Raum-zeitliche Strukturen von Berechnungen in biologisch realistischen neuronalen Netzwerken

Eine Diplomarbeit durchgeführt von

Thomas Natschläger

am Institut für

Grundlagen der Informationsverarbeitung
Theoretical Computer Science
Technische Universität Graz

Betreuer:
Dipl.-Inform. Berthold Ruf
Dipl.-Math. Dr.rer.nat O.Univ.-Prof. Wolfgang Maass

Begutachter:
Dipl.-Math. Dr.rer.nat O.Univ.-Prof. Wolfgang Maass

Graz, Mai 1996

Inhaltsverzeichnis

Kapitel 1

Einleitung

Gehirn und Nervensystem nehmen täglich eine große Zahl von Sinneseindrücken auf und verarbeiten sie in einem vielschichtigen Prozeß, in welchem die Einzelinformationen verknüpft und mit früheren Erfahrungen verglichen werden. Eine zentrale Fragestellung in der Theorie der neuronalen Signalverarbeitung ist dabei das Problem der Kodierung, mit der ein neuronales System Ereignisse der Außenwelt auf interne Zustände abbildet. In den meisten modellmäßigen Beschreibungen sowie in vielen Experimenten wird davon ausgegangen, daß die *mittlere Feuerrate* der Neuronen die wesentliche Information enthält. Ein einzelnes Neuron kann dann durch einen einfachen nichtlinearen Prozessor beschrieben werden, der die Summe der Eingangssignale in eine mittlere Feuerrate als Ausgangssignal verwandelt.

Reale Neuronen senden aber nicht das geglättete Signal einer mittleren Feuerrate an nachgeschaltete Nervenzellen, sondern eine komplizierte Folge von kurzen Spannungsimpulsen, die sogenannten *Aktionspotentiale*. Es wird in letzter Zeit immer mehr vermutet, daß die zeitlich aufgelöste Folge von Aktionspotentialen wichtige Informationen enthält. Es gibt hierzu einige Experimente die diese Behauptung immer mehr untermauern (siehe zum Beispiel [3, 2]).

In dieser Arbeit sollen nun Netzwerke, die aus Modellneuronen bestehen, die den Aspekt der zeitlichen Auflösung berücksichtigen, betrachtet werden. Dabei wird hauptsächlich der Aspekt der Reproduktion von Impulsfolgen von ganzen Netzwerken betrachtet. Das heißt, es wird auch noch ein gewisser räumlicher Faktor von Information hinzugenommen.

Die Fähigkeit, räumlich zeitliche Aspekte von Informationen verarbeiten zu können, ist eine Voraussetzung um auf eine natürliche, das heißt sich dauernd ändernde Umgebung, reagieren zu können. Diese einfache Tatsache erklärt, weshalb biologische Organismen hochentwickelte Mechanismen entwickelt haben, um räumlich zeitliche Muster (spatiotemporäre Muster) erkennen, erzeugen und lernen zu können.

Ein Beispiel wie notwendig es ist, räumlich zeitliche Aspekte von Informationen zu verarbeiten, zeigt die Fähigkeit vieler Lebewesen die Richtung aus der ein Laut kommt, bestimmen zu können. Um dies zu bewerkstelligen, muß es im zentralen Nervensystem Mechanismen geben, die die Zeitdifferenz (zeitlicher Aspekt) der zwei Signale (räumlicher Aspekt) vom linken und vom rechten Ohr verarbeiten. Ein weiteres Beispiel stellt das Vermögen eines Menschen dar, auch sehr undeutlich gesprochene Worte mit dem tatsächlich gemeinten Wort zu assoziieren (spatiotemporäre Assoziation).

Eine geschlossene Theorie der der spatiotemporären Musterverarbeitung zugrundeliegenden neuronalen Prozesse ist vom heutigen Wissensstand aus gesehen, noch lange nicht vorhanden. Aber sogar die kompliziertesten spatiotemporären Muster sind aus einfacheren spatiotemporären Bausteinen zusammengesetzt, die sich im allgemeinen über eine Zeitdauer von etwa einigen Millisekunden (ms) erstrecken. Fix verdrahtet oder in einer verläßlichen Art und Weise gelernt, sind diese Bausteine in bestimmten Gehirnregionen abgespeichert und unterstützen eine verläßliche Wiedergabe oder ein Erkennen des gesamten spatiotemporären Objekts.

Wie können nun solch elementare Sequenzen von Mustern in neuronalen Strukturen repräsentiert werden? Was sind mögliche Mechanismen um spatiotemporäre Muster zu speichern und sie wieder erkennen zu können? Ist es möglich die globalen Berechnungsvorgänge qualitativ und quantitativ zu beschreiben? Welche Relevanz haben diese Modelle für biologische Systeme?

Überblick

In Kapitel 2 werden zunächst die biologischen Grundlagen erläutert. Zudem werden Modelle für Netzwerke angegeben, die den Zeitpunkt des Aktionspotentials in dem Mittelpunkt stellen; zum einen das formale Modell eines Netzwerkes feuernder Neuronen (vgl. [23]), zum anderen das Kompartmentmodell, welches verwendet wird, um die Simulationen durchzuführen (vgl. [22, 7]).

Die folgenden Kapitel versuchen Konstruktionsverfahren anzugeben, wie man bestimmte spatiotemporäre Objekte in Netzwerken speichern kann. Es werden dabei vorallem die sogenannte Projektionsregel und verschiedene Arten der Hebb'schen Lernregel betrachtet. Bezüglich der Hebb'schen Lernregel sind einige Literaturstellen zu nennen, an die hier angeknüpft wird: [16], [5], [15] und [32]. Die Projektionsregel wird in [19] erwähnt. Die Anwendung auf Netzwerke mit mehreren Delays wird in [15] angeregt, wurde aber soweit uns bekannt ist, nicht weiter untersucht.

Es wird auch auf die Speicherkapazität eingegangen. Die ausgeführten Analysen bezüglich der Hebb'schen Lernregel sind in ähnlicher Art in [12] und [25] zu finden. Die Abschätzung, die in Anhang A.1 ausgeführt wurde, ist in dieser Art neu.

In Kapitel 6 wird dann versucht diese Konstruktionsprinzipien auf Netzwerke zu übertragen, deren Neuronen biologisch realistischer sind als jene die in den vorangegangenen Kapiteln betrachtet werden. Es werden dort Simulationen dieser Netzwerke mit dem Softwarepaket GENESIS durchgeführt (vgl. [7]). Simulationen dieser Art, vor allem jene wo die Anwendung der Projektionsregel untersucht wird, wurden, soweit uns bekannt ist, bisher nicht durchgeführt.

Kapitel 2

Grundlagen

In diesem Kapitel soll kurz die Funktionsweise von biologischen Neuronen erläutert werden und darauf aufbauend die in dieser Arbeit verwendeten Modellneuronen beschrieben werden.

2.1 Neuronen, Synapsen und Netzwerke

Das zentrale Nervensystem besteht aus einer Vielzahl von Nervenzellen oder Neuronen, die in einem komplizierten Schema von anatomisch unterscheidbaren Schichten und Arealen angeordnet und miteinander verbunden sind (vgl. zum Beispiel [20]). Im allgemeinen wird angenommen, daß die für die höheren geistigen Leistungen notwendigen neuronalen Strukturen sich in der Großhirnrinde (Kortex) befinden. Dieses wohl am interessantesten neuronale Netzwerk, mit seinen unglaublichen Berechnungsfähigkeiten, besteht aus ca. 10^{13} Neuronen, die in einer schichtartigen Struktur angeordnet sind und bestimmte Verbindungsmuster aufweisen [1]. Man kann auch zwischen verschieden Typen von Neuronen unterscheiden. Die zwei wichtigsten Gruppen stellen die *Pyramidenzellen* und die *Sternzellen* dar. Die Pyramidenzellen, die mit ca. 70% den Hauptanteil des Kortex ausmachen, haben sowohl kurz- als auch weitreichende Verbindungen und wirken im allgemeinen erregend (*exzitatorisch*) auf nachgeschaltete Neuronen, d.h. die Wahrscheinlichkeit, daß das nachgeschaltete Neuron ein Aktionspotential abgibt, (es *feuert*) wird erhöht. Eine Pyramidenzelle wird im Schnitt von ca. 10^4 vorgeschalteten Neuronen kontaktiert. Um diese enorme Menge von Kontakten unterzubringen, besitzt die Pyramidenzelle eine große Oberfläche, die hauptsächlich durch den sogenannten Dendritenbaum gebildet wird. Die Sternzellen hingegen wirken hemmend (*inhibitorisch*) auf ihre nachgeschalteten Neuronen und stellen ca. 25% der Neuronenpopulation im. In Abbildung 2.1 ist ein Ausschnitt aus dem Kortex dargestellt.

Abbildung 2.1: Die linke Seite zeigt die zwei Haupttypen von Zellen im Kortex, Pyramidenzellen (P) und Sternzellen (S), mit ihrer groben Zuordnung zu den sechs Schichten und benennt die wichtigsten morphologischen Teile. Die rechte Darstellung zeigt einen Ausschnitt des Kortex in drei verschiedenen Arten von Färbungen. Der linke Bereich stellt mit Hilfe einer Silberimpregnierung die groben Umrisse der einzelnen Zellen dar. Im mittleren Bereich sind nur die Zellkörper (Soma) dargestellt, während der rechte Abschnitt die Organisation der Nervenfasern darstellt (aus [1]).

2.1.1 Das Neuron

Anatomisch gesehen besteht ein Neuron aus dem Zellkörper oder *Soma*, den *Dendrit* und dem *Axon*, wie dies in Abbildung 2.1 dargestellt ist. Im allgemeinen gelangen über die Dendriten die Signale der vorangeschalteten (präsynaptischen) Neuronen zum Soma. Hier wird, in erster Näherung, genau dann ein Aktionspotential generiert, wenn das eintreffende Signal eine gewisse Schwelle (threshold) überschreitet. Dieses Aktionspotential wird dann entlang des Axons zu vielen weiteren nachgeschalteten (postsynaptischen) Neuronen weitergeleitet.

Der zentrale Punkt der Informationsverarbeitung eines Neurons besteht nun also in der Generierung eines Aktionspotentials in Abhängigkeit der vielen Eingangssignale. Im folgenden werden kurz die wichtigsten biophysikalischen Begriffe erläutert, die zum Verständnis dieses fundamentalen Mechanismus notwendig sind.

Das Neuron ist von einer semipermeablen Zellmembran umgeben. Die Flüssigkeit außerhalb (innerhalb) der Zelle wird als extrazellulär (intrazellulär) bezeichnet. Diese unterscheiden sich hauptsächlich dadurch, daß die vorkommenden Ionen verschiedene Konzentrationen aufweisen. Aufgrund verschiedener Mechanismen stellt sich zwischen dem extrazellulären und intrazellulären Raum eine Potentialdifferenz ein, diese wird als *Membranpotential* bezeichnet.

Die Zellmembran übernimmt viele funktionelle Aufgaben. Im Zusammenhang mit der Generierung eines Aktionspotentials ist vor allem die Steuerung der Ionenkonzentration im Zellinneren wichtig, die durch das Schließen und Öffnen von ionenspezifischen Kanälen bewerkstelligt wird. Es gibt verschiedene Arten solcher Kanäle.

Passive Kanäle Diese Kanäle sind immer für eine oder mehrere spezifische Ionenarten durchlässig und sind für das Zustandekommen des sogenannten *Ruhemembranpotentials* verantwortlich.

Aktive Kanäle Dies sind Kanäle, die in Abhängigkeit von der momentanen Membranspannung und oder der Inonekonzentration von bestimmten Ionen (z.B. Ca^{++}) im Zellinneren ihre Permeabilität ändern können. Kanäle dieser Art sind für das Aktionspotential verantwortlich.

Ionenpumpen Diese energieverbrauchenden Kanäle sorgen dafür, daß im Zellinneren immer die korrekten Ionenkonzentrationen vorhanden sind.

Betrachtet man nun ein Neuron im Ruhezustand (ohne jegliche Eingangssignale) so stellt man fest, daß sich im Inneren der Zelle eine Potentialdifferenz von etwa $-70\,\mathrm{mV}$ gegenüber der die Zelle umgebenden Flüssigkeit einstellt; dies stellt das oben erwähnte Ruhemembranpotential dar. Dies ist auf die ionenspezifische Permeabilität der Zellmembran, den sehr unterschiedlichen Inenkonzentrationen zwi-

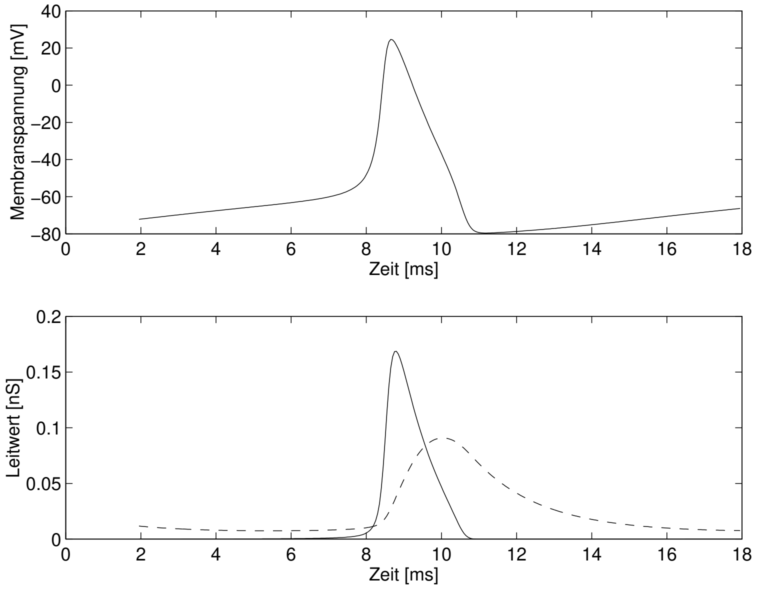


Abbildung 2.2: Darstellung des zeitlichen Verlaufes eines Aktionspotentials. Berechnet nach dem Hodgkin and Huxley Modell (vgl. Anhang B) des Axons eines Riesentintenfisches. Oberes Diagramm: Zeitlicher Verlauf der Membranspannung am Soma. Unteres Diagramm: Zeitlicher Verlauf der Leitfähigkeiten für Na^+ (durchgezogene Linie) und K^+ (gestrichelte Linie).

schen Zellinnerem und Zelläußerem und der Aktivität der Ionenpumpen zurückzuführen. Die zwei wichtigsten Arten von Ionen, die daran beteiligt sind, sind Kalium (K^+) und Natrium (Na^+), wobei sich im Inneren sehr viel mehr K^+ und sehr viel weniger Na^+ als außerhalb der Zelle befindet (vgl. z.B. [29, 18]).

Durch Signale anderer Neuronen, die über Synapsen und den Dendritenbaum, oder künstlich durch eine Mikroelektrode, an das Soma herankommen, wird nun die Membranspannung verändert. Hierbei bezeichnet man eine Änderung in Richtung negativerer Potentiale als *Hyperpolarisation* und andernfalls als *Depolarisation*.

Wird nun die Zelle auf ca. $-50\,\mathrm{mV}$ depolarisiert, so kommt es durch die Aktivität der K^+ und Na^+ Kanäle zur Ausbildung eines Spannungsimpulses mit einer Spitze von ca. $+20\,\mathrm{mV}$. Dieser Impuls, Aktionspotential genannt, ist in Abbildung 2.2 dargestellt. Den qualitativen Verlauf der Änderung der Leitfähigkeit für K^+ und Na^+ kann man ebenfalls dieser Abbildung entnehmen (vgl. [29]).

Neben den aktiven Kanälen für K^+ und Na^+ existieren noch andere, von denen die Kanäle für Ca^{++} auch noch eine wichtige Rolle spielen. Außerdem wurde bisher nur auf die Abhängigkeit von der Membranspannung eingegangen. Man

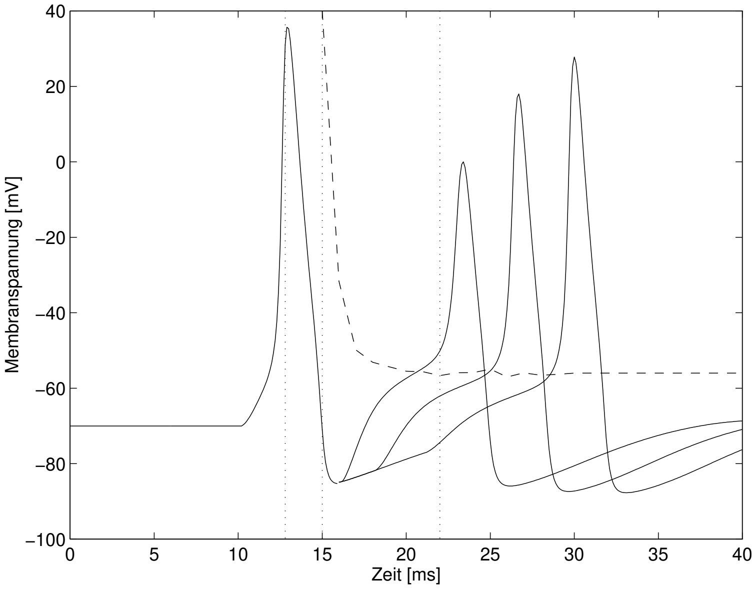


Abbildung 2.3: Darstellung des Refraktärverhaltens eines Hodgkin and Huxley Neurons. In der Zeit zwischen 12.8 ms und 15 ms ist das Neuron absolut refraktär, danach sinkt die Schwelle wieder ziemlich rasch ab. Die drei weiteren Aktionspotentiale sind durch Anregung des Neurons bei 16 ms, 18 ms und 20 ms, also in der relativen Refraktärzeit, entstanden. Die strichlierte Linie stellt den Schwellwert dar. Dieses Diagramm wurde mit dem Simulator GENESIS (vgl. [7]) erstellt.

weiß jedoch, daß einige Kanäle auch von Ca^{++} abhängig sind und daß daraus verschiedene Feuercharakteristika entstehen können (vgl. Kapitel 7 in [7]). Weiters sind diese Kanäle auch mitverantwortlich für verschiedene *Refraktäreigenschaften* von Neuronen. Darunter versteht man das Verhalten eines Neurons gegenüber Eingangssignalen, nachdem es bereits gefeuert hat. Den meisten Neuronen ist gemein, daß sie für eine gewisse Zeit nach einem Aktionspotential auch durch beliebig starke Stimuli nicht wieder zum Feuern gebracht werden können. Diese Phase bezeichnet man als *absolute Refraktärzeit*. Daran schließt sich meist eine sogenannte *relative Refraktärzeit*, in der das Neuron nur mit stärkeren Anregungen als im Ruhezustand zum Feuern gebracht werden kann. Man kann sich dies auch so vorstellen, daß während der absoluten Refraktärzeit die Schwelle des Neurons unendlich ist, und dann in einer gewissen Form wieder abfällt (vgl. Abbildung 2.3).

2.1.2 Axonale Fortleitung des Aktionspotentiales

Nachdem, wie oben bescrieben wurde, ein Aktionspotential am Soma des Neurons generiert wurde, breitet sich dieses entlang des Axons aus. Betrachtet man diesen Vorgang etwas genauer, so stellt man fest, daß das Axon aktive Kanäle enthält, die dafür sorgen, daß das Aktionspotential in seiner Form, daß heißt als relativ enger Spannungsimpuls, immer gleich aussieht. Dies könnte ebenfalls darauf hindeuten, daß der Zeitpunkt eines Aktionspotentials wichtiger ist, als die mittlere Feuerrate. Je nach Art des Axons breitet sich das Aktionspotential mit verschiedenen Geschwindigkeiten aus. Faktoren die dies beeinflussen sind der Durchmesser, chemische Eigenschaften und vor allem ob das Axon mit einer sogenannten *Myelinschicht* überzogen ist oder nicht. Diese Geschwindigkeiten reichen bei unmyelinisierten Axonen von weniger als 1 m/s bis zu 20 m/s (Axon des Riesentintenfisches mit 1 mm Durchmesser). Die myelinisierten Fasern hingegen erreichen Leitungsgeschwindigkeiten bis zu 100 m/s (Nervenfasern die zu Skelettmuskeln führen). Eine detailliertere Ausführung ist z.B. [29] oder [30] zu entnehmen.

2.1.3 Die Synapse

Bis jetzt haben wir gesehen, wie ein Neuron auf Eingangssignale reagiert, nun soll erläutert werden, wie diese Eingangssignale entstehen und welche typischen Formen sie haben. Die Stelle an der ein Axon eines vorgeschalteten Neurons auf ein nächstes trifft, nennt man *Synapse*. Eine vereinfachte Darstellung einer solchen Synapse ist in Abbildung 2.4 zu sehen.

In vereinfachter Form funktioniert eine Synapse folgender Maßen. Erreicht ein Aktionspotential die präsynaptische Nervenendigung, so wird dort ein chemischer Überträgerstoff freigesetzt, *Neurotransmitter* genannt. Dieser diffundiert dann durch den *synaptischen Spalt* zur postsynaptischen Zelle. Dort angelangt, bindet sich der Neurotransmitter an sogenannte Rezeptoren, die nun, wenn sie von einem solchen Botenstoff besetzt sind, die Eigenschaften der Membran lokal verändern. Es werden Ionenkanäle geöffnet, und somit wird das postsynaptische Membranpotential geändert. Je nach Art der Kanäle, die durch die Rezeptoren geöffnet werden, kann dies zu einer (öffnen von Na^+-Kanälen) oder (öffnen von K^+ und Cl^--Kanälen) führen. Eine solcherart entstandene Änderung des Membranpotentials wird *postsynaptisches Potential*, kurz PSP , genannt. Je nach dem ob nun dieses PSP dazu beiträgt das Neuron näher an die Abgabe eines Aktionspotentials heranzubringen oder nicht, unterscheidet man zwischen EPSP (exzitatorisches PSP) und IPSP (inhibitorisches PSP), beziehungsweise zwischen exzitatorischen und inhibitorischen Synapsen . Zwei solche PSP's mit typischen Werten sind in Abbildung 2.4 dargestellt.

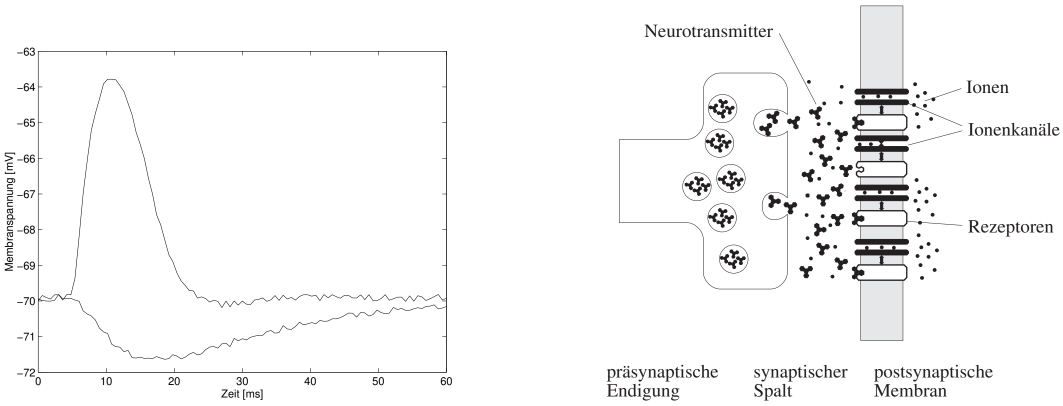


Abbildung 2.4: Im linken Teil der Abbildung ist eine EPSP und IPSP dargestellt. Der rechte Teil der Abbildung zeigt schematisch die Funktionsweise einer Synapse. Erklärung siehe Text.

2.1.4 Integration der postsynaptischen Potentiale

Ein Neuron besitzt im allgemeinen sehr viele über die ganze Oberfläche verteilte Synapsen. Wie summieren sich nun die einzelnen PSP's auf und in welcher Form erreichen sie das Soma, um eventuell ein Aktionspotential auszulösen? Diese Frage wurde schon sehr bald in Angriff genommen und wird in Bezug auf rein passive Dendritenbäume, das heißt sie enthalten keine aktiven Ionenkanäle, relativ gut verstanden (vgl. Kapitel 6 in [7] und [26]).

Qualitativ kann man sagen, daß ein PSP auf dem Weg von der Synapse zum Soma abgeschwächt wird und daß sein zeitlicher Verlauf breiter wird. Zwei PSP überlagern sich in einer prinzipiell nichtlinearen Art und Weise. Je weiter zwei Synapsen räumlich entfernt sind, umso linearer ist die Überlagerung der durch sie erzeugten PSP's.

Ein besonders nichtlinearer Effekt kann auftreten, wenn eine Synapse sehr viele Cl^--Kanäle öffnet. Es kommt dann zu einem „Kurzschluß" der Membran an dieser Stelle und eine EPSP wird durch eine solche Synapse sehr viel mehr verkleinert. Dieser Effekt wird als *shunting inhibition* bezeichnet (vgl. Kapitel 13 [18]).

2.1.5 Synaptische Plastizität

Für eine gegebene Synapse ist der zeitliche Verlauf und damit die Form des PSP, sei es exzitatorisch oder inhibitorisch, festgelegt. Es gibt jedoch viele Hinweise darauf, daß sich die Effektivität einer Synapse verändern kann. Das heißt, die Amplitude des PSP kann sich aufgrund verschiedener Mechanismen verändern. Allgemein ist man der Meinung, daß in diesen Mechanismen unsere Fähigkeit zum

Lernen begründet liegt. Ein Überblick über solche Mechanismen ist in Kapitel 15 in [18] zu finden.

Ein solches Phänomen möchte ich hier erwähnen. Experimente haben gezeigt, daß eine gleichzeitige Aktivierung des post- und präsynaptischen Neurons dazu führt, daß die Effektivität der betroffenen Synapse für längere Zeit gesteigert werden kann. Diese Experimente wurden an exzitatorischen Synapsen im Hippocampus durchgeführt.

Diese Versuche untermauern ein theoretisches Postulat, das *Donald Hebb* bereits im Jahre 1949 so formulierte:

> *When an axon of cell A is near enough to exite cell B and repeteatly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.*

Die Bedeutung des Hebb'schen Postulats und seine experimentellen Untermauerung liegt darin, daß dadurch eine Möglichkeit zur assoziativen Speicherung von häufig vorkommenden oder „gelernten“ Systemzuständen ermöglicht wird.

2.2 Modelle für Netzwerke feuernder Neuronen

Im vorigen Abschnitt wurden die grundlegenden Mechanismen der neuronalen Informationsverarbeitung in biologischen Netzwerken erläutert. Es sollen nun Modelle vorgestellt werden, die folgende Aspekte berücksichtigen können.

1. Die zeitliche Auflösung der Aktionspotentiale anstatt einer mittleren Aktivität eines Neurons.
2. Die endliche Ausbreitungsgeschwindigkeit eines Aktionspotentials am Axon, das heißt Verzögerungszeiten der Signalübertragung zwischen den Neuronen.
3. Eine realistische Modellierung der Auswirkung eines Aktionspotentials auf die postsynaptische Membran z.B. die Form der EPSP und IPSP.
4. Entstehung eines Aktionspotentials wenn das Membranpotential eine bestimmte Schwelle überschreitet.
5. Die Refraktäreigenschaften eines Neurons.

Durch diese 5 Aspekte wird das Auftreten eines Aktionspotentials, im folgenden auch als Impuls bezeichnet, zu einem bestimmten Zeitpunkt als das zentrale Ereignis in den Mittelpunkt gestellt. Dieses Ereignis wird eben im Fachchargon als das „Feuern“ eines Neurons bezeichnet, daraus begründet sich die Bezeichnung *Netzwerke feuernder Neuronen* (NFN) oder im Englischen *Spiking Neuron Network* (SNN).

2.2.1 Formale Definition eines NFN

Ein NFN $\mathcal{N}$ besteht aus

- einem endlichen gerichteten Graphen $\langle V, E\rangle$, wobei die Elemente von $V := \{1, 2, \ldots, N\}$ als Neuronen und die Elemente aus E als Synapsen bezeichnet werden,
- einer Teilmenge $V_{in} \subset V$ von *Inputneuronen*,
- einer Teilmenge $V_{out} \subseteq V$ von *Outputneuronen*,
- einer *Schwellwertfunktion* $\Theta_i : \mathbb{R}^+ \to \mathbb{R} \cup \{\infty\}$ für jedes Neuron $i \in V \setminus V_{in}$ ($\mathbb{R}^+ := \{x \in \mathbb{R} : x \geq 0\}$),
- einer *Antwortfunktion* $\varepsilon_{i,j}^{(d)} : \mathbb{R}^+ \to \mathbb{R}$ für jede Synapse $\langle j, i\rangle^{(d)} \in E$,
- und einer *Gewichtsfunktion* $w_{i,j}^{(d)} : \mathbb{R}^+ \to \mathbb{R}$ für jede Synapse $\langle j, i\rangle^{(d)} \in E$.

Die Antwortfunktionen $\varepsilon_{i,j}^{(d)}$ modellieren den zeitlichen Verlauf der Auswirkungen eines Impulses, der von Neuron j ausgeht und über die Synapse $\langle j, i\rangle^{(d)}$ zu Neuron i gelangt, auf dessen Membran. In den meisten Fällen wird dies das PSP sein, wie es am Soma von Neuron i durch diesen Impuls resultiert. Man kann sich jedoch auch vorstellen, daß $\varepsilon_{i,j}^{(d)}$ die Leitwertänderung in der Membran von Neuron i darstellt. Mögliche Formen für solche Antwortfunktionen sind in Abbildung 2.4 dargestellt. Der Index (d) kennzeichnet, daß prinzipiell mehrere Verbindungen zwischen zwei Neuronen vorhanden sein können.

Die zu $\varepsilon_{i,j}^{(d)}$ korrespondierende Gewichtsfunktion $w_{i,j}^{(d)}$ beschreibt, wie effektiv ein Impuls sich auswirkt. Für den Fall, daß durch $\varepsilon_{i,j}^{(d)}$ ein PSP modelliert wird, wird man unter dem Ausdruck $w_{i,j}^{(d)}\varepsilon_{i,j}^{(d)}$ das tatsächlich am Soma resultierende PSP verstehen.

Der Sinn der Schwellwertfunktion ist es, das Refraktärverhalten eines Neurons modellieren zu können. Ein Beispiel eines Refraktärverhaltens ist in Abbildung 2.3 dargestellt.

Man bezeichnet nun mit $F_i \subseteq \mathbb{R}^+$, $F_i := \left\{t_i^{(1)}, t_i^{(2)}, t_i^{(3)}, \ldots\right\}$ und $t_i^{(s)} < t_i^{(s+1)}$ jene geordnete Menge, die alle Zeitpunkte enthält, zu denen Neuron i feuert. Im weiteren werden diese Mengen als Impulsfolgen (spike trains) bezeichnet. Den Inputneuronen weißt man einfach fixe, vordefinierte Impulsfolgen zu, welche man als den Input des Netzwerkes betrachtet. Die Impulsfolgen der Outputneuronen werden, wie schon der Name nahelegt, als der Output des Netzwerkes definiert. Diese und alle weiteren Impulsfolgen F_i mit $i \in V \setminus V_{in}$ sind durch die Berechnungsvorgänge im Netzwerk bestimmt.

$$\begin{aligned} t_i^{(1)} &:= \inf\{t \in T : h_i(t) \geq \Theta_i(0)\} \\ t_i^{(s+1)} &:= \inf\left\{t \in T : t > t_i^{(s)} \quad \text{und} \quad h_i(t) \geq \Theta_i\left(t - t_i^{(s)}\right)\right\} \end{aligned} \tag{2.1}$$

In dieser Gleichung beschreibt T die Menge der potentiellen Feuerzeiten. Normalerweise wird diese Menge immer mit $\mathbb{R}^+$ gleichgesetzt. Die Funktion $h_i(t)$ beschreibt den zeitlichen Verlauf des Membranpotentials am Soma. Wie nun diese Funktion tatsächlich aussieht hängt von der Form der Antwortfunktionen und der gewählten Form der dendritischen Integration ab.

2.2.2 Ein lineares Modellneuron

Dies ist eine sehr übliche Annahme über die dendritische Integration. Betrachtet man die Form eines typischen EPSP oder IPSP (Abbildung 2.4), so sieht man, daß diese vom Membranruhepotential ausgehen, ein Maximum von einigen mV über oder unter diesem annehmen und schließlich wieder zum Ruhepotential zurückkehren. Da der Wert des Ruhemembranpotentials eine Frage der Definition des Bezugspunktes ist, kann man es genausogut zu 0 mV definieren. Das heißt, daß eine Funktion $\varepsilon_{i,j}^{(d)}(t)$ die Abweichung vom Ruhepotential darstellt. Mit der Schwellwertfunktion ist es genauso so. $\Theta_i(t)$ stellt die Schwelle relativ zum Ruhepotential dar. Aus diesen Überlegungen heraus und unter der Annahme der linearen Integration am Dendriten , kann man für den zeitlichen Verlauf des Membranpotentials folgende Formel niederschreiben.

$$h_i(t) := \sum_{j,d:\langle i,j\rangle^{(d)} \in E} \; \sum_{t_j^{(s)} \in F_j} w_{i,j}^{(d)}(t) \cdot \varepsilon_{i,j}^{(d)}(t - t_j^{(s)}) + h_i^{ext}(t) \tag{2.2}$$

Der Term $h_i^{ext}(t)$ stellt mögliche externe Stimuli dar, welche zum Beispiel mit einer Mikroelektrode eingebracht werden.

2.3 Kompartmentmodelle von Neuronen

Diese Art der Modellierung der Berechnungsvorgänge in Neuronen gewinnt in letzter Zeit immer mehr an Bedeutung, da sie sehr gut geeignet ist, Modelle zu erhalten, die man mit Computern simulieren kann [7]. Die prinzipiellen Ideen, die hinter diesem Modellierungsansatz stehen sind, folgende (Kapitel 4,5 und 6 in [7]), (Kapitel 3 in [22]).

- Die wichtigsten für die Informationsverarbeitung notwendigen Vorgänge laufen in den Strukturen der Zellmembran ab.
- Man teilt das betrachtete Neuron in lauter kleine Teilstücke, von denen man dann annimmt, daß sie räumlich homogene biophysikalische Eigenschaften haben.

Diese kleinen Stücke oder *Kompartments* werden dann mit Hilfe von elektrischen Ersatzschaltbildern beschrieben, die durch die Kabeltheorie motiviert sind, welche erstmals 1965 auf Neuronen angewendet wurde [26].

Im Prinzip ersetzt der Kompartment-Ansatz die kontinuierliche Kabelgleichung (eine partielle Differentialgleichung [26]) durch eine Menge von einfachen Differentialgleichungen. Die Annahme dabei ist, daß, wenn man die Kompartments klein genug wählt, man auch einen sehr kleinen Fehler gegenüber der Kabelgleichung macht.

In Abbildung 2.5 ist schematisch der Übergang vom realen Neuron auf ein aus Kompartments gebildetes Neuron dargestellt. Der Vorteil dieser Art der Modellierung von Neuronen liegt darin, daß man keinerlei Einschränkungen der Membraneigenschaften in einem Kompartment unterliegt. Kompartments können die Membran eines Stückes des Dendritenbaumes genausogut darstellen, wie die Membran eines Axons; ebenso kann man sehr einfach den Einfluß synaptischer Aktivität abbilden. Zusätzlich können beliebig komplizierte dendritische oder axonale Verzweigungen oder andere morphologische Unregelmäßigkeiten, wie zum Beispiel Spines, einfach durch die topologische Anordnung der Kompartments modelliert werden. Ebenso kann die Genauigkeit, mit der man das betrachtete Neuronale System abbilden will in einem großen Bereich gewählt werden. Man kann mit einem Kompartment eine ganze Zelle modellieren, wenn es darum geht, Effekte die durch die Netzwerkstruktur entstehen, zu untersuchen, wie dies in dieser Arbeit der Fall sein wird (Abschnitt 6). Andererseits kann man sehr exakte Einzelzellmodelle erzeugen, um die Berechnungsvorgänge innerhalb einer Zelle abzubilden (vgl. Abschnitt 2.1.1 [7]).

2.3.1 Mathematische Formulierung

Ersatzschaltbild eines Kompartments

Ein Ersatzschaltbild, wie es in Abbildung 2.6 gezeigt ist, stellt die Basis für alle Kompartmentmodelle dar. Die einzelnen Zweige beschreiben die verschiedenen Eigenschaften der Zellmembran, welche von den unterschiedlichen Arten von Kanälen und den Synapsen herrühren (vgl. Abschnitt 2.1.1 und Abschnitt 2.1.3). Eine wichtige Annahme, die bei den meisten Kompartmentmodellen gemacht wird, ist jene, daß man sagt, die gesamte extrzelluläre Flüssigkeit ist, in Bezug

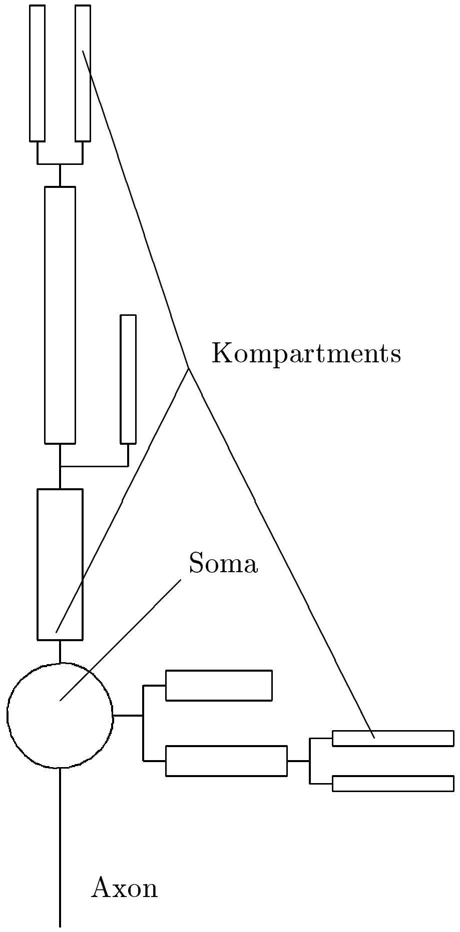


Abbildung 2.5: Darstellung des Übergangs von einem realen Neuron auf ein Kompartmentmodell. Die Länge und Dicke der einzelnen Segmente soll so gewählt werden, daß sie als Isopotentialabschnitte betrachtet werden können. Jedes dieser Segmente wird dann durch ein Ersatzschaltbild entsprechend Abbildung 2.6 ersetzt und wie in Abbildung 2.7 angedeutet zusammengeschaltet.

auf das Zellinnere, auf dem selben Potential (isopotential). Man wählt daher dieses als Bezugspotential; in Abbildung 2.6 ist die extrazelluläre Flüssigkeit als Erdungssymbol dargestellt.

Der Kapazitive Zweig in der Ersatzschaltung resultiert aus der Beobachtung, daß die Ionen, der extra- und intrazellulären Flüssigkeit, durch die für sie nur teilweise durchlässige Membran, getrennt werden, und diese dadurch eine Kapazität darstellt. Der Strom durch die Kapazität ist durch folgenden Zusammenhang gegeben.

$$i_C(t) = C_m \frac{du_m(t)}{dt} \tag{2.3}$$

Zusammen mit dem nächsten Zweig werden dadurch die passiven Eigenschaften

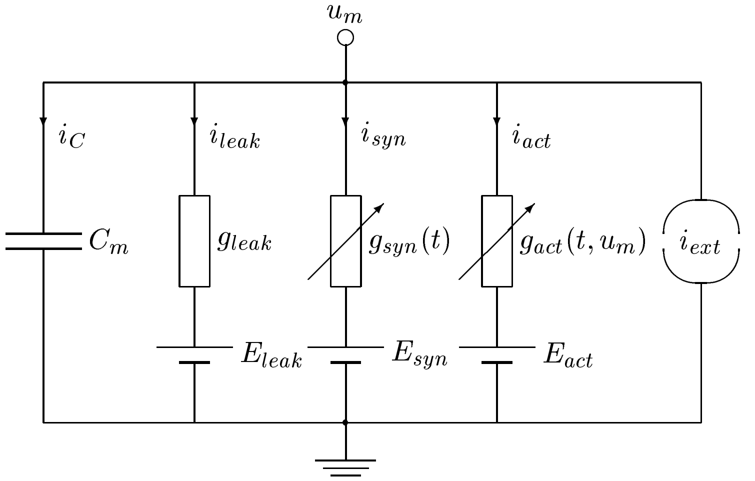


Abbildung 2.6: Ersatzschaltung eines Isopotentialabschnitts eines Stücks einer Zellmembran. Dargestellt sind die wichtigsten drei Arten von Kanälen. Der linke Leitwertzweig stellt mit dem konstanten Leitwert g_{gleak} und der Batterie E_{leak} die passiven Kanäle dar. Synaptische Kanäle werden durch den mittleren Zweig dargestellt, welcher aus einem zeitlich veränderlichen Leitwert $g_{syn}(t)$ in Serie mit der, das Umkehrpotential darstellenden, Batterie E_{syn} besteht. Aktive Kanäle werden durch den rechten Zweig dargestellt. Ein zeit- und spannungsabhängiger Leitwert $g_{act}(t, u_m)$ ist in Serie mit einer Batterie E_{act} geschaltet, welche das Umkehrpotential der beteiligten Ionen darstellt (z.B. Na^+, K^+, Ca^{++}).

der Membran dargestellt. Die Spannungsquelle E_{leak} in Serie mit dem Leitwert g_{leak} stellen jene Kanäle dar, die unabhängig von der Zeit und der Membranspannung u_m eine fixe Größe darstellen. E_{leak} stellt dabei das Umkehrpotential der durch diese Kanäle fließenden Ionen dar (vgl. [18], [29]). $R_{leak} = 1/g_{leak}$ ist dabei jener elektrische Widerstand, der sich diesen Ionen quer zur Membran bietet. Der Strom durch diesen Zweig des Kompartment hängt einfach über das Ohmsche Gesetz mit der Membranspannung zusammen.

$$i_{leak}(t) = g_{leak}(u_m(t) - E_{leak}) \tag{2.4}$$

Eine weitere sehr wichtige Art von Kanälen, sind jene, die durch die Synapsen aktiviert werden. Diese Kanäle werden am einfachsten durch einen zeitabhängigen Leitwert $g_{syn}(t)$ in Serie mit der treibenden elektrischen Kraft der beteiligten Ionen E_{syn} dargestellt. Eine sehr gebräuchlich Form für den zeitlichen Verlauf dieser Leitwertänderung ist die sogenannte *alpha*-Funktion.

$$g_{syn}(t) := g_{max} \frac{t}{\tau_p} e^{(1-t/\tau_p)} \tag{2.5}$$

Diese Funktion erreicht nach τ_p das Maximum von g_{max} und fällt dann wieder auf Null ab. Der zeitlich Verlauf des Stromes durch diesen Zweig ist damit gegeben durch:

$$i_{syn}(t) = g_{syn}(t)(u_m(t) - E_{syn}) \tag{2.6}$$

Der nächste Zweig in Abbildung 2.6 stellt die Klasse der aktiven Kanäle dar. Er besteht aus einer Spannungsquelle E_{act} in Serie mit einem zeit- und spannungsabhängigen Leitwert $g_{act}(t, u_m)$. Je nach Art der Funktion $g_{act}(t, u_m)$ können durch Zweige dieser Art jene Kanäle modelliert werden, die für die Erzeugung von Aktionspotentialen Zuständig sind (vgl. Anhang B), oder gewisse andere Phänomene, wie z.B. subthreshold Schwingungen des Membranpotentials modelliert werden (vgl. Kapitel 7 in [7]). Der Strom durch diesen Zweig ergibt sich wie oben zu:

$$i_{act}(t) = g_{act}(t, u_m)(u_m(t) - E_{act}) \tag{2.7}$$

Durch die ganz rechts dargestellte Stromquelle i_{ext} soll angedeutet werden, wie Ströme, die von außen in die Zelle eingebracht werden in das Modell eingehen.

Der gesamte Strom durch das Membranstück, welches durch ein solches Kompartment dargestellt wird, ist einfach die Summe der Ströme durch die einzelnen Zweige.

$$i_m(t) = C_m \frac{du_m(t)}{dt} + i_{ion}(t) + i_{ext}(t) + g_{leak}(u_m(t) - E_{leak}) \tag{2.8}$$

$$i_{ion}(t) := \sum g_{syn}(t)(u_m(t) - E_{syn}) + \sum g_{act}(t, u_m)(u_m(t) - E_{act}) \tag{2.9}$$

Durch die Summenzeichen in Gleichung 2.9 soll angedeutet werden, daß von jeder dieser Art von Kanälen mehrere pro Kompartment vorhanden sein können.

Zusammenschaltung mehrerer Kompartments

Um eine gesamte Zelle zu modellieren, muß man mehrere der oben beschriebenen Kompartments zusammenschalten. Für eine Kette von drei Kompartments ist dies in Abbildung 2.7 dargestellt.

Der Widerstand R_a stellt dabei den Längswiderstand des Kompartments dar. Der zeitliche Verlauf der Membranspannung hängt nun nicht mehr nur von den Kanälen des einzelnen Kompartments ab, sondern auch von denen der benachbarten.

Der Strom $i_m^j(t)$ durch das Kompartment j (vgl. Abbildung 2.7) ist laut dem ersten Kirchoff'schen Gesetz gleich der Differenz der beiden Längsströme $i_{j,j-1}$ und $i_{j+1,j}$.

$$i_m^j(t) = i_{j,j-1} - i_{j+1,j} \tag{2.10}$$

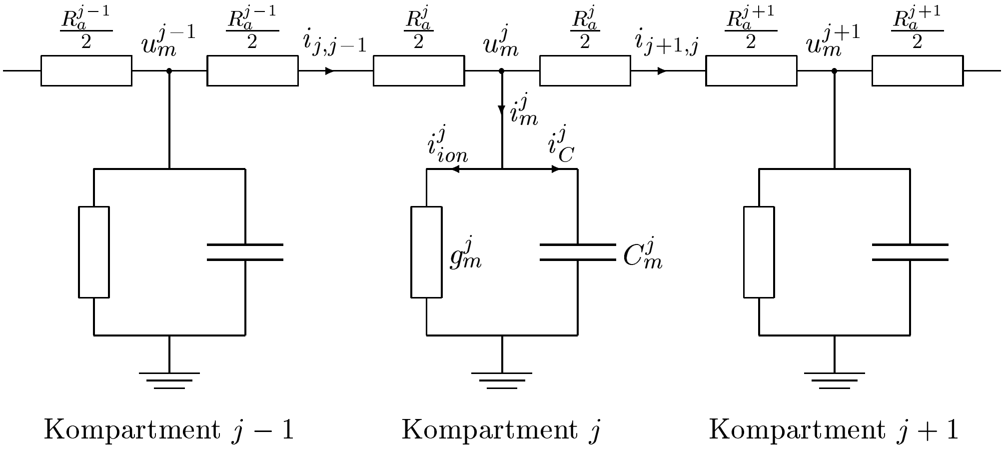


Abbildung 2.7: Ersatzschaltung eines Kompartmentmodells für eine Kette aus drei Segmenten. Die diversen Zweige mit den verschiedenen Arten von Leitwerten (vgl. Abbildung 2.6) werden hier symbolisch durch den Leitwert g_m^j dargestellt. Der Längswiderstand eines Segments wird in zwei Hälften aufgeteilt; das Membranpotential wird dazwischen gemessen. Es wird angenommen, daß der Widerstand der extrzellulären Flüssigkeit gleich Null ist und so eine Isopotentialfläche darstellt, welche hier durch die Erdungssymbole gekennzeichnet ist. Der Strom i_{ion}^j ist durch Gleichung 2.9 gegeben.

Abbildung 2.7 zeigt eine Aufteilung des Membranstromes i_m^j in den kapazitiven Anteil und den durch die diversen Kanäle hervorgerufenen Teil, laut Gleichung 2.9. Die Ströme $i_{j,j-1}$ und $i_{j+1,j}$ können mit Hilfe des Ohm‘schen Gesetzes in Abhängigkeit der Potentialdifferenz zwischen den einzelnen Kompartments ausgedrückt werden. Daraus ergibt sich nun folgende gekoppelte Differenzialgleichung für Kompartment j.

$$C_m^j \frac{du_m^j(t)}{dt} + i_{ion}(t) + i_{ext}(t) = \frac{u_m^{j-1}(t) - u_m^j(t)}{R_a^{j,j-1}} - \frac{u_m^j(t) - u_m^{j+1}(t)}{R_a^{j+1,j}} \tag{2.11}$$

Wobei hier $R_a^{j,j-1}$ und $R_a^{j+1,j}$ den Widerstand zwischen den einzelnen Kompartments darstellt.

$$R_a^{j,j-1} = \frac{1}{2}\left(R_a^{j-1} + R_a^j\right) \quad R_a^{j+1,j} = \frac{1}{2}\left(R_a^j + R_a^{j+1}\right) \tag{2.12}$$

Für einen Verzweigungspunkt im Dendritenbaum muß man nur den entsprechenden Längsstrom, der in das zusätzliche Kompartment fließt, in Gleichung 2.10 hinzufügen. Für ein Anfangs- oder Endstück einer Kette von Kompartments muß einfach der entsprechende Längsstrom aus Gleichung 2.10 gestrichen werden.

2.3.2 Zusammenhang mit NFN

Die in diesem Abschnitt behandelte Art der Modellierung stellt nur eine genauere Methode dar, um die verschiedenen Vorgänge in einem einzelnen Neuron zu beschreiben. Die Art und Weise wie diese Neuronen nun aber miteinander kommunizieren, kann sehr gut mit dem in Abschnitt 2.2.1 vorgestellten Modell eines Netzwerkes feuernder Neuronen in Einklang gebracht werden.

Es wurde dort schon darauf hingewiesen, daß eine Antwortfunktion $\varepsilon_{i,j}$ nicht unbedingt ein PSP repräsentieren muß. Es kann auch, wie im Falle von Kompartmentmodellen sehr angebracht, ein zeitlich Veränderlicher Leitwert $g_{syn}(t)$ modelliert werden. Nimmt man die Form, welche in Gleichung 2.5 vorgeschlagen wird, so kann man weiters die maximale Leitwertänderung g_{max} mit der Gewichtsfunktion $w_{i,j}$ identifizieren. Berücksichtigt man weiters eine nicht verschwindende Verzögerungszeit $d_{i,j}$ zwischen zwei Neuronen, so kann man in diesem Fall für $\varepsilon_{i,j}$ schreiben:

$$\varepsilon_{i,j}(t) = \begin{cases} 0 & \text{falls} \quad t < d_{i,j} \\ \frac{t-d_{i,j}}{\tau_p} e^{(1-(t-d_{i,j})/\tau_p)} & \text{falls} \quad t \geq d_{i,j} \end{cases} \tag{2.13}$$

In vielen Fällen wird, wenn Kompartmentmodelle verwendete werden, zur Generierung eines Aktionspotentials das von Hodgkin und Huxley entwickelte Modell (vgl. Anhang B) verwendet, welches im Prinzip aus 2 Arten aktiver Kanäle besteht. Dieses Modell verbirgt aber in sich schon das gesamte Schwellenverhalten. Das heißt, man muß die Funktion $\Theta_i(t)$ des in Abschnitt 2.2.1 definierten Modells nur noch so wählen, daß man damit den durch das Kompartmentmodell generierten Spannungsimpuls als Spike detektiert. Dies kann man einfach so machen, daß man sagt, der Zeitpunkt des Auftretens des Aktionspotentials wird durch das Überschreiten der 0 mV Schwelle der Membranspannung festgelegt. Da aber dieser Spannungsimpuls ca. 0.5 ms über 0 mV bleibt (vgl. Abbildung 2.2), muß man eine künstliche Refraktärzeit im Bereich einer bis zwei ms einführen, um zu verhindern, daß im formalen Modell nicht zwei Spikes auftreten. Summasummarum kann man mit der folgenden Funktion $\Theta_i(t)$ ein durch ein Kompartmentmodell generiertes Aktionspotential, als einzelnen Spike erkennen.

$$\Theta_i(t) = \begin{cases} 0\,\text{mV} & \text{falls} \quad t \in \{0\} \cup [2\,\text{ms}, \infty) \\ \infty & \text{falls} \quad t \in (0, 2\,\text{ms}) \end{cases} \tag{2.14}$$

Dies ist auch jene Vorgehensweise, die in den Simulationen angewandt wird.

2.3.3 Simulationen von Kompartmentmodellen

Hat man nun das Modell aufgestellt, so ist man natürlich an einer Lösung interessiert. Diese besteht darin, daß man für jedes Neuron das gekoppelte Differezialgleichungssystem nach der Membranspannung $u_m^j(t)$ löst. Aufgrund der vielen möglichen nichtlinearen Elemente (g_{syn}, g_{act}) ist im allgemeinen keine analytische Lösung möglich. Es müssen somit numerische Integrationsverfahren herangezogen werden (Kapitel 19 in [7], Kapitel 13 in [22]).

Es gibt bereits eine Reihe von Softwarepaketen, die es gestatten komfortabel Kompartmentmodelle zu erstellen und zu simulieren. In dieser Arbeit wird der am Caltech entwickelte Simulator GENESIS (GEneral NEural SImulation System) verwendet.

Diese Software ist via *anonymous ftp* vom Server *genesis.bbb.caltech.edu* erhältlich (Unterverzeichnis */pub/genesis*). Das Program läuft auf verschiedenen Unix-Plattformen (SUN SOLARIS, SUN OS 4.0 aufwärts, Linux, Irix 4.01 aufwärts, HPUX) und benötigt das X-Windows System (X11R4 aufwärts).

2.4 Zentrale Mustergeneratoren

In dieser Arbeit sollen Netzwerke untersucht werden, die bestimmte spatiotemporäre Muster, oder Objekte generieren. Im Zusammenhang mit NFN versteht man darunter einfach Netzwerke, deren Outputneuronen gewisse vorgegebene Impulsfolgen F_o aufweisen. Das biologische Pendant dazu sind die sogenannten Central Pattern Generators (CPG), oder zentrale Mustergeneratoren genannt.

CPGs kontrollieren die Muskeln, die bei der Ausführung wohldefinierter rhythmischer Bewegungen beteiligt sind, wie zum Beispiel das Atmen, Kauen, Gehen und Schwimmen so wie vieles mehr. Einige Netzwerke, die CPGs repräsentieren, sind anatomisch genau lokalisiert und bestehen aus einer kleinen Anzahl von Neuronen. Ihr Output besteht aus koherenten oszillatorischen Mustern. Diese Eigenschaften machen CPGs zu guten Kandidaten dafür, an ihnen den Zusammenhang zwischen kollektivem Output und den zugrundeliegenden Netzwerkstrukturen zu untersuchen.

Eine Anzahl grundlegender Prinzipien über CPGs resultieren aus Studien verschiedenster rhythmischer Verhaltensweisen [21].

1. Einen rhythmischen Output kann man auch erhalten wenn keine sensorischen Rückkopplungen von den Muskeln und den durch den CPG kontrollierten Strukturen vorliegen. Diese Eigenschaft konnte eindeutig an isolierten Teilen des Rückenmarks gezeigt werden. Die Outputaktivitäten der Motoneuronen in diesen Präparaten sind denen im lebenden Tier sehr ähnlich.

2. Einige CPGs funktionieren ohne sogenannte Schrittmacher-Neuronen (pacemaker). Diese Schrittmacherzellen sind aufgrund ihrer Membraneigenschaften dazu in der Lage, selbständig (ohne jeglichen Input) periodische Impulsfolgen zu erzeugen. Daraus kann man schließen, daß dem Output eines

solchen CPGs gewisse Eigenschaften der Verbindungsstruktur zugrunde liegen. Ein Beispiel für einen solchen CPG ist derjenige, der den Schwimmrhythmus der Seeschnecke *Tritonia diomedea* steuert.

3. Die selben Motoneuronen können an einer Mehrzahl von verschiedenen rhythmischen Verhaltensweisen beteiligt sein. Diese Beobachtung legt die Vermutung nahe, daß ein CPG in der Lage ist, mehrere spatiotemporäre Muster zu erzeugen. Weiters sind Tiere in der Lage, schnell zwischen den einzelnen Bewegungen umzuschalten, oder sie zu kombinieren.

4. Der Output eines CPG kann durch externe Signale moduliert werden. Dies können eventuelle Rückkopplungen aus den kontrollierten Muskeln oder Steuersignale aus höheren Gehirnregionen sein. Durch solche Modulationen können z.B. die einzelnen Muster ausgewählt, sowie deren Frequenz verändert werden.

Es gibt eine Vielzahl von Untersuchungen über CPGs, siehe zum Beispiel Kapitel 8 in [7] oder [21]. Viele davon betrachten einen CPG als ein System von gekoppelten Oszillatoren. Bei diesem Ansatz, wird meist die zeitliche Auflösung der Aktionspotentiale vernachlässigt und das periodische Outputmuster einer Gruppe von Neuronen dazu herangezogen, um eine Phase in einem hochdimensionalen Raum zu definieren. Die Kopplung dieser Oszillatoren, erfolgt in diesem Modell über Funktionen, welche von der Phasenlage der betrachteten Oszillatoren abhängen, und diese dann verändert.

In dieser Arbeit soll nun versucht werden, die einzelnen Impulse nicht zu vernachlässigen. Es sollen Konstruktionsverfahren erläutert werden, welche zu einem gegebenen Outputmuster ein Netzwerk angeben, das in der Lage ist, dieses Muster zu reproduzieren.

2.5 Allgemeines zu spatiotemporären Mustern in NFN

Unter einem spatiotemporären Muster P in einem NFN $\mathcal{N} = \langle V, E\rangle$ versteht man die Impulsfolgen einer bestimmten Menge $V' \subseteq V$, $V' = \{1, \ldots, N\}$ von Neuronen. Meist wird dies die Menge der Outputneuronen sein.

$$P := \left\{ \tilde{F}_o : o \in V_{out} \right\} \tag{2.15}$$

Es wurde hier $\tilde{F}_o$ für die Impulsfolgen der Outputneuronen geschrieben, um darauf hinzuweisen, daß dies die gewünschte Impulsfolge repräsentiert. Diese kann unter Umständen auch nur ein Teil der tatsächlich generierten Impulsfolge F_o sein, oder um eine gewisse Zeit verschoben, in F_o auftreten. Ein Beispiel dafür wäre, wenn das Muster P mit der Periodendauer T wiederholt werden soll. F_o wäre dann eine Aneinandereihung von Wiederholungen von $\tilde{F}_o$ im zeitlichen Abstand von T.

$$F_o = \left\{ t : t = \tilde{t} + k \cdot T,\ \tilde{t} \in \tilde{F}_o,\ k \in \mathbb{N} \cup \{0\} \right\} \tag{2.16}$$

2.5.1 Zeitlich kontinuierliche Muster

Sind die Impulszeiten $t_o^{(s)} \in \tilde{F}_o$ aus der Menge der reellen Zahlen $\mathbb{R}^+$ und unterliegen sie sonst keinerlei Einschränkungen, so wird dieses Muster P als zeitlich *kontinuierliches Muster* bezeichnet.

2.5.2 Zeitlich diskrete Muster

Gilt für die Impulszeiten $t_o^{(s)} \in \tilde{F}_o$ die Einschränkung, daß sie ein ganzzahliges Vielfaches eines bestimmten Intervalls Δ sind, so wird das Muster $P := \left\{ \tilde{F}_o : o \in V_{out} \right\}$ als zeitlich *diskretes Muster* bezeichnet.

Auf den ersten Blick mag dies als eine sehr starke Einschränkung erscheinen. Wählt man jedoch Δ entsprechend klein,so kann man jedes Muster P relativ gut approximieren. Bedenkt man weiters, daß die zeitliche Genauigkeit mit der die Aktionspotentiale im Kortex auftreten, ca. ±1 ms beträgt ([3]), ist diese Art von Impulsfolgen nicht so idealisiert, wie es anfänglich erscheinen mag.

Aufgrund der Eigenschaft der Diskretheit, kann man ein Muster P auch als eine Sequenz S von Vektoren (oder Symbolen) darstellen.

$$\begin{aligned} S &:= \left\{ \xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(l)} \right\} \\ \xi^{(\mu)} &:= \left[\xi_1^{(\mu)}, \xi_2^{(\mu)}, \ldots \xi_N^{(\mu)} \right]^T \\ \xi_o^{(\mu)} &:= \begin{cases} 1 & \text{falls} \quad \mu \cdot \Delta \in \tilde{F}_o \\ 0 & \text{sonst} \end{cases} \end{aligned} \tag{2.17}$$

Als nächstes sollen kurz einige Begriffe definiert werden, die später noch benötigt werden.

Entsprechend der Notation von [35] bezeichnet man eine Sequenz S als *komplex* (*complex sequence*), wenn eine Teilsequenz mehrere male enthalten ist, andernfalls als einfach (*simple sequence*).

Ein Beispiel für eine komplexe Sequenz wäre: $S^{(1)} = \{a - \underline{b - c} - d - \underline{b - c} - e\}$, während $S^{(2)} = \{a - b - c - d\}$ eine einfache Sequenz darstellt. Um beim Generieren einer Sequenz feststellen zu können, welches Symbol (oder wie oben welcher Vektor) nun als nächstes ausgegeben werden soll, muß man ein oder mehrere Symbole vor dem momentanen Symbol kennen. Man bezeichnet nun die Teilsequenz $\mathcal{C}(x)$, die man benötigt, um ein Symbol x vorherzusagen, als den *Kontext* (*context*) von x und die Länge des Kontextes $|\mathcal{C}(x)|$ als den *Grad* (*degree*) von x. Für das Symbol e der Sequenz $S^{(1)}$ wäre der Kontext $\mathcal{C}(e) = \{d{-}b{-}c\}$. Als den *Grad einer Sequenz S* definiert man das Maximum der Grade der einzelnen Symbole.

$$\deg(S) := \max_{x \in S} \{|\mathcal{C}(x)|\} \tag{2.18}$$

Für die obigen Beispiele erhält man mit dieser Definition die Resultate $\deg(S^{(1)}) = 2$ und $\deg(S^{(2)}) = 1$. Man sieht, eine einfache Sequenz hat einen Grad von 1.

Kapitel 3

Zwei einfache CPGs

In diesem einleitenden Abschnitt sollen anhand einer einfachen Konstruktion die wichtigsten Begriffe in Bezug auf ein NFN erläutert werden.

Die Aufgabenstellung, der wir in diesem Kapitel immer wieder begegnen werden, ist folgende. Gesucht ist ein Netzwerk $\mathcal{N} := \langle V, E\rangle$, welches in der Lage ist, eines von M vorgegebenen Mustern $P^{(\nu)} := \left\{\tilde{F}_o : o \in V_{out}\right\}$, $\nu \in \{1, \ldots, M\}$ zu erzeugen. Welches dieser Muster nun gerade erzeugt wird, soll von den Eingängen des Netzwerkes abhängen.

Eine einfache Lösung dieses Problems besteht darin, die zeitliche Komponente in eine räumliche umzuwandeln. Für alle in einem Muster $P^{(\nu)}$ vorkommenden Feuerzeiten $s^{(v)} \in \tilde{F}^{(\nu)}$, $\tilde{F}^{(\nu)} := \bigcup_{o\in V_{out}} \tilde{F}_o^{(\nu)}$ stellt man im Netzwerk $\mathcal{N}$ ein Neuron $v \in V$ zur Verfügung, welches genau zu dieser Zeit feuert. Diese Neuronen v verbindet man dann mit all jenen Outputneuronen, welche zur Zeit $s^{(v)}$ feuern sollen, in der Art, daß dieser eine Impuls von Neuron v genügt, um das Outputneuron zum Feuern zu bringen.

Um ein Feuern der Neuronen v zum vorgesehenen Zeitpunkt zu erreichen, kann man diese in einer „*Delay-Kette*“ anordnen und die Antwortfunktionen entsprechend wählen. Diese Architektur ist in Abbildung 3.1 dargestellt. Die zu einem Muster $P^{(\nu)}$ korrespondierende Delay-Kette wird im folgenden mit $C^{(\nu)}$ bezeichnet.

Eine andere Möglichkeit besteht darin, die benötigten Verzögerungszeiten sternförmig anzuordnen. Dies ist in Abbildung 3.2 dargestellt. Der zu einem Muster $P^{(\nu)}$ korrespondierende „*Delay-Stern*“ wird im folgenden mit $S^{(\nu)}$ bezeichnet.

3.1 Verzögerungszeiten in der Delay-Kette

Seien nun v und $v+1$ zwei Neuronen aus einer zum Muster $P^{(\nu)}$ korrespondierenden Kette $C^{(\nu)}$, die zu den Zeiten $s^{(v)}$ bzw. $s^{(v+1)}$ feuern sollen, so kann durch geeignete Wahl von $\varepsilon_{v+1,v}(t)$, $w_{v+1,v}$ und $\Theta_{v+1}(t)$ dafür gesorgt werden, daß wenn

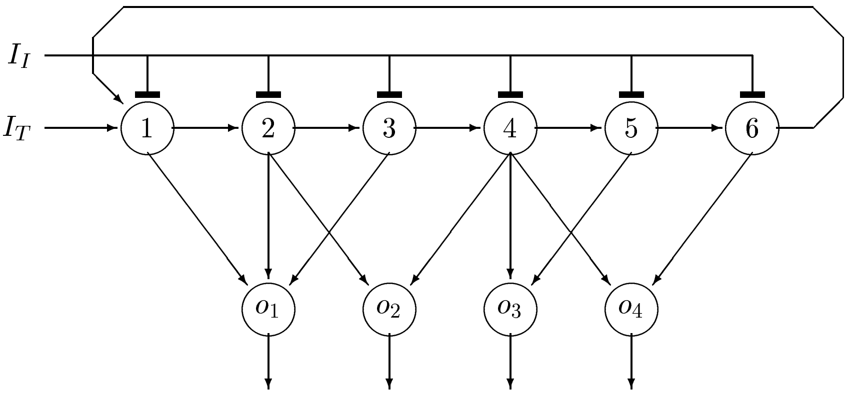


Abbildung 3.1: Darstellung einer Delay-Kette. Die Neuronen 1–6 stellen die Delay-Kette dar. Der Eingang I_T dient dazu, die Kette zu aktivieren. In diesem Bild ist nur die Delay-Kette für ein Muster $P^{(\nu)}$ dargestellt. Weitere Ketten sind einfach „parallel" zu dieser zu schalten. Um eine Überlagerung der einzelnen Muster zu verhindern, könnte eine gerade aktive Kette durch den Eingang I_I, der über inhibitorische Synapsen mit allen Neuronen der Kette verbunden ist, deaktiviert werden. Es ist auch eine Rückkopplung gezeigt, mit der man ein periodisches Muster generieren kann.

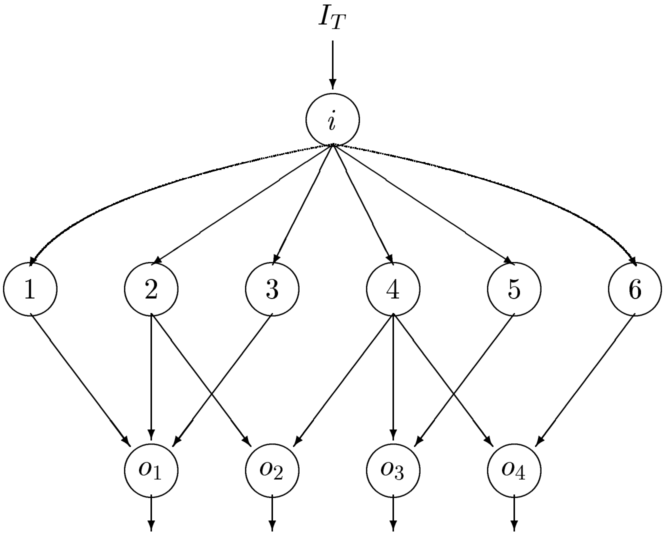


Abbildung 3.2: Darstellung eines Delay-Sterns. Die Neuronen 1–6 stellen den Stern dar. Der Stern wird einfach dadurch aktiviert, in dem man Neuron i zum Feuern bringt.

Neuron v zur Zeit $s^{(v)}$ feuert, Neuron $v+1$ zur Zeit $s^{(v+1)}$ feuert.

$$\begin{aligned}
\varepsilon_{v+1,v}(t) &:= \varepsilon(t - d_{v+1,v}) \\
\varepsilon(t') &:= \begin{cases} A \cdot \frac{t'}{\tau_p} \cdot e^{1-\frac{t'}{\tau_p}} \cdot \cos\left(\frac{\pi t'}{10\tau_p}\right) & \text{falls} \quad 0 \leq t' \leq 5\tau_p \\ 0 & \text{sonst} \end{cases} \qquad (3.1) \\
\Theta_{v+1}(t') &:= \begin{cases} \theta_{v+1} & \text{falls} \quad t' \in \{0\} \cup [\tau_{ref}, \infty) \\ \infty & \text{sonst} \end{cases}
\end{aligned}$$

$\varepsilon(t')$ wurde so gewählt, um eine Funktion zu zeigen, welche die Form eines halbwegs realistischen PSPs modelliert (vgl. Abbildung 2.4). A ist nur ein Skalierungsfaktor, der das Maximum dieser Funktion auf 1 normiert. Der cosinus Term wurde nur hinzugefügt, um zu garantieren, daß die Funktion kontinuierlich gegen 0 geht. Der Ausdruck $d_{v+1,v}$ stellt in dieser Formulierung die Verzögerungszeit zwischen Neuron v und $v+1$ dar, d.h. die Zeit, die der Impuls von Neuron v benötigt, um in Neuron $v+1$ ein PSP der Form $\varepsilon(t')$ auszulösen. Die Funktion $\Theta_{v+1}(t)$ wie sie hier dargestellt ist, modelliert eine absolute Refraktärzeit von τ_{ref} und ist in Abbildung 4.1 dargestellt.

Feuert nun ein Neuron v zur Zeit $s^{(v)}$, so nimmt $h_{v+1}\left(s^{(v+1)}\right)$ laut Gleichung 2.2 und unter der Annahme[1] $h_{v+1}\left(s^{(v)}\right) = 0$ folgenden Wert an.

$$h_{v+1}\left(s^{(v+1)}\right) = w_{v+1,v} \cdot \varepsilon\left(s^{(v+1)} - s^{(v)} - d_{v+1,v}\right) \qquad (3.2)$$

Ist nun Neuron $v+1$ nicht mehr in einer Refraktärphase, so kann man durch geeignete Wahl der Werte von Θ_{v+1}, $w_{v+1,v}$ und $d_{v+1,v}$ dafür sorgen, daß $h_{v+1}\left(s^{(v+1)}\right) = \theta_{v+1}$ ist, d.h. Neuron $v+1$ zu diesem Zeitpunkt einen Impuls abgibt. Diese drei Parameter hängen in folgender Art und Weise zusammen.

$$d_{v+1} = s^{(v+1)} - s^{(v)} - \varepsilon^{-1}\left(\frac{\theta_{v+1}}{w_{v+1,v}}\right) \qquad (3.3)$$

Dabei ist zu beachten, daß $\varepsilon^{-1}(y)$ nur auf einem bestimmten Wertebereich definiert ist (Intervall $[0,1]$) und außerdem im Falle eines realistischen PSPs nicht eindeutig ist. Mit dem Ausdruck $\varepsilon^{-1}(y)$ ist somit die kleinste der möglichen Lösungen gemeint.

Durch die Form von $\varepsilon(t)$ wird in diesem Fall auch eine Bedingung an die absolute Refraktärzeit τ_{ref} gestellt. τ_{ref} muß nämlich groß genug sein, sodaß kein zweites Aktionspotential ausgelöst wird, d.h. größer als die Zeitspanne in der $h_{v+1}(t)$ größer als θ_{v+1} ist.

---

[1] Das heißt, Neuron v hat schon länger nicht mehr gefeuert. Diese Annahme könnte bei periodischen Mustern ein Problem werden (vgl. Text weiter unten).

3.2 Verzögerungszeiten im Delay-Stern

Wählt man bei dieser Architektur dieselben Antwort- und Schwellwertfunktionen, so ergeben sich für die Verzögerungszeiten $d_{v,i}$ vom Initiatorneuron i (vgl. Abbildung 3.2) zu den Timingneuronen $v \in S^{(\nu)}$ folgende Werte:

$$d_{v+1} = s^{(v)} - s^{(i)} - \varepsilon^{-1}\left(\frac{\theta_v}{w_{v,v}}\right) \tag{3.4}$$

Der Ausdruck $s^{(i)}$ gibt die Zeit an, zur der das Neuron i initialisiert wird. Für $\varepsilon^{-1}(y)$ gilt hier dasselbe, wie bei der Delay-Kette angemerkt wurde.

3.3 Die Outputneuronen

Um das Netzwerk vollständig zu definieren, müssen auch die Verbindungen zwischen den Neuronen $v \in C^{(\nu)}$ bzw. $v \in S^{(\nu)}$ und den Outputneuronen $o \in V_{out}$ beschrieben werden. Man nimmt hier nur Synapsen $\langle v, o\rangle$ in E auf, für die gilt $s^{(v)} \in \tilde{F}_o^{(\nu)}$; das heißt nur wenn Neuron o zur Zeit $s^{(v)}$ feuern soll. Man kann für die Neuronen o die selben Funktion wie in Gleichung 3.1 verwenden, jedoch wählt man am besten alle Parameter für alle Verbindungen gleich.

$$\begin{array}{lcl} \varepsilon_{o,v}(t) & := & \varepsilon(t - d_o) \\ w_{o,v}(t) & := & w_o \\ \theta_o & := & \theta \end{array} \tag{3.5}$$

Da ein Impuls, der zur Zeit $s^{(v)}$ von Neuron v generiert wird, eine nicht verschwindende Zeit Δ benötigt, um in einem eventuellen Outputneuron einen Impuls auszulösen, wird die resultierende Impulsfolge $F_o^{(\nu)}$ gegenüber der geforderten, eben um diese Zeit Δ verschoben sein. Dies kann man leicht dadurch beheben, indem man das erste Neuron einer Kette $C^{(\nu)}$, oder das Neuron i im Delay-Stern, um diese Zeit früher zum Feuern bringt.

3.4 Periodische Muster

Bis jetzt haben wir angenommen, daß das Muster $P^{(\nu)}$ einmal durchlaufen wird, und daß dann keine weitere Aktivität mehr vorhanden ist. Man kann jedoch sehr einfach das Muster $P^{(\nu)}$ periodisch wiederholen, indem man das Ende der korrespondierenden Kette $C^{(\nu)}$ wieder an den Anfang zurückführt, dies ist in Abbildung 3.1 dargestellt.

Problematisch kann dies nur dann werden, wenn das Muster $P^{(\nu)}$ relativ kurz ist und ein Neuron $v+1$ einen Impuls von v erhält, und $h_{v+1}(t)$ aber noch nicht 0 ist. In diesem Fall wird Neuron $v+1$ schon früher als $s^{(v+1)} + k \cdot T^{(\nu)}$ $(T^{(\nu)}$

… Periode von $P^{(\nu)}$, $k \in \mathbb{N}$) feuern. Dadurch kann es zu einem Lawineneffekt kommen, nämlich, daß alle Neuronen mit einer sehr hohen Frequenz zu feuern beginnen.

Kapitel 4

Ein rekursiver CPG

Die zwei Netzwerkarchitekturen, die bis jetzt betrachtet wurden, enthalten sehr wenige Verbindungen und man kann eindeutig feststellen, welche Information in welcher Verbindung dargestellt ist. Man bezeichnet eine solche Repräsentation von Information in neuronalen Netzwerken als *lokal* (localized Repräsentation) [34]. In den folgenden Abschnitten wird hingegen ein anderer Ansatz gewählt, um die Information (in diesem Falle spatiotemporäre Muster) zu repräsentieren. In den meisten Fällen hat man es dann mit Netzwerken zu tun, in denen ein Neuron mit sehr vielen oder allen anderen verbunden ist. Der grundlegende Unterschied jedoch ist, daß man einer Verbindung nicht mehr exakt zuordnen kann, welche Information sie beinhaltet. Eine solche Repräsentation nennt man *verteilt.*

Als nächstes soll nun ein Netzwerk mit verteilter Repräsentation vorgestellt werden, das wenn es einen geeigneten Input erhält, zeitlich diskrete spatiotemporäre Muster ausgibt.

4.1 Das Netzwerk

Es sei ein Netzwerk $\mathcal{N} = \langle V, E \rangle$, $V_{out} = V = \{1, 2, \ldots, N\}$ gegeben, in welchem die Antwort-, Schwellwert- und Gewichtsfunktionen wie folgt gewählt werden,

$$\begin{aligned} \varepsilon_{i,j}(t) &:= r(t - d_0) \\ r(t') &:= \begin{cases} 1 & \text{falls} \quad 0 \leq t' \leq \tau_\varepsilon \\ 0 & \text{sonst} \end{cases} \\ w_{i,j}(t) &:= w_{i,j} \\ \Theta_i(t) &:= \begin{cases} \theta_i & \text{falls} \quad t' \in \{0\} \cup [\tau_{ref}, \infty) \\ \infty & \text{sonst} \end{cases} \end{aligned} \tag{4.1}$$

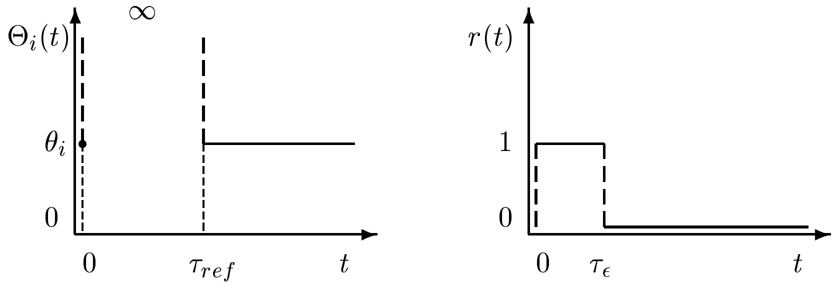


Abbildung 4.1: Links: Darstellung einer Schwellwertfunktion, welche nur eine absolute Refraktärzeit beinhaltet. Rechts: Darstellung eines sehr einfaches PSP.

wobei die Parameter d_0, τ_ε, θ_i, und τ_{ref} folgende Bedingungen erfüllen.

$$d_0 > \tau_{ref} > \tau_\varepsilon \quad \text{und} \quad \theta_i > 0 \tag{4.2}$$

Das heißt, daß ein Impuls der in Neuron j erzeugt wird, in Neuron i ein rechteckiges PSP der Dauer τ_ε auslöst. (Die Funktion $r(t)$ ist in Abbildung 4.1 dargestellt.) Vom Auftreten des Impulses in Neuron j bis zum Einsetzen des PSP in Neuron i vergehen d_0 Zeiteinheiten. Die Funktion $\Theta_i(t)$ ist dieselbe wie in Gleichung 3.1 und ist ebenfalls in Abbildung 4.1 dargestellt. Es wird lineare dendritische Integration (Gleichung 2.2) angenommen.

Wird nun ein solches Netzwerk derart initialisiert, daß eine Menge von Neuronen zur gleichen Zeit $t = 0$ feuern, so können weitere Impulse nur zu Zeiten $t = k \cdot d_0$ auftreten. Dies kann man sich folgendermaßen überlegen.

Sei $t_0 = k \cdot d_0$ ein Zeitpunkt, bis zu dem alle Neuronen nur Impulse zu Zeiten $t = k \cdot d_0$, $k \in \mathbb{N}$, $t \leq t_0$ abgegeben haben. Aufgrund dieser Annahme, und der Voraussetzung, daß $\tau_\varepsilon < d_0$ ist, ist das Potential zum Zeitpunkt t_0 nur noch ein Resultat der Impulse zur Zeit $t_0 - d_0$. Die stereotype Form der PSP und die Tatsache, daß alle Impulse im Neuron i gleichzeitig ankommen, hat zur Folge, daß $h_i(t)$ ebenfalls eine Rechteckimpulsform annimmt, eben nur mit einer anderen Amplitude.

$$h_i(t) = \sum_{j:\langle j,i\rangle \in E} w_{i,j} \cdot r(t - t_0 - d_0) \cdot \mathbf{1}_{t_0 - d_0 \in F_j} \quad t_0 \leq t \leq t_0 + \tau_\varepsilon \tag{4.3}$$

Für Neuronen, die zur Zeit t_0 feuern, ist $h_i(t) \geq \theta_i$. Da aber $\tau_{ref} > \tau_\varepsilon$ ist, können solche Neuronen nur einmal aufgrund dieses „Potentialberges“ (Gleichung 4.3) feuern, denn danach ist $h_i(t)$ wieder 0. Daß zu diesem Zeitpunkt kein weiterer Impuls mehr auftritt, ist durch die Voraussetzung, daß θ_i größer 0, ist gegeben.

Das heißt, der nächste Impuls kann erst wieder auftreten, wenn die zur Zeit t_0 generierten Impulse die nachgeschalteten Neuronen erreicht haben. Da diese

Impulse alle synchron zur Zeit t_0 ausgesendet wurden und das PSP eine Rechteckfunktion ist, werden diese Impulse alle exakt zur Zeit $t_0 + d_0$ ausgelöst werden. Dies ist deshalb möglich, weil $\tau_{ref} < d_0$ ist. Ein Neuron i feuert also zur Zeit $t_0 + d_0$ wenn gilt:

$$\sum_{j:\,\langle j,i\rangle \in E} w_{i,j} \cdot \mathbf{1}_{t_0 \in F_j} \geq \theta_i \tag{4.4}$$

Ein synchrones Feuern der Neuronen zur Zeit t_0 hat also ein synchrones Feuern zum Zeitpunkt $t_0 + d$ zur Folge.

Initialisiert man nun solch ein Netzwerk zur Zeit $t_0 = 0$ derart, daß man bestimmte Neuronen synchron zum Feuern bringt, so erhält man Impulsfolgen F_i, die nur Zeitpunkte enthalten, welche ein Vielfaches von d_0 sind.

$$F_i \cap \left\{ t \in \mathbb{R}^+ : t = k \cdot d_0, k \in \mathbb{N} \cup \{0\} \right\} = F_i \tag{4.5}$$

Man erhält also zeitlich diskrete Muster. Wie man gleich sehen wird, reduziert sich diese Art von Netzwerk auf ein Netzwerk von linearen Schwellwertgattern, und ist damit mathematisch leichter zu handhaben.

4.2 Verbindung zu Hopfield-Netzwerken

Für dieses Netzwerk genügt es also, nur die Zeitpunkte kd_0 zu betrachten. Man kann sich daher einen Vektor $\mathbf{x}(k)$ definieren, der zu den Zeitpunkten $k \cdot d_0$ angibt, welche Neuronen feuern oder nicht.

$$\mathbf{x}(k) := [x_1(k), x_2(k), \ldots, x_N(k)]^T, \quad x_i(k) := \mathbf{1}_{k \cdot d \in F_j} \tag{4.6}$$

Aufgrund der obigen Erklärungen, kann man nun schreiben:

$$x_i(k+1) = \mathrm{stp}\left(\sum_{j=1}^{N} w_{i,j} \cdot x_j(k) - \theta_i \right), \ \mathrm{stp}(h) = \begin{cases} 1 & \text{falls} \quad h \geq 0 \\ 0 & \text{sonst} \end{cases} \tag{4.7}$$

Wird dieser Berechnungsschritt für alle Neuronen gleichzeitig durchgeführt, so beschreibt die Folge von Vektoren $\mathbf{x}(0), \mathbf{x}(1), \mathbf{x}(2), \ldots$ den gesamten zeitlichen Ablauf in diesem Netzwerk.

Mit der Gewichtsmatrix $\mathbf{W} := [w_{i,j}]_{i,j \in \{1,\ldots,N\}}$ und dem Schwellwertvektor $\boldsymbol{\theta} := [\theta_1, \theta_2, \ldots, \theta_N]^T$ ergibt sich damit in Vektorschreibweise folgender Zusammenhang, wobei die Funktion $\mathrm{stp}(h)$ elementweise auf eine Matrix bzw. einen Vektor angewendet wird.

$$\mathbf{x}(k+1) = \mathrm{stp}\left(\mathbf{W}\mathbf{x}(k) - \boldsymbol{\theta} \right) \tag{4.8}$$

Durch diese Definitionen und der Anfangsbedingung, daß zur Zeit $t = 0$ alle Neuronen synchron feuern, sind wir zu einer Gleichung gekommen, in der nur noch diskrete Zeitwerte vorkommen und sich die Berechnung des Netzwerkzustandes, der ja durch $\mathbf{x}(k)$ dargestellt wird, auf Berechnungen einzelner linearer Schwellwertfunktionen reduziert hat. Die ursprüngliche Frage, welche Impulsfolgen F_i, in einem Netzwerk repräsentiert werden können, hat sich in diesem Fall somit auf die Frage reduziert, welche Folgen von Vektoren $\mathbf{x}(0), \mathbf{x}(1), \mathbf{x}(2), \ldots$ durch derartige Netzwerke darstellbar sind.

Man hat es hier also nur noch mit einem Netzwerk aus einzelnen boolschen linearen Schwellwertgattern zu tun, was prinzipiell einem Hopfield-Netzwerk entspricht. Eine Definition des Standardmodells kann man zum Beispiel [19] entnehmen. Wobei hier allerdings als Kodierung eine $\{0,1\}$-Kodierung, im Gegensatz zu den Standard-Hopfield-Netzwerken die eine $\{+1,-1\}$-Kodierung verwenden, vorkommt.

Man kann jedoch zeigen, daß die Art der binären Kodierung keinen grundlegenden Unterschied in der Berechnungsstärke eines linearen Schwellwertgatters ausmacht.

Satz 4.1. *Es sei $y_u(\tilde{\mathbf{x}})$ die boolsche Funktion, die durch ein lineares Schwellwertgatter u mit $\{+1,-1\}$-Kodierung berechnet wird.*

$$y_u(\tilde{\mathbf{x}}) = sgn\left(\textstyle\sum_{j=1}^{N} w_{u,j} \cdot \tilde{x}_j - \theta_u\right), \quad sgn(h) = \begin{cases} +1 & falls \quad h \geq 0 \\ -1 & sonst \end{cases}$$
$$w_{u,j} \in \mathbb{R}, \quad \tilde{x}_j \in \{+1,-1\} \tag{4.9}$$

Weiters sei $y_v(\mathbf{x})$ die boolsche Funktion, die durch ein lineares Schwellwertgatter v mit $\{0,+1\}$-Kodierung berechnet wird.

$$y_v(\mathbf{x}) = stp\left(\sum_{j=1}^{N} w_{v,j} \cdot x_j - \theta_v\right), \quad w_{v,j} \in \mathbb{R}, \quad x_i \in \{0,+1\} \tag{4.10}$$

Setzt man $w_{v,j} = 2w_{u,j}$ und $\theta_v = \theta_u + \sum_{j=1}^{N} w_{u,j}$ so gilt:

$$\begin{array}{lcl} y_u(\tilde{\mathbf{x}}) = -1 \;\Leftrightarrow\; y_v(\mathbf{x}) = 0 & \multirow{2}{*}{mit} & x_i = 0 \;\Leftrightarrow\; \tilde{x}_i = -1 \\ y_u(\tilde{\mathbf{x}}) = +1 \;\Leftrightarrow\; y_v(\mathbf{x}) = 1 & & x_i = 1 \;\Leftrightarrow\; \tilde{x}_i = +1 \end{array} \tag{4.11}$$

Beweis: Aufgrund der Definition von $\mathrm{sgn}(h)$ und $\mathrm{stp}(h)$ sieht man, daß wenn die Argumente h der beiden Funktionen gleich sind, obige Behauptung erfüllt ist. Man muß nur noch zeigen:

$$\sum_{j=1}^{N} w_{v,j} \cdot x_i - \theta_v = \sum_{j=1}^{N} w_{u,j} \cdot \tilde{x}_j - \theta_u \tag{4.12}$$

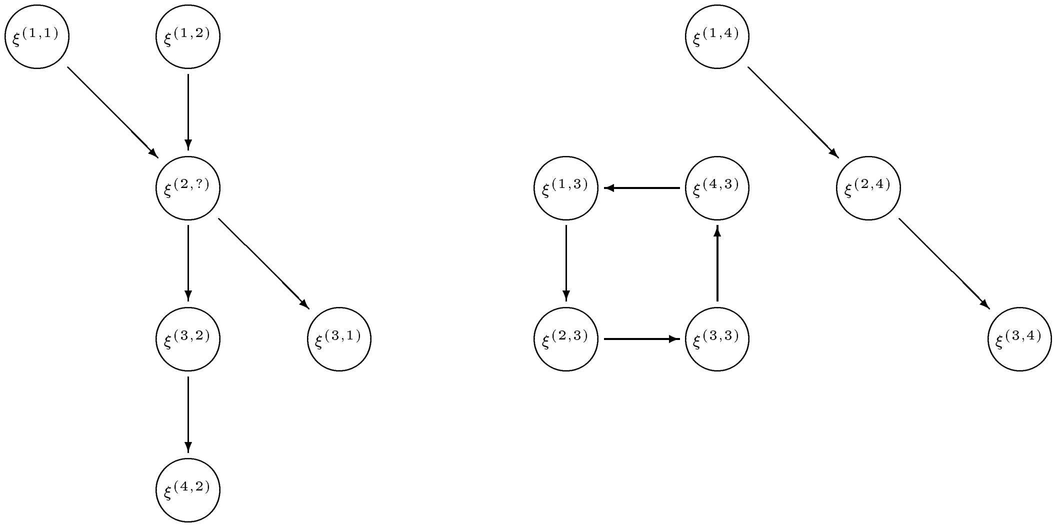


Abbildung 4.2: Links: Zwei Sequenzen $S^{(1)}$ und $S^{(2)}$ von Zuständen, die in einem Hopfield-Netzwerk nicht realisierbar sind, da sie einen gemeinsamen Zustand ($\xi^{(2,?)}$) beanspruchen. Rechts: Zwei Sequenzen $S^{(3)}$ (ein Zyklus) und $S^{(4)}$ (eine lineare Sequenz), die gemeinsam in einem Netzwerk repräsentiert werden können.

Der Zusammenhang zwischen den beiden Kodierungen kann leicht durch die Beziehung $\tilde{\mathbf{x}} = 2\mathbf{x} - 1$ beschrieben werden. Setzt man dies und die obigen Werte für $w_{v,j}$ und θ_v ein, so erhält man:

$$\sum_{j=1}^{N} 2w_{u,j}x_j - \theta_u - \sum_{j=1}^{N} w_{u,j} = \sum_{j=1}^{N} w_{u,j}(2x_j - 1) - \theta_u = \sum_{j=1}^{N} w_{u,j}\tilde{x}_j - \theta_u \qquad (4.13)$$

Womit gezeigt wäre, daß man durch geeignete Wahl der Gewichte dieselben boolschen Funktionen in $\{0, 1\}$ sowohl als auch in $\{-1, +1\}$-Kodierung berechnen kann. □

4.3 Sequenzen in Hopfield-Netzwerken

Aufgrund der Gleichung 4.8 ergeben sich prinzipielle Einschränkungen, denen die Sequenzen, die von einem solchen Netzwerk generiert werden können, unterliegen.

Es ist dies eine Funktion, die jedem Vektor $\mathbf{x}(k)$ eindeutig einen Vektor $\mathbf{x}(k+1)$ zuordnet. Das heißt aber, daß nur einfache Sequenzen erzeugt werden können, die, wenn mehrere vorhanden sind, sich nicht überschneiden dürfen ($S^{(\nu)} \cap S^{(\zeta)} = \emptyset$). Dies ist in Abbildung 4.2 schematisch dargestellt.

Der nächste Punkt, der zu beachten ist, ergibt sich aus der Beobachtung, daß in diesem Fall jedes Neuron als lineares Schwellwertgatter mit binärem Ausgang arbeitet. Vom Gesichtspunkt eines solchen Neurons besteht die Aufgabe eine bestimmte Sequenz zu generieren darin, jeden Input entsprechend zu klassifizieren.

Dies ist allerdings nur dann möglich, wenn durch die vorgegebenen Sequenzen $S^{(\nu)}$, $\nu \in \{1, 2, \dots, M\}$ kein Klassifizierungsproblem induziert wird, welches für ein Neuron i nicht linear separierbar ist (vgl. Abschnitt 4.7).

Führt man sich vor Augen, daß es nur 2^N, also endlich viele Zustände dieses Netzwerkes gibt, und durch Gleichung 4.8 ein eindeutiger gerichteter Graph auf diesen Punkten im Zustandsraum erzeugt wird, ergeben sich für $k \to \infty$ nur zwei Möglichkeiten.

1. Der Zustandsvektor kommt in einen stabilen Punkt, das heißt, $\mathbf{x}(k+1) = \mathbf{x}(k)$. Solche Punkte werden *Fixpunkte* genannt.

2. Der Zustandsvektor bewegt sich auf einer Bahn, die sich schließt, d.h. es gibt ein T so daß gilt $\mathbf{x}(k+T) = \mathbf{x}(k)$. Dies nennt man einen *Grenzzyklus* (limit cycle). Ein Fixpunkt ist ein Grenzzyklus mit der Periode $T = 1$.

Betrachtet man diesen Sachverhalt von einer anderen Seite, so kann man sagen, der Zustandsvektor wird zu einem gewissen asymptotischen Zustand hingezogen. Aus diesem Grund nennt man diese auch *Attraktoren* [13].

Aufgrund dieser Tatsache, sollten die vorgegebenen Sequenzen, wenn sie definiert enden sollen, mit einem vorgegebenen Grenzzyklus abschließen. Für die Sequenz $S^{(3)}$ in Abbildung 4.2 würde dies heißen, daß der Übergang vom Zustand $\xi^{(3,4)}$ auf sich selbst mit in die Konstruktion des Netzwerkes einbezogen werden soll. Wird dies nicht gemacht, so steht nicht fest, welchen Zustand das Netzwerk nach $\xi^{(3,4)}$ einnehmen wird.

Da es für die Konstruktion des Netzwerkes nicht von Bedeutung ist, zu welcher Sequenz ein bestimmter Zustand gehört, sondern nur in welchen Nachfolgezustand dieser übergehen soll, wird im folgenden nur die Menge der zu speichernden Zustandsübergänge („Links“) zwischen den beteiligten Zuständen betrachtet. Die Menge der beteiligten Systemzustände wird im folgenden als $\mathcal{X} = \{\xi^1, \xi^2, \dots \xi^L\}$ bezeichnet werden. Die Zustandsübergänge werden durch die Funktion $\sigma : \{1, \dots, L\} \to \{1, \dots L\}$ dargestellt. Für die Sequenzen 3 und 4 aus Abbildung 4.2 ergibt das:

$$\begin{aligned}
\mathcal{X} &= \{\xi^1, \xi^2, \xi^3, \xi^4, \xi^5, \xi^6, \xi^7\} \\
&= \{\xi^{(1,3)}, \xi^{(2,3)}, \xi^{(3,3)}, \xi^{(4,3)}, \xi^{(1,4)}, \xi^{(2,4)}, \xi^{(3,4)}\} \\
\sigma &= \{(1,2), (2,3), (3,4), (4,1), (5,6), (6,7), (7,7)\}
\end{aligned} \tag{4.14}$$

In den nächsten Abschnitten, sollen nun verschiedene Methoden vorgestellt werden, wie man die Gewichte wählen kann, um gewisse vorgegebene Zustandsübergänge in einem Netzwerk zu manifestieren.

4.4 Die Hebb'sche Lernregel

Die ursprüngliche Formulierung der Hebb'schen Regel besagt, daß das Gewicht einer Synapse dann vergrößert wird, wenn das Feuern des präsynaptischen Neurons mit dem Feuern des postsynaptischen Neurons gut korreliert. Diese Formulierung bedeutet aber, daß keine negativen Gewichte (inhibitorische Synapsen) ausgebildet werden können. Um dies zu ermöglichen, kann man die Regel etwas erweitern und sagen, daß das Gewicht verringert werden soll, wenn die Korrelation der post- und präsynaptischen Aktivität nicht gegeben ist.

Diese Art der Hebb'schen Regel läßt sich sehr schön formulieren, wenn die zu speichernden Muster bzw. Links in einer $\{+1, -1\}$-Kodierung gegeben sind. Seien $\xi^\mu, \xi^{\sigma(\mu)} \in \{+1, -1\}^N$ zwei Muster, die in einer Sequenz zeitlich aufeinander folgen sollen. Man kann dann die Gewichtsänderung $\Delta w_{i,j}$, die durch die Hebb'sche Regel Vorgeschlagen wird, einfach wie folgt formulieren.

$$\Delta w_{i,j} := \eta\, \xi_i^{\sigma(\mu)} \xi_j^\mu, \quad \eta > 0; \tag{4.15}$$

Haben $\xi_i^{\sigma(\mu)}$ und ξ_j^μ dasselbe Vorzeichen, das heißt ihre Aktivität ist korreliert, so wird $w_{i,j}$ um den Betrag η erhöht. Andernfalls (entgegengesetzte Vorzeichen entsprechen unkorrelierter Aktivität) wird $w_{i,j}$ um η verringert.

Hat man nun mehrere Links, die man in einem Netzwerk speichern möchte, so addiert man die Gewichtsänderungen all dieser Links auf.

$$w_{i,j} = \frac{1}{N} \sum_{\mu=1}^{L} \xi_i^{\sigma(\mu)} \xi_j^\mu \tag{4.16}$$

Versteht man die Menge $\mathcal{X}$ der Zustände als Matrix, deren Spalten die Vektoren ξ^μ sind

$$\boldsymbol{\mathcal{X}} := \left[\xi^1\, \xi^2\, \cdots \xi^L\right] \tag{4.17}$$

und bezeichnet man mit $\boldsymbol{\mathcal{X}}^\sigma$ jene Matrix, die entsteht, wenn man die Spalten von $\boldsymbol{\mathcal{X}}$ entsprechen der Funktion σ vertauscht

$$\boldsymbol{\mathcal{X}}^\sigma := \left[\xi^{\sigma(1)}\, \xi^{\sigma(2)}\, \cdots \xi^{\sigma(L)}\right] \tag{4.18}$$

so ergibt sich für die Gewichtsmatrix $\mathbf{W}$ folgender Zusammenhang.

$$\mathbf{W} = \frac{1}{N}\, \boldsymbol{\mathcal{X}}^\sigma \boldsymbol{\mathcal{X}}^T \tag{4.19}$$

Vergleicht man diese Lernregel mit der Lernregel für das Standard-Hopfield-Modell eines assoziativen Speichers, so sieht man, daß hier die Korrelationen von aufeinanderfolgenden Systemzuständen gespeichert werden, während beim Hopfield-Modell die Korrelationen der gewünschten Systemzustände mit sich selbst gespeichert werden. Genau wie bei diesem Modell, werden auch hier die Schwellwerte $\theta_i = 0$ gesetzt.

4.4.1 Ein periodisches Muster der Länge zwei

Da a priori nicht klar ist, daß diese Lernregel auch tatsächlich die vorgegebenen Links speichert, soll anhand eines sehr einfachen Beispiels gezeigt werden, daß diese Regel das Gewünschte leistet.

Sei $\boldsymbol{\mathcal{X}} = [\xi\ \zeta]$, $\xi \neq \zeta$. Diese beiden Muster sollen nun periodisch als Systemzustände des Netzwerkes aufscheinen, wenn das Netzwerk mit einem der beiden initialisiert wird. Aus Gleichung 4.19 ergibt sich für die Gewichtsmatrix:

$$\mathbf{W} = \frac{1}{N}\ \left(\xi\ \zeta^T + \zeta\ \xi^T\right) \tag{4.20}$$

Initialisiert man nun das Netzwerk z.B. mit ξ, d.h. $\mathbf{x}(0) = \xi$, so ergibt sich für $\mathbf{x}(1)$ folgender Wert.

$$\mathbf{x}(1) = \operatorname{sgn}\left(\mathbf{W}\mathbf{x}(0) - \boldsymbol{\theta}\right) = \operatorname{sgn}\left(\mathbf{W}\xi\right) \tag{4.21}$$

$$\mathbf{W}\xi = \frac{1}{N}\left(\xi\ \zeta^T + \zeta\ \xi^T\right)\xi = \frac{1}{N}\left(\xi\left(\zeta^T\ \xi\right) + \zeta\left(\xi^T\ \xi\right)\right) \tag{4.22}$$

Der Ausdruck $1/N(\xi^T\xi)$ stellt nichts anderes dar, als das innere Produkt des Vektors ξ mit sich selber und ist aufgrund der $\{+1, -1\}$-Kodierung gleich 1. Da $\xi \neq \zeta$ ist, ist der Betrag des Ausdrucks $1/N(\zeta^T\xi)$ kleiner 1. Deshalb kann der Vektor $1/N\xi(\zeta^T\xi)$ das Vorzeichen des Vektors ζ nicht ändern. Daraus folgt

$$\mathbf{x}(1) = \operatorname{sgn}\left(\mathbf{W}\xi\right) = \zeta \tag{4.23}$$

Die selben Überlegungen gelten auch für den nächsten Schritt und man erhält:

$$\mathbf{x}(2) = \operatorname{sgn}\left(\mathbf{W}\mathbf{x}(1)\right) = \operatorname{sgn}\left(\mathbf{W}\zeta\right) = \xi \tag{4.24}$$

Man sieht also, daß durch die Hebb'sche Regel tatsächlich gewünschte Zustandsübergänge gespeichert werden können.

In diesem einfachen Beispiel waren nur zwei Links beteiligt. Wie verhält sich ein Netzwerk, wenn mehrere solcher Links gespeichert werden?

4.4.2 Orthogonale Vektoren

Als ersten Einblick kann man sehr leicht zeigen, daß wenn die Vektoren ξ^μ orthogonal zueinander sind, jeder Link korrekt gespeichert werden kann.

Es sei $\mathbf{W} = 1/N\ \boldsymbol{\mathcal{X}}^\sigma \boldsymbol{\mathcal{X}}^T$ und es gelte für zwei Vektoren ξ^μ, ξ^ν, $\mu \neq \nu$, daß sie sind orthogonal sind ($\xi^{\mu T}\ \xi^\nu = 0$).

Betrachtet man nun das Potential $h_i(k+1)$ eines Neurons, wenn zum Zeitpunkt k ein beliebiges Muster ξ^a als Systemzustand anzutreffen war.

$$h_i(k+1) = \sum_{j=1}^{N} w_{i,j}\xi_j^a = \sum_{j=1}^{N}\frac{1}{N}\sum_{\mu=1}^{L}\xi_i^{\sigma(\mu)}\xi_j^\mu\xi_j^a = \sum_{\mu=1}^{L}\xi_i^{\sigma(\mu)}\frac{1}{N}\sum_{j=1}^{N}\xi_j^\mu\xi_j^a \tag{4.25}$$

Für $\mu = a$ wird $1/N \sum_{j=1}^{N} \xi_j^\mu \xi_j^a$ gleich 1. Spaltet man nun dies von der restlichen Summe ab, so erhält man folgenden Ausdruck.

$$h_i(k+1) = \xi_i^{\sigma(a)} + \sum_{\mu=1,\mu\neq a}^{L} \xi_i^{\sigma(\mu)} \frac{1}{N} \left(\xi^{\mu T} \cdot \xi^a\right) \tag{4.26}$$

Da laut Voraussetzung $\xi^{\mu T} \cdot \xi^a$ gleich 0 ist, nimmt das Potential den Wert $\xi_i^{\sigma(a)}$ an. Daraus folgt, daß $\mathbf{x}(k+1)$ wie gewünscht $\xi^{\sigma(a)}$ sein wird.

4.4.3 Unkorrelierte Muster

Die Orthogonalität der Vektoren ist natürlich eine ziemliche Einschränkung. Man sieht in Gleichung 4.26, daß die Forderung nach Orthogonalität ein wenig aufgeweicht werden kann. Um einen korrekten Zustandsübergang zu erreichen, muß gelten:

$$\begin{array}{lll} h_i^N < +1 & \text{falls} & \xi_i^{\sigma(a)} = -1 \\ h_i^N > -1 & \text{falls} & \xi_i^{\sigma(a)} = +1 \end{array} \quad \text{mit} \quad h_i^N = \frac{1}{N} \sum_{\mu=1,\mu\neq a}^{L} \xi_i^{\sigma(\mu)} \left(\xi^{\mu T} \cdot \xi^a\right) \tag{4.27}$$

Man sieht, daß mit steigendem L diese Größe ansteigen wird und dadurch die gespeicherten Links unzuverlässig werden. Für den Fall, daß man sich die einzelnen Elemente der Muster ξ^μ durch unabhängige Zufallsereignisse entstanden vorstellt, kann man eine Abschätzung vornehmen, wie groß die Wahrscheinlichkeit ist, daß ein gegebener Link korrekt gespeichert wird.

Die Wahrscheinlichkeit $p := \Pr\{\xi_i^\mu = +1\}$ $(\Pr\{\xi_i^\mu = -1\} = 1 - p)$, daß ein Element gleich +1 (−1) ist soll in diesem betrachteten Fall gleich 1/2 (1/2) sein. Dies entspricht Mustern, die voneinander unabhängig (unkorreliert) sind.

Betrachtet man nun die einzelnen Variablen in diesem Term als Zufallsvariablen mit den oben beschriebenen Eigenschaften, kann man die Wahrscheinlichkeit abschätzen, mit der die Bedingung in Gleichung 4.27 erfüllt ist.

Schreibt man h_i^N folgendermaßen an,

$$h_i^N = \frac{1}{N} \sum_{\mu=1,\mu\neq a}^{L} \sum_{j=1}^{N} \xi_i^{\sigma(\mu)} \xi_j^\mu \xi_j^a \tag{4.28}$$

so sieht man, daß er eine Summe von Zufallsvariablen darstellt. Nimmt man an, daß die einzelnen Faktoren voneinander unabhängig sind, so kann man den Summanden $\xi_i^{\sigma(\mu)} \xi_j^\mu \xi_j^a$ als eine Zufallsvariable ζ auffassen. Da die Wahrscheinlichkeit $\Pr\{\zeta = +1\}$ $(\Pr\{\zeta = -1\})$ das ζ den Wert +1 (−1) annimmt gleich 1/2 ist, ergibt sich, daß diese Zufallsvariable einen Erwartungswert $\mathrm{E}[\zeta] = 0$ und eine Varianz $\mathrm{Var}(\zeta) = 1$ hat. Der zentrale Grenzwertsatz der Wahrscheinlichkeitstheorie

([28]) besagt nun, daß eine Summe von n Zufallsvariablen ζ mit einem Mittelwert $\overline{\zeta}$ und einer Varianz $\mathrm{Var}(\zeta)$ gegen eine Standardnormalverteilung mit einem Mittelwert von $n\overline{\zeta}$ und einer Varianz von $n\mathrm{Var}(\zeta)$ konvergiert, wenn n gegen unendlich geht. In diesem Fall heißt dies, daß Gleichung 4.28 eine normalverteilte Zufallsvariable h_i^N mit einem Mittelwert von $\mathrm{E}[h_i^N] = \overline{h_i^N} = 0$ und einer Varianz von $\mathrm{Var}(h_i^N) = \rho^2 = (L-1)N/N^2$ darstellt, wenn nur N groß genug wird. Daß diese Annahme schon für $N = 100$ relativ gut erfüllt ist, ist in Abbildung 4.3 A dargestellt.

Sein nun $\varphi(z, \overline{z}, \rho)$ die Dichtefunktion der Normalverteilung mit Mittelwert $\overline{z}$ und Varianz ρ^2 und $\Phi(z, \overline{z}, \rho)$ die dazugehörige Verteilungsfunktion.

$$\Phi(z,\overline{z},\rho) \;:=\; \int_{-\infty}^{z} \varphi(y,\overline{z},\rho)\, dy \tag{4.29}$$

$$\varphi(y,\overline{z},\rho) \;:=\; \frac{1}{\sqrt{2\pi}\rho} e^{-(y-\overline{z})^2/2\rho^2} \tag{4.30}$$

Damit ist die Wahrscheinlichkeit, daß h_i^N die Bedingung in Gleichung 4.27 erfüllt, gegeben durch:

$$\begin{aligned}\Pr\{x_i(k+1) = \xi_i^{\sigma(a)}\} = \Pr\{\xi_i^{\sigma(a)} = +1\}\Pr\{h_i^N > -1\} + \\ + \Pr\{\xi_i^{\sigma(a)} = -1\}\Pr\{h_i^N < +1\}\end{aligned} ; \tag{4.31}$$

Mit $\Pr\{\xi_i^{\sigma(a)} = +1\} = \Pr\{\xi_i^{\sigma(a)} = -1\} = 1/2$ und der symmetrischen Verteilung von h_i^N ($\Pr\{h_i^N > -1\} = \Pr\{h_i^N < 1\}$) ergibt sich daraus:

$$\Pr\{x_i(k+1) = \xi_i^{\sigma(a)}\} = \Phi\left(1, 0, \sqrt{\frac{L-1}{N}}\right) \tag{4.32}$$

Damit nun ein Link korrekt gespeichert ist, muß Gleichung 4.27 für alle $i \in \{1, \ldots N\}$ gelten. Aufgrund der Unabhängigkeit der Zufallsvariablen ist die Wahrscheinlichkeit, daß ein Link korrekt ist, das Produkt der Wahrscheinlichkeiten, daß ein Neuron seinen Zustand korrekt wechselt.

$$\Pr\{\mathbf{x}(k+1) = \xi^{\sigma(a)}\} = \Pr\{x_i(k+1) = \xi_i^{\sigma(a)}\}^N \tag{4.33}$$

Diese Wahrscheinlichkeit ist in Abbildung 4.3 B in Abhängigkeit von $\alpha = L/N$ für verschiedene N dargestellt. Ebenfalls sind die Kurven gezeigt, die sich aus Simulationen der entsprechenden Netzwerke ergeben haben (vgl. Anhang A.3). Man sieht in diesem Diagramm sehr schön, daß, wenn die Anzahl der gespeicherten Links zu groß wird, die Wahrscheinlichkeit, daß ein Link korrekt ist, sehr schnell gegen 0 geht. Dieser Abfall wird umso steiler, je größer N wird. Dieser Verlauf erinnert an die Kurven, die angegeben werden, wenn es sich um die Speicherkapazität von Hopfield-Netzwerken handelt. (vgl. [19]).

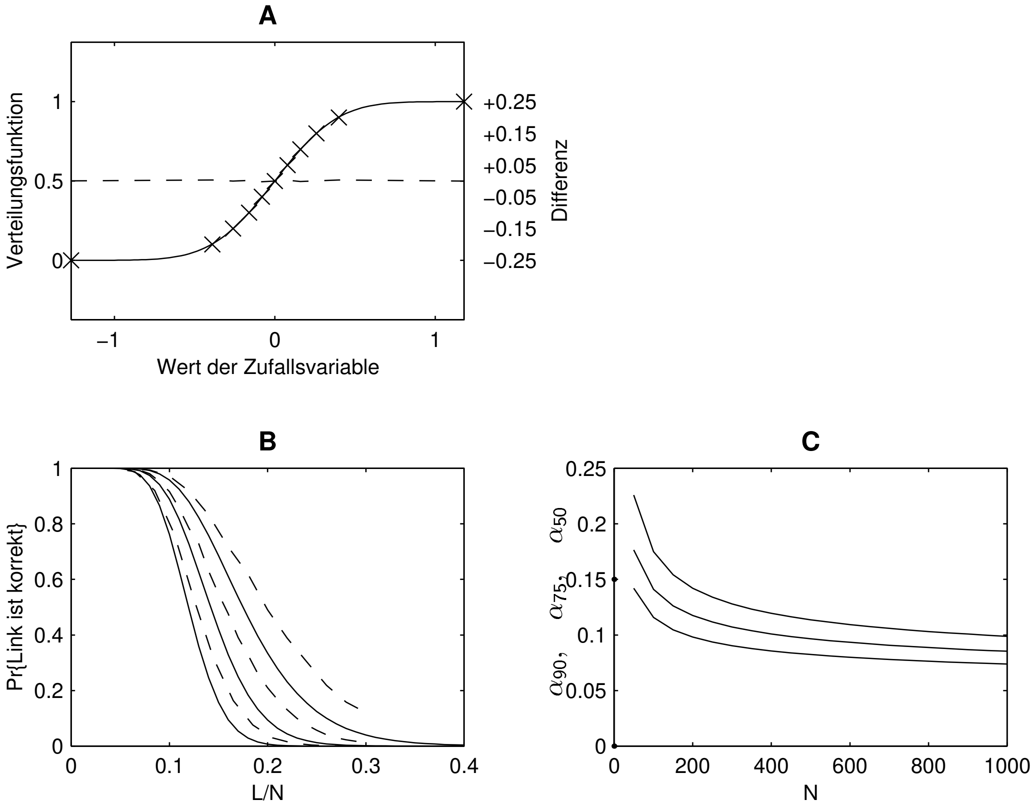


Abbildung 4.3: Zur Speicherkapazität eines rekursiven CPG, der mit der Hebb'schen Regel für {+1,-1}-Kodierung konstruiert wurde. **A**: Die durchgezogene Linie stellt die Verteilungsfunktion einer Normalverteilung mit einem Mittelwert von 0 und einer Varianz von $(L-1)/N = 9/100$ dar. Die Kreuze deuten die ermittelten Werte aus einer Simulation an. Die strichlierte Linie stellt die Differenz zwischen der Abschätzung nach Gleichung 4.29 mit entsprechenden Parametern und der Simulation (vgl. Anhang A.3) dar. **B**: In diesem Diagramm ist die Wahrscheinlichkeit, daß ein Link korrekt gespeichert wird, in Abhängigkeit von N dargestellt. Zu sehen sind Kurven für die Werte $N = 100$, $N = 200$ und $N = 400$ (von rechts nach links). Die durchgezogenen Linien stellen den geschätzten Verlauf dar (Gleichung 4.33). Die strichlierten Linien sind Simulationsergebnisse. **C**: Darstellung der Größen α_{90}, α_{75} und α_{50} (vgl. Text Seite 42) (von unten nach oben) in Abhängigkeit von N.

Im Schnitt werden also $L\,\mathrm{Pr}\{\mathbf{x}(k+1) = \xi^{\sigma(a)}\}$ Links korrekt gespeichert. Möchte man haben, daß z.B. 90% der Links korrekt sind, so ergibt sich eine obere Grenze von L mit $L_{max}^{90} = \alpha_{90}N$. Dabei ist α_{90} (α_x) jener Wert, bei dem die Kurven aus Abbildung 4.3 B die 90% (x%) Marke schneiden. Das heißt, α_x ist die Lösung der Gleichung

$$\Phi\left(1, 0, \sqrt{\alpha_x - \frac{1}{N}}\right)^N = \frac{x}{100} \tag{4.34}$$

α_{90} ist neben α_{75} und α_{50} in Abbildung 4.3 C dargestellt.

4.4.4 Korrelierte Muster

Die Situation verschlechtert sich noch mehr, wenn die Muster eine Korrelation mit einem Hintergrundmuster aufweisen. Setzt man die Wahrscheinlichkeit p nun nicht mehr 1/2, so wird die Korrelation $m = 1/N \, \sum_{j=1}^{N} \xi_j^\mu$ mit dem Muster $[1, \ldots, 1]^T$ (oder die durchschnittliche Aktivität in einem gespeicherten Muster) nicht mehr 0 sein, sondern nimmt den Wert $m = (2p-1)$ an. Dadurch nimmt das innere Produkt zweier Muster $\xi^{\mu T}\, \xi^a$ im Durchschnitt den Wert Nm^2 an. Setzt man dies in Gleichung 4.26 ein, so erhält man eine qualitative Aussage darüber, was passiert, wenn $p \neq 1/2$ ist.

$$h_i(k+1) \approx \xi_i^{\sigma(a)} + \sum_{\mu=1,\mu\neq a}^{L} \xi_i^{\sigma(\mu)} \cdot m^2 \approx \xi_i^{\sigma(a)} + Lm^3 \tag{4.35}$$

Ist $m > 0$ bzw. $p > 1/2$ ($m < 0$ bzw. $p < 1/2$), so ist, wenn L groß genug wird, $h_i(k+1)$ immer 1 (-1), unabhängig von $\xi_i^{\sigma(a)}$. Als grobe Abschätzung kann man sagen, daß dies Eintritt wenn $L > L_{max} = 1/|m^3|$. Man sieht also, daß wenn m von 0 abweicht man kaum noch einen Link korrekt speichern kann.

Um eine etwas quantitativere Aussage über das Verhalten bei korrelierten Mustern zu erhalten, kann man wie im vorigen Abschnitt über unkorrelierte Muster die Verteilungsfunktion von h_i^N (Gleichung 4.28) abschätzen und daraus nach Gleichung 4.31 die Wahrscheinlichkeit, daß ein Link korrekt ist, berechnen.

Diese Ergebnisse sind in Abbildung 4.4 dargestellt. Am deutlichsten bringt das Diagramm B zum Ausdruck wie empfindlich diese Art der Hebb'schen Regel gegen Korrelationen der Muster untereinander ist. Die Ableitung der Verteilungsfunktion von h_i^N für $m \neq 0$ ist in Anhang A.1 ausgeführt, ebenso wie die Abschätzung der Wahrscheinlichkeit $\mathrm{Pr}\{\mathbf{x}(k+1) = \xi^{\sigma(a)}\}$.

4.5 Eine verbesserte Hebb'sche Lernregel

Im letzten Abschnitt haben wir gesehen, daß die Form der Hebb'schen Regel laut Gleichung 4.15, sehr empfindlich gegenüber der Orthogonalität der zugrundelie-

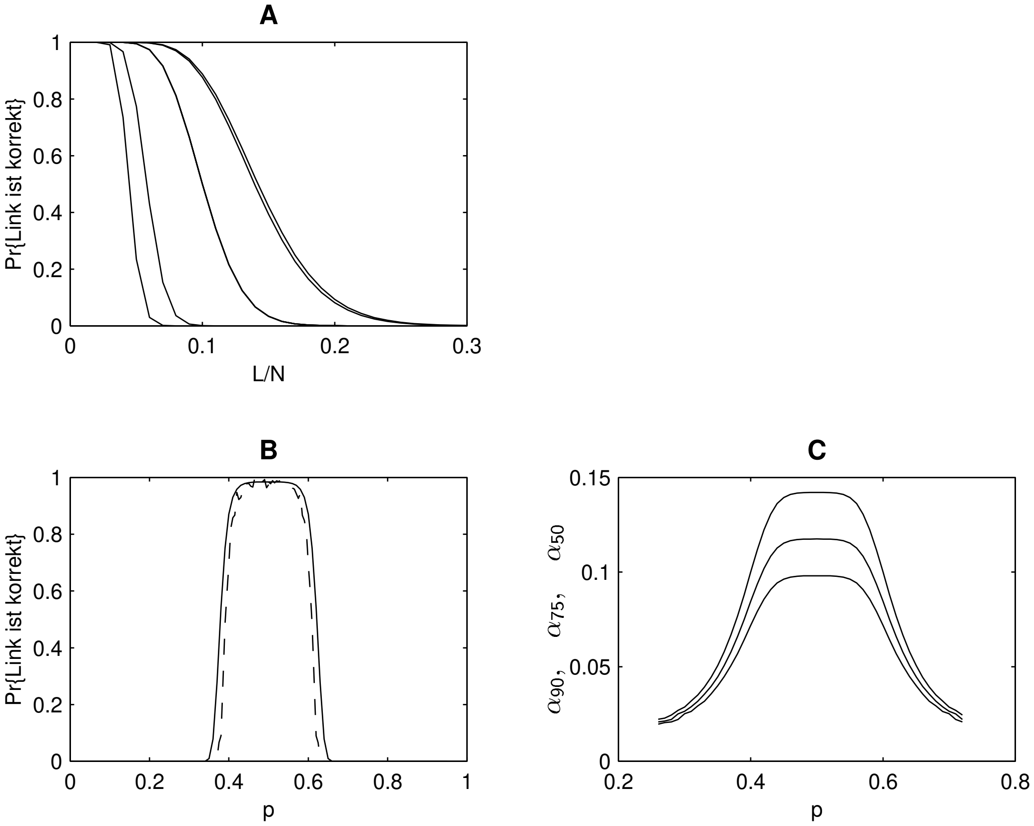


Abbildung 4.4: Zur Speicherkapazität eines rekursiven CPG, der mit der Hebb'schen Regel für {+1,-1}-Kodierung konstruiert wurde, wenn $p \neq 1/2$ ist. **A**: Darstellung der Wahrscheinlichkeit, daß ein Link korrekt ist, in Abhängigkeit von L. Die verschiedenen Kurven ergeben sich durch die verschiedenen Wahrscheinlichkeiten $p = \Pr\{\xi_i^\mu = +1\}$. Zu sehen sind die Kurven für die Werte $p = 0.5, 0.55, 0.6, 0.62, 0.66$ (von rechts nach links). N ist in diesem Fall mit 200 gewählt worden. Die Kurven stellen den geschätzten Verlauf nach Gleichung A.10 dar. **B**: Dieses Diagramm zeigt, wie schnell die Wahrscheinlichkeit, daß ein Link korrekt ist, gegen 0 geht, wenn die Wahrscheinlichkeit p von 1/2 abweicht. Die strichlierte Kurve ist ein Simulationsergebnis, während die durchgezogene Linie eine Abschätzung (Gleichung A.10) darstellt ($N = 200$, $L = 20$). **C**: Dieses Diagramm zeigt die Abhängigkeit der Größen α_{90}, α_{75} und α_{50} (vgl. Text Seite 42) von p. Es sind dies numerische Lösungen der Gleichung $\Pr\{\mathbf{x}(k+1) = \xi^{\sigma(a)}\} = x\%$, wobei x eben entweder 90, 75 oder 50 ist und $\Pr\{\mathbf{x}(k+1) = \xi^{\sigma(a)}\}$ durch Gleichung A.10 gegeben ist.

genden Vektoren ist. Man kann diesen Nachteil jedoch bis zu einem gewissen Grad lindern, indem man die Hebb'sche Regel etwas anders formuliert.

Das Gewicht einer Synapse wird dann erhöht (erniedrigt), wenn die Abweichung von der durchschnittlichen Aktivität des prä- und postsynaptischen Neurons korreliert (unkorreliert) ist ([8]). Sind nun die zu speichernden Links in einer {0,1}-Kodierung gegeben, und ist m die durchschnittliche Aktivität, wenn ein Muster ξ^μ als Netzwerkzustand aufscheint, so kann man diese Art der Hebb'schen Regel folgendermaßen schreiben: (vgl. Abschnitt 7.5.4 in [25])

$$w_{i,j} = \frac{1}{N}\sum_{\mu=1}^{L}(\xi_i^{\sigma(\mu)} - m)(\xi_j^\mu - m); \tag{4.36}$$

Wird nun m kleiner als 1/2, so werden die zu speichernden Muster immer mehr dem Vektor $[0,\ldots,0]^T$ ähnlich, sind also miteinander korreliert. Da aber dann die meisten Elemente 0 sind, und nicht wie im letzten Abschnitt -1, wird das innere Produkt zweier Vektoren auch immer kleiner und dies führt dazu, daß mit sinkendem m die Wahrscheinlichkeit, daß ein Link korrekt ist, steigt.

Man kann nun ähnlich wie vorher, diese Wahrscheinlichkeit abschätzen. Stellt man sich die Vektoren ξ^μ wieder als Zufallsvariablen vor, so kann man m als die Wahrscheinlichkeit $\Pr\{\xi_i^\mu = 1\}$ auffassen. Man teilt nun den Term $h_i(k+1)$ in den durch ξ^a entstandenen „Signal"-Term $h_i^S(k+1)$ und den durch die restlichen Vektoren entstandenen „Noise"-Term $h_i^N(k+1)$ auf.

$$\begin{aligned}
h_i(k+1) &= \sum_{j=1}^{N} w_{i,j}\xi_j^a - \theta_i \\
&= \sum_{j=1}^{N}\frac{1}{N}\sum_{\mu=1}^{L}(\xi_i^{\sigma(\mu)} - m)(\xi_j^\mu - m)\xi_j^a - \theta_i \\
&= \frac{1}{N}(\xi_i^{\sigma(a)} - m)(1-m)\sum_{j=1}^{N}\xi_j^a - \theta_i + \\
&\quad + \frac{1}{N}\sum_{\mu=1,\mu\neq a}^{L}\sum_{j=1}^{N}(\xi_i^{\sigma(\mu)} - m)(\xi_j^\mu - m)\xi_j^a \\
h_i(k+1) &= h_i^S + h_i^N
\end{aligned} \tag{4.37}$$

Im Übergang von der zweiten auf die dritte Zeile, wurde von der Beziehung ${\xi_i^a}^2 = \xi_i^a$ Gebrauch gemacht. Damit $x_i(k+1)$ auch tatsächlich den Wert $\xi_i^{\sigma(a)}$ annimmt, muß gelten:

$$\begin{aligned}
h_i^S\big|_{\xi_i^{\sigma(a)}=1} + h_i^N &> 0 \quad \text{falls} \quad \xi_i^{\sigma(a)} = 1 \\
h_i^S\big|_{\xi_i^{\sigma(a)}=0} + h_i^N &< 0 \quad \text{falls} \quad \xi_i^{\sigma(a)} = 0
\end{aligned} \tag{4.38}$$

Die Wahrscheinlichkeit, daß $h_i(k+1)$ den korrekten Wert in Abhängigkeit von $\xi_i^{\sigma(a)}$ annimmt, ist damit gegeben durch:

$$\begin{aligned}\Pr\{x_i(k+1) = \xi_i^{\sigma(a)}\} &= \Pr\{\xi_i^{\sigma(a)} = 1\} \Pr\{h_i(k+1) > 0\} + \\ &\quad + \Pr\{\xi_i^{\sigma(a)} = 0\} \Pr\{h_i(k+1) < 0\} \\ &= m \left(1 - \Pr\{h_i^S\big|_{\xi_i^{\sigma(a)}=1} + h_i^N > 0\}\right) + \\ &\quad + (1-m) \Pr\{h_i^S\big|_{\xi_i^{\sigma(a)}=0} + h_i^N < 0\}\end{aligned} \tag{4.39}$$

Wird nun N sehr groß, so kann man wie vorher die Größen h_i^S, h_i^N als normalverteilte Zufallsvariablen betrachten und deren Mittelwert und Varianz berechnen. Dies ist in Anhang A.2 ausgeführt.

Um eine möglichst gute Trennung zwischen $h_i^S\big|_{\xi_i^{\sigma(a)}=1}$ und $h_i^S\big|_{\xi_i^{\sigma(a)}=0}$ zu erhalten, wird θ_i genau zwischen die Mittelwerte dieser beiden Größen gelegt. Daraus ergibt sich ein Schwellwert von $\theta_i = \frac{1}{2}m(1-m)(1-2m)$.

Ebenfalls wie im obigen Abschnitt ergibt sich die Wahrscheinlichkeit, daß ein Link korrekt ist, aus dem Produkt der Wahrscheinlichkeiten für alle $i = 1, \ldots, N$.

Die Ergebnisse sind in Abbildung 4.5 dargestellt. Im Gegensatz zur vorigen Regel ergibt sich hier nun, daß mit sinkendem m mehr Links korrekt gespeichert werden können. Diese Tatsache ist sehr schön in den Diagrammen A und B zu sehen.

Auf den ersten Blick ist es vielleicht etwas verwunderlich, daß die Kurve in Diagramm B bei sehr kleinem $p = m$ wieder abfällt. Überlegt man sich jedoch, daß ein Muster, daß nur aus 0 besteht, zu keinem anderen Muster mehr führen kann, da für $m < 1/2$ $\theta_i > 0$ ist. Die Wahrscheinlichkeit, daß dieses Muster auftritt, steigt jedoch an, je kleiner m wird.

In Diagramm C ist der geschätzte Verlauf von α_{90} dargestellt. Dieser fällt hier etwas spitzer aus, als die entsprechende Kurve aus Abbildung 4.4, erreicht jedoch einen höheren Wert.

Mit dieser Art der Hebb'schen Regel lassen sich also Muster mit wenig durchschnittlicher Aktivität relativ gut speichern.

4.6 Die Projektions-Regel

Hier soll eine Methode, die Gewichte festzulegen, vorgestellt werden, die biologisch nicht so sehr realistisch erscheint, jedoch für das Problem der Speicherung von definierten Zustandsänderungen in den hier betrachteten Netzwerken sehr gut geeignet ist.

Diese Methode wird durch folgende Beobachtung motiviert. Die Bedingung, daß ein Link korrekt gespeichert ist, kann für Muster in $\{-1, +1\}$-Kodierung

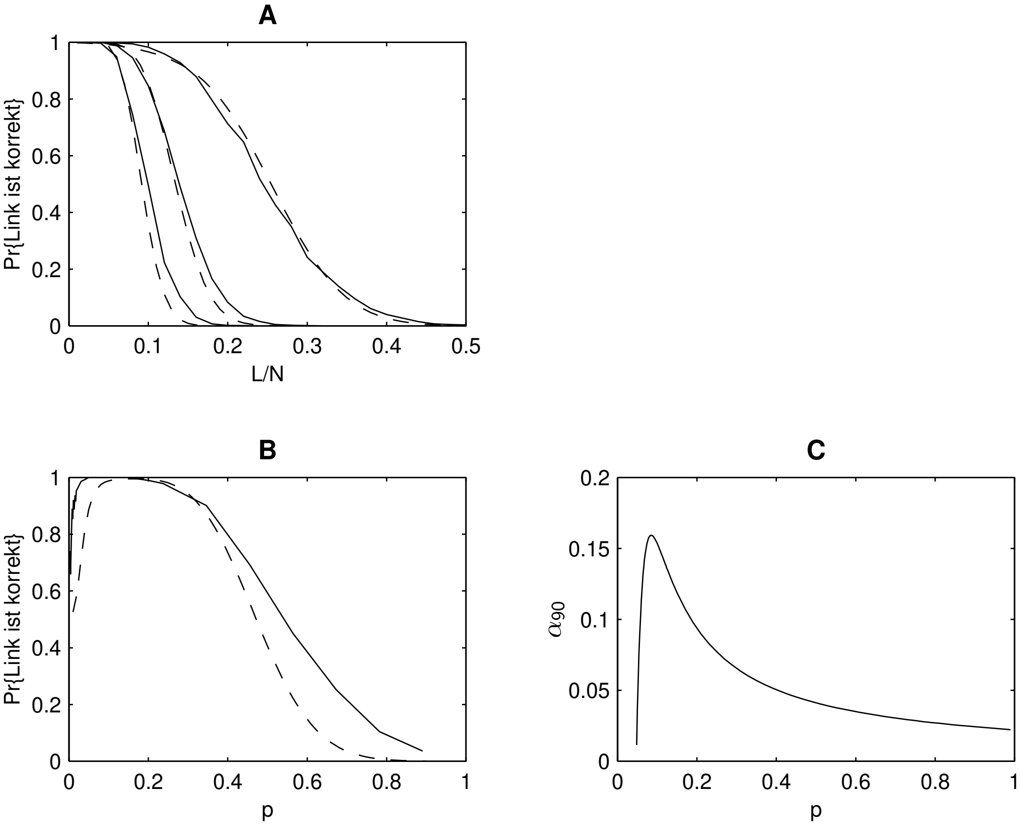


Abbildung 4.5: Zur Speicherkapazität eines rekursiven CPG, der mit der Hebb'schen Regel für {0,1}-Kodierung konstruiert wurde. Die Diagramme zeigen die selben Abhängigkeiten wie in Abbildung 4.4. **A**: Die strichlierten Linien sind die geschätzten Verläufe (Gleichung A.23), die durchgezogenen die Ergebnisse einer Simulation ($N = 500$). Die verschiedenen Kurven ergeben sich wieder für verschiedene Werte von $\Pr\{\xi_i^\mu = 1\} = p$; (von rechts nach links: $p = 0.1$, $p = 0.2$ und $p = 0.3$). **B**: Pr{Link ist korrekt} in Abhängigkeit von p ($N = 500$, $L = 30$). **C**: α_{90} in Abhängigkeit von p.

folgendermaßen formuliert werden.

$$\sum_{j=1}^{N} w_{i,j}\xi_j^\mu - \theta_i = \lambda_i^\mu \xi_i^{\sigma(\mu)}, \quad \lambda_i^\mu > 0 \tag{4.40}$$

Eine sehr einfache Darstellung des Problems ergibt sich, wenn man $\lambda_i^\mu \equiv 1$ wählt und man die Bedingung in Matrixschreibweise formuliert.

$$\mathbf{W}\xi^\mu - \boldsymbol{\theta} = \xi^{\sigma(\mu)} \tag{4.41}$$

Hat man nun wie in den vorhergehenden Abschnitten, nicht nur ein Muster ,sondern L, gegeben, so kann man dies mit den in den Gleichungen 4.17 und 4.18 definierten Matrizen folgendermaßen anschreiben.

$$\mathbf{W}\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}^\sigma + \overline{\boldsymbol{\theta}}, \quad \overline{\boldsymbol{\theta}} = \boldsymbol{\theta} \cdot [1, \dots, 1] \tag{4.42}$$

Für diese Gleichung kann man $\mathbf{W}$ leicht berechnen, wenn die Spalten von $\boldsymbol{\mathcal{X}}$ (also die Muster ξ^μ) linear unabhängig sind.

$$\mathbf{W} = (\boldsymbol{\mathcal{X}}^\sigma + \overline{\boldsymbol{\theta}})(\boldsymbol{\mathcal{X}}^T\boldsymbol{\mathcal{X}})^{-1}\boldsymbol{\mathcal{X}}^T \tag{4.43}$$

Daß dies tatsächlich eine Lösung ist, ist sehr leicht zu überprüfen, indem man einfach $\mathbf{W}\boldsymbol{\mathcal{X}}$ betrachtet und sofort feststellt, daß dies, wie gewünscht, den Wert $\boldsymbol{\mathcal{X}}^\sigma + \overline{\boldsymbol{\theta}}$ annimmt.

Die lineare Unabhängigkeit der Spalten von $\boldsymbol{\mathcal{X}}$ ist deshalb notwendig, da ansonsten die Matrix

$$\boldsymbol{\mathcal{X}}^+ = (\boldsymbol{\mathcal{X}}^T\boldsymbol{\mathcal{X}})^{-1}\boldsymbol{\mathcal{X}}^T \tag{4.44}$$

nicht existieren würde. $\boldsymbol{\mathcal{X}}^+$ wird als *Pseudoinverse* von $\boldsymbol{\mathcal{X}}$ bezeichnet. Aus der Bedingung, daß die Muster linear unabhängig sein müssen, folgt natürlich sofort eine obere Schranke von $L \leq N$. Die lineare Unabhängigkeit ist eine viel schwächere Bedingung, als zum Beispiel die Orthogonalität der Vektoren, die bei der Hebb'schen Regel Voraussetzung dafür ist, um alle Links korrekt speichern zu können. Dies kommt sehr gut in Abbildung 4.6 zum Ausdruck. Es ist dort die durchschnittliche Anzahl der linear unabhängigen Vektoren einer $N \times N$ Matrix mit Elementen aus $\{+1, -1\}$ ($\{0, 1\}$) dargestellt. Man sieht, daß für $N \geq 50$ beinahe über den gesamten Bereich alle N Spalten linear unabhängig sind. Lediglich wenn die Wahrscheinlichkeit $\Pr\{\xi_i^\mu = +1\}$ gegen 0 beziehungsweise gegen 1 geht, sinkt die Anzahl gegen 1 ab.

Es ist interessant festzuhalten, daß, wenn die Vektoren ξ^μ orthogonal zueinander sind, und die Schwellwerte $\theta_i = 0$ sind, Gleichung 4.43 in Gleichung 4.19 übergeht, also dieselbe Gewichtsmatrix erzeugt wie die hebb'sche Regel. Man

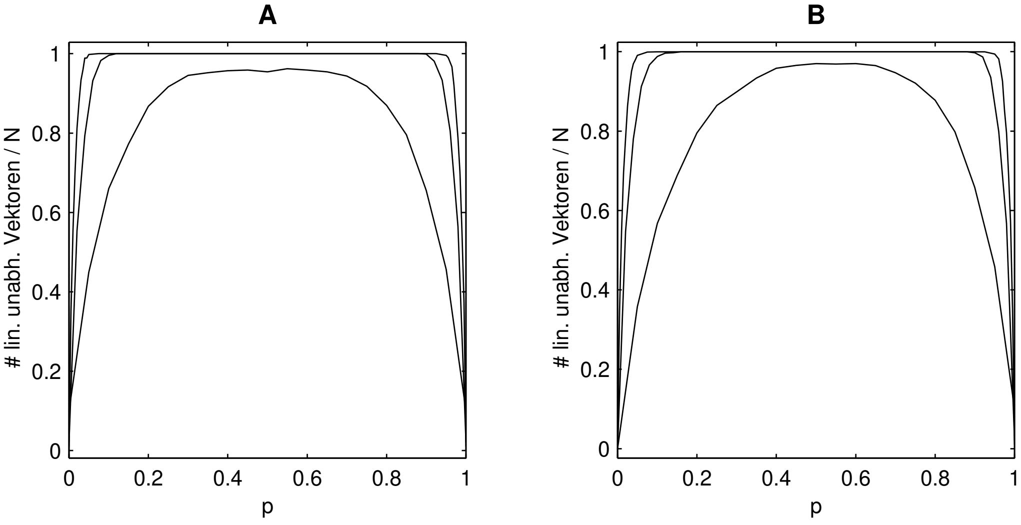


Abbildung 4.6: Darstellung der Anzahl der linear unabhängigen Vektoren in Abhängigkeit der Wahrscheinlichkeit $\Pr\{\xi_i^\mu = +1\} = p$ für verschiedene Werte von N (normiert auf N). Von unten nach oben sind dies $N = 10$, $N = 50$ und $N = 100$. **A**: Elemente der Matrix aus $\{+1, -1\}$. **B**: Elemente der Matrix aus $\{0, 1\}$. Beide Diagramme sind Ergebnisse einer Simulation. Für jeden Wert von N und p wurden 1000 zufällige Matrizen erzeugt und daraus die durchschnittliche Anzahl der linear unabhängigen Vektoren berechnet. Durchgeführt wurde dies mit dem Mathematikprogramm MATLAB®.

kann sich das sehr leicht überlegen, wenn man beachtet, daß aus der Orthogonalität der Vektoren ξ^μ sofort $\boldsymbol{\mathcal{X}}^T\boldsymbol{\mathcal{X}} = N\mathbf{I}_N$ ($\mathbf{I}_N$ … $N \times N$ Einheitsmatrix) folgt und daraus $(\boldsymbol{\mathcal{X}}^T\boldsymbol{\mathcal{X}})^{-1} = \frac{1}{N}\mathbf{I}_N$.

Ein Vorteil dieser Regel ist es, daß man sie auch sehr gut für $\{0, 1\}$-kodierte Muster verwenden kann. In diesem Fall kann man die Bedingung, daß ein Link korrekt ist folgendermaßen schreiben.

$$\mathbf{W}\xi^\mu = \zeta^{\sigma(\mu)}, \quad \text{mit} \quad \begin{array}{ll} \zeta_i^{\sigma(\mu)} > \theta_i & \text{falls} \;\; \xi_i^{\sigma(\mu)} = 1 \\ \zeta_i^{\sigma(\mu)} < \theta_i & \text{falls} \;\; \xi_i^{\sigma(\mu)} = 0 \end{array} \tag{4.45}$$

Daraus ergibt sich dann wie oben für die Gewichtsmatrix folgender Wert.

$$\mathbf{W} = \boldsymbol{\mathcal{Z}}^\sigma(\boldsymbol{\mathcal{X}}^T\boldsymbol{\mathcal{X}})^{-1}\boldsymbol{\mathcal{X}}^T, \quad \text{mit} \quad \boldsymbol{\mathcal{Z}}^\sigma := [\zeta^{\sigma(1)} \cdots \zeta^{\sigma(L)}] \tag{4.46}$$

Die Eingangs schon erwähnte biologische Unplausibilität ist, daß sich die aus dieser Regel ergebenden Gewichte nicht aus lokalen Größen zusammensetzen, sondern alle Elemente von $\boldsymbol{\mathcal{X}}$ in ein Gewicht $w_{i,j}$ eingehen.

4.7 Die Perzeptron–Lernregel

Betrachtet man das Problem der Speicherung von mehreren Links aus der Sicht eines einzelnen Neurons, so ergibt sich für dieses, wie bereits in Abschnitt 4.3 erwähnt, folgendes Klassifizierungsproblem. Die Vektoren ξ^μ sind, entsprechend der Funktion σ und der daraus resultierenden Einteilung dieser Vektoren, einer von zwei möglichen Klassen zuzuordnen. Der Label der Klasse entspricht dem Bit $\xi_i^{\sigma(\mu)}$. Es wird also jedem Neuron i eine Menge $\mathcal{T}_i$ von Beispielen, die es zu erlernen hat, vorgegeben.

$$\mathcal{T}_i := \left\{ \left\langle \xi^1, \xi_i^{\sigma(1)} \right\rangle, \left\langle \xi^2, \xi_i^{\sigma(2)} \right\rangle, \ldots, \left\langle \xi^L, \xi_i^{\sigma(L)} \right\rangle \right\} \tag{4.47}$$

Es ist nun für jedes i unabhängig ein Gewichtsvektor $\mathbf{w}_i = [w_{i,1}, w_{i,2}, \ldots w_{i,N}]$ und ein Schwellwert θ_i zu finden, so daß für alle $\mu \in \{1, \ldots, L\}$ gilt:

$$\xi_i^{\sigma(\mu)} = \operatorname{sgn}\left(\mathbf{w}_i \xi^\mu - \theta_i\right) \quad \text{bzw.} \quad \xi_i^{\sigma(\mu)} = \operatorname{stp}\left(\mathbf{w}_i \xi^\mu - \theta_i\right) \tag{4.48}$$

Um dieses Problem zu lösen, kann man den bekannten Perzeptron Lernalgorithmus verwenden, der für eine $\{-1, +1, \}$-Kodierung folgendermaßen aussieht.

$$\begin{aligned} \mathbf{w}_i(s+1) &= \mathbf{w}_i(s) + \eta \xi^\mu (\xi_i^{\sigma(\mu)} - \operatorname{sgn}(\mathbf{w}_i \xi^\mu - \theta_i)) \\ \theta_i(s+1) &= \theta_i(s) - \eta (\xi_i^{\sigma(\mu)} - \operatorname{sgn}(\mathbf{w}_i \xi^\mu - \theta_i)) \end{aligned} \tag{4.49}$$

Wobei hier s die Lernschritte bezeichnet und η die Lernrate, welche je nach Art der Kodierung unterschiedlich ausfällt. Hat man eine $\{0, 1\}$-Kodierung zugrundeliegen, so ist die Funktion sgn einfach durch stp zu ersetzen. Aufgrund des Perzeptron-Konvergenz-Theorems (vgl. z.B. Abschnitt 3.4 [14]) ist bewiesen, daß, wenn die Mengen $\mathcal{X}_i^+ := \{\xi^\mu : \xi_i^{\sigma(\mu)} = 1\}$ und $\mathcal{X}_i^- := \{\xi^\mu : \xi_i^{\sigma(\mu)} = -1\ (0)\}$ linear separierbar sind, dieser Algorithmus auch tatsächlich eine Lösung findet.

Die Speicherkapazität kann man wieder so definieren, daß man sich fragt, wie wahrscheinlich es ist, daß die durch eine zufällig generierte Menge $\mathcal{X}$ von Mustern und eine zufällig gewählte Funktion σ erzeugten Mengen $\mathcal{X}_i^+$ und $\mathcal{X}_i^-$ linear separierbar sind.

1965 gelang es Cover diese zu berechnen (vgl. Abschnitt 5.7 [14]). Die prinzipielle Aussage dieser Berechnung ist, daß es eine Schranke L_{max} gibt, oberhalb derer die Mengen $\mathcal{X}_i^+$ und $\mathcal{X}_i^-$ nur mehr mit gegen 0 gehender Wahrscheinlichkeit linear separierbar sind. Unterhalb davon ist diese Wahrscheinlichkeit beinahe 1. Dieser Verlauf ist in Abbildung 4.7 A dargestellt. Man sieht, daß mit steigendem N dieser Abfall immer steiler wird.

Man definiert nun die Speicherkapazität α_c als den Wert, bei dem diese Kurve die 50% Marke schneidet. Wie man aus Abbildung 4.7 A sieht, ist dies für den Fall, daß die Wahrscheinlichkeit p des Auftretens von $+1$ in einem Muster $1/2$ ist, gleich 2 ($m = 2p - 1$ ist wie üblich die mittlere Aktivität in einem Muster

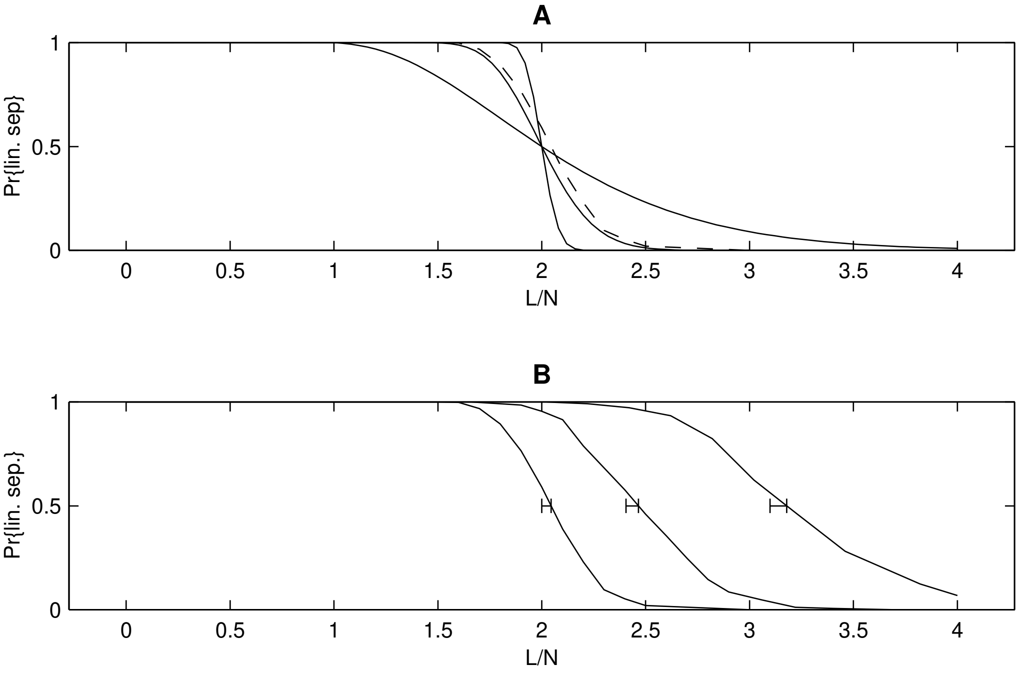
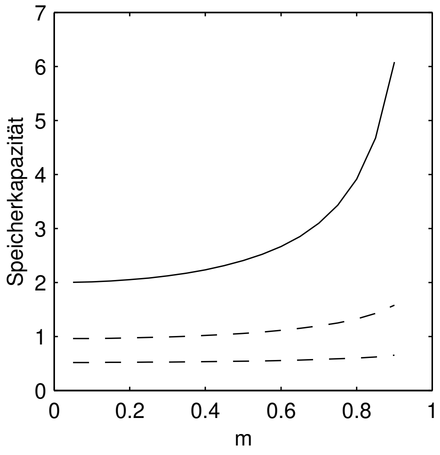


Abbildung 4.7: Darstellung des Verlaufes der Wahrscheinlichkeit, daß ein Perzeptron i die zwei durch $\mathcal{X}$ und σ induzierten Mengen $\mathcal{X}_i^+$ und $\mathcal{X}_i^-$ trennen kann. **A**: Theoretischer Verlauf für $m = 0$ (m ist die mittlere Aktivität in einem Muster ξ^μ(nach Abschnitt 5.7 in [14]). Es sind die Kurven für $N = 5$, 50 und 500 dargestellt (durchgezogene Linien). Je größer N umso steiler ist der Abfall. Die strichlierte Linie, stellt das Ergebnis einer entsprechenden Simulation dar. **B**: Ergebnisse von Simulationen. Zu sehen sind Kurven für die Werte $m = 0$, 0.5 und 0.7 ($N = 50$). Je größer m wird, je später tritt der Abfall gegen 0 ein. Die Markierungen zeigen, wie weit die Simulation von den theoretischen Werten, die der Tabelle in Abbildung 4.8 zu entnehmen sind, entfernt sind. In den Simulationen wurde in über 500 Durchläufen mittels linearer Programmierung entschieden (vgl. [31]), ob die durch Zufall entstandenen Mengen $\mathcal{X}_i^+$ und $\mathcal{X}_i^-$ linear trennbar sind oder nicht. Daraus wurde dann durch Mittelung die Wahrscheinlichkeit Pr{lin. sep.} berechnet.



m	α_c
0.00	2.0
0.10	2.012
0.20	2.052
0.30	2.124
0.40	2.236
0.50	2.405
0.60	2.667
0.70	3.098
0.80	3.912
0.90	6.082

Abbildung 4.8: Das Diagramm zeigt die Abhängigkeit der Speicherkapazität von m. Erstellt nach Gleichung (7.63) in [25]. Die strichlierten Linien stellen den Verlauf dar, wenn $\kappa \neq 0$ ist (bezüglich κ siehe Text). Die Tabelle listet einige Wertepaare entsprechend der durchgezogenen Linie auf ($\kappa = 0$).

ξ^{μ}). Das Ergebnis, der als strichlierten Linie dargestellten Simulation ist ähnlich. Der Unterschied ist in Abbildung 4.7 B als waagrecht Markierung angegeben.

Für den Fall, daß p nicht mehr 1/2 ist, versagt die Abschätzung von Cover. 1987 gelang es jedoch Gardner auch für diesen allgemeineren Fall mit Hilfe der statistischen Mechanik die Speicherkapazität abzuschätzen. In seiner Berechnung kommt zusätzlich noch ein Wert κ vor, welcher im Prinzip angibt, wie weit die einzelnen Punkt von der sie trennenden Hyperebene entfernt sein müssen (vgl. Abschnitt 6.3 in [25]).

Die zentrale Aussage dieser Abschätzung ist, daß mit zu- oder abnehmendem m die Speicherkapazität ansteigt, was eigentlich nicht sonderlich überrascht, da Muster mit hauptsächlich einem Bitwert sehr wenig Information enthalten. Der Verlauf der Speicherkapazität in Abhängigkeit von m ist in Abbildung 4.8 dargestellt. Diese Kurven wurden nach der in [25] angegebenen Formel (7.63) erstellt.

In Abbildung 4.7 sind nun als waagrechte Markierungen die Abweichung zwischen den Werten aus der Tabelle in Abbildung 4.8 und den Simulationen eingezeichnet. Man sieht hier sehr schön, wie nahe die Abschätzung an die Simulationsergebnisse herankommt. Die Differenz beträgt 0.0446, 0.0605 und 0.081 für die entsprechenden Werte 0, 0.5 und 0.7 für m.

Kapitel 5

Mehrere Synapsen pro Neuronenpaar

5.1 Allgemeines

Im letzten Abschnitt haben wir Netzwerke betrachtet, die zwischen einem Neuronenpaar nur eine Synapse mit einer bestimmten Verzögerungszeit $d_{i,j}$ aufwiesen.

Bei der Betrachtung des stark abstrahierten theoretischen Modells hat sich gezeigt, das die Sequenzen die mit solchen Netzwerken generiert werden können, einigen Einschränkungen unterliegen, die alle auf die Lösbarkeit des in Abschnitt 4.7 eingeführten Klassifizierungsproblems zurückzuführen sind (vgl. Abschnitt 4.3).

In diesem Abschnitt sollen nun Netzwerke betrachtet werden, die zwischen zwei Neuronen mehrere Synapsen aufweisen können. Es wird sich herausstellen, daß dadurch die Einschränkungen denen die möglichen Sequenzen unterliegen etwas aufgeweicht werden.

Die biologische Plausibilität dieser Art von Netzwerken kann zur Zeit nicht exakt beurteilt werden, da es noch an Meßmethoden fehlt mit denen man genau eruieren könnte wieviele Verbindungen pro Neuronenpaar tatsächlich existieren.

In den ersten Abschnitten dieses Kapitels werden wir wieder das abstrakte Modelle eines NFN betrachten, welches in Abschnitt 4.1 definiert wurde. Wir haben es also wieder mit zeitlich diskreten Sequenzen zu tun.

5.1.1 Netzwerkdynamik

Das Netzwerk wird nur dahingehen abgeändert, daß Anstelle der einen Funktion $\varepsilon_{i,j}$ nun mehrere Funktionen $\varepsilon_{i,j}^{(d)}$ treten. Dabei wird mit $d \in \mathcal{D}$ ($\mathcal{D} \ldots$ Menge der möglichen Verzögerungszeiten) die Verzögerungszeit bezeichnet. Für den speziellen Fall, daß alle Neuronen wieder synchron feuern sollen, werden diese Delays ein ganzzahliges Vielfaches der in Abschnitt 4.1 eingeführten Verzögerungszeit d_0

sein.

$$\mathcal{D} = \{\delta d_0 : \delta \in \{1 \dots D\}\} \tag{5.1}$$

Wie schon in Abschnitt 4.1 erläutert, kann man das Verhalten eines solchen Netzwerkes einfach anhand des Zustandsvektors $\mathbf{x}(k)$ beschreiben.

$$x_i(k) = \operatorname{sgn}(h_i(k)) = \operatorname{sgn}\left(\sum_{\delta=1}^{D}\sum_{j=1}^{N} x_j(k-\delta) w_{i,j}^{(\delta)} - \theta_i\right) \tag{5.2}$$

Wobei hier δ die diskrete Verzögerungszeit darstellt, welche zum tatsächlichen Delay δd_0 korrespondiert. $w_{i,j}^{(\delta)}$ gibt das Gewicht der Synapse an, welche die Verzögerungszeit δ realisiert.

5.1.2 Komplexe Sequenzen

Die erste Einschränkung, der die Sequenzen bisher unterlagen, war, daß nur einfache Sequenzen erzeugt werden konnten. Da jetzt aber ein Neuron Zugriff auf mehrere vergangene Netzwerkzustände hat, können prinzipiell auch komplexe Sequenzen erzeugt werden (vgl. Abschnitt 2.5).

Es ist daher nicht mehr zielführend, die zu speichernden Sequenzen $S^{(1)}, \dots S^{(M)}$ durch eine Funktion $\sigma : \{1 \dots P\} \to \{1 \dots P\}$ (P ... Anzahl der an den Sequenzen beteiligten Systemzuständen ξ^μ) zu definieren. Statt dessen ist es einfach, entsprechend der Definition des Kontextes eines Symbols einer Sequenz die hier sogenannte Kontextabbildung $\mathbf{c} : \{1 \dots P\} \to \{1 \dots P\}^D$ zu definieren. $\mathbf{c}(\mu) = [c_1(\mu) \cdots c_D(\mu)]^T$ bezeichnet den Kontext des Netzwerkzustandes μ. Das heißt jene D vorausgegangenen Netzwerkzustände $\xi^{c_\delta(\mu)}$, welche $\delta \in \{1, \dots, D\}$ Zeitschritte vorher anliegen sollten, sodaß zum Zeitpunkt k das Netzwerk den Zustand $\mathbf{x}(k) = \xi^\mu$ annimmt ($\mathbf{x}(k-\delta) = \xi^{\mathbf{c}_\delta(\mu)}$).

Die Abbildung $\mathbf{c}$ braucht nicht eindeutig zu sein, es können ja zwei verschiedene Kontexte zum selben Netzwerkzustand führen (vgl. Abbildung 5.1).

Aus diesem Grund kann nun die Anzahl P der beteiligten Netzwerkzustände von der Anzahl der zu speichernden Netzwerkzustandsübergänge L verschieden sein.

Man sieht somit sehr leicht, daß der maximale Grad der Sequenzen, die von einem Netzwerk mit Verzögerungszeiten $\delta \in \{1, \dots D\}$ erzeugt werden können, D beträgt.

5.1.3 Einige Definitionen

Folgende Definitionen werden im Laufe dieses Abschnitts noch öfters benötigt.

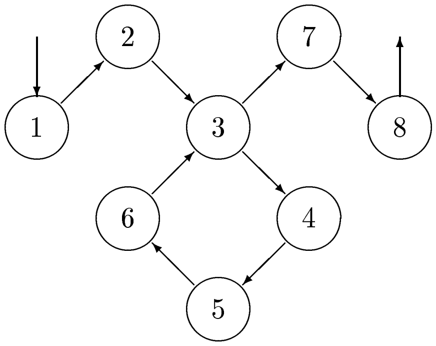


Abbildung 5.1: Eine komplexe Sequenz. Die zu dieser Sequenz gehörende Kontextfunktion wäre nicht eindeutig, da dem Netzwerkzustand 3 zwei verschiedene Kontexte vorausgehen. Einmal wäre dies (1,2) und ein andermal (5,6).

Unter einem *Kontextvektor* $\zeta^{\mu,l}$ versteht man jenen Spaltenvektor, der aus der Aneinanderreihung der Vektoren $\xi^{c_1(\mu)}, \dots, \xi^{c_D(\mu)}$ entsteht. Der Index l soll andeuten, daß es zu einem Vektor ξ^μ ja mehrere Kontexte geben kann (vgl. oben).

Unter der Matrix $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}_{(0)}$ versteht man jene Matrix, die entsteht, wenn man die P Vektoren ξ^μ entsprechen ihrer Kontexte aneinanderreiht. Das soll heißen, daß ein Vektor, der bei den zu speichernden Sequenzen zweimal vorkommt und daher zwei Kontexte hat, auch in der Matrix $\boldsymbol{\mathcal{X}}$ zweimal auftaucht (ein Beispiel wäre der Vektor 3 in Abbildung 5.1).

Die Matrix $\boldsymbol{\mathcal{X}}_{(\delta)}$ ist die Aneinanderreihung der vor δ Zeiteinheiten anzutreffenden Systemzustände, die dann zu ξ^μ führen ($\boldsymbol{\mathcal{X}}_{(\delta)} = [\xi^{\mathbf{c}_\delta(\mu)}]_{\mu=1\dots P}$, wobei alle Zweideutigkeiten der Abbildung $\mathbf{c}$ zu berücksichtigen sind).

Die *Kontextmatrix* $\boldsymbol{\mathcal{C}}$ besteht aus allen L Kontextvektoren $\zeta^{\mu,l}$ und ist daher eine $ND \times L$ Matrix.

$$\boldsymbol{\mathcal{C}} := \left[\zeta^{1,1}\,\zeta^{1,2}\cdots\zeta^{\mu,l}\cdots\right] \quad \text{oder} \quad \boldsymbol{\mathcal{C}} = \begin{bmatrix} \boldsymbol{\mathcal{X}}_{(1)} \\ \boldsymbol{\mathcal{X}}_{(2)} \\ \vdots \\ \boldsymbol{\mathcal{X}}^T_{(D)} \end{bmatrix} \tag{5.3}$$

Die Gewichte $w_{i,j}^{(\delta)}$ lassen sich natürlich zu eine Gewichtsmatrix $\mathbf{W}_{(\delta)} = [w_{i,j}^{(\delta)}]$ $i,j = 1,\dots,N$ zusammenfassen. Die Erweiterte Gewichtsmatrix $\mathbf{W}$ soll im folgenden die Aneinanderreihung der D Gewichtsmatrizen $\mathbf{W}_{(\delta)}$ darstellen.

$$\mathbf{W} := [\mathbf{W}_{(1)}\mathbf{W}_{(2)}\cdots\mathbf{W}_{(D)}] \tag{5.4}$$

Fügt man diesen Definitionen noch jene des erweiterten Zustandsvektors $\widetilde{\mathbf{x}}(k) = [\mathbf{x}^T(k)\ \mathbf{x}^T(k-1)\cdots\mathbf{x}^T(k-D+1)]^T$ hinzu, so kommt man zu einer sehr kompakten Matrixschreibweise für Gleichung 5.2.

$$\mathbf{x}(k) = \operatorname{sgn}(\mathbf{h}(k)) = \operatorname{sgn}\left(\mathbf{W}\widetilde{\mathbf{x}}(k-1) - \boldsymbol{\theta}\right) \tag{5.5}$$

5.2 Analyse des globalen Verhaltens

Für beliebige Matrizen $\mathbf{W}$ kann man sehr wenig über den zeitlichen Verlauf des Zustandsvektors $\mathbf{x}(k)$ aussagen. Um genauere Angaben machen zu können, muß man $\mathbf{W}$ einschränken.

5.2.1 Ein äquivalentes Netzwerk

Um die Untersuchungen zu vereinfachen, soll zunächst ein zu einem gegebenen Netzwerk $\mathcal{N}$ (definiert durch die Gewichtsmatrix $\mathbf{W}$ und den Schwellwertvektor $\boldsymbol{\theta}$) ein Netzwerk $\widehat{\mathcal{N}}$ (definiert durch die Gewichtsmatrix $\widehat{\mathbf{W}}$ und den Schwellwertvektor $\hat{\boldsymbol{\theta}}$) konstruiert werden, welches nur eine Synapse pro Neuronenpaar besitz, aus dessen Zustandsvektor $\widehat{\mathbf{x}}$ man den Zustandsvektor $\mathbf{x}$ des originalen Netzwerkes $\mathcal{N}$ zu jeder Zeit k ablesen kann und umgekehrt (vgl. [17] und [31]).

Betrachten wir dazu das Netzwerk $\widehat{\mathcal{N}}$ mit der Gewichtsmatrix $\widehat{\mathbf{W}}$ und dem Schwellwertvektor $\hat{\boldsymbol{\theta}}$, die wie folgt definiert sein sollen.

$$\widehat{\mathbf{W}} = \begin{bmatrix} \mathbf{W}_{(D)} & \mathbf{W}_{(1)} & \mathbf{W}_{(2)} & \mathbf{W}_{(3)} & \cdots & \mathbf{W}_{(D-1)} \\ \mathbf{W}_{(D-1)} & \mathbf{W}_{(D)} & \mathbf{W}_{(1)} & \mathbf{W}_{(2)} & \cdots & \mathbf{W}_{(D-2)} \\ \mathbf{W}_{(D-2)} & \mathbf{W}_{(D-1)} & \mathbf{W}_{(D)} & \mathbf{W}_{(1)} & \cdots & \mathbf{W}_{(D-3)} \\ \vdots & & & & & \vdots \\ \mathbf{W}_{(1)} & \mathbf{W}_{(2)} & \mathbf{W}_{(3)} & \cdots\cdots\cdots & & \mathbf{W}_{(D)} \end{bmatrix}, \quad \hat{\boldsymbol{\theta}} = \begin{bmatrix} \boldsymbol{\theta} \\ \vdots \\ \boldsymbol{\theta} \end{bmatrix} \tag{5.6}$$

Für die folgenden Betrachtungen ist es am günstigsten, sich den Zustandsvektor $\widehat{\mathbf{x}}(t)$ als Aneinanderreihung der D N-dimensionalen Teilvektoren $\widehat{\mathbf{x}}_{(D)}, \ldots, \widehat{\mathbf{x}}_{(1)}$ zu veranschaulichen ($\widehat{\mathbf{x}}(k) = [\widehat{\mathbf{x}}^T_{(D)}(k)\ \widehat{\mathbf{x}}^T_{(D-1)}(k)\cdots\widehat{\mathbf{x}}^T_{(1)}(k)]^T$).

Die Neuberechnung des Netzwerkzustandes erfolgt nun nicht mehr wie im originalen Netzwerk für alle Neuronen gleichzeitig, sondern nur immer für einen Teilvektor $\widehat{\mathbf{x}}_{(v)}$ parallel. Die einzelnen Teilvektoren, werden in ihrer natürlichen Reihenfolge $1, 2, \ldots, D, 1, 2, \ldots$ durchlaufen. Versteht man unter x MOD y die Funktion $((x-1) \mod y) + 1$ so kann man dieses Berechnungsverfahren folgendermaßen anschreiben.

$$\widehat{\mathbf{x}}_{(v)}(k+1) = \operatorname{sgn}\left(\sum_{\delta=1}^{D} \mathbf{W}_{(\delta)}\ \widehat{\mathbf{x}}_{(D-\delta+v \text{ MOD } D)}(k) - \boldsymbol{\theta}\right) \tag{5.7}$$

falls $v = (k+1) \text{ MOD } D$, andernfalls wird $\widehat{\mathbf{x}}_{(v)}$ nicht neu berechnet ($\widehat{\mathbf{x}}_{(v)}(k+1) = \widehat{\mathbf{x}}_{(v)}(k)$). Die Summe in dieser Gleichung entspricht einfach der Matrizenmultiplikation der v-ten Matrizenreihe von unten von $\widehat{\mathbf{W}}$ mit $\widehat{\mathbf{x}}(k)$. Diese Art von Update wird in der Literatur oft auch als block-sequentieller Update bezeichnet (vgl. Kapitel 1 in [19]).

Vergleicht man Gleichung 5.7 mit Gleichung 5.2 so sieht man, daß hier in Abhängigkeit von δ auf verschiedene räumliche Teile des Zustandsvektors $\widehat{\mathbf{x}}(k)$ zugegriffen wird, und nicht auf den δ Zeitschritte zurückliegenden Netzwerkzustand. Durch diesen Berechnungsvorgang wird jedoch ein Teilvektor $\widehat{\mathbf{x}}_{(v)}$ nur alle D Zeiteinheiten aktualisiert und steht daher inzwischen ebenso als Speicher eines früheren Netzwerkzustandes zur Verfügung.

Sei nun k ein beliebiger Zeitpunkt für den gilt $(k+1) \text{ MOD } D = 1$ und $\widehat{\mathbf{x}}(k) = \widetilde{\mathbf{x}}(k)$, das heißt $\widehat{\mathbf{x}}_{(D)}(k) = \mathbf{x}(k)$, $\widehat{\mathbf{x}}_{(D-1)}(k) = \mathbf{x}(k-1), \ldots, \widehat{\mathbf{x}}_{(1)}(k) = \mathbf{x}(k-D+1)$. Setzt man diese Bedingungen in Gleichung 5.7 ein, so erhält man:

$$\widehat{\mathbf{x}}_{(1)}(k+1) = \operatorname{sgn}\left(\sum_{\delta=1}^{D} \mathbf{W}_{(\delta)}\, \mathbf{x}(k-\delta+1) - \boldsymbol{\theta}\right) \tag{5.8}$$

$$\widehat{\mathbf{x}}_{(v)}(k+1) = \mathbf{x}(k-D+v) \quad \text{falls} \quad v \neq 1 \tag{5.9}$$

$\widehat{\mathbf{x}}_{(1)}(k+1)$ ist aber laut Gleichung 5.5 nichts anderes als $\mathbf{x}(k+1)$. Somit ist der D Zeitschritte zurückliegende Wert $\mathbf{x}(k-D+1)$ von $\mathbf{x}(k+1)$ überschrieben worden. Führt man nun den nächsten Berechnungsschritt durch, das heißt es wird $\widehat{\mathbf{x}}_{(2)}(k+2)$ berechnet so wird $\mathbf{x}(k-D+2)$ überschrieben und zwar mit $\mathbf{x}(k+2)$. Dies ergibt sich einfach durch einsetzen des Vektors $\widehat{\mathbf{x}}(k+1)$ in Gleichung 5.7. Spielt man diese Berechnungsschritte weiter durch (die nächsten D Berechnugsschritte), so ergibt sich, daß jede Berechnung eines Wertes $\widehat{\mathbf{x}}_{(v)}$ einem neuen Netzwerkzustand $\mathbf{x}$ entspricht.

$$\mathbf{x}(k+1) = \widehat{\mathbf{x}}_{(v)}(k+1), \quad \text{mit} \quad v = (k+1) \text{ MOD } D \tag{5.10}$$

Wird das Netzwerk $\widehat{\mathcal{N}}$ zum Zeitpunkt $k = 0$ derart initialisiert, daß $\widehat{\mathbf{x}}(0) = \widetilde{\mathbf{x}}(0)$[1] ist, so kann der zeitliche Verlauf des Vektors $\widehat{\mathbf{x}}(k)$ aus dem des Netzwerkzustandes $\mathbf{x}(k)$ des Netzwerkes $\mathcal{N}$ bestimmt werden und umgekehrt. Dieses Ergebnis ist in folgendem Satz zusammengefaßt.

Satz 5.1. *Das Netzwerk $\widehat{\mathcal{N}}$ welches durch Gleichung 5.6 definiert wird, ist in der Hinsicht dem Netzwerk $\mathcal{N}$ (gegeben durch $\mathbf{W}$ und $\boldsymbol{\theta}$) äquivalent, daß eine eindeutige Zuordnung der Netzwerkzustände nach Gleichung 5.10 gegeben ist, sobald die beiden Netzwerke entsprechend initialisiert werden.*

---

[1] Eine Festlegung des Vektors $\widetilde{\mathbf{x}}(0)$ entspricht einer Initialisierung der Werte $\mathbf{x}(-1)$ bis $\mathbf{x}(-D+1)$.

5.2.2 Erweiterte synaptische Symmetrie

Satz 5.1 kann man nun dazu verwenden, um das originale Netzwerk zu analysieren, ohne sich um die vielen verschiedenen Verbindungen zwischen den einzelnen Neuronenpaaren kümmern zu müssen.

Eine interessant Einschränkung der Gewichtsmatrix $\widehat{\mathbf{W}}$ besteht darin, daß man verlangt, daß sie symmetrisch ist ($\widehat{\mathbf{W}} = \widehat{\mathbf{W}}^T$). Betrachtet man sich die Definition von $\widehat{\mathbf{W}}$ (vgl. Gleichung 5.6), so sieht man, daß man damit die sogenannte *erweiterte synaptische Symmetrie* fordert (vgl. [17]).

$$\mathbf{W}_{(\delta)} = \mathbf{W}^T_{(D-\delta)} \tag{5.11}$$

Fügt man dieser Forderung weiters noch hinzu, daß die Matrix $\mathbf{W}_{(D)}$ auf der Menge $\{-1, 0, +1\}^N$ positiv definit ist[2], so kann man unter der Voraussetzung des in Gleichung 5.7 definierten Berechnungsverfahrens zeigen, daß das erweiterte Netzwerk immer einen Fixpunkt ($\widehat{\mathbf{x}}(k+1) = \widehat{\mathbf{x}}(k)$) erreichen muß.

Um dies zu zeigen, definiert man eine Energiefunktion $E(k)$, von der man anschließenden feststellen kann, daß sie monoton fallend ist.

$$E(k) = 2\widehat{\mathbf{x}}^T(k)\hat{\boldsymbol{\theta}} - \widehat{\mathbf{x}}^T(k)\widehat{\mathbf{W}}\widehat{\mathbf{x}}(k) \tag{5.12}$$

Um die Monotonie festzustellen, betrachtet man am besten die Energiedifferenz $\Delta E = E(k+1) - E(k)$ zweier aufeinander folgender Systemzustände $\widehat{\mathbf{x}}(k+1)$ und $\widehat{\mathbf{x}}(k)$. Mit $\Delta\widehat{\mathbf{x}}(k) = \widehat{\mathbf{x}}(k+1) - \widehat{\mathbf{x}}(k)$ kann man diese in folgender Art beschreiben.

$$\begin{aligned} \Delta E = {} & 2\widehat{\mathbf{x}}^T(k+1)\hat{\boldsymbol{\theta}} - (\widehat{\mathbf{x}}(k) + \Delta\widehat{\mathbf{x}}(k))^T\widehat{\mathbf{W}}(\widehat{\mathbf{x}}(k) + \Delta\widehat{\mathbf{x}}(k)) \\ & - 2\widehat{\mathbf{x}}^T(k)\hat{\boldsymbol{\theta}} + \widehat{\mathbf{x}}^T(k)\widehat{\mathbf{W}}\widehat{\mathbf{x}}(k) \end{aligned} \tag{5.13}$$

$$\begin{aligned} \Delta E = {} & 2\Delta\widehat{\mathbf{x}}^T(k)\hat{\boldsymbol{\theta}} - \Delta\widehat{\mathbf{x}}^T(k)\widehat{\mathbf{W}}\Delta\widehat{\mathbf{x}}(k) + \\ & - \Delta\widehat{\mathbf{x}}^T(k)\widehat{\mathbf{W}}\widehat{\mathbf{x}}(k) - \Delta\widehat{\mathbf{x}}^T(k)\widehat{\mathbf{W}}^T\widehat{\mathbf{x}}(k) \end{aligned} \tag{5.14}$$

Da laut Voraussetzung $\widehat{\mathbf{W}} = \widehat{\mathbf{W}}^T$ ist, ergibt sich weiter

$$\Delta E = 2\Delta\widehat{\mathbf{x}}^T(k)\hat{\boldsymbol{\theta}} - 2\Delta\widehat{\mathbf{x}}^T(k)\widehat{\mathbf{W}}\widehat{\mathbf{x}}(k) - \Delta\widehat{\mathbf{x}}^T(k)\widehat{\mathbf{W}}\Delta\widehat{\mathbf{x}}(k) \tag{5.15}$$

$$\Delta E = -2\Delta\widehat{\mathbf{x}}^T(k)\left(\widehat{\mathbf{W}}\widehat{\mathbf{x}}(k) - \hat{\boldsymbol{\theta}}\right) - \Delta\widehat{\mathbf{x}}^T(k)\widehat{\mathbf{W}}\Delta\widehat{\mathbf{x}}(k) \tag{5.16}$$

$$\Delta E = -2\Delta\widehat{\mathbf{x}}^T(k)\widehat{\mathbf{h}}(k+1) - \Delta\widehat{\mathbf{x}}^T(k)\widehat{\mathbf{W}}\Delta\widehat{\mathbf{x}}(k) \tag{5.17}$$

Berücksichtigt man nun den Berechnungsmechanismus laut Gleichung 5.7 so ergibt sich für $\Delta\widehat{\mathbf{x}}(k)$, das nur an jener Stelle von Null verschiedene Element auf-

---

[2] Eine Matrix $\mathbf{W}$ ist auf einer Menge A dann positiv definit ($\mathbf{W} \geq 0$), wenn für jedes Element $\mathbf{a} \in A$ gilt $\mathbf{a}^T\mathbf{W}\mathbf{a} \geq 0$. Der einfachste Fall eine positiv definiten Matrix ist die Nullmatrix $\mathbf{0}$.

treten können, an der $\widehat{\mathbf{x}}(k)$ auch tatsächlich neu berechnet wird.

$$\Delta\widehat{\mathbf{x}}(k) = \begin{bmatrix} 0 \\ \vdots \\ \Delta\widehat{\mathbf{x}}_{(v)}(k) \\ \vdots \\ 0 \end{bmatrix}, \quad \text{für} \quad v = (k+1) \text{ MOD } D \tag{5.18}$$

Damit ergibt sich für ΔE:

$$\Delta E = -2\Delta\widehat{\mathbf{x}}^T_{(v)}(k)\widehat{\mathbf{h}}_{(v)}(k+1) - \widehat{\mathbf{x}}^T_{(v)}(k)\mathbf{W}_{(D)}\widehat{\mathbf{x}}_{(v)}(k) \tag{5.19}$$

Wobei $\widehat{\mathbf{h}}_{(v)}$ der zu $\widehat{\mathbf{x}}_{(v)}$ korrespondierende Teil des Potentialvektors $\widehat{\mathbf{h}}(t+1) = \widehat{\mathbf{W}}\widehat{\mathbf{x}}(k) - \widehat{\boldsymbol{\theta}}$ ist. Der Zusammenhang $\widehat{\mathbf{x}}_{(v)}(k+1) = \operatorname{sgn}(\widehat{\mathbf{h}}_{(v)}(k+1))$ bzw. $\widehat{\mathbf{x}}_{(v)}(k+1) = \operatorname{stp}(\widehat{\mathbf{h}}_{(v)}(k+1))$ hat zur Folge, daß alle einzelnen Komponenten des Vektorprodukts $\Delta\widehat{\mathbf{x}}^T_{(v)}(k)\widehat{\mathbf{h}}_{(v)}(k+1)$ größer gleich 0 sind. (Dies kann man leicht dadurch prüfen, indem man sich einfach alle möglichen Belegungen von $\widehat{x}_{(v),i}(k)$ bzw. $\widehat{x}_{(v),i}(k+1)$ anschaut und sich daraus das Vorzeichen von $\widehat{h}_{(v),i}(k+1)$ überlegt und alle Resultate zusammensetzt.) Damit und der Voraussetzung, daß $\mathbf{W}_{(D)}$ auf der Menge $\{-1, 0, +1\}^N$ positiv definit ist, sieht man, daß ΔE kleiner gleich 0 ist.

Als nächsten Schritt stellt man nun fest, daß $E(k)$ aus einer endlichen Summe von endlichen Größen besteht und somit nach unten beschränkt ist. Da aber mit jedem Berechnungsschritt $E(k)$ nur kleiner und nicht größer werden kann, muß ein Punkt erreicht werden, ab dem sich die Energie nicht mehr ändert.

Es bleibt nur noch zu zeigen, daß aus der Konvergenz der Energiefunktion auch eine Konvergenz des Zustandsvektors $\widehat{\mathbf{x}}(k)$ folgt. Da $-\Delta E$ eine Summe aus lauter Termen ist, welche ihrerseits alle größer gleich 0 sind ($\Delta\widehat{\mathbf{x}}^T(k)\mathbf{W}_{(D)}\Delta\widehat{\mathbf{x}}(k) \geq 0$, $\Delta\widehat{x}_{(v),i}(k)\widehat{h}_{(v),i}(k+1) \geq 0$), bedeutet $\Delta E = 0$ auch, das jeder einzelne dieser Terme gleich 0 ist.

Für die Summanden $\Delta\widehat{x}_{(v),i}(k)\widehat{h}_{(v),i}(k+1)$ heißt das nun, daß entweder $\Delta\widehat{x}_{(v),i}(k)$ oder $\widehat{h}_{(v),i}(k+1)$ 0 sein muß. Denn Fall, daß $\widehat{h}_{(v),i}(k+1) = 0$ ist, kann man ausschließen, da man aufgrund der endlichen Menge von Punkten, die ein einzelnes Gatter als Input bekommen kann, $\widehat{\mathbf{W}}$ derart abändern kann, so daß für alle $\widehat{\mathbf{x}}(k) \in \{+1, -1\}^{ND}$ ($\widehat{\mathbf{x}}(k) \in \{+1, 0\}^{ND}$) gilt $\widehat{h}_{(v),i}(k+1) = \sum_{j=1}^{ND} \widehat{w}_{i,j}\widehat{\mathbf{x}}_j(k) - \widehat{\theta}_i > 0$. Daher muß $\Delta\widehat{x}_{(v),i}(k) = 0$ sein, womit aus der Konvergenz der Energiefunktion folgt, das auch der Zustandsvektor $\widehat{\mathbf{x}}(k)$ sich nicht mehr ändert.

Was bedeutet das nun für das originale Netzwerk $\mathcal{N}$? Aufgrund des Zusammenhanges, der durch Gleichung 5.10 gegeben ist, kann dieses Netzwerk maximal einen Zyklus der Länge D durchlaufen und zwar dann, wenn alle Teilvektoren $\widehat{\mathbf{x}}_{(v)}$ des erreichten Fixpunktes verschieden sind. Es kann sich jedoch auch eine Periode einstellen, die einem ganzzahligen Bruch von D entspricht. Anders ausgedrückt, kann man sagen, daß ein Fixpunkt im äquivalenten System $\widehat{\mathcal{N}}$ einem Grenzzyklus im originalen Netzwerk entspricht. Der folgende Satz faßt das Ergebnis dieses Abschnitts zusammen.

Satz 5.2. *Ein Netzwerk $\mathcal{N}$, welches pro Neuronenpaar D Verbindungen mit Verzögerungszeiten $\delta \in \{1, \ldots, D\}$ besitz, und dessen Gewichte die sogenannte erweiterte synaptische Symmetrie $w_{i,j}^{(\delta)} = w_{j,i}^{(D-\delta \text{ MOD } D)}$ und die Bedingung, daß $\mathbf{W}_{(D)}$ positiv definit ist, erfüllen, kann nur in einen Grenzzyklus münden, dessen Periode einem ganzzahligen Bruch von D entspricht.*

Die Argumente die zu diesem Satz geführt haben, zeigen einen weiteren Fakt. Der zeitliche Verlauf des Systemzustandes eines Netzwerkes, dessen Gewichte die im obigen Satz formulierten Bedingungen erfüllen, wird durch eine Energie- oder Lyapunovfunktion bestimmt. Auf Grund dessen kann man sich die Bahn von $\mathbf{x}(k)$ im Zustandsraum auch als eine Minimumsuche in einer abstrakten spatiotemporären Energielandschaft vorstellen. Schreibt man nämlich $E(k)$ mit den Größen des originalen Netzwerkes an so erhält man einen Ausdruck wo in die Energie auch vergangene Systemzustände einfließen (vgl. [17]).

$$E(k) = \sum_{a,\delta=1}^{D} \sum_{i,j}^{N} x_i(k-a) w_{i,j}^{(\delta)} x_j(k-\delta) \tag{5.20}$$

(In dieser Gleichung wurde $\boldsymbol{\theta} = \mathbf{0}$ gesetzt). Es ist noch anzumerken, daß ein Zyklus im originalen Netzwerk D verschiedenen Fixpunkten im äquivalenten System entspricht. Diese Fixpunkte sind alle verschobene Versionen eines Fixpunktes (verschoben in Bezug auf die Teilvektoren $\widehat{\mathbf{x}}_{(v)}$). Dies ist leicht einsichtig, da ja a priori nicht feststeht, was der „erste" Zustand in einem Zyklus sein soll.

In den nächsten Abschnitten sollen Methoden vorgestellt werden, wie vorgegebene Sequenzen in einem Netzwerk gespeichert werden können, so daß sie wieder reproduziert werden können.

5.3 Die erweiterte Projektionsregel

5.3.1 Das Prinzip

Das Prinzip ist hier das selbe, wie es schon in Abschnitt 4.6 vorgestellt wurde. Es wird ein Gleichungssystem aufgestellt, welches, wenn nicht zu viele Zustandsübergänge gespeichert werden, sollen exakt gelöst werden kann.

Ein Zustandsübergang ist durch die Kontextfunktion $\mathbf{c}(\mu)$ bzw. den daraus resultierenden Kontextvektoren ζ^{μ_l} definiert. Es soll zum Zeitpunkt k der Systemzustand ξ^{μ} angenommen werden, wenn einer der Kontexte ζ^{μ_l} auftritt ($\widetilde{\mathbf{x}}(k-1) =$

ζ^{μ_l}). Mit den in Abschnitt 5.1.3 definierten Größen $\boldsymbol{\mathcal{X}}$, $\mathbf{W}$, $\boldsymbol{\mathcal{C}}$ und $\boldsymbol{\theta}$, kann man das geforderte verhalten des Netzwerkes in einfacher Form niederschreiben.

$$\boldsymbol{\mathcal{X}} = \mathrm{sgn}(\mathbf{W}\boldsymbol{\mathcal{C}} - \boldsymbol{\theta}) \quad \text{bzw.} \quad \boldsymbol{\mathcal{X}} = \mathrm{stp}(\mathbf{W}\boldsymbol{\mathcal{C}} - \boldsymbol{\theta}) \tag{5.21}$$

Wie in Abschnitt 4.6 bereits ausgeführt wurde, kann man die Gewichtsmatrix $\mathbf{W}$ daraus direkt berechnen.

$$\mathbf{W} = (\boldsymbol{\mathcal{X}} + \overline{\boldsymbol{\theta}})(\boldsymbol{\mathcal{C}}^T\boldsymbol{\mathcal{C}})^{(-1)}\boldsymbol{\mathcal{C}}^T \qquad \text{für } \{+1,\text{-}1\}\text{-Kodierung} \tag{5.22}$$

$$\mathbf{W} = \boldsymbol{\mathcal{Z}}(\boldsymbol{\mathcal{C}}^T\boldsymbol{\mathcal{C}})^{(-1)}\boldsymbol{\mathcal{C}}^T \qquad \text{für } \{0,1\}\text{-Kodierung} \tag{5.23}$$

Wobei hier wieder $\overline{\boldsymbol{\theta}} = \boldsymbol{\theta} \cdot [1 \cdots 1]$ ist, und $\boldsymbol{\mathcal{Z}} = [z_{i,j}]$ mit $i = 1 \ldots N$ und $j = 1 \ldots L$ mit $z_{i,j} > \theta_i$ ($z_{i,j} < \theta_i$) falls das korrespondierende Element in der Matrix $\boldsymbol{\mathcal{X}}$ gleich 1 (0) ist.

Schreibt man jede Gewichtsmatrix $\mathbf{W}_{(\delta)}$ getrennt an, so ergibt sich:

$$\mathbf{W}_{(\delta)} = (\boldsymbol{\mathcal{X}} + \overline{\boldsymbol{\theta}})(\boldsymbol{\mathcal{C}}^T\boldsymbol{\mathcal{C}})^{(-1)}\boldsymbol{\mathcal{X}}^T_{(\delta)} \tag{5.24}$$

Für die folgenden theoretischen Betrachtungen wird die {+1,-1}-Kodierung verwendet und zusätzlich der Schwellwert θ_i gleich 0 gesetzt.

5.3.2 Zur Speicherkapazität

Da die Spalten von $\boldsymbol{\mathcal{C}}$, das sind die Kontextvektoren, linear unabhängig sein müssen, um die Existenz von $(\boldsymbol{\mathcal{C}}^T\boldsymbol{\mathcal{C}})^{(-1)}$ zu garantieren, ergibt sich, das maximal ND (Anzahl der Zeilen von $\boldsymbol{\mathcal{C}}$) Links gespeichert werden können ($L_c = ND$).

Betrachtet man wieder zufällig generiert Muster so ergibt sich schon wie bei der normalen Projektionsregel in Abschnitt 4.6, daß für halbwegs große N (ca. 50) die Anzahl der linear unabhängigen Spalten in $\boldsymbol{\mathcal{C}}$ beinahe immer L beträgt und nur für Werte von p (Wahrscheinlichkeit, das $\xi_i^\mu = +1$ ist) nahe 1 bzw. 0 davon abweicht (vgl. Abbildung 4.6).

Hat man beliebige Sequenzen gegeben die man speichern möchte, so kann man versuchen durch vergrößern von D die lineare Unabhängigkeit der Spalten von $\boldsymbol{\mathcal{C}}$ zu erreichen. Es kann jedoch sein, daß dies nicht gelingt. Es bleibt einem nun noch der Ausweg durch hinzufügen von hidden Neuronen (Neuronen, deren Impulsfolgen nicht betrachtet werden) und geeigneter Outputmuster dieser Neuronen zu versuchen die linear Abhängigkeit der Kontextvektoren zu verhindern.

5.3.3 Speichern einer beliebigen Sequenz

Ein Beispiel an dem man diese Vorgehensweise sehr leicht demonstrieren kann, möchte ich hier vorstellen.

Man möchte nur einen Zyklus im Netzwerk speichern. Das heißt die Spaltenvektoren der Matrix $\boldsymbol{\mathcal{X}}$ sollen periodisch als Netzwerkzustände aufscheinen.

Man muß nun D solange erhöhen, bis die Kontextmatrix $\boldsymbol{\mathcal{C}}$ mindestens soviele Zeilen (ND) wie Spalten (L) hat. Sind dann die Spalten immer noch nicht linear unabhängig, so kann man D solange erhöhen, bis $D = L$ ist. Eine weitere Erhöhung hat keinen Sinn mehr, da die Zeilen, welche dadurch zu $\boldsymbol{\mathcal{C}}$ hinzugefügt werden, die selben wären, wie die ersten von $\boldsymbol{\mathcal{C}}$ und somit keinen Beitrag mehr liefern können, irgendwelche Spalten linear unabhängig zu machen.

Ein einfaches Beispiel wäre, wenn ein Neuron folgende Sequenz der Periode 6 durchlaufen soll.

$$\ldots\underbrace{0\,0\,0\,0\,1\,1}_{6}\;\underbrace{0\,0\,0\,0\,1\,1}_{6}\ldots$$

Wählt man nun für D das Maximum von $L = 6$, so ergibt sich die folgende Kontextmatrix:

$$\boldsymbol{\mathcal{C}}_0 = \begin{bmatrix}\zeta^1 & \cdots & \zeta^6\end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Die Spalten von $\boldsymbol{\mathcal{C}}$ sind aber noch immer nicht linear unabhängig, da $\zeta^1 - \zeta^2 + \zeta^3 - \zeta^4 + \zeta^5 - \zeta^6$ gleich $\mathbf{0}$ ist.

Fügt man jedoch ein hidden Neuron ein, welches die Sequenz

$$\ldots\underbrace{0\,0\,0\,0\,0\,1}_{6}\;\underbrace{0\,0\,0\,0\,0\,1}_{6}\ldots$$

durchlaufen soll, so wird nach jeder Zeile in der obigen Matrix $\boldsymbol{\mathcal{C}}_0$ eine Zeile eingefügt, welche nur an eine Position eine 1 hat. Diese Position ist für jede Zeile verschieden. Würde man die Zeilen dieser neuen Kontextmatrix $\boldsymbol{\mathcal{C}}_1$ so sortieren, daß jene, die durch das Hinzufügen des hidden Neurons entstanden sind, die ersten 6 sind, so würden diese die Einheitsmatrix bilden und daher ist $\boldsymbol{\mathcal{C}}_1$ pseudoinvertierbar (d.h. $(\boldsymbol{\mathcal{C}}_1^T\boldsymbol{\mathcal{C}}_1)^{(-1)}$ existiert). Man kann also eine Gewichtsmatrix $\mathbf{W}$ nach Gleichung 5.23 berechnen.

Da aber alleine die Zeilen die vom hidden Neuron herrühren die lineare Unabhängigkeit garantieren, ist es egal wieviele Neuronen nun tatsächlich vor dem Einfügen des hidden Neurons bereist vorhanden waren.

Das Resultat, das man aus diesem Beispiel gewinnt, ist in folgendem Satz zusammengefaßt.

Satz 5.3. *Jeder beliebige Zyklus der Länge L kann in einem Netzwerk mit maximal $D = L$ Verbindungen und maximal einem hidden Neuron, mit Hilfe der erweiterten Projektionsregel gespeichert werden.*

5.3.4 Speichern mehrerer beliebiger Sequenzen

Wie nun die Sequenzen von hidden Neuronen aussehen müssen, wenn nicht nur ein Zyklus gespeichert werden soll, sondern mehrere, die nicht gleich lang sind, hängt von diesen Sequenzen selbst ab. Es gibt die Möglichkeit diese Sequenzen einfach per Zufall zu wählen.

Manchmal können jedoch Fälle auftreten, in denen die zu speichernden Sequenzen eine Kontextmatrix ergeben, die unabhängig von den Prototypvektoren ξ^μ immer einen Spaltenrang kleiner L aufweist. Dieses Problem kann dann auftreten, wenn auf relativ wenigen Prototypvektoren sehr viele Links gespeichert werden sollen, d.h L/P ist groß.

Um dieses zu veranschaulichen, sei hier ein kleines Beispiel angegeben. Im folgenden werden nur die Indizes der 4 Prototypvektoren ξ^1 bis ξ^4 geschrieben. Es sollen die folgenden fünf Zyklen gespeichert werden.

$$S^1 = \{4,1\}, \quad S^2 = \{3,4,4\}, \quad S^3 = \{1\}, \quad S^4 = \{3,1\}, \quad S^5 = \{2,3\}$$

Insgesamt sind also $L = 10$ Links zu speichern. Wählt man nun für D den größtmöglichen sinnvollen Wert[3] von 6, so ergibt sich folgende Kontextmatrix (Es sind auch hier nur die Indizes μ der Vektoren ξ^μ angegeben).

$$\widetilde{\boldsymbol{\mathcal{C}}} = \begin{bmatrix} 1 & 4 & 4 & 3 & 4 & 1 & 1 & 3 & 3 & 2 \\ 4 & 1 & 4 & 4 & 3 & 1 & 3 & 1 & 2 & 3 \\ 1 & 4 & 3 & 4 & 4 & 1 & 1 & 3 & 3 & 2 \\ 4 & 1 & 4 & 3 & 4 & 1 & 3 & 1 & 2 & 3 \\ 1 & 4 & 4 & 4 & 3 & 1 & 1 & 3 & 3 & 2 \\ 4 & 1 & 3 & 4 & 4 & 1 & 3 & 1 & 2 & 3 \end{bmatrix}$$

Egal wie groß nun N ist und wie die Prototypvektoren ξ^μ aussehen, kann aufgrund dieser Anordnung der Indizes die tatsächliche Kontextmatrix $\boldsymbol{\mathcal{C}}$ nur 62 verschiedene Zeilen aufweisen (Dieser Wert wurde durch ausprobieren aller möglichen Belegungen einzelner Bits der Vektoren ξ^μ ermittelt.). Von diesen 62 Zeilen, sind jedoch nur 9 linear unabhängig. Das heißt aber auch, daß maximal 9 Spalten linear unabhängig sind und somit $(\boldsymbol{\mathcal{C}}^T\boldsymbol{\mathcal{C}})^{(-1)}$ nicht existiert.

Anhand dieses Beispiels kann man sehen, daß das Hinzufügen von hidden Neuronen für die Projektionsregel nicht unbedingt zum Ziel führen muß.

5.3.5 Speichern gleichlanger Sequenzen

Dieser Fall ist eher für theoretische Betrachtungen interessant. Es ergibt sich hier nämlich eine Gewichtsmatrix $\mathbf{W}$, die jene Bedingungen erfüllt, die in Satz 5.2

---

[3] $D > 6$ ist hinfällig, da dann die weiteren Zeilen von $\boldsymbol{\mathcal{C}}$ nur Wiederholungen der ersten Zeilen sind. 6 ergibt sich als kleinstes gemeinsames Vielfaches der Längen der einzelnen Sequenzen.

vorausgesetzt werden (erweiterte syn. Symmetrie und positive Definitheit von $\mathbf{W}_{(D)}$). Diese Eigenschaften sollen in diesem Abschnitt gezeigt werden.

Es sollen M Zyklen der Länge D gespeichert werden. Das heißt die Anzahl der Verbindungen pro Neuronenpaar wird ident mit der Länge der zu speichernden Zyklen gewählt. Man kann daher für die Matrizen $\boldsymbol{\mathcal{X}}_{(\delta)}$ ($\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}_{(0)}$) und $\boldsymbol{\mathcal{C}}$ schreiben.

$$\boldsymbol{\mathcal{X}}_{(\delta)} = \left[\xi^{1,1} \quad \cdots \quad \xi^{D,1} \quad \xi^{1,2} \quad \cdots \xi^{D,M}\right], \quad \boldsymbol{\mathcal{C}} = \begin{bmatrix} \boldsymbol{\mathcal{X}}_{(1)} \\ \vdots \\ \boldsymbol{\mathcal{X}}_{(D)} \end{bmatrix} \tag{5.25}$$

Damit ergibt sich für eine Gewichtsmatrix $\mathbf{W}_{(\delta)}$

$$\mathbf{W}_{(\delta)} = \boldsymbol{\mathcal{X}}_{(0)} (\boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{C}})^{(-1)} \boldsymbol{\mathcal{X}}_{(\delta)}^T \tag{5.26}$$

Wenn man die Matrizen $\boldsymbol{\mathcal{X}}_{(\delta)}$ betrachtet, so sieht man, daß eine Matrix $\boldsymbol{\mathcal{X}}_{(\kappa)}$ aus der Matrix $\boldsymbol{\mathcal{X}}_{(0)}$ dadurch entsteht, daß die Spalten von $\boldsymbol{\mathcal{X}}_{(0)}$ κ mal nach einem bestimmten Schema vertauscht werden. Eine Spaltenvertauschung kann leicht durch eine rechtseitige Multiplikation mit einer Permutationsmatrix R erreicht werden.

$$\boldsymbol{\mathcal{X}}_{(\kappa)} = \boldsymbol{\mathcal{X}}_{(0)} R^{\kappa} \quad \text{bzw.} \quad \boldsymbol{\mathcal{X}}_{(\kappa)} R^{-\kappa} = \boldsymbol{\mathcal{X}}_{(0)} \tag{5.27}$$

Die $L \times L$ Matrix R besteht in jeder Zeile und Spalten aus genau einem Einselement, die restlichen Einträge sind alle 0. Diese Eigenschaft führt dazu, daß das Produkt RR^T gleich der Einheitsmatrix ist. Daraus ergibt sich der im folgenden wichtige Zusammenhang $R^T = R^{-1}$. Was passiert, wenn man die Matrix R auf $\boldsymbol{\mathcal{C}}$ anwendet? Als erstes wird man meinen, daß die Spalten vertauscht werden. Wenn man sich jedoch die Struktur von $\boldsymbol{\mathcal{C}}$ genauer betrachtet, so wird man erkennen, daß

$$\boldsymbol{\mathcal{C}} R^{-\kappa} = \begin{bmatrix} \boldsymbol{\mathcal{X}}_{(1-\kappa \text{ MOD } D)} \\ \vdots \\ \boldsymbol{\mathcal{X}}_{(D-\kappa \text{ MOD } D)} \end{bmatrix} \tag{5.28}$$

ist und feststellen, das eigentlich nur die Zeilen vertauscht wurden.

Mit diesen Überlegungen, kann man nun zeigen, daß man $\mathbf{W}_{(\delta)}$ auf D verschiedene Arten berechnen kann.

$$\begin{aligned}
\mathbf{W}_{(\delta)} &= \boldsymbol{\mathcal{X}}_{(\kappa)} (\boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{C}})^{(-1)} \boldsymbol{\mathcal{X}}_{(\kappa+\delta \text{ MOD } D)}^T && (5.29) \\
&= \boldsymbol{\mathcal{X}}_{(0)} R^{\kappa} (\boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{C}})^{(-1)} (\boldsymbol{\mathcal{X}}_{(\delta)} R^{\kappa})^T && (5.30) \\
&= \boldsymbol{\mathcal{X}}_{(0)} (R^{-\kappa})^{(-1)} (\boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{C}})^{(-1)} (R^{\kappa})^{(-1)} \boldsymbol{\mathcal{X}}_{(\delta)}^T && (5.31) \\
&= \boldsymbol{\mathcal{X}}_{(0)} (R^{\kappa} (\boldsymbol{\mathcal{C}}^T \boldsymbol{\mathcal{C}}) R^{-\kappa})^{(-1)} \boldsymbol{\mathcal{X}}_{(\delta)}^T && (5.32) \\
&= \boldsymbol{\mathcal{X}}_{(0)} \left((\boldsymbol{\mathcal{C}} R^{-\kappa})^T \boldsymbol{\mathcal{C}} R^{-\kappa} \right)^{(-1)} \boldsymbol{\mathcal{X}}_{(\delta)}^T && (5.33)
\end{aligned}$$

Es wurde hier von den Tatsachen Gebrauch gemacht, daß $\mathcal{C}^T\mathcal{C}$ symmetrisch ist und für zwei quadratische Matrizen die zwei Gleichungen $\mathbf{A}^T\mathbf{B}^T = (\mathbf{BA})^T$ und $\mathbf{A}^{-1}\mathbf{B}^{-1} = (\mathbf{BA})^{-1}$ gelten.

Überlegt man sich noch, daß eine Matrizenmultiplikation der Art $\mathbf{A}^T\mathbf{A}$ invariant gegenüber Zeilenvertauschungen ist, so erhält man aus Gleichung 5.33 wieder Gleichung 5.26 ($\mathcal{C}R^{-\kappa}$ ist eine Zeilenvertauschung). Die Berechnung von $\mathbf{W}_{(\delta)}$ ist also, wie oben behauptet, unabhängig von κ.

Mit dieser Feststellung kann man nun leicht die erweiterte synaptische Symmetrie beweisen.

$$\mathbf{W}_{(D-\delta \text{ MOD } D)} = \mathcal{X}_{(\kappa)}(\mathcal{C}^T\mathcal{C})^{(-1)}\mathcal{X}^T_{(D-\delta+\kappa \text{ MOD } D)} \tag{5.34}$$

$$= \left(\mathcal{X}_{(D-\delta+\kappa \text{ MOD } D)}(\mathcal{C}^T\mathcal{C})^{(-1)}\mathcal{X}^T_{(\kappa)}\right)^T \tag{5.35}$$

Setzt man nun $\kappa = \delta$ so ergib sich:

$$\mathbf{W}_{(D-\delta \text{ MOD } D)} = \left(\mathcal{X}_{(0)}(\mathcal{C}^T\mathcal{C})^{(-1)}\mathcal{X}^T_{(\delta)}\right)^T \tag{5.36}$$

$$\mathbf{W}_{(D-\delta \text{ MOD } D)} = \mathbf{W}^T_{(\delta)} \tag{5.37}$$

Die zweite Voraussetzung in Satz 5.2, ist $\mathbf{W}_{(D)} \geq 0$ ($\mathbf{W}_{(D)}$ ist positiv definit). Um dies zu zeigen, hilft die Beobachtung, daß man aus der κ-Unabhängigkeit von Gleichung 5.29 herleiten kann, daß die Gewichtsmatrix $\widehat{\mathbf{W}}$ des äquivalenten Netzwerkes $\widehat{\mathcal{N}}$ durch den Zusammenhang

$$\widehat{\mathbf{W}} = \mathcal{C}(\mathcal{C}^T\mathcal{C})^{(-1)}\mathcal{C}^T \tag{5.38}$$

gegeben ist. Daraus kann man sich ganz einfach die Idempotentz[4] von $\widehat{\mathbf{W}}$ herleiten ($\widehat{\mathbf{W}}^2 = \mathcal{C}(\mathcal{C}^T\mathcal{C})^{(-1)}\mathcal{C}^T\mathcal{C}(\mathcal{C}^T\mathcal{C})^{(-1)}\mathcal{C}^T = \mathcal{C}(\mathcal{C}^T\mathcal{C})^{(-1)}\mathcal{C}^T = \widehat{\mathbf{W}}$). Überlegt man sich, was dies nun in Bezug auf $\mathbf{W}_{(D)}$ heißt, so ergibt sich

$$\mathbf{W}_{(D)} = \sum_{\delta=1}^{D}\mathbf{W}_{(\delta)}\mathbf{W}_{(D-\delta \text{ MOD } D)} = \sum_{\delta=1}^{D}\mathbf{W}_{(\delta)}\mathbf{W}^T_{(\delta)} = \mathbf{W}\mathbf{W}^T \tag{5.39}$$

Daraus wiederum erhält man, daß $\mathbf{v}^T\mathbf{W}_{(D)}\mathbf{v}$ für jeden Vektor $\mathbf{v}$ größer gleich 0 ist ($\mathbf{v}^T\mathbf{W}_{(D)}\mathbf{v} = \mathbf{v}^T\mathbf{W}\mathbf{W}^T\mathbf{v} = \mathbf{v}^T\mathbf{W}(\mathbf{v}^T\mathbf{W})^T = \|\mathbf{v}^T\mathbf{W}\|^2 \geq 0$). Damit wäre folgender Satz gezeigt.

Satz 5.4. *Speichert man mit Hilfe der erweiterten Projektionsregel nur Zyklen der Länge D in einem Netzwerk, welches Verzögerungszeiten* $\delta \in \{1, \ldots D\}$ *beinhaltet und wird* $\theta_i = 0$ *gesetzt, so erfüllen die resultierenden Gewichtsmatrizen* $\mathbf{W}_{(\delta)}$ *die erweiterte synaptische Symmetrie. Weiters ist* $\mathbf{W}_{(D)}$ *positiv definit.*

---

[4] Eine Matrix $\mathbf{A}$ ist indempotent wenn $\mathbf{A}^2 = \mathbf{A}$.

Eine Bemerkung, die man hier noch hinzufügen sollte, ist, daß man die Diagonalelemente von $\mathbf{W}_{(D)}$ zu 0 setzen kann, ohne die gewünschte Funktion des Netzwerkes zu beeinträchtigen. Diese Maßnahme schließt dann nämlich die triviale Lösung $\mathbf{W}_{(D)} = \mathbf{I}_N$ und $\mathbf{W}_{(\delta)} = \mathbf{0}$ aus ($\mathbf{I}_N \ldots$ $N \times N$ Einheitsmatrix). Ein Netzwerk dieser Art würde ja jeden Zyklus wiederholen, mit dem es initialisiert wird.

Um zu zeigen, daß das Nullsetzen von ${\mathbf{W}_{(D)}}_{i,i}$ keinen Einfluß auf die gespeicherten Zyklen hat, schaut man sich an, welche Konsequenz die Idempotentz von $\widehat{\mathbf{W}}$ auf $\widehat{w}_{i,i}$ hat. Man erhält daraus den Zusammenhang

$$\widehat{w}_{i,i}(1 - \widehat{w}_{i,i}) = \sum_{i=1}^{ND} \sum_{j=1,j\neq i}^{ND} \widehat{w}_{i,j}^2 \tag{5.40}$$

Diese Gleichung impliziert, daß $0 \leq \widehat{w}_{i,i} \leq 1$ ist. Betrachtet man nun die Matrix $\widetilde{W} = \widehat{\mathbf{W}} - \mathrm{diag}(\widehat{w}_{1,1}, \widehat{w}_{2,2}, \ldots, \widehat{w}_{ND,ND})$ so kann man leicht feststellen, daß die Elemente von $\widetilde{W}\zeta^\mu$ immer dasselbe Vorzeichen wie diejenigen von ξ^μ haben. Aufgrund der Definition von $\widehat{\mathbf{W}}$ (vgl. Gleichung 5.6) sieht man, daß die Elemente $\widehat{w}_{i,i}$ die Diagonalelemente von $\mathbf{W}_{(D)}$ sind.

5.4 Die erweiterte Hebb'sche Lernregel

5.4.1 Zum Prinzip

Wie soll man nun das Hebb'sche Postulat, das auf Seite 12 angeführt ist, auf diese Art von formalen Netzwerken übertragen? Betrachten wir dazu eine Verbindung mit der Verzögerungszeit δ. Entsprechend der Formulierung von Hebb, soll das entsprechende Gewicht $w_{i,j}^{(\delta)}$ dann erhöht werden, wenn die Aktivität von j auch zu einer Aktivität von i führt. Aufgrund der Verzögerungszeit, vergehen δ Zeitschritte, bis sich eine eventuelle Aktivität von j zu i ausgebreitet hat. Das heißt, es wäre sinnvoll, das Gewicht $w_{i,j}^{(\delta)}$ entsprechend der Korrelation der Aktivitäten $x_j(k-\delta)$ und $x_i(k)$ zu verändern.

Wendet man nun dieses Prinzip zur Speicherung von Sequenzen an, welche wieder durch die Prototypvektoren ξ^μ und die Kontextabbildung $\mathbf{c}(\mu)$ gegeben sind, so kann man das Gewicht $w_{ij}^{(\delta)}$ folgendermaßen anschreiben.

$$w_{i,j}^{(\delta)} = \frac{1}{N} \sum_{\mu=1}^{P} \sum_{l} \xi_i^\mu \; \xi_j^{c_{\delta,l}(\mu)} \qquad \text{für \{-1,+1\}-Kodierung} \tag{5.41}$$

$$w_{i,j}^{(\delta)} = \frac{1}{N} \sum_{\mu=1}^{P} \sum_{l} (\xi_i^\mu - m)(\xi_j^{c_{\delta,l}(\mu)} - m) \qquad \text{für \{0,1\}-Kodierung} \tag{5.42}$$

Der Index l beschreibt die verschiedenen Kontexte, die es zu einem Prototypvektor geben kann. Die Größe m beschreibt wie schon in Abschnitt 4.5 die mittlere Aktivität des Netzwerks. Mit den auch im vorigen Abschnitt verwendeten Matrizen, kann man dies in einer sehr kompakten Form angeben.

$$\mathbf{W}_{(\delta)} = \frac{1}{N}\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{X}}_{(\delta)}^T \tag{5.43}$$

$$\mathbf{W}_{(\delta)} = \frac{1}{N}(\boldsymbol{\mathcal{X}} - \mathbf{m})(\boldsymbol{\mathcal{X}}_{(\delta)} - \mathbf{m})^T \tag{5.44}$$

Die Matrix $\mathbf{m} = [m_{i,j}]$ ist nur die Matrixform von m, d.h. $m_{i,j} = m$ für alle i und j. Verwendet man die Kontextmatrix, so kann man sofort anschreiben.

$$\mathbf{W} = \frac{1}{N}\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{C}}^T \tag{5.45}$$

$$\mathbf{W} = \frac{1}{N}(\boldsymbol{\mathcal{X}} - \mathbf{m})(\boldsymbol{\mathcal{C}} - \mathbf{m})^T \tag{5.46}$$

Daß durch die so ermittelte Gewichtsmatrix auch tatsächlich die gewünschten Sequenzen gespeichert werden können, kann man sich am leichtesten veranschaulichen, wenn alle Vektoren ξ^μ orthogonal sind. Daraus ergibt sich, daß auch alle Kontextvektoren zueinander orthogonal sind und damit $\boldsymbol{\mathcal{C}}^T\boldsymbol{\mathcal{C}}$ die Einheitsmatrix bildet. Daraus folgt sofort

$$\operatorname{sgn}(\mathbf{W}\boldsymbol{\mathcal{C}} - \boldsymbol{\theta}) = \operatorname{sgn}(\frac{1}{N}\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{C}}^T\boldsymbol{\mathcal{C}} - \boldsymbol{\theta}) = \operatorname{sgn}(\frac{1}{N}\boldsymbol{\mathcal{X}} - \boldsymbol{\theta}) = \boldsymbol{\mathcal{X}} \tag{5.47}$$

falls $\boldsymbol{\theta}$ geeignet gewählt wird (z.B $\boldsymbol{\theta} = \mathbf{0}$).

Ist die Korrelation zwischen den einzelnen Prototypvektoren jedoch nicht 0, sonder größer, so wird dies für die exakte Reproduzierung des gewünschten Zyklus ein Problem. Weiters hat auch die Anzahl der gespeicherten Zyklen einen Einfluß auf die Qualität mit der die Zyklen wiedergegeben werden (vgl. Abschnitt 4.4).

5.4.2 Zur Speicherkapazität

Eine Abschätzung wieviele Links (bzw. Zyklen) abgespeichert werden können, wird mathematisch dann handhabbar, wenn man sich wieder auf den Fall zurückzieht wo alle zu speichernden Zyklen die Länge D haben (vgl. Abschnitt 5.3.5). Es ergibt sich nämlich auch mit der Hebb'schen Regel eine Gewichtsmatrix $\mathbf{W}$, die die Eigenschaft der erweiterten synaptischen Symmetrie besitzt und weiters $\mathbf{W}_{(D)}$ positiv definit ist.

Erweiterte synaptische Symmetrie

Um diese Eigenschaften zu zeigen, geht man so wie in Abschnitt 5.3.5 vor. Als erstes zeigt man wieder, daß die Berechnung von $\mathbf{W}_{(\delta)}$ unabhängig von der Ausgangsmatrix $\boldsymbol{\mathcal{X}}_{(\kappa)}$ ist.

$$\begin{aligned} \mathbf{W}_{(\delta)} &= \frac{1}{N}\boldsymbol{\mathcal{X}}_{(\kappa)}\boldsymbol{\mathcal{X}}^T_{(\delta+\kappa \text{ MOD } D)} = \frac{1}{N}\boldsymbol{\mathcal{X}}_{(0)}R^\kappa(\boldsymbol{\mathcal{X}}_{(\delta)}R^\kappa)^T \\ &= \frac{1}{N}\boldsymbol{\mathcal{X}}_{(0)}R^\kappa(R^\kappa)^T\boldsymbol{\mathcal{X}}^T_{(\delta)} = \frac{1}{N}\boldsymbol{\mathcal{X}}_{(0)}\boldsymbol{\mathcal{X}}^T_{(\delta)} \end{aligned} \tag{5.48}$$

Bezüglich der Matrix R gilt dasselbe, wie bereits in Abschnitt 5.3.5 ausgeführt wurde. Daraus folgt sofort die erweiterte synaptische Symmetrie.

$$\begin{aligned} \mathbf{W}_{(D-\delta)} &= \frac{1}{N}\boldsymbol{\mathcal{X}}_{(\kappa)}\boldsymbol{\mathcal{X}}^T_{(D-\delta+\kappa \text{ MOD } D)} = \frac{1}{N}(\boldsymbol{\mathcal{X}}_{(D-\delta+\kappa \text{ MOD } D)}\boldsymbol{\mathcal{X}}^T_{(\kappa)})^T \\ &= \frac{1}{N}(\boldsymbol{\mathcal{X}}_{(0)}\boldsymbol{\mathcal{X}}^T_{(\delta)})^T = \mathbf{W}^T_{(\delta)} \end{aligned} \tag{5.49}$$

Im Übergang auf die vorletzte Zeile, wurde $\kappa = \delta$ gewählt.

Damit man zur Abschätzung der Speicherkapazität, das äquivalente System heranziehen kann, muß man noch zeigen, daß $\mathbf{W}_{(D)}$ positiv definit ist. Da $\boldsymbol{\mathcal{X}}_{(0)} = \boldsymbol{\mathcal{X}}_{(D)}$ ist, ist $\mathbf{W}_{(D)}$ durch $1/N\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{X}}^T$ gegeben und ist daher positiv definit.

$$\mathbf{v}^T\mathbf{W}_{(D)}\mathbf{v} = \mathbf{v}^T\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{X}}^T\mathbf{v} = \mathbf{v}^T\boldsymbol{\mathcal{X}}(\mathbf{v}^T\boldsymbol{\mathcal{X}})^T = \|\mathbf{v}^T\boldsymbol{\mathcal{X}}\|^2 \geq 0 \tag{5.50}$$

Statistische Mechanik

Die Analyse der Speicherkapazität, kann nun anhand des äquivalenten Systems durchgeführt werden. Da die Gewichtsmatrix $\widehat{\mathbf{W}}$ symmetrisch ist, können die Methoden der statistischen Mechanik angewandt werden.

Für den Fall, daß man die Wahrscheinlichkeit p (Auftreten einer 1 in ξ^μ) 1/2 setzt und zusätzlich $\mathbf{W}_{(D)} = \mathbf{0}$ verlangt, wurde die Analyse in [15] durchgeführt.

Das Ergebnis dieser Analyse ist jenem der Analyse des Standard Hopfieldmodelles sehr ähnlich. Oberhalb einer kritischen Anzahl M_c von gespeicherten Zyklen sinkt die Reproduzierbarkeit drastisch ab. Es hat sich ebenfalls herausgestellt, daß diese Schranke sich linear mit N verhält ($M_c = \alpha_c N$). Es muß hier erwähnt werden, daß die Anzahl L der gespeicherten Links MD beträgt. Damit ist die kritische Anzahl von Links durch $L_c = \alpha_c ND$ gegeben. Da im äquivalenten System auch ND Neuronen vorhanden sind, ist es nicht erstaunlich, daß sich hier für α_c sehr ähnliche Werte wie für das Standardhopfieldmodell ergeben haben. In der folgenden kleinen Tabelle sind einige Werte für α_c in Abhängigkeit von D angegeben.

D	2	3	4	5	∞
α_c	0.1	0.11	0.116	0.12	0.138

Kapitel 6

In Richtung Biologie

Die in den Abschnitten 3 und 4 betrachteten Modellneuronen sind nur eine sehr grobe Annäherung an das biologische Vorbild. In diesem Abschnitt soll nun versucht werden, aus etwas realistischeren Modellneuronen die dort vorgestellten Netzwerke zu konstruieren. In anschließenden Simulationen dieser Netzwerke soll das prinzipielle Verhalten demonstriert werden.

6.1 Das Zellmodell

Morphologie

Das Modellneuron, das verwendet wird, ist ein Kompartmentmodell (vgl. Abschnitt 2.3). Da damit größere Netzwerke simuliert werden sollen, um die aus der Netzwerkstruktur entstehenden Eigenschaften zu untersuchen, wurde ein relativ einfaches Modell gewählt. In Abbildung 6.1 ist das Ersatzschaltbild dieses, aus zwei Kompartments bestehenden Modells, dargestellt. Ein Kompartment stellt den Dentritenbaum dar; das andere das Soma. Die Werte der einzelnen Elemente sind in Tabelle 6.1 angegeben.

Die Modellierung des gesamten Dentritenbaumes durch ein Kompartment wird dadurch motiviert, daß wenn die Verzweigungen des Dentritenbaumes der sogenannten „3/2-power“-Regel folgen, dieser dann als unverzweigter Dendrit einer bestimmten Länge betrachtet werden kann (vgl. [26]).

Generierung des Aktionspotentials

Die Zweige mit den Leitwerten g_{Na} und g_K sind für die Generierung eines Aktionspotentials zuständig. Die Abhängigkeit dieser Leitwerte von der Zeit und der Membranspannung ist dafür ausschlaggebend. Das von Hodgkin und Huxley dafür entwickelte Modell ist in Anhang B erklärt.

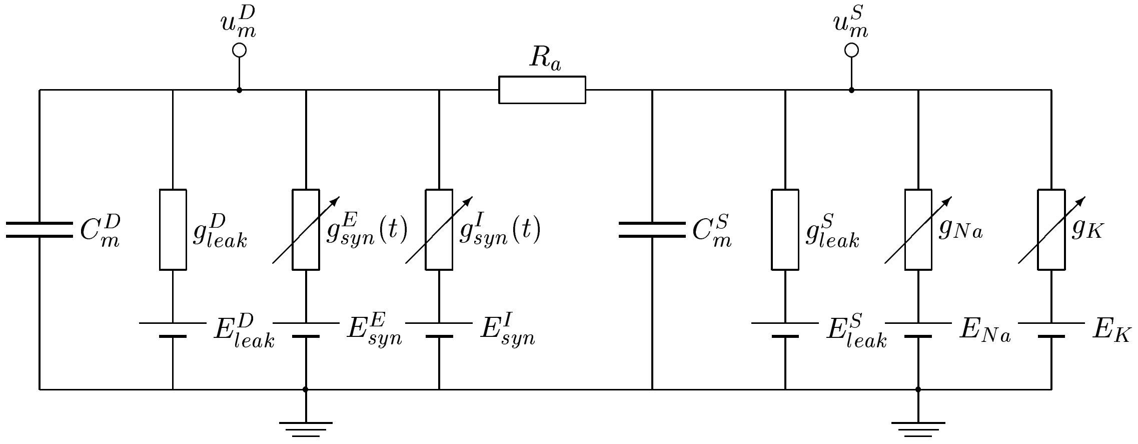


Abbildung 6.1: Ersatzschaltbild des Zellmodelles mit zwei Kompartments. Der linke Teil stellt den Dentritenbaum dar; der rechte das Soma. Für jede Synapse wird ein entsprechender Zweig g_{syn}^E oder g_{syn}^I eingefügt. Der Längswiderstand R_a ergibt sichaus den beiden Widerständen R_a^S und R_a^D: $R_a = 1/2(R_a^D + R_a^S)$.

Dendrit		Soma		Einheit
g_{leak}^D	0.9425	g_{leak}^S	8.482	nS
E_{leak}^D	-70	E_{leak}^S	-49.98	mV
C_m^D	3.142	C_m^S	28.27	pF
R_a^D	4775	R_a^S	12.73	$K\Omega$
l^D	50	l^S	30	μm
d^D	2	d^S	30	μm
A^D	314.2	A^S	2827	μm^2

Tabelle 6.1: In dieser Tabelle sind die Werte der passiven Eigenschaften der Modellzelle angegeben. $g_{leak} = 1/R_{leak}$ berechnet sich über die Fläche aus dem spezifischen Membranwiderstand R_M. (Tabelle 6.2). A gibt die Oberfläche des entsprechenden Kompartments an.

Beschreibung	Zeichen	Wert	Einheit
Umkehr-potentiale	E_{Na}	55	mV
	E_K	-90	mV
	E_{CL}	-65	mV
Spezifische Membran-konstanten	R_M	0.333	Ωm^2
	C_M	0.01	F/m^2
	R_A	0.3	Ωm

Tabelle 6.2: In dieser Tabelle sind die Werte für die Umkehrpotentiale der wichtigsten Ionenarten angegeben, die in diesem einfachen Kompartmentmodell verwendet werden. Die Umkehrpotentiale ergeben sich aus der Nernst'schen Gleichung in die die Ionenkonzentrationen innerhalb und außerhalb der Zelle eingehen. (vgl. Kapitel 2 in [18]). Weiters sind die spezifischen Membrankenngrößen angegeben. Aus diesen berechnen sich zusammen mit der Länge und dem Durchmesser eines Kompartments die Werte in Tabelle 6.1 (vgl. Kapitel 4 in [18]).

Das Modell, das hier verwendet wird, ist eine leichte Abwandlung davon. Es wurden die Werte A_m^α, A_m^β usw. (Werte der Zeile A in Tabelle B.1) doppelt so groß angenommen, um ein etwas schnelleres Aktionspotential und eine kürzere Refraktärzeit zu erzielen (vgl. Anhang C.1).

Synapsen

Die Synapsen werden durch die zeitlich veränderlichen Leitwerte g_{syn}^E und g_{syn}^I modelliert. Der zeitliche Verlauf ist durch die allgemeine Form

$$g_{syn}(t) := A \frac{g_{max}}{\tau_1 - \tau_2} \left(e^{-\frac{t}{\tau_1}} - e^{-\frac{t}{\tau_2}} \right), \quad \text{für} \quad \tau_1 > \tau_2 \tag{6.1}$$

gegeben. A ist ein Faktor, der diese Funktion so normiert, daß sie ein Maximum von g_{max} erreicht. Die zwei Zeitkonstanten τ_1 und τ_2 bestimmen die Anstiegs- bzw. Abfallzeit. Je nach Art der Synapse und des betrachteten Netzwerkes liegen diese Werte bei 2–10 ms. Im allgemeinen sind inhibitorische Synapsen etwas langsamer als die exzitatorischen und haben daher höhere Werte von τ_1 und τ_2.

In diesem Fall wurde für die exzitatorische Synapse $\tau_1 = \tau_2 = 2\,\text{ms}$ gewählt; für die inhibitorischen Synapsen $\tau_1 = \tau_2 = 5\,\text{ms}$ (vgl. Tabelle 9.1 in [22]). Da exzitatorische Synapsen meist einen Einstrom von Na Ionen verursachen, wird für diese Synapsen $E_{syn}^E = E_{Na} = 55\,\text{mV}$ verwendet. Im Gegensatz dazu bewirken inhibitorische Synapsen eine Permeabilität der Membran für K und Cl, deshalb wird für E_{syn}^I ein Wert von $E_K = -90\,\text{mV}$ verwendet.

Wenn im folgenden vom Gewicht einer Synapse gesprochen wird, dann ist damit die Maximale Leitwertänderung g_{max} gemeint. Wird auf die Angabe der Einheit Siemens (kurz S) verzichtet, so bezieht sich der Wert auf 10^{-9} Siemens (nS).

Wie man nun ein solches Zellmodell mit Hilfe von GENESIS erstellt ist in Anhang C.1 abgedruckt. Eine Einführung in die Programmierung von GENESIS ist dem zweiten Teil in *The Book of GENESIS* ([7]) zu entnehmen.

6.2 Charakteristische Eigenschaften

Bevor man nun mit diesem Zellmodell Netzwerke konstruieren kann, welche in den Gewichten der Synapsen und den Delays die Information speichern, sollte man zunächst den quantitativen Einfluß von präsynaptischen Aktionspotentialen ermitteln.

6.2.1 Schwellwert

Als erstes ist es notwendig, zu wissen wie groß man g_{max}^E einer exzitatorischen Synapse machen muß, um mit einem einzigen ankommenden Spike ein Aktionspotential am Soma generieren zu können.

Mit dem in Anhang C.2 abgedruckten Programm wurde dieser Wert G_{th} ermittelt. Weiters wird in diesem Programm auch festgestellt wie groß der Schwellwert U_{th} in Bezug auf die Membranspannung ist (dieser Wert bezieht sich auf die Membranruhespannung von $-70\,\mathrm{mV}$). Es haben sich folgende Werte ergeben.

$$G_{th} = 1.306\,\mathrm{nS} \tag{6.2}$$

$$U_{th} = 16.19\,\mathrm{mV} \tag{6.3}$$

Wenn man wie allgemein angenommen wird, sagt, daß erst ca. 30 gleichzeitige EPSP's ein Aktionspotential erzeugen können, so heißt das für dieses Modell, daß eine Synapse ein g^E_{max} von ca. $43pS$ $(43 \cdot 10^{-12}S)$ aufweist.

Es ist noch anzumerken, daß die durch das HH-Modell (vgl. Anhang B) definierte Generierung eines Aktionspotentials ein sehr scharf ausgeprägtes Schwellwertverhalten aufweist.

6.2.2 Interaktion synchroner Spikes

Für die Implementierung der in den Abschnitten 3 und 4 beschriebenen Netzwerke ist es notwendig zu wissen, wie sich zwei Spikes auswirken, welche gleichzeitig eine exzitatorische und eine inhibitorische Synapse aktivieren. Um dies zu untersuchen, wurde folgendes Simulationsexperiment durchgeführt. Für ein Gewicht g^E_{max}, welches ausreicht, daß bei einem ankommenden Spike ein Aktionspotential ausgelöst wird, wurde jenes Gewicht $\overline{g}^I_{max}(g^E_{max})$ einer inhibitorischen Synapse ermittelt, daß benötigt wird, um zu verhindern, daß die Zelle feuert (vgl. Anhang C.2). Das Ergebnis ist in Abbildung 6.2 dargestellt.

Bei Gewichtspaaren (g^E_{max}, g^I_{max}), welche oberhalb der durchgezogenen Linie liegen, feuert das Neuron bei synchroner Aktivierung der Synapsen nicht. Liegt der Punkt jedoch unterhalb, so reicht das EPSP aus, um trotz des IPSP ein Aktionspotential zu generieren.

Daß der Verlauf dieser Trennlinie sehr linear ist, kann man sich durch folgende Überlegung etwas veranschaulichen. Wären $g^E_{syn}(t)$ und $g^I_{syn}(t)$ Sprungfunktionen, so würde für $t \to \infty$ das Potential u^D_m am Dendriten folgenden asymptotischen Wert annehmen (vgl. Kapitel 6 in [7]).

$$u^D_m(\infty) = \frac{g^E_{max}(E^E_{syn} - U_R) + g^E_{max}(E^I_{syn} - U_R)}{g^D_{leak} + g^E_{max} + g^I_{max}} \tag{6.4}$$

Liegt dieser Wert über dem Schwellwert U_{th} so wird ein Aktionspotential generiert. Aus dieser Bedingung kann man sich nun die Funktion $\tilde{g}^I_{max}(g^E_{max})$ berechnen. Eine einfache Umformung ergibt:

$$\begin{aligned} \tilde{g}^I_{max}(g^E_{max}) &= g^E_{max}\frac{E^E_{syn} - U_R - U_{th}}{U_{th} - (E^I_{syn} - U_R)} - \frac{g^D_{leak}U_{th}}{U_{th} - (E^I_{syn} - U_R)} && (6.5) \\ &= k_{\tilde{g}}g^E_{max} + o_{\tilde{g}} && (6.6) \end{aligned}$$

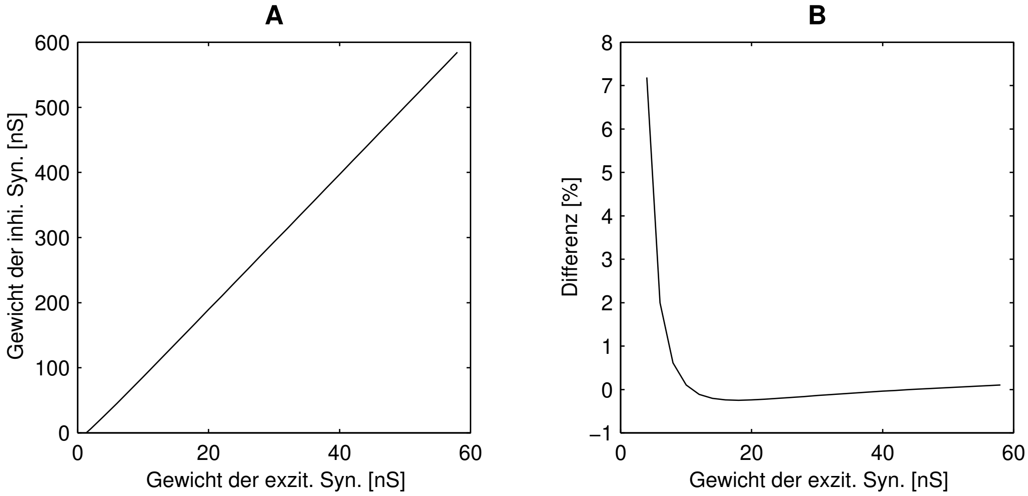


Abbildung 6.2: **A**: Darstellung des Gewichts $\overline{g}^I_{max}(g^E_{max})$ (vgl. Text). Oberhalb der durchgezogenen Linie feuert das Neuron nicht; unterhalb wird ein Aktionspotential generiert. **B**: Darstellung der Abweichung der Funktion $\overline{g}^I_{max}(g^E_{max})$ von einer durch lineare Regression bestimmten Geraden ($10.45 g^E_{syn} - 20.45$).

Man erhält also einen linearen Zusammenhang. Daß die Faktoren $k_{\tilde{g}}$ und $o_{\tilde{g}}$ die man aus dieser Rechnung gewinnt, jedoch stark von denen abweichen, die man aus dem Experiment erhält, kommt daher, daß verschiedene τ_1 und τ_2 für die exzitatorische und die inhibitorische Synapse verwendet wurden.

In Diagramm B in Abbildung 6.2 ist die Abweichung der ermittelten Funktion $\overline{g}^I_{max}(g^E_{max})$ zu einer Geraden dargestellt. Diese Gerade wurde mittels einer linearen Regression bestimmt. Man sieht, daß nur im Bereich von kleinen Werten für g^E_{max} eine etwas größere Abweichung vorhanden ist. Die so ermittelte Gerade besitzt eine Steigung von k_g und einen Offset von o_g, welche folgende Werte annehmen.

$$k_g = 10.45 \tag{6.7}$$

$$o_g = -20.45 \quad \text{in} \quad \text{nS} \tag{6.8}$$

Im folgenden wird sich zeigen, daß dieses Ergebnis gut dazu verwendet werden kann, um die in Abschnitt 4 angegebenen Lernregeln zu implementieren.

6.2.3 Timing des Aktionspotentials

Eine weitere wichtige Frage ist, zu welchem Zeitpunkt das Aktionspotential auftritt, wenn eine exzitatorische und eine inhibitorische Synapse mit verschiedenen

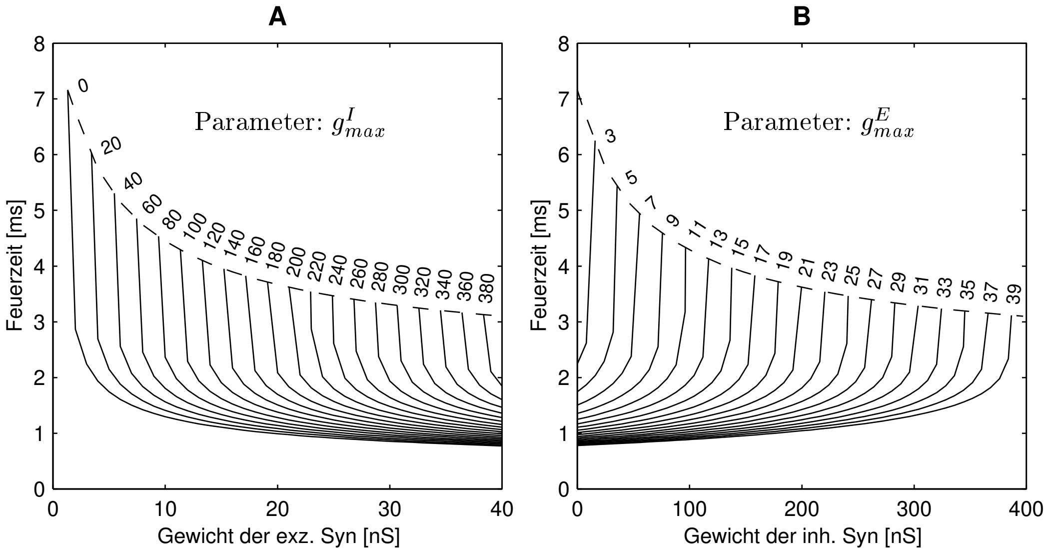


Abbildung 6.3: Darstellung des Zusammenhanges zwischen den Gewichten der inhibitorischen Synapse (g^I_{max}), der exzitatorischen Synapse (g^E_{max}) und dem daraus resultierenden Feuerzeitpunkt t_s des Neurons. Die Feuerzeit t_s bezieht sich auf das Eintreffen der Spikes in den Synapsen. **A**: In diesem Diagramm ist die Abhängigkeit von g^E_{max} dargestellt, wobei als Parameter g^I_{max} angegeben ist. **B**: Hier ist die Abhängigkeit von g^I_{max} mit g^E_{max} als Parameter gezeigt. Die strichlierte Linie stellt die Zeitpunkte t_s entlang der Kurve $\overline{g}^I_{max}(g^E_{max})$ dar (vgl. Abbildung 6.2 A).

Gewichten synchron aktiviert werden. Für Wertepaare (g^E_{max}, g^I_{max}), bei denen ein Aktionspotential erzeugt wird, wurde die Zeit t_s gemessen, die vom Ankommen des Spikes (Zeitpunkt $t = 0$) bis zum Spitzenwert des Aktionspotentials vergeht. Dies wurde für jene Werte durchgeführt, die unterhalb der in Abbildung 6.2 A angegebenen Kurve liegen (vgl. Anhang C.2). Das Ergebnis dieses Experiments ist in Abbildung 6.3 dargestellt.

Man erkennt einen nichtlinearen Zusammenhang. Der qualitative Verlauf ist einfach zu erklären. Je größer g^E_{max} umso schneller feuert das Neuron, je größer g^I_{max} umso mehr wird das Feuern verzögert.

Die strichlierte Linie stellt eine Grenze dar. Oberhalb dieser Linie können keine Werte existieren, da dies Gewichtswerten entsprechen würde, die in Abbildung 6.2 A oberhalb der Trennlinie liegen würden. Die strichlierte Linie zeigt jene Zeitpunkte, die man erhält, wenn die exzitatorische Synapse ein Gewicht von g^E_{max} und die inhibitorische ein Gewicht von $\overline{g}^I_{max}(g^E_{max})$ aufweist.

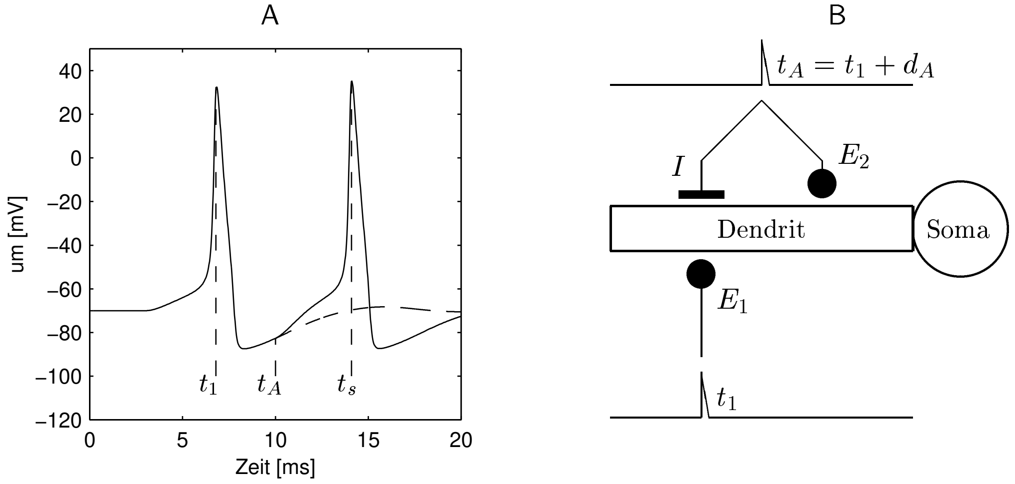


Abbildung 6.4: **A**: Veranschaulichung der im Text verwendeten Zeitpunkte t_1, t_A (erneute Aktivierung der Synapsen) und t_s. **B**: Schematische Darstellung der beteiligten Synapsen und deren Aktivierungszeitpunkte.

6.2.4 Refraktärverhalten

Beschäftigt man sich mit den Zeitpunkten von Aktionspotentialen, so ist es nicht nur interessant, wie groß der Stimulus kurz nach dem Auftreten eines Aktionspotentials sein muß, um die Zelle erneut zum Feuern zu bringen, sondern auch zu welchem Zeitpunkt dies geschieht.

Um diese Frage etwas näher zu erläutern, betrachten wir die in Abbildung 6.4 B dargestellte Versuchsanordnung. Zur Zeit t_1 wird die Synapse E_1 aktiviert. Sie besitzt ein Gewicht von $g_{max}^{E_1}$, welches groß genug ist, die Zelle zum Feuern zu bringen. Eine kurze Zeit $d_A = t_A - t_1$ (vgl. Abbildung 6.4 A) später, werden die Synapsen E_2 und I aktiviert. Die Gewichte $g_{max}^{E_2}$ und g_{max}^{I} dieser Synapsen werden dabei variiert und es wird dann der Zeitpunkt t_s des Auftretens eines eventuellen Aktionspotentials gemessen.

Im ersten Experiment wurde zunächst bestimmt, wie groß $g_{max}^{E_2}$ sein muß, damit überhaupt wieder ein Aktionspotential generiert werden kann. Dies wurde für verschiedene Werte von d_A und $g_{max}^{E_1}$ durchgeführt (vgl. Anhang C.2). Das Ergebnis, ist in Abbildung 6.5 dargestellt.

Man sieht, daß man für kleine d_A hohe Werte für $g_{max}^{E_2}$ benötigt um die Zelle erneut zum Feuern zu bringen. Auf den ersten Blick könnte man verwundert sein, weshalb diese Kurve nicht einfach wieder gegen den vorher ermittelten Wert $G_{th} = 1.306$ abfällt, sondern diesen unterschreitet. Dies kann man sich jedoch erklären, wenn man den Verlauf der Membranspannung am Soma für z.B $g_{max}^{E_1} = 12\,\mathrm{nS}$ betrachtet (strichlierte Kurve in Abbildung 6.4 A). Man sieht, daß ein kleiner Höcker nach dem Aktionspotential auftritt. Dies verursacht nun, daß bei

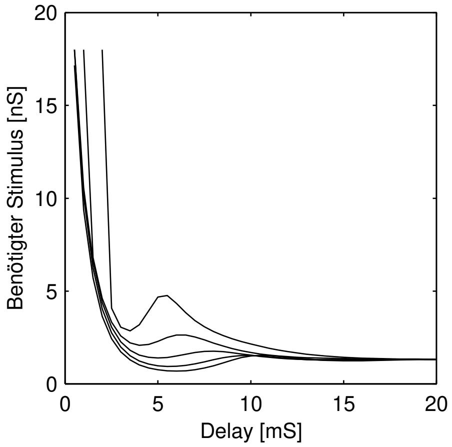


Abbildung 6.5: Quantitative Analyse des Refraktärverhaltens. In diesem Diagramm ist das Gewicht, das benötigt wird um erneut ein Aktionspotential auszulösen, in Abhängigkeit des Zeitraums zwischen dem ersten Aktionspotential und dem Eintreffen des stimulierenden Spikes dargestellt (im Diagramm als Delay bezeichnet). Es sind hier Kurven für verschiedene Werte von g^E_{max} abgebildet (1.5, 3, 6, 12 und 24 nS).

gewissen Werten von d_A (im Bereich vom 3–9 ms) das Neuron leichter erregbar ist. Auffallend ist noch, daß je größer $g^{E_1}_{max}$ wird (der Stimulus der zum ersten Aktionspotential geführt hat), das Minimum der in Abbildung 6.5 dargestellten Kurven immer größer wird. Das heißt, je stärker der erste Stimulus, umso stärker muß auch der zweite sein.

Im zweiten Experiment wurde der Zeitpunkt t_s des zweiten Aktionspotentials genauer untersucht. Dies wurde so durchgeführt, daß für verschiedene Wertepaare von $g^{E_1}_{max}$, d_A und $g^{E_2}_{max}$ der relative Feuerzeitpunkt $t'_s = t_s - t_1 - d_A$ bestimmt wurde (vgl. Anhang C.2). Das Ergebnis ist in Abbildung 6.6 zu sehen.

Bei kleinen Werten von d_A dauert es natürlich am längsten bis das Aktionspotential entsteht. Die Kurven fallen jedoch relativ rasch ab. Das Neuron kann am schnellsten wieder zum Feuern gebracht werden, wenn $g^{E_1}_{max}$ klein ist (Abbildung 6.6 A). Der Effekt, den der Höcker nach einem Aktionspotential (Abbildung 6.4 A) auslöst, der oben bereits beschrieben wurde, tritt hier ebenfalls wieder auf. Dies erkennt man daran, daß im Bereich von kleinen d_A Kurven für Werte von $g^{E_2}_{max}$ (1.25) die kleiner als G_{th} (1.306) sind, existieren (Abbildungen 6.6 A und B). Es ist natürlich klar, daß mit steigendem $g^{E_2}_{max}$ t'_s sinkt.

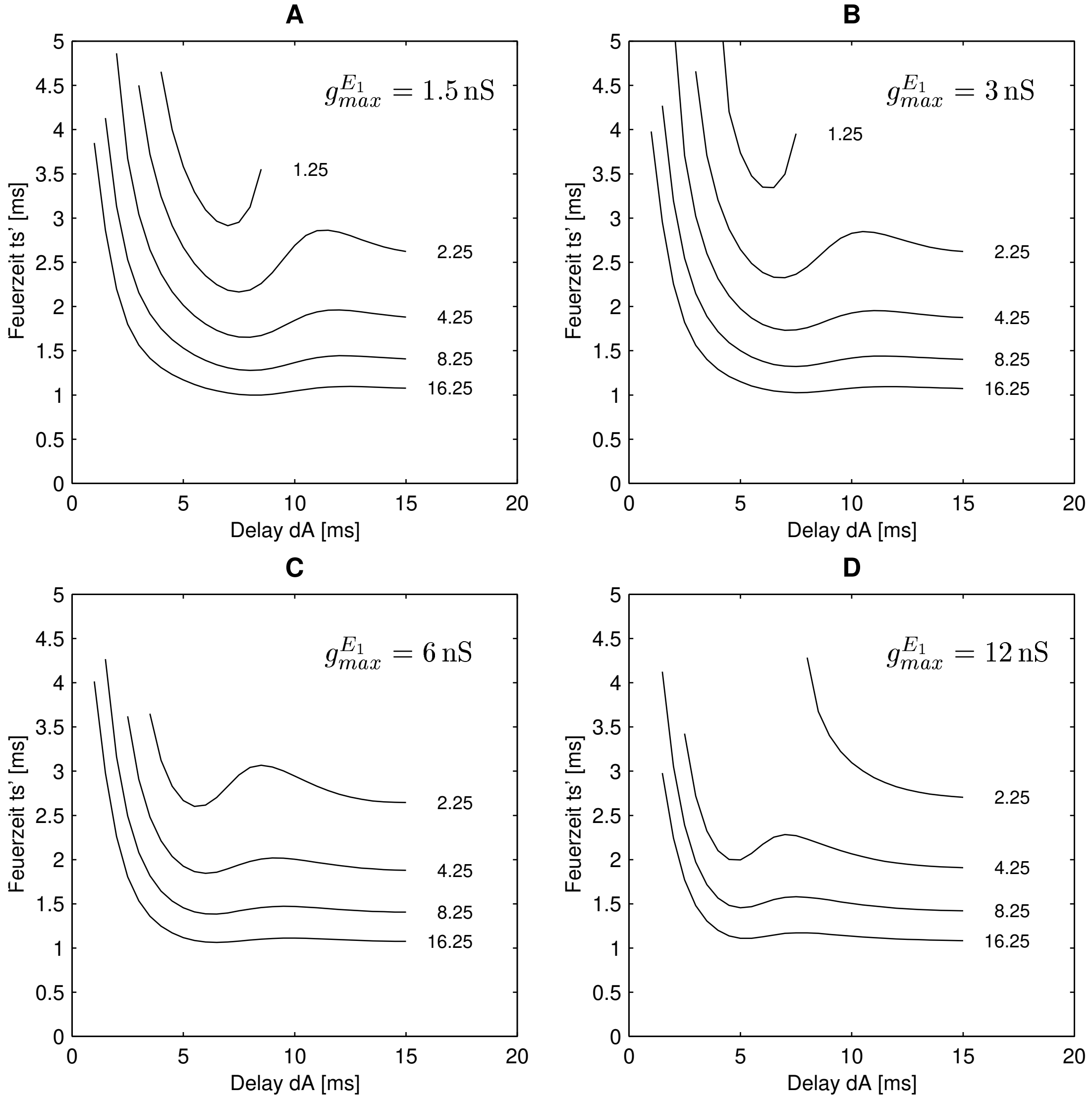


Abbildung 6.6: Zeitpunkt eines Aktionspotentials, nachdem das Neuron, bereits einmal gefeuert hat. Auf der x-Achse ist jeweils das Zeitintervall vom ersten Aktionspotential bis zur Aktivierung der exzitatorischen Synapse aufgetragen (Delay d_A). Auf der y-Achse ist der Feuerzeitpunkt relativ zum Zeitpunkt der Aktivierung der Synapse aufgetragen ($ts' = t_s - d_A - t_A$). Die vier Diagramme zeigen die Verläufe jeweils für einen bestimmten Wert des Gewichts derjenigen Synapse, welche zur Auslösung des ersten Aktionspotential geführt hat ($g_{max}^{E_1}$). In den einzelnen Diagrammen, ist nun der Verlauf Feuerzeitpunkt versus Delay für verschiedene Gewichte $g_{max}^{E_2}$, der als zweites stimulierten Synapse dargestellt.

6.3 Die Projektions Regel

In diesem Abschnitt soll nun das theoretische Modell eines NFN aus Abschnitt 4.6 und das Kompartmentmodell kombiniert werden. Das heißt, es soll ein Netzwerk konstruiert werden, welches aus lauter gleichen Neuronen, des im vorigen Abschnitt vorgestellten Typs, besteht. Die Verzögerungszeiten zwischen den einzelnen Neuronen werden wie im theoretischen Fall für alle Neuronenpaare gleich d gesetzt.

6.3.1 Anwendung auf das Kompartmentmodell

Wie kann man nun diese Berechnungsregel für die Gewichte auf dieses Kompartmentmodell anwenden? Dazu betrachten wir das Ergebnis aus dem Abschnitt über die Interaktion zweier synchroner Spikes (vgl. Seite 72), welches in Abbildung 6.2 dargestellt ist. Dort wird ausgesagt, daß bei synchroner Aktivierung einer exzitatorischen und einer inhibitorischen Synapse mit den Gewichten g^E_{max} und g^I_{max} das Neuron dann feuert (nicht feuert), wenn g^I_{max} kleiner (größer) $\overline{g}^I_{max}$ ist. Vom Standpunkt der Konstruktion eines Netzwerkes kann man fordern, daß die Gewichtspaare (g^e_{max}, g^I_{max}) noch eindeutiger in den Bereichen liegen, wo sie zum Feuern (Nichtfeuern) des Neurons führen. Man kann daher die Bedingung an die Gewichte folgendermaßen formulieren:

$$\begin{aligned} \textit{Neuron feuert} \Leftrightarrow g^I_{max} &< L(g^E_{max}) && (6.9)\\ \textit{Neuron feuert nicht} \Leftrightarrow g^I_{max} &> U(g^E_{max}) && (6.10) \end{aligned}$$

Wobei U (L) eine echt obere (untere) Schranke von $\overline{g}^I_{max}$ ist. Da die Funktion $\overline{g}^I_{max}$ beinahe linear ist, können die Funktionen U und L auch sehr gut als lineare Funktionen dargestellt werden.

$$\begin{aligned} L(g^E_{max}) &:= g^E_{max}\, k_L + o_L && (6.11)\\ U(g^E_{max}) &:= g^E_{max}\, k_U + o_U && (6.12) \end{aligned}$$

Wählt man für $k_L = k_U$ jenen Wert $k_g = 10.45$, welcher sich aus der linearen Regression (vgl. Seite 73) ergibt, so muß $o_L \leq -21.86$ und $o_U \geq -13.65$ sein, damit U (L) eine obere (untere) Schranke von $\overline{g}^I_{max}$ ist.

Gelangen alle Spikes synchron zu den Synapsen $E_1, \ldots, E_n$ und $I_1, \ldots, I_m$ eines Neurons, so hat dies den selben Effekt, wie wenn eine exzitatorische und eine inhibitorische Synapse synchron aktiviert werden würde, deren Gewichte g^E_{max} und g^I_{max} der Summe der Gewichte der einzelnen Synapsen entsprechen (vgl. Abbildung 6.7)

Dies gilt nur dann, wenn der zeitliche Verlauf der Leitwertänderung für alle exzitatorischen und alle inhibitorischen Synapsen untereinander bis auf die Amplitude gleich sind, da dann die Summe der Ströme durch die exzitatorischen

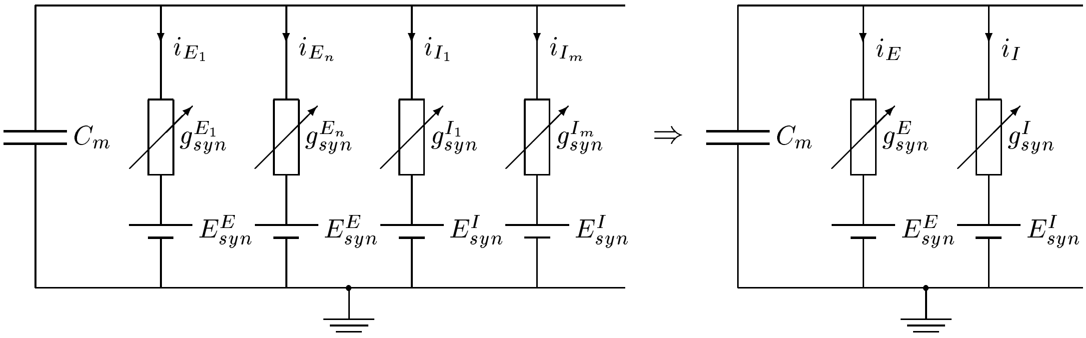


Abbildung 6.7: Reduktion mehrerer exzitatorischer und inhibitorischer Synapsen zu jeweils einer, deren Gewicht der Summe der Einzelgewichte der korrespondierenden Synapsenart entspricht, für den Fall, daß alle Synapsen synchron aktiviert werden. $g^E_{max} = \sum g^{E_j}_{max}$, $g^I_{max} = \sum g^{I_j}_{max}$.

(inhibitorischen) Synapsen $E_1, \ldots, E_n$ $(I_1, \ldots, I_m)$ als ein Strom durch eine einzelne exzitatorische (inhibitorische) Synapse beschrieben werden kann.

$$i_{E_j}(t) = g^{E_j}_{max} g_{syn}(t)(u_m(t) - E^E_{syn}) \tag{6.13}$$

$$\begin{aligned} \sum_{j=1}^{n} i_{E_j}(t) &= g^E_{syn}(t)(u_m(t) - E^E_{syn}) \sum_{j=1}^{n} g^{E_j}_{max} && (6.14) \\ &= g^E_{max} g_{syn}(t)(u_m(t) - E^E_{syn}) && (6.15) \\ &= i_E(t) && (6.16) \end{aligned}$$

$g_{syn}(t)$ ist hier jene Funktion, die die Form eines EPSP's festlegt. Das selbe kann man natürlich auch für die inhibitorischen Synapsen anschreiben.

Damit kann man nun für ein Neuron die Bedingung, daß es aufgrund eines Volleys synchroner Spikes feuert (nicht feuert) auch so formulieren:

$$k_g \sum g^{E_j}_{max} - \sum g^{I_j}_{max} > -o_L \Leftrightarrow \textit{Neuron feuert} \tag{6.17}$$

$$k_g \sum g^{E_j}_{max} - \sum g^{I_j}_{max} < -o_U \Leftrightarrow \textit{Neuron feuert nicht} \tag{6.18}$$

Geht man nun wieder zur Notation über, wie sie in Abschnitt 4.3 eingeführt wurde, so heißt dies, daß ein Neuron zum diskreten Zeitpunkt $k+1$ dann feuert, wenn zur diskreten Zeit k obige Bedingung erfüllt ist. Ist nun $\mathbf{x}(k)$ der Zustandsvektor zum Zeitpunkt k (in $\{0,1\}$-Kodierung), so kann man für Neuron i $x_i(k+1)$ in folgender Weise angeben.

$$x_i(k+1) = \begin{cases} 1 \quad \text{falls} \quad k_g \sum\limits_{\langle j,i\rangle \in E^E} g_{i,j} x_i(k) - \sum\limits_{\langle j,i\rangle \in E^I} g_{i,j} x_i(k) + o_L > 0 \\ 0 \quad \text{falls} \quad k_g \sum\limits_{\langle j,i\rangle \in E^E} g_{i,j} x_i(k) - \sum\limits_{\langle j,i\rangle \in E^I} g_{i,j} x_i(k) + o_U < 0 \end{cases} \tag{6.19}$$

Dabei stellt E^E (E^I) die Menge der exzitatorischen (inhibitorischen) Synapsen dar. $g_{i,j}$ entspricht g_{max} jener Synapse, welche die beiden Neuronen j und i verbindet. Mit der Konvention

$$w_{i,j} = k_g g_{i,j} \quad \text{falls} \quad \langle j,i\rangle \in E^E \tag{6.20}$$

$$w_{i,j} = -g_{i,j} \quad \text{falls} \quad \langle j,i\rangle \in E^I \tag{6.21}$$

erhält man nun eine sehr ähnliche Schreibweise, wie zu Beginn von Abschnitt 4.6.

$$x_i(k+1) = \begin{cases} 1 & \text{falls} \quad \sum_{j=1}^{N} w_{i,j} x_j(k) > -o_L \\ 0 & \text{falls} \quad \sum_{j=1}^{N} w_{i,j} x_j(k) < -o_U \end{cases} \tag{6.22}$$

Zur Berechnung der Gewichte $w_{i,j}$ nach der Projektionsregel, kann man nun die Ungleichungen durch Gleichungen ersetzen (vgl. Gleichung 4.41).

6.3.2 Speichern von Links

Hat man eine Menge von gewünschten Zustandsänderungen durch die Matrix $\boldsymbol{\mathcal{X}} = [\xi^1, \ldots, \xi^L]$ und die Funktion σ gegeben, so kann man nun die Bedingungen an die Gewichte $w_{i,j}$ als Menge von Gleichungen anschreiben.

$$\sum_{j=1}^{N} w_{i,j} \xi_j^{\mu} = \tilde{\xi}_j^{\sigma(\mu)} := \begin{cases} -o_L & \text{falls} \quad \xi_j^{\sigma(\mu)} = 1 \\ -o_U & \text{falls} \quad \xi_j^{\sigma(\mu)} = 0 \end{cases} \tag{6.23}$$

Oder in Matrixschreibweise ($\widetilde{\boldsymbol{\mathcal{X}}}^{\sigma} = [\tilde{\xi}^{\sigma(1)}, \ldots, \tilde{\xi}^{\sigma(L)}]$):

$$\mathbf{W}\boldsymbol{\mathcal{X}} = \widetilde{\boldsymbol{\mathcal{X}}}^{\sigma} \tag{6.24}$$

Damit ergibt sich folgende Gewichtsmatrix $\mathbf{W}$:

$$\mathbf{W} = \widetilde{\boldsymbol{\mathcal{X}}}^{\sigma} (\boldsymbol{\mathcal{X}}^T \boldsymbol{\mathcal{X}})^{-1} \boldsymbol{\mathcal{X}} \tag{6.25}$$

Mit Hilfe von Gleichung 6.20 kann man nun sehr leicht bestimmen, ob $\langle j,i\rangle$ eine exzitatorische oder eine inhibitorische Synapse ist und welcher Wert sich für g_{max} sich. Falls $w_{i,j}$ kleiner als 0 ist, so ist die Synapse $\langle j,i\rangle$ inhibitorisch mit $g_{i,j} = -w_{i,j}$; andernfalls handelt es sich um eine exzitatorische Synapse mit $g_{i,j} = w_{i,j}/k_g$.

In dieser Konstruktion wird stillschweigend vorausgesetzt, daß unabhängig von der Gewichtskonstellation der Synapsen eines Neurons der Zeitraum zwischen dem Eintreffen der synchronen Spikes und dem neuerlichen Feuern des Neurons bei allen Neuronen im Netzwerk gleich groß ist. Dies ist notwendig, um zu erreichen, daß alle Synapsen eines Neurons wieder gleichzeitig aktiviert werden, sodaß man wieder die Bedingung laut Gleichung 6.17 anwenden kann. Wie man aber im

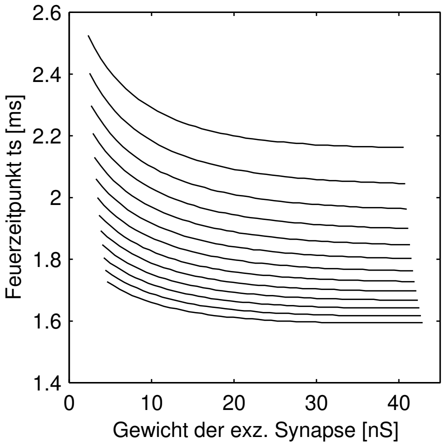


Abbildung 6.8: Darstellung der Zeit t_s, die von der synchronen Aktivierung der Synapsen bis zur Entstehung des Aktionspotentials vergeht für Gewichtswerte von g_{max}^E und g_{max}^I die auf der Geraden $g_{max}^I = k_g g_{max}^E + o_L$ liegen. Die verschiedenen Kurven entstehen durch Variation von o_L (oberste Kurve $-o_L = 22\,\mathrm{nS}$ bis zur untersten Kurve $-o_L = 50\,\mathrm{nS}$).

Abschnitt *Timing des Aktionspotentials* (Seite 73) sehen kann, hängt diese Zeit sehr wohl von den Gewichten der Synapse ab. Man kann jedoch durch geeignete Wahl von o_L und o_U erreichen, daß diese Zeit nur sehr wenig in Abhängigkeit der Gewichte schwankt. In Abbildung 6.8 ist der Zeitraum t_s für Gewichtspaare $g_{max}^I = k_g g_{max}^E + o_L$ für verschiedene Werte von o_L aufgetragen. Diese Gewichtspaare sind ja jene, welche auftreten, wenn die Gewichte nach Gleichung 6.25 berechnet werden.

Man sieht, daß mit steigendem Betrag von o_L (Kurven weiter unten) die Abhängigkeit vom Gewicht immer kleiner wird.

6.3.3 Eine erste Simulation

Um einen Eindruck zu geben, wie nun ein derart konstruierter CPG arbeitet, sei hier ein kleines Beispiel angeführt.

Um möglichst sicher zu gehen, daß die Refraktärzeit schon vorbei ist, wenn ein Neuron wieder aktiviert wird, wählt man am einfachsten $d_{i,j} = d$ relativ groß. Betrachtet man Abbildung 6.5, so scheint ein Wert von 12 ms als plausibel. Für o_L und o_U werden die oben angeführten Werte (Ergebnisse des Experiments, welches im Abschnitt über die Interaktion synchroner Spikes (Seite 72) durchgeführt wurde.) von $o_L = -21.86$ und $o_U = -13.65$ verwendet. Um die Darstellung übersichtlich zu halten, wurden $N = 10$ Neuronen gewählt, welche einen Zyklus von $L = 8$ Mustern periodisch wiederholen sollen. Die Muster $\mathcal{X}$ wurden rein zufällig generiert.

Die Berechnung der Gewichtsmatrix $\mathbf{W}$ bzw. der Leitwertmatrix $\mathbf{G} = [g_{i,j}]$ $i,j \in \{1,\ldots,N\}$ erfolgt mittels MATLAB®. Das Ergebnis, wird dann über eine Datei bestimmten Formats an den Simulator GENESIS weitergegeben, wo dann das Netzwerkmodell erstellt und simuliert wird (vgl. Anhang C.3).

Um überhaupt Aktivität im Netzwerk zu erhalten, muß man einen gewissen Stimulus bieten. Dies wird in den Simulationen so gemacht, daß jene Neuronen, welche in einem beliebigen der L Muster aktiv sein sollen, einen Spike erhalten, welcher sie zum Feuern bringt. Danach erhält das Netzwerk keinerlei Input mehr.

In Abbildung 6.9 ist ein Beispiel angegeben. In Abbildung 6.9 A ist die zugrundeliegende Matrix $\boldsymbol{\mathcal{X}}$ dargestellt. Diese 8 Muster sollen zyklisch durchlaufen werden. In Abbildung 6.9 B ist die resultierende Leitwertmatrix dargestellt. Die Fläche eines Quadrates stellt den Betrag von $g_{i,j}$ dar, während die Farbe angibt, ob es sich um eine exzitatorische (hell) oder inhibitorische (dunkel) Synapse handelt. Zur Größenordnung ist zu sagen, daß die stärkste exzitatorische Synapse ein g_{max} von 1.959 nS aufweist; das Maximum bei den inhibitorischen Synapsen beträgt 9.7 nS.

In Abbildung 6.9 C ist dann das Ergebnis einer Simulation, die sich über eine halbe Sekunde erstreckt zu sehen. Man sieht, daß die Neuronen sehr schön sychron feuern, und exakt jenes Muster wiedergeben, welches durch die Matrix $\boldsymbol{\mathcal{X}}$ (Abbildung 6.9 A) definiert ist. Der Abstand zwischen den diskreten Zeitpunkten ist mit 14.85 ms größer als der Delay von 12 ms, da es eine gewisse Zeit dauert bis das Aktionspotential entsteht. Diese Differenz von 2.85 ms entspricht ziemlich genau dem Wert, den man aus Abbildung 6.8 ablesen kann.

6.4 Die Hebb'sche Lernregel

In diesem Abschnitt soll ein kleines Beispiel eines CPG gezeigt werden, der mit Hilfe der in Abschnitt 4.5 vorgestellten Hebb'schen Lernregel konstruiert wurde. Die einzige Überlegung, die man hier anstellen muß, ist die Skalierung der Gewichte. Bei der theoretischen Betrachtung wurde aufgrund der mittleren Aktivität in einem Muster ein beinah optimaler Schwellwert θ_i festgelegt (vgl. Gleichung A.24). Es ist aber so, daß der Schwellwert bei dem hier betrachteten Zellmodell fix ist, und als maximale Leitwertänderung ausgedrückt, den Wert $G_{th} = 1.306$ annimmt. Man kann jenen Faktor, der θ_i auf G_{th} skaliert verwenden, um die nach der Hebb'schen Regel berechneten Gewichte

$$\tilde{w}_{i,j} = \frac{1}{N} \sum_{j=1}^{N} (\xi_i^{\sigma(\mu)} - m)(\xi_j^{\mu} - m) \tag{6.26}$$

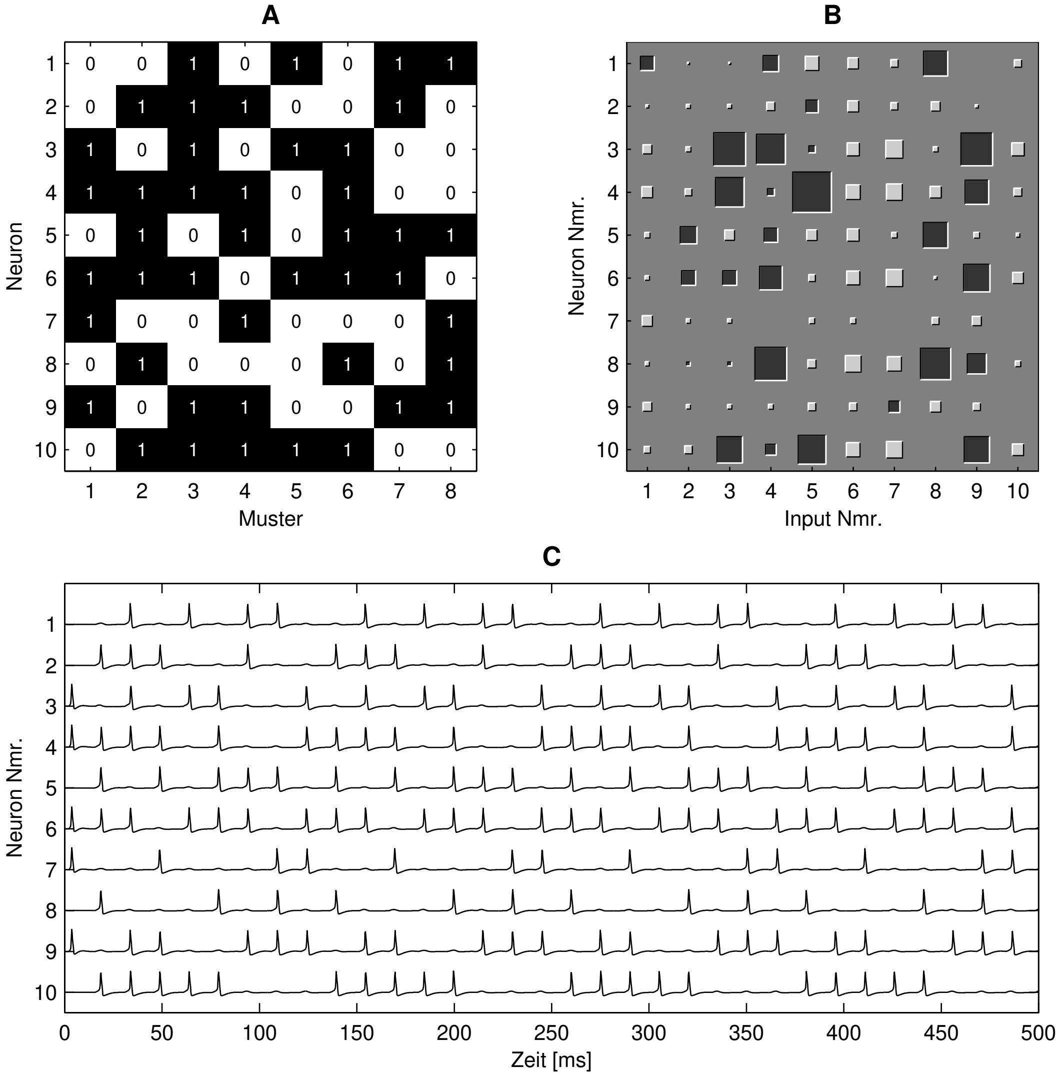


Abbildung 6.9: Beispiel für einen CPG der nach der Projektionsregel konstruiert ist. **A**: Die zugrundeliegende Matrix des zu speichernden Zyklus. **B**: Die aus Gleichung 6.25 resultierende Leitwertmatrix (Hinton-Diagramm). Die Fläche der Quadrate gibt g_{max} der Synapse an. Die Färbung die Art der Synapse. Hell entspricht exzitatorisch; dunkel inhibitorisch. Das Maximum der exzitatorischen Synapsen beträgt 1.959 nS; das der inhibitorischen 9.7 nS. **C**: Hier ist der Verlauf der Membranspannung der 10 beteiligten Neuronen dargestellt. Die ersten Spikes der Neuronen 3, 4, 6, 7 und 9 wurden durch einen externen Stimulus erzeugt, danach erhält das Netzwerk keinerlei Input mehr. Das Ergebnis stimmt mit den Erwartungen überein.

so zu skalieren, daß sie für den gegebenen Schwellwert G_{th} optimal sind.

$$w_{i,j} = \frac{G_{th}}{\theta_i} \tilde{w}_{i,j} \tag{6.27}$$

Diese Gewichte kann man wieder verwenden, um die Art der Synapse und die maximale Leitwertänderung g_{max} zu bestimmen (vgl. Gleichung 6.20).

6.4.1 Ein kleines Beispiel

Wie beim vorigen Beispiel zur Projektionsregel wird auch hier der Delay wieder relativ groß gewählt ($d_{i,j} = d = 12\,\mathrm{ms}$). Um zu demonstrieren, daß man sehr leicht auch mehrere Zyklen in einem Netzwerk speichern kann, wurden auf vier zufälligen Muster zwei Zyklen abgebildet. Das heißt, daß Muster 1 (3) und 2 (4) sich ständig wiederholen sollen, wenn das Netzwerk mit einem dieser beiden angestoßen wird ($\sigma(1) = 2$, $\sigma(2) = 1$, $\sigma(3) = 4$, $\sigma(4) = 3$).

In Abbildung 6.10 ist dieses Beispiel graphisch veranschaulicht. In der ersten Hälfte der Simulation, wird derZyklus 1,2,1,2, … durchlaufen. Dann wird mittels eines starken inhibitorischen Inputs jegliche Aktivität im Netzwerk ausgelöscht. Zum Zeitpunkt $t = 250\,\mathrm{ms}$ wird dann mit Muster 3 stimuliert und es wird wie erwartet der Zyklus 3,4,3,4, … durchlaufen.

6.5 Die erweiterte Projektionsregel

Nachdem in den vorhergehenden Abschnitten erläutert wurde, wie man die formalen Netzwerke aus den Abschnitten 4.5 und 4.6 auf das Kompartmentmodell übertragen kann, soll anhand eines weiteren Beispiels gezeigt werden, daß sich auch die Modelle mit mehr als einer Verbindung pro Neuronenpaar in ein entsprechendes Netzwerk im Kompartmentmodell umwandeln lassen.

6.5.1 Die Leitwertmatrix

Das Prinzip, wie man die Projektionsregel auf das hier verwendete Kompartmentmodell anwendet, ist bereits in Abschnitt 6.3 genauer ausgeführt worden. Überträgt man diese Erkenntnisse, so kann man hier die Gewichtsmatrix folgendermaßen angeben:

$$\mathbf{W} = \tilde{\boldsymbol{\mathcal{X}}}(\boldsymbol{\mathcal{C}}^T\boldsymbol{\mathcal{C}})^{(-1)}\boldsymbol{\mathcal{C}}^T \tag{6.28}$$

(bzgl. $\tilde{\boldsymbol{\mathcal{X}}}$ vgl. Gleichung 6.24). Dann kann man wieder Gleichung 6.20 verwenden, um die Leitwertmatrix $\mathbf{G}$ zu berechnen.

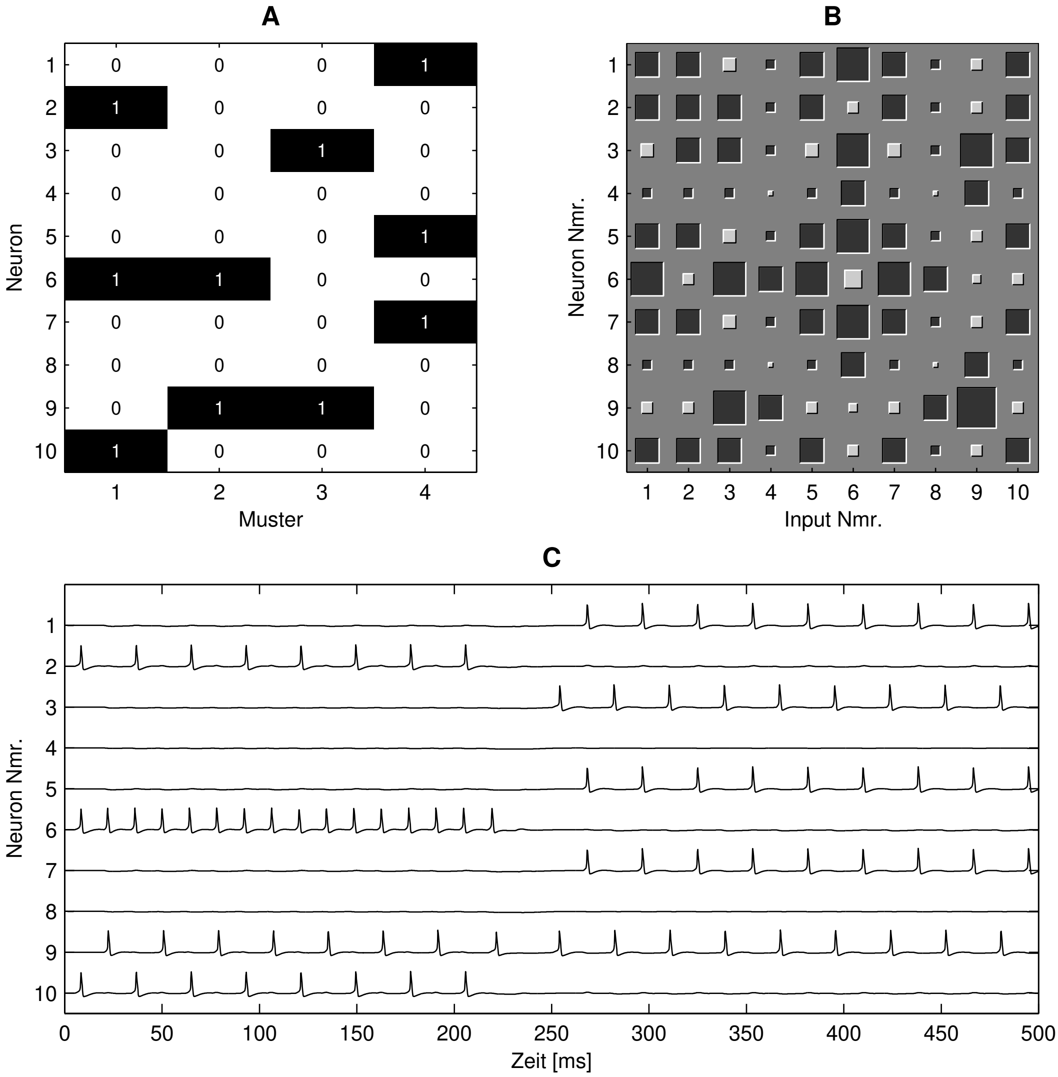


Abbildung 6.10: Beispiel für einen CPG, der nach der Hebb'schen Regel konstruiert ist. **A**: Die zugrundeliegende Matrix der zu speichernden Zyklen. Es sollen die Muster 1 und 2 bzw. 3 und 4 periodisch wiederholt werden, je nach dem wie das Netzwerk initialisiert wird. **B**: Die aus Gleichung 4.36 resultierende Leitwertmatrix. Das Maximum der exzitatorischen Synapsen beträgt 3.7 nS; das der inhibitorischen 18.13 nS. **C**: Hier ist der Verlauf der Membranspannung der 10 beteiligten Neuronen dargestellt. Bis zum Zeitpunkt $t = 220\,\mathrm{ms}$ werden die Muster 1 und 2 periodisch wiederholt, nach dem das Muster 1 zum Initialisieren verwendet wurde. Dann wurde über inhibitorische Synapsen diese Muster ausgelöscht und durch anlegen des Musters 3 eine periodische Wiederholung von Muster 3 und 4 bewirkt.

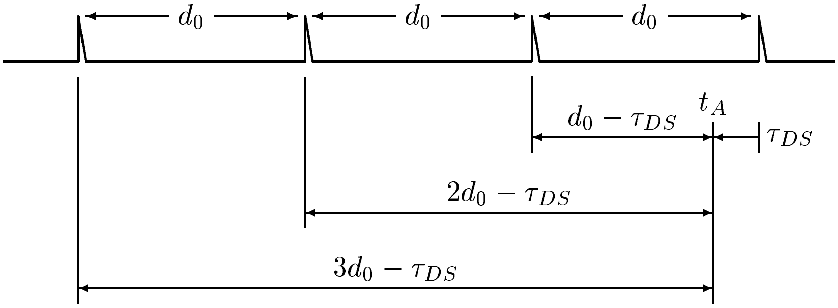


Abbildung 6.11: Darstellung des korrekten Timings von Spikes und den daraus resultierenden Verzögerungszeiten (vgl. Text).

6.5.2 Die Verzögerungszeiten

Damit das Netzwerk wie gewünscht zu den Zeiten kd_0 synchron feuert, muß man die Delays richtig wählen. In der formalen Betrachtung wurde nämlich jene Zeit τ_{DS} vernachlässigt, die benötigt wird, bis aus einem ankommenden Volley von Spikes sich ein Aktionspotential ausbildet. Würde man die Delays wählen, wie es zu Beginn von Abschnitt 5.1.1 angegeben ist, so würden die Spikes nicht mehr alle synchron ankommen.

Wie das Timing der Spikes und des Zeitpunkts t_A, zu dem ein Neuron den Input bekommen muß, korrekt aussieht, ist in Abbildung 6.11 zu erkennen. In dieser dieser Abbildung kann man leicht sehen, daß die kürzeste Verzögerungszeit den Wert $d_0 - \tau_{DS}$ annehmen soll. Die weiteren Verzögerungszeiten d_δ ergeben sich dann zu $d_0 - \tau_{DS} + \delta d_0$.

Wie groß nun τ_{DS} ist, hängt wie Abschnitt 6.3 (vgl. Abbildung 6.8 erläutert von o_L ab. Bei einem Wert von $o_L = -23$ hat sich gezeigt, daß eine Wahl von $\tau_{DS} = 3\,\text{ms}$ in diesem Modell zu guten Ergebnissen führt.

6.5.3 Simulationsergebnisse

Es soll hier ein Beispiel eines CPG gezeigt werden, welches klarstellt, daß man mit dieser Art der Gewichtsbestimmung die gewünschten Zyklen ziemlich exakt speichern kann.

Es wird ein Netzwerk von $N = 20$ Neuronen betrachtet. Dieses Netzwerk soll $M = 3$ Zyklen der Periodendauer $T = d_0\overline{T} = 15 \cdot 10 = 150\,\text{ms}$ oder anders ausgedrückt $L = M\overline{T} = 30$ Links speichern. Man sieht, da $L > N$ ist, daß dies mit einem Netzwerk mit nur einer Verbindung pro Neuronenpaar nicht mehr möglich wäre. Die zu speichernden Zyklen selbst wurden zufällig generiert. Sie werden wie Abschnitt 5.3 erklärt durch die Matrix $\boldsymbol{\mathcal{X}}$ repräsentiert. Die Wahrscheinlichkeit p des Auftretens einer 1 in $\boldsymbol{\mathcal{X}}$ wurde mit 0.3 angenommen. Wie schon bei den bisherigen Simulationen wurde auch hier $d_0 = 15\,\text{ms}$ gewählt. Daraus ergeben sich die folgenden Verzögerungszeiten: $d_1 = d_0 - \tau_{DS} = 12\,\mathrm{ms}$, $d_2 = 27\,\mathrm{ms}$, $d_3 = 42\,\mathrm{ms}$ und $d_4 = 57\,\mathrm{ms}$. Es wurden 4 Verbindungen pro Neuronenpaar gewählt.

Das Ergebnis der Simulation dieses Netzwerkes ist in Abbildung 6.12 dargestellt. Die Diagramme geben die Zeitpunkte eines Aktionspotentials an (+ und • Markierungen). Um zu sehen, ob die 3 Zyklen auch reproduziert werden können, wurde für eine Zeit von ca. 80 ms dem Netzwerk ein Stimulus geboten, der mit einem der 3 gespeicherten Zyklen konsistent ist (daher 3 Diagramme). Die + Markierungen kennzeichen jene Spikes, die Aufgrund dieses externen Stimulus entstanden sind. Danach arbeitet das Netzwerk frei von jeglichen externen Signalen. Die • Markierungen zeigen somit jene Aktionspotentiale, die das Netzwerk von selbst generiert hat.

Um zu visualisieren, daß diese Impulsfolgen auch etwas mit den gewünschten Zyklen zu tun haben, ist der erwartete Zyklus mit Hilfe von ○ Markierungen eingezeichnet. Es ist sehr schön zu sehen, wie zu jedem gewünschten Spike auch tatsächlich ein vom Netzwerk generierter Impuls auftritt.

Erfahrungen aus den Simulationen

An dieser Stelle sollen einige Erfahrungen genannt werden, die sich aus den Simulationen CPG's solcher Art, ergeben haben.

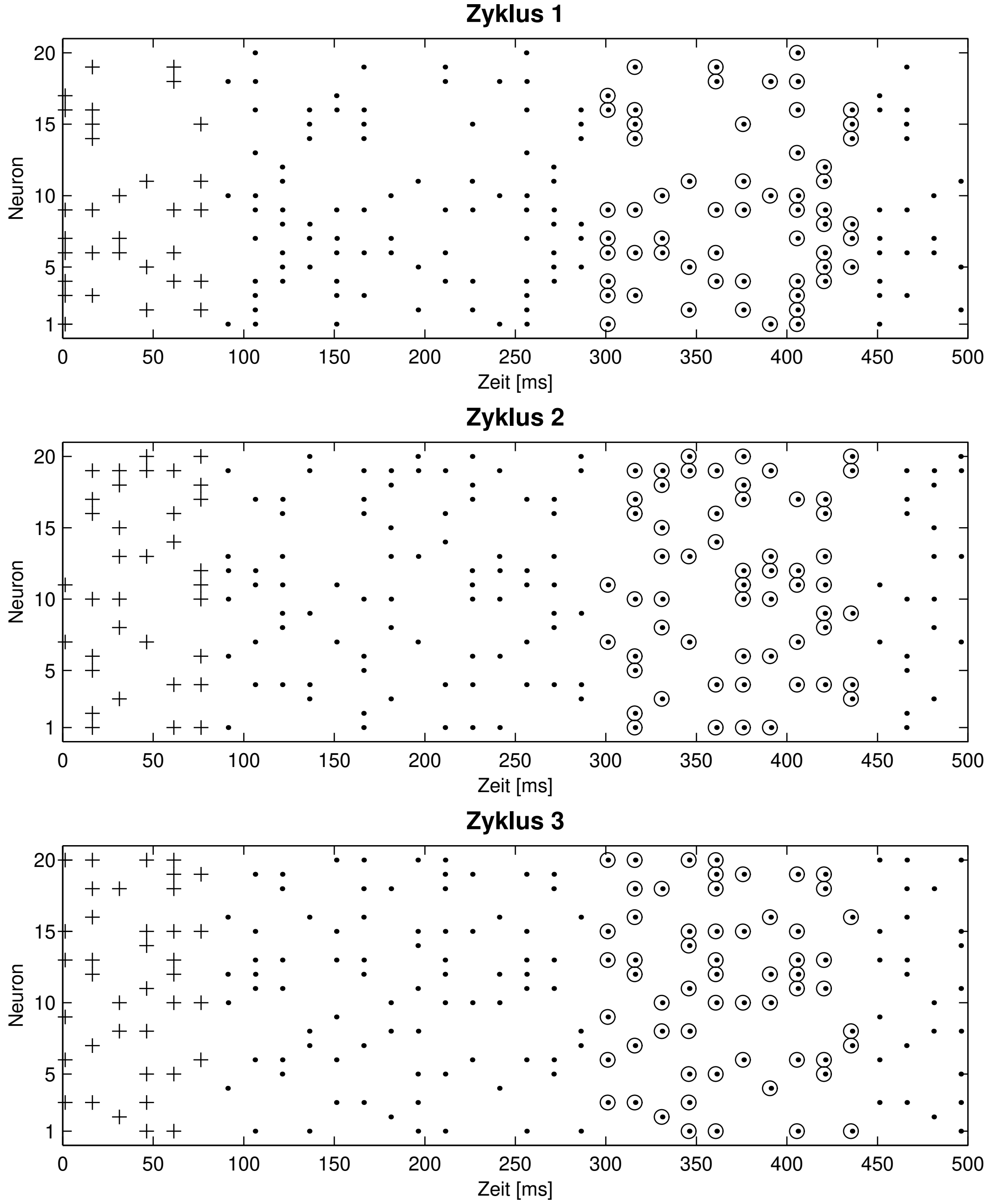
Es hat sich gezeigt, daß man die theoretische Speicherkapazität von $L_c = ND$ sehr gut ausschöpfen kann. Dies ist jedoch nur dann möglich, wenn p nicht zu groß wird. Durch die Simulationen hat sich ergeben, daß eine obere Grenze von ca. 0.6-0.7 nicht überschritten werden sollte. Wenn dies der Fall ist, hat sich meist folgendes Ergebnis eingestellt: Einige Neuronen haben begonnen ständig zu Feuern; andere hingegen haben ganz aufgehört.

Die Frage ob dieser Zustand eintritt oder nicht, ist sehr eng damit verbunden wie stark die Summe der exzitatorischen Inputs eines Neurons streuen. Präzise formuliert heißt das folgendes: Man betrachtet für alle Neuronen und deren gewünschten Spikes folgende Größe (für alle i und μ für die $\xi_i^\mu = 1$ ist).

$$G_{i,\mu} = \sum_{\delta=1}^{D} \sum_{j=1}^{N} g_{i,j}^{(\delta)}\, \zeta_j^\mu\, \mathbf{1}_{\langle j,i \rangle \textit{ ist exz.}} \tag{6.29}$$

(ζ^μ ... Kontextvektor, $g_{i,j}^{(\delta)}$ ... maximale Leitwertänderung der Synapse die die Verbindung δ repräsentiert) Es hat sich gezeigt, daß, wenn diese Größen zu stark streuen, das Netzwerk die gewünschten Sequenzen nicht reproduziert. Dieses Argument wurde bereits auf Seite 80 angesprochen.

Aus den Simulationen hat sich ergeben, daß es für eine gute Wiedergabe der gewünschten Zyklen empfehlenswert ist, wenn Minimal- bzw. Maximalwerte von 1.5 nS bzw. 4.5 nS eingehalten werden.



Zyklus 1

Zyklus 2

Zyklus 3

Abbildung 6.12: Impulsfolgen eines CPG, der nach der erweiterten Projektionsregel konstruiert wurde. Die 3 Diagramme zeigen den zeitlichen Verlauf, je nach dem zu welchem der 3 gespeicherten Zyklen, der Stimulus (gekennzeichnet durch die + Markierungen) konsistent ist. Die ○ Markierungen kennzeichnen eine Periode des gewünschten Zyklus. Man sieht, daß dieser sehr exakt mit den tatsächlich generierten Impulsfolgen übereinstimmt (• Markierungen).

Bewegt man sich mit der Anzahl L der zu speichernden Links auf L_c zu, so ist es bei zu großem p sehr leicht möglich, daß die Obergrenze überschritten wird.

6.6 Die erweiterte Hebb'sche Regel

Hier soll wieder anhand einer Simulation gezeigt werden, daß auch die erweiterte Hebb'sche Lernregel geeignet ist, diese hier betrachteten „biologischen" Netzwerke zu konstruieren.

Die Idee, wie die Gewichte zu skalieren sind, um aus der Gewichtsmatrix nach Gleichung 5.42 die Leitwerte zu berechnen, ist ja bereits in Abschnitt 6.4 ausgeführt. Zu den Verzögerungszeiten ist dasselbe zu sagen, wie bei der erweiterten Projektionsregel.

Das Netzwerk, daß hier simuliert wurde, besteht aus $N = 30$ Neuronen. Um in den aus der Theorie vorhergesagten Speicherkapazitätsgrenzen zu bleiben, wurde die Länge der Zyklen und die Anzahl der Verbindungen gleich gewählt und zwar $D = 5$. Die mittlere Aktivität $m = p$ wurde mit 0.2 angenommen. Das ist wie in Abschnitt 4.5 erläutert ein passabler Wert. Es sollen $M = 4$ Zyklen gespeichert werden. In Links ausgedrückt, entspricht dies $L = MD = 20$ Links, und liegt somit weit oberhalb der Speichergrenze (ca. $0.15N$) eines Netzwerkes, mit nur einer Verbindung pro Neuronenpaar.

In Abbildung 6.13 ist das Ergebnis von 3 Simulationsdurchläufen zu sehen. In jedem Diagramm, wurde mit einem anderen Stimulus initialisiert (konsistent zu einem der gespeicherten Zyklen). Die Markierungen haben die selben Bedeutungen wie in Abbildung 6.12. Wie zu sehen ist, stimmen die vom Netzwerk produzierten Impulsfolgen gut mit denen überein, die gespeichert wurden. Lediglich bei Zyklus 2 bleiben einige erwartete Spikes aus (kein • in den ○ Markierungen, z.B. in der Impulsfolge von Neuron 18).

Es hat sich gezeigt, daß es bei diesem Netzwerk auch relativ wichtig ist, daß die im vorigen Abschnitt definierte Größe $G_{i,\mu}$ in einem bestimmten Bereich bleibt, da ansonsten die Wiedergabe der gespeicherten Zyklen nicht mehr funktioniert. Bleibt man jedoch in den von der Theorie vorgegebenen Speichergrenzen, so streuen diese Größen auch tatsächlich relativ wenig, und das Netzwerk erfüllt die gewünschte Funktionsweise.

6.7 Eine zeitpunktabhängige Konstruktion

Eine der größten biologischen Unplausibilitäten die die bis jetzt behandelten Netzwerke aufweisen, stellt die Verwendung von relativ großen Verzögerungszeiten dar. In [16, 5] wird zwar argumentiert, daß es im Kortex einen Anteil von ca. 20 - 40 % von Axonen gibt, welche Verzögerungszeiten von 100 bis 200 ms aufwei-

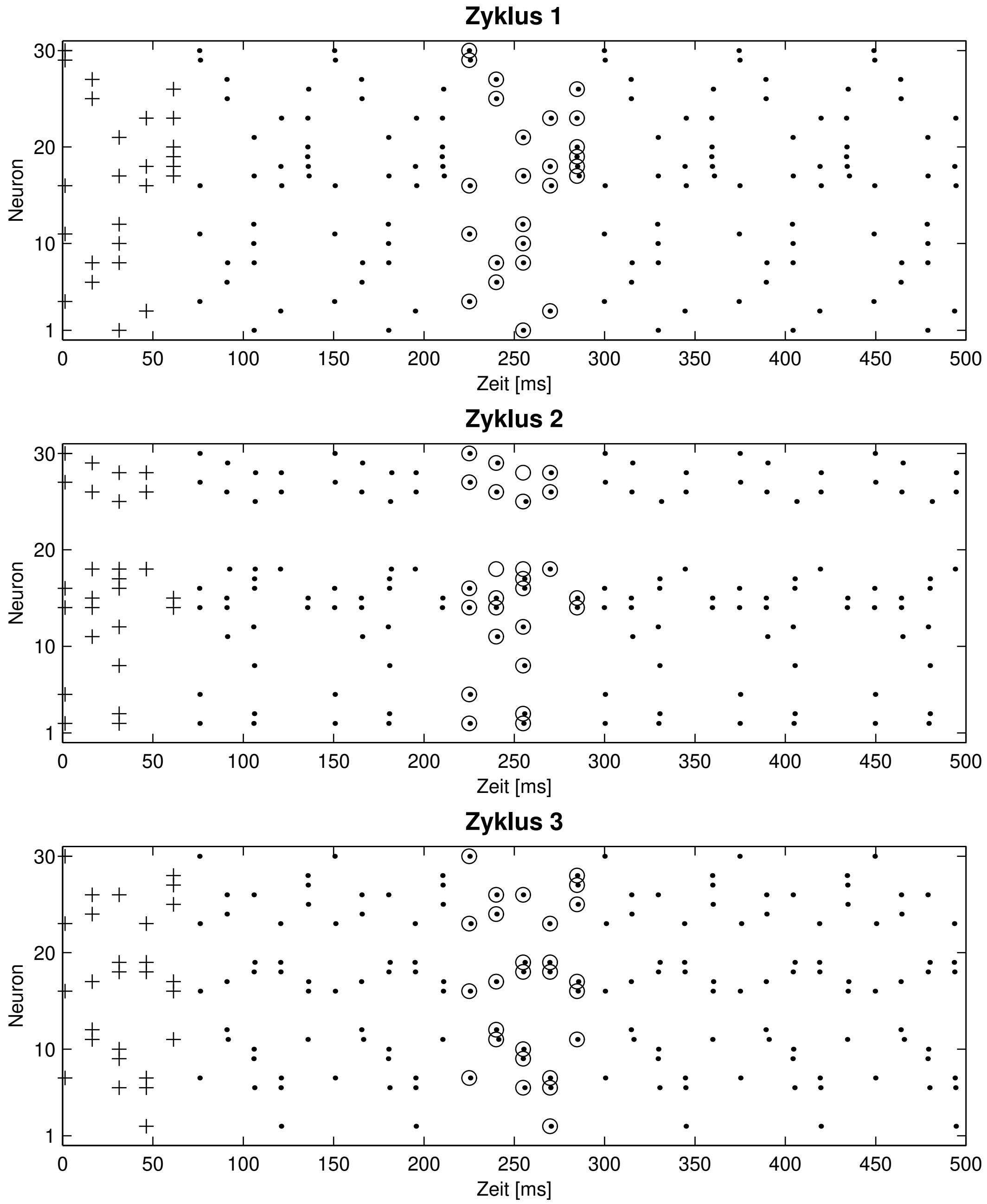


Abbildung 6.13: Impulsfolgen eines CPG, der nach der erweiterten Hebb'schen Regel konstruiert wurde. Die 3 Diagramme zeigen den zeitlichen Verlauf, je nach dem zu welchem der 4 gespeicherten Zyklen, der Stimulus (gekennzeichnet durch die + Markierungen) konsistent ist. Die ○ Markierungen kennzeichnen eine Periode des gewünschten Verlaufes. Man sieht, daß dieser gut mit dem tatsächlichen (• Markierungen) übereinstimmt.

sen, jedoch wird sonst überall von Verzögerungszeiten im Bereich einiger weniger Millisekunden ausgegangen.

Hier möchte ich nun eine Konstruktion (vgl. [10]) vorstellen, die mit solch relativ kurzen Delays auskommt, insofern sich dies mit dem zu speicherden spatiotemporären Muster vereinbaren läßt. Muster, die wie in den vorhergehenden Abschnitten, für 10 ms, und länger keine Spikes aufweisen, können natürlich mit solch kurzen Delays kaum gespeichert werden.

6.7.1 Biologische Motivation

Die Idee basiert auf jüngsten Ergebnissen aus der Neurobiologie. Es wurde nämlich festgestellt, daß ein Aktionspotential in den Pyramidenzellen (unterstützt durch aktive Kanäle in der Membran des Dendriten) sich *aktiv* auch am Dendritenbaum ausbreitet (vgl. [33]). Folglich ist die Information, ob ein, bei einer Synapse ankommender, Spike Einfluß auf die Generierung eines Aktionspotentials haben könnte, zumindest prinzipiell, einige Zeit später lokal an der Synapse verfügbar. In neuesten Experimenten (vgl. [24]) wurde festgestellt, daß sich die Stärke einer Synapse tatsächlich in Abhängigkeit der Zeitdifferenz des prä- und postsynaptischen Spikes verändert.

Formal wird man einfach eine Funktion $v(t_i - t_j^A)$ verwenden, die diese Zeitdifferenz zwischen dem eintreffenden Spike (t_j^A) und dem eventuell generierten Aktionspotential (t_i) bewertet.

Welche Form ist für $v(t_i - t_j^A)$ realistisch? Die folgenden Annahmen sind rein spekulativ und haben sich für die Simulation als günstig erwiesen. Es dauert eine gewisse Zeit τ_{DS}, bis das EPSP überhaupt zum Soma gelangt. Anschließend muß das Aktionspotential wieder am Dendritenbaum zurücktransportiert werden, was wiederum Zeit (τ_{SD}) benötigt. Liegt dieser dendritische Spike in einem gewissen Zeitfenster $[t_j^A + T_0, t_j^A + T_1]$ der Breite $\Delta = T_1 - T_0$, so soll die Stärke der Synapse erhöht werden. Das Zeitfenster könnte zum Beispiel durch das Vorhandensein des Neurotransmitters oder eines ähnlichen Stoffes ausgebildet werden. Ein denkbare Form für $v(t_i - t_j^A)$ wäre daher:

$$v(\tau) = \begin{cases} 1 & T_0 \le \tau \le T_1 \\ 0 & \text{sonst} \end{cases} \quad \text{mit} \quad \begin{array}{l} T_0 \le \tau_{DS} + \tau_{SD} \\ T_1 \ge \tau_{DS} + \tau_{SD} + \Delta_{chem} \end{array} \tag{6.30}$$

Δ_{chem} kennzeichnet die Breite des Zeitfensters, welches durch einen wie auch immer gearteten biochemischen Prozeß gebildet wird. Im Zeitraum zwischen T_0 und T_1 könnte man auch einen glockenförmigen Verlauf der Funktion wählen.

6.7.2 Anwendung zum Speichern von spatiotemporären Mustern

Wie kann man nun diese Lernregel anwenden, um gewisse vorgegebene Impulsfolgen F_i einzelner Neuronen i zu lernen ($F_i = \{t_i^{(s)}\}$).

Man betrachtet für jeden Spike s des Neurons i (Zeitpunkt $t_i^{(s)}$), welche Spikes anderer Neuronen j in Frage kommen, diesen zu generieren. Feuert Neuron j zur Zeit t_j, so wird dieser Spike über eine oder mehrere axonale Verbindungen mit jeweils verschiedenen Verzögerungszeiten $d_{i,j}^{\delta}$ an das Neuron i herangeführt. Liegt nun $t_i^{(s)}$ in dem günstigen Zeitfenster $[t_j + d_{i,j}^{\delta} + T_0, t_j + d_{i,j}^{\delta} + T_1]$ so wird die Synapse verstärkt. Daraus ergibt sich folgende Berechnungsvorschrift für das Gewicht einer Synapse, als Resultat eines zu lernenden Musters.

$$w_{i,j}^{\delta} = \sum_{t_i^{(s)} \in F_i} \frac{1}{A^{(s)}} \sum_{t_j \in F_j} v\left(t_i^{(s)} - t_j - d_{i,j}^{\delta}\right) \tag{6.31}$$

$$A^{(s)} = \left|\left\{t_j \in F_j : v(t_i^{(s)} - t_j - d_{i,j}^{\delta}) > 0\right\}\right| \tag{6.32}$$

$A^{(s)}$ ist die Anzahl jener präsynaptischen Spikes, die dafür verantwortlich gesehen werden, daß der Spike zur Zeit $t_i^{(s)}$ entstehen kann. Motiviert wird diese Normierung durch folgende rein spekulative Überlegung: Es könnte ja sein, daß ein rückwärtslaufendes Aktionspotential eine gewisse Menge eines, sagen wir Lernbotenstoffes, freisetzt. Dieser wird dann gleichmäßig zu jenen Synapsen gesendet, die „vor kurzem“ aktiviert wurden. (Die zum Soma strömenden EPSP's könnten ja eine Spur am Dendritenbaum hinterlassen. Es könnten dies irgendwelche Molekühle sein.) Je nach Menge des Lernbotenstoffes, der bei einer Synapse ankommt, wird diese verstärkt. Je mehr Synapsen also „bedient“ werden müssen, umso weniger werden sie verstärkt.

Will man mehrere Muster in einem Netzwerk speichern, so addiert man die Gewichte einfach auf.

Bevor die Ergebnisse einer Simulation vorgestellt werden sollen, noch eine Bemerkung zur prinzipiellen Funktionsweise dieser Lernregel. Betrachtet man Gleichung 6.31 etwas genauer, so stellt man fest, daß nur die Gewichte von Synapsen verstärkt werden, deren Verzögerungszeit mit einer Zeitdifferenz zweier Spikes einigermaßen übereinstimmt. Es findet also eine gewisse Selektion der vorhandenen Synapsen statt.

6.7.3 Simulationsergebnisse

Um die Gewichtswerte nach Gleichung 6.31 für das Kompartmentmodell verwenden zu können, müssen sie noch etwas skaliert werden. Als ein für die Simulation günstiger Wert hat sich ein Faktor von 2 herausgestellt. Betrachtet man dies von einer anderen Seite, so kann man sagen, daß das Netzwerk 2 Lerndurchgänge benötigt, um die Muster zu erlernen.

Es wurde ein Netzwerk mit $N = 200$ Neuronen konstruiert und simuliert. Drei Zyklen mit einer Periodendauer von 50 ms sollten gelernt werden. Jedes Neuron sollte in diesem Zeitraum einmal feuern. Der Zeitpunkt wurde auf einem Raster von 0.5 ms gleichverteilt gewählt. Als Ausgangsbasis wurden pro Neuronenpaar $D = 4$ Verbindungen zur Verfügung gestellt. Die Verzögerungszeiten wurden gleichverteilt aus einem Intervall von 1 ms bis 4 ms gezogen. Danach wurden die Gewichte nach Gleichung 6.31 bestimmt. Aus den in Abschnitt 6.2 durchgeführten Untersuchungen, haben sich für τ_{DS} und τ_{SD} Werte von 3 ms bzw. 0.3 ms ergeben. Für die Breite des Zeitfensters wurde 1 ms gwählt.

Um nun zu sehen, daß das Netzwerk auch das gewünschte leistet, wurde das Netzwerk mit einem Stimulus, der zu einem der gespeicherten Sequenzen konsistent ist für 25 ms initialisiert.

Die Ergebnisse dieser Simulationsläufe sind in den drei Spikeplots in Abbildung 6.14 zu sehen (Bedeutung der Markierungen siehe vorangegangene Abschnitte). Es sind nur 30 der 200 Neuronen dargestellt.

Aus den Ergebnissen dieser Simulationen kann man zwei Dinge schließen. Erstens ist es möglich in einem Netzwerk spatiotemporäre Muster mit einer zeitlichen Auflösung im ms Bereich zu speichern, und diese dann auch wieder zu reproduzieren. Dies ist eigentlich bemerkenswert, wenn man bedenkt, daß ein durchschnittliches EPSP doch ca. 15 ms lang ist. Zweitens kann man sehen, daß es möglich ist mehrere Muster in einem Netzwerk zu speichern.

Zur Netzwerkstruktur ist noch zu sagen, daß hier wirklich nur exzitatorische Synapsen verwendet wurden. Dies ist natürlich in keiner Weise realistisch, würde man nämlich nur ein wenig Noise hinzufügen, so würde sich das Netzwerk dadurch sehr schnell aufschaukeln und alle Neuronen mit der maximalen Frequenz zu Feuern beginnen.

6.8 Zusammenfassung

In dieser Arbeit wurden ziemlich „rohe“ Modelle eines NFN betrachtet. In allen Netzwerken wird ein und der selbe Typ von Neuron verwendet, wobei hingegen in realen biologischen Systemen mehrere Klassen von Neuronentypen vorkommen. Die Anzahl der Verbindungen zwischen zwei Neuronen ist für jedes Paar gleich, obwohl die in der Biologie auftretenden Verbindungsmuster weitaus komplexer sind. Außerdem hat jedes Neuron Verbindungen zu allen anderen Neuronen. In biologischen Systemen kann man jedoch, anhand der Verbindungsstruktur, verschiedene Areale und Subsysteme identifizieren. Aufgrund all dieser Aspekte können die vorgestellten Modelle nur eine grobe Approximation eines realen biologischen Netzwerkes darstellen.

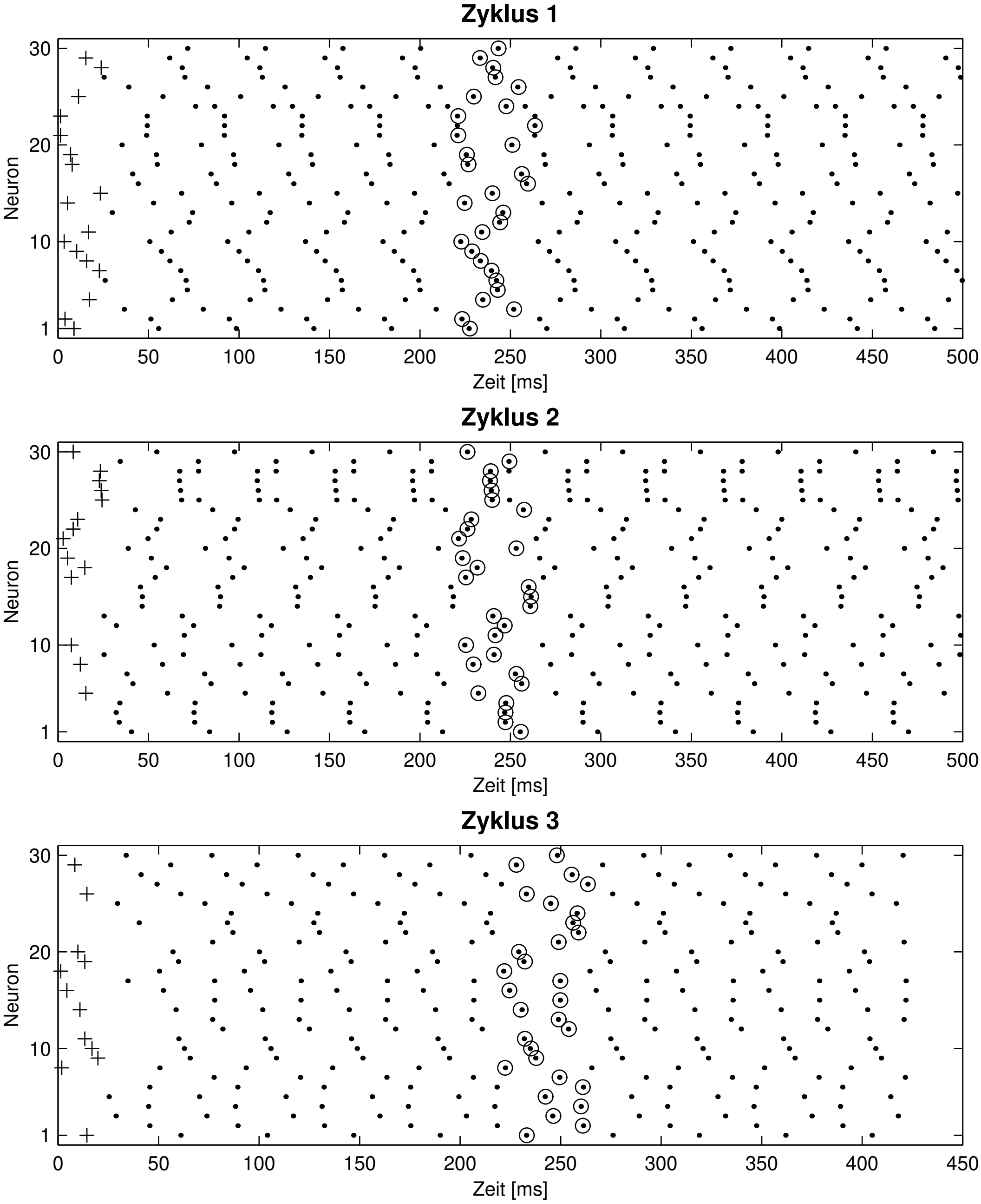


Abbildung 6.14: Wiedergabe von gespeicherten spatiotemporären Mustern. Die Muster wurden mit Hilfe von Gleichung 6.31 in ein Netzwerk von $N = 200$ Neuronen gespeichert (vgl. Text).

Andererseit beinhalten die verwendeten Modellneuronen viele Eigenschaften, die in guter Übereinstimmung mit biologischen Neuronen stehen. Dazu gehören die zeitlichen Parameter der PSP's (Anstiegszeit, Abfallzeit) sowie ein sehr realistisches Refraktärverhalten, welches durch das HH-Modell bestimmt wird (vgl. Abschnitt 6.2.4).

Aufbauend auf diesen Modellneuron wurde nun anhand von Simulationen gezeigt, daß es möglich ist Netzwerke aufzubauen, die vorgegebene spatiotemporäre Muster speichern. Diese Muster unterliegen jedoch, aufgrund der verwendeten Konstruktionen, einigen Einschränkungen. Für die Projektions- und die Hebb'sche Regel dürfen Spikes nur zu bestimmten Zeiten $k \cdot \Delta$ (Δ ca. 10 ms) auftreten. Es können jedoch mit diesen Konstruktionsverfahren viele Muster gespeichert werden. Bei der zuletzt besprochenen Lernregel, liegt der Fall umgekehrt. Es können keine Muster gespeichert werden, die für einen Zeitraum in der Größenordnung von 5 ms keinen Spike aufweisen. Diese Einschränkung ergibt sich daraus, daß die größte Verzögerungszeit in diesem Bereich liegt. Im Gegensatz zur Projektionsregel, muß hier die mittlere Netzwerkaktivität sehr gering sein, um eine gute Speicherung des Musters zu erzielen. Dies gilt auch für die Hebb'sche Regel, die ja eigentlich nur eine Erweiterung von Gleichung 6.31 ist. Die Genauigkeit, mit der die gespeicherten Muster wiedergegeben werden, ist jedoch bei allen Netzwerktypen gleich und liegt im Bereich von 1 ms.

In dieser Arbeit wurden nicht alle möglichen Konstruktionsverfahren angegeben. Ein Konzept, welches sehr ähnlich ist, ist das der „Synfire-Chains“, welches schon vor einigen Jahren von Abeles eingeführt wurde (vgl. [1]). Die Kernaussage ist, daß ein Neuron viel mehr erregt wird, wenn mehrere EPSP's gleichzeitig (synchron) ankommen. Die Idee der synchronen Spikes spielt auch in dieser Arbeit eine wichtige Rolle.

Ein anderer Ansatz wird in [21] gewählt. Es werden dort zwei Verbindungen zwischen zwei Neuronen verwendet, wobei die zweite eine größere Verzögerungszeit aufweist und die Übergänge von einem Zustand in den nächsten speichert, während die erste Verbindung Information über die einzelnen Zustände selbst speichert.

Anhang A

Speicherkapazität eines rekursiven CPG

A.1 Die Hebb'sche Regel, {+1,-1}-Kodierung

Die erste Möglichkeit, die einem einfällt, die Verteilungsfunktion des Terms h_i^N laut Gleichung 4.28 abzuschätzen, wäre, den Mittelwert und die Varianz, des als Zufallsvariable zu betrachtenden Terms $\zeta := \xi_i^{\sigma(\mu)} \xi_j^\mu \xi_j^a$, auszurechnen und den zentralen Grenzwertsatz der Wahrscheinlichkeitstheorie anzuwenden. Daraus ergibt sich folgende Verteilungsfunktion $F_{h_i^N}(z)$ für den Noise-Term.

$$\mathrm{E}[\zeta] = p^3 + 3p(1-p) - (1-p)^3 - 3p^2(1-p) = m^3 \tag{A.1}$$

$$\begin{aligned} \mathrm{Var}(\zeta) &= \mathrm{E}\left[(\zeta - m^3)^2\right] \\ &= (1-m^3)^2(p^2 + 3p(1-p)^2) + \\ &\quad + (-1-m^3)^2((1-p)^3 + 3p^2(1-p)) \\ &= 1 - m^6 \end{aligned} \tag{A.2}$$

$$F_{h_i^N}(z) = \Pr\{h_i^N \le z\} = \Phi\left(z, (L-1)m^3, \sqrt{\frac{L-1}{N}(1-m^6)}\right) \tag{A.3}$$

Wobei hier wie üblich p die Wahrscheinlichkeit angibt, mit der $\xi_i^\mu = +1$ ist und $m = 2p - 1$ die mittlere Aktivität $1/N \sum_{j=1}^N \xi_j^\mu$ in einem Muster ξ^μ darstellt. $\Phi(z, \overline{z}, \rho)$ ist die Verteilungsfunktion einer Normalverteilung mit Mittelwert $\overline{z}$ und Varianz ρ^2 wie sie in Gleichung 4.29 definiert ist.

Es hat sich nun in Vergleichen mit Simulationen herausgestellt, daß diese Abschätzung besonders für kleine L sehr ungenau ist, da in diesem Fall für $p \neq 1/2$ h_i^N keine einfache Normalverteilung mehr ist.

Um dies in den Griff zu bekommen, kann man eine genauere Abschätzung $F_{h_i^N}(z)$ der Verteilungsfunktion für h_i^N erhalten, wenn man die Summe in Gleichung 4.28 in $L-1$ Terme aufspaltet und die einzelnen Summanden $Z_\mu = 1/N \sum_{j=1}^{N} \xi_j^\mu \xi_j^a$ als normalverteilte Zufallsvariablen betrachtet.

$$h_i^N = \sum_{\mu=1,\mu\neq a}^{L} \xi_i^{\sigma(\mu)} \frac{1}{N} \sum_{j=1}^{N} \xi_j^\mu \xi_j^a = \sum_{\mu=1,\mu\neq a}^{L} \xi_i^{\sigma(\mu)} Z_\mu \tag{A.4}$$

Ist N relativ groß, so kann man eben wieder den zentralen Grenzwertsatz der Wahrscheinlichkeitstheorie anwenden, und aus dem Mittelwert und der Varianz von $\xi_j^\mu \xi_j^a$ auf die Verteilungsfunktion der Zufallsvariablen Z_μ schließen.

$$\begin{aligned} \mathrm{E}[Z_\mu] &= N\mathrm{E}\left[\frac{1}{N}\xi_j^\mu \xi_j^a\right] \\ &= N\left(\frac{1}{N}(p^2+(1-p)^2) - \frac{1}{N}2p(1-p)\right) \\ &= m^2 \end{aligned} \tag{A.5}$$

$$\begin{aligned} \mathrm{Var}(Z_\mu) &= N\mathrm{E}\left[\frac{1}{N^2}\left(\xi_j^\mu \xi_j^a - m^2\right)^2\right] \\ &= \frac{1}{N}\left((1-m^2)^2(p^2+(1-p)^2) + (-1-m^2)^2 2p(1-p)\right) \\ &= \frac{1}{N}(1-m^4) \end{aligned} \tag{A.6}$$

Die Verteilungsfunktion von h_i^N ergibt sich nun aus folgenden Überlegungen. Für eine fixe Belegung der Variablen $\xi_i^{\sigma(\mu)}$ mit $+1$ bzw. -1, ist die Wahrscheinlichkeit, daß die dann resultierende Summe der Zufallsvariablen Z_μ kleiner als z ist, bestimmt durch die Anzahl l der $\xi_i^{\sigma(\mu)}$ die den Wert $+1$ annehmen.

$$\begin{aligned} \Pr\left\{\sum_{\mu=1,\mu\neq a}^{N} \xi_i^{\sigma(\mu)} Z_\mu \leq z\right\} &= \Pr\{Z_{u_1} + \ldots + Z_{u_l} - Z_{v_1} - \ldots - Z_{v_{L-1-l}} \leq z\} \\ &= \Phi\left(z, lm^2 - (L-1-l)m^2, \sqrt{\frac{L-1}{N}(1-m^4)}\right) \end{aligned} \tag{A.7}$$

In obiger Gleichung stellen die Indizes u_j (v_j) jene μ dar, für die $\xi_i^{\sigma(\mu)} = +1$ (bzw. -1) ist. Beim Übergang von der ersten auf die zweite Zeile wurde von der Tatsache gebrauch gemacht, daß die Verteilung einer Summe zweier Normalverteilter Zufallsvariablen Z_1 und Z_2 wieder eine normalverteilte Zufallsvariable mit Mittelwert $\mathrm{E}[Z_1] + \mathrm{E}[Z_2]$ und Varianz $\mathrm{Var}(Z_1) + \mathrm{Var}(Z_2)$ ist.

Die Wahrscheinlichkeit, daß h_i^N kleiner als z ist, ergibt sich aus der Wahrscheinlichkeit des Auftretens eines Musters mit l $\{+1\}$-Elementen, multipliziert mit Gleichung A.7 aufsummiert über alle $l \in \{0, \ldots L-1\}$.

Die Wahrscheinlichkeit, das eine Zufallsfolge der Länge $L-1$ l mal $+1$ und $L-1-l$ mal -1 enthält, ist wie man sich leicht überlegt, durch folgende Gleichung gegeben.

$$\binom{L-1}{l} p^l (1-p)^{L-1-l} \tag{A.8}$$

Daraus ergibt sich folgende geschätzte Verteilungsfunktion für h_i^N.

$$\begin{aligned} F_{h_i^N}(z, N, L, m) = \sum_{l=0}^{L-1} & \binom{L-1}{l} p^l (1-p)^{L-1-l} \\ & \Phi\left(z, lm^2 - (L-1-l)m^2, \sqrt{\frac{L-1}{N}(1-m^4)}\right) \end{aligned} \tag{A.9}$$

Nun sind wir in der Lage die Wahrscheinlichkeit, daß ein ein Link korrekt gespeichert wird anzugeben. Setzt man nämlich die Gleichungen 4.31, 4.33 und A.9 zusammen, so erhält man:

$$\begin{aligned} \Pr\{\mathbf{x}(k+1) = \xi^{\sigma(a)}\} = \Big(& p(1 - F_{h_i^N}(-1, N, L, m)) + \\ & (1-p) F_{h_i^N}(1, N, L, m)\Big)^N \end{aligned} \tag{A.10}$$

Im Vergleich mit Simulationen hat sich herausgestellt, daß dies eine sehr viel bessere Annäherung darstellt, als Gleichung A.3. Den Kurven in Abbildung 4.4 liegt diese Gleichung zugrunde.

A.2 Die Hebb'sche Regel, {0,1}-Kodierung

In diesem Abschnitt soll die Abschätzung der Wahrscheinlichkeit $\Pr\{\mathbf{x}(k+1) = \xi^a\}$, daß ein Link mit Hilfe der Hebb'schen Regel nach Gleichung 4.36 korrekt gespeichert wurde, detailiert ausgeführt werden (vgl. Kapitel 7.5.4 [25]). Die prinzipiellen Überlegungen und Gleichungen sind ja bereits in Abschnitt 4.5 ausgeführt. Es müssen hier nur noch die Verteilungsfunktionen für die folgenden drei Größen

C				
$\xi_i^{(\mu)}$	ξ_j^μ	ξ_j^a	$X(C)$	$P(C)$
0	0	0	0	$(1-m)^3$
0	0	1	m^2	$m(1-m)^2$
0	1	0	0	$m(1-m)^2$
0	1	1	$m(m-1)$	$m^2(1-m)$
1	0	0	0	$m(1-m)^2$
1	0	1	$m(m-1)$	$m^2(1-m)$
1	1	0	0	$m^2(1-m)$
1	1	1	$(1-m)^2$	m^3

Tabelle A.1: Werte zur Berechnung des Mittlewertes und der Varianz von ζ

abgeschätzt werden.

$$h_i^S\big|_{\xi_i^{\sigma(a)}=1} = (1-m)^2 \frac{1}{N} \sum_{j=1}^{N} \xi_j^a - \theta_i \tag{A.11}$$

$$h_i^S\big|_{\xi_i^{\sigma(a)}=0} = -m(1-m) \frac{1}{N} \sum_{j=1}^{N} \xi_j^a - \theta_i \tag{A.12}$$

$$h_i^N = \frac{1}{N} \sum_{\mu=1,\mu\neq a}^{L} \sum_{j=1}^{N} (\xi_i^{\sigma(\mu)} - m)(\xi_j^\mu - m)\xi_j^a \tag{A.13}$$

Ist N relativ groß, so können alle drei Größen als normalverteilte Zufallsvariablen betrachtet werden, deren Mittelwert und Varianz sich aus denen der Terme ξ_j^a und $\zeta := (\xi_i^{\sigma(\mu)} - m)(\xi_j^\mu - m)\xi_j^a$ berechnen lassen.

$$\mathrm{E}[\xi_j^a] = 1m + 0(1-m) = m \tag{A.14}$$

$$\mathrm{Var}(\xi_j^a) = (1-m)^2 m + (0-m)^2(1-m) = m(1-m) \tag{A.15}$$

Die für den zweiten Term ζ notwendigen Werte und Wahrscheinlichkeiten zur Berechnung des Mittelwertes und der Varianz sind in Tabelle A.1 zusammengefaßt. Daraus ergibt sich:

$$\mathrm{E}[\zeta] = \sum_C X(C)P(C) = 0 \tag{A.16}$$

$$\mathrm{Var}(\zeta) = \sum_C (X(C) - \mathrm{E}[\zeta])^2 P(C) = m^3(1-m)^2 \tag{A.17}$$

Wendet man nun wieder den zentralen Grenzwertsatz der Wahrscheinlichkeitstheorie an, so ergeben sich folgende Mittelwerte und Varianzen für die obigen drei

Größen.

$$\overline{h_i^1} = \mathrm{E}\left[h_i^S\big|_{\xi_i^{\sigma(a)}=1}\right] = (1-m)^2 m - \theta_i \tag{A.18}$$

$$\rho_1^2 = \mathrm{Var}\left(h_i^S\big|_{\xi_i^{\sigma(a)}=1}\right) = \frac{1}{N} m (1-m)^5 \tag{A.19}$$

$$\overline{h_i^0} = \mathrm{E}\left[h_i^S\big|_{\xi_i^{\sigma(a)}=0}\right] = m^2 (m-1) - \theta_i \tag{A.20}$$

$$\rho_1^2 = \mathrm{Var}\left(h_i^S\big|_{\xi_i^{\sigma(a)}=0}\right) = \frac{1}{N} m^3 (1-m)^3 \tag{A.21}$$

$$\begin{aligned}
\overline{h_i^N} &= \mathrm{E}[h_i^N] \\
&= \frac{(L-1)N}{N} \sum_C X(C) P(C) = 0 \\
\rho_N^2 &= \mathrm{Var}(h_i^N) \\
&= \frac{(L-1)N}{N^2} \sum_C (X(C) - \overline{h_i^N})^2 P(C) \\
&= \frac{L-1}{N} m^3 (1-m)^2
\end{aligned} \tag{A.22}$$

Mit diesen Größen kann man nun für Gleichung 4.38 eine geschlossene Formel angeben, wobei wie im vorigen Abschnitt von der Tatsache Gebrauch gemacht wird, daß die Summer zweier normalverteilter unabhängiger Zufallsvariablen wieder eine normalverteilte Zufallssvariable ist, deren Mittelwert und Varianz durch die Summe der einzelnen Mittelwerte und Varianzen gegeben ist.

$$\begin{aligned}
\Pr\{\mathbf{x}(k+1) = \xi^{\sigma(a)}\} = \Bigg(m \left(1 - \Phi\left(0, \overline{h_i^1} + \overline{h_i^N}, \sqrt{\rho_1^2 + \rho_N^2}\right)\right) + \\
+ (1-m) \Phi\left(0, \overline{h_i^0} + \overline{h_i^N}, \sqrt{\rho_0^2 + \rho_N^2}\right)\Bigg)^N
\end{aligned} \tag{A.23}$$

Eine Frage die hier noch auftritt, ist die Wahl des Threshold θ_i. Um eine möglichst gute Trennung zu erhalten, legt man diesen am besten genau in die Mitte der beiden Mittelwerte $\overline{h_i^1}$ und $\overline{h_i^0}$. Daraus ergibt sich folgender Threshold

$$\theta_i = \frac{1}{2} m (1-m)(1-2m) \tag{A.24}$$

und dadurch erhält man für die beiden Mittelwerte folgende Werte:

$$\overline{h_i^0} = -\frac{1}{2} m (1-m) \tag{A.25}$$

$$\overline{h_i^1} = \frac{1}{2} m (1-m) \tag{A.26}$$

A.3 Simulationen zu Abschnitt 4.4

An dieser Stelle soll kurz angemerkt werden, wie die Simulationen, deren Ergebnisse in den Abbildungen 4.3, 4.4 und 4.5 dargestellt sind durchgeführt wurden.

All diese Simulationen wurden mit dem Mathematik Programm MATLAB® erstellt. Um ein Beispiel zu geben wie so ein MATLAB®-Programm aussieht, welches die durchschnittlich Anzahl von korrekt gespeicherten Links in einer $\{+1, -1\}$-Kodierung ermittelt, sei hier das wesentliche eines solchen Programmes abgedruckt.

```
N = 100; % Anzahl der Neuronen
L = 10;  % Anzahl der zu speichernden Links
p = 0.5; % Wahrscheinlichkeit des Auftretens von +1

X      = hardlims(p-rand(N,L)); % zufällige Muster generieren
sigma  = randperm(L);           % eine beliebige Funktion sigma
Xsigma = X(:,sigma);            % X entsprechend umordnen

W = 1/N * Xsigma * X';          % die Gewichtsmatrix berechnen
T = zeros(N,L);                 % die Thresholds zu Null setzen

CorrectLinks = sum(sum(abs(hardlims(W*X-T)-Xsigma))==0);
                     % CorrLinks Links wurden korrekt gespeichert
```

Dieser Kern wird nun sehr oft wiederholt, um eine statistische Signifikanz zu erlangen. Für die Kurven in den oben angeführten Diagrammen, wurde dieser Kern ca. 3000 mal pro Parametereinstellung (N,L und p) wiederholt und dabei der Mittelwert der Variablen `CorrLinks` berechnet.

Anhang B

Das Hodgkin-Huxley Modell

B.1 Allgemeines

Das heutige Verständnis der Erregbarkeit von Neuronen wurde sehr wesentlich von den Arbeiten von Hodgkin und Huxley beeinflußt. In einer Serie von fünf Artikeln, veröffentlicht im Jahr 1952, beschrieben sie die Schlüsseleigenschaften der Zellmembran, die einem Aktionspotential zugrunde liegen.

Aufgrund vieler Experimente gelang es ihnen, diejenigen Ströme und und Ionenarten zu identifizieren, die hauptsächlich zur Erzeugung eines Aktionspotentials beitragen. All diese Experimente wurden am sehr großen Motoaxon des Tintenfisches durchgeführt (Das Axon hat einen Durchmesser von ca. 1 Millimeter.). Das Modell, das daraus entwickelt wurde, ist jedoch sehr viel allgemeiner, man kann damit auch Ströme und Leitwerte beschreiben, die in diesem Axon nicht vorkommen (vgl. Kapitel 7 [7])

Aus den Experimenten ging als erstes hervor, daß hauptsächlich Natrium (Na) und Kalium (K) an der Entstehung eines Aktionspotentials beteiligt sind. Es mußte nun noch geklärt werden, welchen zeitlichen Verlauf diese Ionenströme besitzen. Mit Hilfe von sogenannten „voltage-clamp“ Versuchen, hat sich herausgestellt, daß diese Ströme von der Membranspannung abhängig sind.

B.2 Das mathematische HH-Modell

Obwohl Hodgkin und Huxley noch nichts über die spannungsabhängigen Kanäle wußten, als sie ihr Modell erstellten, ist es einfach die spannungsabhängigen Eigenschaften ihres Modells in dieser Terminologie zu beschreiben.

Die makroskopischen Leitwerte g_{Na} und g_K (vgl. Abbildung 6.1) des Modells kann man sich als das Resultat der Aktivität von sehr vielen mikroskopischen Kanälen vorstellen. Jeden einzelnen dieser Kanäle kann man als eine Menge von physikalischen Gattern betrachten, welche den Ionenfluß durch den Kanal steuern.

Sind all diese Gatter im durchlässigen Zustand, so ist der Kanal offen und es können Ionen hindurchfließen. Ist jedoch eines dieser Gatter undurchlässig, so ist der Kanal geschlossen und es können keine Ionen passieren.

Die Spanungsabhängigkeit im Hodgkin und Huxley Modell (HH-Modell) kommt nun dadurch Zustande, daß man annimmt, daß die Wahrscheinlichkeit p_i mit der ein Gatter des Typs i durchlässig ist, von der Membranspannung u_m abhängt.

Betrachtet man nun nicht einen Kanal sondern sehr viele, so kann man p_i auch als den Bruchteil von Gattern diese Typs auffassen, die durchlässig sind; $(1 - p)$ stellt somit den Anteil dar, welcher undurchlässig ist. Eine grundlegende Annahme im HH-Modell ist, daß die Änderung zwischen dem durchlässigen und undurchlässigen Zustand einer Dynamik erster Ordnung gehorcht.

$$\frac{dp_i(t)}{dt} = \alpha_i(u_m)(1 - p_i(t)) - \beta_i(u_m)p_i(t) \tag{B.1}$$

In dieser Gleichung stellen α_i und β_i die spannungsabhängigen Änderungsparameter dar, welche die „undurchlässig zu durchlässig“ und „durchlässig zu undurchlässig“ Änderungsraten beschreiben. Wird die Spannung u_m auf einem fixen Wert U_C gehalten, so erreicht der Anteil der durchlässigen Gatter eines Typs i einen fixen Wert $(dp_i(t)/dt = 0)$ wenn t gegen ∞ geht.

$$p_{i,\infty} := \lim_{t \to \infty} p_i(t)\big|_{u_m \equiv U_C} = \frac{\alpha_i(U_C)}{\alpha_i(U_C) + \beta_i(U_C)} \tag{B.2}$$

Der zeitliche Verlauf für diesen Fall einer unveränderlichen Membranspannung $u_m \equiv U_C$ ist durch eine einfache Exponentialfunktion mit folgender Zeitkonstante beschrieben.

$$\tau_i(U_C) = \frac{1}{\alpha_i(U_C) + \beta_i(U_C)} \tag{B.3}$$

Wenn nun ein bestimmter Kanal eines Typs k offen ist, trägt er einen kleinen fixen Wert zur gesamten makroskopischen Leitfähigkeit g_k dieser Art von Kanälen bei. Ist jedoch auch nur ein Gatter dieses Kanals undurchlässig, so ist dieser Beitrag gleich Null. Der Leitwert g_k ist also proportional zur Anzahl der offenen Kanäle, welche nun wieder proportional zur Wahrscheinlichkeit, daß alle Gatter eines Kanals durchlässig sind, ist. Damit ergibt sich für den makroskopischen Leitwert g_k, welcher aus Gattern i besteht, folgender Zusammenhang.

$$g_k(t, u_m) = g_k^{max} \prod_i p_i(t, u_m) \tag{B.4}$$

Wobei g_k^{max} den maximalen Leitwert angibt, wenn alle Gatter durchlässig sind.

B.3 Bestimmung der Gatterarten

Die Gleichung B.1 bis B.4 wurden bisher so allgemein gehalten, weil es damit möglich ist, eine Vielzahl von Leitwertarten zu beschreiben, nicht nur jene die von Hodgkin und Huxley im Tintenfischriesenaxon gefunden wurden.

Mit diesen zugrundeliegenden Gleichungen, sind nun Hodgkin und Huxley darangegangen, zu bestimmen, welche Arten von Gattern zur Modellierung der Na und K Kanäle notwendig sind. Mit den oben bereits erwähnten „voltage-clamp"-Experimenten, ist es ihnen gelungen zu eruieren, welche Gatter benötigt werden (vgl. Abschnitt 4.4 [7], und Kapitel 6 in [18]).

Es hat sich herausgestellt, daß man für den Na-Strom drei Gatter des Typs m und eines des Typs h benötigt; für den K-Strom vier des Typs n. Damit ergeben sich für die makroskopischen Leitwerte g_{Na} und g_K folgende Zusammenhänge:

$$g_{Na}(t, u_m) = g_{Na}^{max} {p_m}^3(t, u_m) p_h(t, u_m) \tag{B.5}$$

$$g_K(t, u_m) = g_K^{max} {p_n}^3(t, u_m) \tag{B.6}$$

Jedes dieser Gatter folgt nun einer Dynamik erster Ordnung, mit eigenen Änderungsraten.

$$\frac{dp_m(t)}{dt} = \alpha_m(u_m)(1 - p_m(t)) - \beta_m(u_m) p_m(t) \tag{B.7}$$

$$\frac{dp_h(t)}{dt} = \alpha_h(u_m)(1 - p_h(t)) - \beta_h(u_m) p_h(t) \tag{B.8}$$

$$\frac{dp_n(t)}{dt} = \alpha_n(u_m)(1 - p_n(t)) - \beta_n(u_m) p_n(t) \tag{B.9}$$

Ebenfalls aus den „voltage-clamp"-Experimenten konnten die Größen $p_{i,\infty}(u_m)$, $\tau_i(u_m)$ für alle drei benötigten Gatter bestimmt werden und daraus α_i und β_i. Die folgenden Gleichungen sind empirisch bestimmte Gleichungen, welche an die Daten aus den Experimenten angepaßt wurden.

$$\alpha_m(u_m) = \frac{A_m^\alpha (u_m - U_m^\alpha)}{e^{(\frac{u_m - U_m^\alpha}{B_m^\alpha})} - 1} \qquad \beta_m(u_m) = A_m^\beta e^{(\frac{u_m - U_m^\beta}{B_m^\beta})} \tag{B.10}$$

$$\alpha_h(u_m) = A_h^\alpha e^{(\frac{u_m - U_h^\alpha}{B_h^\alpha})} \qquad \beta_h(u_m) = \frac{A_h^\beta}{e^{(\frac{u_m - U_h^\beta}{B_h^\beta})} + 1} \tag{B.11}$$

$$\alpha_n(u_m) = \frac{A_n^\alpha (u_m - U_n^\alpha)}{e^{(\frac{u_m - U_n^\alpha}{B_n^\alpha})} - 1} \qquad \beta_n(u_m) = A_n^\beta e^{(\frac{u_m - U_n^\beta}{B_n^\beta})} \tag{B.12}$$

Die Parameter A, B, und U zu diesen Gleichungen sind in Tabelle B.1 angegeben. Um einen Eindruck von diesen Funktionen zu bekommen, sind sie in Abbildung B.1 graphisch dargestellt.

	Gatter m		Gatter h		Gatter n	
Parameter	α	β	α	β	α	β
A	$-0.1 \cdot 10^6$	4000	70	1000	$-1 \cdot 10^4$	125
B	-0.01	-0.018	-0.02	-0.01	-0.01	-0.08
U	$0.025 + U_R$	U_R	U_R	$0.03 + U_R$	$0.01 + U_R$	U_R

Tabelle B.1: Parameter zu den Gleichungen B.10, B.11 und B.12. Diese Werte sind so angegeben, daß die Membranspannung u_m in V einzusetzen ist und zwar bezogen auf das Zelläußere als Referenzpunkt. U_R bezeichnet dabei das Membranruhepotential, welches in dieser Arbeit mit $-70\,\text{mV}$ angenommen wird.

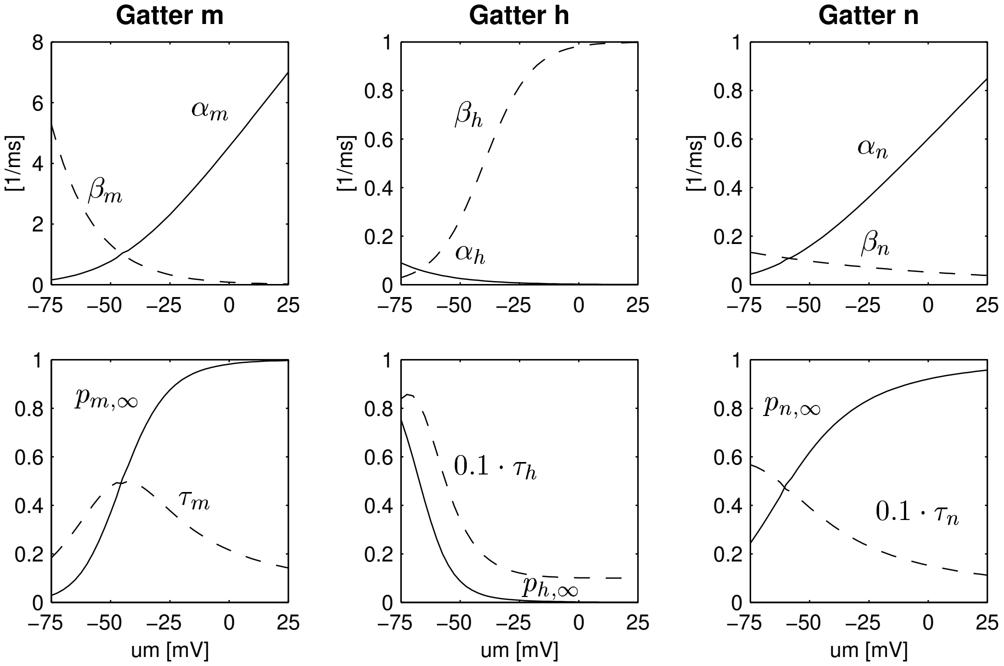


Abbildung B.1: Aktivierungsparameter für das Hodgin-Huxley-Modell. In der oberen Reihe sind die Parameter α_i (durchgezogen) und β_i (strichliert) dargestellt; in der unteren Reihe $p_{i,\infty}$ (durchgezogen) und τ_i (strichliert, in ms).

Zur Vervollständigung des Modell fehlen nun nur noch die Größen g_{Na}^{max} und g_K^{max}. Hodgkin und Huxley geben diese Werte in Bezug auf die betrachtete Membranfläche an.

$$g_{Na}^{max} = 1200 S/m^2 \tag{B.13}$$

$$g_K^{max} = 360 S/m^2 \tag{B.14}$$

Dies ist auch jene Art und Weise, wie sie in GENESIS Zelldefinitionen angegeben wird (vgl. Anhang C.1).

Anhang C

Listings wichtiger GENESIS Programme

C.1 Das Zellmodell

Das Zellmodell wird mit Hilfe des Cellreaders (vgl. Kapitel 16 [7]) konstruiert. Dazu ist es notwendig, die benötigten Teile im sogenannten *library* bereitzustellen. Dies sind die benötigten Kompartments und Kanäle.

Die Kompartments

Zunächsteinmal wird ein Standardtyp eines Kompartments definiert; einer für das Soma und einer für den Dendritenbaum.

File: Compartments.g

```
//genesis

// ===============================================
//  Biophysikalische Konstanten
//  (Alle Einheiten in SI Norm)
// ===============================================

// IONIC EQUILIBRIUM POTENTIALS
float  ENA       =  55e-3             // V
float  ECA       =  70e-3             // V
float  ECL       = -65e-3             // V
float  EK        = -90e-3             // V

float  EREST_ACT = -70e-3             // V
float  ELEAK     = EREST_ACT + 0.016  // V

// TYPICEL MEMBRANE PARAMETERS
float RM = 0.33 // ohm-m^2
float RA = 0.3  // ohm-m
float CM = 0.01 // F/m^2

// TYPICAL CELL DIMENSIONS
float SOMA_D  = 30.0e-6                   // m
float SOMA_L  = 30.0e-6                   // m
float SOMA_A  = PI * SOMA_L * SOMA_D      // m^2
float DEND_D  = 1.5e-6                    // m
float DEND_L  = 100.0e-6                  // m
float DEND_A  = PI * DEND_L * DEND_D      // m^2


//================================================
// Ein Prototyp eines Kompartments
//================================================

function Make_compartment

  if (!{exists compartment})
    create compartment compartment
    addfield  compartment Shape
  end

  setfield compartment \
    Cm     {CM * SOMA_A} \                  // F
    Ra     {4.0*RA*SOMA_L / SOMA_A} \       // ohm
    Em     {EREST_ACT} \                    // V
    Rm     {RM / SOMA_A} \                  // ohm
    dia    {SOMA_D} \                       // m
    len    {SOMA_L} \                       // m
    inject 0.0    \                         // A
    Shape  "cylinder"
end

//================================================
// Typisches Soma Kompartment
//================================================

function Make_soma
  if (!{exists soma})
    Make_compartment
    move compartment soma
  end

  setfield soma \
           Em     {ELEAK}     \             // V
           initVm {EREST_ACT}               // V
end

//================================================
// Typisches Dentriten Kompartment
//================================================

function Make_dend
```

```
  if (!{exists dend})
    create compartment dend
    addfield  dend Shape
  end

  setfield dend \
    Cm     {CM * DEND_A} \                // F
    Ra     {4.0*RA*DEND_L / DEND_A} \     // ohm
    Rm     {RM / DEND_A} \                // ohm
    Em     {EREST_ACT} \                  // V
    initVm {EREST_ACT} \                  // V
    dia    {DEND_D} \                     // m
    len    {DEND_L} \                     // m
    inject 0.0   \                        // A
    Shape  "cylinder"
end
```

Die Kanäle

In diesem File werden die verschiedenen Kanalarten definiert; synaptische und Aktivierungskanäle. Das Objekt `axonhilloc` wird vom Cellreader wie ein Kanal behandelt.

File: FastChannels.g

```
// genesis

//=================================================
// Kanaele fuer exzitatorische und inhibitorische
// Synapsen.
//=================================================

function Make_ExCh2

  if ({exists ExCh2})
    return
  end

  create synchan2 ExCh2

  setfield ExCh2 \
           tau1 0.002 \  // sec
           tau2 0.002 \  // sec
           gmax 1e-9  \  // S
           Ek 0.045      // V
end

function Make_InCh2
  if ({exists InCh2})
    return
  end

  create synchan2 InCh2

  setfield InCh2 \
           tau1 0.005 \  // sec
           tau2 0.005 \  // sec
           gmax 1e-9  \  // S
           Ek  -0.08     // V
end

//=================================================
// Kanaele, die ein Aktionspotential am Soma
// simulieren.
//=================================================

function Make_SNa_fast
  if ({exists SNa_fast})
    return
  end

  create channelC SNa_fast

  setfield SNa_fast \
           tau1 0.0002 \   // sec
           tau2 0.0002 \   // sec
           gmax 1e-9  \    // S
           Ek   0.045      // V
end

function Make_SK_fast
  if ({exists SK_fast})
    return
  end

  create channelC SK_fast

  setfield SK_fast \
           tau1 0.001 \   // sec
           tau2 0.001 \   // sec
           gmax 1e-9  \    // S
           Ek   -0.08      // V
end

//=================================================
// Aktive Na und K Kanaele zur Erzeugung eines AP.
//               tabchan - Objekt
//=================================================

function Make_Na_fast_tab

  if ({exists Na_fast_tab})
    return
  end

  create tabchannel Na_fast_tab

  setfield Na_fast_tab \
    Ek {ENA}  \          // V
    Gbar {1280*SOMA_A} \ // S
    Xpower 3 \
    Ypower 1 \
    Zpower 0

  setupalpha Na_fast_tab X \
   {0.2e6*(EREST_ACT+0.025)} -0.2e6 -1.0 \
   {-1.0*(0.025+EREST_ACT)}  -0.010      \
   8.0e3                      0.0    0.0 \
   {-1.0*(0.0  +EREST_ACT)}   0.018      \
   -size  {TABCHANNEL_SIZE} \
   -range {TABCHANNEL_RANGEMIN} \
          {TABCHANNEL_RANGEMAX}

  setupalpha Na_fast_tab Y \
     140 0.0 0.0 {-1.0*(0.0 +EREST_ACT)}  0.02 \
    2000 0.0 1.0 {-1.0*(0.03+EREST_ACT)} -0.01 \
    -size  {TABCHANNEL_SIZE} \
    -range {TABCHANNEL_RANGEMIN} \
```

```
           {TABCHANNEL_RANGEMAX}
end

function Make_K_fast_tab

  if ({exists K_fast_tab})
    return
  end

  create tabchannel K_fast_tab

  setfield K_fast_tab \
   Ek     {EK} \         // V
   Gbar   {360*SOMA_A} \ // S
   Xpower 4 \
   Ypower 0 \
   Zpower 0

  setupalpha K_fast_tab X \
   {2.0e4*(EREST_ACT+0.010)} -2.0e4 -1.0 \
   {-1.0*(0.010+EREST_ACT)}  -0.010      \
   250                        0.0    0.0 \
   {-1.0*(0.0  +EREST_ACT)}  0.080       \
   -size  {TABCHANNEL_SIZE} \
   -range {TABCHANNEL_RANGEMIN} \
          {TABCHANNEL_RANGEMAX}
end

//=================================================
// Aktive Na und K Kanaele zur Erzeugung eines AP.
//              hh_channel - Objekt
//=================================================

function Make_Na_fast_hh

  if ({exists Na_fast_hh})
    return
  end

  create hh_channel Na_fast_hh
  setfield Na_fast_hh \
  Ek      {ENA} \               // V
    Gbar    {1280 * SOMA_A}   \          // S
    Xpower  3.0   \
    Ypower  1.0   \
    X_alpha_FORM {LINOID} \
    X_alpha_A    -0.2e6 \               // 1/sec
    X_alpha_B    -0.010 \               // V
    X_alpha_V0   { 0.025 + EREST_ACT } \ // V
    X_beta_FORM  {EXPONENTIAL} \
    X_beta_A     8.0e3 \                // 1/sec
    X_beta_B     -18.0e-3 \             // V
    X_beta_V0    { 0.0 + EREST_ACT } \   // V
    Y_alpha_FORM {EXPONENTIAL} \
    Y_alpha_A    140.0 \                // 1/sec
    Y_alpha_B    -20.0e-3 \             // V
    Y_alpha_V0   { 0.0 + EREST_ACT } \   // V
    Y_beta_FORM  {SIGMOID} \
    Y_beta_A     2e3 \                  // 1/sec
    Y_beta_B     -10.0e-3 \             // V
    Y_beta_V0    { 30.0e-3 + EREST_ACT } // V
end

function Make_K_fast_hh

  if ({exists K_fast_hh })
    return
  end

  create hh_channel K_fast_hh
  setfield K_fast_hh \
    Ek     {EK} \                       // V
    Gbar   {360 * SOMA_A} \             // S
    Xpower 4.0 \
    Ypower 0.0 \
    X_alpha_FORM {LINOID} \
    X_alpha_A    -20.0e3 \              // 1/V-sec
    X_alpha_B    -10.0e-3 \             // V
    X_alpha_V0   {10.0e-3+EREST_ACT} \ // V
    X_beta_FORM  {EXPONENTIAL} \
    X_beta_A     250 \                  // 1/sec
    X_beta_B     -80.0e-3 \             // V
    X_beta_V0    {0.0+EREST_ACT}        // V
end

//================================================
// Diese Objekt dien dazu einen Spike zu
// detektieren, und dann an eine Synapse via
// SPIKE Message weiterzuleiten.
//================================================

function Make_axonhilloc

  if ( !{exists axonhilloc} )
    create spikegen axonhilloc
  end

  setfield axonhilloc \
           abs_refract  0.003 \ // sec
           thresh      -0.010 \ // V
           output_amp   1       // dimensionless
end
```

Die Zelldefinition

Nun wird der Zelldefinitionsfile für den Cellreader zusammengestellt.

File: fastcell.g

```
*relative
*cartesian
*lambda_warn

*set_global RM 0.333
*set_global RA 0.3
*set_global CM 0.01
*set_global EREST_ACT -0.070

*set_compt_param RM         0.3333   // ohm*m^2
*set_compt_param RA         0.3      // ohm*m
*set_compt_param CM         0.01     // F/m^2
*set_compt_param EREST_ACT -0.070    // V

*set_compt_param ELEAK -0.05088392604
soma none 30 0 0 30 Na_fast_hh 1200 \
                    K_fast_hh  360 axonhilloc -0.0

*set_compt_param ELEAK -0.070
dend soma 50 0 0 2 ExCh2 -1e-9 InCh2 -1e-9
```

Generieren des Zellmodells

Schlußendlich, wird nun alles zusammengefügt, und mit der Funktion `readcell` die Zelle generiert.

File: MakeACell.g

```
// genesis

include Compartments.g
include FastChannels.g

if ( !{exists /library} )
  create neutral /library
end

pushe   /library

  Make_compartment

  Make_Na_fast_hh
  Make_K_fast_hh

  Make_Na_fast_tab
  Make_K_fast_tab

  Make_ExCh2
  Make_InCh2

  Make_axonhilloc

pope
disable /library

readcell fastcell.p /cell
```

C.2 Analyse des Zellmodells

In Diesem Abschnitt sind jene Programme abgedruckt, mit welchen die in Abschnitt 6.2 erklärten Experimente durchgeführt wurden. Es sind jeweils nur die elementaren Teile angeführt.

Zum Beispiel wird vorausgesetzt, daß die Zelle schon geladen ist, und `/cell` heißt. Die in diesen Simulationen verwendeten Objekte `SpikeTrain`, `SpikeRecorder` und `MinMax` sind in der Standardversion von GENESIS nicht enthalten. Dies Objekt wurden in C geschrieben, und in GENESIS eingebunden (vgl. Kapitel 20 [6]).

Berechnung des Schwellwertes

Dieses Programm sucht mit Hilfe einer „binären“ Suche zischen zwei Vorgegebenen Werten den Schwellwert G_{th} und U_{th} (vgl. Abschnitt 6.2).

File: CalcThreshold.g

```
...
...
...

// Einen Spikegenerator erzeugen

   create SpikeTrain /st
   call /st ADDSPIKES 0.002

// Einen Spikerecorder erzeugen

   create SpikeRecorder /spikedetect
   setfield /spikedetect filename dummy
   addmsg /cell/soma/axonhilloc /spikedetect SPIKE

// Die Spitzenwerte der EPSP's messen

   // am Soma
   create  MinMax /MM
   addmsg /cell/soma /MM ABSINPUT Vm 1000 70

   // am Dentrit
   create MinMax /mm
   addmsg /cell/dend /mm ABSINPUT Vm 1000 70

// Die folgende Funktion bestimmt das Gewicht,
// das notwendig ist, um die Zelle zum Feuern
// zu bringen.

float GTh       // Schwellwert bzgl. Leitwert
float UThS      // Schwellwert bzbl. Membranspg.
float UThD      // deto am Dentriten.
float wMin =  0
float wMax =  8

function CalcTh( channel )

  str channel
  GTh = wMax

  while ( {abs {wMax-wMin}} >= 0.00005 )

    setfield {channel} synapse[0].weight {GTh}
    echo {GTh}

    reset > /dev/null
    step {Tmax} -time

    if({getfield /spikedetect lastspiketime}==-1)
      wMin = GTh                // kein Spike
```

```
      UThS = {getfield /MM max} // generiert
      UThD = {getfield /mm max}
    else
      wMax = GTh                // jetzt schon
    end
    GTh = {wMin+wMax}/2
  end
end

// Bestimmen des Thresholds fuer die ex. Syn

  str channel = "/cell/dend/ExCh2"

  addmsg /st {channel} SPIKE

  CalcTh {channel}

  echo UthS: {UThS} UthD: {UThD} GTh: {GTh}
```

Interaktion synchroner Spikes

Es wird hier jenes Gewicht $\overline{g}^{I}_{max}$ einer inhibitorischen Synapse gesucht, sodaß ein EPSP, welches durch eine exzitatorische Synapse mit dem Gewicht g^{E}_{max} erzeugt wurde, gerade nicht mehr ausreicht, um die Zelle zum Feuern zu bringen.

File: CalcW1W2.g

```
...
...
...

// Die beiden Synapsen synchron aktivieren

   float IS = 0.002
   create SpikeTrain /st
   call /st ADDSPIKES {IS}

   addmsg /st /cell/dend/ExCh2 SPIKE
   addmsg /st /cell/dend/InCh2 SPIKE

// den möglichen Spike detektieren

   create SpikeRecorder /spikedetect
   setfield /spikedetect filename dummy

   addmsg /cell/soma/axonhilloc /spikedetect SPIKE
   setfield /cell/soma/axonhilloc thresh -0.00


// Die folgende Funktion sucht nun jenes W2,
// welches verhindert, daß ein Aktionspotential
// erzeugt wird

float W2max // in diesem Bereich wird
float W2min // nach W2 gesucht.
float W1
float W2
float LS    // Zeitpunkt des Spikes

function FindW2(channel)

  str channel
  W2max = 400
  W2min = 0

  W2 = {W2max}

  while ( {abs {W2max-W2min}} >= 0.00001 )

    setfield {channel} synapse[0].weight {W2}

    reset > /dev/null
    step {Tmax} -time

    if ({getfield /spikedetect lastspiketime}==-1)
      W2max = W2   // kein Spike -> W2 zu grosz
    else
      W2min = W2   // hier schon -> W2 zu klein
      LS = {getfield /spikedetect lastspiketime}
    end
    W2 = {W2min+W2max}/2

  end
end

// Die Daten sichern

   str FILE = "W1W2.dat"
   openfile {FILE} w

// fuer einige Werte von W1 wird nun W2 gesucht

   for(W1=2; W1<=40; W1 = W1 + 2 )
     setfield /cell/dend/ExCh2 \
       synapse[0].weight {W1}
     echo {W1} -n
     FindW2 /cell/dend/InCh2
     echo " " {W2max} {W2min} {LS-IS*1e3}
     writefile {FILE} {W1} {W2max} {W2min} \
                      {LS-IS*1e3}
   end
   closefile {FILE}
```

Zeitpunkt des Aktionspotentials

Es wird hier der Zeitpunkt eines Aktionspotentials in Abhängigkeit der Gewichte einer exzitatorischen und einer inhibitorischen Synapse untersucht, welche synchron aktiviert werden.

File: CalcTiming.g

```
...
...
...
```

```
// die beiden Synapsen synchron akivieren

   float S1 = 0.002
   create SpikeTrain /st1
   call /st1 ADDSPIKES {S1}
   addmsg /st1 /cell/dend/ExCh2  SPIKE
   addmsg /st1 /cell/dend/InCh2  SPIKE

// den Feuerzeitpunkt merken

   create SpikeRecorder /spikedetect
   setfield /spikedetect filename dummy
   addmsg /{Cell}/soma/axonhilloc \
     /spikedetect SPIKE

// fuer verschiedene Wertepaare (W1,W2) den
// Feuerzeitpunkt merken.

// notwendige Variablen

   float W1
   float W2
   float W1max = 40
   float W2max = 400
   float nW1 = 40
   float nW2 = 200
   float dW1 = W1max / nW1
   float dW2 = W2max / nW2
   float SpikeDelay


// Die Daten speichern

   str FILE = "Timing.dat"
   openfile {FILE} w

// fuer verschiedene Wertepaare (W1,W2) den
// Feuerzeitpunkt merken.

   for(W1=dW1; W1<=W1max; W1=W1+dW1)
     for(W2=0; W2<W2max; W2=W2+dW2)

       setfield /cell/dend/ExCh2 \
         synapse[0].weight {W1}
       setfield /cell/dend/InCh2 \
         synapse[0].weight {W2}

       reset > /dev/null
       echo {W1} {W2} -n
       step {Tmax} -time
       SpikeDelay = {getfield /spikedetect \
                                   lastspiketime}
       if ( {SpikeDelay} != -1 )
         echo " "{SpikeDelay-S1*1000}
         writefile {FILE} {W1} {W2} \
                          {SpikeDelay-S1*1000}
       else
         echo " " -1
         writefile {FILE} {W1} {W2} -1
       end

     end
   end
   closefile {FILE}
```

Refraktärverhalten

Im ersten Programm wird der benötigte Stimulus in Abhängigkeit des Zeitraums seit dem letzten Aktionspotential ermittelt (vgl. Abschnitt 6.2.

File: CalcRefract.g

```
...
...
...

// die zwei ex. Synapsen zu verschiedenen Zeiten
// mit verschiedenen Gewichten aktivieren.

   create SpikeTrain /st1
   create SpikeTrain /st2
   addmsg /st1 /cell/dend/ExCh2 SPIKE
   addmsg /st2 /cell/dend/ExCh2 SPIKE

// den Rsultierenden zweiten Spike aufzeichnen

   create SpikeRecorder /sr
   setfield /sr filename dummy
   addmsg /cell/soma/axonhilloc /sr SPIKE
   setfield /cell/soma/axonhilloc abs_refract 8e-4

// Feuerzeitpunkt der Synapse E1

   float S1 = 0.001
   call /st1 ADDSPIKES {S1}

// Feuerzeitpunkt der Synapse E2

   float S2 = 0.005


// Die folgende Funktion ist wie jene, welche den
// Schwellwert bestimmt. Hier jedoch in Ab-
// haengigkeit eines bereits vorausgegangenen
// Aktionspotentials

float GTh
float wMin =  0.5
float wMax =  50
float ST1
float ST2
float ET

function CalcTh( channel, syn )

  int   syn
  str   channel
  float ts

  GTh  =  wMax

  while ( {abs {wMax-wMin}} > 0.0001 )

    setfield {channel} synapse[{syn}].weight {GTh}

    reset > /dev/null
    step {ET} -time
    call /sr  RESET
    step {Tmax} -time
```

```
    if ( {getfield /sr lastspiketime} = -1 )
      wMin = GTh
    else
      wMax = GTh
    end
    GTh = {wMin+wMax}/2
  end

end

// delay und W1 werden als Parameter verwendet

   float delay
   float W1

// Die Ergebnisse sichern

   str FILE = "Refrakt.dat"
   openfile {FILE} w

// fuer diverse Wertepaare von (W1,delay) den
// benötigten Stimulus berechnen.

   foreach W1 ( 1.5 3 6 12 24 )

  // bevor die Schleife fuer delay durchlaufen
  // wird, wird festgestellt, wann das erste
  // Aktionspotential zu Ende ist (ET).

     setfield /cell/dend/ExCh2 \
       synapse[0].weight {W1}
     reset > /dev/null
     ST1 = {getfield /sr lastspiketime}
     while ({ST1==-1})
       step 0.001 -time
       ST1 = {getfield /sr lastspiketime}
     end
     while ({getfield /cell/soma Vm} >= -0.07)
       step 0.0005 -time
       ET = {getstat -time}
     end

     for(delay=0.5; delay<=20; delay=delay+0.5)
       wMin =  0
       wMax =  18

    // Zeitpunkt für den zweiten Stimulus setzen

       call /st2 DELETESPIKES
       S2 = (ST1+delay)*1e-3
       call /st2 ADDSPIKES {S2}
       Tmax = (14+delay)*1e-3

       echo {W1} {delay} {ST1} " " -n
       CalcTh /cell/dend/ExCh2 1
       echo {GTh} {ST2-ST1}

       writefile {FILE} {W1} {delay} {ST1} \
                        {GTh} {ST2-ST1}
     end
   end

  closefile {FILE}
```

Das nächste Programm untersucht den Zeitpunkt des Auftretens eines Aktionspotentials, nach dem die Zelle zum Zeitpunkt t_1 bereits gefeuert hat (vgl. Abschnitt 6.2).

File: CalcRefrTiming.g

```
...
...
...

// Die Synapse E1 und E2 zu verschiedenen
// Zeitpunkten mit versch. Gewichten aktivieren

   create SpikeTrain /st1
   create SpikeTrain /st2
   addmsg /st1 /cell/dend/ExCh2 SPIKE
   addmsg /st2 /cell/dend/ExCh2 SPIKE

// Die Feuerzeit des zweiten Aktionspotentials
// ermitteln

   create SpikeRecorder /sr
   setfield /sr filename dummy
   addmsg /cell/soma/axonhilloc /sr SPIKE
   setfield /cell/soma/axonhilloc abs_refract 8e-4

// Aktivierungszeit der Synapsen E1 und E2

   float S1 = 0.001
   float S2 = 0.005
   call /st1 ADDSPIKES {S1}


// Ein paar Variablen

   float ST1
   float ST2
   float ET
   float delay // entspricht dS im Text
   float W1    // entspricht g_max^E1
   float W2    // entspricht g_max^E2

// Daten speichern

   str FILE = "RefrTiming.dat"
   openfile {FILE} w

// Die einzelnen Parametereinstellungen
// durchlaufen

   foreach W1 ( 1.5 3 6 12 24 )

  // bevor die Schleife fuer delay durchlaufen
  // wird, wird festgestellt, wann das erste
  // Aktionspotential zu Ende ist (ET).

     setfield /cell/dend/ExCh2 \
       synapse[0].weight {W1}
     reset > /dev/null
     ST1 = {getfield /sr lastspiketime}
     while ({ST1==-1})
       step 0.001 -time
       ST1 = {getfield /sr lastspiketime}
     end
     while ({getfield /cell/soma Vm} >= -0.07)
       step 0.0005 -time
       ET = {getstat -time}
     end
```

```
  for(delay=0.5; delay<=15; delay=delay+0.5)

    for(W2=0.5;W2<=20;W2=W2+0.25)

      setfield /cell/dend/ExCh2 \
        synapse[1].weight {W2}

      call /st2 DELETESPIKES
      S2 = (ST1+delay)*1e-3
      call /st2 ADDSPIKES {S2}
      Tmax = (14+delay)*1e-3

      echo {W1} {delay} {W2} {ST1} " " -n
      reset > /dev/null
      step {ET} -time
      call /sr RESET
      step {Tmax} -time
      ST2 = {getfield /sr lastspiketime}
      echo {ST2}

      writefile {FILE} {W1} {W2} {delay} \
                       {ST1} {ST2}
    end
  end
end

closefile {FILE}
```

C.3 Simulation von Netzwerken

Es soll hier kurz erläutert werden, wie die mit den verschiedenen Lernregeln und Konstruktionsverfahren erstellten Netzwerke (im speziellen die Leitwertmatrizen) in ein GENESIS-Modell umgewandelt werden können. Weiters ist eine kurze Funktion angegeben, welche eine bestimmte Anzahl von Spike generierenden Objekten (`SpikeTrain`) erzeugt und die definierten Feuerzeiten jedes dieser Objekte von einer Datei einliest.

Netzwerkdefinition

Die Netzwerkarchitektur wird durch eine Datei mit folgendem Format festgelegt.

```
# Anzahl der Neuronen

3

# Liste von Verbindungen
# von zu Gewicht Delay

   1   2       3   0.8
   2   3       4   1.0
   3   1       1   2.3
   2   2     0.5  -1.8

# Ende der Netzwerkdefinition
```

Eine Zeile die mit `#` beginnt, wird als Kommentar behandelt. Die erste nicht leere Zeile muß eine Zahl enthalten, welche die Anzahl der Neuronen angibt (hier mit N bezeichnet). Die nachfolgenden Zeilen legen die Verbindungen (Synapsen) zwischen den von 1 bis N durchnumerierten Neuronen fest.

Die erste Zahl ist die Identifizierung des präsynaptischen Neurons; die zweite Nummer legt das postsynaptische Neuron fest. Dahinter wird das Gewicht (in nS) der Synapse angegeben. Ist der Wert positiv, so wird eine exzitatorische Synapse erzeugt, andernfalls eine inhibitorische. Der letzte Wert einer Zeile gibt die Verzögerungszeit zwischen dem Auftreten des Aktionspotentials am Soma des präsynaptischen Neurons bis zur Auswirkung desselben an der Synapse an. Als ein ungewünschter Nebeneffekt, ist zu erwähnen, das der File mit einer Kommentarzeile abgeschlossen werden muß, auf die kein Zeilenumbruch mehr folgen darf. Die Funktion, welche einen solchen File einliest ist im folgenden abgedruckt.

File: ReadNetwork.g

```
int nNeurons
str NetworkPath
str NetworkPrefix = "n"
```

```
//
// Diese Variablen dienen als Uebergabe
// Variablen
//

str N1     // Nummer des praesyn. Neurons
str N2     // Nummer des postsyn. Neurons
str G      // Gmax der Synapse
str D      // Delay

function ReadConnection
//
// Diese Funktion liest eine Zeile aus dm File ein,
// und ueberprueft ob es ein Kommentar ist.
//
  N1 = {argv 1}
  N2 = {argv 2}
  G  = {argv 3}
  D  = {argv 4}

  if ( {argc} == 0 )
    return 0
  end
  if ( {strcmp {substring {N1} 0 0} "#"} != 0 )
    return 1
  else
    return 0
  end

end

function BuildFullConnected ( fn, cell, net )
//
// Diese Funktion liest Zeile für Zeile des
// Files fn ein.
// Als erstes werden N Neuronen vom Typ
// cell erstellt.
// Danach werden die Verbindungen erzeugt.
//

  str fn
  str cell
  str net

  int   N
  int   N_def = 0  // Flag ob N definiert ist

  int   neuron1
  int   neuron2
  float weight
  float delay

  int   nConnections = 0
  int   i
  int   synnr

  NetworkPath = {net}

  openfile {fn} read

  while ( !{eof {fn}} )
    //
    // eine Zeile einlesen
    //
    if ({ReadConnection {readfile {fn}}} == 1)
      if ( {N_def} == 0 )
        //
        // Erstellen der N Zellen
        //
        nNeurons = {N1}
        N_def = 1
        echo
        echo Creating {nNeurons} \
             neurons of type {cell} ...
        for( i=1; i<= {nNeurons}; i = i + 1 )
          copy {cell} {net}/{NetworkPrefix}{i}
        end
        echo Done.
        echo
        echo Creating connections ...
      else
        neuron1 = {N1}
        neuron2 = {N2}
        weight  = {G}
        delay   = {D}
        if ( {weight} > 0 )
          //
          // eine exzitatorische Synapse
          //
          addmsg {net}/{NetworkPrefix}{neuron1}\
                 /soma/axonhilloc \
                 {net}/{NetworkPrefix}{neuron2}\
                 /dend/ExCh2 SPIKE

          synnr = {getfield \
            {net}/{NetworkPrefix}{neuron2}\
            /dend/ExCh2 nsynapses}-1

          setfield \
            {net}/{NetworkPrefix}{neuron2}\
            /dend/ExCh2 \
            synapse[{synnr}].weight {weight}\
            synapse[{synnr}].delay  {{D}*1e-3}
        else
          //
          // eine inhibitorische Synapse
          //
          addmsg {net}/{NetworkPrefix}{neuron1}\
                 /soma/axonhilloc\
                 {net}/{NetworkPrefix}{neuron2}\
                 /dend/InCh2 SPIKE

          synnr = {getfield {net}/n{neuron2}\
            /dend/InCh2 nsynapses} - 1

          setfield \
            {net}/{NetworkPrefix}{neuron2}\
            /dend/InCh2 \
            synapse[{synnr}].weight {abs{weight}}\
            synapse[{synnr}].delay  {{D}*1e-3}
        end
        nConnections = nConnections + 1
      end
    end
  end
  echo {nConnections} connections created.
  echo
  closefile {fn}
end
```

Festlegen des Inputs eines Netzwerkes

Da viele Netzwerke einen gewissen Input brauchen, erscheint es sinnvoll, ein Tool zu haben, welches es einem einfach ermöglicht anzugeben, zu welchem Zeitpunkt ein Neuron einen externen Input (Spike) erhalten soll. In den Simulationen in dieser Arbeit, wurde dazu eine Funktion verwendet, die die benötigten Informationen aus einem File einliest.

```
# Beispiel zum Initialisieren des
# obigen Netzwerkes
# Neuron  t1 t2 t3 t4 ... tn
       1  10 16 17 23
       2   8 12 32
#
```

Wie im obigen Abschnitt, ist auch hier eine Zeile mit dem Zeichen `#` am Anfang eine Kommentarzeile. Die restlichen Zeilen beinhalten die Nummer des zu aktivierenden Neurons (erste Zahl) und die Zeitpunkte (in ms), zu denen das Neuron einen Spike erhalten soll (beliebig viele Zeiten in einer Zeile). Die Funktion, die diesen File einliest ist im Anschluß abgedruckt. Die unerwünschte Abschlußzeile muß auch hier eingefügt werden.

File: ReadInput.g

```
int nInputs
str InputPath
str InputPrefix  = "st"

function ReadSpikeTrain
//
// Diese Funktion erhaelt eine Liste von Zeiten
// die sie an ein SpikeTrain Objekt uebergibt.
//
  int i
  str nmr;

  if ( {argc} == 0 )
    return 0
  end
  if ( {strcmp {substring {argv 1} 0 0} "#"} != 0 )
    nmr = {argv 1}
    create SpikeTrain\
           {InputPath}/{InputPrefix}{nmr}
    for(i=2; i<={argc}; i=i+1)
      call {InputPath}/{InputPrefix}{nmr}\
           ADDSPIKES {{argv {i}}*1e-3}
    end
    return 1
  else
    return 0
  end
end


function BuildInput ( fn, path )
//
// Diese Funktion durchlaeuft alle Zeilen der
// Datei fn, und uebergibt sie an
// ReadSpikeTrain
//
  str fn
  str path
  int   nST = 0

  InputPath = {path}

  openfile {fn} read

  while ( !{eof {fn}} )
    if ( {ReadSpikeTrain {readfile {fn}}} == 1 )
      nST = nST + 1;
    end
  end

  echo {nST} spike sources generated.

  nInputs = {nST}

  closefile {fn}
end
```

Um die Simulation in Gang zu bringen, muß nur noch der Input mit dem Netzwerk durch geeignete `SPIKE`-Messages verbunden werden.

Literaturverzeichnis

[1] M. Abeles. *Corticonics: neural circuits of the cerebral cortex.* Canbridge University Press, 1991.

[2] M. Abeles, H. Bergman, E. Margalit, and E. Vaadia. Spationtemporal Firing Patterns in the Frontal Cortex of Behaving Monkeys. *Journal of Neurophysiologie*, 70(4):1629–1638, October 1993.

[3] M. Abeles, Y. Prut, H. Bergman, E. Vaadia, and A.Aertsen. Integration, synchronity and periodicity. In Aertsen, editor, *Brain Theory; spation-temporal aspects of brain function*, pages 149–181. ELSEVIER, 1993.

[4] M. Abeles, E. Vaadia, H. Bergman, Y.Prut, I. Haalman, and H. Slovin. Dynamics of Neuronal Interactions in the Frontal Cortex of Behaving Monkeys. *Concepts in Neuroscience*, 4(2):131–158, 1993.

[5] A.Herz, B.Sulzer, R. Khün, and J. L. van Hemmen. Hebbian learning reconsidered: Representation of static and dynamic objects in associative neural nets. *Biological Cybernetics*, 60:457–467, 1989.

[6] Dave Beeman, Upi Bhalla, et al. *GENESIS Reference Manual.* California Institute of Technologie, August 1995. included in the GENESIS distribution.

[7] James M. Bower and David Beeman. *The Book of GENESIS: exploring realistic neural model with the GEneral NEural SImulation System.* Springer-Verlag, Inc. Published by TELOS, New York, 1995.

[8] Thomas H. Brown and Sumatra Chattarji. Hebbian synaptic plasticity. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 454–459. MIT Press, 1995.

[9] Jeffrey Dean and Holk Cruse. Motor pattern generation. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 600–605. MIT Press, 1995.

[10] Wolfram Gerstner and J.L. van Hemmen. *Coding and Information Processing in Neural Networks*, chapter 1, pages 1–93. Physics of Neural Networks. Springer Verlag, New York, 1994.

[11] Leon Glass. Computing with attractors. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 186–189. MIT Press, 1995.

[12] M. Herrmann, J. A. Hertz, and A. Prügel-Bennett. Analysis of synfire chains. Technical report, NORDITA, Blegdamsvej 17, DK-2100 Copenhagen, Denmark, Jänner 1995.

[13] John Hertz. Computing with attractors. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 230–234. MIT Press, 1995.

[14] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[15] A. Herz, Z. Li, and J.L. van Hemmen. Statistical mechanics of temporal association in neural networks with transmission delays. *Physical Review Letters*, 66:1370–1373, 1991.

[16] A. Herz, B. Sulzer, R. Khün, and J.L. van Hemmen. The hebb rule: Storing static and dynamic objects in an associative neural network. *Europhysics Letters*, 7:663–669, 1988.

[17] Andreas V.M. Herz. Spatiotemporal Association in Neural Networks. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 902–905. MIT Press, 1995.

[18] Daniel Johnston and Samuel Miao Sin Wu. *Foundations of Cellular Neurophysiology.* MIT Press, 1995.

[19] Yves Kamp and Martin Hasler. *Recursive Neural Networks for Associative Memory.* Wiley-Interscience Series in Systems and Optimazition. John Wiley and Sons, New York, 1990.

[20] James P. Kelly and Jane Dodd. Anatomical organization of the nervous system. In Eric R. Kandel, James H. Schwartz, and Thomas M. Jessel, editors, *Principles of Neuroscience*, chapter 19. Prentice-Hall, third edition, 1991.

[21] David Kleinfeld and Haim Sompolinsky. Associative network models for central pattern generators. In Christof Koch and Idan Segev, editors, *Methods in Neural Modeling: From Synapses to Networks*, chapter 7, pages 195–246. MIT Press, 1992.

[22] Christof Koch and Idan Segev, editors. *Methods in Neural Modeling: From synapses to networks.* MIT Press, 1992.

[23] Wolfgang Maass. Lower bounds for the computational power of networks of spiking neurons. IIG—Report-Series 398, Institute for Theoretical Computer Science, October 1994.

[24] H. Makram and B. Sakmann. abstract 788.11 in the Prceedings of Neuroscience, 1995.

[25] Pierre Peretto, editor. *An Introduction to the Modeling of Neural Networks.* Cambridge University Press, 1992.

[26] Wilfrid Rall. Cable theory for dentritic neurons. In Christof Koch and Idan Segev, editors, *Methods in Neural Modeling: From Synapses to Networks*, chapter 2, pages 9–62. MIT Press, 1992.

[27] Raphael Ritz, Wulfram Gerstner, and J. L. van Hemmen. *Associative Binding and Segregation in a Network of Spiking Neurons*, chapter 5, pages 1–93. Physics of Neural Networks. Springer Verlag, New York, 1994.

[28] Sheldon M. Ross. *Introduction to Probability Models.* ACADEMIC PRES, INC., 1989.

[29] Robert F. Schmitt, editor. *Neuro- und Sinnesphysiologie.* Springer Verlag, 1993.

[30] Gordon M. Sheperd. *Neurobiology*, chapter 5. Oxford University Press, 1994.

[31] Kai-Yeung Siu, Vwani Roychowdhury, and Thomas Kailath. *Discrete neural computation ; a theoretical foundation.* information and system sciences series. Prentice-Hal, 1995.

[32] H. Sompolinsky and L. Kanter. Temporal Association in Asymetric Neural Networks. *Physical Review Letters*, 57:2861–2864, 1986.

[33] G.J. Stuart and B. Sakmann. Active propagation of somatic action potentials into neocortical pyramidal cell dendrites. *Nature*, 367:69–72, 1994.

[34] Simon Thorpe. Localized versus distributed representations. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 549–552. MIT Press, 1995.

[35] De Liang Wang. Temporal pattern processing. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 967–971. MIT Press, 1995.

Index