stabile Zustände von Anfangszustand $\mathbf{x_{anf}}$	Absoluter Energieinhalt des Anfangszustands	Anfangszustand $\mathbf{x_{anf}}$ gleich ξ?
0	**4**	**4**	**4**	4	-2	
1	**27**	**4**	**4**, 6, 12, 20, **27**, 31	4, 27	2	
2	**27**	31	**4**, 5, 12, 20, **27**, 31	4, 27	2	
3	**27**	31	**4**, 5, 6, **27**, 31	4, 27	2	
4	**4**	**4**	**4**	4	-10	-ξ
5	**4**	**4**	**4**	4	-2	
6	**4**	**4**	**4**	4	-2	
7	**27**	**4**	**4**, **27**, 31	4, 27	2	
8	31	**27**	**4**, 5, 6, 20, **27**, 31	4, 27	2	
9	**27**	**27**	**4**, 5, 12, **27**, 31	4, 27	2	
10	**27**	**27**	**4**, 6, 12, **27**	4, 27	2	
11	**27**	**27**	**27**	27	-2	
12	**4**	**4**	**4**	4	-2	
13	**27**	**4**	**4**, **27**, 31	4, 27	2	
14	**27**	**27**	**4**, **27**, 31	4, 27	2	
15	**27**	**27**	**27**	27	2	
16	**4**	**27**	**4**, 5, 6, 12,**27**, 31	4, 27	2	
17	**27**	**27**	**4**, 5, 20, **27**, 31	4, 27	2	
18	**27**	**27**	**4**, 6, 20, **27**, 31	4, 27	2	
19	**27**	**27**	**27**	27	-2	
20	**4**	**4**	**4**	4	-2	
21	**4**	**4**	**4**, **27**, 31	4, 27	2	
22	**4**	**27**	**4**, **27**, 31	4, 27	2	
23	**27**	**27**	**27**	27	2	
24	31	**27**	**4**, 12, 20, **27**, 31	4, 27	2	
25	**27**	**27**	**27**	27	-2	
26	**27**	**27**	**27**	27	-2	
27	**27**	**27**	**27**	27	-10	ξ
28	**4**	**27**	**4**, **27**, 31	4, 27	2	
29	**27**	**27**	**27**	27	2	
30	**27**	**27**	**27**	27	2	
31	**27**	**27**	**27**	27	-2	

Tabelle 3.3: HOPFIELD-*Netz aus 5 Neuronen, Gewichtsmatrix (3.5). Netzzustände, die durch Abfrage von allen möglichen Anfangszuständen (Spalte 1), asynchron sequentiell (Spalte 2), asynchron sequentiell rücklaufend (Spalte 3) und mit allen möglichen Permutationen der sequentiellen Reihenfolge (Spalte 4) entstehen. Stabile Zustände fett. Spalte 5: Alle erreichbaren stabilen Zustände (evtl. erst durch weitere Abfrageschritte). Energieinhalt des Anfangszustands (Spalte 6)*

2. Beispiel. Es wurde ein Netz aus 5 Neuronen verwendet. Ferner wurde versucht, folgende Muster einzuspeichern.

$$\begin{aligned} \xi^{(\mathbf{1})} &= \begin{pmatrix} -1 & +1 & +1 & -1 & -1 \end{pmatrix}, \\ \xi^{(\mathbf{2})} &= \begin{pmatrix} -1 & +1 & +1 & +1 & -1 \end{pmatrix}, \\ \xi^{(\mathbf{3})} &= \begin{pmatrix} +1 & -1 & -1 & +1 & -1 \end{pmatrix}. \end{aligned} \tag{3.6}$$

Nach HEBB ergibt sich folgende Gewichtsmatrix:

$$\mathbf{w} = \begin{pmatrix} 0 & -3 & -3 & 1 & 1 \\ -3 & 0 & 3 & -1 & -1 \\ -3 & 3 & 0 & -1 & -1 \\ 1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 \end{pmatrix}. \tag{3.7}$$

Die Ergebnisse sind in Tab. 3.4 dargestellt. Es treten 2 (6,25%) stabile Zutände auf, die aber nur einem der drei Muster entsprechen, sowie 12 (37,5%) in Attraktionsbecken und 18 (56,25%) nicht- zuordenbare Zustände. Von Zuständen, die in einem Attraktionsbecken liegen, kann der stabile Zustand je nach Abfragereihenfolge auf unterschiedlichen Trajektorien erreicht werden (Zustände 28, 29, 30).
Die Muster, die nicht stabile Zustände ergaben, liegen in den Attraktionsbecken von $\xi^{(\mathbf{1})}$ oder $-\xi^{(\mathbf{1})}$.

Anfangszustand $\mathbf{x_{anf}}$	Zustand nach sequentieller Abfragereihenfolge	Zustand nach sequentiell rücklaufender Abfragereihenfolge	erreichbare Zustände nach Betrachtung aller Permutationen der sequentiellen Reihenfolge	erreichbare stabile Zustände von Anfangszustand $\mathbf{x_{anf}}$	Absoluter Energieinhalt des Anfangszustands	Anfangszustand $\mathbf{x_{anf}}$ gleich $\xi^{(?)}$?
0	**19**	**19**	**12**, 13, 14, **19**, 23, 27, 29	12, 19	6	
1	**19**	**19**	**12**, 13, **19**, 23, 27, 29	12, 19	2	
2	**19**	**19**	**12**, 14, **19**, 23, 27, 30	12, 19	2	
3	**19**	**19**	**19**	19	2	
4	**12**	13	**12**, 13, 14, 17, 18, **19**, 23, 29, 30	12, 19	2	
5	29	13	**12**, 13, 17, **19**, 23, 29	12, 19	2	
6	30	14	**12**, 14, 18, **19**, 23, 30	12, 19	2	
7	**19**	14	**12**, 13, 14, 17, 18, **19**, 29, 30	12, 19	6	
8	**12**	13	**12**, 13, 14, 17, 18, **19**, 27, 29, 30	12, 19	2	
9	**19**	13	**12**, 13, 17, **19**, 27, 29	12, 19	2	
10	**19**	14	**12**, 14, 18, **19**, 27, 30	12, 19	2	
11	**19**	14	**12**, 13, 14, 17, 18, **19**, 29, 30	12, 19	6	
12	**12**	**12**	**12**	12	-14	$\xi^{(\mathbf{1})}$
13	**12**	**12**	**12**	12	-10	$-\xi^{(\mathbf{3})}$
14	**12**	**12**	**12**	12	-10	$\xi^{(\mathbf{2})}$
15	**12**	**12**	**12**	12	-2	
16	**19**	**19**	**19**	19	-2	
17	**19**	**19**	**19**	19	-10	$-\xi^{(\mathbf{2})}$
18	**19**	**19**	**19**	19	-10	$\xi^{(\mathbf{3})}$
19	**19**	**19**	**19**	19	-14	$-\xi^{(\mathbf{1})}$
20	**12**	**19**	**12**, 13, 14, **19**, 29, 30	12, 19	6	
21	29	**19**	**12**, 13, **19**, 29	12, 19	2	
22	30	**19**	**12**, 14, **19**, 30	12, 19	2	
23	**19**	**19**	**19**	19	2	
24	**12**	**19**	**12**, 13, 14, **19**, 29, 30	12, 19	6	
25	**19**	**19**	**12**, 13, **19**, 29	12, 19	2	
26	**19**	**19**	**12**, 14, **19**, 30	12, 19	2	
27	**19**	**19**	**19**	19	2	
28	**12**	13	**12**, 13, 14	12	2	
29	**12**	13	**12**, 13	12	2	
30	**12**	14	**12**, 14	12	2	
31	**12**	14	**12**, 13, 14, **19**	12, 19	6	

Tabelle 3.4: HOPFIELD-*Netz aus 5 Neuronen, Gewichtsmatrix (3.7). Netzzustände, die durch Abfrage von allen möglichen Anfangszuständen (Spalte 1), asynchron sequentiell (Spalte 2), asynchron sequentiell rücklaufend (Spalte 3) und mit allen möglichen Permutationen der sequentiellen Reihenfolge (Spalte 4) entstehen. Stabile Zustände fett. Spalte 5: Alle erreichbaren stabilen Zustände (evtl. erst durch weitere Abfrageschritte). Energieinhalt des Anfangszustands (Spalte 6)*

3. Beispiel. Nimmt man die fundamentalen Muster in einem Netz aus 5 Neuronen

$$\begin{aligned}
\xi^{(\mathbf{1})} &= \begin{pmatrix} +1 & -1 & +1 & +1 & +1 \end{pmatrix}, \\
\xi^{(\mathbf{2})} &= \begin{pmatrix} +1 & +1 & +1 & +1 & +1 \end{pmatrix}, \\
\xi^{(\mathbf{3})} &= \begin{pmatrix} +1 & +1 & -1 & +1 & -1 \end{pmatrix}, \\
\xi^{(\mathbf{4})} &= \begin{pmatrix} +1 & -1 & +1 & +1 & -1 \end{pmatrix}, \\
\xi^{(\mathbf{5})} &= \begin{pmatrix} +1 & -1 & +1 & -1 & +1 \end{pmatrix},
\end{aligned} \tag{3.8}$$

so ergibt sich nach Gleichung (2.24) folgende Gewichtsmatrix

$$\mathbf{w} = \begin{pmatrix} 0 & -1 & 3 & 3 & 1 \\ -1 & 0 & -3 & 1 & -1 \\ 3 & -3 & 0 & 1 & 3 \\ 3 & 1 & 1 & 0 & -1 \\ 1 & -1 & 3 & -1 & 0 \end{pmatrix}. \tag{3.9}$$

Die Ergebnisse sind in Tab. 3.5 dargestellt. Auch hier treten 2 (6,25%) stabile Zustände auf ($\xi^{(\mathbf{1})}$ und $-\xi^{(\mathbf{1})}$; die anderen 4 Muster liegen in deren Attraktionsbecken), 12 (37,5%) liegen in Attraktionsbecken (erreichen die stabilen Zustände aber teilweise auf unterschiedlichen Trajektorien, Zustände 5, 7, 24, 26) und 18 Zustände (56,25%) sind nicht- zuordenbar.

Anfangszustand $\mathbf{x_{anf}}$	Zustand nach sequentieller Abfragereihenfolge	Zustand nach sequentiell rücklaufender Abfragereihenfolge	erreichbare Zustände nach Betrachtung aller Permutationen der sequentiellen Reihenfolge	erreichbare stabile Zustände von Anfangszustand $\mathbf{x_{anf}}$	Absoluter Energieinhalt des Anfangszustands	Anfangszustand $\mathbf{x_{anf}}$ gleich $\xi^{(?)}$?
0	**8**	**8**	**8**	8	-6	$-\xi^{(\mathbf{2})}$
1	**8**	**8**	5, **8**, 21, **23**	8, 23	-2	
2	26	**8**	**8**, 22, **23**, 26	8, 23	2	
3	31	**8**	5, **8**, 21, 22, **23**, 26, 31	8, 23	10	
4	**23**	21	0, **8**, 21, **23**	8, 23	2	
5	**23**	21	21, **23**	23	-6	$-\xi^{(\mathbf{3})}$
6	**23**	21	0, **8**, 9, 10, 21, **23**, 26, 27, 29, 31	8, 23	6	
7	**23**	21	21, **23**	23	2	
8	**8**	**8**	**8**	8	-14	$-\xi^{(\mathbf{1})}$
9	**8**	**8**	**8**	8	-6	$-\xi^{(\mathbf{4})}$
10	**8**	**8**	**8**	8	-10	$-\xi^{(\mathbf{5})}$
11	31	**8**	**8**, **23**, 26, 31	8, 23	2	
12	0	9	0, 5, 7, **8**, 9, 10, 11, 21, **23**, 26, 27, 31	8, 23	6	
13	**23**	9	**8**, 9, 21, **23**	8, 23	2	
14	**23**	9	**8**, 9, 10, 21, **23**, 26, 27, 29, 31	8, 23	6	
15	**23**	9	**8**, 9, 21, **23**, 26, 27	8, 23	6	
16	**8**	**23**	5, **8**, 9, 10, 22, **23**, 26, 31	8, 23	6	
17	**8**	17	5, **8**, 9, 10, 22, **23**, 26, 31	8, 23	6	
18	26	22	**8**, 10, 22, **23**, 26	8, 23	2	
19	31	22	**8**, 10, 22, **23**, 26, 31	8, 23	6	
20	**23**	**23**	**23**	23	2	
21	**23**	**23**	**23**	23	-10	$\xi^{(\mathbf{5})}$
22	**23**	**23**	**23**	23	-6	$\xi^{(\mathbf{4})}$
23	**23**	**23**	**23**	23	-14	$\xi^{(\mathbf{1})}$
24	**8**	10	**8**, 10	8	2	
25	**8**	10	**8**, 10, **23**, 26, 31	8, 23	6	
26	**8**	10	**8**, 10	8	-6	$\xi^{(\mathbf{3})}$
27	31	10	**8**, 10, **23**, 26, 31	8, 23	2	
28	0	**23**	0, 5, 7, **8**, 9, 10, 11, **23**, 26	8, 23	10	
29	**23**	**23**	**23**	23	2	
30	**23**	**23**	**8**, 10, **23**, 26	8, 23	-2	
31	**23**	**23**	**23**	23	-6	$\xi^{(\mathbf{2})}$

Tabelle 3.5: HOPFIELD-*Netz aus 5 Neuronen, Gewichtsmatrix (3.9). Netzzustände, die durch Abfrage von allen möglichen Anfangszuständen (Spalte 1), asynchron sequentiell (Spalte 2), asynchron sequentiell rücklaufend (Spalte 3) und mit allen möglichen Permutationen der sequentiellen Reihenfolge (Spalte 4) entstehen. Stabile Zustände fett. Spalte 5: Alle erreichbaren stabilen Zustände (evtl. erst durch weitere Abfrageschritte). Energieinhalt des Anfangszustands (Spalte 6)*

4. Beispiel. Ein anderes Beispiel besteht aus einem Netzwerk mit $N = 6$ Neuronen, in dem die Muster mit $p = 4$ ebenfalls viel zu viele sind, was aber nach Abschnitt 2.6.1 sowieso für die meisten kleinen Netze der Fall ist. Diese Muster wurden zufällig gewählt:

$$\begin{aligned}
\xi^{(\mathbf{1})} &= \begin{pmatrix} -1 & +1 & -1 & -1 & +1 & -1 \end{pmatrix}, \\
\xi^{(\mathbf{2})} &= \begin{pmatrix} -1 & +1 & +1 & -1 & -1 & +1 \end{pmatrix}, \\
\xi^{(\mathbf{3})} &= \begin{pmatrix} -1 & -1 & -1 & +1 & +1 & +1 \end{pmatrix}, \\
\xi^{(\mathbf{4})} &= \begin{pmatrix} +1 & -1 & +1 & -1 & +1 & -1 \end{pmatrix}.
\end{aligned} \tag{3.10}$$

Aus diesen Mustern erhält man nach Gleichung (2.24) die Gewichtsmatrix

$$\mathbf{w} = \begin{pmatrix} 0 & -2 & +2 & 0 & 0 & -2 \\ -2 & 0 & 0 & -2 & -2 & 0 \\ +2 & 0 & 0 & -2 & -2 & 0 \\ 0 & -2 & -2 & 0 & 0 & +2 \\ 0 & -2 & -2 & 0 & 0 & -2 \\ -2 & 0 & 0 & +2 & -2 & 0 \end{pmatrix}. \tag{3.11}$$

Die Ergebnisse sind in Tab. 3.6 zusammengefaßt. Da dieses Netz für die restlichen Zustände eine vollständige Symmetrie zu den dargestellten Zuständen aufweist, ist nur eine Hälfte der Zustände angegeben. Es treten insgesamt 8 (12,5%) stabile Zustände auf (alle eingespeicherten Muster und deren Antimuster) aber keine Zustände in Attraktionsbecken. 87,5% der Zustände sind also nicht- zuordenbar. Von vielen dieser Zustände können alle stabilen Zustände erreicht werden.
Wird dieses Netz synchron abgefragt, treten bei allen nicht- zuordenbaren Zuständen Zyklen auf, d.h. die Muster werden nicht erreicht.

Anfangszustand $\mathbf{x_{anf}}$	Zustand nach sequentieller Abfragereihenfolge	Zustand nach sequentiell rücklaufender Abfragereihenfolge	erreichbare Zustände nach Betrachtung aller Permutationen der sequentiellen Reihenfolge	erreichbare stabile Zustände von Anfangszustand $\mathbf{x_{anf}}$	Absoluter Energieinhalt des Anfangszustands	Anfangszustand $\mathbf{x_{anf}}$ gleich $\xi^{(?)}$?
0	**56**	**7**	**7**, **18**, **21**, 23, **25**, 29, **38**, 39, **42**, **45**, 46, **56**, 57, 58	7, 18, 21, 25, 38, 42, 45, 56	10	
1	**25**	**7**	6, **7**, **18**, **21**, 23, **25**, 29, **38**, **42**, **45**, **56**, 57	7, 18, 21, 25, 38, 42, 45, 56	6	
2	**42**	**38**	**7**, **18**, **38**, **42**	7, 18, 38, 42	-2	
3	**21**	**38**	6, **7**, **18**, **21**, 23, **25**, **38**, **42**	7, 18, 21, 25, 38, 42	2	
4	**42**	**7**	**7**, 17, **18**, **21**, 23, **25**, **38**,39, **42**, **45**, 46, **56**	7, 18, 21, 25, 38, 42, 45, 56	6	
5	**21**	**7**	**7**, **21**	7, 21	-6	
6	**38**	**7**	**7**, **38**	7, 38	-6	
7	**7**	**7**	**7**	7	-10	$\xi^{(3)}$
8	**56**	**21**	**7**, **18**, **21**, **25**, 29, 34, **38**, **42**, **45**, **56**, 57	7, 18, 21, 25, 38, 42, 45, 56	6	
9	**56**	**21**	**7**, **21**, **25**, 29, **42**, **45**, **56**, 57	7, 21, 25, 42, 45, 56	2	
10	**42**	**18**	**7**, **18**, **25**, 34, **38**, **42**, **56**, 58	7, 18, 25, 38, 42, 56	2	
11	**45**	**18**	5, 6, **7**, 17, **18**, **21**, 23, 24, **25**, 29, 34, **38**, 39, 40, **42**, **45**, 46, **56**, 57, 58	7, 18, 21, 25, 38, 42, 45, 56	6	

Fortsetzung der Tabelle auf der nächsten Seite

Fortsetzung der Tabelle der vorigen Seite						
12	**42**	**21**	**7**, 17, **18**, **21**, 23, **25**, 34, **38**, 39, **42**, **45**, **56**, 57, 58	7, 18, 21, 25, 38, 42, 45, 56	10	
13	**45**	**21**	**7**, **21**, **25**, **45**	7, 21, 25, 45	-2	
14	**38**	**21**	5, **7**, **18**, **21**, **25**, 34, **38**, 39, **42**, **45**, **56**, 58	7, 18, 21, 25, 38, 42, 45, 56	6	
15	**38**	**21**	5, **7**, **21**, **25**, **38**, 39, **42**, **45**	7, 21, 25, 38, 42, 45	2	
16	**25**	**21**	**18**, **21**, **25**, **56**	18, 21, 25, 56	-2	
17	**25**	**21**	**21**, **25**	21, 25	-6	
18	**18**	**18**	**18**	18	-6	$\xi^{(1)}$
19	**21**	**18**	**7**, **18**, **21**, **25**	7, 18, 21, 25	-2	
20	**18**	**21**	**7**, 17, **18**, **21**, 23, **25**, **38**, **56**	7, 18, 21, 25, 38, 56	2	
21	**21**	**21**	**21**	21	-10	$-\xi^{(4)}$
22	**7**	**21**	**7**, **18**, **21**, **38**	7, 18, 21, 38	-2	
23	**7**	**21**	**7**, **21**	7, 21	-6	
24	**56**	**25**	**25**, **56**	25, 56	-6	
25	**25**	**25**	**25**	25	-10	$\xi^{(2)}$
26	**42**	**56**	**18**, **25**, **42**, **56**	18, 25, 42, 56	-2	
27	**21**	**56**	**7**, 17, **18**, **21**, 24, **25**, **42**, **56**	7, 18, 21, 25, 42, 56	2	
28	**42**	**25**	**7**, 17, **18**, **21**, 23, **25**, **38**, 40, **42**, **45**, **56**, 58	7, 18, 21, 25, 38, 42, 45, 56	6	
29	**21**	**25**	**21**, **25**	21, 25	-6	
30	**38**	**25**	5, **7**, 17, **18**, **21**, **25**, 34, **38**, 39, 40, **42**, **45**, **56**, 57	7, 18, 21, 25, 38, 42, 45, 56	10	
31	**7**	**25**	5, **7**, 17, **18**, **21**, 24, **25**, **38**, 39, **42**, **45**, **56**	7, 18, 21, 25, 38, 42, 45, 56	6	

Tabelle 3.6: HOPFIELD-*Netz aus 6 Neuronen, Gewichtsmatrix (3.11). Netzzustände, die durch Abfrage von allen möglichen Anfangszuständen (Spalte 1), asynchron sequentiell (Spalte 2), asynchron sequentiell rücklaufend (Spalte 3) und mit allen möglichen Permutationen der sequentiellen Reihenfolge (Spalte 4) entstehen. Stabile Zustände fett. Spalte 5: Alle erreichbaren stabilen Zustände. Energieinhalt des Anfangzustands (Spalte 6). Es ist nur die 1. Hälfte der Anfangszustände dargestellt, die 2. Hälfte ist symmetrisch dazu*

Die Untersuchungen bestätigen die in Abschnitt 3.1.1 gemachten Aussagen und ergänzen sie:

1. *In allen Fällen treten neben stabilen Zuständen mit oder ohne Attraktionsbecken nicht- zuordenbare Zustände auf. In den gewählten Beispielen steigt ihr prozentualer Anteil mit steigender Zahl der stabilen Zustände.*

2. *Die Größe einzelner Attraktionsbecken nimmt mit steigender Zahl der stabilen Zustände ab und kann auf Null sinken.*

3. *Zustände innerhalb von Attraktionsbecken können zum stabilen Zustand über unterschiedliche Trajektorien konvergieren, je nach Abfragereihenfolge.*

4. *Die verwendeten Muster ergaben nicht nur dann stabile Zustände, wenn sie den geringsten Enerieinhalt hatten (z. B. $\xi^{(1)}$ in Beispiel 4).*

3.2 Fehlerkorrigierendes Verhalten an Netzwerken aus 7, 8 oder 10 Neuronen

Die in Abschnitt 3.1 gefundenen stabilen Zustände waren symmetrisch zueinander. Das heißt, daß zu jedem gefundenem stabilen Zustand der Antizustand auch ein stabiler Zustand war. Dies kommt daher, daß ein fundamentales Muster ξ und sein fundamentales Antimuster $-\xi$ je dieselbe Gewichtsmatrix nach Gleichung (2.21) ergeben. Wie in Abschnitt 2.6.1 beschrieben, speichern die Gewichtsfaktoren die Korrelationen der Zustände zwischen zwei Neuronen und nicht die Zustände selbst. Diese Korrelationen können durch Eingabe weiterer fundamentaler Muster gestört sein. Doch dies beeinflußt die Symmetrie zwischen Muster und Antimuster nicht (Abschnitt 3.1). Nur

in der Dynamik durch die Vorzeichenfunktion trat ein unsymmetrisches Verhalten auf (vergleiche Abb. 3.1).
Im folgenden werden weitere Netzwerke nach dem gleichen Verfahren, wie in Abschnitt 3.1 vorgestellt, untersucht.

Gegeben sei ein **Netz aus N = 7 Neuronen**. In ihm sollen folgende Muster, die zufällig gewählt wurden, gespeichert werden.

$$\begin{aligned}
\xi^{(1)} &= \begin{pmatrix} -1 & +1 & -1 & -1 & +1 & +1 & +1 \end{pmatrix}, \\
\xi^{(2)} &= \begin{pmatrix} -1 & -1 & -1 & -1 & +1 & -1 & +1 \end{pmatrix}, \\
\xi^{(3)} &= \begin{pmatrix} +1 & +1 & +1 & -1 & +1 & +1 & +1 \end{pmatrix}, \\
\xi^{(4)} &= \begin{pmatrix} +1 & +1 & +1 & -1 & +1 & -1 & -1 \end{pmatrix}. \qquad (3.12)
\end{aligned}$$

Nach Gleichung (2.24) ergibt sich folgende Gewichtsmatrix

$$\mathbf{w} = \begin{pmatrix}
0 & +2 & +4 & 0 & 0 & 0 & -2 \\
+2 & 0 & +2 & -2 & +2 & +2 & 0 \\
+4 & +2 & 0 & 0 & 0 & 0 & -2 \\
0 & -2 & 0 & 0 & -4 & 0 & -2 \\
0 & +2 & 0 & -4 & 0 & 0 & +2 \\
0 & +2 & 0 & 0 & 0 & 0 & +2 \\
-2 & 0 & -2 & -2 & +2 & +2 & 0
\end{pmatrix}. \qquad (3.13)$$

Tab. 3.7 zeigt alle Zustände, die entweder stabile Zustände sind oder in einem Attraktionsbecken liegen. An Tab. 3.7 erkennt man, daß alle Antimuster als stabile Zustände gespeichert wurden. Dagegen liegen das fundamentale Muster $\xi^{(2)}$ im Attraktionsbereich von $\xi^{(1)}$ und das fundamentale Muster $\xi^{(4)}$ im Attraktionsbereich von $\xi^{(3)}$. Die Muster $\xi^{(2)}$ und $\xi^{(4)}$ sind also nicht gespeichert. Man beachte, daß die zugehörigen Antimuster $-\xi^{(2)}$ bzw. $-\xi^{(4)}$ zwar stabile Zustände sind, aber kein Attraktionsbecken haben. *Muster und Antimuster müssen also nicht gleichzeitig stabile Zustände ergeben.* Wieder können die Attraktionsbecken von Muster und Antimuster unterschiedlich groß sein (vergleiche $\xi^{(1)}$ und $-\xi^{(1)}$ sowie $\xi^{(4)}$ und $-\xi^{(4)}$). Mehrfach werden auch die stabilen Zustände auf unterschiedlichen Trajektorien erreicht.
Ingesamt waren in diesem Netz 6 stabile Zustände (4,7%), 5 Zustände in Attraktionsbecken eines stabilen Zustands (3,9%) und 117 Zustände (91,4%) nicht- zuordenbar.

Anfangszustand $\mathbf{x_{anf}}$	Zustand nach sequentiller Abfragereihenfolge	Zustand nach sequentiell rücklaufender Abfragereihenfolge	Zustände nach Betrachtung aller Permutationen der sequentiellen Reihenfolge	Stabile Zustände von Anfangszustand $\mathbf{x_{anf}}$	Hammingabstand zum Zustand Spalte 5	absoluter Energieinhalt von $\mathbf{x_{anf}}$	Anfangszustand $\mathbf{x_{anf}}$ gleich $\xi^{(?)}$?
5	7	**39**	7, **39**	39	2	-16	$\xi^{(\mathbf{2})}$
7	**39**	**39**	**39**	39	1	-16	
8	**8**	**8**	**8**	8	0	-20	$-\xi^{(\mathbf{3})}$
11	**11**	**11**	**11**	11	0	-16	$-\xi^{(\mathbf{4})}$
37	7	**39**	7, **39**	39	1	-12	
39	**39**	**39**	**39**	39	0	-20	$\xi^{(\mathbf{1})}$
87	**119**	**119**	**119**	119	1	0	
88	**88**	**88**	**88**	88	0	-20	$-\xi^{(\mathbf{1})}$
116	**119**	118	118, **119**	119	2	-16	$\xi^{(\mathbf{4})}$
117	**119**	118	118, **119**	119	1	-12	
118	**119**	**119**	**119**	119	1	-16	
119	**119**	**119**	**119**	119	0	-20	$\xi^{(\mathbf{3})}$
122	**122**	**122**	**122**	122	0	-16	$-\xi^{(\mathbf{2})}$

Tabelle 3.7: HOPFIELD-*Netz aus 7 Neuronen, Gewichtsmatrix (3.13). Netzzustände, die durch asynchrone Abfrage sequentiell (Spalte 2), sequentiell rücklaufend (Spalte 3) und mit allen möglichen Permutationen der sequentiellen Reihenfolge (Spalte 4) entstehen. Es sind nur Anfangszustände aufgenommen, die entweder stabile Zustände sind oder im Attraktionsbecken von stabilen Zuständen liegen. Alle nicht aufgeführten Anfangszustände sind nicht- zuordenbar. Stabile Zustände fett. Spalte 5: Alle erreichbaren stabilen Zustände. Hammingabstände der Anfangszustände zum jeweilig erreichten stabilen Zustand (Spalte 6). Energieinhalt des Anfangzustands (Spalte 7)*

Mit dem gleichen Verfahren soll ein **Netz mit N = 10 Neuronen** untersucht werden. Folgende, zufällig ausgewählte Muster sollen in das Netz eingespeichert werden:

$$\begin{aligned}
\xi^{(1)} &= \begin{pmatrix} -1 & -1 & +1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 \end{pmatrix}, \\
\xi^{(2)} &= \begin{pmatrix} -1 & +1 & +1 & +1 & -1 & +1 & -1 & +1 & -1 & -1 \end{pmatrix}, \\
\xi^{(3)} &= \begin{pmatrix} -1 & -1 & -1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 \end{pmatrix}, \\
\xi^{(4)} &= \begin{pmatrix} +1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & +1 & -1 \end{pmatrix}. \qquad (3.14)
\end{aligned}$$

Nach HEBB ergibt sich folgende Gewichtsmatrix:

$$\mathbf{w} = \begin{pmatrix}
0 & 0 & 0 & 0 & +2 & -2 & 0 & -4 & +2 & +2 \\
0 & 0 & 0 & 0 & -2 & -2 & 0 & 0 & -2 & +2 \\
0 & 0 & 0 & 0 & -2 & +2 & 0 & 0 & +2 & -2 \\
0 & 0 & 0 & 0 & +2 & +2 & -4 & 0 & -2 & -2 \\
+2 & -2 & -2 & +2 & 0 & 0 & -2 & -2 & 0 & 0 \\
-2 & -2 & +2 & +2 & 0 & 0 & -2 & +2 & 0 & -4 \\
0 & 0 & 0 & -4 & -2 & -2 & 0 & 0 & +2 & +2 \\
-4 & 0 & 0 & 0 & -2 & +2 & 0 & 0 & -2 & -2 \\
+2 & -2 & +2 & -2 & 0 & 0 & +2 & -2 & 0 & 0 \\
+2 & +2 & -2 & -2 & 0 & -4 & +2 & -2 & 0 & 0
\end{pmatrix}. \qquad (3.15)$$

Tab. 3.8 zeigt die Anfangszustände, die stabile Zustände sind oder im Attraktionsbecken eines stabilen Zustands liegen. Man erkennt, daß nur die Muster $\xi^{(1)}$,$\xi^{(2)}$,$\xi^{(4)}$,$-\xi^{(1)}$,$-\xi^{(3)}$ und $-\xi^{(4)}$ stabile Zustände sind, nicht aber $\xi^{(3)}$ und $-\xi^{(2)}$.

Anfangszustand $\mathbf{x_{anf}}$	Zustand nach sequentiller Abfragereihenfolge	Zustand nach sequentiell rücklaufender Abfragereihenfolge	Zustände nach Betrachtung aller Permutationen der sequentiellen Reihenfolge	Stabile Zustände von Anfangszustand $\mathbf{x_{anf}}$	Hammingabstand zum Zustand Spalte 5 und ξ	absoluter Energieinhalt von $\mathbf{x_{anf}}$	Anfangszustand $\mathbf{x_{anf}}$ gleich $\xi^{(?)}$?
30	**158**	**158**	**158**	158	1	-16	
116	**244**	**244**	**244**	244	1, 0	-40	$\xi^{(3)}$
118	**244**	**244**	**244**	244	2, 1	-24	
158	**158**	**158**	**158**	158	0	-32	$\xi^{(1)}$
244	**244**	**244**	**244**	244	0, 1	-40	
246	**244**	**244**	**244**	244	1, 2	-32	
269	**269**	**269**	**269**	269	0	-32	$-\xi^{(4)}$
333	**269**	**269**	**269**	269	1	-16	
395	**907**	911	**907**, 911	907	1	-24	
399	**907**	911	**907**, 911	907	2	-24	
414	**158**	**158**	**158**	158	1	-24	
452	**468**	**468**	**468**	468	1	-40	
468	**468**	**468**	**468**	468	0	-40	$\xi^{(2)}$
469	**468**	**468**	**468**	468	1	-16	
476	**468**	**468**	**468**	468	1	-24	
553	**811**	**811**	**811**	811	2, 1	-24	
555	**811**	**811**	**811**	811	1, 0	-40	$-\xi^{(2)}$
619	**811**	**811**	**811**	811	2, 1	-24	
626	**754**	**754**	**754**	754	1	-24	
670	**158**	**158**	**158**	158	1	-16	
754	**754**	**754**	**754**	754	0	-32	$\xi^{(4)}$
755	**754**	**754**	**754**	754	1	-16	
762	**754**	**754**	**754**	754	1	-16	
809	**811**	**811**	**811**	811	1, 2	-32	
811	**811**	**811**	**811**	811	0, 1	-40	
865	**865**	**865**	**865**	865	0	-32	$-\xi^{(1)}$
869	**865**	**865**	**865**	865	1	-16	
875	**811**	**811**	**811**	811	1, 2	-24	
881	**865**	**865**	**865**	865	1	-16	
906	**907**	**907**	**907**	907	1	-16	
907	**907**	**907**	**907**	907	0	-40	$-\xi^{(3)}$
911	**907**	**907**	**907**	907	1	-24	
923	**907**	**907**	**907**	907	1	-16	
1010	**754**	**754**	**754**	754	1	-16	

Tabelle 3.8: HOPFIELD- *Netzwerk aus $N = 10$ Neuronen. Gewichtsmatrix (3.15). Spalte 1: Anfangszustände, die entweder stabile Zustände sind oder im Attraktionsbereich stabiler Zustände liegen. Abfrage asynchron sequentiell (Spalte 2), asynchron sequentiell rücklaufend (Spalte 3) und mit allen Permutationen der sequentiellen Reihenfolge (Spalte 4), stabile Zustände fett. Spalte 5: Erreichte stabile Zustände von Anfangszustand aus. Spalte 6: Hamming Abstände von Anfangszustand zum stabilen Zustand aus Spalte 5 und, wenn verschieden, zu ξ. Spalte 7: Absoluter Energieinhalt des Anfangszustands*

In der Tab. 3.8 treten zum ersten Male stabile Zustände auf (244 und 811), die nicht den fundamentalen Zuständen zuzuordnen sind. Der Hamming Abstand dieser zwei stabilen Zustände zu dem nächstgelegen fundamentalen Muster $\xi^{(\mathbf{3})}$ und $-\xi^{(\mathbf{2})}$ beträgt 1. Allerdings sind diese Zustände keine zusätzlichen stabilen Zustände, die neben den fundamentalen Muster existieren. Vielmehr liegen sie sehr nahe an den Mustern $\xi^{(\mathbf{3})}$ und $-\xi^{(\mathbf{2})}$, die selbst keine stabilen Zustände sind, aber diese Muster sind ihrerseits im Attraktionsbecken dieser Zustände. Das bedeutet, daß die Zustände 244 und 811 in der Gewichtsmatrix $\mathbf{w}$ als fundamentale Muster stabile Zustände mit einem residuellen Fehler (siehe Abschnitt 2.6.1) sind und ein Attraktionsbecken besitzen. *Es ist daher eine Fehlerkorrektur möglich.*
In diesem Netz gibt es insgesamt 8 (0,78%) stabile Zustände (6 Muster oder Antimuster und 2 mit residuellem Fehler), 26 (2,54%) im Attraktionsbecken und 990 (96,68%) Zustände, die je nach Abfragereihenfolge in verschiedene stabile Zustände streben können (nicht- zuordenbare Zustände).
Gleichung (2.6) berechnet das innere Argument y_i jedes einzelnen Neurons i. Wenn man ein fundamentales Muster $\xi^{(\mathbf{b})}$ mit $1 \leq b \leq p$ als Anfangsmuster in das Netz eingibt, und wenn dieses fundamentale Muster stabil sein soll, so muß für nach Gleichung (2.15) gelten

$$\xi_i^{(b)} = \text{ sgn}[y_i], \text{ für alle Neuronen } i. \tag{3.16}$$

Nun aber läßt sich das innere Argument y_i für jedes Neuron i nach Gleichung (2.36) in Analogie zur Energiefunktion in einen Signalterm y_s und in einen Rauschterm y_r aufteilen. Die Tab. 3.9 illustriert dies am Beispiel des fundamentalen Musters $\xi^{(\mathbf{3})}$. Aus dieser Tabelle erkennt man, daß für alle Neurone bis auf Neuron 3 der Zustand im fundamentalen Muster mit dem Vorzeichen des inneren Arguments y_i übereinstimmt. Ferner kann man erkennen, daß für alle Neurone bis auf Neuron 3 der Betrag des Signalterms $|y_s|$ größer ist als

	Neuron i mit $i =$									
	1	2	3	4	5	6	7	8	9	10
fund. Muster $\xi^{(3)}$	-1	-1	-1	+1	+1	+1	-1	+1	-1	-1
inneres Argument y_i	-8	-4	0	12	4	12	-12	8	-8	-12
davon Signalterm y_s	-9	-9	-9	9	9	9	-9	9	-9	-9
davon Rauschterm y_r	1	5	9	3	-5	3	-3	-1	1	-3

Tabelle 3.9: *Vergleich des Anfangszustands $\xi^{(3)}$ (Zeile 1) mit dem inneren Argument y_i (Zeile 2) für jedes Neuron i, $1 \leq i \leq 10$ (Spalte 2 bis 11). Das innere Argument y_i ist eine Addition des Signalterms (3. Zeile) und des Rauschterms (4. Zeile). Dabei ist für das Neuron 3 mit der Vorzeichenfunktion nach Gleichung (2.7) sgn$[0] \neq -1$. Deshalb ist das fundamentale Muster $\xi^{(3)}$ kein stabiler Zustand des Netzwerkes*

der Betrag des Rauschterms $|y_r|$. Neuron 3 allerdings besitzt einen gleichen Betrag des Signal- wie auch des Rauschterms. Mit der Vorzeichenfunktion nach Gleichung (2.7) ist $\xi_3^{(3)} \neq \text{sgn}[y_3]$. Damit gilt Gleichung (3.16) nicht, $\xi^{(3)}$ ist somit kein stabiler Zustand.
Das fundamentale Antimuster $-\xi^{(3)}$ unterscheidet sich vom fundamentalen Muster $\xi^{(3)}$ und von dessen innerem Argument y_i für jedes Neuron i nur jeweils im Vorzeichen. Das gilt auch für Signal- und Rauschterm. Für das Antimuster $-\xi^{(3)}$ ist jedoch die Gleichung (3.16) für jedes Neuron i, also auch für Neuron 3, mit der Vorzeichenfunktion nach Gleichung (2.7) erfüllt. Somit ist das Antimuster $-\xi^{(3)}$ ein stabiler Zustand.
Nur die nichtlineare Vorzeichenfunktion in Gleichung (2.7) gibt also für den Fall, in dem der Betrag des Signalterms $|y_s|$ und der Betrag des Rauschterms $|y_r|$ gleich ist, an, ob ein stabiler Zustand vorliegt oder nicht. *Während die Signumfunktion in Tab. 3.2 die Größe der Attraktionsbecken beeinflußt hatte, beeinflußt sie hier die Stabilität der fundamentalen Muster.*
Dieselben Verhältnisse wie für das fundamentale Muster $\xi^{(3)}$ und das Neuron 3 treten auch beim fundamentalen Muster $-\xi^{(2)}$ im Neuron 2 auf.

Im folgenden wird ein HOPFIELD-**Netzwerk aus N = 8 Neuronen** gezeigt, bei dem sich viele der bisherigen Phänomene zeigen und überlagern. Es sollen folgende Muster eingespeichert werden, die zufällig gewählt wurden.

$$\begin{aligned}
\xi^{(1)} &= \begin{pmatrix} -1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 \end{pmatrix}, \\
\xi^{(2)} &= \begin{pmatrix} +1 & +1 & +1 & +1 & -1 & +1 & +1 & -1 \end{pmatrix}, \\
\xi^{(3)} &= \begin{pmatrix} -1 & +1 & -1 & +1 & +1 & +1 & +1 & -1 \end{pmatrix}. \qquad (3.17)
\end{aligned}$$

Daraus ergibt sich nach HEBB folgende Gewichtsmatrix:

$$\mathbf{w} = \begin{pmatrix}
0 & +1 & +3 & -1 & -1 & -1 & -1 & +1 \\
+1 & 0 & +1 & +1 & +1 & +1 & +1 & -1 \\
+3 & +1 & 0 & -1 & -1 & -1 & -1 & +1 \\
-1 & +1 & -1 & 0 & -1 & +3 & +3 & -3 \\
-1 & +1 & -1 & -1 & 0 & -1 & -1 & +1 \\
-1 & +1 & -1 & +3 & -1 & 0 & +3 & -3 \\
-1 & +1 & -1 & +3 & -1 & +3 & 0 & -3 \\
+1 & -1 & +1 & -3 & +1 & -3 & -3 & 0
\end{pmatrix}. \qquad (3.18)$$

Die Ergebnisse sind in Tab. 3.10 dargestellt. Das Netz zeigt 4 stabile Zustände, von denen aufgrund der Symmetrie nur 2 in Tab. 3.10 dargestellt sind.
In diesem Netz entsprechen zwei stabile Zustände (9 und 246) dem fundamentalen Muster bzw. Antimuster $\pm\xi^{(2)}$, während zwei stabile Zustände (86 und 169) nicht den einzuspeichernden Mustern $\pm\xi^{(1)}$ und $\pm\xi^{(3)}$ zuzuordnen sind. Allerdings liegen diese Muster jeweils im Attraktionsbecken von Zustand 86 oder 169. Damit ist die Behauptung, ein fundamentales Muster als Anfangszustand kann nicht weit weg laufen (s. Abschnitt 2.6.1), nachvollzogen. Man kann hier aber nicht von einem residuellen Fehler reden, wie in Tab. 3.8. Denn z. B. der stabile Zustand 86 hat sowohl $\xi^{(3)}$ als auch $\xi^{(1)}$ in seinem Attraktionsbecken. *Es gibt also stabile Zustände, die nicht für die Mustererkennung zu gebrauchen sind.*
Insgesamt treten in diesem Netz 4 (1,6%) stabile Zustände, 44 (17,2%) Zustände in Attraktionsbecken und 208 (81,2%) nicht- zuordenbare Zustände auf.

Anfangszustand $\mathbf{x_{anf}}$	Zustand nach sequentiller Abfragereihenfolge	Zustand nach sequentiell rücklaufender Abfragereihenfolge	Zustände nach Betrachtung aller Permutationen der sequentiellen Reihenfolge	Stabile Zustände von Anfangszustand $\mathbf{x_{anf}}$	Hammingabstand zum Zustand Spalte 5	absoluter Energieinhalt von $\mathbf{x_{anf}}$	Anfangszustand $\mathbf{x_{anf}}$ gleich $\xi^{(?)}$?
6	22	**86**	22, **86**	86	2	-8	
8	**9**	**9**	**9**	9	1	-6	
9	**9**	**9**	**9**	9	0	-24	$-\xi^{(2)}$
11	**9**	**9**	**9**	9	1	-6	
13	**9**	**9**	**9**	9	1	-6	
14	**86**	**86**	22, **86**, 94	86	3	-6	
18	22	**86**	22, **86**	86	2	-8	
20	22	**86**	22, **86**	86	2	-8	
22	**86**	**86**	**86**	86	1	-30	$\xi^{(1)}$
23	22	**86**	22, **86**	86	2	-8	
25	**9**	**9**	**9**	9	1	-6	
26	94	**86**	22, **86**, 94	86	3	-6	
28	94	**86**	22, **86**,94	86	3	-6	
30	**86**	**86**	**86**	86	2	-24	
31	94	**86**	22, **86**, 94	86	3	-6	
70	22	94	22, **86**, 94	86	1	-6	
78	**86**	94	**86**, 94	86	2	-8	
82	22	**86**	22, **86**, 94	86	1	-6	
84	22	**86**	22, **86**, 94	86	1	-6	
86	**86**	**86**	**86**	86	0	-32	
87	22	**86**	22, **86**, 94	86	1	-6	
90	94	**86**	**86**, 94	86	2	-8	
92	94	**86**	**86**, 94	86	2	-8	
94	**86**	**86**	**86**	86	1	-30	$\xi^{(3)}$
95	94	**86**	**86**, 94	86	2	-8	

Tabelle 3.10: HOPFIELD- *Netz aus $N = 8$ Neuronen. Gewichtsmatrix (3.18). Anfangszustände (Spalte 1), die stabile Zustände sind oder in einem Attraktionsbecken liegen. Es ist nur die 1. Hälfte dieser Anfangszustände dargestellt, die 2. Hälfte ist symmetrisch dazu. Zustände nach asynchroner Abfrage sequentiell (Spalte 2), sequentiell rücklaufend (Spalte 3) und mit allen Permutationen der sequentiellen Reihenfolge (Spalte 4). Stabile Zustände fett. Spalte 5: Erreichbarer stabiler Zustand. Spalte 6: Hamming Abstand des Anfangszustands zum stabilen Zustand. Spalte 7: Energieinhalt des Anfangszustands*

Kapitel 4

Netzwerke aus 32 Neuronen

4.1 Einführung zufälliger Abfrageblöcke und Vergleich mit den Ergebnissen aus Abschnitt 3.1

Das bisherige Untersuchungsverfahren war nur bei Netzwerken mit bis zu 10 Neuronen möglich. Bei größeren Netzwerken war der Rechenaufwand zu groß. Deshalb wurden für die Untersuchung der größeren Netzwerke folgende Veränderungen vorgenommen: Es wurden statt der Verwendung aller Permutationen der sequentiellen Reihenfolge zufällige Abfrageblöcke verwendet, und es wurde nicht mehr von allen möglichen Anfangszuständen ausgegangen. Zunächst soll untersucht werden, ob man mit zufälligen Abfrageblöcken zu ähnlichen Ergebnissen kommt wie bei der Verwendung aller Permutationen der sequentiellen Abfragereihenfolge. Das soll am Beispiel 3 vom Abschnitt 3.1.2 erfolgen.

Gegeben ist das Netzwerk aus $N = 5$ Neuronen mit der Gewichtsmatrix (3.9). Spalte 4 aus Tab. 3.5 zeigt die Zustände, die nach Verwendung aller Permutationen der sequentiellen Reihenfolge erreicht wurden. Da alle möglichen Zustände als Anfangszustände gewählt wurden, kann man von den nach einem Abfrageblock erreichten Zuständen alle weiteren erreichbaren Zustände ablesen. So lassen sich die Trajektorien von allen Zuständen zu den entsprechenden stabilen Zuständen bilden.
Im folgenden wurde nun zum Vergleich der in Abschnitt 2.4.5 geschilderte zufällige Abfragemodus verwendet und jeweils nur ein einziger Abfrageblock

angewandt. So wurde z. B. die Abfragereihenfolge zufällig mit

$$\mathbf{A} = \begin{pmatrix} 3 & 1 & 2 & 4 & 1 & 5 \end{pmatrix}. \tag{4.1}$$

bestimmt. Sie besitzt die Eigenschaft, daß sie alle Neurone i, $1 \leq i \leq 5$ enthält.
Man geht z. B. von dem Anfangszustand $\mathbf{x_{dec}}(0) = 2$ aus.
Zu diskreten Zeitpunkten t, $1 \leq t \leq 6$ errechnet Gleichung (2.5) den neuen Zustand des Neurons i, welches in der Abfragereihenfolge $\mathbf{A}$ nach (4.1) im Element $t = i$ bestimmt ist. Mit dem asynchronen Berechnungsmodus ergibt dies folgende Zustände.

$x_3(t = 1) = \text{sgn}[-2] = -1$. Dies ergibt Gesamtzustand $\mathbf{x_{dec}}(1) = 2$ mit
$\mathbf{x}(1) = \begin{pmatrix} -1 & -1 & -1 & +1 & -1 \end{pmatrix}$.

$x_1(t = 2) = \text{sgn}[\pm 0] = +1$. Dies ergibt Gesamtzustand $\mathbf{x_{dec}}(2) = 18$ mit
$\mathbf{x}(2) = \begin{pmatrix} +1 & -1 & -1 & +1 & -1 \end{pmatrix}$.

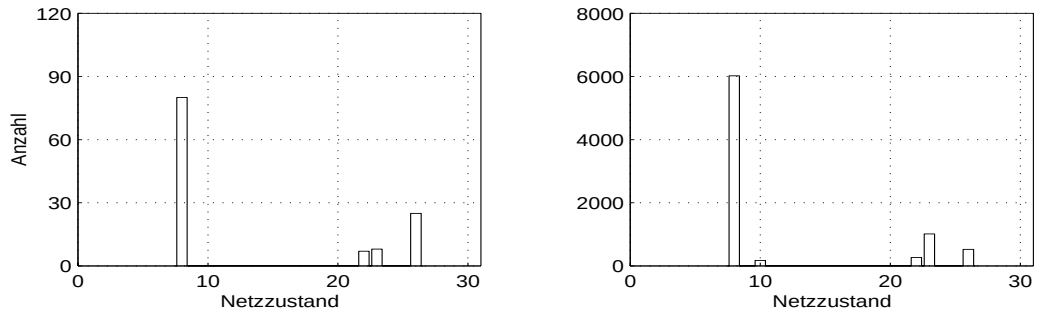
$x_2(t = 3) = \text{sgn}[+4] = +1$. Dies ergibt Gesamtzustand $\mathbf{x_{dec}}(3) = 26$ mit
$\mathbf{x}(3) = \begin{pmatrix} +1 & +1 & -1 & +1 & -1 \end{pmatrix}$.

$x_4(t = 4) = \text{sgn}[+4] = +1$. Dies ergibt Gesamtzustand $\mathbf{x_{dec}}(4) = 26$ mit
$\mathbf{x}(4) = \begin{pmatrix} +1 & +1 & -1 & +1 & -1 \end{pmatrix}$.

$x_1(t = 5) = \text{sgn}[-2] = -1$. Dies ergibt Gesamtzustand $\mathbf{x_{dec}}(5) = 10$ mit
$\mathbf{x}(5) = \begin{pmatrix} -1 & +1 & -1 & +1 & -1 \end{pmatrix}$.

$x_5(t = 6) = \text{sgn}[-8] = -1$. Dies ergibt den Gesamtzustand nach jetzt abgeschlossenem Abfrageblock $\mathbf{x_{dec}}(6) = 10$ mit
$\mathbf{x}(6) = \begin{pmatrix} -1 & +1 & -1 & +1 & -1 \end{pmatrix}$.

Wenn man diesen Zustand mit Zuständen vergleicht, die nach Anwendung aller Permutationen der sequentiellen Reihenfolge (Spalte 4 in Tab. 3.5) von dem Anfangszustand $\mathbf{x_{dec}} = 2$ erreicht wurden, so fehlt dort dieser Zustand. Es kommen also nach Verwendung von zufälligen Abfrageblöcken erreichbare Zustände vor, die mit allen Permutationen der sequentiellen Reihenfolge nicht auftraten. Betrachtet man alle erreichbaren Zustände vom Anfangszustand 26 (wird vom Zustand 2 aus erreicht) aus in Tab. 3.5, so ist Zustand 10 erreichbar. Den Zustand 10 kann man deshalb einen **weiterführenden Zustand** nennen, weil er vom Zustand 2 aus in mehr als einem Abfrageblock



A B

Abbildung 4.1: HOPFIELD- *Netz aus 5 Neuronen. Gewichtsmatrix (3.9), Anfangszustand* $\mathbf{x_{dec}} = 2$. *A: Vergleich der Anzahl der Zustände, die bei den 120 Permutationen der sequentiell asynchronen Abfragereihenfolge erreicht wurden. B: Anzahl der Zustände, die bei 8 000 zufälligen Abfrageblöcken erreicht wurden. Der zufällige Abfragemodus erreicht einen weiteren Zustand 10. Das Verhältnis der Summe der Zustände 8, 10 und 26 (Attraktionsbecken von Zustand 8) und der Summe der Zustände 22 und 23 (Attraktionsbecken von Zustand 23) ist ungefähr wie in A*

der Permutationen der sequentiellen Abfragereihenfolge erreicht wird.
Im Besonderen erfolgt zum Zeitpunkt $t = 2$ eine Änderung des Zustands von Neuron 1. Zu einem späteren Zeitpunkt $t = 5$ wird der Zustand des Neurons 1 auf seinen Ausgangszustand zurückgesetzt. Diese Zurücksetzung war bei der Betrachtung der sequentiellen Reihenfolge und deren Permutationen nicht möglich, da jedes Neuron nur einmal seinen neuen Zustand nach Gleichung (2.5) berechnete. Betrachtet man den jeweiligen Zustand nach der sequentiellen Abfragereihenfolge und allen möglichen Permutationen, so stellt man fest, daß man nicht zum Zustand 8, sondern zum stabilen Zustand 23 und dessen Attraktionsbecken vom Anfangszustand 2 nur dann gelangen kann, wenn Neuron 1 vor Neuron 2, 3, 4, und danach Neuron 3 vor Neuron 2 abgefragt wird. Das ergibt eine Anzahl von zusammen 15 möglichen Permutationen (siehe Abb. 4.1). Im gewählten Beispiel (Gleichung (4.1)) wurde Neuron 2 nach Neuron 1 und vor Neuron 3 abgefragt. Daraufhin ist das Attraktionsbecken von Zustand 23 nicht mehr erreichbar.
Das Histogramm in Abb. 4.1A zeigt, daß bei 15 der insgesamt 120 Permutationen (12,5%) das Netz zu den Zuständen 22 oder 23 strebt, wenn man vom Anfangszustand 2 beginnt. Alle anderen Permutationen streben zu den Zuständen 8 und 26 (87,5%), wobei 26 im Attraktionsbecken von 8 liegt.

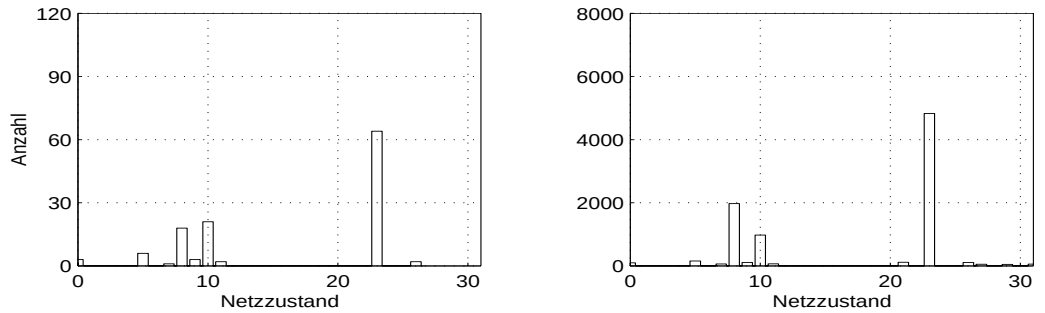


Abbildung 4.2: HOPFIELD- *Netz aus 5 Neuronen. Gewichtsmatrix (3.9), Anfangszustand* $\mathbf{x_{dec}} = 28$. *A: Anzahl der Zustände, die bei den 120 Permutationen der sequentiell asynchroner Abfragereihenfolge erreicht wurden. B: Anzahl der Zustände, die bei 8 000 zufälligen Abfrageblöcken erreicht wurden.*

Generiert man 8 000 zufällige Abfrageblöcke, und betrachtet zu jedem Abfrageblock den erreichten Zustand, so erhält man, wie im Histogramm der Abb. 4.1B ersichtlich, einen weiteren erreichbaren Zustand 10 (weiterführender Zustand). Ferner streben 16% aller Abfrageblöcke zu den Zuständen 22 und 23 und 84% zu den Zuständen 8, 10 oder 26. Das Verhältnis ist ungefähr dem von Abb. 4.1A.
Betrachtet man beim gleichen Netzwerk den Anfangszustand $\mathbf{x_{dec}} = 28$, so treten bei zufälliger Abfragereihenfolge Zustände auf, die man mit den Permutationen der sequentiellen Reihenfolge weder in einem noch in mehreren Abfrageblöcken (weiterführende Zustände) erreichen kann. Abb. 4.2A stellt die Anzahl der Zustände dar, die durch die Permutationen der sequentiellen Abfragereihenfolge erreicht wurden (4. Spalte der Tab. 3.5). Abb. 4.2B zeigt die Anzahl der Zustände, die bei 8 000 zufällig gewählten Abfrageblöcken erreicht werden. Die Zustände 21, 27, 29, 31 treten in Abb. 4.2A nicht auf. Dabei kann man die Zustände 21 und 31 als weiterführende Zustände bezeichnen, denn sie können auch durch 2 sequentielle Abfrageblöcke erreicht werden. Dagegen sind die Zustände 27 und 29 mit sequentieller Abfrage nicht zu erreichen. Man kann vom Anfangszustand 28 aus den Zustand 29 dann erreichen, wenn man mit der Abfragereihenfolge $\mathbf{A} = \begin{pmatrix} 1 & 4 & \dots \end{pmatrix}$ beginnt. Zum ersten Zeitpunkt erhält man den Zustand 12, zum zweiten den Zustand 14. Von dort aus kann man die Zustände 27 und 29 erreichen, wenn alle Neuronen abgefragt werden (Tab. 3.5). So sind weitere Zustände aufgrund von Querverbindungen und durch wiederholende Abfragen erreichbar.

Mit zufälligen Abfrageblöcken kann man also von einem Anfangszustand aus neben den erreichbaren Zuständen durch Abfrage der Permutationen der sequentiellen Reihenfolge noch weiterführende Zustände (durch die längere Abfragereihenfolge) und aufgrund von Querverbindungen weitere Zustände (durch wiederholende Abfragen) erreichen. Man beachte aber dabei, daß bei der Verwendung zufälliger Abfrageblöcke nur Zustände erreicht werden können, die auch nach der Verwendung der Abfrage von allen Permutationen der sequentiellen Reihenfolge von allen Anfangszuständen aus erreicht wurden. *Die Verwendung zufälliger Abfrageblöcke führt also vorwiegend zum gleichen Ergebnis wie die Verwendung einer Folge von Abfrageblöcken aus Permutationen der sequentiellen Reihenfolge. Abweichungen treten nur mit geringer Wahrscheinlichkeit auf.*

	$\xi^{(1)}$	$\xi^{(2)}$	$\xi^{(3)}$	$\xi^{(4)}$	$\xi^{(5)}$	$\xi^{(6)}$
$\xi^{(1)}$	0	16	18	8	17	15
$\xi^{(2)}$	16	0	16	16	19	19
$\xi^{(3)}$	18	16	0	16	17	15
$\xi^{(4)}$	8	16	16	0	15	17
$\xi^{(5)}$	17	19	17	15	0	18
$\xi^{(6)}$	15	19	15	17	18	0

Tabelle 4.1: HOPFIELD- *Netz aus 32 Neuronen. Gewichtsmatrix für die fundamentalen Muster (4.2) nach* HEBB. *Hamming Abstände zwischen den einzelnen fundamendamentalen Mustern. Sie sind jeweils ungefähr* $\frac{N}{2}$ *voneinander entfernt, bis auf die fundamentalen Muster 1 und 4*

4.2 Einspeisung von $p = 6$ fundamentalen Mustern in ein Netzwerk von 32 Neuronen, Bildung eines Trajektorien- Baumes, nähere Untersuchung der Attraktionsbecken von stabilen Zuständen

Für das weitere Vorgehen sollen in einem neuronalen Netz aus $N = 32$ Neuronen stabile Zustände und das genaue Aussehen ihrer Attraktionsbecken bestimmt werden. Folgende 6 fundamentale Muster werden zufällig gewählt.

$$\begin{aligned}
\xi^{(\mathbf{1})}_{\mathbf{dec}} &= 4\ 065\ 956\ 275 \\
\xi^{(\mathbf{2})}_{\mathbf{dec}} &= 2\ 888\ 508\ 259 \\
\xi^{(\mathbf{3})}_{\mathbf{dec}} &= 0\ 354\ 225\ 751 \\
\xi^{(\mathbf{4})}_{\mathbf{dec}} &= 4\ 039\ 187\ 123 \\
\xi^{(\mathbf{5})}_{\mathbf{dec}} &= 0\ 015\ 545\ 496 \\
\xi^{(\mathbf{6})}_{\mathbf{dec}} &= 4\ 130\ 835\ 516
\end{aligned} \tag{4.2}$$

Betrachtet man die Hamming Abstände zwischen den einzelnen fundamentalen Mustern (Tab. 4.1), so sind sie jeweils ungefähr gleich groß, mit Ausnahme des Abstandes zwischen den fundamentalen Mustern 1 und 4. Wenn man die Gewichtsmatrix nach HEBB aufstellt, so ergibt sich eine Gewichtsmatrix, die aus 7 verschiedenen Werte mit der Verteilung nach Abb. 4.3 aufgebaut ist.

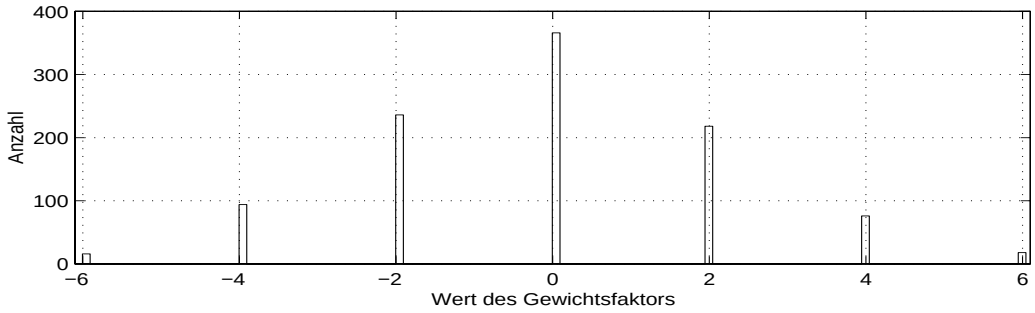


Abbildung 4.3: HOPFIELD- *Netz aus $N = 32$ Neuronen mit $p = 6$ eingespeicherten Mustern (4.2). Verteilung der 7 verschiedenen Werte bei Bildung der Gewichtsmatrix nach* HEBB

Bildung eines Trajektorien- Baumes. Das bisherige Verfahren, von allen möglichen Anfangszuständen auszugehen, war hier nicht mehr möglich. Statt dessen wurde nur von wenigen Anfangszuständen ausgegangen, die zunächst zufällig ausgewählt wurden. Es bestand dabei die Hoffnung, bei der Verfolgung aller Trajektorien, die von einem Zustand sehr hoher Energie ausgehen, viele stabile Zustände, und zwar Muster und spurious states, zu finden. Die später gewählten Anfangzustände waren dann jeweils bestimmte Hamming Abstände von den stabilen Zuständen entfernt.
Ich beginne mit einem Anfangszustand, der zufällig gewählt wurde

$$\mathbf{x_{anf,dec}} = 1\ 366\ 904\ 109. \tag{4.3}$$

Von dort aus beginne ich

(a) mit der asynchron sequentiellen Reihenfolge mit $\mathbf{A} = \begin{pmatrix} 1 & 2 & \dots & 32 \end{pmatrix}$. Daraufhin erhält man den Zustand $\mathbf{x_{dec}} = 387\ 370\ 076$. Dieser Zustand ist bereits in einem Attraktionsbecken des spurious states

$$\mathbf{x_{sp_1,dec}} = 255\ 257\ 676. \tag{4.4}$$

Um dies festzustellen, werden zufällige Abfrageblöcke (Definition s. Abschnitt 2.4.5) hintereinander gesetzt. Dies geschieht solange, bis ein stabiler Zustand erreicht und verifiziert ist. Eine solche Folge von Abfrageblöcken wird im folgendem Abfrageblockfolge genannt. Die Zwischenzustände nach den einzelnen Abfrageblöcken fehlen bei dieser Darstellung. Bei 24 000 zufällig gewählten Abfrageblockfolgen (bis ein stabiler Zustand erreicht ist) strebte das Netz immer in den spurious state

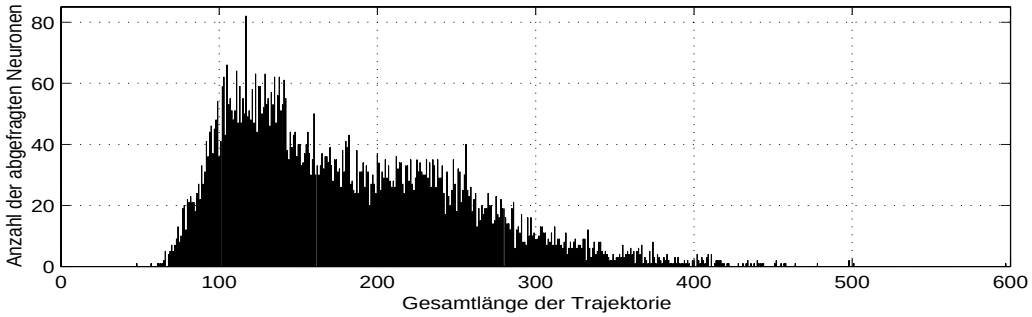


Abbildung 4.4: HOPFIELD- *Netz aus 32 Neuronen. Gewichtsmatrix für die Muster (4.2) nach* HEBB. *Histogramm der Längen (Abszisse) von insgesamt 8 000 Abfrageblöcken vom Anfangszustand* 387 370 076. *Dabei strebt das Netz mit unterschiedlichen Zeitlängen zum spurious state* $\mathbf{x_{sp_1,dec}}$ *(4.4)*

$\mathbf{x_{sp_1,dec}}$ (4.4). Dabei wurden aus programmtechnischen Gründen die Längen der einzelnen Abfrageblöcke und Zahl der Abfrageblöcke einer Folge nicht ermittelt. Bei weiteren 8 000 Trajektorien wurde die Gesamtlänge der jeweiligen Abfrageblockfolgen untersucht, d. h. die Anzahl der abgefragten Neuronen, bis dieser spurious state erreicht wurde, angegeben. Diese Längen variieren sehr stark (Abb. 4.4). Man erreicht also diesen spurious state $\mathbf{x_{sp_1,dec}}$ (4.4) auf unterschiedlichen Trajektorien.

(b) Fragt man den Zustand (4.3) mit der asynchron sequentiell rücklaufenden Abfragereihenfolge ab, so erreicht man Zustand

$$\mathbf{x_{sp_2,dec}} = 1\ 389\ 730\ 972. \tag{4.5}$$

Dieser Zustand ist ein weiterer spurious state.

(c) Ein weiterer Abfrageblock wurde aus einer Abfragereihenfolge aus Permutationen der sequentiellen Reihenfolge entnommen (zufällige Permutation), z. B.

$$\begin{aligin} A = (&24\ 23\ 28\ 16\ 5\ 31\ 26\ 11\ 1\ 7\ 20\ 12\\ &10\ 29\ 32\ 19\ 2\ 30\ 15\ 21\ 14\ 4\ 8\ 13\\ &3\ 22\ 25\ 9\ 6\ 17\ 18\ 27). \end{aligin} \tag{4.6}$$

Damit erreicht man vom Zustand (4.3) aus Zustand 1 406 508 181. Von dort aus strebte das Netz bei 8 000 zufälligen Abfrageblockfolgen mit

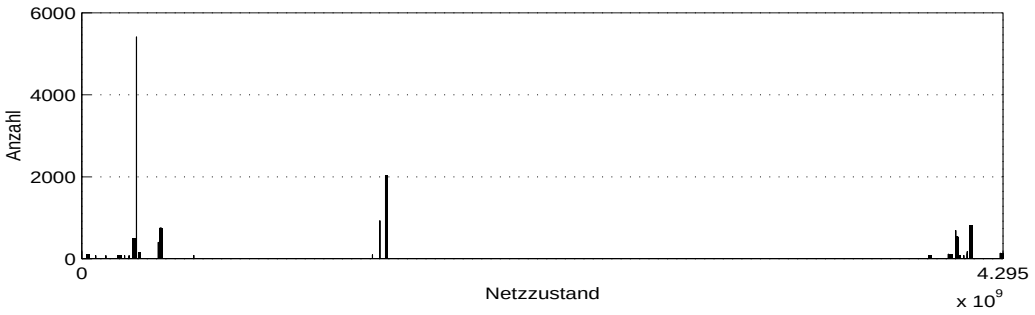


Abbildung 4.5: HOPFIELD- *Netz aus 32 Neuronen. Gewichtsmatrix aus den fundamentalen Mustern (4.2) nach* HEBB. *Anfangszustand (4.3). Anzahl der jeweils erreichten stabilen Zustände bei der Untersuchung von 12 000 Trajektorien. Fett: fundamentale Muster. Dünn: spurious states*

87% zum spurious state $\mathbf{x_{sp_1,dec}}$ (4.4) und mit 13% zum fundamentalen Muster $-\xi^{(\mathbf{2})}$.

Geht man vom Anfangszustand (4.3) nicht mit den drei eben genannten Abfrageblöcken (a) - (c) aus, sondern mit einer Aneinanderreihung zufälliger Abfrageblöcke, so kann man insgesamt 31 stabile Zustände erreichen. 12 000 Trajektorien vom Anfangszustand (4.3) strebten in sehr unterschiedlicher Anzahl diese 31 Zustände an (Abb. 4.5). Fast die Hälfte der Trajektorien strebten zu dem spurious state $\mathbf{x_{sp_1,dec}}$ (4.4). Unter diesen 31 verschiedenen stabilen Zuständen gab es 11 Zustände, die den fundamentalen Mustern und ihren Antimustern entsprechen, 20 waren also spurious states.

Untersuchung der Attraktionsbecken von stabilen Zuständen. Es soll als erstes das fundamentale Muster $\xi^{(\mathbf{3})}$ und seine Umgebung untersucht werden. Zunächst ermittelt man alle Zustände, die innerhalb eines gewissen Hamming Abstands um einen stabilen Zustand liegen. Um zu bestimmen, ob diese Zustände in einem Attraktionsbecken liegen, kann man aber nicht alle Permutationen der sequentiellen Reihenfolge anwenden,weil der Rechenaufwand zu groß wäre. Vielmehr beschränkt man sich auf eine gewisse Anzahl zufälliger Abfrageblockfolgen. Im Abschnitt 4.1 wurde festgestellt, daß diese Beschränkung zu weitgehend übereinstimmenden Ergebnissen führt.

(1) Man ermittelt zunächst alle N Zustände, die 1 Hamming Abstand von $\xi^{(\mathbf{3})}$ entfernt sind. Von dort aus strebten alle Zustände bei 8 000 Folgen von Abfrageblöcken in das fundamentale Muster $\xi^{(\mathbf{3})}$.

(2) Im nächsten Schritt ermittelt man alle möglichen $\binom{N}{k} = 496$ Anfangszustände, die $k = 2$ Hamming Abstände vom stabilen Muster $\xi^{(3)}$ entfernt sind. Verwendet man nur die asynchron sequentielle Reihenfolge, dann streben alle diese Anfangszustände zum fundamentalen Muster mit Ausnahme des Anfangszustands

$$\mathbf{x}_{\mathbf{anf_{as},dec}} = 354\ 225\ 748. \tag{4.7}$$

Verwendet man von allen diesen Anfangszuständen die asynchron rücklaufende Abfragereihenfolge, so streben alle Anfangszustände zum fundamentalen Muster $\xi^{(3)}$ mit Ausnahme des Zustands

$$\mathbf{x}_{\mathbf{anf_{asr},dec}} = 354\ 225\ 757. \tag{4.8}$$

Benutzt man die Abfragereihenfolge (4.6), so streben alle Anfangszustände zum fundamentalen Muster $\xi^{(3)}$ mit Ausnahme des Zustands

$$\mathbf{x}_{\mathbf{anf_{apz}}} = 354\ 225\ 758. \tag{4.9}$$

Benutzt man weitere 8 000 zufällige Permutationen der sequentiellen Reihenfolge, und wendet sie auf alle Anfangszustände mit einem Hamming Abstand 2 vom fundamentalen Muster $\xi^{(3)}$ an, so streben alle Zustände in das fundamentale Muster mit Ausnahme der Zustände (4.7 - 4.9). Dies gilt auch für 8 000 zufällige, einzelne Abfrageblöcke.
Von den 528 Anfangszuständen mit bis zu 2 Hamming Abständen vom fundamentalen Muster $\xi^{(3)}$ gehören also 525 zum Attraktionsbecken und 3 (4.7 - 4.9) sind nicht- zuordenbare Zustände.

In der Literatur wird normalerweise ein Attraktionsbecken so untersucht, daß man nicht von allen Anfangszuständen mit einem bestimmten Hamming Abstand ausgeht, sondern nur von einem Teil davon. So könnte man hier rein zufällige Anfangszustände innerhalb eines Hamming Abstandes von 2 um das fundamentale Muster und rein zufällige Abfragereihenfolgen auswählen. Bei Anwendung dieses Verfahrens habe ich bei 8 000 Versuchen gefunden, daß alle Anfangszustände im Attraktionsgebiet liegen. Die Wahrscheinlichkeit, daß bei einer bestimmten Abfragereihenfolge genau der eine Zustand, der bei dieser Abfragereihenfolge nicht in das Muster strebt, ausgewählt wird, ist äußerst gering $(0,2\%)$. *Die ermittelte Größe eines Attraktionsbeckens hängt also von der Art der Untersuchung ab.*

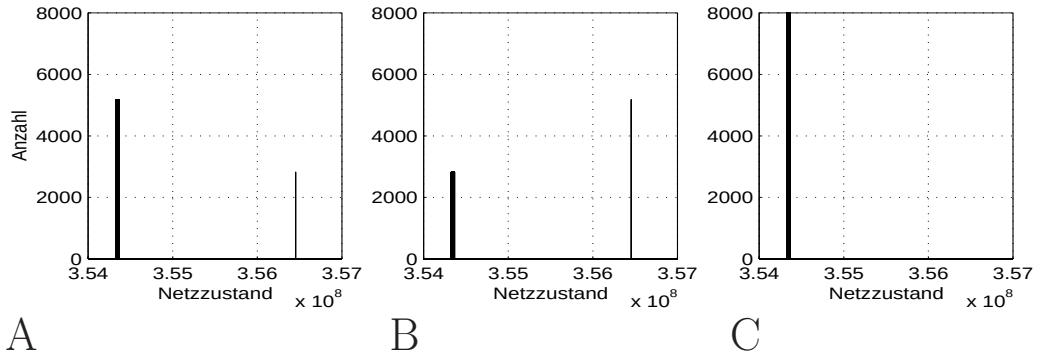


Abbildung 4.6: HOPFIELD- *Netz aus 32 Neuronen. Gewichtsmatrix aus den fundamentalen Mustern (4.2) nach* HEBB. *Erreichbare stabile Zustände von Anfangszustand A: (4.7), B: (4.8) und C: (4.9) von je 8 000 Trajektorien durch zufällige Abfrageblöcke. Fett: fundamentales Muster* $\xi^{(\mathbf{3})}$

Man kann von den drei Anfangszuständen (4.7 - 4.9) Trajektorien- Bäume mit zufälliger Abfragereihenfolge bilden. Die erreichbaren stabilen Zustände sind in Abb. 4.6 dargestellt. Dabei wird manchmal ein weiterer spurious state

$$\mathbf{x_{sp_3,dec}} = 356\ 453\ 980 \tag{4.10}$$

erreicht. Hier wird der erste Unterschied zwischen einer zufälligen Auswahl von Permutationen der sequentiellen Reihenfolge und von zufälligen Abfragereihenfolgen deutlich. Während man vom Anfangszustand (4.9) z. B. mit der Abfragefolge nach (4.6) zum spurious state $\mathbf{x_{sp_3,dec}}$ (4.10) kommt, wurde dieser Zustand mit zufälligen Abfrageblöcken nicht erreicht, d. h. mit Abfrageblöcken, in denen Neuronen mehrfach abgefragt werden können.

(3) Wenn man Anfangszustände mit $k = 3$ Hamming Abständen von $\xi^{(\mathbf{3})}$ untersucht, so wurden von den $\binom{N}{k} = 4\ 960$ verschiedenen Anfangszuständen jeweils bis zu 20 zufällig gewählte Trajektorien verwendet. Die stabilen Zustände, in die das Netz gestrebt ist, wurden verglichen. Dabei stellte sich heraus, daß 387 Anfangszustände nicht nur in $\xi^{(\mathbf{3})}$, sondern auch in andere stabile Zustände streben können. Diese stabilen Zustände waren das spurious state $\mathbf{x_{sp_2,dec}}$ (4.10), noch weitere 5 spurious states und das fundamentale Muster $-\xi^{(\mathbf{1})}$. Diese 387 Anfangszustände sind also nicht- zuordenbar. Die restlichen 4 573 Anfangszustände erreichten $\xi^{(\mathbf{3})}$. Dabei ist allerdings nicht ausgeschlossen, daß bei einer Erhöhung der Anzahl der Abfrageblockfolgen von einem

Teil dieser Zustände auch andere stabile Zustände hätten erreicht werden können.

(4) Bei $\binom{N}{k} = 35\,960$ Zuständen, die $k = 4$ Hamming Abstand vom fundamentalen Muster $\xi^{(\mathbf{3})}$ entfernt sind, konnten sehr wenige Zustände gefunden werden, die im Attraktionsbecken von $\xi^{(\mathbf{3})}$ liegen. Vielmehr waren noch weitere fundamentale Muster wie $\xi^{(\mathbf{1})}$ und $\xi^{(\mathbf{4})}$ von diesen Zuständen aus erreichbar.

(5) Unter den $\binom{N}{k} = 201\,376$ möglichen Anfangszuständen mit Hamming Abstand $k = 5$ ist einer der spurious state $\mathbf{x}_{\mathbf{sp_3,dec}}$ (4.10). Es gibt in dieser Entfernung noch einen weiteren spurious state, einen Zustand, der in dem Attraktionsbecken dieses weiteren spurious states liegt und zwei weitere, die ausschließlich in 2 bzw. 3 verschiedene spurious states streben können. Andere Anfangszustände bildeten bei bis zu 20 untersuchten Abfrageblockfolgen mindestens eine davon zu einem fundamentalen Muster.

Auf die gleiche Weise, wie für $\xi^{(\mathbf{3})}$ ausführlich dargestellt, wurden nun die Attraktionsbecken der anderen fundamentalen Muster und der gefundenen spurious states untersucht. Dabei wurden nur von den insgesamt 5 488 Anfangszuständen ausgegangen, die bis zu 3 Hamming Abstände vom jeweiligen stabilen Zustand entfernt waren (bei $\xi^{(\mathbf{3})}$ waren ja schon bei 2 Hamming Abstand nicht- zuordenbare Zustände aufgetreten). Von jedem dieser Zustände wurden 20 Abfrageblockfolgen untersucht.
Abb. 4.7 und Tab. 4.2 zeigen die Ergebnisse. Es zeigte sich, daß die Attraktionsbecken der spurious states kleiner und in fast allen Fällen deutlich kleiner waren als die der fundamentalen Muster. Nur zwei spurious states hatten Attraktionsbecken, die fast so groß waren wie die kleinsten der fundamentalen Muster.

Will man die Gesamtanzahl der Zustände in Attraktionsbecken und der nichtzuordenbaren Zustände ermitteln, ist man hier auf Nährungen angewiesen. Nimmt man die Zahlen der Tabelle 4.2, kommt man zu einer Näherung der Anzahl der Zustände in Attraktionsbecken aus den folgenden beiden Gründen. Die Zahlen sind möglicherweise zu groß, weil mit einer Erhöhung der Abfrageblockfolgen der eine oder andere Zustand auch noch zu einem anderen stabilen Zustand gestrebt wäre. Sie sind aber möglicherweise auch zu

Stabiler Zustand	Spezifikation	Größe der Attraktionsbecken
13 185 683		363
15 545 496	$\xi^{(5)}$ (4.2)	988
65 336 476		0
147 403 715		17
149 451 144		69
164 131 779	$-\xi^{(6)}$ (4.2)	4 375
166 228 936		40
166 491 336		770
183 169 416		1
199 685 260		25
200 209 868		3
220 881 735		837
229 011 020	$-\xi^{(1)}$ (4.2)	4 548
255 257 676	$\mathbf{x_{sp_1,dec}}$ (4.4)	1 404
255 780 172	$-\xi^{(4)}$ (4.2)	1 588
354 225 751	$\xi^{(3)}$ (4.2)	5 098
356 453 980	$\mathbf{x_{sp_3,dec}}$ (4.10)	126
522 112 839		93
1 122 301 080		102
1 355 095 192		143
1 389 730 972	$\mathbf{x_{sp_2,dec}}$ (4.5)	338
1 406 459 036	$-\xi^{(2)}$ (4.2)	4 520

Tabelle 4.2: HOPFIELD- *Netz aus 32 Neuronen. Gewichtsmatrix für die Muster (4.2) nach* HEBB. *Größe der Attraktionsbecken (Spalte 2) der fundamentalen Muster und der spurious states (Spalte 1). Es ist nur die 1. Hälfte der stabilen Zustände dargestellt, die 2. Hälfte ist symmetrisch dazu*

klein, weil manche Attraktionsbecken auch bis zu einem Hamming Abstand von 4, 5 oder größer reichen.
Unter der relativ wahrscheinlichen Vorraussetzung, daß alle stabilen Zustände gefunden werden, ergeben sich 44 ($1,02 \cdot 10^{-6}\%$) stabile Zustände und näherungsweise 58 896 (0,0014%) Zustände in Attraktionsbecken. Es verbleiben $4,295 \cdot 10^{9}$ (99,9%) nicht- zuordenbare Zustände.

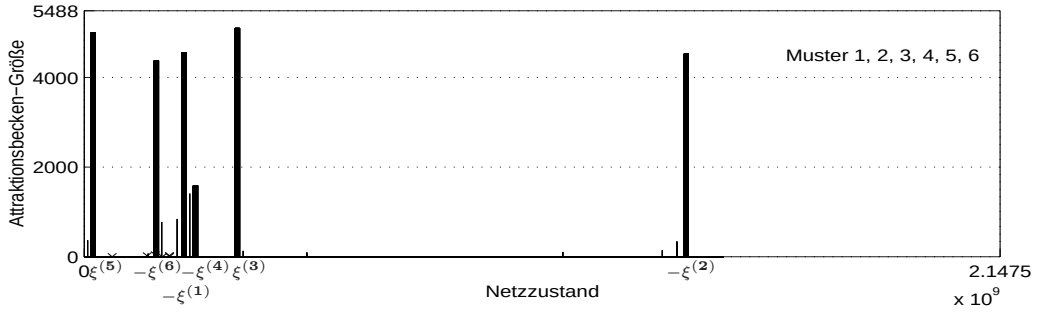


Abbildung 4.7: HOPFIELD- *Netz aus 32 Neuronen. Oben rechts: eingespeicherte fundamentale Muster (s. 4.2). Gewichtsmatrix nach* HEBB. *Vergleich der Größen der Attraktionsbecken bis zu 3 Hamming Abstand vom jeweiligen fundamentalen Muster (fett) bzw. von den gefundenen spurious states. Es ist nur die 1. Hälfte der Netzzustände dargestellt, die 2. Hälfte ist symmetrisch dazu. Kreuz: Attraktionsbecken ist kleiner als 50*

4.3 HOPFIELD- Netz aus 32 Neuronen mit unterschiedlicher Anzahl fundamentaler Muster nach HEBB

Nimmt man die fundamentalen Muster aus Abschnitt 4.2, und kombiniert sie in unterschiedlicher Weise, so kann man von ihnen ingesamt

$$\sum_{i=2}^{5} \binom{6}{i} = 62 \tag{4.11}$$

verschiedene Netzwerke bilden. Dabei werden jeweils mindestens eines der fundamentalen Muster eingespeichert.
In jedem Netz sollen alle stabilen Zustände ermittelt und deren Attraktionsbecken bestimmt werden. Dabei werden von mehreren Zuständen hoher Energie 12 000 Trajektorien zu stabilen Zuständen gebildet. Sind so die stabilen Zustände festgestellt, werden von ihnen die Attraktionsbecken ermittelt, indem von Anfangszuständen bis zu 3 Hamming Abstand jeweils 20 Abfrageblockfolgen auf die in Abschnitt 4.2 geschilderte Weise gebildet werden.
Die folgenden Histogramme in Abb. 4.8 bis Abb. 4.18 zeigen von jedem Netz die gefundenen stabilen Zustände und die Größe derer Attraktionsbecken.

In der Tab. 4.3 sind dazu die Gesamtanzahl der stabilen Zustände, die Gesamtzahl der Zustände in Attraktionsbecken und die Gesamtzahl nicht- zuordenbarer Zustände angegeben.

Der Vergleich der Abb. 4.8 bis Abb. 4.18 zeigt, daß die Attraktionsbecken der vorhandenen spurious states stets kleiner sind als die der fundamentalen Muster. Es kann auch vorkommen, daß spurious states keine Attraktionsbecken haben.
In allen Fällen, in denen die Muster $\xi^{(1)}$ und $\xi^{(4)}$ (nur 8 Hamming Abstände voneinander entfernt) gleichzeitig eingespeichert wurden, verkleinerten sich deren Attraktionsbecken.
Die Anzahl der spurios states kann für $p = 3$ eingespeicherte Muster zwischen 2, 4 und 8 variieren, für $p = 4$ sogar zwischen 4 und 52. Spurious states traten also erst oberhalb von 2 eingespeicherten Mustern auf. Ihre Anzahl war aber dann nicht mit der Anzahl der eingespeicherten Muster korreliert. Die Größe der Attraktionsbecken der spurious states nahm mit steigender Zahl der stabilen Zustände ab und konnte auf Null sinken.
Ferner gibt es 58 Netze, die symmetrische stabile Zustände und entsprechende Attraktionsbecken aufweisen, 4 Netze (Abb. 4.17) weisen dies jedoch nicht auf. 2 dieser Netze haben zwar symmetrisch liegende spurious states, doch mit jeweils zum Teil unterschiedlichen Größen der Attraktionsbecken (s. auch Tab. 3.7), 2 weitere besitzen unsymmetrisch liegende spurious states (s. auch Tab. 3.8 für unsymmetrische residuelle Fehler).

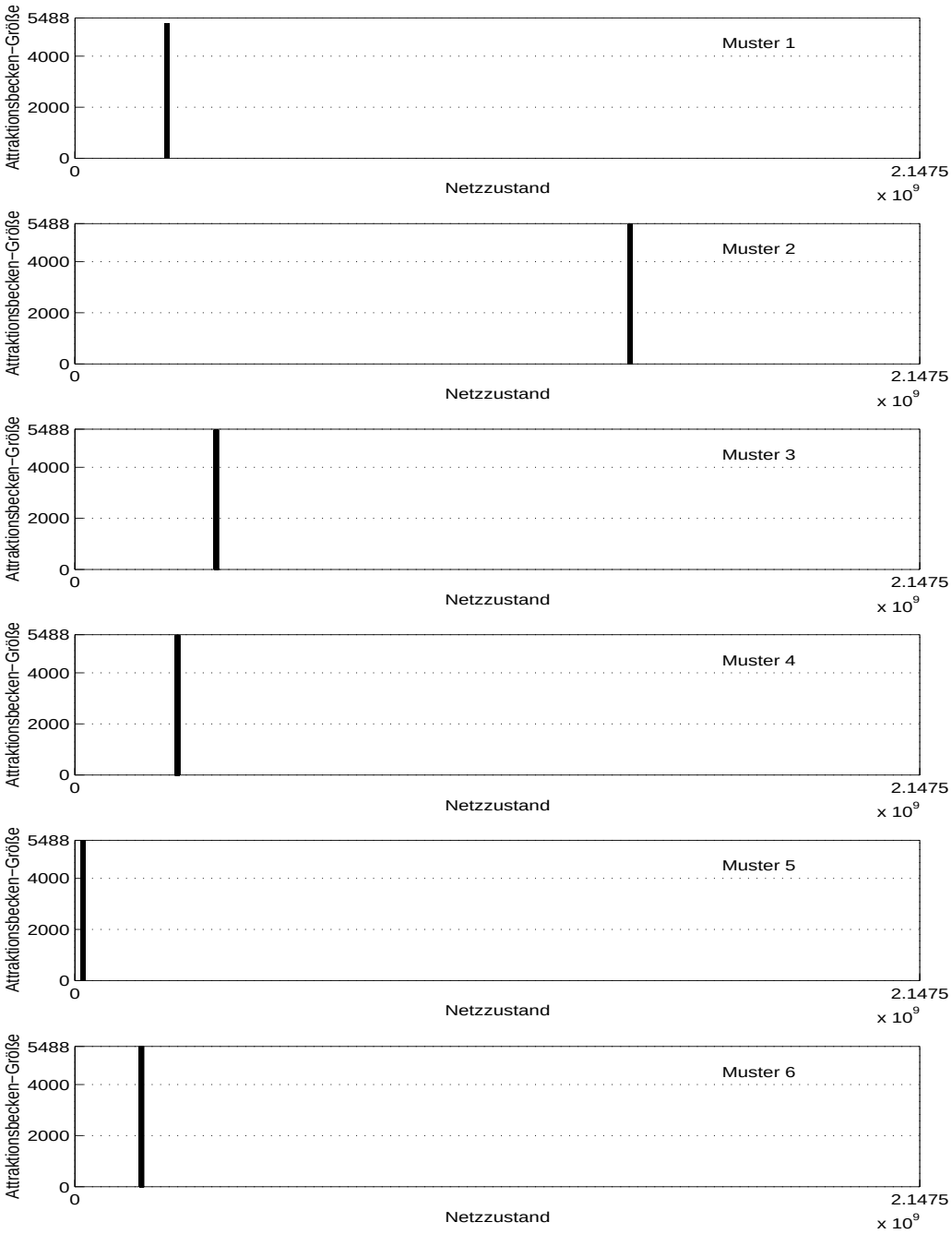


Abbildung 4.8: HOPFIELD- *Netz aus 32 Neuronen mit jeweils nur einem eingespeicherten Muster. Oben rechts: fundamentales Muster aus (4.2), aus dem die Gewichtsmatrix nach* HEBB *berechnet wurde. Größe des Attraktionsbeckens bis zu 3 Hamming Abstand vom fundamentalen Muster (fett). Es ist nur die 1. Hälfte der Netzzustände dargestellt, die 2. Hälfte ist symmetrisch dazu*

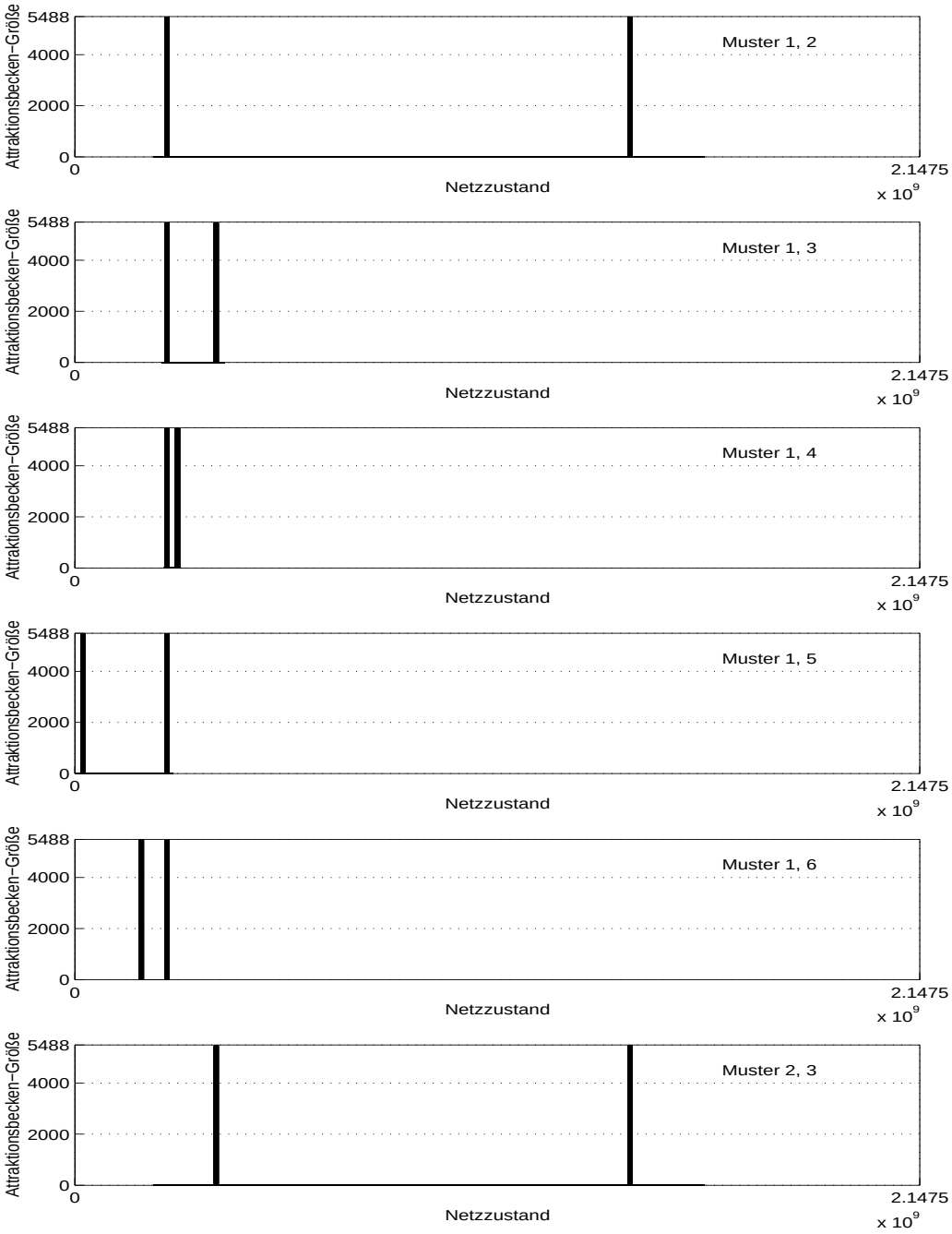


Abbildung 4.9: HOPFIELD- *Netz aus 32 Neuronen mit zwei eingespeicherten Mustern. Oben rechts: fundamentale Muster aus (4.2), aus denen die Gewichtsmatrix nach* HEBB *berechnet wurde. Größe der Attraktionsbecken bis zu 3 Hamming Abstand vom fundamentalen Muster (fett). Es ist nur die 1. Hälfte der Netzzustände dargestellt, die 2. Hälfte ist symmetrisch dazu*

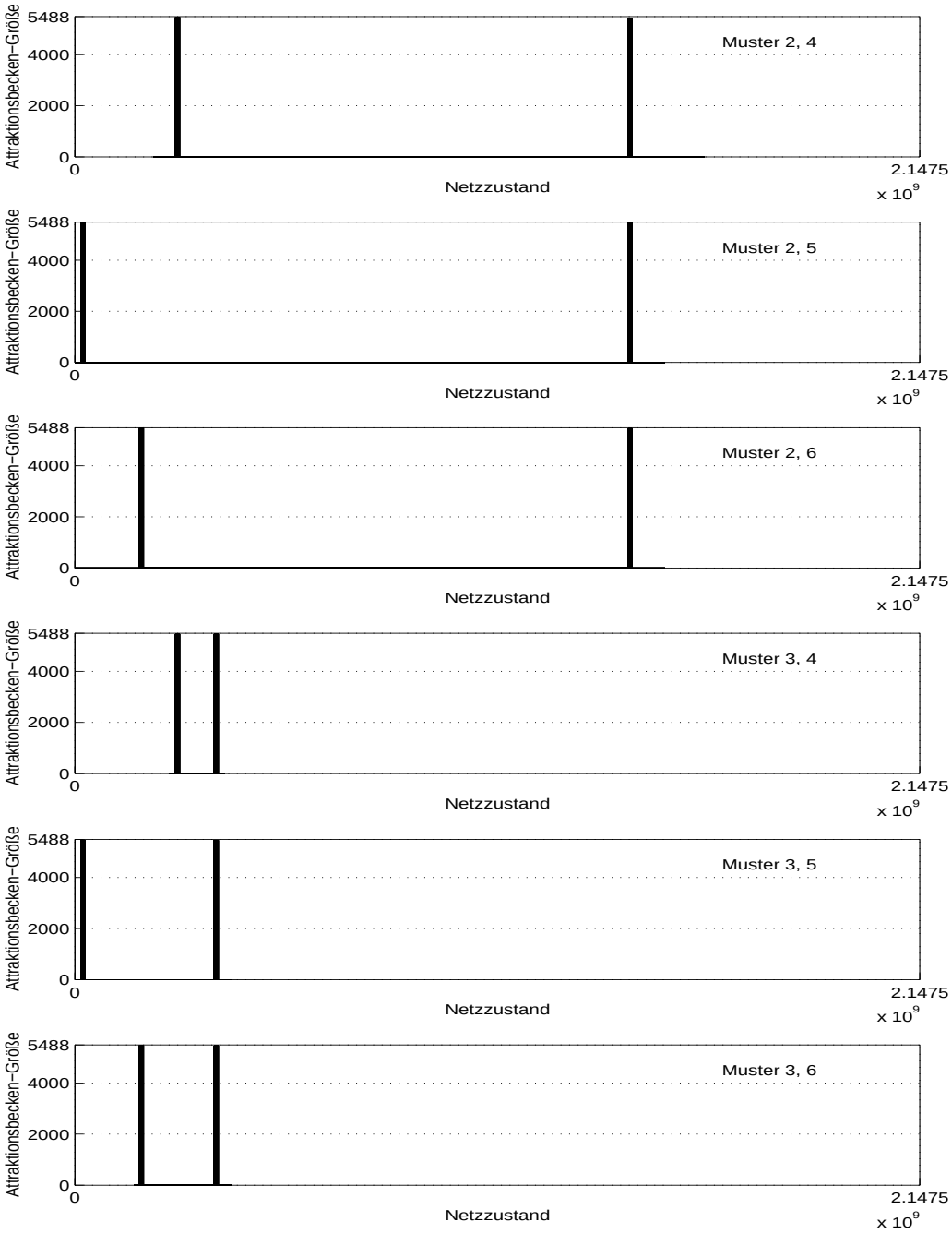


Abbildung 4.10: HOPFIELD- *Netz aus 32 Neuronen mit zwei eingespeicherten Mustern. Oben rechts: fundamentale Muster aus (4.2), aus denen die Gewichtsmatrix nach* HEBB *berechnet wurde. Größe der Attraktionsbecken bis zu 3 Hamming Abstand vom fundamentalen Muster (fett). Es ist nur die 1. Hälfte der Netzzustände dargestellt, die 2. Hälfte ist symmetrisch dazu*

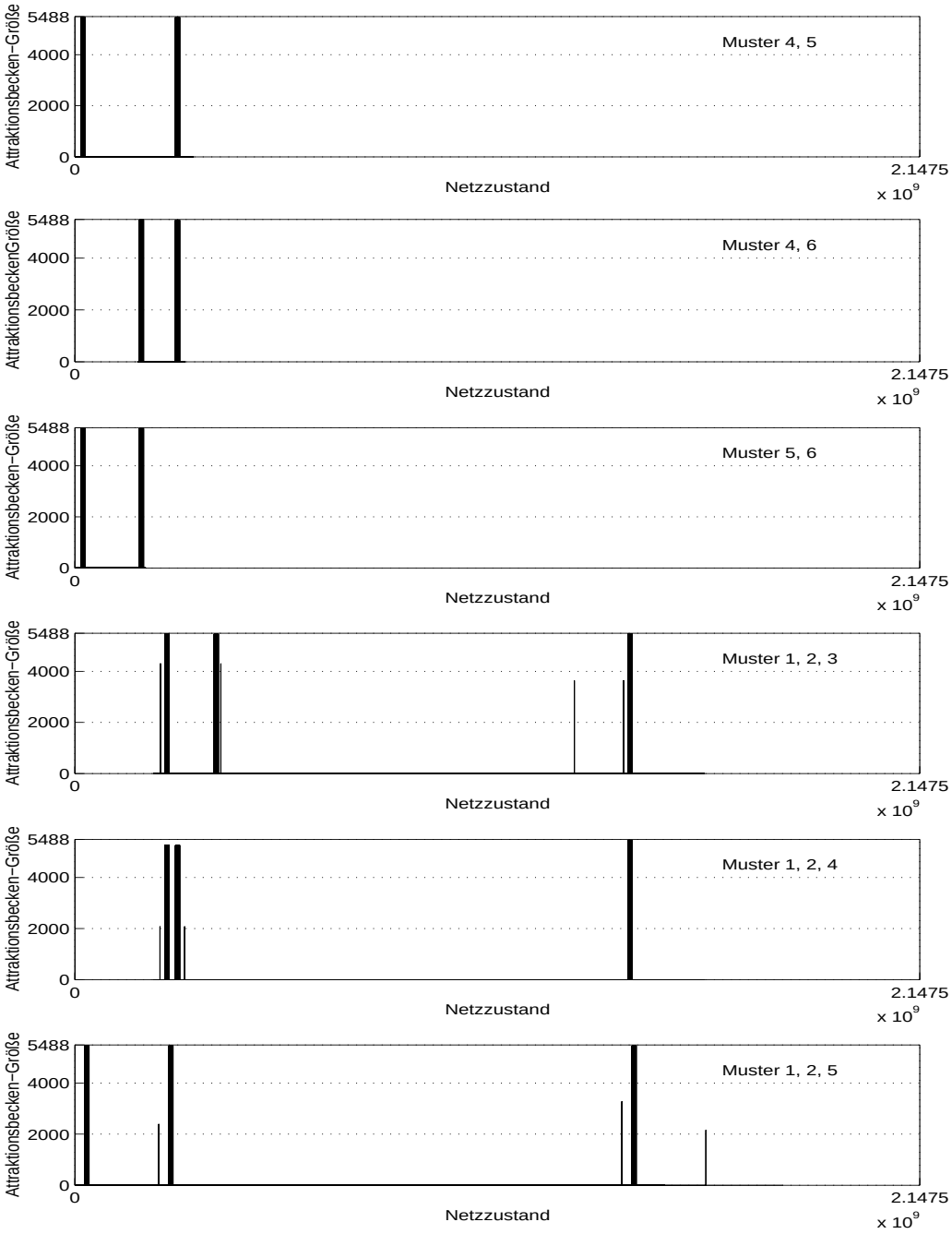


Abbildung 4.11: HOPFIELD- *Netz aus 32 Neuronen mit 2 bzw. 3 eingespeicherten Mustern. Oben rechts: fundamentale Muster aus (4.2), aus denen die Gewichtsmatrix nach* HEBB *berechnet wurde. Größe der Attraktionsbecken bis zu 3 Hamming Abstand vom fundamentalen Muster (fett) und vom gefundenen spurious state. Es ist nur die 1. Hälfte der Netzzustände dargestellt*

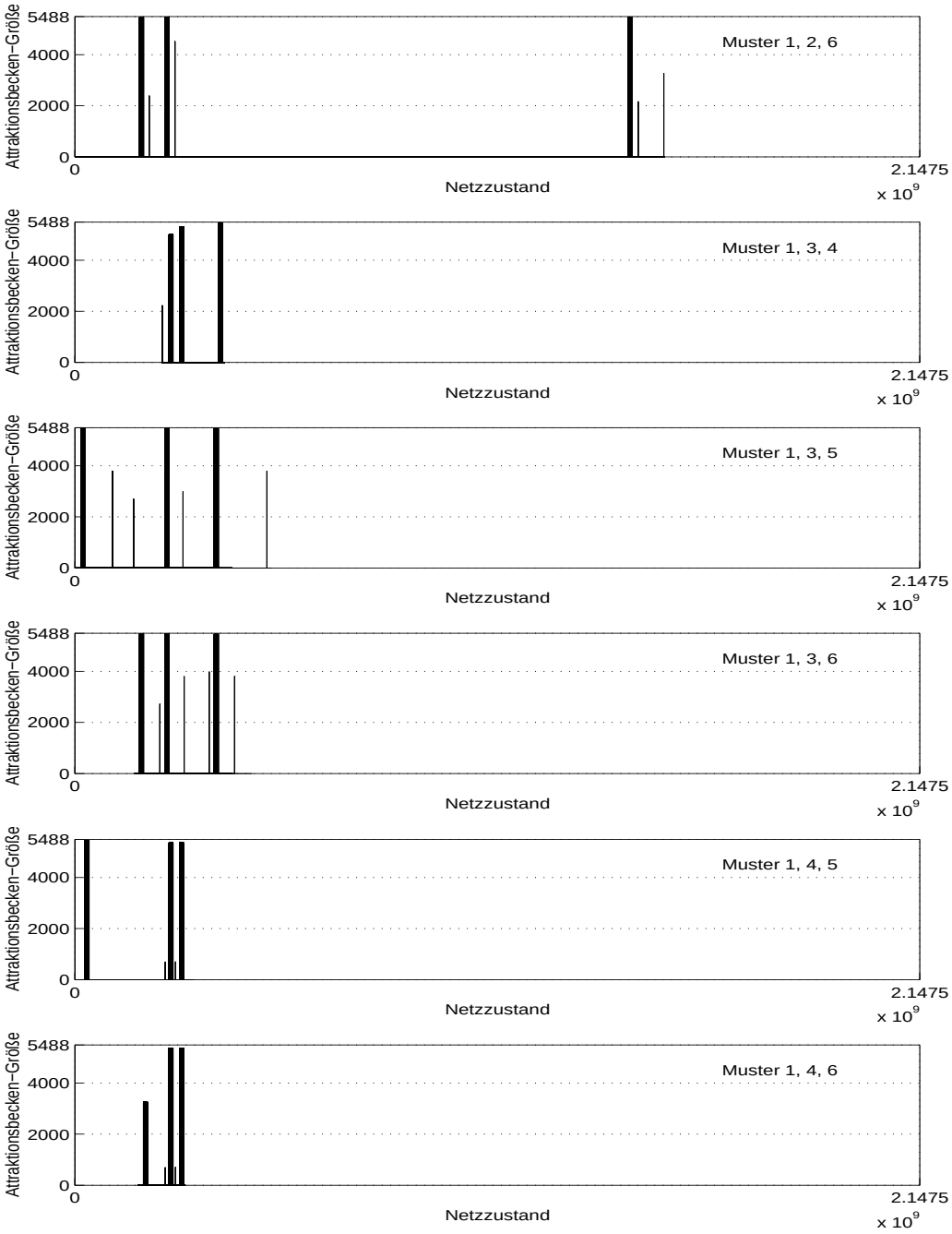


Abbildung 4.12: HOPFIELD- *Netz aus 32 Neuronen mit 3 eingespeicherten Mustern. Oben rechts: fundamentale Muster aus (4.2), aus denen die Gewichtsmatrix nach* HEBB *berechnet wurde. Größe der Attraktionsbecken bis zu 3 Hamming Abstand vom fundamentalen Muster (fett) und vom gefundenen spurious state. Es ist nur die 1. Hälfte der Netzzustände dargestellt*

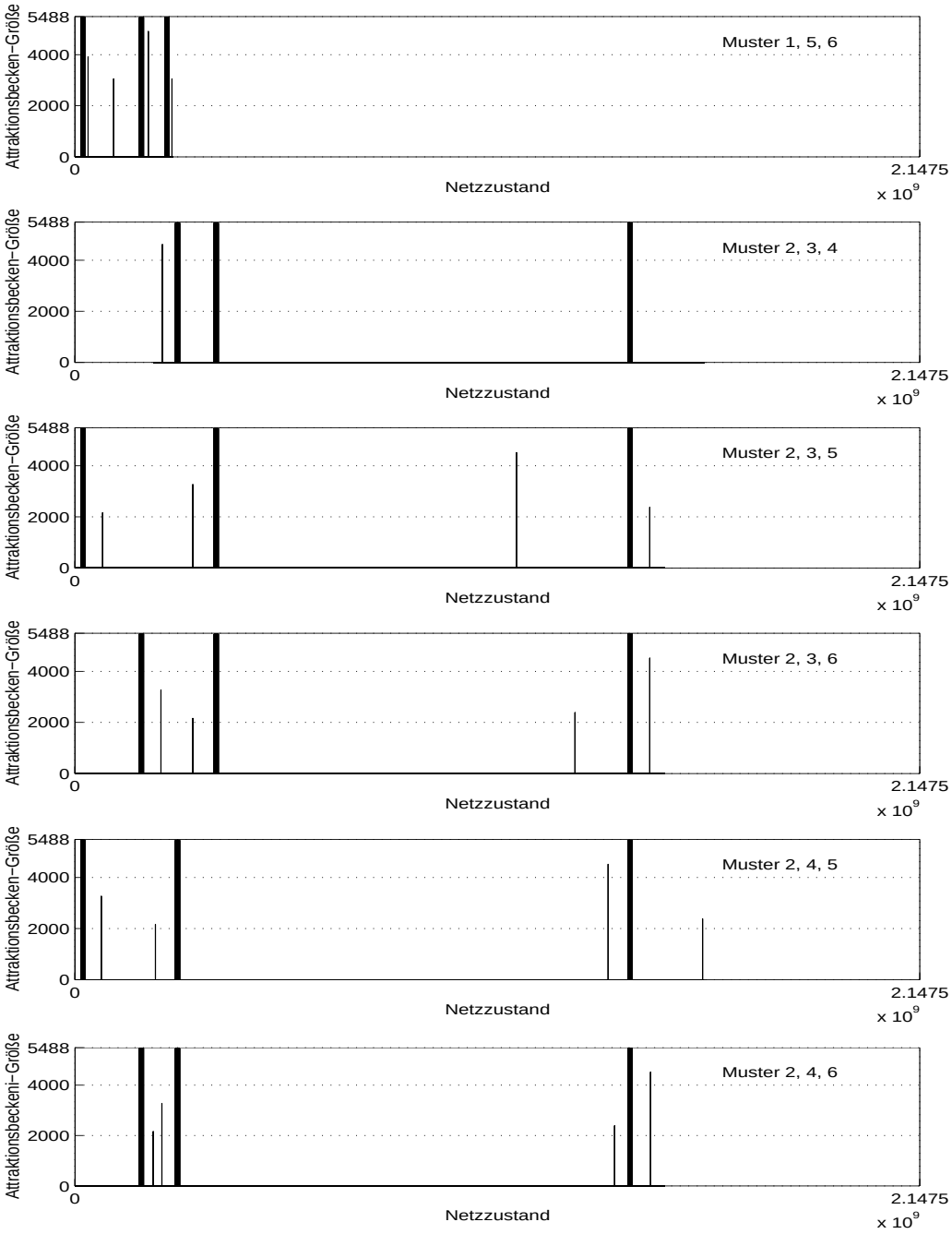


Abbildung 4.13: HOPFIELD- *Netz aus 32 Neuronen mit 3 eingespeicherten Mustern. Oben rechts: fundamentale Muster aus (4.2), aus denen die Gewichtsmatrix nach* HEBB *berechnet wurde. Größe der Attraktionsbecken bis zu 3 Hamming Abstand vom fundamentalen Muster (fett) und vom gefundenen spurious state. Es ist nur die 1. Hälfte der Netzzustände dargestellt*

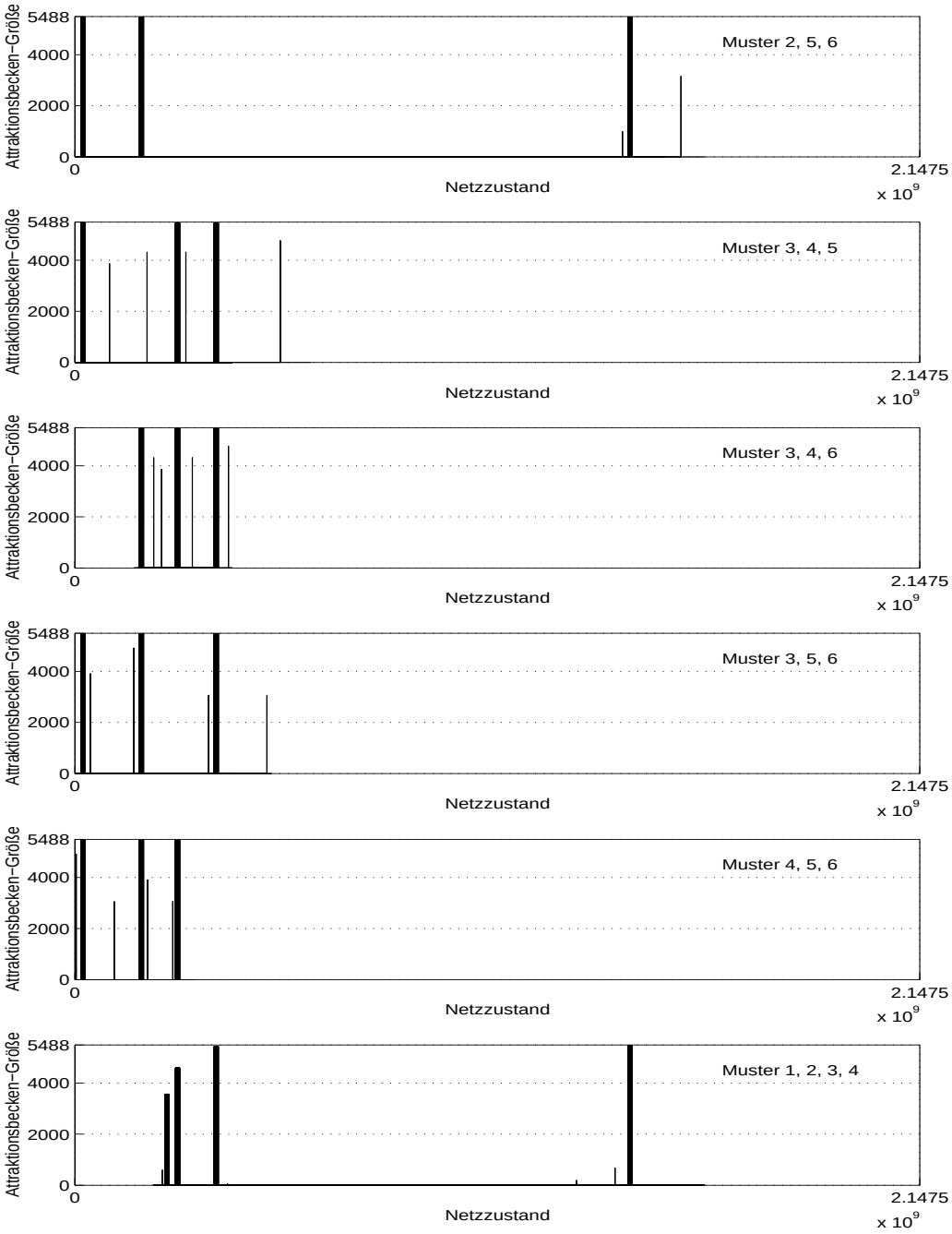


Abbildung 4.14: HOPFIELD- *Netz aus 32 Neuronen mit 3 bzw. 4 eingespeicherten Mustern. Oben rechts: fundamentale Muster aus (4.2), aus denen die Gewichtsmatrix nach* HEBB *berechnet wurde. Größe der Attraktionsbecken bis zu 3 Hamming Abstand vom fundamentalen Muster (fett) und von den gefundenen spurious states. Es ist nur die 1. Hälfte der Netzzustände dargestellt*

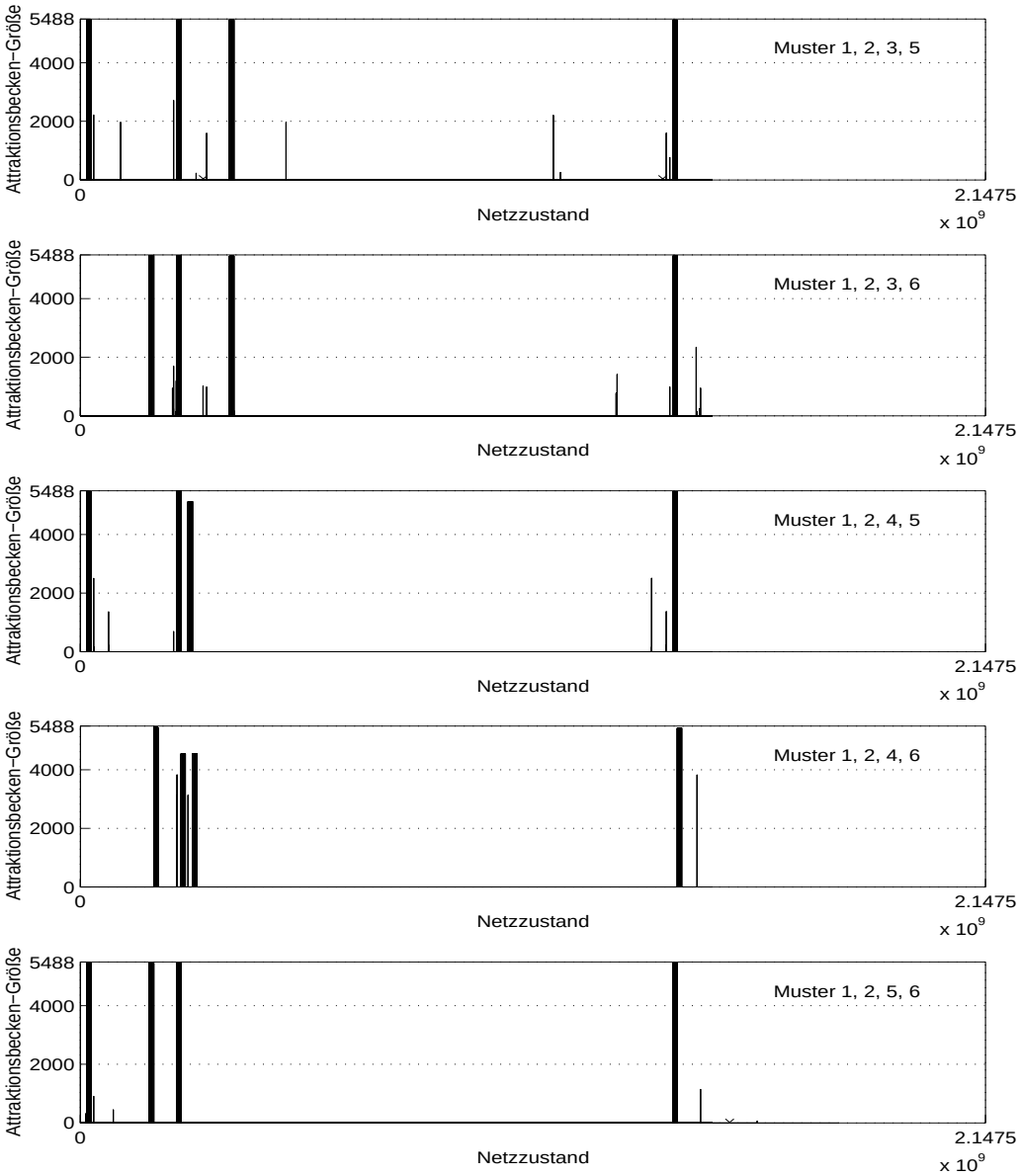


Abbildung 4.15: HOPFIELD- *Netz aus 32 Neuronen mit 4 eingespeicherten Mustern. Oben rechts: fundamentale Muster aus (4.2), aus denen die Gewichtsmatrix nach* HEBB *berechnet wurde. Größe der Attraktionsbecken bis zu 3 Hamming Abstand vom fundamentalen Muster (fett) und von den gefundenen spurious states. Es ist nur die 1. Hälfte der Netzzustände dargestellt*

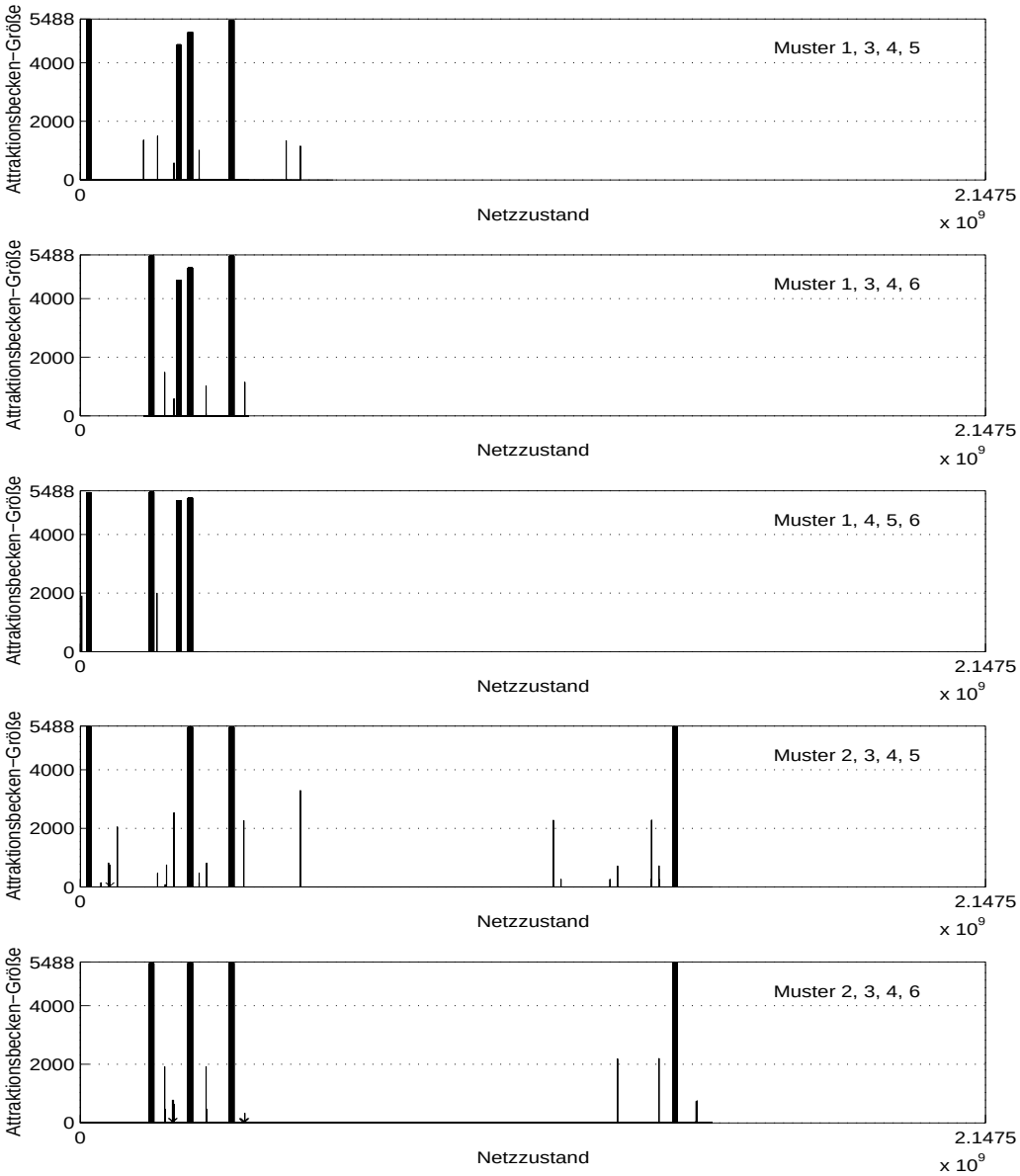


Abbildung 4.16: HOPFIELD- *Netz aus 32 Neuronen mit 4 eingespeicherten Mustern. Oben rechts: fundamentale Muster aus (4.2), aus denen die Gewichtsmatrix nach* HEBB *berechnet wurde. Größe der Attraktionsbecken bis zu 3 Hamming Abstand vom fundamentalen Muster (fett) und von den gefundenen spurious states. Es ist nur die 1. Hälfte der Netzzustände dargestellt*

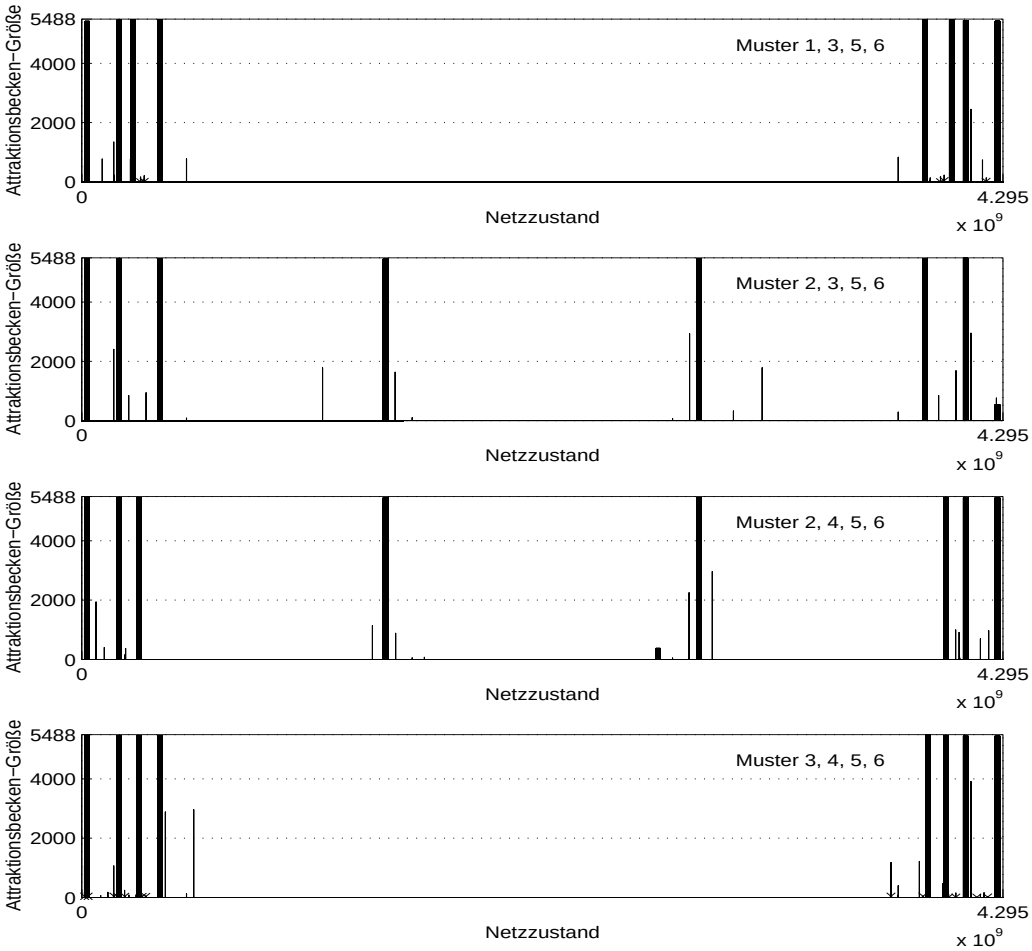


Abbildung 4.17: HOPFIELD- *Netz aus 32 Neuronen mit 4 eingespeicherten Mustern. Oben rechts: fundamentale Muster aus (4.2), aus denen die Gewichtsmatrix nach* HEBB *berechnet wurde. Größe der Attraktionsbecken bis zu 3 Hamming Abstand vom fundamentalen Muster (fett) und von den gefundenen spurious states. Man beachte dabei, daß die Netze nicht symmetrisch sind*

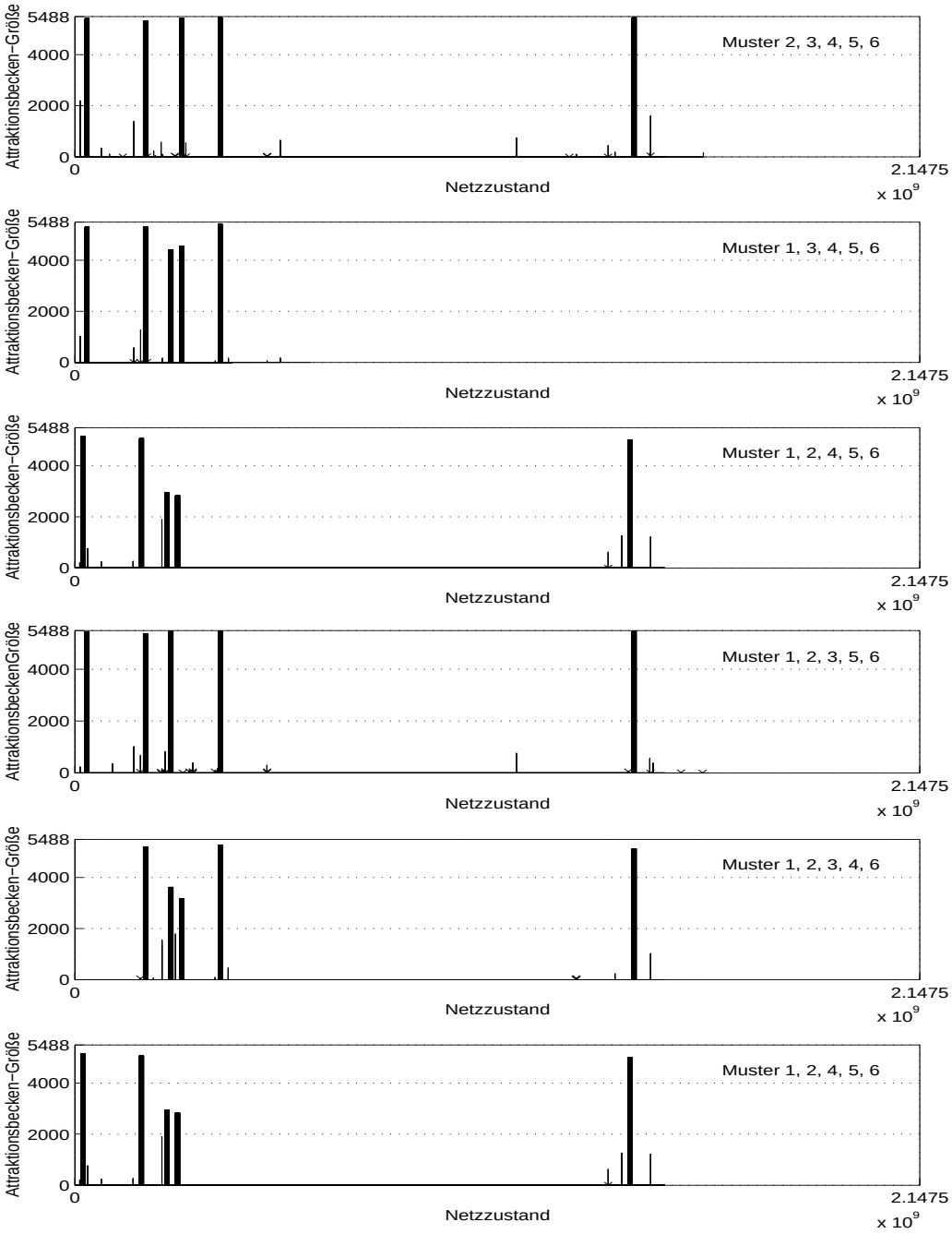


Abbildung 4.18: HOPFIELD- *Netz aus 32 Neuronen mit 5 eingespeicherten Mustern. Oben rechts: fundamentale Muster aus (4.2), aus denen die Gewichtsmatrix nach* HEBB *berechnet wurde. Größe der Attraktionsbecken bis zu 3 Hamming Abstand vom fundamentalen Muster (fett) und von den gefundenen spurious states. Es ist nur die 1. Hälfte der Netzzustände dargestellt*

Eingespeicherte, fundamentale Muster aus (4.2)	Gesamtzahl der stabilen Zustände	Gesamtzahl der Zustände in Attraktionsbecken	Gesamtzahl nicht- zuordenbarer Zustände
1	2 ($4,66 \cdot 10^{-8}$%)	10 528 (0,0002%)	$4,295 \cdot 10^{9}$ (99,9%)
2	2 ($4,66 \cdot 10^{-8}$%)	10 912 (0,0003%)	$4,295 \cdot 10^{9}$ (99,9%)
3	2 ($4,66 \cdot 10^{-8}$%)	10 916 (0,0003%)	$4,295 \cdot 10^{9}$ (99,9%)
4	2 ($4,66 \cdot 10^{-8}$%)	10 924 (0,0003%)	$4,295 \cdot 10^{9}$ (99,9%)
5	2 ($4,66 \cdot 10^{-8}$%)	10 936 (0,0003%)	$4,295 \cdot 10^{9}$ (99,9%)
6	2 ($4,66 \cdot 10^{-8}$%)	10 974 (0,0003%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2	4 ($9,31 \cdot 10^{-8}$%)	21 826 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 3	4 ($9,31 \cdot 10^{-8}$%)	21 824 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 4	4 ($9,31 \cdot 10^{-8}$%)	21 824 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 5	4 ($9,31 \cdot 10^{-8}$%)	21 824 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 6	4 ($9,31 \cdot 10^{-8}$%)	21 824 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 3	4 ($9,31 \cdot 10^{-8}$%)	21 826 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 4	4 ($9,31 \cdot 10^{-8}$%)	21 846 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 5	4 ($9,31 \cdot 10^{-8}$%)	21 826 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 6	4 ($9,31 \cdot 10^{-8}$%)	21 798 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
3, 4	4 ($9,31 \cdot 10^{-8}$%)	21 826 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
3, 5	4 ($9,31 \cdot 10^{-8}$%)	21 828 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
3, 6	4 ($9,31 \cdot 10^{-8}$%)	21 774 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
4, 5	4 ($9,31 \cdot 10^{-8}$%)	21 828 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
4, 6	4 ($9,31 \cdot 10^{-8}$%)	21 858 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
5, 6	4 ($9,31 \cdot 10^{-8}$%)	21 824 (0,0005%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 3	10 ($2,33 \cdot 10^{-7}$%)	46 292 (0,0011%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 4	10 ($2,33 \cdot 10^{-7}$%)	40 314 (0,0009%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 5	14 ($3,26 \cdot 10^{-7}$%)	57 404 (0,0013%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 6	10 ($2,33 \cdot 10^{-7}$%)	57 412 (0,0013%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 3, 4	8 ($1,86 \cdot 10^{-7}$%)	35 982 (0,0008%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 3, 5	14 ($3,26 \cdot 10^{-7}$%)	63 504 (0,0015%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 3, 6	14 ($3,26 \cdot 10^{-7}$%)	61 243 (0,0014%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 4, 5	10 ($2,33 \cdot 10^{-7}$%)	35 132 (0,0008%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 4, 6	10 ($2,33 \cdot 10^{-7}$%)	35 136 (0,0008%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 5, 6	14 ($3,26 \cdot 10^{-7}$%)	62 598 (0,0015%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 3, 4	14 ($3,26 \cdot 10^{-7}$%)	69 664 (0,0016%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 3, 5	14 ($3,26 \cdot 10^{-7}$%)	57 410 (0,0013%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 3, 6	14 ($3,26 \cdot 10^{-7}$%)	57 418 (0,0013%)	$4,295 \cdot 10^{9}$ (99,9%)

Fortsetzung der Tabelle auf der nächsten Seite

Fortsetzung der Tabelle der vorigen Seite			
Eingespeicherte, fundamentale Muster aus (4.2)	Gesamtzahl der stabilen Zustände	Gesamtzahl der Zustände in Attraktionsbecken	Gesamtzahl nicht- zuordenbarer Zustände
2, 4, 5	14 ($3,26 \cdot 10^{-7}$%)	57 388 (0,0013%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 4, 6	14 ($3,26 \cdot 10^{-7}$%)	57 422 (0,0013%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 5, 6	10 ($2,33 \cdot 10^{-7}$%)	41 016 (0,0009%)	$4,295 \cdot 10^{9}$ (99,9%)
3, 4, 5	14 ($3,26 \cdot 10^{-7}$%)	67 258 (0,0016%)	$4,295 \cdot 10^{9}$ (99,9%)
3, 4, 6	14 ($3,26 \cdot 10^{-7}$%)	67 272 (0,0016%)	$4,295 \cdot 10^{9}$ (99,9%)
3, 5, 6	14 ($3,26 \cdot 10^{-7}$%)	62 596 (0,0015%)	$4,295 \cdot 10^{9}$ (99,9%)
4, 5, 6	14 ($3,26 \cdot 10^{-7}$%)	62 602 (0,0015%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 3, 4	28 ($6,52 \cdot 10^{-7}$%)	43 064 (0,0010%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 3, 5	34 ($7,92 \cdot 10^{-7}$%)	80 292 (0,0019%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 3, 6	60 ($1,40 \cdot 10^{-6}$%)	88 924 (0,0021%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 4, 5	28 ($6,52 \cdot 10^{-7}$%)	62 124 (0,0014%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 4, 6	16 ($3,72 \cdot 10^{-7}$%)	67 638 (0,0016%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 5, 6	28 ($6,52 \cdot 10^{-7}$%)	29 266 (0,0006%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 3, 4, 5	22 ($5,12 \cdot 10^{-7}$%)	55 130 (0,0013%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 3, 4, 6	22 ($5,12 \cdot 10^{-7}$%)	55 208 (0,0013%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 3, 5, 6	40 ($9,31 \cdot 10^{-7}$%)	63 011 (0,0015%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 4, 5, 6	12 ($2,79 \cdot 10^{-7}$%)	50 316 (0,0012%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 3, 4, 5	54 ($1,26 \cdot 10^{-6}$%)	87 092 (0,0020%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 3, 4, 6	52 ($1,21 \cdot 10^{-6}$%)	72 286 (0,0017%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 3, 5, 6	27 ($6,29 \cdot 10^{-7}$%)	63 972 (0,0015%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 4, 5, 6	30 ($6,98 \cdot 10^{-7}$%)	54 945 (0,0013%)	$4,295 \cdot 10^{9}$ (99,9%)
3, 4, 5, 6	52 ($1,21 \cdot 10^{-6}$%)	40 414 (0,0009%)	$4,295 \cdot 10^{9}$ (99,9%)
2, 3, 4, 5, 6	72 ($1,68 \cdot 10^{-6}$%)	74 160 (0,0017%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 3, 4, 5, 6	16 ($3,73 \cdot 10^{-7}$%)	56 974 (0,0013%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 4, 5, 6	36 ($8,38 \cdot 10^{-7}$%)	68 800 (0,0016%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 3, 5, 6	76 ($1,77 \cdot 10^{-6}$%)	68 134 (0,0016%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 3, 4, 6	34 ($7,92 \cdot 10^{-7}$%)	58 184 (0,0014%)	$4,295 \cdot 10^{9}$ (99,9%)
1, 2, 3, 4, 5	48 ($1,12 \cdot 10^{-6}$%)	56 418 (0,0013%)	$4,295 \cdot 10^{9}$ (99,9%)

Tabelle 4.3: HOPFIELD- *Netz aus 32 Neuronen mit jeweils unterschiedlicher Anzahl gespeicherter Muster. Spalte 1: fundamentale Muster aus (4.2), aus denen die Gewichtsmatrix nach* HEBB *berechnet wurde. Gesamtzahlen der stabilen Zustände (Spalte 2), der Zustände in Attraktionsbecken (Spalte 3) und der nicht- zuordenbaren Zustände (Spalte 4)*

Kapitel 5

Vergleich von Netzwerken, in denen die gleichen Muster nach HEBB bzw. nach der Projektionsregel gespeichert wurden

5.1 Untersuchung an einem kleinen Netzwerk

Netzwerke, in denen die Muster mit der Projektionsregel (s. Abschnitt 2.6.2) gespeichert wurden, lassen sich auf die gleiche Weise untersuchen wie Netzwerke, die mit der Lernregel nach HEBB behandelt wurden. Es war mit der Lernregel nach HEBB z. B. nicht möglich, die fundamentalen Muster (3.17) in einem HOPFIELD- Netz mit 8 Neuronen zu speichern (s. Abschnitt 3.2 und Tab. 3.10). Mit der Projektionsregel ist dies jedoch möglich. So bekommt man für diese fundamentalen Muster (3.17) die Gewichtsmatrix

$$w = \begin{pmatrix} 0 & 0.18 & 0.41 & -0.05 & -0.18 & -0.05 & -0.05 & 0.05 \\ 0.018 & 0 & 0.18 & 0.09 & 0.36 & 0.09 & 0.09 & -0.09 \\ 0.41 & 0.18 & 0 & -0.05 & -0.18 & -0.02 & -0.05 & 0.05 \\ -0.05 & 0.09 & -0.05 & 0 & -0.09 & 0.23 & 0.23 & -0.23 \\ -0.18 & 0.36 & -0.18 & -0.09 & 0 & -0.09 & -0.09 & 0.09 \\ -0.05 & 0.09 & -0.02 & 0.23 & -0.09 & 0 & 0.23 & -0.23 \\ -0.05 & 0.09 & -0.05 & 0.23 & -0.09 & 0.23 & 0 & -0.23 \\ 0.05 & -0.09 & 0.05 & -0.23 & 0.09 & -0.23 & -0.23 & 0 \end{pmatrix}. \tag{5.1}$$

Anfangszustand $\mathbf{x_{anf}}$	Zustand nach sequentiller Abfragereihenfolge	Zustand nach sequentiell rücklaufender Abfragereihenfolge	Zustände nach Betrachtung aller Permutationen der sequentiellen Reihenfolge	Stabile Zustände von Anfangszustand $\mathbf{x_{anf}}$	Hammingabstand zum Zustand Spalte 5	absoluter Energieinhalt von $\mathbf{x_{anf}}$	Anfangszustand $\mathbf{x_{anf}}$ gleich $\xi^{(?)}$?
0	**9**	**9**	**9**	9	2	-0,59	
1	**9**	**9**	**9**	9	1	-1,77	
3	**9**	**9**	**9**	9	2	-0,59	
5	**9**	**9**	**9**	9	2	-0,59	
6	**22**	**22**	**22**	22	1	-0,95	
8	**9**	**9**	**9**	9	1	-0,95	
9	**9**	**9**	**9**	9	0	-2,5	$-\xi^{(\mathbf{2})}$
11	**9**	**9**	**9**	9	1	-0,95	
13	**9**	**9**	**9**	9	1	-0,95	
17	**9**	**9**	**9**	9	2	-0,59	
18	**22**	**22**	**22**	22	1	-0,95	
20	**22**	**22**	**22**	22	1	-0,95	
22	**22**	**22**	**22**	22	0	-2,5	$\xi^{(\mathbf{1})}$
23	**22**	**22**	**22**	22	1	-0,59	
25	**9**	**9**	**9**	9	1	-0.95	
72	**9**	**9**	**9**	9	2	-0,59	
73	**9**	**9**	**9**	9	1	-1,77	
75	**9**	**9**	**9**	9	2	-0,59	
77	**9**	**9**	**9**	9	2	-0,59	
78	**94**	**94**	**94**	94	1	-0.95	
89	**9**	**9**	**9**	9	2	-0.59	
90	**94**	**94**	**94**	94	1	-0,95	
92	**94**	**94**	**94**	94	1	-0,95	
94	**94**	**94**	**94**	94	0	-2,5	$\xi^{(\mathbf{3})}$
95	**94**	**94**	**94**	94	1	-0,95	

Tabelle 5.1: *Netzwerk aus 8 Neuronen. Gewichtsmatrix (5.1). Anfangszustände (Spalte 1), die stabile Zustände sind oder in einem Attraktionsbecken liegen. Es ist nur die 1. Hälfte dieser Anfangszustände dargestellt, die 2. Hälfte ist symmetrisch dazu. Zustände nach asynchroner Abfrage sequentiell (Spalte 2), sequentiell rücklaufend (Spalte 3) und mit allen Permutationen der sequentiellen Reihenfolge (Spalte 4). Stabile Zustände fett. Spalte 5: Erreichbarer stabiler Zustand. Spalte 6: Hamming Abstand des Anfangszustands zum stabilen Zustand. Spalte 7: Energieinhalt des Anfangszustands*

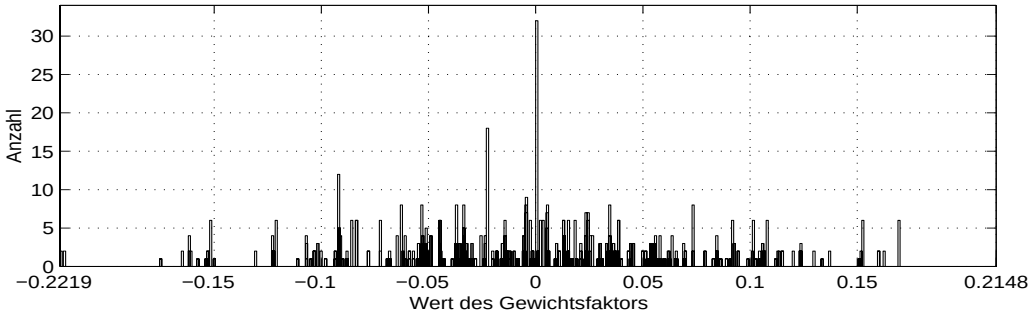


Abbildung 5.1: *Netzwerk aus 32 Neuronen für die fundamentalen Mustern (4.2) nach der Projektionsregel. Verteilung der verschiedenen Gewichtsfaktoren*

Die Ergebnisse sind in Tab. 5.1 dargestellt. Es ergeben sich demnach insgesamt 6 (2,3%) stabile Zustände (3 Muster und 3 Antimuster), 44 (17,2%) Zustände in Attraktionsbecken und 206 (80,5%) nicht- zuordenbare Zustände. Gegenüber dem unter Abschnitt 3.2 (Tab. 3.10) untersuchten Netz ist also die Gesamtgröße der Attraktionsbecken und die Zahl nicht- zuordenbarer Zustände fast gleich geblieben.

5.2 Anwendung des Verfahrens nach Abschnitt 4.2 in einem Netzwerk aus 32 Neuronen

Nimmt man die fundamentalen Muster (4.2) und bildet die Gewichtsmatrix mit der Projektionsregel, so ergibt sich die Verteilung der Gewichtsfaktoren wie in Abb. 5.1. Sucht man mit dieser Gewichtsmatrix nach stabilen Zuständen und bestimmt dann die jeweiligen Größen der Attraktionsbecken (bis 3 Hamming Abstand), so stellt man folgendes fest (Abb. 5.2): Es gibt 174 ($4,05 \cdot 10^{-6}$%) stabile Zustände, 82028 (0,0019%) Zustände in Attraktionsbecken und $4,295 \cdot 10^{9}$ (99,9%) nicht- zuordenbare Zustände. Im Vergleich zur Lernregel nach Hebb sind die Gesamtanzahl der stabilen Zustände und der Zustände in Attraktionsbecken gestiegen. Die maximale Größe des Attraktionsbeckens eines spurious states ist wesentlich geringer als die minimale eines fundamentalen Musters.

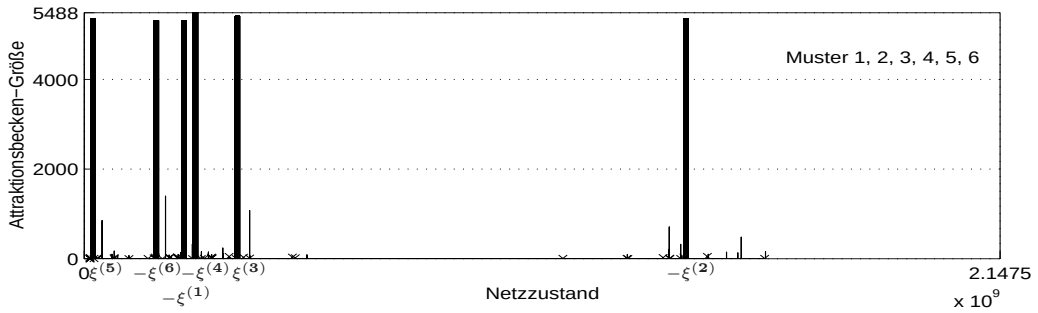


Abbildung 5.2: *Netzwerk aus 32 Neuronen. Gewichtsmatrix für die fundamentalen Muster (4.2) nach der Projektionsregel. Vergleich der Größen der Attraktionsbecken bis zu 3 Hamming Abstand vom jeweiligen fundamentalen Muster (fett) bzw. von den gefundenen spurious states. Es ist nur die 1. Hälfte der Netzzustände dargestellt, die 2. Hälfte ist symmetrisch dazu. Kreuz: Attraktionsbecken ist kleiner als 50*

Kapitel 6

Folgerungen

6.1 Allgemeine Aussagen aus den Untersuchungen

Es wurden assoziative Speicher mit asynchroner Abfrage (Permutationen der sequentiellen Reihenfolge bzw. zufällige Reihenfolge) mit bis zu 32 Neuronen untersucht. In einem solchen Netz lassen sich stabile Zustände (fundamentale Muster oder spurious states), Zustände in Attraktionsbecken und nichtzuordenbare Zustände (Zustände, die je nach Abfragereihenfolge in unterschiedliche stabile Zustände streben) unterscheiden.
Stabile Zustände treten mit oder ohne Attraktionsbecken auf. Der prozentuale Anteil der nicht- zuordenbaren Zustände steigt mit steigender Zahl der stabilen Zustände.
Obwohl die Gewichtsmatrix in allen untersuchten Fällen symmetrisch war, traten Unsymmetrien in den Netzzuständen als Folge der verwendeten Vorzeichenfunktion auf. Das kann sich in der Lage (residueller Fehler) oder dem Vorhandensein der stabilen Muster und in der Größe der Attraktionsbecken ausdrücken.
Die Größe der Attraktionsbecken nimmt mit steigender Zahl der stabilen Zustände ab. Die Attraktionsbecken der fundamentalen Muster waren in allen untersuchten Fällen größer als die der spurious states. In einem solchen Attraktionsbecken liegen Zustände, die nur einen geringen Hamming Abstand zum stabilen Zustand haben, d. h. sich nur in wenigen Pixeln von diesen unterscheiden.
Zustände innerhalb von Attraktionsbecken können den stabilen Zustand auf unterschiedlichen Trajektorien erreichen, je nach Abfragereihenfolge. Die

Attraktionsbecken waren in vielen Fällen nicht symmetrisch und ihre „Ränder“ waren manchmal unterschiedlich hoch.
Spurious states traten erst bei etwas größeren (> 10 Neuronen) Netzen auf und dort erst oberhalb von zwei eingespeicherten Mustern. Ihre Anzahl ist dann aber weder mit der Zahl der eingespeicherten Muster noch mit der Art der Einspeicherung (Lernregel nach Hebb oder Projektionsregel) korreliert.

6.2 Folgerungen für die Optimierung assoziativer Speicher für die Mustererkennung

Es war das Ziel der Arbeit, assoziative Speicher im Hinblick auf ihre Verwendung für die Mustererkennung zu untersuchen. Ein mustererkennendes System sollte folgende Bedingungen erfüllen:

(1) Translations- Invarianz zwischen dem eingespeicherten und dem zu erkennenden Muster, d. h. das Muster sollte unabhängig von seiner Position im Gesichtsfeld erkannt werden.

(2) Größen- bzw. Abstands- Invarianz (bis zu einem gewissen Grad), d. h. ein Muster sollte bis zu einem gewissen Grad unabhängig von seinem Abstand zum erkennenden System und unabhängig von gewissen Schwankungen in seiner Größe erkannt werden.

(3) Gewisse Fehlertoleranz, d. h. unvollständige oder verrauschte Muster sollten richtig zugeordnet werden.

(4) Eindeutige Unterscheidung verschiedener Muster.

(5) Unterscheidung von Mustern und Nicht- Mustern, d. h. die bedeutsamen Muster müssen von zufälligen Anordnungen zu trennen sein.

Ein assoziativer Speicher kann von den genannten Punkten folgende leisten:

(a) Die Punkte (1) und (2) sind grundsätzlich nicht erfüllbar, d. h. Muster können dann nicht als gleichartig erkannt werden, wenn sie parallel zueinander verschoben sind oder sich nur in der Größe unterscheiden. Der assoziative Speicher beurteilt die Ähnlichkeit von Mustern ausschließlich nach der Zahl ihrer übereinstimmenden Pixel (Ritter, 1992). Will man trotzdem solche Muster mit assoziativen Speichern erkennen, müssen beide Invarianzen von einem vorgeschaltetem System

gebildet werden. Dieses System muß dafür sorgen, daß entsprechende Muster immer an der gleichen Stelle und in der entsprechenden Größe auf dem neuronalen Netz auftauchen.
Da Tiere mit einer gewissen Wahrscheinlichkeit assoziative Speicher für die Mustererkennung verwenden (s. Abschnitt 1.3.2), besteht auch eine gewisse Wahrscheinlichkeit, daß dieses Problem prinzipiell lösbar ist.

(b) Die Punkte (3) bis (5) können geleistet werden. Dabei sind Fehlertoleranzen möglich, sofern ein angelegtes Muster sich im Attraktionsbecken des eingespeicherten Musters befindet, d. h. sich nur in wenigen Pixeln von ihm unterscheidet. Sofern die Attraktionsbecken fundamentaler Muster klar voneinander getrennt sind (was allerdings nicht bei allen untersuchten Netzen in vollem Umfang der Fall war, siehe Abschnitt 4.2), können verschiedene Muster klar voneinander getrennt werden.
Ein Problem bei der Verwendung der asynchronen Abfrage ist die Unterscheidung von Muster und Nicht- Muster, weil viele Anfangszustände je nach Abfragereihenfolge in unterschiedliche stabile Zustände streben (sogenannte nicht- zuordenbare Zustände). Es dürfen also nur solche Anfangszustände einem bestimmten Muster zugeordnet werden, die dieses Muster unabhängig von der Abfragereihenfolge erreichen (und nur sie wurden deshalb hier einem Attraktionsbecken zugeordnet). Es sind also eine größere Zahl unterschiedlicher Abfrageblöcke notwendig, um einen Anfangszustand einem bestimmten Muster sicher zuordnen zu können. In den hier verwendeten Netzen stellten 20 verschiedene zufällige Abfrageblockfolgen eine hinreichende Näherung dar. Wenn also nach dieser Zahl von Abfrageblockfolgen immer das gleiche fundamentale Muster erreicht wird, ist der betreffende Anfangszustand als ein Muster erkannt, auch wenn er dem Muster nicht genau entspricht.
Nicht- Muster entsprechen entweder spurious states, Anfangszuständen, die in Attraktionsbecken von spurious states liegen oder nicht- zuordenbaren Zuständen. Anfangszustände, die in Attraktionsbecken von spurious states liegen bzw. spurious states sind, sind mit wenigen Abfrageblockfolgen daran zu erkennen, daß sie zu einem stabilen Zustand streben, der nicht ein Muster repräsentiert.

Die nicht- zuordenbaren Zustände sind nur mit größerem Aufwand von Zuständen im Attraktionsbecken eines Musters zu unterscheiden. Es genügt nicht, nur mehrere einzelne Abfrageblöcke anzuwenden, da auch Zustände innerhalb eines Attraktionsbeckens auf unterschiedlichen Trajektorien den stabilen Zustand erreichen können, nach einem Abfrageblock also ebenfalls bei unterschiedlichen Zuständen enden können. Man muß also so viele Abfrageblockfolgen anwenden, bis zum ersten Mal ein anderer stabiler Zustand erreicht wird (nicht- zuordenbarer Zustand $\equiv$ Nicht- Muster) oder immer wieder das gleiche Muster erreicht wird ($\equiv$ Muster, s. oben). Die Wahrscheinlichkeit, daß ein zufälliger Anfangszustand einem nicht- zuordenbaren Zustand entspricht, ist sehr hoch. Deshalb kann die Unterscheidung von Muster und Nicht-Muster sehr aufwendig werden.

Aus dem Gesagten ergeben sich Ziele für die Optimierung assoziativer Speicher für die Mustererkennung. Eine solche Optimierung könnte z. B. durch iterative Verfahren geschehen.

(i) Vergrößerung der Attraktionsbecken von Mustern. Dies ist prinzipiell möglich (s. Abschnitt 5.2).

(ii) Die „Ränder“ der Attraktionsbecken der Muster sollten möglichst „gleich hoch“ und nicht in Richtung benachbarter Muster „tiefer“ sein (wie z. B. offensichtlich zwischen den Mustern $\xi^{(\mathbf{1})}$ und $\xi^{(\mathbf{4})}$ der Netze von Abschnitt 4.2 und 4.3). Dieses Problem kann möglicherweise nicht nur durch eine Optimierung der assoziativen Speicher selbst, sondern auch durch eine entsprechende Codierung der Muster erreicht werden (z. B. dadurch, daß der Bereich, in dem sich die Muster unterscheiden, vergrößert auf dem assoziativen Speicher abgebildet wird).

(iii) Die Attraktionsbecken aller Muster sollten ungefähr gleich groß sein.

(iv) Da Nicht- Muster in Form von spurious states schnell erkannt werden können, ist es nicht nötig, die Zahl der spurious states und die Größe ihrer Attraktionsbecken zu minimieren.

Anhang A

A.1 Implementierung des asynchronen Abfragemechanismus

Die Funktion `timestep_asyncall` (Abb. A.1) berechnet je nach Anwendungsfall entweder den Netzzustand nach Verwendung eines einzigen beliebigen Abfragereihenfolge- Blocks `A` oder die Netzzustände nach Verwendung mehrerer beliebiger Abfrageblöcke einer bestimmten Länge.
Ihre Übergabeparameter sind `w`, ein Zeiger der Größe N^2, der in seinen Elementen die Gewichtselemente in dieser Reihenfolge $w_{11}, w_{21}, \ldots, w_{N1}, w_{12}, w_{22}, \ldots, w_{N2}, w_{13}, w_{23}, \ldots, w_{N3}, w_{14} \ldots w_{N4}, \ldots, w_{NN}$ beinhaltet. Dabei muß vor der Übergabe gewährt sein, daß die Gewichtung auf sich selbst, also w_{ii}, gleich Null ist. Die Größe des Netzes `N` muß dieser Unterfunktion ebenfalls mit übergeben werden. Ein weiterer Übergabeparameter ist der Anfangszustand `x_anf`. Dabei muß bereits vor dem Aufruf dieser Funktion sichergestellt worden sein, daß die Länge des Zeigers von `x_anf` N ist, und ihre Elemente gleich ± 1 sind.
Die Abfragereihenfolge `A` und die Länge der Abfragereihenfolge l werden ebenfalls als Parameter übergeben, der Parameter `h` bestimmt in seinem Wert den oben aufgeführten Anwendungsfall. Ist `h` gleich 1, so ist $\mathbf{A}$ ein Vektor der Länge l. Ist h größer 1, so ist `A` eine Matrix der Größe $h \times l$, wenn h die Anzahl der Zeilen und l die Anzahl der Spalten bestimmt. Dabei stehen die h verschiedenen Abfragereihenfolgen als Zeilenvektoren in der Matrix $\mathbf{A}$.
Diese Verallgemeinerung ist vor allem für die in Abschnitt 2.4.4 angegebenen Permutationen der sequentiellen Abfragereihenfolge nützlich, die alle die gleiche Länge $l = N$ haben. Dadurch ist der Parameter `x_ende` eine $h \times l$ Matrix, in denen die Endzustände nach der Berechnung mit der jeweiligen Abfragereihenfolge in Zeilen stehen.

Zunächst wird der Anfangszustand `x_anf` in die Zeilen von `x_ende` kopiert und die Zustände der Neuronen nach der Gleichung (2.5) mit Verwendung der Abfragereihenfolge `A` berechnet. Da die Neuronen nach Gleichung (2.3) von 1 bis N durchnumeriert sind, die Positionen in den Zeigern aber je von 0 bis $N-1$ sind, muß diese Poistionsänderung in den Zeigern mit berücksichtigt werden. Die Vorzeichenfunktion nach Gleichung (2.7) ist sehr empfindlich gegenüber Rundungsfehlern in der Programmiersprache C. Deshalb muß man z. B. auf 10 Stellen nach dem Komma runden, um das Vorzeichen des inneren Arguments (2.6) sicher zu erkennen.
Wenn man diese Funktion als mex- File in Matlab einbinden will, benutzt man in Matlab folgende Syntax:

$$\textbf{x\_ende=timestep\_asyncall(w,x\_anf,A)} \tag{A.1}$$

Matlab liefert Funktionen (s. Abb. A.9 für eine andere Funktion), die die benötigte Speicherallokation der Zeiger und die Umschreibung einer Matrix in einen Zeiger in eine Reihenfolge (s. o.) übernehmen. Die Größe der eingegebenen Matrizen bzw. Vektoren werden bestimmt und als Parameter `N,` l `, h` in die Funktion übergeben. Der berechnete Zustand `x_ende` wird in Matlab als Vektor bzw. als Matrix ausgegeben.
Funktionen werden in Matab interpretativ verarbeitet, d. h. daß in der Laufzeit Machinencode generiert wird, was allerdings viel Zeit in Anspruch nimmt. Deshalb werden rechenintensive Funktionen im C- Quellcode geschrieben, wo der Maschinencode vorab in einer Kompilierphase erstellt wurde und zur Laufzeit aufgerufen wird.

```
static
timestep_asyncall(A,N,h,l,w,x_anf,x_ende)
double *w,*x_anf,*x_ende;
unsigned int *A,N,h,l;
{
   int A_j,i,j,k;
   double y_j;

   for (i=0;i<h;i++) {
      /* Kopie von x_anf in die i.te Zeile von x_ende */
      for (j=0;j<N;j++) {
         x_ende[i+j*h]=x_anf[j];
      }
      /* Durchführung von l updates in der i.ten Zeile */
      for (j=0;j<l;j++) {
         /* Gleichung (2.5) mit Neuron A_j und             */
         /* Berücksichtigung von Positionsverschiebungen   */
         /* bei der Verwendung von Zeigern in C, im Gegen- */
         /* satz zur Bezeichnung von Neuronen nach (2.3).  */
         A_j=A[i+j*h]-1;
         /* y_j nach Gleichung (2.6) des Neurons A_j */
         y_j=0;
         for (k=0;k<N;k++) {
            y_j=y_j+w[A_j+N*k]*x_ende[i+k*h];
         }
         /* Berechnung auf 10 Stellen nach dem Komma genau */
         y_j=ceil(y_j*1e11)/1e11;
         /* Vorzeichenfunktion (2.7) */
         if (y_j>=0) {
            x_ende[i+A_j*h]=1;
         } else {
            x_ende[i+A_j*h]=-1;
         }
      }
   }
   return;
}
```

Abbildung A.1: Funktion `timestep_asyncall` im C- Quellencode

A.2 Bestimmung der Abfragereihenfolgen

Um die Abfragereihenfolge zu bestimmen, kann man folgende Funktionen verwenden. Eine Bestimmung aller **Permutationen der sequentiellen Reihenfolge** (Kap. 3 und Abschnitt 5.1) wird in Abb. A.2 vorgeschlagen und implementiert. Dabei müssen die Parameter `N`, `h`= $N!$ und ein zu Null initialisierter Zeiger `A` der Größe $h \times N$ mit in die Funktion übergeben werden. In `A` stehen dann in den Zeilen die N! verschiedenen Abfragereihenfolgen. Diese Funktion kann bis maximal $N = 10$ angewendet werden. Ist die Matrix `A` berechnet, läßt sie sich als Übergabeparameter in die Funktion `timestep_asyncall` einsetzen. Die Funktion `timestep_asyncall` (s. Abschnitt A.1) berechnet dann die dazugehörigen erreichten Zustände, aus denen sich die Tabellen und Histogramme in Kap. 3 zusammensetzen.
Zur Bestimmung einer **zufälligen Permutation der sequentiellen Reihenfolge** (Kap. 4 und Abschnitt 5.2) benutzt man die Funktion `randperm` (Abb. A.3). Dabei werden N Zufallszahlen sortiert. Die Abfragereihenfolge `A` gibt dabei die ursprüngliche Position (plus 1) der Zufallszahl an.
Wenn man einen **zufälligen Abfrageblock** bestimmen will, kann man die Funktion `randblock` (Abb. A.4) benutzen. Hier kontrolliert ein zweiter Zeiger die Einträge in `A`. Erst wenn alle Neurone zufällig gewählt sind, ist diese Funktion `randblock` beendet. Die Länge l des Abfrageblocks wird in die erste Position des Zeigers `lz` übergeben. Diese Länge l des Abfrageblocks darf die Zahl der Dimensioierung (DIM) nicht überschreiten.

```
static
allperm(A,h,l)
unsigned int *A,h,l;
{
   int faculzaehl,faculzaehlmin1,i,j,k,step,zaehl;

   for (i=0;i<l*h;i++) {
      A[i]=0;
   } /* Initialisierung von A für zunächst zwei Neuronen */
   A[0]=1;A[1]=2;A[h]=2;A[h+1]=1;
   for (zaehl=3;zaehl<=l;zaehl++) {
      /* Bildung von faculzaehl=zaehl! */
      /* und faculzaehlmin1=(zaehl-1)! */
      faculzaehl=1;faculzaehlmin1=1;
      for (j=1;j<=zaehl;j++) {
         if (j<zaehl) {
            faculzaehlmin1=faculzaehlmin1*j;
         }
         faculzaehl=faculzaehl*j;
      }
      /* iterative Kopie von A von zaehl-1 nach A von zaehl.  */
      /* Dabei bei wird A von zaehl-1 um je eine Position nach*/
      /* rechts und unterhalb vorangehender Blöcke verschoben.*/
      for (i=1;i<=zaehl;i++) {
         for (j=0;j<=zaehl-2;j++) {
         step=0;
            for (k=(i-1)*faculzaehlmin1;k<i*faculzaehlmin1;k++) {
                A[k+h*((i+j+zaehl-1) % zaehl)]=A[step+h*j];
               step=step+1;
            }
         } /* Belegung der noch zu Null initialisierten A[i] */
         for (k=(i-1)*faculzaehlmin1;k<i*faculzaehlmin1;k++) {
            A[k+h*((i+zaehl-2) % zaehl)]=((zaehl-1) % l)+1;
         }
      }
   }
   return;
}
```

Abbildung A.2: *Funktion* `allperm` *im C- Quellencode*

```
#include <math.h>
#include <stdlib.h>

static
randperm(A,N,randv)
unsigned int *A,N,*randv;
{
   int buffer,buffer2,i,j,m;

   /* Bildung von N beliebiger Zufallszahlen und      */
   /* sequentieller Inizialisierung von A mit 1 bis N */
   for (i=0;i<N;i++) {
      randv[i]= (int) ceil((double) rand()/32768 *N);
      /* Die Verwendung der ceil() Funktion aus <math.h> und */
      /* des Terms (/32768*N) ist nicht unbedingt erforder-  */
      /* lich. Dann müssen aber *randv und buffer in Zeile 6 */
      /* und 8 neu angepaßt werden.                          */
      A[i]=i+1;
   }
   /* Sortierung von randv. A[i] gibt dabei die       */
   /* ursprüngliche Position von randv_i (plus 1) an. */
   for (i=0;i<N;i++) {
      m=i;
      for (j=i;j<N;j++) {
         if (randv[j]>randv[m]) {
            m=j;
         }
         buffer=randv[i];
         buffer2=A[i];
         randv[i]=randv[m];
         A[i]=A[m];
         randv[m]=buffer;
         A[m]=buffer2;
      }
   }
   return;
}
```

Abbildung A.3: *Funktion* `randperm` *im C- Quellencode*

```
#include <math.h>
#include “mex.h“
#define DIM N*100

static
randblock(A,N,lz,inlist)
unsigned int *A,N,*lz,*inlist;
{
   int i,j,found,step;

   step=0;
   for (i=0;i<DIM;i++) {
      A[i]=(double) ceil((double)rand()/32768*N);
      if (i==0) { /* Erster Inhalt: inlist[0] */
         inlist[step]=A[i]; step=step+1;
      } else {
         j=0;found=0;
         while (j<step) {
            if (A[i]==inlist[j]) {
               found=1;
            }
            j=j+1;
         }
         if (found==0) { /* Eintrag in inlist[step] */
            inlist[step]=A[i]; step=step+1;
         }
      }
      if (step==N) {
         /* Mindestens jedes Neuron einmal in A */
         lz[0]=i+1; /* Länge von A */
         i=DIM;
      }
   }
   if (step != N) {
      mexErrMsgTxt(“Die Dimensionierung ist zu klein gewählt“);
   }
   return;
}
```

Abbildung A.4: *Funktion* `randblock` *im C- Quellencode*

A.3 Implementierung der in Kap. 4 verwendeten Verfahren

Bei der **Bildung eines Trajektorien- Baumes** wird von einem Anfangszustand aus ein neuer Zustand nach der Verwendung eines Abfrageblocks berechnet (Abb. A.5). Man kann mehrere Abfrageblöcke hintereinander verwenden, wenn man den bereits erreichten Zustand als Anfangszustand für den neuen Abfrageblock nimmt. Diese Abfrageblockfolgen werden gebraucht, wenn man stabile Zustände sucht. So wird in der Funktion `timestepq` ein Zustand mit dem nächsten verglichen. Stimmen sie überein, wird die Funktion beendet, und der so gefundene stabile Zustand in `x_ende` ausgegeben.
In der Funktion `timestepq` (Abb. A.5) werden die anderen Funktionen `randblock` (Abb. A.4) und `timestep_asyncall` (Abb. A.1) aufgerufen. Die Funktion `randblock` generiert zufällige Abfrageblöcke. Äquivalent kann auch die Funktion `randperm` aufgerufen werden. Die Abfrageblöcke werden von Zwischenzustand zu Zwischenzustand überschriebenn. Will man die vollständige Reihenfolge, in der Neurone abgefragt werden, bestimmen, muß man die jeweiligen Abfrageblöcke in einen weiteren Zeiger kopieren.

Um stabile Zustände darstellen zu können, formt man die stabilen Zustände mit der Funktion `zustand2zahl` (Abb. A.6) in die Dezimalzahl um.

In Kap. 4 wird 12 000 mal die Funktion `timestepq` (Abb. A.5) angewandt, und die stabilen Zustände als Dezimalzahlen in einen Zeiger der Länge 12 000 eingespeichert. Anschließend werden die verschiedenen stabilen Zustände nach dem Prinzip, wie es in der Funktion `randblock` eingeführt wurde, extrahiert.

```
static
timestepq(A,N,lz,inlist,w,x_anf,x_ende,x_merk)
double *w,*x_anf,*x_ende,*x_merk;
unsigned int *A,N,*lz,*inlist;
{
   int i,sum,quality;

   /* Kopie von x_anf in x_ende bzw. x_merk */
   for (i=0;i<N;i++) {
      x_ende[i]=x_anf[i];
      x_merk[i]=x_anf[i];
   }
   quality=1;
   while (quality == 1) {
      randblock(A,N,lz,inlist);
      timestep_asyncall(A,N,1,lz[0],w,x_merk,x_ende);
      /* Vergleich zwischen dem neuem und dem Anfangszustand */
      sum=0;
      for (i=0;i<N;i++) {
         if (x_merk[i] == x_ende[i]) {
            sum=sum+1;
         }
      }
      if (sum == N) {
         quality=0;
      } else {
         quality=1;
      }
      /* Beendigung bei Übereinstimmung. Ansonsten werden   */
      /* vom erreichten Zustand weitere Berechnungen durch- */
      /* geführt. Dabei wird A von neuem überschrieben.     */
      for (i=0;i<N;i++) {
         x_merk[i]=x_ende[i];
      }
   }
   return;
}
```

Abbildung A.5: *Funktion* `timestepq` *im C- Quellcode*

```
static
zustand2zahl(binaer,zahl,zustand,N,h)
double *binaer,*zahl,*zustand;
unsigned int N,h;
{
   int i,j,k,step;
   double power;

   /* Umformung in Binärwerte nach Gleichung (2.20) */
   for (i=0;i<(N*h);i++) {
      binaer[i]=(zustand[i]+1)/2;
   }
   /* Umformung der Binärzahl in eine Dezimalzahl   */
   for (i=0;i<h;i++) {
      zahl[i]=0;step=0;
      for (j=N;j>0;j--) {
         power=1;
         for (k=(j-1);k>0;k--) {
            power=power*2;
         }
         zahl[i]=zahl[i]+power*binaer[i+h*step];
         step=step+1;
      }
   }
   return;
}
```

Abbildung A.6: *Funktion* `zustand2zahl` *im C- Quellencode*

Für die **Untersuchung der stabilen Zustände** werden in `hamming3` (Abb.A.7) vom stabilen Zustand `x_st` aus $\left[\binom{N}{1}+\binom{N}{2}+\binom{N}{3}\right]$ Anfangszustände in `hmv` gebildet. Dabei muß vor Aufruf der Funktion sichergestellt sein, daß `x_st` ein stabiler Zustand ist. Dies kann man mit `timestep_asyncall` und der sequentiellen Abfragereihenfolge (Gleichung (2.17)) überprüfen.
Diese Anfangszustände werden in die Funktion `abis3h` (Abb. A.8) übergeben. Dann wird jeder einzelne Anfangszustand in `x_anf` kopiert, ferner Abfrageblockfolgen gebildet und der stabile Zustand mit der Funktion `timestepq` (Abb. A.5) ermittelt.
Ist dieser erreichte Zustand der initialisierte stabile Zustand `x_st`, so werden weitere Abfrageblockfolgen bis maximal 20 gebildet. Sobald der erreichte Zustand vom initialisierten stabilen Zustand abweicht, wird die Untersuchung abgebrochen und der nächste Anfangszustand genommen. Streben alle 20 Abfrageblockfolgen zum stabilen Zustand `x_st`, werden sie in die Variable `x_inside` aufgenommen. `x_inside` beinhaltet somit die Anfangszustände im Attraktionsbecken.
Da sich ihre Größe erst zur Laufzeit bestimmen läßt, muß im Initialierungsprogramm für die Funktion `abis3h` (`mexFunction`, Abb. A.9) ein temporärer Zeiger `x_tmp` derselben Größe wie `hmv` angelegt werden. Nach Bestimmung der Anzahl der Zustände im Attraktionsbecken wird dann ein neuer Zeiger `x_inside` angelegt, in dem diese Anfangszustände vollständig in diesen neuen Zeiger kopiert und ausgegeben werden.

Man kann mit den Funktionen `timestep_asyncall` (Abb. A.1), `randblock` (Abb. A.3), `timestepq` (Abb. A.5), `hamming3` (Abb. A.7), `abis3h` (Abb. A.8) und `mexFunction` (Abb. A.9) z. B. das Programm abis3h.c bilden. Mit `cmex`, ein von Matlab geliefertes Programm, läßt sich das Programm `abis3h.c` zu `abis3h.mex` kompilieren. Das Programm `abis3h.mex` ruft man in Matlab mit folgender Syntax auf:

$$\textbf{x\_inside=abis3h(w,x\_st)}. \tag{A.2}$$

```
static
hamming3(N,hmv,x_st)
double *hmv,*x_st;
unsigned int N;
{
   int ges,i,j,k,l,step2,step3;

   ges=((N)+(N*(N-1)/2)+(N*(N-1)*(N-2)/6));
   for (i=0;i<N;i++) {
      for (j=0;j<N;j++) {
         hmv[ges*j+i]=x_st[j];
      }
      hmv[ges*i+i]=hmv[ges*i+i]*(-1);
   }
   /* Hamming 2, 3 von Hamming 1 kopiert und verändert */
   step2=N;step3=N+N*(N-1)/2; /* Startwerte in *hmv    */
   for (i=0;i<N;i++) {
      for (j=i+1;j<N;j++) {
         for (k=0;k<N;k++) {
            hmv[ges*k+step2]=hmv[ges*k+i];
         }
         hmv[ges*j+step2]=hmv[ges*j+step2]*(-1);
         step2=step2+1;
         for (k=j+1;k<N;k++) {
            for (l=0;l<N;l++) {
               hmv[ges*l+step3]=hmv[ges*l+i];
            }
            hmv[ges*j+step3]=hmv[ges*j+step3]*(-1);
            hmv[ges*k+step3]=hmv[ges*k+step3]*(-1);
            step3=step3+1;
         }
      }
   }
   return;
}
```

Abbildung A.7: *Funktion* `hamming3` *im C- Quellencode*

```
static
abis3h(A,N,hmv,hz,lz,inlist,w,x_anf,x_ende,x_inside,x_merk,x_st)
double *hmv,*w,*x_anf,*x_ende,*x_inside,*x_merk,*x_st;
unsigned int *A,N,*hz,*lz,*inlist;
{
   int found,ges,h,i,j,k,step;

   h=0;ges=((N)+(N*(N-1)/2)+(N*(N-1)*(N-2)/6));
   hamming3(N,hmv,x_st); /* Bestimmung der Anfangzustände */
   for (i=0;i<ges;i++) {
      for (j=0;j<N;j++) {
         x_anf[j]=hmv[i+ges*j];
      }
      j=0;found=0;
      while (j<20) { /*Durchlauf von max 20 Abfrageblockfolgen*/
         timestepq(A,N,lz,inlist,w,x_anf,x_ende,x_merk);
         step=0; /* stabiler Zustand == erreichter Zustand ? */
         for (k=0;k<N;k++) {
            if (x_ende[k]==x_st[k]) {
               step=step+1;
            }
         }
         if (step==N) {/* Wenn ja,weitere Abfrageblockfolgen */
            j=j+1;
         } else {   /* sonst Untersuchung des nächsten x_anf */
            found=1;j=20;
         }
      }
      if (found==0) {  /* x_anf gehört zum Attraktionsbecken */
         for (j=0;j<N;j++) {
            x_inside[h+ges*j]=x_anf[j];
         }
         h=h+1;
      }
   }
   hz[0]=h;   /* x_inside hat die Größe h×N */
   return;
}
```

Abbildung A.8: *Funktion* `abis3h` *im C- Quellencode*

```
mexFunction(nlhs,plhs,nrhs,prhs)
int nlhs,nrhs;
Matrix *plhs[],*prhs[];
{
   double *hmv,*w,*x_anf,*x_ende,*x_inside,*x_merk,*x_st,*x_tmp;
   unsigned int *A,N,*hz,*lz,*inlist;
   int ges,i,j;

   ges=((N)+(N*(N-1)/2)+(N*(N-1)*(N-2)/6));
   if (nrhs != 2) {
      mexErrMsgTxt("Diese Funktion benötigt 2 Eingänge.");
   } else if (nlhs > 1){
      mexErrMsgTxt("Diese Funktion benötigt maximal 1 Ausgang.");
   }
   /* Erhalt von Eingabevariablen */
   N=mxGetM(prhs[0]);  /* Jetzt Symmetrie etc. prüfen, * /
   w=mxGetPr(prhs[0]); /* wird hier übergangen.        * /
   x_st=mxGetPr(prhs[1]);
   /* Allokation und Zuordnung der inizialisierten Zeiger */
   plhs[1]=mxCreateFull(1,DIM,REAL);A=mxGetPr(plhs[1]);
   plhs[2]=mxCreateFull(ges,N,REAL);hmv=mxGetPr(plhs[2]);
   plhs[3]=mxCreateFull(1,1,REAL);hz=mxGetPr(plhs[3]);
   plhs[4]=mxCreateFull(1,1,REAL);lz=mxGetPr(plhs[4]);
   plhs[5]=mxCreateFull(1,N,REAL);inlist=mxGetPr(plhs[5]);
   plhs[6]=mxCreateFull(1,N,REAL);x_anf=mxGetPr(plhs[6]);
   plhs[7]=mxCreateFull(1,N,REAL);x_ende=mxGetPr(plhs[7]);
   plhs[8]=mxCreateFull(1,N,REAL);x_merk=mxGetPr(plhs[8]);
   plhs[9]=mxCreateFull(ges,N,REAL);x_tmp=mxGetPr(plhs[9]);
   /* Berechnung in Funktionen */
   abis3h(A,N,hmv,hz,lz,inlist,w,x_anf,x_ende,x_tmp,x_merk,x_st);
   /* Allokation und Zuordnung des Ausgabe- Zeigers */
   plhs[0]=mxCreateFull(hz[0],N,REAL);x_inside=mxGetPr(plhs[0]);
   for (i=0;i<N;i++) {
      for (j=0;j<hz[0];j++) {
         x_inside[i*hz[0]+j]=x_tmp[ges*i+j];
      }
   }
   return;
}
```

Abbildung A.9: *Funktion* `mexFunction` *im C- Quellencode*

Literaturverzeichnis

[1] Amari, S. und Yanai, H-F. (1993) *Statistical Neurodynamics of Various Types of Associate Nets*, in Hassoun, M. H.: Associative Neural Memories (Oxford University Press, New York)

[2] Amit, D. J., Gutfreund, H. und Sompolinsky, H. (1985a) *Storing Infinite Number of Patterns in a Spin-glass Model of Neural Networks,* Phys. Rev. Lett., **55**, 1530-1533

[3] Amit, D. J., Gutfreund, H. und Sympolinsky, H. (1985b) *Spin-glass Model of Neural Networks,* Phys. Rev., **A32**, 1007-1018

[4] Amit, D.J. (1992) *Modeling Brain Function: The World of Attractor Neural Networks* (Cambridge University Press, Cambridge)

[5] Bergdoll, S. und Koch, U. T. (1995) *BIOSIM - A biological neural network simulator for research and teaching, featuring interactive graphical user interface and learning capabilities,* Neorocomputing, **8** 93-112

[6] Cruse, H. (1972) *Versuch einer quantitativen Beschreibung des Formensehens der Honigbiene,* Kybernetik, **11**, 185-200

[7] Cruse, H. (1996) *Neural Networks as Cybernetic Systms* (Thieme Verlag, Stuttgart)

[8] Diederich, S. und Opper, M. (1987) *Learning of Correlated Patterns in Spin-Glass Networks by Local Learning Rules,* Phys. Rev. Lett. 58, **9**, S. 949

[9] Eeckmann, F. H. (1992) *Analysis and Modeling of Neural Systems* (Kluwer Academic)

[10] Gardner, E. J. (1988) *The Space of Interactions in Neural Network Models,* J. Phys., **A** 21, 257-270

[11] Golden, R. M. (1993) *Stability and Optimization Analyses of the Generalized Brain-State-in-a-Box Neural Network Model,* Journal of Mathematical Psychology, **37**, 282-298

[12] Hassoun, M. H. (1995) *Fundamentals of Artificial Neural Networks* (The MIT Press, Cambridge)

[13] Haykin, S. (1994) *Neural Networks- A Comprehensive Foundatation* (Prentice- Hall, New Jersey)

[14] Hebb, D. O. (1949) *The Organisation of Behaviour* (Wiley, New York)

[15] Herz, A. V. M. (1996) *Global Analysis of Recurrent Neural Networks*, in Domany, E.; Van Hemmen, J. L. und Schulten, K.: Models of Neural Networks III- Association, Generalization, and Representation (Springer Verlag, New York)

[16] Hopfield, J. J. (1982) *Neural networks and pysical systems with emergent collective computational abilities*, Proc. Nat. Acad. Sci. USA **79**, S. 2554

[17] Kamp, Y. und Hasler M. (1990) *Recursive Neural Networks for Associative Memory* (John Wiley & Sons, Chichester)

[18] Kanter, I. und Sompolinsky, H. (1987) *Associative Recall of Memory without Errors,* Phys. Rev, **A** 35, S. 380

[19] Kohonen, T. (1988) *Self-Organisation and Associative Memory* (Springer Verlag, Berlin)

[20] Komlós, J. und Paturi, R. (1988) *Convergence Results in an Associative Memory Model,* Neural Networks, **1**, S. 239-250

[21] Komlós, J. und Paturi, R. (1993) *Convergence Analysis of Associative Memories*, in Hassoun, M. H.: Associative Neural Memories (Oxford University Press, New York)

[22] Krauth, W., Nadal, J.-P. und Mézard, M. (1988) *The Role of Stability and Symmetry in the Dynamics of Neural Networks,* J. Phys., **21**, 2995-3011

[23] Liu, D. und Lu, Z. (1997) *A New Synthesis Approach for Feedback Neural Networks Based on the Percepron Training Algorithm,* Trans. Neural Networks, **8**, 1468

[24] Maier, S. H. (1995) *Analyse von Assoziativen Speichern und deren Dimensionierung mit Hilfe von Separatrices* (Diplomarbeit am Institut für Netzwerk- und Systemtheorie, Prof. Dr.-Ing. habil. E. Lüder, Universität Stuttgart)

[25] Mazza, C. (1997) *On the Storage Capacity of Nonlinear Neural Networks,* Neural Networks, **10**, 593-597

[26] McEliece, R. J., Posner, E. C., Rodemich, E. R. und Venkatesh, S. S. (1987), *The Capacity of the Hopfield Associative Memory*, IEEE Trans. Information Theory, **33**, 461-482

[27] Montgomery, B. L. und Vijaya Kumar, B.V.K. (1986), *Evaluation of the Use of the Hopfield Neural Network Model as a Nearest-neighbor Algorithm*, Appl. Opt., **25**, 3759-3766

[28] Newman, C. M. (1988) *Memory Capacity in Neural Network Models: Rigorous Lower Bounds,* Neural Networks, **1**, 223-238

[29] Ott P. (1997) *Bilderkennung mit assoziativen Speichern und deren massiv- parallele, optoelektronische Realisierung* (Doktorarbeit am Institut für Netzwerk- und Systemtheorie, Prof. Dr.-Ing. habil. E. Lüder, Universität Stuttgart)

[30] Pancha, G. und Venkatesh, S.S. (1993) *Feature and Memory-Selective Error Correlation in Neural Associative Memory*, in Hassoun, M. H.: Associative Neural Memories (Oxford University Press, New York)

[31] Perfetti, R. *A Neural Network to Design Neural Networks,* Trans. on Circuits and Systems, **38**, 1099-1103

[32] Personnaz, L., Guyon, I. und Dreyfus, G. (1986) *Collective Computational Properties of Neural Networks: New Learning Mechanisms,* Physical Rewiew A, **34**, 4217-4228

[33] Reichert, H. (1990) *Neurobiologie* (Thieme Verlag, Stuttgart)

[34] Ritter, H., Martinetz, T. und Schulten, K. (1992) *Neuronale Netze- Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke,* (Addison- Wesley, Bonn)

[35] Schnelle, T. (1992) *Spinglasmodelle für neuronale Netzwerke und ihre Behandlung mit Methoden der statistischen Physik* (Dissertation am Institut für Theoretische Physik, Prof. Dr. rer. nat. habil. R. Glaser, Humboldt- Universität Berlin)

[36] Selverston, A. I. und Moulins, M. (1987) *The Crustacean Stomatogastric System* (Springer Verlag, Berlin)