# Mustererkennung mit mehrschichtigen Netzen von dynamischen Neuronen


Eine Diplomarbeit von

Martin Bogensperger

am Institut für

**Grundlagen der Informationsverarbeitung**

Theoretical Computer Science

**Technische Universität Graz**


Betreuer:
*Dipl.-Inform. Dr. Berthold Ruf*
*Dipl.-Ing. Univ.-Ass. Thomas Natschläger*

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

Graz, August 1999

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

Das Gehirn hat tagtäglich eine Vielzahl an Aufgaben zu bewältigen. Es hilft uns bei selbstverständlich erscheinenden Dingen wie z. B. Gesichter zu erkennen. Es ist aber auch fähig, schwierige Probleme zu lösen. Einige Verhaltensweisen sind dem Menschen angeboren, aber jeder Mensch besitzt auch die lebenswichtige Fähigkeit zu lernen. Im Nervensystem sind also Mechanismen vorhanden, die es erlauben, Sinneseindrücke zu kodieren, zu filtern und zu speichern, sowie aus diesen Informationen zu lernen. Die Ergebnisse dieser „Berechnungen" machen es uns erst möglich zu existieren.

Die Forscher auf dem Gebiet der Neurophysiologie sind noch weit davon entfernt, die exakte Arbeitsweise biologischer neuronaler Netze zu verstehen. Man geht davon aus, daß Informationen im Gehirn in den Verbindungen zwischen den Neuronen gespeichert sind. Bei Lernprozessen werden neue Verbindungen zwischen Neuronen geschaffen oder bestehende Verbindungen verändert. Um bessere Einblicke in die Funktionsweise von neuronalen Netzstrukturen zu erhalten, werden biologische neuronale Netze mit künstlichen neuronalen Netzen nachgeahmt. 1943 entwarfen die amerikanischen Forscher McCulloch und Pitts die ersten künstlichen Netze. Seit damals wurde eine Vielzahl neuer Netzwerkstrukturen und Lernalgorithmen ersonnen. Die Geschichte der Neurophysiologie ist in einigen Büchern zu finden. Das Buch *Neurocomputing: Foundations of Research* [2] faßt z. B. 43 Artikel zusammen, die für diesen Zweig der Wissenschaft von historischem Interesse sind.

Eine wichtige Fragestellung ist, wie reale Zustände der Außenwelt intern im *Nervensystem* abgebildet werden. Noch vor einigen Jahren war man der Meinung, daß die wesentlichen Informationen in den *mittleren Feuerraten* von Neuronen enthalten seien. Seit einiger Zeit wird aber auch ein anderer Ansatz verfolgt. In realen Neuronen treten kurze Spannungsimpulse *Aktionspotentiale* genannt, auf. Im Vordergrund des neuen Ansatzes stehen die genauen Zeitpunkte von Aktionspotentialen. Experimente mit visueller Mustererkennung deuten darauf hin, daß die Information in der zeitlichen Abfolge der Aktionspotentiale kodiert sein muß. So zeigten Messungen bei Menschen und Affen, daß sie in der Lage sind, Muster in ca. 100 ms zu erkennen (vgl. [25]), obwohl mindestens 10 Schichten von Synapsen zwischen der Retina und dem Schläfenlappen liegen. Das bedeutet, daß in dem Bereich der Kortex, die für Visualisierung zuständig ist, eine Schicht ihre Berechnungen in ungefähr 10 ms durchführen muß. Berücksichtigt man die Feuerrate von weniger als 100 Hz der Neuronen in dieser Region, so wäre schon ein Intervall von über 10 ms notwendig, um die mittlere Feuerrate zu bestimmen. Aufgrund dieser Beobachtung scheint die Kodierung in mittleren Feuerraten eher unwahrscheinlich zu sein.

Viele Fortschritte in der Erforschung neuronaler Netze hatten mit neuen Netzwerkstrukturen und Lernalgorithmen zu tun. Ebenso wichtig ist aber die Möglichkeit, diese neuen Konzepte mit den heutigen leistungsfähigen Computern testen zu können. Immer schnellere Computer haben in den letzten Jahren auch aus der klassischen Neurobiologie den Zweig der *computational neuroscience* entstehen lassen. Es stellte sich heraus, daß die immer genauer werdenden physiologischen und anatomischen Messungen am Nervensystem nicht ausreichten, um die Funktionsweise biologischer neuronaler Netze zu erforschen. Mit neurobiologischen Simulationsprogrammen hat man eine relativ einfache Möglichkeit gefunden, Vorgänge in biologisch realistischen Netzen zu beobachten. Im zweiten Teil dieser Arbeit werden Ergebnisse, die mit einem solchen Simulator erzielt worden sind, beschrieben.

## Überblick

In den ersten beiden Kapiteln werden zunächst die Grundlagen erläutert. Kapitel 2 befaßt sich mit den biologischen Grundlagen. Es werden die Grundkomponenten für den Aufbau eines neuronalen Netzwerkes gezeigt und die für eine Simulation notwendigen mathematischen Formulierungen dieser Komponenten dargestellt. Ein Abschnitt behandelt die von den Neurophysiologen Hodgkin und Huxley aufgestellten Theorien über die Generierung von sogenannten Aktionspotentialen. Die beiden Wissenschaftler erhielten für diese Arbeit später den Nobelpreis.

Kapitel 3 soll eine kurzen Einblick in künstliche neuronale Netze bieten, also Aufbau und Bestandteile von Netzwerken sowie Lernalgorithmen, die auf solche Netzwerke angewandt werden können. Im Speziellen wird dabei auf den sehr bekannten Backpropagation Algorithmus eingegangen. Dieses Lernverfahren wird später bei den simulierten biologischen Netzwerken eingesetzt.

In Kapitel 4 wird der theoretische Kern dieser Arbeit vorgestellt. Bisher waren künstliche Netzwerke nur Versuche, biologische neuronale Strukturen nachzuempfinden. Mit dem vorgestellten Konstrukt sollte es theoretisch möglich sein, künstliche Netzwerke in biologisch realistische Netzwerke überzuführen.

Schließlich werden die vorigen Kapitel in Kapitel 5 zusammengeführt und einige Simulationsergebnisse gezeigt. Verwendet wird dabei das Softwarepaket GENESIS (vgl. [6], [3] und Anhang A), das Computersimulationen von biologisch realistischen Neuronen ermöglicht. Für kleine Netzwerke wird untersucht, ob die in Kapitel 4 beschriebene Theorie auf Computermodelle von realen Neuronen übertragbar ist. Dabei werden zuerst künstliche Netzwerke mit gleichwertigen Computermodellen biologisch realistischer Netzwerke verglichen. In weiterer Folge wird dann versucht, direkt auf diese Computermodelle den Backpropagation Lernalgorithmus anzuwenden.

# Kapitel 2

# Grundlagen

Über die Architektur und Funktionsweise einzelner biologischer Neuronen ist schon sehr viel bekannt. Die biologischen Grundlagen dazu werden im ersten Teil dieses Kapitels kurz dargestellt. Darauf aufbauend gelangt man zu mathematischen Formulierungen einzelner Komponenten, auf deren Basis auch verschiedenste Simulationsprogramme funktionieren.

## 2.1 Aufbau des Nervensystems

Das menschliche Nervensystem kann grob in ein Zentralnervensystem, bestehend aus Gehirn und Rückenmark, und ein peripheres Nervensystem unterteilt werden. Die für höhere geistige Leistungen notwendigen neuronalen Strukturen befinden sich im Zentralnervensystem. Das periphere Nervensystem ist für die Verteilung von Signalen im Körper zuständig. Bedingt durch den unterschiedlichen Aufgabenbereich dieser beiden Systeme treten natürlich unterschiedliche Formen von Nervenzellen auf. Beim Zentralnervensystem handelt es sich um ein gigantisches Netzwerk von geschätzten $10^{10}$ Nervenzellen, auch *Neuronen* genannt. Die beiden wichtigsten Grundtypen von Zellen sind *Pyramidenzellen* und *Sternzellen*. Die große Mehrheit der Neuronen ist vom Typ der *Pyramidenzellen* (bis zu 85%). Daher werden die übrigen Zellen oft unter dem Begriff „Nichtpyramidenzellen“ zusammengefaßt. Verbunden sind die Neuronen durch sogenannte *Synapsen*. Die Zahl der Synapsen ist – je nach Gehirnteil – um den Faktor 1000 bis 10000 mal größer als die der Neuronen und liegt in der Größenordnung von $10^{14}$. Jedes Neuron wird also von ca. $10^4$ anderen Neuronen kontaktiert. Für diese riesige Anzahl von Verbindungen benötigen die Neuronen eine große Oberfläche, die durch den *Dendritenbaum* gebildet wird. Zur Weiterleitung ihrer Informationen an andere Neuronen dient das *Axon*, das sich auch verzweigt und wiederum über Synapsen Verbindungen zu anderen Neuronen herstellt. Die Hauptunterschiede zwischen Pyramidenzellen und Sternzellen liegen einerseits in der Länge der

Axone und andererseits in der Art dieser Verbindungen. Sternzellen haben nur kurze Axone, und sie wirken hauptsächlich hemmend auf nachgeschaltete Neuronen. Pyramidenzellen besitzen kurze sowie lange Axone, die bis ins Rückenmark reichen können und meist erregend auf nachgeschaltete Neuronen wirken. In Abbildung 2.1 sind verschiedene Zelltypen mit ihren wichtigsten morphologischen Teilen zu sehen.

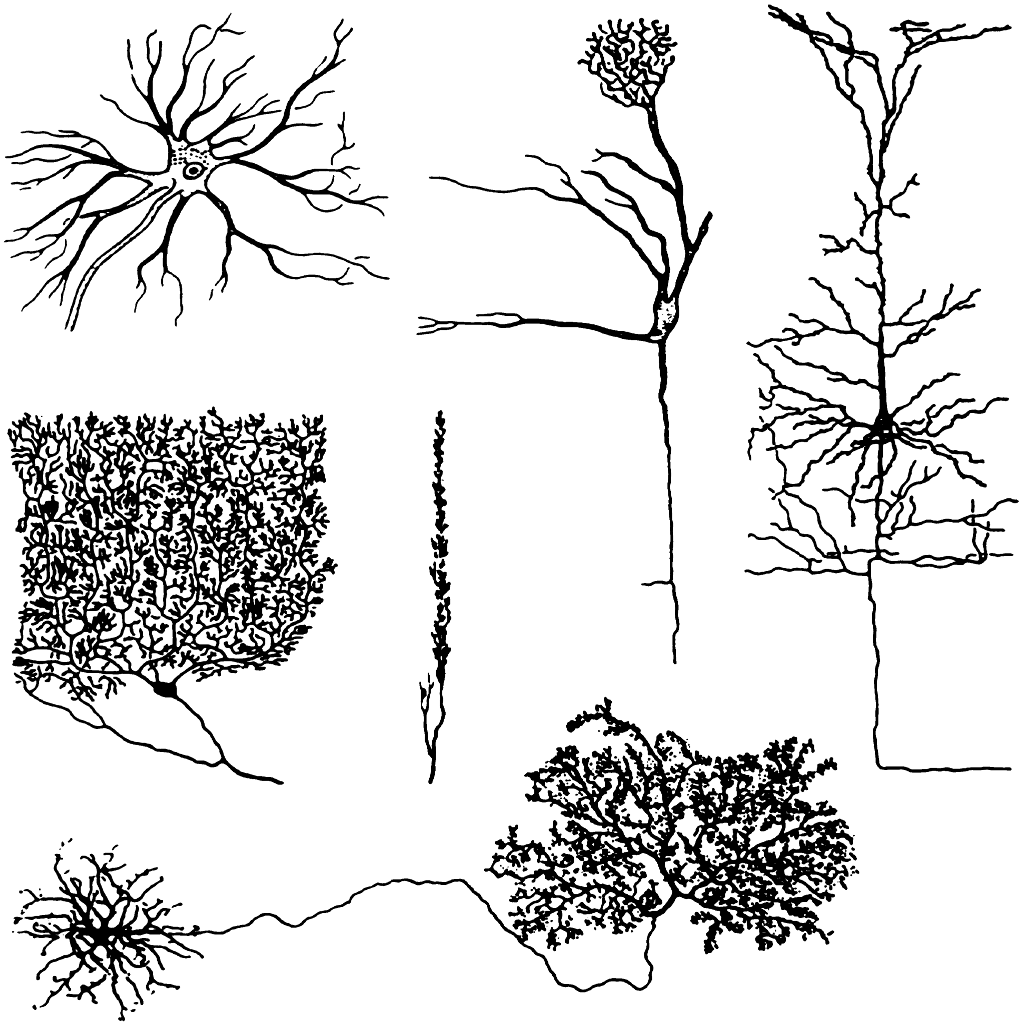


**Abbildung 2.1:** Diese Beispiele von Neuronen im Nervensystem zeigen die vielfältige Morphologie. Von links oben im Uhrzeigersinn: Spinales Motorneuron, Mitralzelle der Geruchsnerven, Pyramidenzelle aus dem Kortex, horizontale Zelle aus der Retina und eine Purkinje Zelle (Vorderansicht und Seitenansicht) aus dem Cerebellum (aus [9]).

### 2.1.1 Neuronen

Ein Neuron besteht aus dem Zellkörper oder *Soma* und aus 2 Arten von Fortsätzen: den *Dendriten* und dem *Axon*. Vom Soma gehen meist mehrere Dendriten ab, die sich – ähnlich Ästen eines Baumes – stark verzweigen können. Es wird daher oft der Ausdruck *Dendritenbaum* verwendet. An verschiedenen Stellen des Dendritenbaumes gibt es Verbindungen (*Synapsen*) zu vorangeschalteten (präsynaptischen) Neuronen. In den Dendriten wird die Erregung anderer Nervenzellen (Input–Signale) aufgenommen und zum Soma hingeleitet. Überschreiten die im Soma aufsummierten Inputs eine gewisse Schwelle (*threshold*), so wird ein sogenanntes *Aktionspotential* generiert und entlang des Axons zu anderen (postsynaptischen) Neuronen fortgeleitet.

Im Zusammenhang mit schnellen Berechnungen in der Kortex nimmt man an, daß die Inputs und Outputs in Aktionspotentialen kodiert sind. Da die Aktionspotentiale nach dem „alles–oder–nichts Prinzip" generiert werden, könnte man meinen, es handle sich hier um ein binäres System. Entscheidend (und zentraler Punkt in dieser Arbeit) ist aber der Zeitpunkt, an dem ein Aktionspotential generiert wird.

Wie sieht nun so ein Aktionspotential aus, und wie wird es im Detail vom Neuron erzeugt? Neuronen sind von einer Zellmembran umgeben, welche aus einer Doppelschicht von Lipiden besteht. Im Inneren und außerhalb der Membran befindet sich eine mit Ionen angereicherte wässerige Salzlösung. Für geladene Teilchen stellt die Membran ein Diffusionshindernis dar, sie hat deshalb einen hohen elektrischen Widerstand.

Die hauptsächlich vorkommenden Ionenarten sind Kaliumionen ($K^+$), Natriumionen ($\mathrm{Na}^+$), Chlorionen ($\mathrm{Cl}^-$) und Calciumionen ($\mathrm{Ca}^{+2}$). Zur Steuerung der Ionenkonzentrationen sind in die Lipiddoppelschicht der Membran Proteinmoleküle eingebettet, die als Ionenkanäle funktionieren. Man kann sich diese Kanäle als Schalter oder Pumpen vorstellen. Die 3 wichtigsten Arten von Ionenkanälen und deren Funktion wird hier kurz zusammengefaßt (vgl. [9], [21]):

**Ionenpumpen:** Sie können entgegen ihrem Konzentrationsgradienten (= Unterschied der Ionenkonzentration innerhalb und außerhalb der Zellmembran) Ionen durch die Membran pumpen. Dieser Prozeß ist natürlich energieaufwendig, sodaß über ein Drittel des Energieverbrauches einer Zelle von den Ionenpumpen beansprucht wird. Die wichtigste dieser Pumpen ist die **$\mathrm{Na}^+$-$\mathrm{K}^+$–Pumpe**. Sie pumpt in einem Pumpzyklus jeweils 3 $\mathrm{Na}^+$ Ionen aus dem Zellkern hinaus und dafür 2 $\mathrm{K}^+$ Ionen hinein und ist damit verantwortlich für den $\mathrm{Na}^+$ und $\mathrm{K}^+$ Konzentrationsunterschied innerhalb und außerhalb der Zelle.

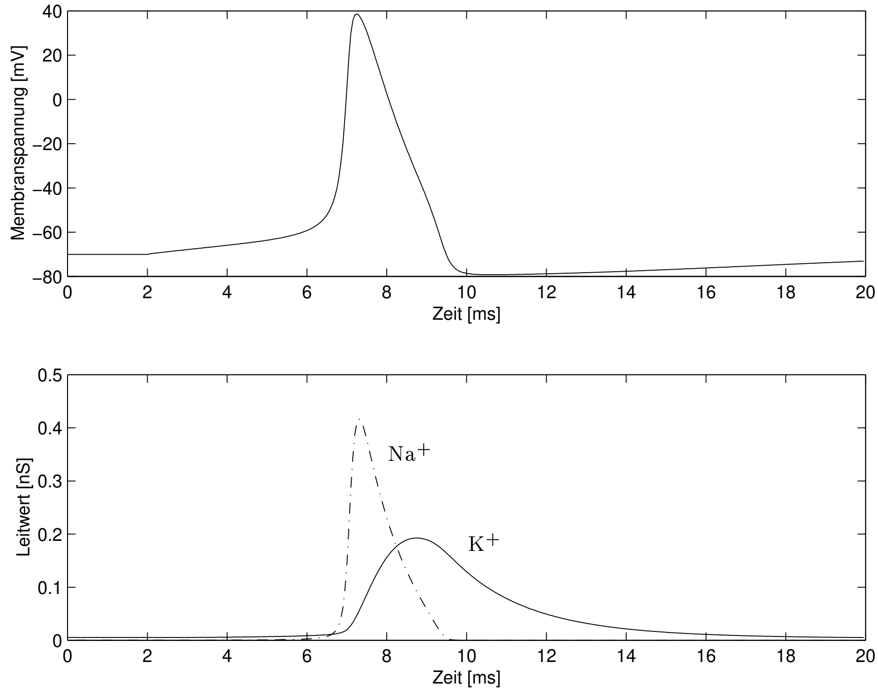



**Abbildung 2.2:** Zeitlicher Verlauf eines Aktionspotentials berechnet nach dem Hodgkin–Huxley Modell des Axons eines Riesentintenfisches. Das obere Diagramm zeigt den Verlauf der Membranspannung am Soma. Im unteren Diagramm sieht man die Leitfähigkeit der aktiven Kanäle für $Na^+$ (gestrichelte Linie) und $K^+$ (durchgezogene Linie). Die Öffnung der $Na^+$ Kanäle bewirkt die Depolarisation der Zelle bis zu ca. +40 mV. Die etwas verzögerte Öffnung der $K^+$ Kanäle repolarisiert die Zelle anschließend wieder. Dieses Bild wurde mit dem Simulationsprogramm GENESIS erstellt (siehe Anhang A).

**Passive Kanäle:** Diese Kanäle sind permanent für eine Ionenart durchlässig und verbrauchen keine Energie. Sie werden nur durch den jeweiligen Ionenkonzentrationsgradienten angetrieben. Die passiven Kanäle sorgen somit für das Zustandekommen des *Membranruhepotentials*.

**Aktive Kanäle:** Diese Kanäle erzeugen unter anderem die Aktionspotentiale. Man kann sie sich als spannungsabhängige Schalter vorstellen. Erreicht das Potential im Inneren der Zelle einen gewissen Wert, so öffnen sich diese Kanäle plötzlich und lösen dadurch ein Aktionspotential aus (siehe Abbildung 2.2).

Durch die oben vorgestellten Kanäle befinden sich nun intrazellulär viele $K^+$ Ionen und nur wenige $Na^+$ und $Cl^-$ Ionen, und extrazellulär ist die $Na^+$ und $Cl^-$ Konzentration hoch und die $K^+$ Konzentration niedrig. Die unterschiedliche Ionenkonzentration im Ruhezustand (d. h. es kommen keine Eingangssignale von

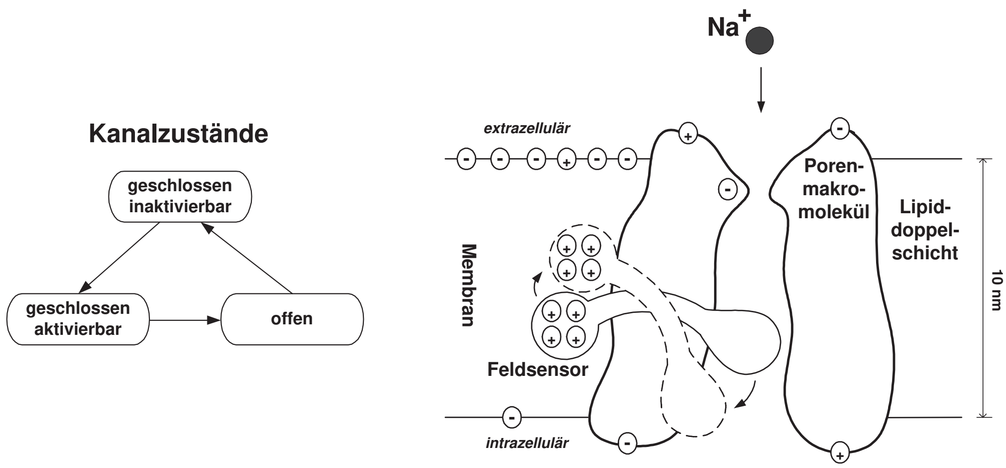



**Abbildung 2.3:** Links: Schema der Zustände der Na–Kanäle. Rechts: Modellschema eines $Na^+$–Kanals der Membran. Die Größenverhältnisse der Membrankomponenten und Ionen sind etwa maßstabsgetreu (aus [21]).

den Dendriten) hat eine Potentialdifferenz zwischen Zellinnerem und der die Zelle umgebenden Flüssigkeit zur Folge. Dieses Membranruhepotential liegt je nach Zelltyp zwischen -40 mV und -90 mV.

Abhängig von den Inputs über den Dendritenbaum kann das Membranpotential am Soma verändert werden. Spannungsänderungen in die positive Richtung werden als *Depolarisation* und in die negative Richtung als *Hyperpolarisation* bezeichnet. Ist die Depolarisation des Somas so groß, daß sie eine gewisse Schwelle erreicht (siehe Abbildung 2.2: Membranruhepotential = -70 mV, *Threshold* bei ca. -60 mV), so öffnen sich die aktiven Ionenkanäle, und es kommt zu einer Spannungsspitze von etwa +30 mV bis +40 mV, dem sogenannten Aktionspotential. Aufgrund des zeitlichen Verlaufs dieser Spannungsspitze spricht man auch von einem *Spike*, oder man sagt, ein Neuron *feuert*. Diese Spikes haben je nach Zelltyp einen charakteristischen Zeitverlauf, der jedoch bei einer bestimmten Zelle sehr konstant ist. Allen Spikes gemeinsam ist die kurze Anstiegszeit bis zur Spannungsspitze, sie beträgt ca. 1 ms.

Für die genaue Form der Spikes sind die oben erwähnten Ionen wie $Ca^{+2}$, $Cl^-$ und einige andere verantwortlich. Man weiß zusätzlich, daß die "Schalt"–Eigenschaften aktiver Ionenkanäle von den anderen vorhandenen Ionenarten beeinflußt werden, wodurch auch verschiedene Feuercharakteristika entstehen können (vgl. [6, Kapitel 7],[21]).

Die aktiven Ionenkanäle sind außerdem für ein weiteres Phänomen verantwortlich: das *Refraktärzeitverhalten* eines Neurons. Die $Na^+$ Kanäle befinden sich nämlich in einem der folgenden drei Zustände. Aus einem *geschlossenen, aktivierbaren Zustand* wird nach der Depolarisierung ein *Offenzustand* erreicht, worauf wiederum ein *geschlossener, inaktiver Zustand* folgt (vgl. Abbildung 2.3).

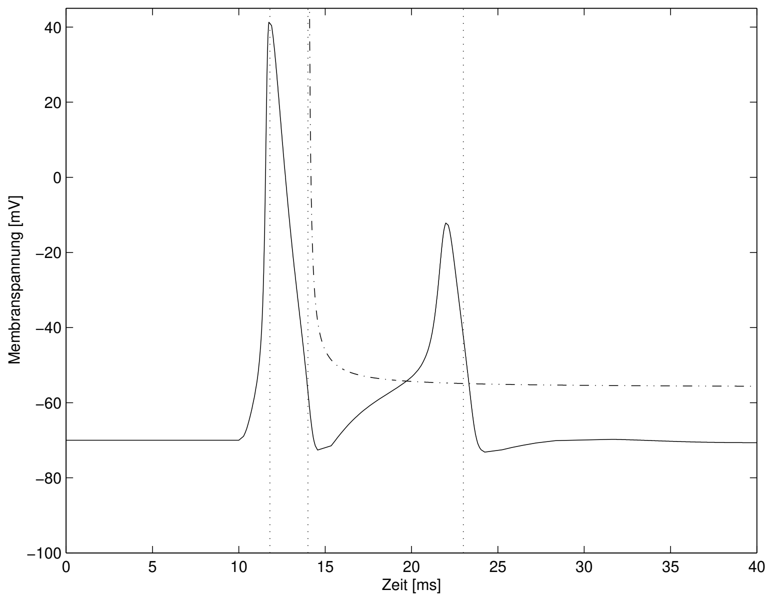



**Abbildung 2.4:** Darstellung des Refraktärzeitverhaltens eines Neurons. Nachdem ein Aktionspotential ausgelöst wurde, ist das Neuron in der Zeit zwischen 11.8 ms und 14 ms absolut refraktär. In der relativen Refraktärzeit sinkt die Schwelle rasch wieder ab und erreicht ihren Normalwert (strichlierte Linie). Bei 15 ms wird das Neuron in gleicher Art wie bei 10 ms nocheinmal erregt. Da die Schwelle zu diesem Zeitpunkt aber höher ist als normal, ist es nicht mehr so leicht, ein Aktionspotential auszulösen. Dieses Bild wurde mit dem Simulationsprogramm GENESIS erstellt (siehe Anhang A).

Während des geschlossenen, inaktiven Zustandes kann das Neuron auch durch eine Depolarisation, die weit über die normale Schwelle hinausgeht, nicht zum Feuern gebracht werden.
Die Refraktärzeit unterteilt sich in eine *absolute Refraktärzeit* und eine *relative Refraktärzeit*. In der absoluten Refraktärzeit kann das Neuron gar nicht zum Feuern gebracht werden, in der relativen Refraktärzeit nur durch Depolarisieren der Membran über den normalen *threshold*. In Abbildung 2.4 ist der *threshold* eines Neurons als zeitabhängige Funktion abgebildet (strichlierte Linie).

### 2.1.2 Fortleitung des Aktionspotentials

Wurde am Axonhügel (Fortsatz des Somas in Richtung Axon) ein Aktionspotential generiert, so übernimmt das Axon jetzt die Aufgabe, dieses Signal zu verteilen. Da das Verhältnis von Membranwiderstand zu Axoninnenwiderstand ungünstig ist (vgl. Abschnitt 2.2), ist die passive Ausbreitung von Potentialänderungen und Strömen in Nervenzellen auf wenige Millimeter begrenzt. Das Axon funktioniert also nicht wie ein Kabel, sondern über einen aktiven Prozeß zur Fortleitung der Aktionspotentiale.

Dieser axonale Prozeß funktioniert wie folgt: tritt an einer Stelle des Axons ein Aktionspotential auf, so breitet sich die Depolarisation an dieser Stelle passiv elektronisch auf benachbarte Membranstellen aus. Diese benachbarten depolarisierten Membranstellen lösen wiederum ein weiteres Aktionspotential aus.

Die Geschwindigkeit der axonalen Fortleitung ist bei verschiedenen Nervenfasertypen sehr unterschiedlich. Hauptsächlich hängt sie vom Durchmesser der Nervenfasern ab und ob diese *myelinisiert* sind. Bei unmyelinisierten Axonen werden Fortleitungsgeschwindigkeiten von weniger als 1 m/s bis zu 20 m/s erreicht (20 m/s werden z. B. beim Axon des Riesentintenfisches erreicht, dessen Durchmesser 1 mm beträgt). Bei myelinisierten Nervenfasern können jedoch Geschwindigkeiten bis zu 120 m/s erreicht werden (Verbindungen zu Skelettmuskeln; dort beträgt der Durchmesser der Nervenfasern ca. 15 $\mu$m). Weitere Details sind z. B. [21], [23] oder [24] zu entnehmen.

### 2.1.3 Synapsen

Neuronen sind also sozusagen die Bausteine des Nervensystems. Untereinander verbunden sind diese Bausteine durch Synapsen. Diese Synapsen sind aber nicht nur Kontaktstellen, sie sind die eigentlichen Rechenelemente des Gehirns. Es gibt verschiedenste Synapsentypen, und die exakte biologische Funktionsweise ist bei allen sehr komplex. Hier soll daher nur der prinzipielle biologische Mechanismus der synaptischen Übertragung vorgestellt werden.

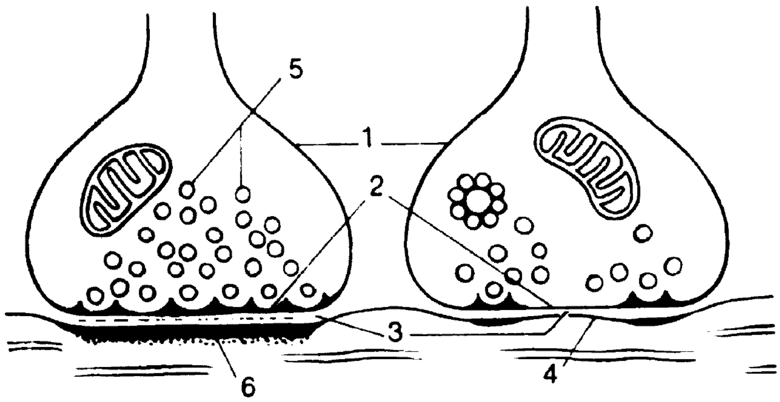



**Abbildung 2.5:** Zwei Synapsen nach *Gray* mit unterschiedlicher Dichte der postsynaptischen Membran. 1 synaptischer Endknopf; 2 präsynaptische Membran; 3 synaptischer Spalt; 4 postsynaptische Membran; 5 Vesikel; 6 Verdichtungszone der postsynaptischen Membran (Rezeptoren). Die Funktionsweise von chemischen Synapsen wird im Text genauer erläutert (Bild aus [10]).

Es gibt 2 Arten von Synapsen: *elektrische* und *chemische.* Elektrische Synapsen spielen eine wichtige Rolle bei Glia–Zellen und zwischen Neuronen, die sich erst entwickeln. Bei ausgewachsenen Säugetieren hingegen sind sie nur noch selten zu finden. Die Erregungsübertragung zwischen Neuronen findet dann hauptsächlich über chemische Synapsen statt.

Der axonale Teil einer chemischen Synapse besteht aus der *synaptischen Endigung*, die eine Art Knopfform hat. Hierin befinden sich zahlreiche *synaptische Bläschen (Vesikel)*, die einen chemischen Überträgerstoff (*Neurotransmitter*) enthalten. Gegenüber des präsynaptischen Endknopfes befindet sich die postsynaptische Membran. Getrennt werden sie durch den *synaptischen Spalt* (vgl. Abbildung 2.5). Ein Aktionspotential am präsynaptischen Endknopf der Synapse depolarisiert dort die Zellmembran. Daraufhin verbinden sich die Vesikel mit der Zellmembran und setzen die enthaltenen Neurotransmitter frei. Diese diffundieren durch den synaptischen Spalt und reagieren mit an der postsynaptischen Membran befindlichen Rezeptoren. Wird ein Rezeptor von einem solchen Botenstoff besetzt, so verändern sich lokal die postsynaptischen Membraneigenschaften. Wiederum gibt es 2 Möglichkeiten: entweder es tritt eine kurze Depolarisation, die als *exzitatorisches postsynaptisches Potential* (EPSP) bezeichnet wird, oder eine Hyperpolarisation, bezeichnet als *inhibitorisches postsynaptisches Potential* (IPSP), auf. Man unterscheidet deshalb auch *exzitatorische Synapsen* und *inhibitorische Synapsen*, wobei erstere eher an den Ästen der Dendriten, und zweitere meist nahe am Soma auftreten. Die Form solcher PSP ist aus Abbildung 2.6 zu entnehmen.

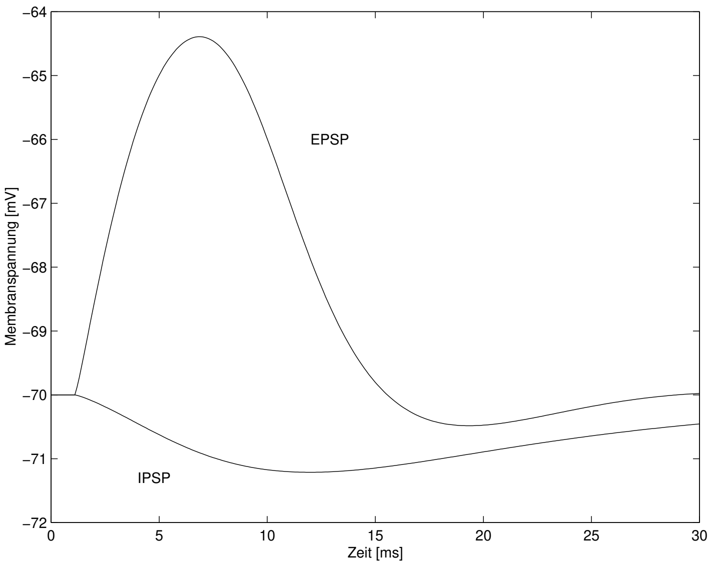



**Abbildung 2.6:** Typischer Zeitverlauf eines exzitatorischen postsynaptischen Potentials (EPSP) und eines inhibitorischen postsynaptischen Potentials (IPSP). Dieses Bild wurde mit dem Simulationsprogramm GENESIS erstellt (siehe Anhang A).

### 2.1.4 Integration postsynaptischer Potentiale

Über den Dendritenbaum verteilt kann eine Nervenzelle einige 1000 Synapsen besitzen. Es stellt sich die Frage, wie diese Inputs von anderen Neuronen zusammenspielen. Geht man von passiven Dendritenbäumen aus (diese enthalten keine aktiven Ionenkanäle), so summieren sich die Ströme, und damit auch die Potentiale der PSP von gleichzeitig aktivierten Synapsen, räumlich auf. Bei weit vom Soma entfernten Synapsen ist jedoch noch zu beachten, daß durch die Länge des Dendriten die PSP einen Größenverlust erleiden (vgl. [13, Kapitel 2]). Auf seinem Weg zum Soma wird das Signal aufgrund der Kabelgleichung (siehe Abschnitt 2.2.2) exponentiell abgeschwächt und sein zeitlicher Verlauf verbreitert. Bei den biologisch realistischeren aktiven Dendritenbäumen gibt es Stellen (vor allem an Verzweigungspunkten des Baumes), an denen Verstärkungsprozesse einsetzen. Treffen genügend viele PSP's ungefähr zur gleichen Zeit an solch einem Punkt ein, so wird das Signal zusätzlich verstärkt.

Prinzipiell tritt die Summenbildung der PSP überall in der Nervenzelle auf. Der wesentliche Summationsort ist jedoch der schon erwähnte Axonhügel, da an dieser Stelle des Neurons die Schwelle für ein Aktionspotential am niedrigsten ist und diese dort ausgelöst werden.

Zum räumlichen Aspekt der Summation kommt natürlich noch der zeitliche Aspekt hinzu. Einerseits treten die EPSP und IPSP nicht alle gleichzeitig an der Nervenzelle auf, und andererseits kommt es auch durch die verschiedenen Längen der Dendritenäste zu Zeitverzögerungen. Ein Dendritenbaum kann daher sehr komplexe, nichtlineare „Rechenoperationen“ ausführen. Auf die Eigenschaft der räumlichen Summation passiver Dendriten baut auch das im Mittelpunkt dieser Arbeit stehende theoretische Konstrukt auf (vgl. Abschnitt 4).

### 2.1.5 Synaptische Plastizität

Glücklicherweise ist der zeitliche Verlauf der PSP von Synapsen nur der groben Form nach vorgegeben. Besonders in den letzten Jahren stellte man fest, daß es viele Möglichkeiten gibt, wie sich die Übertragungseigenschaften von Synapsen verändern können. Man nennt diesen Mechanismus *synaptische Plastizität* und glaubt, daß in ihm die Fähigkeit zu lernen und Information zu speichern begründet liegt.

Im einfachsten Fall ändert sich nur die Effektivität einer Synapse, d. h. die Amplitude des PSP verändert sich. Regeln, denen solche synaptischen Effektivitätsänderungen folgen, nennt man *Lernregeln*. Eine der bekanntesten dieser Lernregeln wurde schon 1949 von *Donald Hebb* als theoretisches Postulat aufgestellt. Sie besagt, daß die Effektivität einer Synapse gesteigert wird, wenn präund postsynaptische Aktivitäten gleichzeitig auftreten. In Experimenten an exzitatorischen Synapsen im Hippocampus konnte diese Theorie später bestätigt werden.

Heute kennt man noch viele andere Eigenschaften von Synapsen, die ein enormes Rechenpotential beinhalten. Es wird z. B. nicht bei jedem präsynaptischen Spike ein PSP ausgelöst, sondern nur mit einer gewissen Wahrscheinlichkeit. Das führt zu einer statistischen Sicht der synaptischen Übertragung (vgl. [9, Kapitel 11]).
Das Phänomen der *synaptischen Bahnung* ist ein anderes Beispiel. Treffen 2 Spikes zeitlich kurz hintereinander ein, so hat das zweite PSP meist eine größere Amplitude als das erste. Erreicht jedoch eine ganze Folge von Spikes (*Spike Train*) die Synapse, nimmt die Amplitude der PSP nach dem 2–4 PSP wieder ab (*synaptische Depression*). Man vermutet, daß die Synapse nicht schnell genug neue Vesikel nachladen kann (vgl. [12, Kapitel 13]).

## 2.2 Das Kompartmentmodell

Die oben dargestellten Eigenschaften von biologischen Neuronen müssen mathematisch modelliert werden, um eine Simulation am Computer zu ermöglichen. Da schon von Membranspannungen, Ionenströmen und Membranwiderstand die Rede war, ist es naheliegend, dafür elektrische Ersatzschaltungen zu verwenden. Aus Abbildung 2.1 ist die große Vielfalt von Zellformen zu erahnen, die entscheidend das Verhalten eines Neurons beeinflußt. Die Berücksichtigung dieser topologischen Eigenschaften von Neuronen führt zum *Kompartmentmodell* (vgl. Kapitel 2, 4, 5 und 6 in [6], [12] oder [13]).

Die hinter dem Kompartmentmodell stehende Idee ist recht einfach. Man unterteilt ein Neuron in lauter kleine Stücke, die *Kompartments*, von denen man annimmt, daß sie auf dem gleichen elektrischen Potential sind. Jedes dieser Teilstücke wird dann durch sein elektrisches Ersatzschaltbild beschrieben. Dann werden die Ersatzschaltbilder wieder verbunden, und so entsteht ein Modell des ganzen Neurons.

### 2.2.1 Die Membrangleichung

Zuerst wird die Membranspannung definiert. Sie ist gleich der Potentialdifferenz zwischen intrazellulärer und extrazellulärer Flüssigkeit und abhängig von der Zeit.

$$u_m(t) = u_i(t) - u_e(t) \tag{2.1}$$

Bei den meisten Kompartmentmodellen wird die Annahme gemacht, daß die gesamte extrazelluläre Flüssigkeit isopotential ist, d. h. es gibt extrazellulär keine Potentialunterschiede. Daher wählt man dieses Potential als Bezugspotential

($u_e(t) = 0$). Die extrazelluläre Flüssigkeit wird in den folgenden elektrischen Ersatzschaltbildern als Erdungssymbol dargestellt.

### Membrankapazität

Die Membran trennt also intrazelluläre und extrazelluläre Flüssigkeit und damit auch die Ladungen $Q$ in diesen Flüssigkeiten. Man kann die Membran also als Kondensator beschreiben. Die Kapazität C der Membran ist ein Maß dafür, wieviel Ladung sich auf beiden Seiten befindet, um eine Spannung von $u_m$ aufbauen zu können. Der Zusammenhang zwischen $u$ und $Q$ ist durch die Gleichung $Q = C_m\, u_m$ gegeben. Differenziert man diese Gleichung nach $t$, so erhält man den Strom $i_C$ durch die Kapazität.

$$i_C(t) = C_m \, \frac{du_m(t)}{dt} \tag{2.2}$$

Üblicherweise wird bei Zellen nicht die Membrankapazität angegeben, sondern die *spezifische Membrankapazität* in $\mu F/cm^2$. Die tatsächliche Kapazität erhält man durch Multiplikation mit der Membranoberfläche.

### Membranwiderstand

Die zweite passive Membrankomponente sind die passiven Kanäle. Diese Kanäle sind immer offen, Ionen können hindurchfließen und verursachen dadurch einen Strom. Dieser Strom wird durch das Umkehrpotential $E_{leak}$ der durch die Kanäle fließenden Ionen getrieben. Mit dem Ohmschen Gesetz ergibt sich:

$$i_{leak}(t) = g_{leak}\,(u_m(t) - E_{leak})\,. \tag{2.3}$$

Der Membranwiderstand $R_{leak} = 1/g_{leak}$ ist also der Widerstand, der sich den Ionen beim Durchqueren der Kanäle entgegenstellt. Auch $R_{leak}$ wird meist als *spezifischer Membranwiderstand* in $\Omega \cdot cm^2$ angegeben. Den wahren Membranwiderstand erhält man, indem man den spezifischen Membranwiderstand durch die Membranoberfläche dividiert.

### Synaptische Kanäle

Als nächstes werden die synaptisch aktivierten Kanäle modelliert. Dazu wird im Ersatzschaltbild ein Zweig mit einem veränderlichen Widerstand bzw. Leitwert in Serie mit einer treibenden elektrischen Kraft $E_{syn}$ eingefügt. Dadurch ergibt sich ein synaptischer Strom von

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

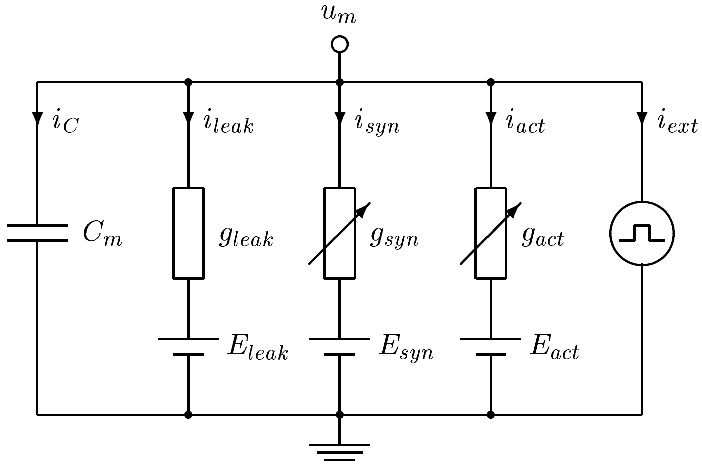



**Abbildung 2.7:** Ersatzschaltbild eines Stückes der Zellmembran. Durch die beiden linken Zweige werden die passiven Eigenschaften der Zellmembran beschrieben. Dies sind die Membrankapazität und der Strom durch die passiven Ionenkanäle. Der mittlere Zweig stellt die synaptischen Kanäle dar und der nächste Zweig die aktiven Kanäle. Mit den veränderlichen Leitwerten $g_{syn}(t)$ und $g_{act}(t, u_m)$ können PSP und Aktionspotentiale modelliert werden. Die Art der beteiligten Ionenkanäle (z. B. $Na^+$, $K^+$, $Ca^{+2}$, $Cl^-$) wird durch die Wahl der Spannungsquellen $E_{syn}$ und $E_{act}$ berücksichtigt. Mit $i_{ext}$ gehen schließlich auch etwaige externe Ströme ins Modell ein.

Der tatsächliche Verlauf der EPSP und IPSP ist durch den Zeitverlauf des veränderlichen Leitwertes bestimmt. Approximieren läßt sich der Verlauf mit der von Rall (1967) erstmals dazu verwendeten *alpha–Funktion*

$$g_{syn}(t) = g_{max} \, \frac{t}{\tau_p} \, e^{\left(1-\frac{t}{\tau_p}\right)} \tag{2.5}$$

Die Funktion steigt in der Zeit $\tau_p$ bis zu einem Maximalwert von $g_{max}$ an und fällt danach wieder auf Null ab. Je nach Wahl von $E_{syn}$ wird eine exzitatorische Synapse ($E_{syn} >$ Membranruhepotential) oder eine inhibitorische Synapse ($E_{syn} <$ Membranruhepotential) modelliert.

### Aktive Kanäle

Der nächste Zweig im Ersatzschaltbild repräsentiert die aktiven Kanäle. Ähnlich wie bei den synaptischen Kanälen gibt es einen veränderlichen Leitwert, der aber nicht nur zeitabhängig, sondern auch spannungsabhängig ist (vgl. Abschnitt 2.3). Durch die Form der Funktion $g_{act}(t, u_m)$ und durch die Spannungsquelle $E_{act}$ können die verschiedenen Arten von aktiven Kanälen modelliert werden. Wie oben ergibt sich der Strom zu

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

**Ersatzschaltbild eines Kompartments**

Ganz rechts in Abbildung 2.7 ist noch eine Stromquelle eingefügt. Durch sie werden extern in die Zelle eingebrachte Ströme berücksichtigt (z. B. durch eine Elektrode bei Versuchen oder Messungen).

Der gesamte Strom durch das Membranstück eines Kompartments ist durch das 1. Kirchhoff'sche Gesetz gegeben. Da verschiedene synaptische und aktive Kanäle in einem Kompartment vorkommen können, wird für diese jeweils ein Summenzeichen geschrieben.

$$\begin{aligned} i_m(t) &= i_C + i_{leak} + i_{syn} + i_{act} + i_{ext} \\ &= C_m \frac{du_m(t)}{dt} + g_{leak}\left(u_m(t) - E_{leak}\right) \\ &\quad + \sum g_{syn}(t)\left(u_m(t) - E_{syn}\right) + \sum g_{act}(t, u_m)\left(u_m(t) - E_{act}\right) \\ &\quad + i_{ext} \end{aligned} \tag{2.7}$$

## 2.2.2 Die Kabelgleichung

Der Dendritenbaum nimmt flächen- und volumsmäßig den größten Teil einer Nervenzelle ein; die meisten Synapsen befinden sich an den Ästen des Baumes und nicht direkt am Soma. Um die kabelartige Struktur der Dendriten zu untersuchen, wurde schon bald die Kabeltheorie verwendet[1]. Ausgehend von einem passiven Dendritenbaum soll nun die lineare Kabeltheorie kurz angeschnitten werden (vgl. [9], [13]).

An jeder Stelle eines zylindrischen Membransegmentes kann der Strom in longitudinale Richtung (entlang der x–Achse) oder durch die passive Membran fließen (siehe Abbildung 2.8). Positiver Strom soll dabei in Richtung größerwerdender $x$ Werte fließen, und $r_a$ ist der spezifische Widerstand der intrazellulären Flüssigkeit in $\Omega/cm$. Für den Strom in longitudinaler Richtung ergibt sich:

$$\frac{1}{r_a}\frac{\partial u_m(x,t)}{\partial x} = -i_a(x,t) \tag{2.8}$$

Die Änderung des Stromes $i_a$ in x-Richtung pro Längeneinheit muß gleich dem Strom $i_m$ durch die Membran pro Längeneinheit sein, also

$$\frac{\partial i_a(x,t)}{\partial x} = -i_m(x,t) = -\left(c_m \frac{\partial u_m(x,t)}{\partial t} + \frac{1}{r_m}\left(u_m(x,t) - E_{leak}\right)\right) \tag{2.9}$$

[1] Begründet wurde die Theorie von Lord Kelvin Mitte des 18. Jahrhunderts. In den vierziger Jahren dieses Jahrhunderts wurde sie dann erstmals von Hodgkin und Rushton auf Axone angewendet.

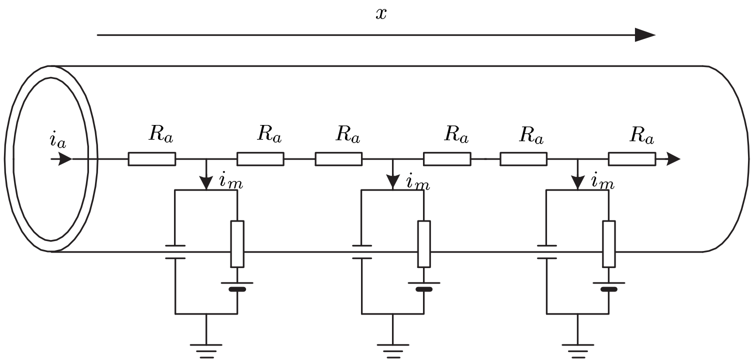



**Abbildung 2.8:** Hier ist schematisch ein zylindrisches Stück einer Nervenzelle dargestellt. Die Widerstände $R_a$ und der Strom $i_a$ spiegeln den Widerstand und den Strom durch die intrazelluläre Flüssigkeit wider. Der passive Ionenstrom durch die Zellmembran wird mit den schon erwähnten R-C-Zweigen berücksichtigt.

$r_m$ und $c_m$ sind dabei spezifischer Membranwiderstand (in $\Omega cm$) und spezifische Membrankapazität (in $F/cm$). Setzt man Gleichung (2.9) in (2.8) ein und multipliziert mit $r_m$, so kommt man zur linearen Kabelgleichung

$$\lambda^2 \frac{\partial^2 u_m(x,t)}{\partial x^2} = \tau_m \frac{\partial u_m(x,t)}{\partial t} + (u_m(x,t) - E_{leak}) \qquad (2.10)$$

mit $\lambda = \sqrt{\frac{r_m}{r_a}}$ und $\tau_m = r_m\, c_m$.

Die Lösung von (2.10) hängt natürlich von Anfangs- und Randbedingungen ab. Im stationären Fall ($\partial u/\partial t = 0$, $u(x)|_{x=0} = u_0$) wird aus der partiellen Differentialgleichung eine gewöhnliche Differentialgleichung der Form

$$\lambda^2 \frac{\partial^2 u(x)}{\partial x^2} - u = 0, \qquad (2.11)$$

deren Lösungen in Abbildung 2.9 dargestellt sind.

### Zusammenschaltung von Kompartments

Mit dem Kompartmentansatz für Dendriten wird die Kabelgleichung durch endliche diskrete Stücke approximiert. Daraus resultiert ein System gewöhnlicher Differentialgleichungen, das typischerweise mit numerischen Methoden gelöst wird (vgl. [6, Kapitel 5], [13, Kapitel 13]). Der Übergang von einem vereinfacht dargestellten biologischen Neuron zu seinem Kompartmentmodell ist in Abbildung 2.10 dargestellt.

Der Vorteil einer Modellierung von Neuronen mit dem Kompartmentansatz liegt darin, daß Kompartments sowohl Stücke von einem Dendritenbaum als auch

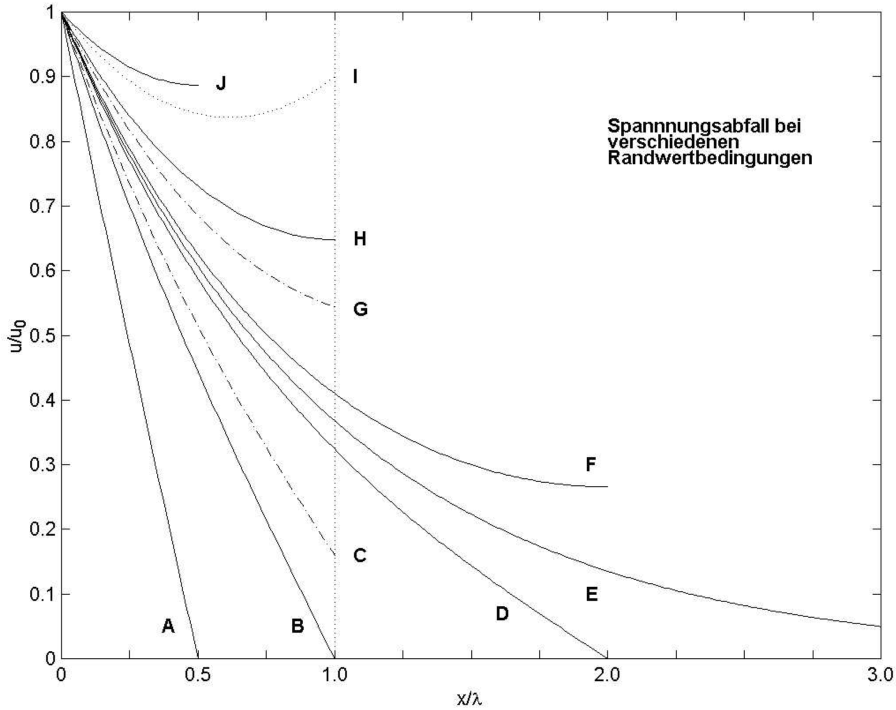
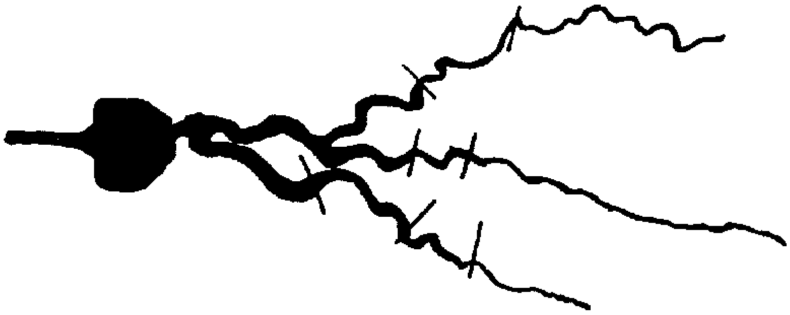
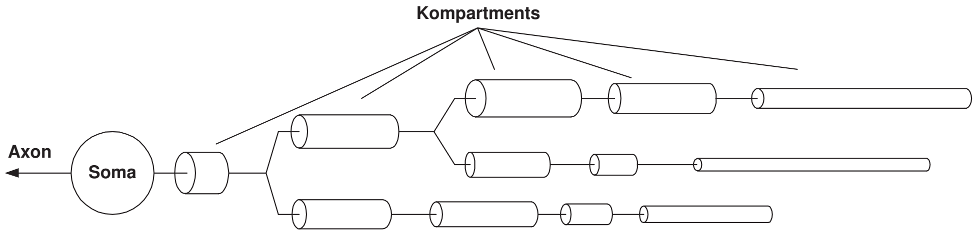


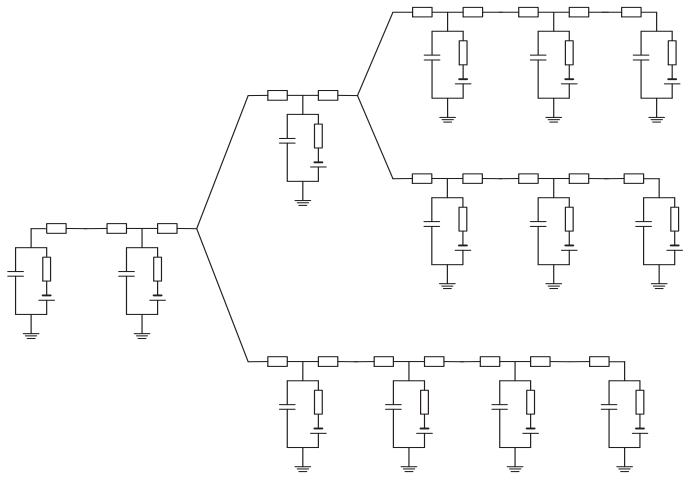

**Abbildung 2.9:** Lösungen der Kabelgleichung als Funktion der dimensionslosen Größe $X = x/\lambda$ für verschiedene Randbedingungen. Die erste Randbedingung $u(X)|_{X=0} = u_0$ ist bei allen Fällen gleich. Kurve **E** zeigt die Lösung im Falle eines halbunendlichen Leiters. Es ergibt sich eine einfache Exponentialfunktion der Form $u(X) = u_0\, e^{-X}$. Für die Kurven **A**, **B** und **D** ist die zweite Randbedingung $u(X) = 0$ für $X = 0.5$, $X = 1.0$ bzw. $X = 2.0$. Die Kurven **F**, **H** und **J** stellen Lösungen bei offenen Leiterenden dar ($\frac{du(X)}{dX} = 0$ für $X = 2.0$, $X = 1.0$ bzw. $X = 0.5$).**C** und **G** zeigen die Lösung für $\frac{du(X)}{dX}|_{X=1} = -4u$ und $\frac{du(X)}{dX}|_{X=1} = -u/4$. Bei Kurve **I** schließlich wird die Spannung am zweiten Leiterende auch auf einem konstanten Wert gehalten ($u(1) = 0.9 \cdot u_0$) (aus [13]).



(a)




(b)



(c)

**Abbildung 2.10:** Bild (a) zeigt ein charakterisiertes reales Neuron mit 3 Dendritenästen. In Bild (b) ist das Neuron bereits in einzelne Kabelstücke mit unterschiedlichen Längen und Durchmessern unterteilt. Im nächsten Schritt werden die Kabelstücke durch die entsprechenden elektrischen Ersatzschaltbilder ersetzt, und man erhält das Kompartmentmodell des Neurons, das in Bild (c) dargestellt ist (aus [6]).

ein Axonstück oder ein Soma darstellen können. Synaptische Aktivitäten lassen sich ebenfalls relativ leicht in das Modell einbinden. Weiters können mit dem Kompartmentansatz topologische Anordnungen im Modell berücksichtigt werden. Strukturen weit verzweigter Dendritenbäume können damit ebenfalls modelliert werden, man muß nur die einzelnen Äste als einzelne Kompartments betrachten. Weiters ist die Genauigkeit, mit der man neuronale Systeme abbildet, in einem großen Bereich wählbar. Mit einem Kompartment kann man eine ganze Zelle modellieren, wenn es darum geht, ganze Netzwerke von Neuronen zu untersuchen. Ebenso kann man durch eine exakte Modellierung einer Zelle mit vielen Kompartments genaue Simulationen einzelner Neuronen durchführen.

## 2.3 Das Hodgkin–Huxley Modell

Die Erregbarkeit von Neuronen beschäftigte in den dreißiger und vierziger Jahren dieses Jahrhunderts viele Physiologen. Einen wesentlichen Beitrag zum heutigen Verständnis von Aktionspotentialen leisteten die beiden Engländer *Hodgkin* und *Huxley*. Sie führten mit damals neuartigen Meßmethoden[2] Experimente an Axonen von Riesentintenfischen durch. Das Motoaxon eines Tintenfisches erreicht bis zu 1 mm Durchmesser, was die Messungen erheblich erleichterte[3]. 1952 veröffentlichten Hodgkin und Huxley 5 Artikel über ihre Experimente und stellten ein allgemeines Modell vor, das die Schlüsseleigenschaften der Zellmembran beschreibt. Mit dem Konzept ihres Modells können jedoch auch Ströme und Leitwerte beschrieben werden, die im Axon des Tintenfisches nicht vorkommen. Für ihre Arbeit erhielten Hodgkin und Huxley 1963 den Nobelpreis für Physiologie und Medizin. Genauere Darstellungen des HH-Modells findet man in [12, Kapitel 6], in [6, Kapitel 4] oder in [9, Kapitel 6].

### 2.3.1 Das Gattermodell

Mit ihren Experimenten gelang es Hodgkin und Huxley die hauptsächlich zur Erzeugung von Aktionspotentialen beitragenden Ionenarten (Na– und K–Ionen ) zu identifizieren. Unter Einsatz von Pharmazeutika war es ihnen möglich, die zu den Ionenarten gehörenden Ströme $I_{Na}$ und $I_K$ zu messen. Das dazugehörige Ersatzschaltbild der Membran, von dem sie ausgingen, ist in Abbildung 2.11 zu sehen.

Um die Zeitabhängigkeit und die Spannungsabhängigkeit der Ionenkanäle berücksichtigen zu können, entwickelten Hodgkin und Huxley das Gattermodell.

[2] *voltage clamping* und *space clamping*.

[3] Typische Axone in der Kortex sind über 1000 mal kleiner.

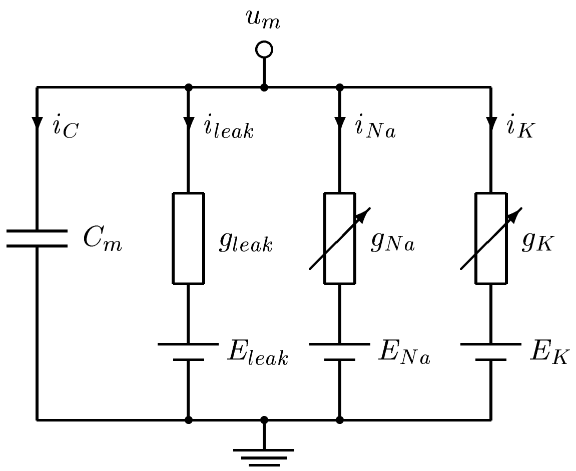



**Abbildung 2.11:** Ersatzschaltbild der Zellmembran nach Hodgkin und Huxley. Die beiden linken Zweige beschreiben wieder die passiven Membraneigenschaften. Durch die Zweige mit den veränderlichen Leitwerten $g_{Na}$ bzw. $g_K$ werden die für die Generierung eines Aktionspotentials zuständigen Ionenarten berücksichtigt.

Die Leitwerte $g_{Na}$ und $g_K$ sind dabei das Resultat der Aktivität vieler mikroskopischer Kanäle. Jeden dieser Kanäle kann man als eine Menge von physikalischen Gattern betrachten, welche den Ionenfluß durch den Kanal steuern. Ein Gatter ist nun mit einer Wahrscheinlichkeit $p$ im durchlässigen Zustand (d. h., daß Ionen hindurchfließen können) bzw. mit Wahrscheinlichkeit $1-p$ im undurchlässigen Zustand (d. h., daß keine Ionen passieren können). Betrachtet man die Summe der Kanäle, so kann man $p$ auch als Bruchteil der Gatter auffassen, die im durchlässigen Zustand sind. Mit den spannungsabhängigen Änderungsparametern $\alpha(u_m)$ und $\beta(u_m)$ werden die Änderungsraten "undurchlässig zu durchlässig" und "durchlässig zu undurchlässig" beschrieben.

$$p \quad \underset{\alpha}{\overset{\beta}{\rightleftharpoons}} \quad 1-p \tag{2.12}$$

Die Änderung zwischen durchlässigem und undurchlässigem Zustand gehorcht einer Dynamik 1. Ordnung:

$$\frac{dp}{dt} = \alpha(u_m)\,(1-p(t)) - \beta(u_m)\,p(t) \tag{2.13}$$

Hält man die Membranspannung $u_m$ konstant auf einem Wert $U_C$, so erhält man daraus einen stationären Wert($dp/dt = 0$) für $t \to \infty$:

$$\begin{aligned} \frac{dp}{dt} = 0 &= \alpha(U_C)\,(1-p_\infty) - \beta(U_C)\,p_\infty \\ p_\infty &= \frac{\alpha(U_C)}{\alpha(U_C)+\beta(U_C)} \end{aligned} \tag{2.14}$$

Mit der Anfangsbedingung $p(t)|_{t=0} = p_0$ und einer Zeitkonstante $\tau(U_C) = 1 \,/\, (\alpha(U_C) + \beta(U_C))$ erhält man als Lösung der Differentialgleichung (2.13) eine Exponentialfunktion der Form

$$p(t) = p_\infty - \left[ (p_\infty - p_0)\, e^{-\frac{t}{\tau}} \right] \tag{2.15}$$

Nimmt man weiters an, daß eine Anzahl $Q$ von unabhängigen Gattern für das Schalten eines Ionenkanals zuständig ist, so hat der Kanal folgendes Zeitverhalten:

$$P(t) = [p(t)]^{\,Q} = \left[ p_\infty - \left[ (p_\infty - p_0)\, e^{-\frac{t}{\tau}} \right] \right]^Q \tag{2.16}$$

### 2.3.2 Bestimmung der Gatterarten

Der Strom in dem von Hodgkin und Huxley verwendeten Ersatzschaltbild ergibt sich wie schon in Abschnitt 2.2 zu

$$\begin{aligned} i_m(t) &= C_m \frac{du_m}{dt} + g_{leak}\,(u_m - E_{leak}) \\ &\quad + g_{Na}(u_m, t)\,(u_m - E_{Na}) + g_K(u_m, t)\,(u_m - E_K) \end{aligned} \tag{2.17}$$

Mit den ermittelten Werten aus ihren zahlreichen Experimenten versuchten sie die Leitwerte $g_{Na}(u_m, t)$ und $g_K(u_m, t)$ analytisch zu beschreiben. Sie schlugen vor, die Leitwerte als Produkt von Gattervariablen $P$ $(0 \leq P \leq 1)$ und dem maximalen Leitwert $g^{max}$ zu schreiben, also

$$g_{Na}(u_m, t) = P_{Na}(u_m, t)\, g_{Na}^{max} \tag{2.18}$$
$$g_K(u_m, t) = P_K(u_m, t)\, g_K^{max} \tag{2.19}$$

Aus den gemessenen Zeitverläufen von $g_{Na}$ und $g_K$ stellten Hodgkin und Huxley fest, daß für den Na–Kanal drei Gattervariablen des Typs $m$ und eine des Typs $h$ und für den K–Kanal vier des Typs $n$ benötigt wurden.

$$g_{Na}(u_m, t) = m^3(u_m, t)\, h(u_m, t)\, g_{Na}^{max} \tag{2.20}$$
$$g_K(u_m, t) = n^4(u_m, t)\, g_K^{max} \tag{2.21}$$

In ihrem Modell gaben Hodgkin und Huxley die Werte von $g_{Na}^{max}$ und $g_K^{max}$ bezogen auf die Membranoberfläche an, und zwar mit $g_{Na}^{max} = 1200\, S/m^2$ und $g_K^{max} = 360\, S/m^2$. Für die Gattervariablen gilt analog zu Punkt 2.3.1

$$\frac{dm}{dt} = \alpha_m\,(1 - m) - \beta_m\, m \qquad m_\infty = \frac{\alpha_m}{\alpha_m + \beta_m} \qquad \tau_m = \frac{1}{\alpha_m + \beta_m} \tag{2.22}$$
$$\frac{dh}{dt} = \alpha_h\,(1 - h) - \beta_h\, h \qquad h_\infty = \frac{\alpha_h}{\alpha_h + \beta_h} \qquad \tau_h = \frac{1}{\alpha_h + \beta_h} \tag{2.23}$$
$$\frac{dn}{dt} = \alpha_n\,(1 - n) - \beta_n\, n \qquad n_\infty = \frac{\alpha_n}{\alpha_n + \beta_n} \qquad \tau_n = \frac{1}{\alpha_n + \beta_n} \tag{2.24}$$

Bei Messungen mit verschiedenen Spannungen $U_C$ wurden dann die Werte für $m_\infty$, $h_\infty$, $n_\infty$, $\tau_m$, $\tau_h$, $\tau_n$ bestimmt und daraus die Werte $\alpha_m$, $\alpha_h$, $\alpha_n$, $\beta_m$, $\beta_h$, $\beta_n$ berechnet. So kamen sie auf die folgenden Gleichungen:

$$\alpha_m(u_m) = A_m^\alpha \, \frac{(u_m - U_m^\alpha)}{e^{\frac{u_m - U_m^\alpha}{B_m^\alpha}} - 1} \qquad \beta_m(u_m) = A_m^\beta \, e^{\frac{u_m - U_m^\beta}{B_m^\beta}} \tag{2.25}$$

$$\alpha_h(u_m) = A_h^\alpha \, e^{\frac{u_m - U_h^\alpha}{B_h^\alpha}} \qquad \beta_h(u_m) = \frac{A_h^\beta}{e^{\frac{u_m - U_h^\beta}{B_h^\beta}} + 1} \tag{2.26}$$

$$\alpha_n(u_m) = A_n^\alpha \, \frac{(u_m - U_n^\alpha)}{e^{\frac{u_m - U_n^\alpha}{B_n^\alpha}} - 1} \qquad \beta_n(u_m) = A_n^\beta \, e^{\frac{u_m - U_n^\beta}{B_n^\beta}} \tag{2.27}$$

Für die in Tabelle 2.1 angegebenen Parameter und bei Annahme eines Ruhepotentials von -70 mV sind die Zeitverläufe für die $\alpha$'s, $\beta$'s in Abbildung 2.12 dargestellt. Abbildung 2.13 zeigt den Verlauf der Gattervariablen bei einer Depolarisation der Membran von -70mV auf 0mV.

| Parameter | $A^\alpha$ | $B^\alpha$ | $U^\alpha$ | $A^\beta$ | $B^\beta$ | $U^\beta$ |
|---|---|---|---|---|---|---|
| Gatter m | $-10^5$ | $-0.01$ | $0.025 + U_R$ | 4000 | $-0.018$ | $U_R$ |
| Gatter h | 70 | $-0.02$ | $U_R$ | 1000 | $-0.01$ | $0.03 + U_R$ |
| Gatter n | $-10^4$ | $-0.01$ | $0.01 + U_R$ | 125 | $-0.08$ | $U_R$ |

**Tabelle 2.1:** Parameter der Gleichungen (2.25), (2.26) und (2.27). $U_R$ ist dabei das Membranruhepotential bezogen auf die extrazelluläre Flüssigkeit. Bei den Kurven in Abbildung 2.12 wurde das Ruhepotential mit -70mV angenommen ($U_R = -0.07$).

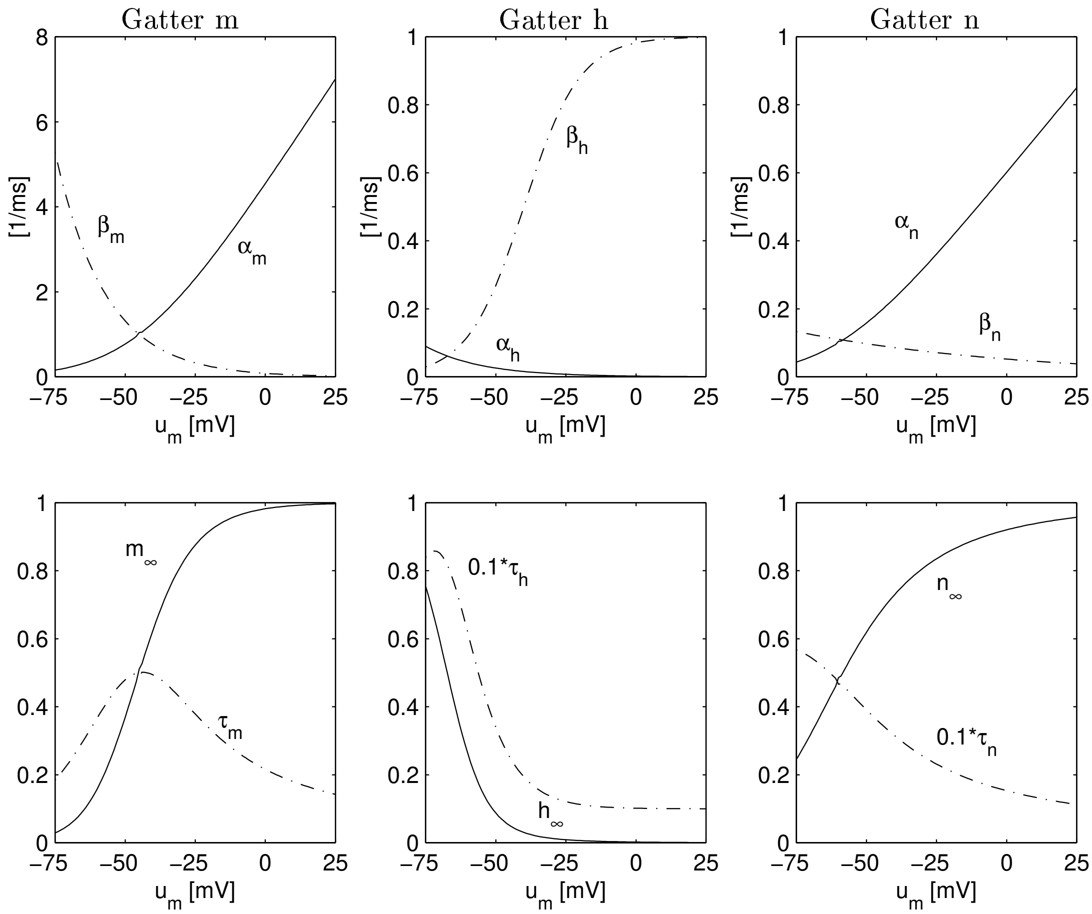



**Abbildung 2.12:** Aktivierungsparameter des Hodgkin–Huxley–Modell. Die oberen Diagramme zeigen den Verlauf der Parameter $\alpha_i$ (durchgezogen) und $\beta_i$ (strichpunktiert) in Abhängigkeit von der Membranspannung $u_m$. In den unteren Diagrammen ist der Verlauf der Parameter $m_\infty$, $h_\infty$, $n_\infty$ (durchgezogen, dimensionslos) und der $\tau_i$ (strichpunktiert, in ms) zu sehen.

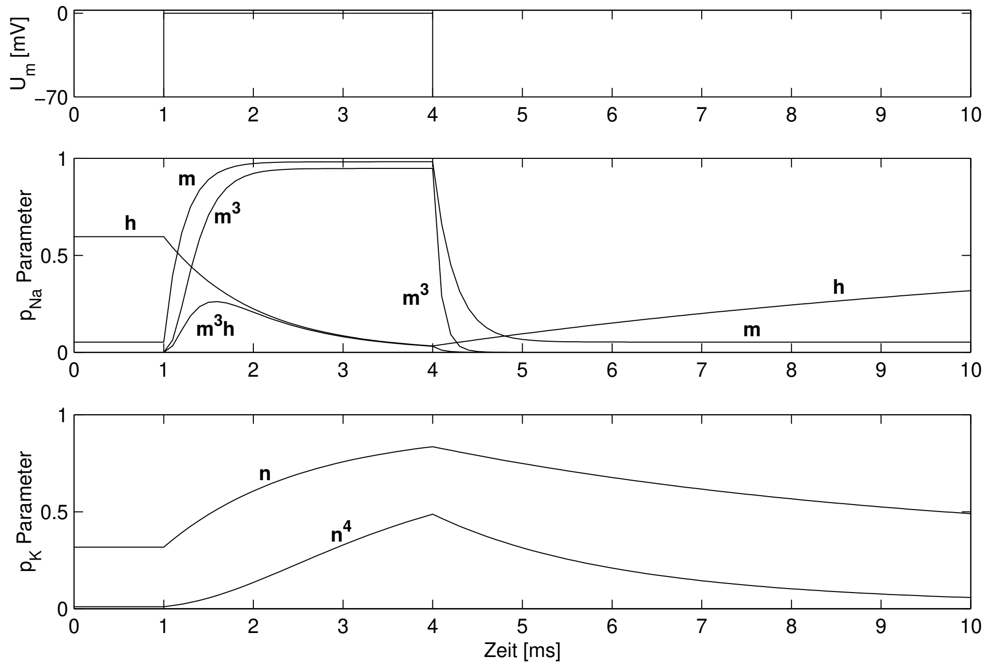



**Abbildung 2.13:** Zeitverläufe von $m$, $m^3$, $h$, $m^3h$, $n$ und $n^4$ bei einer 3 ms andauernden Depolarisation der Membran von -70 mV auf 0 mV. $m$ und $n$ folgen dabei einem $(1 - e^{-t/\tau})$ Zeitverlauf (die Gattervariablen $m$ und $n$ werden also durch Depolarisation aktiviert), während $h$ einem $e^{-t/\tau}$ Zeitverlauf folgt (d. h. Inaktivierung durch Depolarisation). Der unterschiedliche Zeitverlauf resultiert aus den verschiedenen Randbedingungen ($h_0 < h_\infty$, $m_\infty < m_0$, $n_\infty < n_0$).

# Kapitel 3

# Netzwerke und Lernalgorithmen

Im vorangegangenen Kapitel wurden der Aufbau und die grobe Funktionsweise von biologischen Neuronen vorgestellt. Außerdem wurde eine Möglichkeit, diese Neuronen mathematisch zu beschreiben, erläutert. Dieses Kapitel behandelt nun die theoretischen Grundbegriffe von künstlichen neuronalen Netzen und Lernregeln, die in solchen Netzen angewendet werden können.

## 3.1 Allgemeines über Netzwerke

### 3.1.1 Netzwerk Architektur

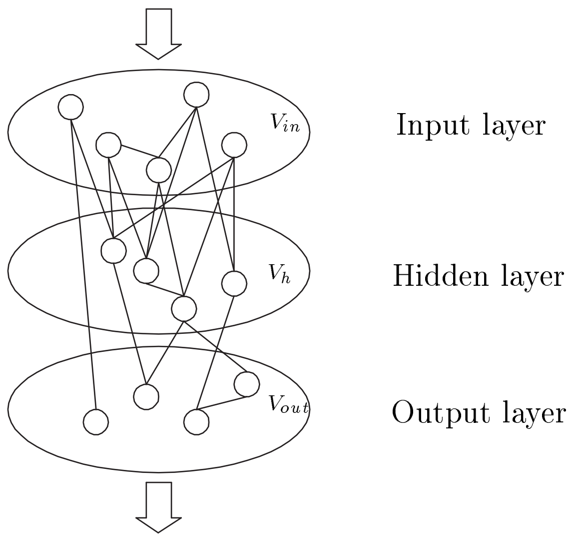



**Abbildung 3.1:** Schematische Darstellung einer Netzwerke Topologie (aus [5]).

Definition eines *neuronalen Netzwerkes* (NN) $\mathcal{N}$:

Ein NN $\mathcal{N}$ besteht aus

- einem gerichteten Graphen $G = \langle V, E \rangle$. Die Knoten $V$ des Graphen werden dabei auch als Gatter bezeichnet und die Kanten $E \subseteq V \times V$ als Verbindungen zwischen den Gattern.
- einer Teilmenge $V_{in} = \{in_1, \ldots, in_n\} \subseteq V$ von $n$ *Input–Knoten*
- einer Teilmenge $V_{out} = \{out_1, \ldots, out_k\} \subseteq V$ von $k$ *Output–Knoten*
- einer Teilmenge $V_h \subseteq V$ von *Hidden–Knoten*, sodaß gilt: $V_{in} \cup V_h \cup V_{out} = V$ und $(V_{in} \cup V_{out}) \cap V_h = \emptyset$
- einer Aktivierungsfunktion $\varphi_i : \mathbb{R} \rightarrow W \subseteq \mathbb{R}$ für jeden Knoten $i \in V \setminus V_{in}$
- einer Gewichtsfunktion[1] $w_{j,i} : E \rightarrow \mathbb{R}$ für jede Kante $\langle i, j \rangle \in E$
- und einem *Bias* $b_j \in \mathbb{R}$ für jeden Knoten $i \in V \setminus V_{in}$

Mit dieser Definition lassen sich nun Grundtypen von Netzwerktopologien beschreiben (vgl. [5]):

- *2-Schicht Netzwerk (perceptron):* Das Netzwerk besitzt nur Input– und Output–Knoten ($V_h = \emptyset$) und Verbindungen $E \subseteq (V_{in} \times V_{out})$.
- *Mehrschichtiges Netzwerk (multilayer perceptron, feed–forward Netzwerk):* Die Hidden–Knoten sind in einzelnen Schichten (*layers*) angeordnet, d. h. $V_h = V_{h_1} \cup \ldots \cup V_{h_{l-1}}$ und $V_{h_i} \cap V_{h_j} = \emptyset$ für $i \neq j$. Verbindungen existieren nur zwischen adiazenten Schichten, also $E \supseteq (V_{in} \times V_{h_1}) \cup (V_{h_1} \times V_{h_2}) \cup \ldots \cup (V_{h_{l-1}} \times V_{out})$. Wenn $E = (V_{in} \times V_{h_1}) \cup (V_{h_1} \times V_{h_2}) \cup \ldots \cup (V_{h_{l-1}} \times V_{out})$ gilt, so spricht man von einem voll verbundenen *feed–forward* Netzwerk.
- *Voll vernetztes Netzwerk (z. B. Hopfield Netzwerk):* Zwischen allen Knoten im Netz existieren Verbindungen, d. h . $E = V \times V$ mit symmetrischen Gewichten $w_{i,j} = w_{j,i}$. Beispiel: Hopfield Netzwerke können zum Speichern von vorgegebenen Input–Mustern verwendet werden.
- *Rekurrentes Netzwerk:* Der Graph eines solchen Netzwerkes enthält Kreise (d. h. es gibt mindestens einen Weg in $G$ mit $\{\langle i_1, i_2 \rangle, \langle i_2, i_3 \rangle, \ldots, \langle i_n, i_1 \rangle\} \subseteq E$) und somit Rückkopplungen.

In dieser Arbeit werden nur 2–Schicht Netzwerke und *feed–forward* Netzwerke betrachtet.

[1] Achtung: Die Gewichte werden in einigen Büchern mit $w_{i,j}$ für eine Kante von Knoten i zu Knoten j notiert.

### 3.1.2 Gatter

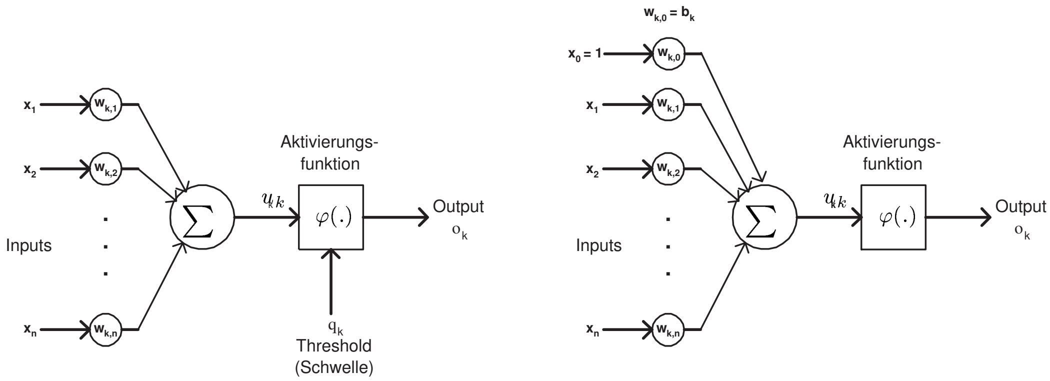



**Abbildung 3.2:** Nichtlineare Modelle eines Gatters. Links ist ein Modell mit externem *threshold* abgebildet. Rechts ist ein äquivalentes Modell des linken Gatters zu sehen. Der *threshold* wurde durch einen zusätzlichen Input und einem dazugehörigen Biaswert ersetzt (aus [8]).

Die Grundbausteine eines neuronalen Netzes sind Gatter. Sie führen sozusagen die Berechnungen eines Netzes aus. Abbildung 3.2 zeigt ein detailiertes Schema eines Gatters. Man kann ein Gatter $k$ in 3 Basiselemente aufgliedern, und zwar in:

1. eine Menge von Input–Signalen mit den assoziierten Gewichten. Jedes Input–Signal $x_i$ wird mit dem dazugehörigen Gewicht $w_{k,i}$ multipliziert.

2. ein Summenpunkt $u_k$, an dem die Linearkombination der gewichteten Inputs gebildet wird.

3. einer Aktivierungsfunktion $\varphi$, die die Amplitude des Outputs eines Gatters limitiert.

In Abbildung 3.2 links ist zusätzlich noch ein externer *threshold* $\theta_k$ vorhanden. Mathematisch kann man den Output eines Gatters k wie folgt beschreiben.

$$o_k = \varphi(u_k - \theta_k) = \varphi\left(\sum_{i=1}^{n} w_{k,i}\, x_i - \theta_k\right) \tag{3.1}$$

Damit man nicht bei jedem Gatter den *threshold* $\theta$ extra mitschreiben muß, wird meist eine mathematisch äquivalente Notation verwendet. Dabei wird ein zusätzlicher fixer Input–Wert $x_0 = +1$ und ein Gewicht $w_{k,0} = b_k$ eingeführt. Das

Gewicht $b_k = -\theta_k$ wird auch als *Bias* bezeichnet. Die Formel (3.1) kann dann umgeschrieben werden zu

$$o_k = \varphi\left(\sum_{i=1}^{n} w_{k,i}\, x_i + w_{k,0}\right) = \varphi\left(\sum_{i=0}^{n} w_{k,i}\, x_i\right) \tag{3.2}$$

### 3.1.3 Aktivierungsfunktionen

Die Aktivierungsfunktion $\varphi(.)$ definiert den Output–Wert $o_k$ eines Gatters in Abhängigkeit seiner gewichteten Input–Summe oder Aktivierung. Die am häufigsten verwendete Aktivierungsfunktion ist sicherlich die *sigmoide Funktion*(siehe Abbildung 3.3 (c)), die definiert ist als

$$\varphi(u) = \frac{1}{1 + e^{-a \cdot u}} \tag{3.3}$$

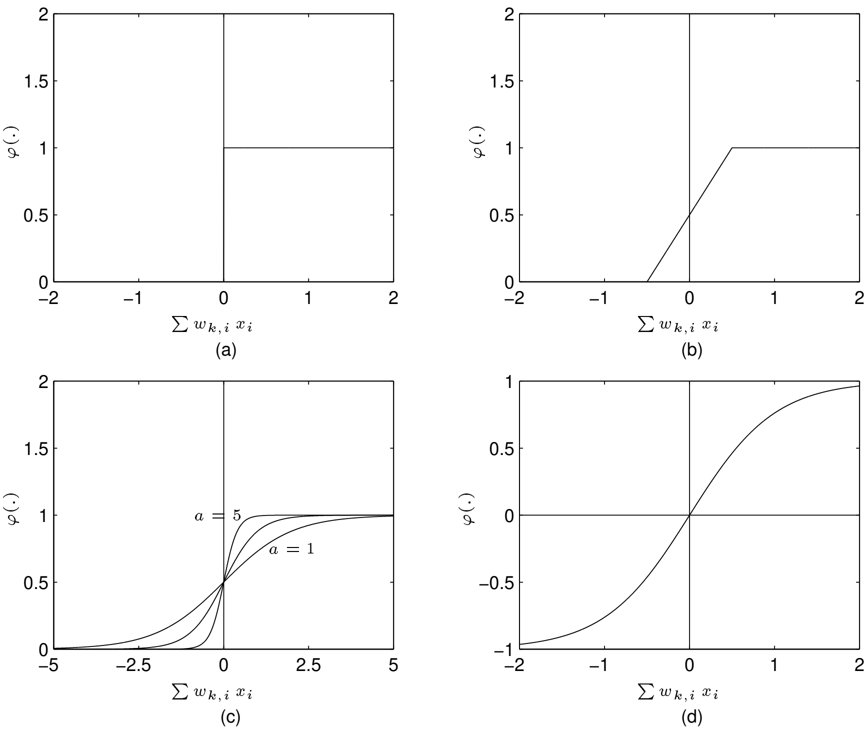



**Abbildung 3.3:** Funktionsgraphen einiger typischen Aktivierungsfunktionen $\varphi(.)$ in Abhängigkeit von der gewichteten Input–Summe $\sum_{i=0}^{n} w_{k,i}\, x_i$ eines Gatters. (a) *threshold* Funktion, (b) Stückweise lineare Funktion, (c) Sigmoide Funktion mit Parametern $a = 1$, $a = 2$ und $a = 5$, (d) Tangens Hyperbolicus.

Für *gradient–descent* Lernverfahren, wie den *Backpropagation Lernalgorithmus* (siehe Abschnitt 3.2.1), wird gefordert, daß $\varphi(.)$ überall differenzierbar ist. In der Praxis hat sich aber gezeigt, daß die exakte Form der Aktivierungsfunktion nicht entscheidend für die Berechnungskraft und die Lerneigenschaften eines NN ist, solange sie monoton steigend, fast überall differenzierbar und beschränkt (also $\lim_{u \to -\infty} \varphi(u)$ und $\lim_{u \to \infty} \varphi(u)$ haben endliche Werte) ist.

## 3.2 Lernverfahren

Die Grundelemente und der Aufbau eines NN sind nun bekannt, aber was kann solch ein Netzwerk? Ein Netzwerk erhält einen Input und liefert nach der Berechnung einen Output. Input– und Outputwerte werden als Vektoren geschrieben. Ein Input für das Netz ist ein Vektor $\vec{x} = (x_1, x_2, \ldots, x_n) \subseteq \mathbb{R}^n$, wobei $x_i$ der Input des Gatters $in_i \subseteq V_{in}$ ist. Der Output des Netzes ist ein Vektor $\vec{y} = (y_1, y_2, \ldots, y_k) \subseteq \mathbb{R}^k$. $y_i$ ist dabei der Output des Gatters $out_i \subseteq V_{out}$. Die Information an den Input– und Output–Gattern wird auch als *Representation* bezeichnet.

Ein NN erhält also Inputs und berechnet mit diesen Informationen einen Output. Das gleiche Prinzip findet man auch bei Computerprogrammen. Der Unterschied zwischen Computerprogrammen und NN ist nun folgender:
Das Computerprogramm hat einen fix implementierten Programmcode, der von jemandem erstellt werden mußte. Auf die selben Inputwerte wird das Programm immer gleich reagieren und die selben Ergebnisse liefern. Bei einem NN $\mathcal{N}$ gibt es kein Programm im obigen Sinne. Die Berechnung des Netzes ist durch die Gewichtswerte $w_{j,i}$ und die Biaswerte $b_j$ der Gatter in $\mathcal{N}$ festgelegt. Manchmal wird ein Vektor $\underline{w}$ eingeführt, der alle Gewichte und Biaswerte in $\mathcal{N}$ beinhaltet. Für das NN $\mathcal{N}$ schreibt man dann auch $\mathcal{N}_{\underline{w}}$, wobei $\mathcal{N}_{\underline{w}}$ eine Funktion $\vec{x} \mapsto \mathcal{N}_{\underline{w}}(\vec{x})$ von $\mathbb{R}^n$ in $\mathbb{R}^k$ definiert. Diesen Vektor $\underline{w}$ könnte man sozusagen als „Programm von $\mathcal{N}$“ bezeichnen. Weiters wird ein NN auch nicht „programmiert“, sondern man läßt es lernen. Das bedeutet, das Netz erhält in seiner Lernphase eine Liste $L$ von Inputvektoren und kann seinen Gewichtsvektor $\underline{w}$ aufgrund einer vorgegebenen Lernregel selbständig verändern und sich dadurch an diese Inputs anpassen.

Man unterscheidet zwei Grundtypen von Lernverfahren:

- *Überwachtes Lernen*

  und

- *Unüberwachtes Lernen*

Beim überwachten Lernen (*supervised learning*) wird dem NN zu jedem Input $\vec{x}$ ein Output $\vec{t}$ vorgegeben. Das Netz besitzt für den Lernvorgang sozusagen einen Lehrer, der immer die richtigen Output–Werte liefert. Aufgrund dieser vorgegebenen Soll–Output–Werte (auch Ziele oder *targets* genannnt) kann ein Fehler zum tatsächlichen Output bestimmt werden. Diese Fehler bilden die Grundlage für die Gewichtsänderungen im Netz.

Wenn dem NN hingegen keine Information über den Output–Wert gegeben wird, dann handelt es sich um unüberwachtes Lernen (*unsupervised learning*). In diesem Fall muß das Netz selbständig Muster, Regelmäßigkeiten, Korrelationen oder Cluster in den Inputdaten erkennen.

### 3.2.1 Der Backpropagation Lernalgorithmus

Der *Backpropagation Algorithmus* oder genauer *error Backpropagation Algorithmus* zählt zu den supervised Lernverfahren[2]. Zur Anwendung kommt dieser Algorithmus in mehrschichtigen *feed–forward* Netzwerken oder auch *multilayer perceptron* (MLP) genannt (vgl. Abschnitt 3.1.1).

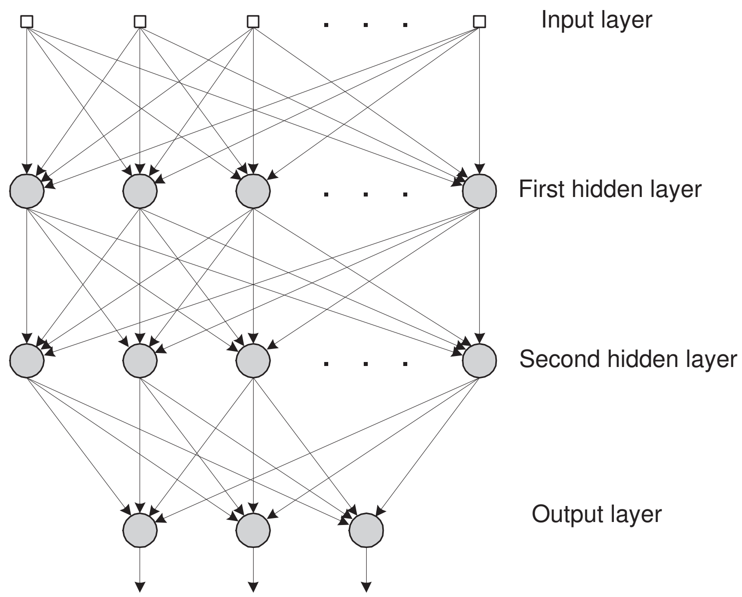



**Abbildung 3.4:** Graph eines *multilayer perceptron* mit zwei *hidden layers*.

Die Grundidee des Backpropagation Algorithmus geht auf eine Doktorarbeit von Werbos aus dem Jahr 1974 zurück [27]. Der Begriff „Backpropagation“ taucht aber erst 1985 in der Literatur auf. Von Rumelhart, Hinton und Williams wurde die Idee von Werbos wieder aufgegriffen [20]. Populär wurde der Algorithmus schließlich durch eine Veröffentlichung von Rumelhart und McClelland 1986. Die

[2]In der Literatur wird der Backpropagation Algorithmus auch oft nur kurz mit *Backprop* bezeichnet.

Entwicklung des *Backprop* Lernalgorithmus war ein Durchbruch für das Gebiet der neuronalen Netze, da es erstmals eine effiziente Methode gab, um mehrschichtige *feed–forward* Netze zu trainieren. Zum Thema Backpropagation siehe z. B. [8, Kapitel 6], [17, Kapitel 7], [7, Kapitel 11] oder [4, Kapitel 4].

MLPs sind in Schichten oder *layers* angeordnet. Verbindungen existieren nur zwischen aufeinanderfolgenden Schichten. Von jedem Gatter in einer Schicht gibt es Verbindungen zu allen Gattern in der nächsten Schicht. Abbildung 3.4 zeigt den Graphen eines *multilayer perceptron* mit zwei *hidden layers*. Der Signalfluß im Netzwerk in Vorwärtsrichtung verläuft von oben nach unten.

Bevor man mit dem Lernen bzw. mit dem Trainieren des NN beginnen kann, benötigt man eine Menge von sogenannten Trainingsbeispielen (*trainings pattern*). D. h. man hat eine Liste $L = \langle \vec{x}_k, \vec{t}_k \rangle_{k=1,\dots,i}$ von i vorgegebenen Beispielen aus $\mathbb{R}^n \times \mathbb{R}^k$. $\vec{x}_k$ ist dabei ein Inputvektor und $\vec{t}_k$ der zugehörige Soll–Outputvektor (*target*). Für jedes Element in der Liste $L$ kann der *error Backpropagation* Lernprozeß nun in 2 Schritte gegliedert werden (siehe Abbildung 3.5):

**Forward Pass:** Aus dem Inputvektor $\vec{x}_k$ wird aufgrund der aktuellen Gewichte der Outputvektor $\vec{y}_k$ berechnet. Die Berechnung erfolgt dabei schichtweise. Da die Gatter einer Schicht voneinander unabhängig sind, kann die Berechnung der Gatteroutputs pro Schicht parallel erfolgen.

**Backward Pass:** Aus der Differenz zwischen dem Outputvektor $\vec{y}_k$ und dem Targetvektor $\vec{t}_k$ kann ein Fehlerwert ermittelt werden. Dieser Fehler wird nun aufgrund der Lernregel rückwärts durch das Netz (wieder schichtweise) propagiert (daher auch der Name Backpropagation). Jedem Gatter wird dabei ein Fehlerwert zugewiesen.

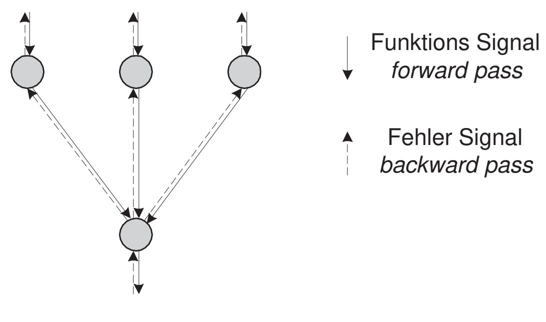



**Abbildung 3.5:** Dieser Ausschnitt aus einem MPL soll den Signalfluß beim Lernvorgang mit dem *backprop* Algorithmus verdeutlichen.

### 3.2.2 Herleitung des Backpropagation Algorithmus

Ziel des Lernens ist es, eine Fehlerfunktion $E_{\underline{w}}(L)$ zu minimieren. Man hofft, daß das Netz dann für ähnliche Beispiele auch die richtigen Outputwerte liefert. Der Algorithmus verändert den Gewichtsvektor $\underline{w}$ mittels *gradient descent* bezüglich $E_{\underline{w}}(L)$. $\underline{w}$ wird also mit einer gewissen Schrittweite $\eta$ (= Lernrate) in Richtung $-\frac{\partial}{\partial \underline{w}} E_{\underline{w}}(L)$, d. h. in Richtung des steilsten Abstiegs der von $E$ gebildeten Fehlerfläche, bewegt.

Voraussetzungen für die folgende Herleitung sind nur ein beliebiges *feed–forward* Netzwerk, dessen Gatter differenzierbare, nichtlineare Aktivierungsfunktionen besitzen, und eine beliebige differenzierbare Fehlerfunkiton $E_{\underline{w}}$. Jedes Gatter $j \in V \backslash V_{in}$ im Netz berechnet eine gewichtete Inputsumme der Form

$$u_j = \sum_{i=0}^{n} w_{j,i}\, o_i. \tag{3.4}$$

Bei allen Gattern $j$ in $\mathcal{N} \in V \backslash V_{in}$ wird ab jetzt vorausgesetzt, daß ein zusätzlicher Input $x_0 = +1$ mit zugehörigem Biaswert $b_j = w_{j,0}$ vorhanden ist und diese daher nicht mehr extra erwähnt werden müssen (vgl. Abschnitt 3.1.2). Die Summe in (3.4) wird mittels einer nichtlinearen Aktivierungsfunktion transformiert, und man erhält den Output eines Gatters $j$

$$o_j = \varphi(u_j). \tag{3.5}$$

Gatter in der ersten *hidden layer* haben als Inputs $o_i$ im Summenterm natürlich die Inputs des Netzes $x_i$. In der Outputschicht werden die $o_j$ mit $y_j$ bezeichnet. Für ein Trainingsbeispiel $\langle \vec{x}, \vec{y} \rangle$ sollen nun die Berechnungen der Gatteroutputs in allen Schichten des Netzes abgeschlossen sein (= *forward pass*). $E_{\underline{w}} = E_{\underline{w}}(\langle \vec{x}, \vec{y} \rangle)$ hängt nur über die Inputsumme $u_j$ des Gatters $j$ von $w_{j,i}$ ab. Deswegen kann für die Berechnung der partiellen Ableitung $\frac{\partial}{\partial w_{j,i}} E_{\underline{w}}$ die Kettenregel verwendet werden. Es gilt also

$$\frac{\partial E_{\underline{w}}}{\partial w_{j,i}} = \frac{\partial E_{\underline{w}}}{\partial u_j} \cdot \frac{\partial u_j}{\partial w_{j,i}}. \tag{3.6}$$

Für den 1. Term wird eine neue Variable

$$\delta_j := \frac{\partial E_{\underline{w}}}{\partial u_j} \tag{3.7}$$

eingeführt, wobei $\delta_j$ als Fehlerzuweisung zu Gatter $j$ angesehen werden kann. Der 2. Term läßt sich leicht aus (3.4) berechnen:

$$\frac{\partial u_j}{\partial w_{j,i}} = o_i. \tag{3.8}$$

Aus (3.6) wird mit (3.7) und (3.8)

$$\frac{\partial E_{\underline{w}}}{\partial w_{j,i}} = \delta_j \, o_i. \tag{3.9}$$

Zum Schluß bleibt noch die Berechnung der $\delta_j$ übrig. Dazu werden 2 Fälle unterschieden. Für alle Outputgatter $k$ erhält man $\delta_k$ wiederum durch Anwendung der Kettenregel und mit (3.5)

$$\delta_k = \frac{\partial E_{\underline{w}}}{\partial u_k} = \frac{\partial E_{\underline{w}}}{\partial y_k} \cdot \frac{\partial y_k}{\partial u_k} = \frac{\partial E_{\underline{w}}}{\partial y_k} \cdot \varphi'(u_k). \tag{3.10}$$

Für alle Gatter $j$ in den h*hidden layers* muß nochmals die Kettenregel angewendet werden. Man erhält

$$\delta_j = \frac{\partial E_{\underline{w}}}{\partial u_j} = \sum_k \frac{\partial E_{\underline{w}}}{\partial u_k} \cdot \frac{\partial u_k}{\partial a_j}, \tag{3.11}$$

wobei die Summe über alle Gatter $k$ läuft, für die Gatter $j$ ein Inputgatter ist. Setzt man die Definition (3.7) in (3.11) ein, und benützt man die Gleichungen (3.4) und (3.5), so wird $\delta_j$ zu

$$\delta_j = \varphi'(u_j) \cdot \sum_k w_{k,j} \, \delta_k. \tag{3.12}$$

Aus (3.12) geht hervor, daß die $\delta$'s Schicht für Schicht im Netzwerk zurückpropagiert werden können, da nur die $\delta$'s aus der nächsten Schicht bekannt sein müssen. Zusammengefaßt sind die einzelnen Schritte des Backpropagation Algorithmus folgende:

1. Lege einen Inputvektor $\vec{x}$ an und berechne alle Gatteroutputs in den *hidden layers* und im Outputlayer mit (3.4) und (3.5).

2. Berechne die $\delta$'s in der Outputschicht mit (3.10).

3. Berechne rückwärts Schicht für Schicht die $\delta$'s in den *hidden layers* mit (3.12).

4. Ermittle die Gewichtsänderung $\Delta\, w_{j,i} := -\eta \, \frac{\partial E_{\underline{w}}}{\partial w_{j,i}} = -\eta \, \delta_j \, o_i$ (Gleichung (3.9)).

Am häufigsten werden sicherlich sigmoide *feed-forward* Netzwerke (Gatter haben sigmoide Aktivierungsfunktionen) verwendet. Die Ableitung der sigmoiden Funktion (3.3) ergibt

$$\varphi'(u) = \varphi(u)(1 - \varphi(u)). \tag{3.13}$$

Wenn als Fehlerfunktion eine quadratische Fehlersumme

$$E_{\underline{w}}(\langle \vec{x}, \vec{t} \rangle) = \frac{1}{2} \sum_{i=1}^{k} (y_k - t_k)^2 \tag{3.14}$$

verwendet wird, so erhält man aus den obigen allgemeinen Formeln die 3 Regeln des *backprop* Algorithmus für diesen speziellen Fall (bei Repräsentation des n–ten Trainingsbeispiels):

(a) Gewichtsänderung:

$$w_{j,i}^{n+1} = w_{j,i}^{n} + \Delta w_{j,i} = w_{j,i}^{n} - \eta\, \delta_j\, o_i \tag{3.15}$$

(b) Für alle Outputgatter $k$

$$\delta_k = y_k^n\, (1 - y_k^n)\, (t_k - y_k) \tag{3.16}$$

(c) Für alle anderen Gatter $j \in V_h$

$$\delta_j = y_k^n\, (1 - y_k^n) \sum_{k:\langle j,k \rangle \in E} w_{k,j}\, \delta_k \tag{3.17}$$

### 3.2.3 Anwendung und Varianten des Backpropagation Algorithmus

Bevor ein Problem mit einem *feed–forward* Netzwerk und der Backpropagation Lernregel gelöst werden kann, müssen noch einige Fragen geklärt werden. Hier sollen nur kurz ein paar Probleme, die dabei auftreten können, erwähnt werden. Für detailliertere Ausführungen hierzu siehe z.B. [8, Kapitel 6] oder [17].

- Wahl der Netzwerkarchitektur $\mathcal{N}$:

  Die Zahl der Input– und Outputgatter ist durch das gegebene Problem bzw. durch die gegebenen Trainingsbeispiele vorgegeben. Die Anzahl der Schichten und auch die Zahl der Gatter in den verborgenen Schichten ist aber frei wählbar. Wird die Architektur zu klein gewählt (d. h. zu wenig Gatter), so kann es sein, daß kein geeignetes $\underline{w}$ existiert, um die Fehlerfunktion $E_{\underline{w}}(L)$ klein zu machen. Diesen Fall bezeichnet man auch als *underfitting*. Der Nachteil einer zu groß gewählten Architektur besteht im sogenannten *overfitting*. Dabei kann sich das Netzwerk „zu gut“ an die Trainingsbeispiele anpassen, und es ist nicht in der Lage bei Eingabe von nicht trainierten Beispielen zu verallgemeinern.

- Initialisation der Gewichte in $\mathcal{N}$:

  Der Gewichtsvektor $\underline{w}_I$ der initialen Gewichte sollte so gewählt sein, daß $E_{\underline{w}_I}(L)$ sensitiv auf Änderungen einer Komponente von $\underline{w}_I$ reagiert. Wenn man die Lernregeln (3.10) - (3.12) betrachtet, so ist es sinnvoll, daß $\varphi'(u)$ maximal wird. Dies ist bei sigmoider Aktivierungsfunktion der Fall, wenn $u = 0$ ist. Bei der Initialisierung der Gewichte werden daher meist gleichverteilte Werte in einem „kleinen Bereich" um 0 verwendet.

- Welche Variante des Backpropagation Verfahrens wird verwendet:

  In vielen Fällen konvergiert das konventionelle Backpropagation Verfahren nur langsam zu einem Minimum der Fehlerfunktion. Die sigmoide Aktivierungsfunktion hat flache Täler in der Fehlerfunktion zur Folge. Dadurch ergeben sich kleine Schrittgrößen und daher eine große Anzahl von Iterationen des Algorithmus. Es sind viele Varianten von Backpropagation vorgeschlagen worden, die die Konvergenzgeschwindigkeit erhöhen, z. B.:

  - *Backpropagation mit variabler Lernrate* $\eta$

    Die Konvergenzgeschwindigkeit wird auch durch die Lernrate beeinflußt. Bei kleinen Schrittgrößen sind viele Iterationen notwendig, um ein Minimum zu erreichen. Zu große Schrittgrößen können dazu führen, daß lokale Minima übersprungen werden. Bei einer von Hush und Salas eingeführten Variante von *backprop* wird die Information der Gradientenrichtung ausgenutzt. Werden oft Korrekturen in die selbe Richtung von $E_{\underline{w}}$ vorgenommen, so ist die Schrittgröße zu klein und $\eta$ wird erhöht. Wenn nur wenige Schritte in dieselbe Richtung erfolgen, so wird $\eta$ verkleinert, um eine feinere Suche in Gradientenrichtung zu ermöglichen.

  - *Backpropagation mit Momentum (oder Impuls)*

    In der Regel (3.15) wird ein sogenannter Momentum–Term

    $$w_{j,i}^{n+1} = w_{j,i}^{n} + \eta\, \delta_j\, o_i + \alpha(w_{j,i}^{n} - w_{j,i}^{n-1}) \tag{3.18}$$

    mit geeignetem Parameter $\alpha > 0$ eingeführt. Bei jeder Korrektur von $w_{j,i}$ wird auch die zuletzt vorgenommene Korrektur berücksichtigt. Der Momentum–Term beschleunigt den Lernvorgang in flachen Gebieten der Fehlerfunktion.

  - *Backpropagation mit weight–decay*

    Die Idee ist, eine Dynamik in die Lernregel einzuführen, die kleine Gewichte gegen 0 zieht. Man hofft, dadurch automatisch „überflüssige" Verbindungen und Gatter zu eliminieren. Die psychische Interpretation dieses Ansatzes wäre ein „Vergessen" wenig interessanter Details in den Daten. Die Regel (3.18) wird dann zu

$$w_{j,i}^{n+1} = w_{j,i}^{n} + \eta\, \delta_j\, o_i - \gamma\, w_{j,i}^{n} + \alpha(w_{j,i}^{n} - w_{j,i}^{n-1}) \tag{3.19}$$

mit einer Konstante $\gamma > 0$.

- Wann kann das Training des Netzes abgebrochen werden?

  Es gibt einige Kriterien, die als Abbruchbedingung für das Training verwendet werden können. Da man eine Fehlerfunktion minimieren will, ist es logisch, die Bedingungen für lokale oder globale Minima dieser Funktion zu verwenden. Das bedeutet, man hat ein solches Minimum gefunden, wenn der Gradient der Fehlerfunktion 0 ist (oder man ist nahe an einem Minimum, wenn der Gradient sehr klein ist). Ein Nachteil dieser Methode ist, daß man bei jeder Iteration zusätzlich die Norm des Gradienten berechnen muß. Eine andere Möglichkeit besteht darin, die Fehlerfunktion selbst als Kriterium zu verwenden. Bei einem Minimum sollte sich $E_{\underline{w}}$ nur mehr wenig verändern. Das bedeutet, man bricht das Training ab, wenn die absolute Änderung von $E_{\underline{w}}$ pro Epoche genügend klein ist.

- Wahl des Trainingsmodus:

  In der Praxis muß beim Trainingsvorgang die Menge der Trainingsbeispiele sehr oft durchlaufen werden (einige 1000 mal). Ein komplter Durchlauf durch die Liste der Trainingsbeispiele wird als *Epoche* bezeichnet. Es wird zwischen zwei Trainingsmodi unterschieden, dem *batch Modus* und dem *sequentiellen Modus*. Beim *batch* Modus werden Gewichtsänderungen im Netz jeweils nach dem Durchlauf durch die gesamte Trainingsmenge, also nach jeder Epoche, durchgeführt. Die $\Delta w_{j,i}$ jedes einzelnen Trainingsbeispiels werden dabei aufsummiert. Im sequentiellen Modus hingegen werden die Gewichte nach Präsentation jedes einzelnen Trainigsbeispiels geändert. Die Vorteile des sequentiellen Modus liegen darin, daß es für sogenanntes „*online* lernen“ verwendet werden kann. Die Liste der Trainingsbeispiele kann also auch unendlich lang sein. Weiters wird weniger lokaler Speicher benötigt, da die Summenbildung der $\Delta w_{j,i}$ entfällt. Welcher Modus besser zum Trainieren eines Netzes geeignet ist, hängt jedoch sehr stark vom gestellten Problem ab.

## 3.3 Universales Approximationstheorem

Am Ende dieses Kapitels soll noch kurz ein Theorem angesprochen werden, das besagt, daß jede stetige Funktion mit einem Netzwerk mit nur einer verborgenen Schicht beliebig genau approximiert werden kann. Die exakten mathematischen Beweise sind in [28, Kapitel 1] zu finden.

**Theorem 1 (Universales Approximationstheorem)** *Jede stetige Funktion $F : [0\,,1]^n \rightarrow [0\,,1]^k$ kann mit einem feed–forward Netzwerk $\mathcal{N}_{\underline{w}}$ der Tiefe 2 (eine verborgene Schicht) und nicht polynomieller Aktivierungsfunktion (also z. B. sigmoide Funktion, Exponentialfunktion,. . . ) beliebig genau approximiert werden.*

$$\forall\, \varepsilon > 0 \;\exists\, \mathcal{N}_{\underline{w}} \textit{der Tiefe 2 sodaß } \forall \vec{x} \in [0,1]^n : \|F(\vec{x}) - \mathcal{N}_{\underline{w}}(\vec{x})\| \leq \varepsilon$$

In der Theorie wird jedoch nirgends angegeben, wieviele Gatter man in der verborgenen Schicht für ein bestimmtes $\varepsilon$ benötigt. Das bedeutet, daß die Netzwerkarchitektur beliebig groß werden könnte. Man wählt daher in der Praxis dennoch Netzwerke mit einer 3–Schicht oder 4–Schicht Architektur, in der Hoffnung, damit einige Gatter einsparen zu können.

# Kapitel 4

# Berechnungen mit spikenden Neuronen

Bis jetzt wurden die biologischen Eigenschaften von Neuronen und Synapsen in Kapitel 2 und der Aufbau und die Eigenschaften künstlicher neuronaler Netze und deren Gatter in Kapitel 3 betrachtet. Es stellt sich die Frage, ob sich diese beiden Modelle nicht irgendwie vereinigen lassen könnten. Wenn man beide Modelle vergleicht, so liegt auf der Hand, daß von der Netzwerkstruktur her Gatter auf Neuronen und die Verbindungen im Netz auf Synapsen abgebildet werden. Ein Problem stellen jedoch die analogen Inputsignale und Gewichtswerte für Gatter künstlicher Netzwerke dar. Biologische Neuronen erzeugen ja immer gleich verlaufende Aktionspotentiale, d. h. sozusagen nur Outputs mit dem Wert 1, wenn das Neuron feuert, und 0 sonst. Auch Synapsen sind nicht einfach nur Gewichte, mit denen der Inputwert multipliziert wird, sie erzeugen zeitabhängige Spannungsverläufe — die EPSP's und IPSP's (siehe Abbildung 2.6).

Der traditionelle Ansatz war, die analogen Inputvariablen von Gattern in den Feuerraten von Neuronen zu kodieren. Dabei werden die über einen gewissen Zeitraum auftretenden Spikes gemittelt und so ein analoger Wert kodiert. Im Zusammenhang mit schnellen Berechnungen in der Kortex scheint der Ansatz, gemittelte Feuerraten zu verwenden, aber fragwürdig zu sein. Experimente aus den letzten Jahren haben gezeigt, daß in vielen biologischen neuronalen Systemen in den Zeitpunkten einzelner Spikes die wesentliche Information kodiert ist (siehe dazu beispielsweise [1], [22], [25] oder [26]).

In diesem Kapitel wird ein anderer Weg aufgezeigt, um sigmoide neuronale Netze mit Netzen spikender Neuronen zu simulieren. Die Theorie stammt von Maass und ist in [14] ausführlich beschrieben. Die Basis bildet das Konzept der temporalen Kodierung (*temporal coding*) von analogen Variablen. Gatter aus künstlichen neuronalen Netzen werden in ein Neuronenmodell übergeführt, worin die Feuerzeiten der postsynaptischen Neuronen von der gewichteten Summe der präsynaptischen Neuronen abhängen. Mit diesem Neuronenmodellen kann dann ein Netzwerk spikender Neuronen aufgebaut werden, das in der Theorie die gleiche Berechnungskraft wie ein künstliches *feed–forward* Netzwerk haben sollte.

## 4.1 $\pi$–Gatter

Die Berechnungseinheiten eines neuronalen Netzes sind die Gatter. Sie sind aus Abschnitt 3.1.2 bekannt. Hier wird aus später ersichtlichen Gründen eine etwas modifizierte Notation verwendet. Die Inputs eines Gatters $G$ sollen jetzt mit $s_1, \ldots, s_{n-1} \in [0, \gamma]$ bezeichnet werden und die Gewichte mit $r_1, \ldots, r_{n-1} \in \mathbb{R}$. Der Biaswert wird nicht mit $r_0$, sondern mit $r_n$ bezeichnet. Für die Aktivierungsfunktion $\varphi : \mathbb{R} \to [0, \gamma]$ von $G$ erhält man also $\varphi(\sum_{i=1}^{n-1} r_i\, s_i + r_n)$ (vgl. 3.2). Der Parameter $\gamma > 0$ bestimmt den Wertebereich der Inputs und Outputs des Gatters $G$. Der Einfachheit halber wird wieder angenommen, daß jedes Gatter noch einen zusätzlichen Input $s_n$ mit einem konstanten Wert $c \in (0, \gamma]$ hat. Nachdem man $r_n$ reskaliert hat, ergibt sich die Aktivierungsfunktion $f_G$ von $G$

$$f_g(s_1, \ldots, s_n) = \varphi\left(\sum_{i=1}^{n} r_i \cdot s_i\right) \tag{4.1}$$

mit $s_n = c$. Wie schon erwähnt wird hauptsächlich die sigmoide Funktion (3.3) als Aktivierungsfunktion verwendet. Aus praktischen Gründen empfiehlt sich hier jedoch die Verwendung einer stückweise linearen Aktivierungsfunktion $\pi_\gamma : \mathbb{R} \to [0, \gamma]$. $\pi_\gamma$ soll definiert sein durch

$$\pi(y) = \begin{cases} 0 & \text{, wenn} \quad y < 0 \\ y & \text{, wenn} \quad 0 \leq y \leq \gamma \\ \gamma & \text{, wenn} \quad y > \gamma \end{cases} \tag{4.2}$$

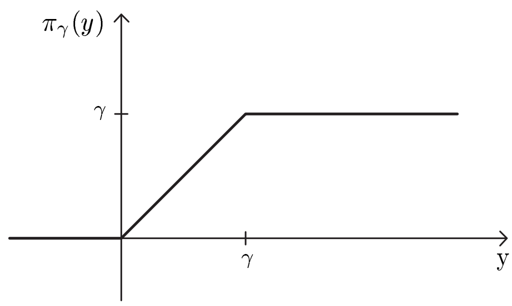



**Abbildung 4.1:** Funktionsverlauf der stückweise linearen Aktivierunsfunktion $\pi_\gamma$.

Ein Gatter mit solch einer Aktivierungsfunktion wollen wir als $\pi$–Gatter ($\pi$–*gate*) bezeichnen. Die Funktion erfüllt alle in Abschnitt 3.1.3 gestellten Bedingungen einer Aktivierungsfunktion, und so bleibt die Berechnungskraft eines Netzes, bestehend aus $\pi$–Gattern, erhalten.

## 4.2 Netzwerke feuernder Neuronen (NFN)

In Abschnitt 3.1.1 ist eine allgemeine Definition eines künstlichen neuronalen Netzes vorgekommen. In analoger Weise wird hier eine Definition eines Netzwerkes feuernder Neuronen gegeben. Ziel ist es, ein Netzwerkmodell zu erhalten, mit dem biologisch realistische Netze möglichst gut beschrieben werden können. Im Modell sollten daher die in Kapitel 2 vorgestellten Eigenschaften biologischer Neuronen berücksichtigt werden. Im Mittelpunkt des Modells stehen die Zeitpunkte des Auftretens von Aktionspotentialen, also das sogenannte „Feuern“ von Neuronen. Daher kommt auch der Ausdruck *Netzwerke feuernder Neuronen* (NFN) bzw. im Englischen *Spiking Neuron Networks* (SNN).

Die einzige wichtige und i.a. auch übliche Annahme ist, daß sich postsynaptische Potentiale im Dendritenbaum linear summieren. Diese Modelle von Neuronen mit linearer dendritischer Integration heißen im Englischen *leaky integrate and fire neurons*.

Definition eines Netzwerkes feuernder Neuronen $\mathcal{N}$:

Ein NFN $\mathcal{N}$ besteht aus

- einem endlichen gerichteten Graphen $G = \langle V, E \rangle$. Die Knoten $V$ des Graphen werden als Neuronen bezeichnet und die Kanten $E$ als Synapsen.
- einer Teilmenge $V_{in} = \{in_1, \ldots, in_n\} \subseteq V$ von $n$ *Input–Neuronen*
- einer Teilmenge $V_{out} = \{out_1, \ldots, out_k\} \subseteq V$ von $k$ *Output–Neuronen*
- einer *Schwellwertfunktion* $\Theta_i : \mathbb{R}^+ \to \mathbb{R}^+$ für jedes Neuron $j \in V \backslash V_{in}$
- einer *Antwortfunktion* $\varepsilon_{j,i} : \mathbb{R}^+ \to \mathbb{R}^+$ für jede Synapse $\langle i, j \rangle \in E$
- einer *Gewichtsfunktion* $w_{j,i} : \mathbb{R}^+ \to \mathbb{R}^+$ für jede Synapse $\langle i, j \rangle \in E$
- einem *Verzögerungswert* $d_{j,i} \in \mathbb{R}^+$ für jede Synapse $\langle i, j \rangle \in E$

Mit dieser Definition können nun folgende Gesichtspunkte biologischer Neuronen berücksichtigt werden:

(a) Die zeitliche Auflösung der Aktionspotentiale. Es werden also nicht mehr mittlere Feuerraten, sondern die exakten Feuerzeiten des Neurons betrachtet. Wir definieren eine geordnete Menge $F_i \subseteq \mathbb{R}^+$, $F_i := \left\{t_i^{(1)}, t_i^{(2)}, t_i^{(3)}, \dots\right\}$ und $t_i^{(s)} < t_i^{(s+1)}$, wobei $t_i^{(s)}$ der Zeitpunkt ist, zu dem ein Neuron $i$ zum $s$-ten mal feuert. Diese Impulsfolgen eines Neurons werden auch als *Spike Trains* bezeichnet. Den Inputneuronen wird eine Impulsfolge zugewiesen. Alle Impulsfolgen der Neuronen $j \in V \backslash V_{in}$ sind durch die Berechnungen im Netzwerk festgelegt.

(b) Eine realistische Form der Signalübertragung zwischen Neuronen. Mit dem Verzögerungswert $d_{j,i}$ und der Antwortfunktion $\varepsilon_{j,i}$ kann einerseits die endliche Ausbreitungsgeschwindigkeit eines Aktionspotentials im Axon als auch der Impulsverlauf eines postsynaptischen Potentials im Dendritenbaum modelliert werden. Für $\varepsilon_{j,i}$ wird üblicherweise die alpha-Funktion (vgl. 2.5) oder sogar eine etwas verallgemeinerte Form der alpha-Funkion (siehe Gleichung (5.2)) verwendet.

Für die theoretischen Betrachtungen in diesem Kapitel wird eine sehr einfache Form für $\varepsilon_{j,i}$ gewählt. Der typische Verlauf eines EPSP bzw. IPSP ist schon aus Abbildung 2.6 bekannt. Wichtig für die folgende Theorie ist jeweils nur der erste Teil, in dem die Funktionen steil ansteigen (EPSP's) bzw. abfallen (IPSP's). Dieser Teil wird mit einer linearen Funktion approximiert. Wir legen $\varepsilon_{j,i} : \mathbb{R}^+ \to \mathbb{R}^+$ folgendermaßen fest:

- *exzitatorische Antwortfunktion*

$$\varepsilon_{j,i}(t) \begin{cases} = 0 & \text{, falls} \quad t \in [0, d_{j,i}] \\ = t - d_{j,i} & \text{, falls} \quad t \in [d_{j,i}, d_{j,i} + \Delta] \\ \geq \varepsilon_{j,i}(d_{j,i} + \Delta) & \text{, falls} \quad t \in [d_{j,i} + \Delta, d_{j,i} + \Delta + \gamma] \end{cases} \tag{4.3}$$

  und

- *inhibitorische Antwortfunktion*

$$\varepsilon_{j,i}(t) \begin{cases} = 0 & \text{, falls} \quad t \in [0, d_{j,i}] \\ = -(t - d_{j,i}) & \text{, falls} \quad t \in [d_{j,i}, d_{j,i} + \Delta] \\ \leq \varepsilon_{j,i}(d_{j,i} + \Delta) & \text{, falls} \quad t \in [d_{j,i} + \Delta, d_{j,i} + \Delta + \gamma] \end{cases} \tag{4.4}$$

Der Verzögerungswert $d_{j,i} \geq 0$ wird auch *delay* genannt. $\Delta$ und $\gamma$ sind Konstanten mit $\Delta \geq 0$ und $0 < \gamma \leq \Delta/2$. Mit der Gewichtsfunktion $w_{j,i}$ für eine Synapse kann schließlich noch eine synaptische Plastizität (vgl. Abschnitt 2.1.3) ins Modell eingebracht werden, das bedeutet, daß die Stärke der synaptischen Verbindungen dadurch festgelegt wird. Für die Auswirkung eines Aktionspotentials an der postsynaptischen Membran erhält man also den Ausdruck $w_{j,i} \cdot \varepsilon_{j,i}$.

(c) Die Entstehung von Aktionspotentialen durch Überschreitung eines Schwellwertes und die Refraktärzeiteigenschaften eines Neurons. Mit (a) und (b) sowie unter der Annahme einer linearen Integration der Potentiale im Dendritenbaum kann das postsynaptische Potential im Zellkern eines Neurons $j$ zur Zeit $t$ als Summe aller an den Dendritenästen bis dahin aufgetretenen PSP's geschrieben werden:

$$P_j(t) := \sum_{i:\langle i,j\rangle \in E} \sum_{s \in F_j : s < t} w_{j,i} \cdot \varepsilon_{j,i}(t-s). \tag{4.5}$$

Überschreitet $P_j(t)$ den Schwellwert $\Theta_j(t)$, so wird das Neuron $j$ zum Feuern gebracht. Die Schwellwertfunktion $\Theta_j(t)$ gibt an, wie leicht ein Neuron zum Feuern gebracht werden kann. Will man die Refraktärzeiteigenschaften eines Neurons einbringen, so muß man die Schwellwertfunktion vorangegangene Aktionspotentiale berücksichtigen. Sei $t'$ der Zeitpunkt des letzten Feuerns von Neuron $j$. Dann muß die Schwellwertfunktion $\Theta_j(t-t')$ für kleine Werte von $t-t'$ sehr groß werden. Für genügend große Werte von $t-t'$ soll $\Theta_j(t-t')$ wieder den ursprünglichen Schwellwert $\Theta_j(0)$ erreichen.

Für die folgende Konstruktion ist der genaue Verlauf von $\Theta_j$ nicht wichtig, da nur jeweils der erste Spike jedes Neurons entscheidend ist. $\Theta_j(t-t')$ muß also nur genügend groß sein, damit j nicht zweimal kurz hintereinander zum Feuern gebracht werden kann.

## 4.3 Timing des Aktionspotentials

Wir benötigen noch einen Mechanismus, der den Schwellwert $\Theta := \Theta_j(0)$ eines Neurons $j$ im Ruhezustand für eine kurze Zeit erhöhen kann. Eine biologische Erklärung für solch eine kurzzeitige Erhöhung des Schwellwertes wäre z. B. eine große Anzahl von zufällig feuernden Neuronen, die über inhibitorische Synapsen Verbindung zu Neuron $j$ haben. Jede dieser Synapsen soll weit vom Zellkern entfernt am Dendritenbaum liegen. Somit hat ein einzelnes dieser IPSP's nur geringe Auswirkung auf das Potential an der Triggerzone (am Axonhügel) des Neurons. Alle IPSP's zusammen erzeugen aber ein ziemlich gleichmäßiges negatives Potential $BN^-$ an der Triggerzone.

Formell soll jedes Neuron $j$ solche Neuronen mit inhibitorischen Synapsen besitzen. Damit haben wir formell die Möglichkeit, während eines kurzen Zeitintervalls ein negatives Potential $BN^- < 0$ zu erzeugen. Weiters nehmen wir an, $BN^-$ sei in einem von uns geforderten Zeitintervall konstant. Die Neuronen, die für die Erzeugung von $BN^-$ zuständig sind, nennen wir *inhibitorische Poolneuronen.*

Es ist schon länger bekannt, daß EPSP's und IPSP's die Feuerzeiten von biologischen Neuronen verschieben können. Dieser Effekt soll nun ausgenützt werden, um analoge Berechnungen in temporaler Kodierung auszuführen. Mit den Definitionen aus Abschnitt 4.2 und dem inhibitorischen Neuronenpool bzw. $BN^-$, haben wir nun die Möglichkeit, die Feuerzeit $t_j$ eines Neurons $j$ in Abhängigkeit der Feuerzeiten $t_{x_1}, \ldots, t_{x_n}$ seiner Inputneuronen $x_1, \ldots, x_n$ mathematisch zu beschreiben:

$$t_j = \frac{\Theta - BN^-}{\sum_{i=1}^n w_{j,i}} + \frac{\sum_{i=1}^n w_{j,i} \cdot (t_{x_i} + d_{j,i})}{\sum_{i=1}^n w_{j,i}} \tag{4.6}$$

Man sieht, daß ein feuerndes Neuron eine lineare Funktion in Abhängigkeit temporaler kodierter Inputs berechnen kann. Für den Fall, daß alle $n$ PSP's zur Zeit $t_j$ noch in ihrer linearen Anstiegsphase (Abstiegsphase) sind, kann dies exakt bewiesen werden. Aber auch in biologischen Systemen, in denen die $t_{x_i}$ zu weit auseinanderliegen um diese Bedingung zu erfüllen, können damit lineare Funktionen approximiert werden (siehe dazu auch Anmerkung 3 in Abschnitt 4.4.2).

## 4.4 Die Konstruktion des Netzwerks

Es sei ein beliebiges $\pi_\gamma$–Gatter $G$ mit $\gamma > 0$ gegeben. Dieses $\pi_\gamma$–Gatter berechnet die Funktion $f_G : [0,\gamma]^n \to [0,\gamma]$. Für Gewichte $r_1, \ldots, r_n \in \mathbb{R}$ und beliebige Inputs $s_1, \ldots, s_n \in [0,\gamma]$ ergibt sich

$$f_G(s_1, \ldots, s_n) = \begin{cases} 0 & \text{, wenn } \sum_{i=1}^n r_i \cdot s_i < 0 \\ \sum_{i=1}^n r_i \cdot s_i & \text{, wenn } 0 \le \sum_{i=1}^n r_i \cdot s_i \le \gamma \\ \gamma & \text{, wenn } \sum_{i=1}^n r_i \cdot s_i > \gamma. \end{cases} \tag{4.7}$$

Für ein gegebenes $\pi_\gamma$–Gatter $G$ und für eine beliebiges $\varepsilon$ mit $0 < \varepsilon < \gamma$ soll nun ein Netzwerk feuernder Neuronen $\mathcal{N}_{G,\varepsilon}$ konstruiert werden, das $f_G$ mit einer Genauigkeit kleiner gleich $\varepsilon$ approximiert. Für den Output $\mathcal{N}_{G,\varepsilon}(s_1, \ldots, s_n)$ gilt also

$$|\mathcal{N}_{G,\varepsilon}(s_1, \ldots, s_n) - f_G(s_1, \ldots, s_n)| \le \varepsilon \tag{4.8}$$

für einen beliebigen Inputvektor $s_1, \ldots, s_n \in [0,\gamma]$.

Zusätzlich zu den Inputs $s_1, \ldots, s_n$ benötigt man einen Input $s_0$ für das Netzwerk $\mathcal{N}_{G,\varepsilon}$. Dieser Input ermöglicht es, die Gewichte passend zu dem gegebenen $\pi_\gamma$–Gatter $G$ zu normieren. $s_0$ soll — wie die anderen Inputs — in zeitlicher Kodierung gegeben sein. Das Netz hat jetzt $n+1$ Inputneuronen $a_0, \ldots, a_n$, die zum Zeitpunkt $T_{in} - s_i$ feuern, wobei $T_{in}$ eine Konstante ist.

Wir konstruieren nun ein Neuron $v$ in $\mathcal{N}_{G,\varepsilon}$. Im Neuron $v$ treten aufgrund seiner Inputs $n+1$ PSP's $h_0(t), \dots, h_n(t)$ auf. Zusätzlich soll $v$ noch Inputs von anderen Neuronen erhalten, dessen Feuerzeiten aber nur von der Konstante $T_{in}$ abhängen.

Die Feuerzeit $t_v$ von Neuron $v$ stellt den Output von $\mathcal{N}_{G,\varepsilon}(s_1, \dots, s_n)$ in temporaler Kodierung dar, d. h.

$$t_v = T_{out} - \mathcal{N}_{G,\varepsilon}(s_1, \dots, s_n) \tag{4.9}$$

für eine weitere Konstante $T_{out}$, die nicht von $s_1, \dots, s_n$ abhängt.

Mit den Definitionen aus Abschnitt 4.2 können wir für die Antwortfunktionen $h_i(t)$ schreiben: $h_i(t) = w_{a_i,v} \cdot \varepsilon_{a_i,v}(t - (T_{in} - s_i))$.

Mit (4.3) bzw. (4.4) und der Annahme, daß alle *delay*–Werte $d_{a_i,v} = d$ gleich groß sind, erhalten wir für die Antwortfunktionen

$$h_i(t) = \begin{cases} 0 & \text{, falls} \quad t - (T_{in} - s_i) < d \\ w_i \cdot (t - (T_{in} - s_i) - d) & \text{, falls} \quad d \leq t - (T_{in} - s_i) \leq d + \Delta \end{cases} \tag{4.10}$$

Die Gewichte $w_i$ sind $w_i = w_{a_i,v}$ im Falle einer exzitatorischen Synapse und $w_i = -w_{a_i,v}$ im Falle einer inhibitorischen Synapse.

Wir setzen voraus, daß das Neuron $v$ längere Zeit nicht mehr gefeuert hat, das bedeutet, daß $\Theta_v(t - t') = \Theta_0$. Außerdem soll der Input aus dem inhibitorischen Neuronenpool $BN^-$ konstant sein, wie es in Abschnitt 4.3 beschrieben wurde. Das Neuron $v$ feuert, wenn die Summe der PSP's plus dem im betrachteten Zeitraum konstanten Potential $BN^-$ den Schwellwert $\Theta$ überschreitet. Die Feuerzeit $t_v$ läßt sich also durch die Gleichung

$$\Theta = \sum_{i=0}^{n} h_i(t_v) + BN^- = \sum_{i=0}^{n} w_i \cdot (t_v - (T_{in} - s_i) - d) + BN^- \tag{4.11}$$

bestimmen, unter der Bedingung, daß

$$d \leq t - (T_{in} - s_i) \leq d + \Delta \quad \text{für} \quad i = 0, \dots, n \tag{4.12}$$

gilt. Setzen wir den zusätzlichen Input $s_0 \equiv 0$, so kann (4.11) umgeformt werden zu

$$t_v = \frac{\Theta - BN^-}{\sum_{i=0}^{n} w_i} + T_{in} + d - \frac{\sum_{i=1}^{n} r_i \cdot s_i}{\sum_{i=0}^{n} w_i}. \tag{4.13}$$

Die Feuerzeit $t_v$ erfüllt Bedingung (4.12), wenn $-s_i \leq t_v - T_{in} - d \leq \Delta - s_i$ für $i = 0, \dots, n$ gilt. Daraus folgt, daß für jedes $s_i \in [0, \gamma]$, $t_v$ Bedingung (4.12) erfüllt, wenn

$$0 \leq \frac{\Theta - BN^- - \sum_{i=1}^{n} r_i \cdot s_i}{\sum_{i=0}^{n} w_i} \leq \Delta - \gamma \tag{4.14}$$

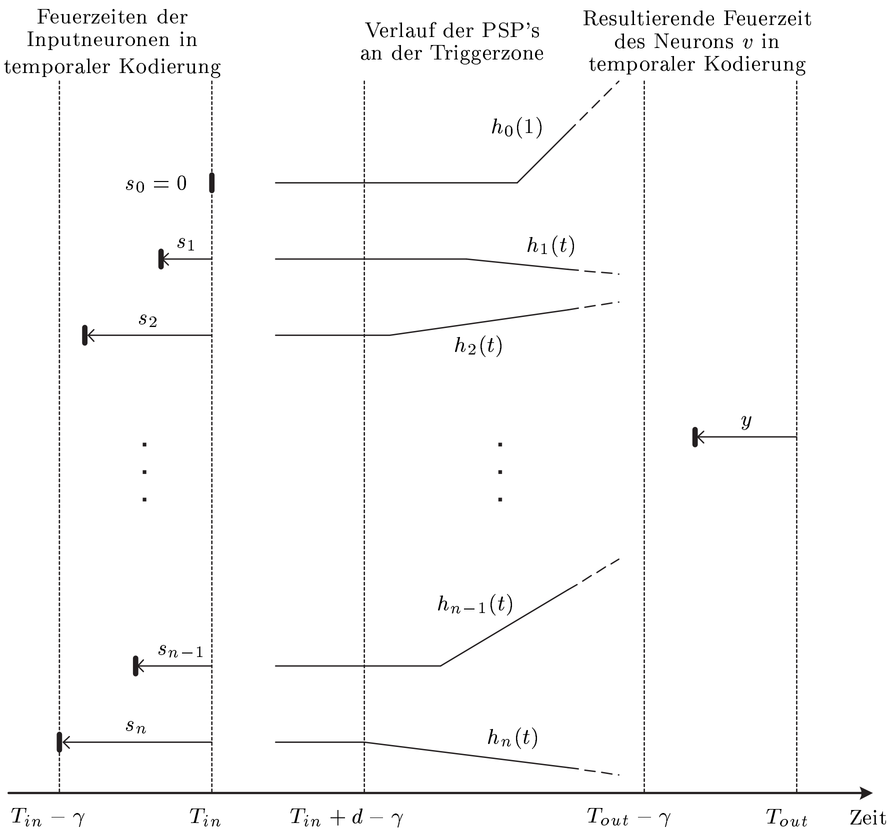



**Abbildung 4.2:** Illustration der Simulation eines $\pi_\gamma$ Gatters durch ein feuerndes Neuron. Die Input Werte $s_0, \ldots, s_n$ und der Outputwert $y$ sind dabei temporal kodiert.

gilt.

Wir führen nun einen Faktor $\lambda > 0$ ein und setzen $w_i := \lambda \cdot r_i$ für $i = 1, \ldots, n$. Die $r_i$ sind dabei die Gewichtswerte des $\pi_\gamma$–Gatters $G$. Das Gewicht $w_0$ soll so gewählt werden, daß $\sum_{i=0}^{n} = \lambda$. Definieren wir die Konstante $T_{out}$ durch

$$T_{out} := \frac{\Theta - BN^-}{\lambda} + T_{in} + d, \tag{4.15}$$

so können wir (4.13) umschreiben zu

$$t_v = T_{out} - \textstyle\sum_{i=1}^{n} r_i \cdot s_i \tag{4.16}$$

und (4.14) zu

$$0 \leq \frac{\Theta - BN^-}{\lambda} - \textstyle\sum_{i=1}^{n} r_i \cdot s_i \leq \Delta - \gamma \tag{4.17}$$

Bei geeigneter Wahl der Konstanten $\gamma$, $\lambda$ und $BN^-$, sodaß die Ungleichungen

$$\gamma \leq \frac{\Theta - BN^-}{\lambda} \leq \Delta - \gamma \tag{4.18}$$

erfüllt sind, ist auch die Ungleichung (4.14) erfüllt. Wir haben also für alle Inputvektoren $(s_1, \ldots, s_n) \in [0, \gamma]$, für die $\sum_{i=1}^{n} ri \cdot si \in [0, \gamma]$ gilt, folgendes gezeigt: Die Feuerzeit $t_v$ des Neurons $v$ stellt den Output $f_G(s_1, \ldots, s_n) = \sum_{i=1}^{n} ri \cdot si$ des simulierten $\pi_\gamma$–Gatters $G$ in temporaler Kodierung dar.

Abbildung 4.2 zeigt, wie man obige Konstruktion des Netzwerkes $\mathcal{N}_{G,\varepsilon}$ver- anschaulichen kann. Die zeitliche Kodierung der Inputs und Outputs ist in Abschnitt 5.3.1 etwas näher beschrieben.

### 4.4.1 Die Neuronenpools

Es bleibt noch zu zeigen, daß (4.8) auch für Werte von $\sum_{i=1}^{n} r_i \cdot s_i \notin [0, \gamma]$ erfüllt ist. Hier soll nur die grobe Beweisidee dazu angegeben werden.

**1. Fall:** $\sum_{i=1}^{n} r_i \cdot s_i < 0$

Ist die gewichtete Summe $\sum_{i=1}^{n} r_i \cdot s_i < 0$, so soll das Neuron $v$ spätestens zum Zeitpunkt $t_v = T_{out}$ feuern. In temporaler Kodierung entspricht diese Feuerzeit dem Wert $y = \mathcal{N}_{G,\varepsilon}(s_1, \ldots, s_n) = 0$. Man erreicht dies durch Einsatz eines exzitatorischen Neuronenpools, der unabhängig vom Inputvektor $(s_1, \ldots, s_n)$ das Potential an der Triggerzone zum Zeitpunkt $t = T_{out}$ über den Schwellwert $\Theta$ hebt. Dadurch wird das Neuron $v$ zum Feuern gebracht.

**2. Fall:** $\sum_{i=1}^{n} r_i \cdot s_i > \gamma$

Ist die gewichtete Summe $\sum_{i=1}^{n} r_i \cdot s_i > \gamma$, so würde das Neuron zu früh feuern, also zu einem Zeitpunkt $t < T_{out} - \gamma$. Durch Einsatz eines inhibitorischen Neuronenpools kann das Potential an der Triggerzone während des Intervalls $[T_{out} - \gamma - \Delta, T_{out} - \gamma]$ etwas verringert werden. Das Potential steigt dadurch nicht über den Schwellwert $\Theta$ an, und das Neuron kann frühestens zur Zeit $t = T_{out} - \gamma$ feuern.

Mit diesen beiden Maßnahmen kann theoretisch gezeigt werden, daß $|T_{out} - \sum_{i=1}^{n} r_i \cdot s_i| \leq \varepsilon$ für $\sum_{i=1}^{n} r_i \cdot s_i < 0$ und $|(T_{out} - \gamma) - \sum_{i=1}^{n} r_i \cdot s_i| \leq \varepsilon$ für $\sum_{i=1}^{n} r_i \cdot s_i < 0$ ist. Damit ist (4.8) gezeigt. Die resultierende Aktivierungsfunktion $\varphi$ von $\mathcal{N}_{G,\varepsilon}$ ist in Abbildung 4.3 zu sehen. Die Funktion ist in den Intervallen $(-\infty, 0]$, $[\varepsilon, \gamma - \varepsilon]$, $[\gamma, \infty]$ linear. In den Intervallen $[0, \varepsilon]$ und $[\gamma - \varepsilon, \gamma]$ weicht die Funktion etwas vom Verlauf von $\pi_\gamma$ ab. Die Abweichung kommt durch die exzitatorischen bzw. inhibitorischen Neuronenpools zustande.

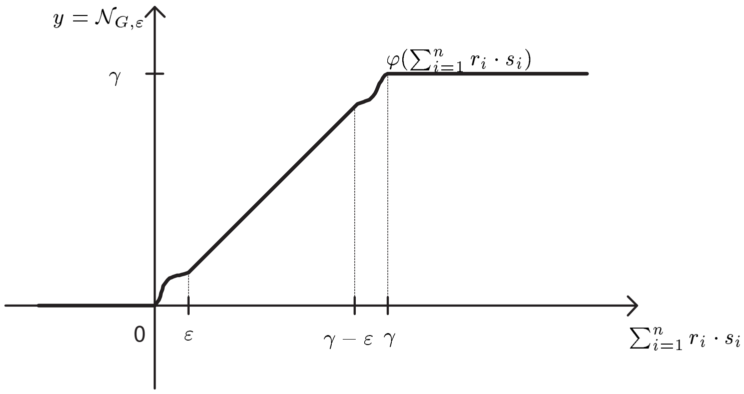



**Abbildung 4.3:** Die Abbildung zeigt die resultierende Aktivierungsfunktion $\varphi$ eines Neurons $v$, das das gegebene $\pi_\gamma$ Gatter simuliert.

### 4.4.2 Anmerkungen

1. Eine biologisch realistischere Annahme ist, daß $P_v(t)$ und $\Theta_v(t)$ von Rauschsignalen gestört sind. $P_v(t)$ und $\Theta_v(t)$ werden dabei neu definiert durch

$$P_v^{noise}(t) := P_v(t) + \alpha_v(t)$$

und

$$\Theta_v^{noise}(t - t') := \Theta_v(t - t') + \beta_v(t).$$

$\alpha_v(t)$ und $\beta_v(t)$ beschreiben den Einfluß unbekannter Störgrößen. Meist wird für diese Störgrößen eine spezielle Wahrscheinlichkeitsverteilung (z. B. weißes Rauschen) verwendet. Es kann sogar für beliebige, absolut beschränkte Störfunktionen auch gezeigt werden, daß die Bedingung (4.8) hält. Der Beweis ist in [14] zu finden.

2. Mit dem Ergebnis aus Abschnitt 4.4 läßt sich das universale Approximationstheorem (vgl. Abschnitt 3.3) auf Netzwerke spikender Neuronen umlegen. D. h., wenn ein NN $N$ eine beliebige stetige Funktion $F : [0, \gamma]^n \to [0, \gamma]^k$ mit einer Genauigkeit $\varepsilon$ approximiert, dann können wir auch ein Netzwerk feuernder Neuronen $\mathcal{N}$ konstruieren, das $F$ mit einer Genauigkeit von $\varepsilon$ approximiert. Das Theorem besagt auch, daß $F$ mit nur einer verborgenen Schicht von Neuronen approximiert werden kann. Berücksichtigt man das Zeitverhalten biologisch realistischer Neuronen, so kann der Output von $\mathcal{N}$ in weniger als 20 ms berechnet werden.

3. Auch ohne Verwendung der Neuronenpools ist es möglich, mit Netzwerken spikender Neuronen künstliche *feed–forward* Netzwerke zu approximieren.

Dabei wird der konkrete Verlauf der PSP's von biologischen Neuronen ausgenützt. Außerdem wird nicht mehr von der speziellen Aktivierungsfunktion $\pi_\gamma$ ausgegangen. Ein weiterer Vorteil dieses Ansatzes ist, daß die linearen Anstieg– bzw. Abfallsegmente der PSP's sich nicht mehr überschneiden müssen.

Betrachten wir ein Neuron $v'$ und dazu ein Neuron $v$ aus der Vorgängerschicht von $v'$, so können wir folgendes feststellen: Feuert $v$ später als $T_{out}$, dann hat das resultierende PSP zur Zeit $t_{v'}$ wo $v'$ feuert immer noch den Wert 0. $v$ verliert dadurch seinen Einfluß auf die Neuronen in der nächsten Schicht.
Feuert $v$ hingegen früher als $T_{out} - \gamma$, so wird das resultierende PSP zur Zeit $t_{v'}$ schon im Bereich seines Maximalwertes sein. Dort steigt das PSP viel langsamer an als in seiner „linearen" Anfangsphase, bzw. es fällt schon wieder ab. Auch dieses Neuron hat weniger Einfluß auf die nächste Schicht.

Diese beiden Effekte sind mit dem „Sättigungsbereich" ($= |u|$ sehr groß in (3.3)) einer sigmoiden Aktivierungsfunktion zu vergleichen. Damit kann ein NFN auch ohne der biologisch wenig realistischen Annahme von Neuronenpools und ohne der Annahme, daß all PSP's sich in ihrer „linearen" Phase überlappen, komplexe Berechnungen in temporaler Kodierung ausführen.

Weitere Anmerkungen zur biologischen Relevanz des vorgestellten Konstrukts und der Anwendbarkeit von Lernregeln darauf sind [14] zu entnehmen.

# Kapitel 5

# Simulationen mit biologisch realistischen Neuronen

Es soll hier untersucht werden, ob die Theorie aus Kapitel 4 auch auf Computermodelle von Neuronen übertragbar ist. Dazu wird als erstes ein Kompartmentmodell eines Neurons festgelegt. Zur Simulation der Kompartmentmodelle wird der Simulator GENESIS verwendet (vgl. Anhang A). Danach wird das charakteristische Verhalten dieses Modellneurons eingehender betrachtet, da die Eigenschaften eines einzelnen Neurons auch später in einem Netzwerk berücksichtigt werden müssen. In weiterer Folge werden künstliche Netzwerke mit in GENESIS simulierten Netzwerken verglichen. Zum Schluß wird noch versucht, den Backpropagation Algorithmus auf simulierte Netzwerke in GENESIS anzuwenden.

## 5.1 Das Zellmodell

### 5.1.1 Aufbau

Als Zellmodell wird ein Kompartmentmodell verwendet (vgl. Abschnitt 2.2). Je komplizierter ein solches Modellneuron ist, desto genauer können biologische Sachverhalte berücksichtigt werden. Die Simulationszeit nimmt dabei jedoch zu. Da später ganze Netzwerke mit diesen Modellneuronen modelliert werden sollen, wird hier ein relativ einfaches Zellmodell gewählt. Am einfachsten wäre ein Modell, das nur aus 2 Kompartments besteht: Ein Kompartment für das Soma und eines für den Dendritenbaum. Alle synaptischen Verbindungen zu diesem Neuron würden dann im Kompartment, das den Dendritenbaum repräsentiert, liegen. Eine solche Anordnung führt aber leider dazu, daß das aus den einzelnen PSP's resultierende Potential nicht gleich der Summe der PSP's ist (vgl. dazu [13, Kapitel 2]). Man muß daher ein räumlich strukturierteres Neuronenmodell verwenden.

Ein Neuron soll also ab jetzt aus $n+4$ Kompartments bestehen, wobei $n$ für die Anzahl der Inputs eines Neurons steht (im Netzwerk ist $n$ also immer die Anzahl der Neuronen in der Vorgängerschicht). Die einzelnen Kompartments sind: das Soma, der Stamm des Dendritenbaumes und $n+2$ Dendritenäste. Dabei sind $n$ Äste für die Inputs $s_1, \ldots, s_n$ und jeweils ein Ast für den Input des Biasneurons und des Referenzneurons (vgl. Abschnitt 4.4) vorgesehen. Abbildung 5.1 zeigt, wie man sich das Zellmodell schematisch vorstellen kann.

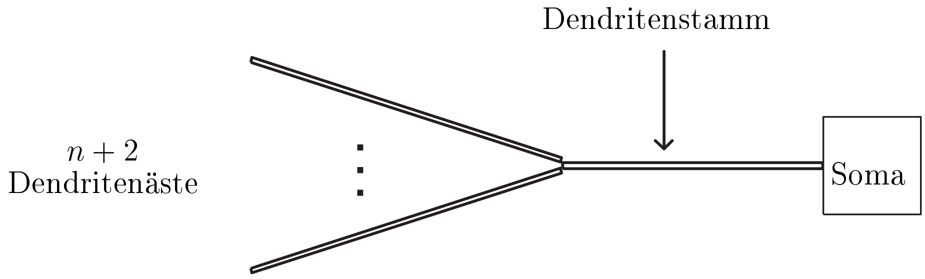



**Abbildung 5.1:** Schema des verwendeten Neuronenmodells. Die einzelnen Kompartments haben eine zylindrische Form. Die Größenverhältnisse sind maßstabsgetreu. Die tatsächlichen Werte sind den Tabellen in Abbildung 5.2 und 5.3 zu entnehmen.

Die spezifischen Membrankonstanten und Umkehrpotentiale sind in Tabelle 5.1 aufgelistet. Daraus können mit den Abmessungen für die einzelnen Kompartments die Werte von $A$, $C_m$, $R_a$ und $g_{leak}$ errechnet werden durch:

$$\begin{aligned} A &= d \cdot \pi \cdot l \\ g_{leak} &= \frac{1}{R_{leak}} = \frac{A}{R_M} \\ C_m &= A \cdot C_M \\ R_a &= \frac{4 \cdot R_A \cdot l}{d^2 \cdot \pi} \end{aligned} \tag{5.1}$$

## 5.1.2 Die Kompartments

Wir kommen nun zur genauen Spezifikation der einzelnen Kompartments. Im linken Teil von Abbildung 5.2 ist das elektrische Ersatzschaltbild des Zellkörpers zu sehen. Da dort das Aktionspotential generiert werden soll, sind die im Hodgkin–Huxley Modell vorgestellten Leitwertzweige $g_{Na}$ und $g_K$ vorhanden. Die Formeln und Aktivierungsparameter dieser aktiven Kanäle sind Abschnitt 2.3 zu entnehmen.

| Beschreibung | Zeichen | Wert | Einheit |
|---|---|---|---|
| Umkehr-potentiale | $E_{Na}$ | 45 | $mV$ |
| | $E_K$ | -82 | $mV$ |
| | $E_{syn(E)}$ | 45 | $mV$ |
| | $E_{syn(I)}$ | -82 | $mV$ |
| Spezifische Membran-konstanten | $R_M$ | 0.333 | $\Omega m^2$ |
| | $C_M$ | 0.01 | $F/m^2$ |
| | $R_A$ | 0.3 | $\Omega m$ |

**Tabelle 5.1:** Die Tabelle gibt die Umkehrpotentiale für die Natrium- und Kaliumionen an. Da exzitatorische Synapsen meist einen Einstrom von Natriumionen und inhibitorische Synapsen einen Einstrom von Kalium und Chlorionen verursachen, werden die Umkehrpotentiale gleich den Umkehrpotentialen von $Na$- und $K$–Kanälen gewählt. Die Werte für die Umkehrpotentiale ergeben sich aus der *Nernst´schen* Gleichung (vgl. [9, Kapitel 2]). Die spezifischen Membrankonstanten sind in Abschnitt 2.2 schon erklärt worden. Aus diesen Werten lassen sich mit Hilfe der Formeln 5.1 die passiven Membraneigenschaften errechnen.

Abbildung 5.3 stellt das Ersatzschaltbild des Stammes bzw. eines Astes des Dendritenbaumes dar. In diesen Kompartments sind zusätzlich zu den passiven Eigenschaften der Membran die zeitlich veränderlichen Leitwerte $g^E_{syn}$ und $g^I_{syn}$ eingefügt. Dadurch werden exzitatorische und inhibitorische Synapsen modelliert. Im Simulator GENESIS wird der Zeitverlauf postsynaptischer Potentiale mit einer allgemeineren Form der alpha–Funktion beschrieben:

$$g_{syn}(t) := A \, \frac{g_{max}}{\tau_1 - \tau_2} \left( e^{-\frac{t}{\tau_1}} - e^{-\frac{t}{\tau_1}} \right), \quad \text{für} \quad \tau_1 > \tau_2. \qquad (5.2)$$

Der Faktor $A$ normiert die Funktion so, daß sie ein Maximum von $g_{max}$ erreicht. Mit den Zeitkonstanten $\tau_1$ und $\tau_2$ kann die Anstiegs– und Abfallzeit verändert werden. Im Allgemeinen haben exzitatorische und inhibitorische Synapsen ein unterschiedliches Zeitverhalten. Inhibitorische Synapsen sind meist langsamer, d. h. sie haben größere Werte für $\tau_1$ und $\tau_2$.

Für die Simulationen wurde bei den exzitatorischen Synapsen $\tau_1 = \tau_2 = 3\,ms$ und bei den inhibitorischen Synapsen $\tau_1 = \tau_2 = 10\,ms$ gewählt. Die Umkehrpotentiale $E^E_{syn}$ und $E^I_{syn}$ sind Tabelle 5.1 zu entnehmen.

Die Kompartments werden, wie es Abbildung 5.1 zeigt, zusammengeschaltet. Hilfe zum Aufbau solcher Anordnungen von Kompartments im Simulationsprogramm GENESIS findet sich im zweiten Teil von [6] und in [3] (vgl. dazu auch Anhang A).

| Soma | | Einheit |
|---|---|---|
| $l^S$ | 30 | $\mu m$ |
| $d^S$ | 30 | $\mu m$ |
| $A^S$ | 2827 | $\mu m^2$ |
| $g^S_{leak}$ | 8.482 | $nS$ |
| $E^S_{leak}$ | -59.4 | $mV$ |
| $C^S_m$ | 28.27 | $pF$ |
| $R^S_a$ | 12.732 | $K\Omega$ |

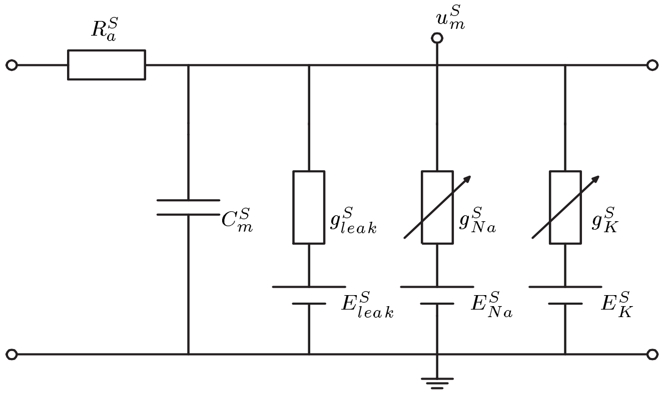
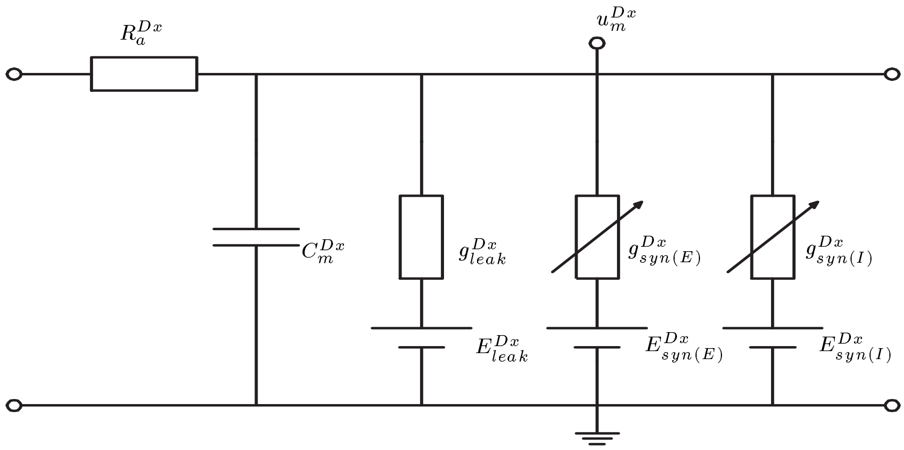



**Abbildung 5.2:** Abmessungen und elektrisches Ersatzschaltbild des Zellkörpers (Soma). Wie sich die einzelnen Werte berechnen lassen ist im Text angegeben.




| Dendritenstamm | | Einheit |
|---|---|---|
| $l^{DS}$ | 80 | $\mu m$ |
| $d^{DS}$ | 2 | $\mu m$ |
| $A^{DS}$ | 503 | $\mu m^2$ |
| $g^{DS}_{leak}$ | 1.508 | $nS$ |
| $E^{DS}_{leak}$ | -70 | $mV$ |
| $C^{DS}_m$ | 5.03 | $pF$ |
| $R^{DS}_a$ | 7.64 | $M\Omega$ |

| Dendritenast | | Einheit |
|---|---|---|
| $l^{DA}$ | 100 | $\mu m$ |
| $d^{DA}$ | 1.5 | $\mu m$ |
| $A^{DA}$ | 471 | $\mu m^2$ |
| $g^{DA}_{leak}$ | 1.414 | $nS$ |
| $E^{DA}_{leak}$ | -70 | $mV$ |
| $C^{DA}_m$ | 4.71 | $pF$ |
| $R^{DA}_a$ | 16.97 | $M\Omega$ |

**Abbildung 5.3:** Hier ist das elektrische Ersatzschaltbild eines Dendritenteiles abgebildet. Die Abmessungen und die Werte im Ersatzschaltbild sind für den Dendritenstamm (DS) (Tabelle unten links) und einen Dendritenast (DA) (Tabelle unten rechts) angegeben.

## 5.2 Analyse des Zellmodells

Bevor man mit dem obigen Modellneuron ganze Netzwerke zusammenfüget, ist es ratsam, das Modell des einzelnen Neurons genauer zu analysieren. Es geht also um die Frage, wie sich postsynaptische Aktionspotentiale auf die Feuerzeit eines Neurons auswirken.

Um die Arbeit zu erleichtern, wird ab jetzt angenommen, daß bei allen vorkommenden Synapsen die Verzögerungszeiten $d_{j,i}$ gleich Null sind.

### 5.2.1 Anstiegszeit

In der Theorie wird gefordert, daß alle postsynaptischen Potentiale entweder in ihrer Anstiegsphase oder Abfallphase linear sind. Dieser Teil der Antwortfunktion wird in Kapitel 4 durch eine Gerade approximiert (vgl. Abschnitt4.2).

Wir wollen die Anstiegszeit für unser Kompartmentmodell ermitteln, die wir benötigen, um die Werte für $\Delta$ und $\gamma$ zu erhalten. Für gleiche Werte der Zeitkonstanten $\tau_1 = \tau_2$ erreicht $g_{syn}(t)$ den Maximalwert $g_{max}$ nach einer Zeit von $t = \tau_1$ nach Eintreffen eines Spikes an der Synapse. Die anfängliche lineare Phase der Funktion $g_{syn}(t)$ ist aber etwas kürzer. In einem Neuronenmodell mit nur einem Dendritenast, dem Dendritenstamm und dem Soma wurden für verschiedene Werte von $g_{max}$ die Spannungsverläufe von EPSP's und IPSP's aufgezeichnet. Zum Zeitpunkt $t = 1$ ms tritt jeweils ein Aktionspotential an der Synapse auf. Die Kurvenschar in Abbildung 5.4 zeigt den gemessenen Potentialverlauf am Soma.

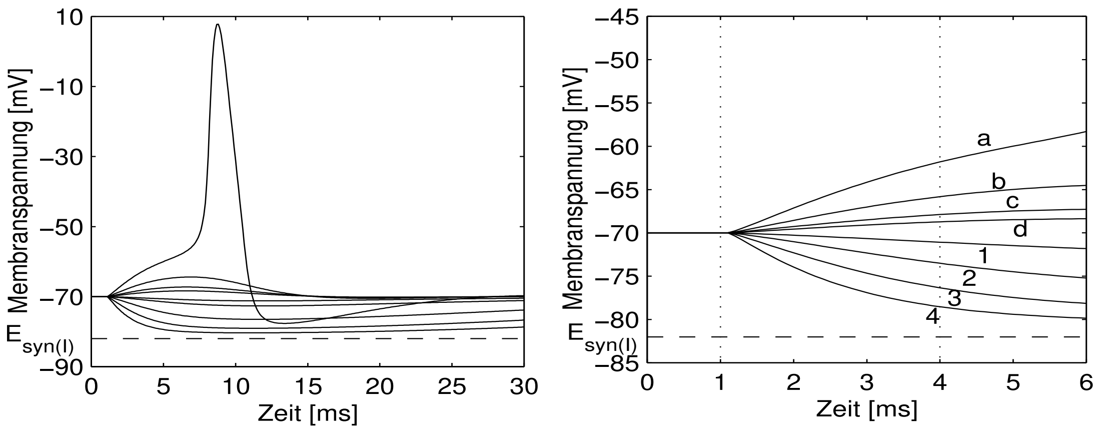



**Abbildung 5.4:** Links: Kurvenschar von EPSP's und IPSP's für verschiedene Werte von $g_{max}$. Die Aktionspotentiale treten jeweils zum Zeitpunkt $t = 1$ ms an der Synapse auf. Rechts: Detailausschnitt des linken Bildes. Die Werte von $g_{max}$ waren dabei: **a**: $g_{max(E)}$ = 2 nS, **b**: $g_{max(E)}$ = 1 nS, **c**: $g_{max(E)}$ = 0.5 nS, **d**: $g_{max(E)}$ = 0.3 nS, **1**: $g_{max(I)}$ = 5 nS, **2**: $g_{max(I)}$ = 20 nS, **3**: $g_{max(I)}$ = 50 nS, **4**: $g_{max(I)}$ = 100 nS

Es ist zu erkennen, daß in den ersten 3 ms nach Eintreffen des Spikes ein linearer Spannungsverlauf am Soma auftritt. Wir legen daher für unser Neuronenmodell ein $\Delta = 3$ ms fest. Aus der Bedingung $0 < \gamma \leq \Delta/2$ aus Abschnitt 4.2 ergibt sich ein Wert für $\gamma$ von $\gamma \leq 1.5$ ms.

### 5.2.2 Schwellwert

Im Soma sind Hodgkin–Huxley Kanäle vorhanden, die ein Aktionspotential auslösen. Dies geschieht, wenn im Soma eine gewisse Schwellspannung überschritten wird. Da wir später nur noch mit den Gewichten der Synapsen arbeiten wollen, interessiert uns hier aber nicht so sehr die Schwellspannung selbst. Wichtiger ist, wie groß der maximale Leitwert einer exzitatorischen Synapse sein muß, damit ein Neuron zum Feuern gebracht werden kann. Wir bezeichnen diese Schwelle mit $G_{th}$.

Schaut man sich das elektrische Ersatzschaltbild des Neuronenmodells genauer an, so ist klar, daß dieser Schwellwert $G_{th}$ von der Anzahl der Dendritenäste abhängt. Je mehr Dendritenäste ein Neuron hat, desto mehr parallele Leckstromzweige sind vorhanden. Der Strom fließt also nicht nur vom Dendritenast zum Soma, sondern ein Teil fließt auch in die anderen Dendritenäste ab.

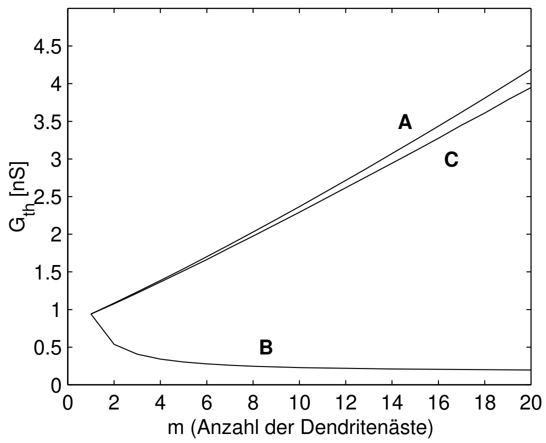



**Abbildung 5.5:** Ergebnisse der Messungen des Leitwertes $G_{th}(m)$. Kurve **A**: Nur an einem von $m$ Dendritenästen wurde ein Aktionspotential angelegt. Kurve **B** und **C**: An allen $m$ Dendritenästen wurde zur gleichen Zeit ein Aktionspotential angelegt. Kurve **B** zeigt den Schwellwert $G_{th}$ an einem Ast und **C** die Summe der Schwellwerte aller $m$ Äste.

Der erforderliche Leitwert $G_{th}$ wurde mittels binärer Suche ermittelt. Im Versuch wird der Leitwert $g_{max}$ erhöht, falls nach Erreichen des Maximalwertes des PSP kein Aktionspotential ausgelöst wurde. Wenn aber bis zu diesem Zeitpunkt im Soma bereits ein Spike generiert wurde, wird $g_{max}$ verringert. Die Anzahl der Dendritenäste ging dabei von 1 bis 20.

Die Messung wurde auf 2 verschiedene Arten durchgeführt. Im ersten Fall werden zwar $i$ ($1 \leq i \leq 20$) Dendritenäste generiert, aber nur an einem Ast wird ein Aktionspotential angelegt. Kurve **A** in Abbildung 5.5 zeigt das Ergebnis dieses Versuches. Im zweiten Fall wird an jedem der $i$ Äste gleichzeitig ein Aktionspotential angelegt. Kurve **B** bzw. **C** zeigen das Ergebnis.

Das Programm ermittelt auch die Schwellspannung $U_{th}$ am Soma. Die Werte aus den einzelnen Versuchen schwanken aufgrund der Programmierung ein wenig. Aus den ermittelten Werten wurde der Mittelwert gezogen.

$$\begin{aligned} \text{Fall } \mathbf{A}: \quad & U_{th}^{avg} = 13,1\,\mathrm{mV} \\ \text{Fall } \mathbf{B}, \mathbf{C}: \quad & U_{th}^{avg} = 11,8\,\mathrm{mV} \end{aligned}$$

Die Werte beziehen sich auf die Membranruhespannung von -70 mV.

Aus der kleinen Abweichung der Kurven **A** und **C** ist zu sehen, daß die dendritische Integration nicht ganz linear ist, da sonst beide Kurven gleich verlaufen müßten. Durch die Konstruktion des Modellneurons mit einem Dendritenstamm und mehreren Dendritenästen kann aber eine annähernde lineare Integration der PSP's am Dendriten erreicht werden.

Bei allen weiteren Versuchen werden für den Schwellwert $G_{th}(m)$ die Werte aus Kurve **C** verwendet.

### 5.2.3 Interaktion von EPSP's und IPSP's

In Netzwerken kommen auch negative Gewichtswerte vor. In unserem Neuronenmodell bedeutet ein negatives Gewicht die Aktivierung einer inhibitorischen Synapse. Wie schon aus Abbildung 2.6 ersichtlich ist, haben EPSP's und IPSP's unterschiedliche Zeitverläufe. Insbesondere verlaufen IPSP's viel flacher als EPSP's.

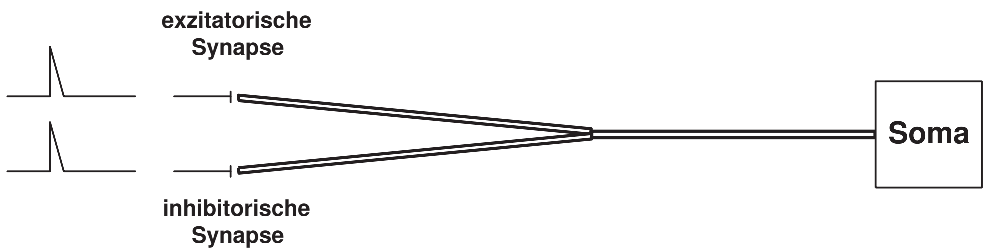



**Abbildung 5.6:** Dieses Neuronenmodell wurde verwendet, um die Auswirkung synchroner Spikes auf exzitatorische und inhibitorische Synapsen zu zeigen.

Um diesen Unterschied auszugleichen, wurde folgender Versuch durchgeführt: Zuerst wird ein Neuronenmodell mit nur 2 Dendritenästen generiert. An einem

Ast soll sich eine exzitatorische Synapse und am anderen Ast eine inhibitorische Synapse befinden. Die Anordnung ist in Abbildung 5.6 zu sehen. An beiden Synapsen tritt zur gleichen Zeit $t = t_0$ ein Aktionspotential auf.

Für einen gegebenen Leitwert $g_{max(E)}$ der exzitatorischen Synapse wird nun der Leitwert $g_{max(I)}$ gesucht, sodaß das resultierende Potential am Soma im Intervall $[t_0, t_0 + \Delta]$ möglichst auf dem Ruhepotential von -70 mV bleibt. Für die so ermittelten Wertepaare $(g_{max(E)}, g_{max(I)})$ heben sich die Wirkungen des EPSP und des IPSP in diesem Intervall also auf. Kurve **A** in Abbildung 5.7 zeigt den Verlauf von $g_{max(I)}(g_{max(E)})$.

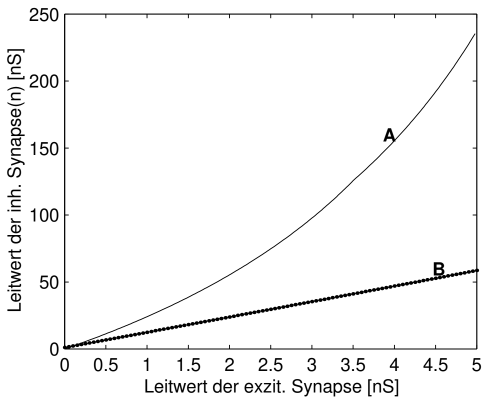



**Abbildung 5.7:** Die Kurven zeigen den Funktionsverlauf von $g_{max(I)}(g_{max(E)})$. Kurve **A**: Ergebnis mit Modellschema aus Abbildung 5.6. Für die Wertepaare auf der Kurve heben sich die Wirkungen von EPSP und IPSP auf. Kurve **B**: Ergebnis bei verfeinertem Modellschema (Erklärung im Text). Eine mittels linearer Regression ermittelte Gerade durch die Meßpunkte hat die Form $g_{max(I)} = 11,5 \cdot g_{max(E)} + 0.98$.

Ungünstigerweise ist der Verlauf dieser Kurve sehr nichtlinear. Verantwortlich dafür ist die Unsymmetrie der treibenden Spannungsquellen für exzitatorische und inhibitorische Synapsen. Für $g_{max(E)} \to \infty$ könnte am exzitatorischen Dendritenast eine Membranspannung von 45 mV — also 115 mV über dem Ruhepotential — auftreten. Für $g_{max(I)} \to \infty$ wird aber maximal eine Membranspannung von -82 mV — -12 mV unter dem Ruhepotential — erreicht.

Eine Möglichkeit diese Unlinearität auszugleichen, wäre eine weitere Verfeinerung des Neuronenmodells. Das Verhältnis der beiden treibenden Spannungsquellen in Bezug auf das Ruhepotential beträgt $115 : 12 \approx 10 : 1$. Fügt man an den Ast mit der inhibitorischen Synapse noch 10 weitere Teiläste mit inhibitorischen Synapsen an, und wiederholt den Versuch, so ergibt sich Kurve **B** in Abbildung 5.7. Der angegebene inhibitorische Leitwert ist die Summe der 10 Leitwerte in den zusätzlichen Ästen. Die Kurve verläuft jetzt ungefähr linear. Diese weitere Verfeinerung des Modells hätte aber eine starke Zunahme der Simulationszeiten zur Folge. Deswegen wird das alte Modell beibehalten.

### 5.2.4 Gewichte

Im Folgenden wird nicht mehr von Leitwerten von Synapsen gesprochen, sondern nur mehr von Gewichten. Gemeint ist mit dem Gewicht aber immer noch die maximale Leitwertänderung $g_{max}$ einer Synapse. Zu unterscheiden ist aber zwischen positiven und negativen Gewichten. An jedem Ast des Neuronenmodells sind zwei Synapsen vorhanden: eine exzitatorische und eine inhibitorische. Mit diesen beiden Synapsen wird nun eine Verbindung zwischen zwei Neuronen $i$ und $j$ modelliert. Bei einem positiven Gewichtswert ($w_{j,i} \geq 0$) wird der maximale Leitwert der exzitatorischen Synapse gleich dem Gewicht $w_{j,i}$ gesetzt. Der Leitwert der inhibitorischen Synapse wird auf 0 gesetzt. Ist das Gewicht jedoch negativ ($w_{j,i} < 0$), dann wird der Leitwert der exzitatorischen Synapse auf 0 gesetzt. In Abschnitt 5.2.3 haben wir gesehen, daß die Wirkung einer inhibitorischen Synapse nicht gleich der Wirkung einer exzitatorischen Synapse ist. Daher wird der Leitwert der inhibitorischen Synapse gleich dem Gewicht $w_{j,i}$, multipliziert mit einem Faktor $f_{inh}$, gesetzt. D. h.

$$g_{max(E)} = \begin{cases} w_{j,i} & \text{, falls} \quad w_{j,i} \geq 0 \\ 0 & \text{, sonst} \end{cases} \tag{5.3}$$

$$g_{max(I)} = \begin{cases} -w_{j,i} \cdot f_{inh} & \text{, falls} \quad w_{j,i} < 0 \\ 0 & \text{, sonst.} \end{cases} \tag{5.4}$$

Die Gewichte haben die Einheit Siemens (S). Wenn keine Angabe der Einheit bei Gewichten erfolgt, so bezieht sich der Wert auf $10^{-9}$ Siemens (nS).

### 5.2.5 Referenzzeitpunkt

Als nächstes müssen wir den Referenzzeitpunkt eines Neurons $j$ bestimmen. In Abschnitt 4.4 wird dieser Zeitpunkt mit $T_{out}$ bezeichnet. Bevor der Output eines Neurons überhaupt bewertet werden kann, muß dieser Referenzzeitpunkt bekannt sein.

Aus der Messung des Schwellwertes (vgl. Abschnitt 5.2.2) ist der Leitwert $G_{th}$ bekannt, der gerade ausreicht, um ein Neuron zum Feuern zu bringen. Da wir wollen, daß Neuron $j$ feuert, muß die Summe der Gewichte $\lambda$ des Neurons, größer gleich dem Schwellwert $G_{th}$ sein ($\lambda$ wurde in Abschnitt 4.4 durch $\lambda = \sum_i w_{j,i}$ definiert). Wir wählen nun einen Faktor $\kappa \geq 1$ und legen damit den Wert $\lambda$ fest:

$$\lambda = \kappa \cdot G_{th}(m) \tag{5.5}$$

$\lambda$ ist also über $G_{th}$ von der Anzahl $m$ der Dendritenäste abhängig. An jedem Dendritenast wird nun das Gewicht auf $w_{j,i} = \lambda/m$ gesetzt. Zu einem gegebenen

Zeitpunkt $t = T_{in}$ feuern alle $m$ Neuronen, die eine Verbindung zu Neuron $j$ haben. Der Zeitpunkt $t_j$ des Feuerns von Neuron $j$ wird als Referenzzeitpunkt $T_{out}$ verwendet (siehe Abbildung 5.8).

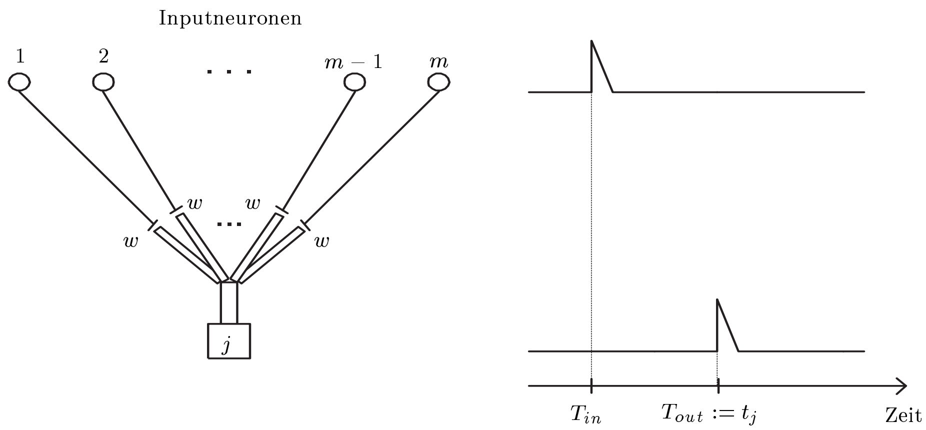



**Abbildung 5.8:** Diese Bild soll die Bestimmung des Referenzzeitpunktes eines Neurons $j$ verdeutlichen. Die Gewichte des Neurons werden alle auf den Wert $w = \lambda/m$ gesetzt. Zu einem gegebenen Zeitpunkt $T_{in}$ feuern die $m$ Neuronen, die Verbindung zu $j$ haben. Da die Summe der Gewichte gleich $\lambda > G_{th}(m)$ ist, wird dadurch auch das Neuron $j$ zum Feuern gebracht. Der Zeitpunkt $t_j$, an dem $j$ feuert, wird als Referenzzeitpunkt $T_{out}$ verwendet.

Der Referenzzeitpunkt für die Inputneuronen wird in Abschnitt 4.4 mit $T_{in}$ bezeichnet. Damit nicht immer zwischen Input– und Outputneuronen unterschieden werden muß, bezeichnen wir den Referenzzeitpunkt eines Neurons nur mehr mit $T_{ref}$. Wie die Referenzzeitpunkte in einem ganzen Netzwerk bestimmt werden, ist in Abschnitt 5.4.1 geschildert.

### 5.2.6 Refraktärverhalten

Schon bei der Konstruktion des NFN in Abschnitt 4.4 wurde deutlich, daß in einem Berechnungszyklus (*feed-forward pass*) des Netzes jeweils nur die ersten Spikes der Neuronen betrachtet werden. Das Refraktärverhalten des Neuronenmodells kann daher vernachlässigt werden.

Programmtechnisch wird in der GENESIS–Simulation folgendermaßen vorgegangen: Für jedes Neuron gibt es ein Objekt, das den Spannungsverlauf am Soma „beobachtet“. Bei einem Anstieg dieser Spannung über 0 mV registriert das Objekt einen Spike. Durch den Zeitpunkt, zu dem dies geschieht, ist auch der Output eines Neurons bestimmt. Solche Objekte werden in GENESIS als *Spike–Detektoren* bezeichnet.

Bei diesen *Spike–Detektoren* läßt sich einfach eine Refraktärzeit setzen. In dem eingestellten Zeitraum, der hier nur genügend groß gewählt werden muß, werden unabhängig von der Spannung am Soma keine weiteren Spikes detektiert, d. h. es wird nur der erste Spike betrachtet.

## 5.3 Die Aktivierungsfunktion

Gatter in künstlichen NN haben eine Aktivierungsfunktion $\varphi$, die die Amplitude des Gatteroutputs beschränkt (vgl. Kapitel 3.1.3). Die Aktivierungsfunktion wird dabei für die Gatter des Netzwerkes festgelegt. Bei unserem Modellneuron kann die Aktivierungsfunktion aber leider nicht einfach festgelegt werden. Sie ergibt sich aus den Feuerzeiten des Modellneurons in Abhängigkeit von der gewichteten Summe der Inputs des Neurons. Theoretisch sollte sich die in Abbildung 4.3 dargestellte Aktivierungsfunktion ergeben. Welche Form die Aktivierungsfunktion aber tatsächlich hat, soll jetzt festgestellt werden. Dazu wird zuerst das Prinzip der temporalen Kodierung noch einmal erklärt und dann der Versuchsaufbau dargestellt. Ähnliche Versuche mit Aktivierungsfunktionen sind in [19] beschrieben.

### 5.3.1 Zeitliche Kodierung

Ab jetzt müssen Inputwerte für Neuronen zeitlich kodiert sein. Darum wird das Prinzip noch einmal verdeutlicht. Die Feuerzeit $t_i$ jedes Neurons $i$ repräsentiert einen analogen Wert $s_i$. Ist also der Referenzzeitpunkt $T_{ref}$ bekannt und ein $\gamma$ $(0 < \gamma \leq \Delta/2)$ festgelegt, so kann durch die Wahl eines Intervalls $[a_0, a_1]$ jeder Wert $s_i \in \mathbb{R}$ zeitlich kodiert werden (vgl. Abbildung 5.9). Die Kodierung ist sehr einfach. Aus dem Verhältnis

$$\frac{a_1 - a_0}{s_i - a_0} = \frac{T_{ref} - (T_{ref} - \gamma)}{T_{ref} - t_i} = \frac{\gamma}{T_{ref} - t_i}$$

erhält man die Feuerzeit $t_i$:

$$t_i = T_{ref} - \gamma \, \frac{s_i - a_0}{a_1 - a_0} \tag{5.6}$$

Um die Arbeit zu vereinfachen, wird für alle weiteren Beispiele $a_0 = 0$ und $a_1 = 1$ verwendet. Außerdem legen wir ab jetzt ein Kodierungsintervall von 1 ms fest, d. h. $\gamma = 1$ ms. (5.6) vereinfacht sich dadurch zu

$$t_i = T_{ref} - s_i. \tag{5.7}$$

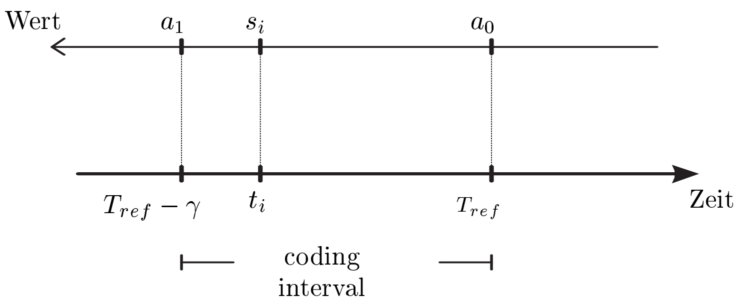



**Abbildung 5.9:** Hier wird das Prinzip der zeitlichen Kodierung noch einmal deutlich gemacht (zur Erklärung siehe Text).

### 5.3.2 Versuchsaufbau

Mit diesen Vorkenntnissen können wir daran gehen, die Aktivierungsfunktion der verwendeten Modellneuronen zu ermitteln.

Es wird ein in Abschnitt 5.1 beschriebenes Modellneuron $j$ verwendet. $j$ besteht also aus $n+4$ Kompartments bei $n$ Inputneuronen. Außer von den $n$ Inputneuronen $s_1, \ldots, s_n$ bekommt das Neuron noch Inputs von einem Referenzneuron $s_0 \equiv 0$ und einem Biasneuron $s_{n+1}$. Der Input des Biasneurons soll ab jetzt immer $s_{n+1} = 1$ sein. Für ein $\kappa$ kann aus (5.5) der Faktor $\lambda$ berechnet werden. Für diesen Aufbau wird, wie in Punkt 5.2.5 beschrieben, der Referenzzeitpunkt $T_{ref}$ ermittelt.
Danach wird für jede Synapse ein zufälliger Gewichtswert in einem Intervall $[w_{min}, w_{max}]$ und ein zufälliger Wert für die gewichtete Summe $\sum_{i=1}^{n+1} w_{j,i} \cdot s_i$ gewählt. Zu den Gewichten $w_{j,i}$ wird ein Inputvektor ($s_i \in [0, \gamma]$) ermittelt, sodaß $\sum_{i=1}^{n+1} w_{j,i} \cdot s_i$ gleich der gewählten gewichteten Summe ist. Gibt es keinen solchen Inputvektor, dann wird ein neuer, zufälliger Gewichtsvektor verwendet. Ist ein Inputvektor gefunden worden, der die Bedingung erfüllt, so muß noch das Gewicht $w_{j,0} = \lambda - \sum_{i=1}^{n+1} w_{j,i} \cdot s_i$ gesetzt werden.

Die drei linken Diagramme in Abbildung 5.10 zeigen die Ergebnisse für $\kappa = 1.8$, $f_{inh} = 12$ und für verschiedene Anzahl von Inputneuronen ($n = 2$, $n = 5$ und $n = 10$). Aus Abschnitt 5.2.3 ist bekannt, daß sich bei größer werdenden exzitatorischen Leitwerten das Verhältnis zu den inhibitorischen Leitwerten verändert. Daher wird für $w_{min} = -1.5 \cdot G_{th}$ und $w_{min} = 1.5 \cdot G_{th}$ gewählt. Dadurch bleibt man im Anfangsbereich der Kurve **A** in Abbildung 5.7. Man erhält einen linearen Zusammenhang zwischen der gewichteten Summe $\sum_{i=1}^{n+1} w_{j,i} \cdot s_i$ und dem Output $T_{ref} - t_j$ des Neurons $j$. Zu beachten ist aber, daß sich keine eindeutige Funktion ergibt. D. h., für zwei unterschiedliche Gewichts– und Inputvektoren $\vec{w}$, $\vec{s}$ und $\vec{w}'$, $\vec{s}'$ mit $\sum w_{j,i} \cdot s_i = \sum w'_{j,i} \cdot s'_i$ werden sich im Allgemeinen zwei leicht unterschiedliche Feuerzeiten $t_j$ und $t'_j$ ergeben.

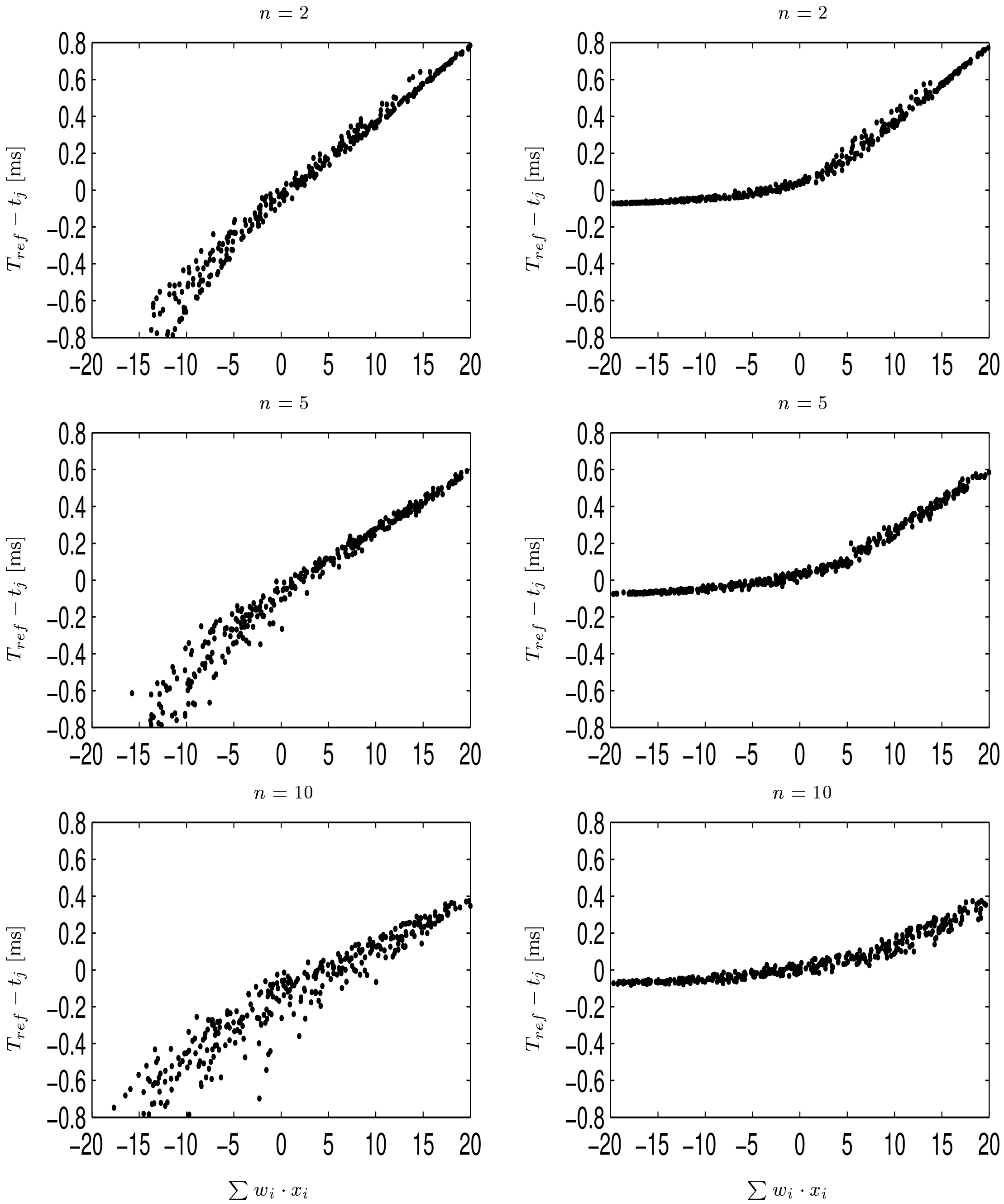



**Abbildung 5.10:** Die Abbildungen zeigen den Verlauf der Aktivierungsfunktion eines Neurons mit $n = 2$, $n = 5$ und $n = 10$ Inputneuronen. Die Wahl der Gewichte und der Inputs ist im Text erläutert. Die Faktoren $\kappa$ und $f_{inh}$ haben die Werte $\kappa = 1.8$ und $f_{inh} = 12$. Die linken Diagramme zeigen den Verlauf der Aktivierungsfunktion ohne Poolneuron. Bei den rechten Diagrammen wurde ein Poolneuron verwendet. Der negative Teil der Funktion wird dadurch abgeschnitten.

### 5.3.3 Poolneuronen

Die Einführung der in Abschnitt 4.4.1 beschriebenen Neuronenpools ist in einem Fall relativ schwierig. Der inhibitorische Neuronenpool, der ein Feuern vor dem Zeitpunkt $T_{ref} - \gamma$ verhindern soll, ist nur schwer zu handhaben. Man kann zwar vor dem Zeitpunkt $T_{ref} - \gamma$ ein negatives Potential (bezogen auf das Membranruhepotential von - 70 mV) am Soma erzeugen, da aber die IPSP's relativ flach auslaufen, bleibt auch nach dem Zeitpunkt $T_{ref} - \gamma$ dieses negative Potential noch zu lange bestehen. Dieses Potential nimmt zu starken Einfluß auf die Feuerzeiten. In dieser Arbeit wurde deswegen kein inhibitorischer Neuronenpool eingesetzt.

Der exzitatorische Neuronenpool ist leicht ins Modell einzubringen. Die Summe der Auswirkungen von vielen Poolneuronen mit exzitatorischen Synapsen zum Soma wird mit einem einzigen Neuron, das eine exzitatorische synaptische Verbindung zum Dendritenstamm hat, modelliert. Das Gewicht dieser Synapse wird extrem hoch gesetzt (hier: 100 $\mu$S). Nachdem die Referenzzeit $T_{ref}$ eines Neurons bekannt ist, läßt man das Poolneuron eine kurze Zeit $\Delta_{pool(E)}$ vor $T_{ref}$ feuern. Für $\Delta_{pool(E)}$ wurde 0,12 ms gewählt. Das resultierende EPSP steigt sehr steil an und bringt dadurch das Neuron kurze Zeit später zum Feuern. Die rechten Diagramme in Abbildung 5.10 zeigen den neuen Verlauf der Aktivierungsfunktion mit dem erläuterten exzitatorischen Poolneuron.

Die Diagramme in Abbildung 5.10 zeigen, daß die Feuerzeit $t_j$ des Modellneurons $j$ annähernd linear von der gewichteten Summe $\sum_{i=1}^{n+1} w_{j,i} \cdot s_i$ abhängt. Durch den Einsatz eines exzitatorischen Poolneurons kann die Aktivierungsfunktion nach unten hin begrenzt werden. Ohne Verwendung eines inhibitorischen Neuronenpools, ist die Aktivierungsfunktion nach oben absolut durch den Wert $T_{ref}^{out} - (T_{ref}^{in} - \gamma)$ beschränkt, da vor $T_{ref}^{in} - \gamma$ keine Inputspikes auftreten.
Bei zu unterschiedlichen Gewichtswerten (sehr große und sehr kleine Gewichtswerte an den verschiedenen Dendritenästen) treten zunehmend nichtlineare Effekte bei der Summation der PSP's im Dendritenbaum auf. Deswegen sollten die Gewichtswerte in einem bestimmten Bereich bleiben (hier bestimmt durch das Intervall $[w_{min}, w_{max}]$).
Ein weiteres Problem stellen die schon erwähnten Unterschiede zwischen EPSP's und IPSP's dar (vgl. Abschnitt 5.2.3). Der Faktor $f_{inh}$ kann diesen Unterschied nur bedingt ausgleichen, da er eigentlich nur für ein bestimmtes positives Gewicht genau stimmt, und nicht für das ganze Intervall $[0, w_{max}]$ der positiven Gewichte. Um das Problem zu verdeutlichen wurde für den Fall $n = 5$ die Aktivierungsfunktion nocheinmal ermittelt, wobei aber nur positive Gewichte im Intervall $[0, w_{max}]$ zugelassen waren. Abbildung 5.11 zeigt den Vergleich zwischen den beiden Versuchen.

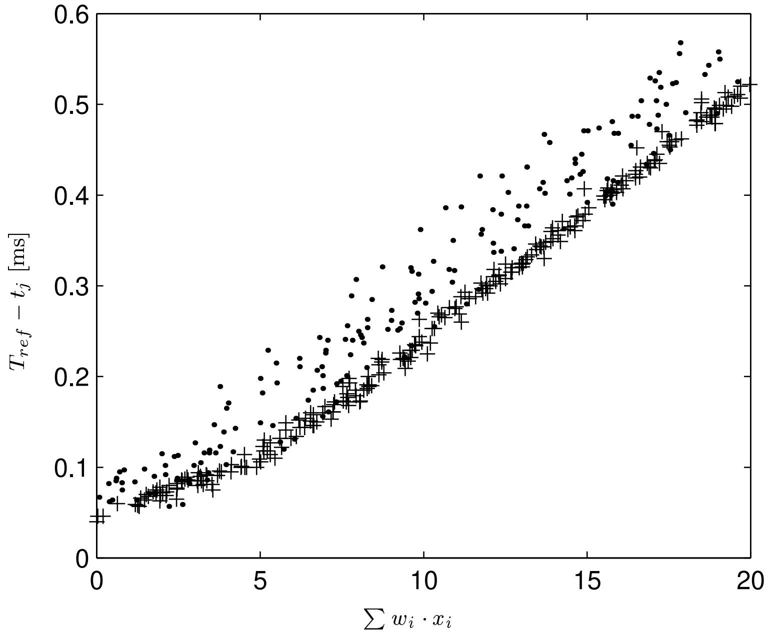



**Abbildung 5.11:** Die Abbildung zeigt den Vergleich der Aktivierungsfunktionen, wobei im ersten Fall positive und negative gleichverteilte Gewichte im Intervall $[w_{min}, w_{max}]$ und im zweiten Fall nur positive gleichverteilte Gewichte im Intervall $[0, w_{max}]$ zugelassen waren. In beiden Fällen wurde ein Modellneuron mit 5 Inputneuronen ($n = 5$) und den Parametern $\kappa = 1.8$ und $f_{inh} = 12$ verwendet. Die mit $\cdot$ markierten Punkte sind die Ergebnisse aus dem ersten Versuch mit $w_{min} = -1.5 \cdot G_{th}$ und $w_{max} = 1.5 \cdot G_{th}$ (vgl. Abbildung 5.10 Mitte rechts). Die mit + markierten Punkte ergeben sich im zweiten Fall, wieder mit $w_{max} = 1.5 \cdot G_{th}$. Die Streuung der Feuerzeiten $t_j$ ist im zweiten Fall deutlich geringer.

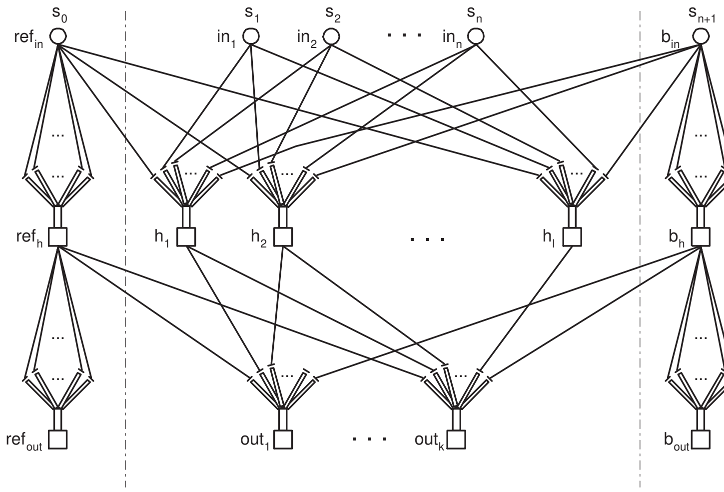
## 5.4 Vergleich zwischen künstlichen Netzen und dem Kompartmentmodell

Es soll nun ein Vergleich zwischen einem künstlichen Netzwerk und einem NFN, bestehend aus Kompartmentmodellen von Neuronen, gemacht werden. Als Vorlage zur Erstellung dieses NFN dient das Konstruktionsprinzip aus Abschnitt 4.4. Die künstlichen Netzwerke werden mit der Neural Network Toolbox von MATLAB® berechnet. Als Aktivierungsfunktion wird dabei immer eine positiv lineare Funktion eingesetzt, da sie den Aktivierungsfunktionen in Abbildung 5.10 (rechts) am ähnlichsten ist. Sie hat die Form

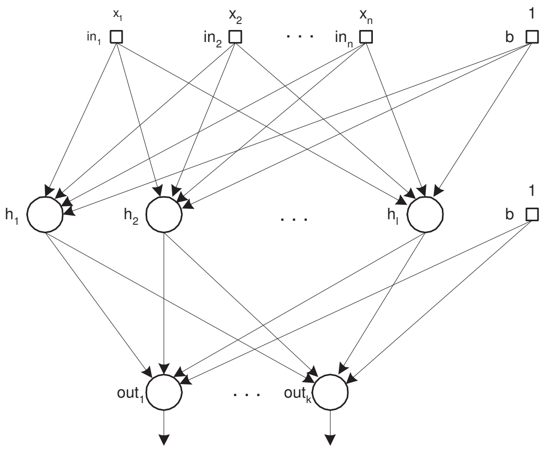
$$\varphi(x) = \begin{cases} 0 & \text{, wenn} \quad x < 0 \\ x & \text{, sonst} \end{cases} \tag{5.8}$$

und wird in MATLAB® mit *poslin(x)* bezeichnet.

### 5.4.1 Aufbau des Netzwerks in GENESIS




(a)




(b)

**Abbildung 5.12:** Bild (a) zeigt das Schema eines Netzwerks in GENESIS mit einer verborgenen Schicht. Die Neuronen in der Inputschicht feuern zu den durch die Inputs $s_1, \ldots, s_n$ gegebenen Zeitpunkten $t_{s_i} = T_{ref}^{in} - s_i$. Der Input des Referenzneurons ist $s_0 \equiv 0$. Der Input des Biasneurons soll bei allen weiteren in dieser Arbeit vorgestellten Beispielen den Wert 1 haben. Die Neuronen links und rechts der strichlierten Linien dienen nur zur Berechnung der Referenzzeitpunkte bzw. der Biaswerte der einzelnen Schichten. Die zusätzlichen exzitatorischen Poolneuronen wurden hier nicht eingezeichnet. In Bild (b) ist das Schema eines künstlichen NN dargestellt. Jedes Gatter in diesem Netzwerk hat als zusätzlichen Input den Biaswert 1.

Wie erfolgt nun der Aufbau eines Netzwerks in GENESIS? Für jedes Gatter des künstlichen Netzwerks wird hier das Kompartmentmodell eines Neurons eingesetzt. Alle Modellneuronen in einer Schicht des Netzwerks haben gleich viele Dendritenäste, da sie gleich viele Inputs aus der Vorgängerschicht erhalten. Der Referenzzeitpunkt eines Neurons hängt bei einem gegebenen Faktor $\kappa$ nur von der Anzahl der Dendritenäste ab. Daher ist der Referenzzeitpunkt für alle Neuronen in einer Schicht gleich (vgl. Abschnitt 5.2.5). Nach dem gleichen Prinzip wie die Referenzzeitpunkte werden auch die Biaswerte für jede Schicht ermittelt. Für einen gegebenen Biaswert $s_{n+1}$ feuert das Biasneuron $b_{in}$ zur Zeit $t_{b_{in}} = T^{in}_{ref} - s_{n+1}$ (vgl. (5.7)). Da die Bias– und Referenzneuronen jeder Schicht gleich aufgebaut sind, feuern die Biasneuronen in den folgenden Schichten zum Zeitpunkt $t_{b_l} = T^l_{ref} - s_{n+1}$ ($l$ beizeichnet die jeweilige Schicht des Netzwerks, also $l = h_1, h_2, \ldots, out$). Das bedeutet, daß die Neuronen in jeder Schicht den Wert $s_{n+1}$ als zusätzlichen Bias–Inputwert erhalten. Bei allen folgenden Beispielen wird $s_{n+1} = 1$ gesetzt.

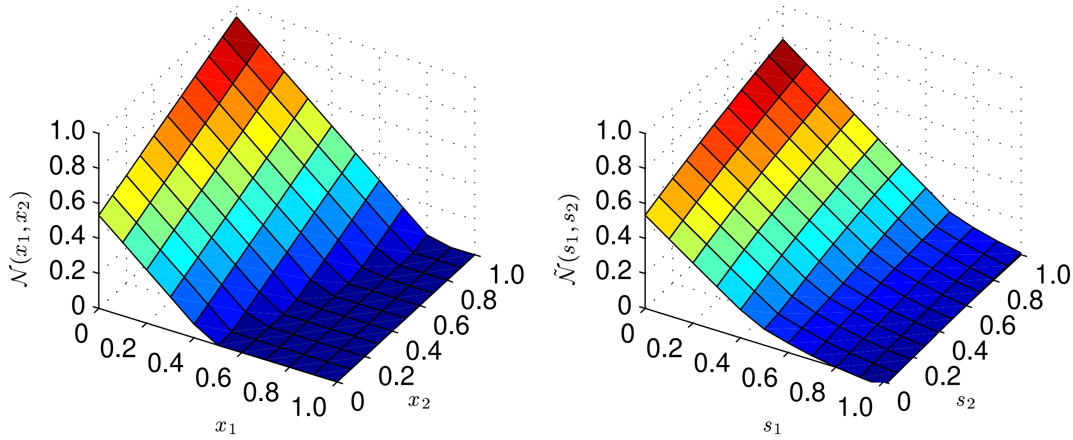
Abbildung 5.12(a) zeigt, wie man sich das in GENESIS erstellte Netzwerk vorstellen kann. Die zusätzlichen exzitatorischen Poolneuronen (vgl. Abschnitt 5.3.3 wurden in diesem Schema nicht eingezeichnet. Im Vergleich dazu ist in Abbildung 5.12(b) ein künstliches Netzwerk dargestellt. Damit im Folgenden diese beiden Netzwerke nicht verwechselt werden, sind die verwendeten Notationen in Tabelle 5.4.1 zusammengefaßt.

| | Netzwerk in GENESIS | Netzwerk in MATLAB® |
|---|---|---|
| Inputs | $s_1, \ldots, s_n$ | $x_1, \ldots, x_n$ |
| Gewichte | $w_{j,i}$ | $r_{j,i}$ |
| Output | $\tilde{\mathcal{N}}(s_1, \ldots, s_n)$ | $\mathcal{N}(x_1, \ldots, x_n)$ |

### 5.4.2 Ein einzelnes Neuron

Als erstes wird ein einzelnes Gatter mit 2 Inputs betrachtet. Mit MATLAB® wurden zufällig Gewichte gewählt und die Form des Gatteroutputs für $x_i \in [0, 1]$ berechnet (Abbildung 5.13(a)). Dann wird mit GENESIS ein Modellneuron mit 2 Inputs generiert und mit zufällig gewählten Gewichten initialisiert. Nachdem das Modellneuron 150 Epochen lang mit den Zielwerten des künstlichen Gatters trainiert wurde, zeigt es einen Output, der mit dem des künstlichen Gatters fast identisch ist. Da der Knick in der Aktivierungsfunktion des Modellneurons nicht so scharf ist, kann auch der scharfe Knick des vorgegebenen Gatters nicht nachgeahmt werden. Das Training erfolgt nach dem *Backpropagation* Algorithmus (vgl. Abschnitt 3.2.1). Der Einsatz des *Backpropagation* Algorithmus in GENESIS wird in Abschnitt 5.5 noch genauer erklärt.

Wie man sieht, sind die Berechnungen eines solch einfachen Gatters mit einem feuernden Neuron ebenfalls durchführbar und erlernbar.




(a) (b)

**Abbildung 5.13:** Versuch ein einzelnes Gatter mit 2 Inputs ($n = 2$) mit einem Modellneuron in GENESIS zu simulieren. Die Parameter sind: $\kappa = 1.8$, $f_{inh} = 12$; Initialisierung: $w_{min} = -0.5 * G_{th}$, $w_{max} = 1.5 * G_{th}$. Diagramm (a): Output $\mathcal{N}(x_1, x_2)$ des Gatters. Die Gewichte des Gatters wurden zufällig gewählt. Diagramm (b): Output des trainierten GENESIS Modellneurons.

### 5.4.3 Ein kleines Netzwerk

Im zweiten Versuch werden die Outputs eines kleinen Netzwerks in MATLAB® und GENESIS verglichen. Die Netzwerke haben dabei 2 Inputneuronen, 3 verborgene Neuronen und ein Outputneuron. Wieder werden in MATLAB® zufällig Gewichte gewählt und der Output des künstlichen Netzes berechnet. Die Gewichte $r_{j,i}$ des künstlichen Netzwerkes werden mit $G_{th}(n+2)$ multipliziert, und daraus die Gewichte des NFN berechnet, d. h.

$$w_{j,i} = G_{th}(n+2) \cdot r_{j,i}. \tag{5.9}$$

Dann wird der Output des GENESIS Netzwerks mit diesen Gewichten $w_{j,i}$ ermittelt und mit dem Output des künstlichen Netzes verglichen.

Abbildung 5.14 zeigt die Ergebnisse für 3 unterschiedliche Initialisierungen des künstlichen Netzwerks. Auch hier sind deutlich die Ähnlichkeiten der Outputs zu erkennen.

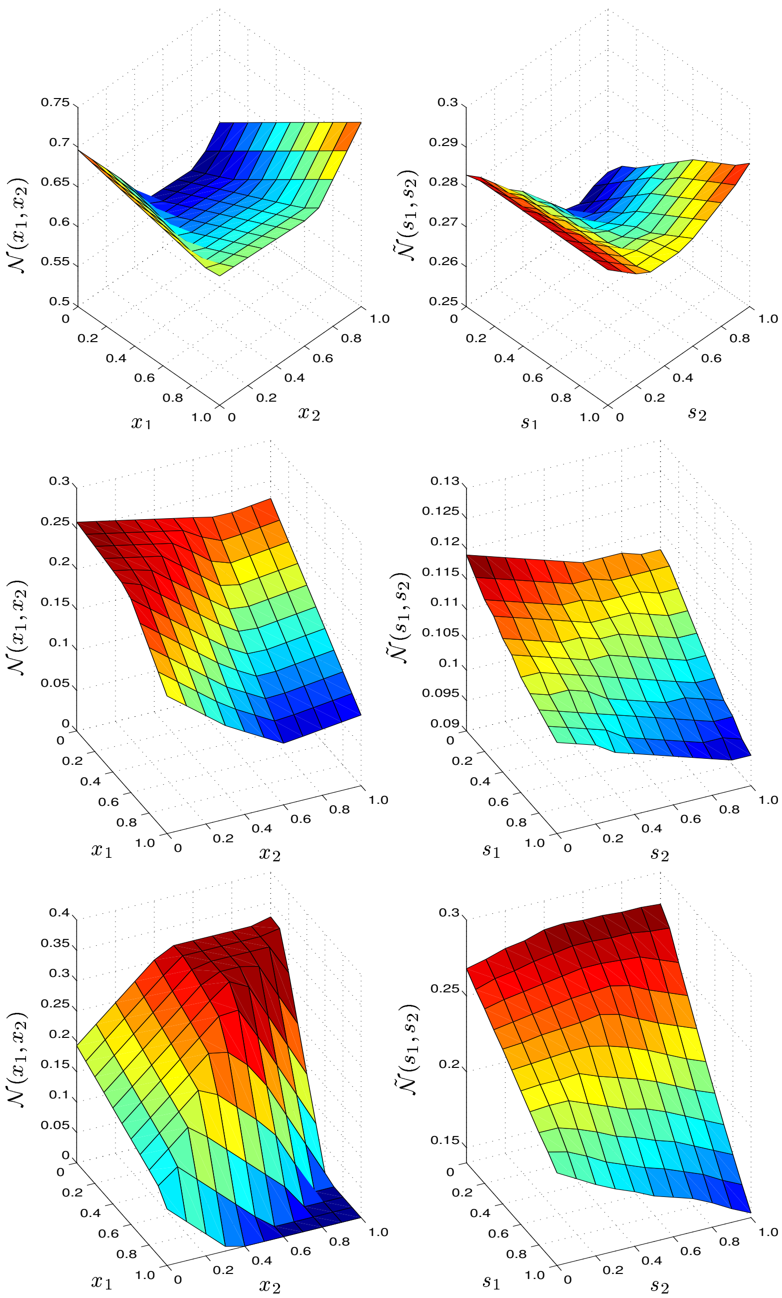



**Abbildung 5.14:** Vergleich zwischen einem künstlichen Netzwerk und einem Netzwerk feuernder Neuronen in GENESIS. Die Netzwerkstruktur besteht bei beiden Netzwerken aus 2 Inputneuronen, 3 verborgenen Neuronen und einem Outputneuron. Die linken Diagramme zeigen jeweils den Output $\mathcal{N}(x_1, x_2)$ des künstlichen Netzwerks mit zufällig gewählten Gewichten $r_{j,i}$. Die rechten Diagramme zeigen den Output $\tilde{\mathcal{N}}(s_1, s_2)$ des Netzwerks in GENESIS. Die Gewichtswerte $w_{j,i}$ wurden mit (5.9) aus den Gewichten $r_{j,i}$ berechnet. Die Parameter des NFN sind: $\kappa = 1.8$, $f_{inh} = 12$.

## 5.5 Lernen in Netzwerken feuernder Neuronen

Unser Ziel war es, auf ein NFN in GENESIS den *Backpropagation* Algorithmus anzuwenden. Dazu müssen die $\delta$'s für alle Neuronen im Netzwerk berechnet werden (vgl. Abschnitt 3.2). Um die Formeln (3.10) und (3.11) anwenden zu können, benötigen wir noch die Ableitung der Aktivierungsfunktion der Neuronen.
Da die Aktivierungsfunktion $\varphi(.)$ eines Neuronenmodells in GENESIS keine Funktion ist (vgl. Abschnitt 5.3), kann $\varphi'(.)$ nicht analytisch bestimmt werden. Wir nehmen hier als Ableitung eine Sprungfunktion

$$\varphi'(.) = \begin{cases} 1 & \text{, falls } \sum w_i \cdot s_i > 0 \\ 0 & \text{, sonst} \end{cases} \tag{5.10}$$

an (siehe Abbildung 5.15). Für große gewichtete Summen $\sum w_i \cdot s_i$ flacht die Aktivierungsfunktion wieder ab. Dort stimmt die gewählte Funktion $\varphi'(.)$ nicht mehr. Solch hohe gewichtete Summen werden aber bei den betrachteten Netzwerken nicht erreicht.

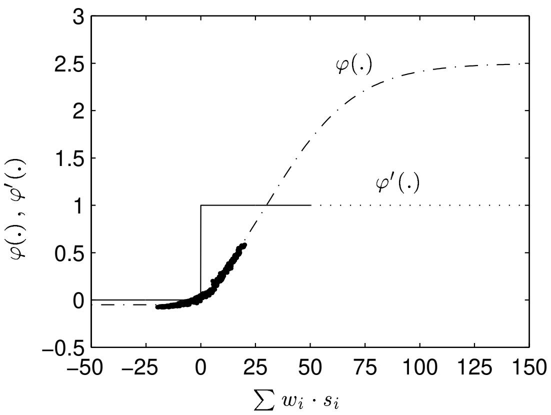



**Abbildung 5.15:** Darstellung der Aktivierungsfunktion $\varphi(.)$ eines Modellneurons in GENESIS und der gewählten Ableitung $\varphi'(.)$. Für $-20 \leq \sum w_i \cdot s_i \leq 20$ sind dieselben Punkte wie in Abbildung 5.10 aufgetragen. Die strichpunktierte Linie zeigt, wie die Aktivierungsfunktion für kleinere bzw. größere gewichtete Summen weiter verlaufen würde. Die durchgezogene Linie zeigt die gewählte Funktion $\varphi'(.)$. Im strichlierten Bereich der Funktion nähert die gewählte Ableitung $\varphi'(.)$ nicht mehr die tatsächliche Ableitung der Aktivierungsfunktion an. Diese großen gewichteten Summen $\sum w_i \cdot s_i$ werden aber bei den folgenden Versuchen nicht erreicht.

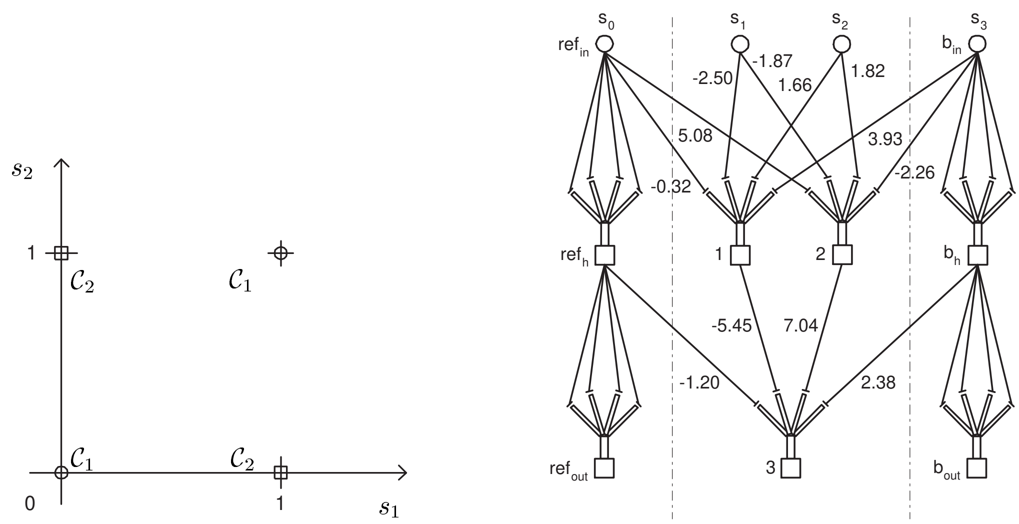
Nachdem man eine Fehlerfunktion festgelegt hat — wir wählen hierfür eine quadratische Fehlersumme (siehe Gleichung (3.14)) — kann man die Gewichtsänderungen im Netzwerk berechnen. Im Folgenden werden zwei Beispiele angeführt, in denen kleine NFN in GENESIS mit dem *Backpropagation* Algorithmus trainiert wurden.

### 5.5.1 Das XOR–Problem

Wir wollen das XOR–Problem betrachten, das auch als spezieller Fall eines Klassifikationsproblems angesehen werden kann. Die XOR–Funktion ist von einem einzelnen Gatter nicht mehr berechenbar, da die beiden auftretenden Klassen nicht linear separierbar sind. Dies kann sehr leicht mit dem Diagramm in Abbildung 5.16(a) veranschaulichet werden. Die Wahrheitstafel der XOR–Funktion sieht folgendermaßen aus:

| $s_1$ | $s_2$ | $s_1 \oplus s_2$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

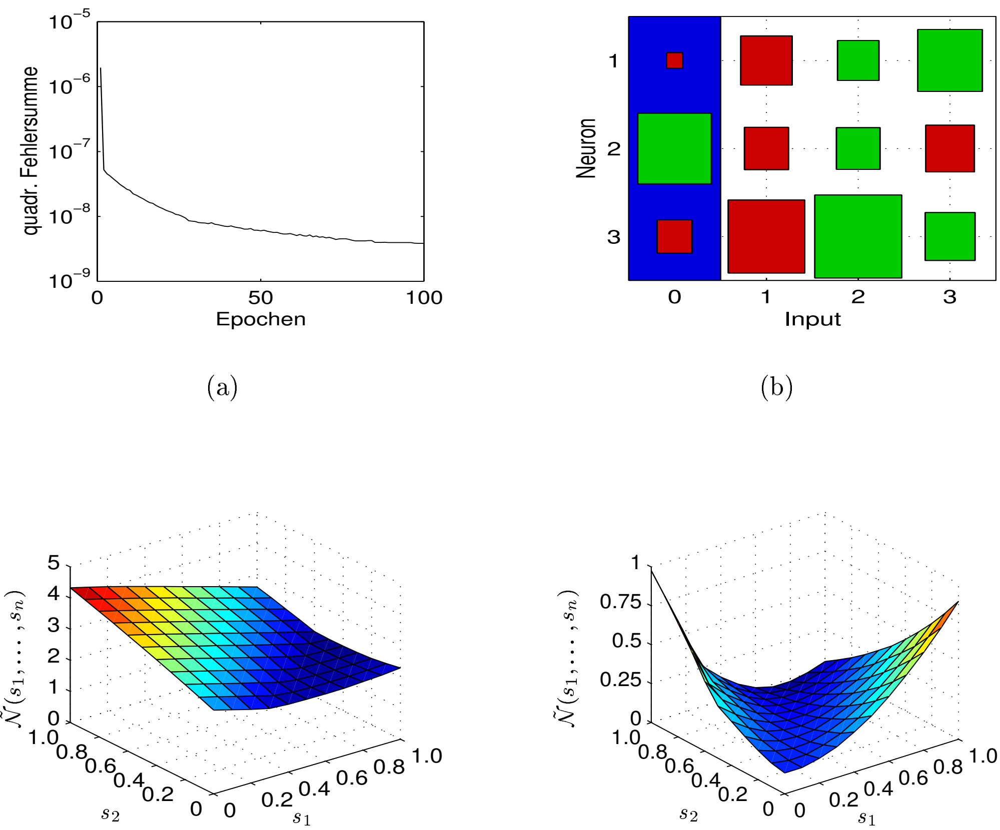
Ein Netzwerk mit zwei verborgenen Neuronen und einem Outputneuron wird mit diesen vier Trainingsbeispielen trainiert (siehe Abbildung 5.16(b)). Um schon vor Beginn des Trainings in die Nähe einer Lösung zu gelangen, wird für ein




(a) (b)

**Abbildung 5.16:** Links sieht man die Klasseneinteilung, die durch die XOR-Funktion gegeben ist. Man kann erkennen, daß die beiden Klassen nicht durch eine Gerade trennbar sind. Rechts ist die Netzwerkstruktur abgebildet. Mit den angegebenen Gewichten erhält man den in Abbildung 5.17(d) gezeigten Output. Die Gewichte an den Referenzneuronen (links) bzw. Biasneuronen haben alle den Wert 0.69 nS. Alle anderen Gewichte sind ebenfalls in nS angegeben.

künstliches Netzwerk mit der gleichen Netzwerkstruktur und einer positiv linearen Aktivierungsfunktion eine Lösung ermittelt. Die Gewichtswerte dieser Lösung werden dann, wie schon im vorigen Abschnitt, in Gewichte des NFN umgerechnet ($w_{j,i} = G_{th}(4) \cdot r_{j,i}$). Dann wird das NFN nocheinmal mit dem *Backpropagation* Algorithmus 100 Epochen lang mit den 4 Trainigsbeispielen aus der XOR–Wahrheitstafel weitertrainiert. So erreicht man mit einem vertretbaren Zeitaufwand ein Netzwerk in GENESIS bestehend aus Modellneuronen, das das XOR–Problem löst. Das Ergebnis und die Parameter des Netzes sind aus Abbildung 5.17 zu entnehmen. Die dazugehörigen Gewichtswerte der Lösung in nS sind in Abbildung 5.16(b) angegeben.

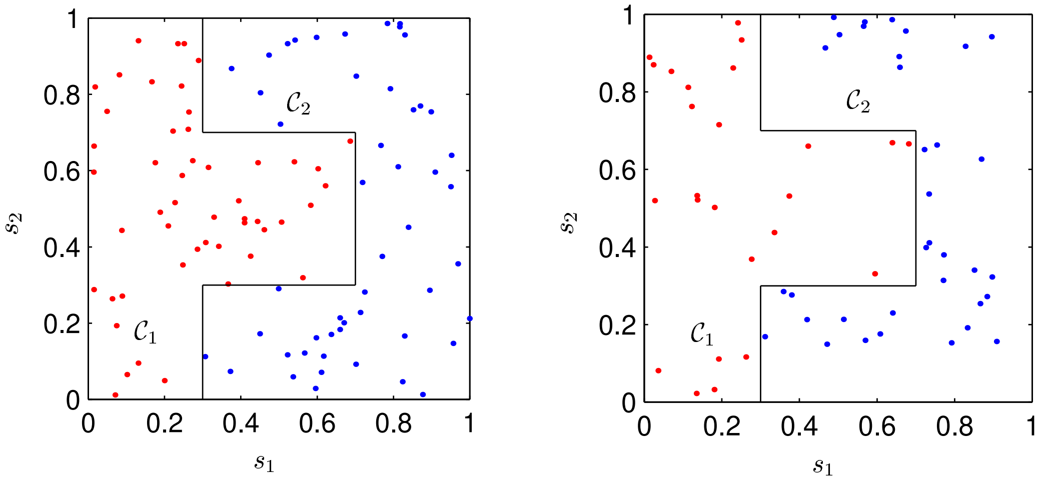



(a) (b)

(c) (d)

**Abbildung 5.17:** Ein NFN mit 2 verborgenen Neuronen und einem Outputneuron wird trainiert, um das XOR–Problem zu lösen. Das Netz hat folgende Parameter: $\kappa = 2.0$, $f_{inh} = 12$. Diagramm (a): quadratische Fehlersumme beim Training. Diagramm (b): Hinton Diagramm der in Abbildung 5.16(b) angegebenen Gewichte. Input 0 ist jeweils der Input des Referenznerons und Input 3 der Input des Biasneurons. Diagramm (c): Output des Netzes nach Umrechnung der Gewichte aus dem künstlichen Netzwerk. Diagramm (d): Output des NFN nach 100 Trainingsepochen.

## 5.5.2 Ein Klassifizierungsproblem




(a) (b)

**Abbildung 5.18:** Die Trainingsmenge (Diagramm (a)) und die Testmenge (Diagramm (b)) für das Klassifizierungsproblem.

Ein NFN soll dazu benützt werden, ein zweidimensionales Klassifizierungsproblem zu lösen. Das Einheitsquadrat wird in zwei nichtkonvexe Regionen $\mathcal{C}_1$ und $\mathcal{C}_2$ unterteilt. Für Punkte, die in der Region $\mathcal{C}_1$ liegen, soll das Netzwerk den Output 1 liefern, für Punkte in $\mathcal{C}_2$ den Wert 0. Die gewählte Netzwerkstruktur hat 2 Inputneuronen, 7 verborgene Neuronen und 1 Outputneuron. Die anderen Parameter des Netzwerkes sind $\kappa = 2.0$, $f_{inh} = 12$ und $\gamma = 1$ ms. Abbildung 5.18 links zeigt die Trainingsmenge, mit der das Netzwerk trainiert wurde. In Abbildung 5.18 rechts ist die Testmenge zu sehen. Beide Mengen bestehen aus zufällig gleichverteilten Punkten $(x_1, x_2) \in [0, 1] \times [0, 1]$. Es sind jeweils 50 Punkte in $\mathcal{C}_1$ und $\mathcal{C}_2$ der Trainingsmenge. Bei den Testbeispielen sind 25 Punkte aus $\mathcal{C}_1$ und 35 Punkte aus $\mathcal{C}_2$.

Das Netzwerk wird mit zufälligen Gewichten aus $[-0.5 \cdot G_{th}, 1.5 \cdot G_{th}]$ initialisiert. Die Klassifizierung der Testbeispiele nach der Initialisierung des Netzwerks ist in Abbildung 5.19 links abgebildet. Die eingekreisten Punkte, das sind hier alle 25 Punkte in $\mathcal{C}_1$, wurden dabei vom Netz falsch klassifiziert. Nach 200 Trainingsepochen werden dem Netz nocheinmal die Testbeispiele präsentiert. Das Netzwerk klassifiziert nur mehr 7 Testbeispiele falsch. Die falsch klassifizierten Beispiele sind in Abbildung 5.19 rechts zu erkennen. Das Training wurde nach 200 Epochen wegen zu langer Rechenzeit abgebrochen.

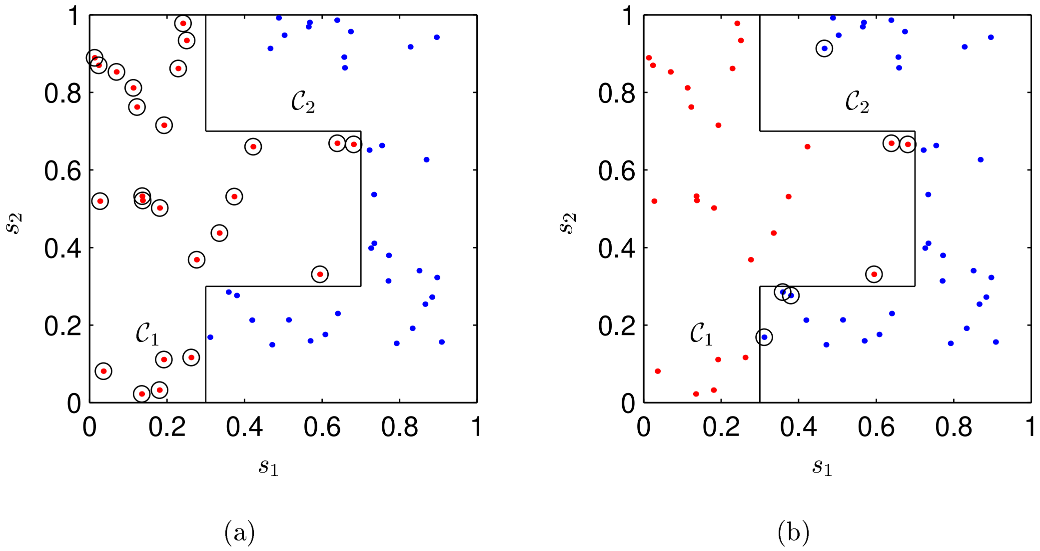



**Abbildung 5.19:** Klassifizierung der Testbeispiele durch das Netzwerk. Die analogen Outputwerte des Netzwerks werden dabei auf 0 bzw. 1 gerundet. Diagramm (a): Nach der Gewichtsinitialisation werden alle 25 Punkte in $\mathcal{C}_1$ falsch klassifiziert (eingekreiste Punkte). Diagramm (b): Nach 200 Trainingsepochen macht das Netzwerk nur noch bei 7 Testbeispielen einen Fehler.

Bei einigen weiteren Beispielen (Klassifizierungsprobleme und Funktionsapproximationen) konnten keine zufriedenstellenden Lösungen erzielt werden. Im Wesentlichen treten bei den Versuchen zwei Probleme auf:

- Es ergeben sich sehr lange Simulationszeiten. Je nach Größe des Netzwerkes und der Trainingsmenge können Trainingszeiten im Bereich von einigen Stunden bis zu einigen Tagen auftreten.

- Die Form der Aktivierungsfunktion ist für die Lerneigenschaften relativ ungünstig. Es kommt sehr stark auf die Initialisierung des Netzwerkes an, ob man durch Anwendung der Lernregel überhaupt in die Nähe einer akzeptablen Lösung kommen kann.
  Durch Einsatz inhibitorischer Neuronenpools, die die Aktivierungsfunktion der Modellneuronen auch nach oben hin beschränken (vgl. Abschnitt 5.3.3), könnten die Lerneigenschaften eines Netzwerkes wahrscheinlich deutlich verbessert werden.

## 5.6 Zusammenfassung

Obwohl die in dieser Arbeit verwendeten Neuronenmodelle nur sehr vereinfachte Sachverhalte biologischer Neuronen darstellen können, ist es prinzipiell möglich, mit ihnen komplexe Funktionen $F : [0, \gamma]^n \rightarrow [0, \gamma]^k$ in temporaler Kodierung zu berechnen. Es ist davon auszugehen, daß biologische Neuronen eine viel größere Berechnungskraft als das verwendete Modell besitzen. Relativ genau wurden nur die Generierung des Aktionspotentials durch das HH–Modell (vgl. Abschnitt 2.3) und die Form der PSP's biologischer Neuronen in die Simulation eingebracht. Nicht beachtet wurden hingegen z. B. die weite Verzweigungsstruktur des Dendritenbaumes, verschieden lange Verzögerungszeiten zwischen den Neuronen oder aktive Verstärkungsprozesse im Dendriten.

Die in der Theorie angeführte Konstruktion eines NFN aus einem künstlichen neuronalen Netzwerk ist in der Praxis nicht so einfach. Es müssen einige Faktoren berücksichtigt werden, die theoretisch einfach behandelt werden können, in der Simulation jedoch einige Schwierigkeiten bereiten. Laut Konstruktion wird für den Faktor $\lambda$ nur gefordert, daß $\lambda > 0$ und $\lambda$ genügend groß gewählt werden muß. In der Simulation ist der Bereich für die Wahl von $\lambda$ viel mehr eingeschränkt. Weiters haben der multiplikative Faktor $f_{inh}$ für die negativen Gewichte und die Anzahl der Dendritenäste eines Neurons Auswirkungen auf die charakteristischen Eigenschaften des Modellneurons.

Verwendet man realistische Zeitkonstanten um PSP's zu simulieren, so wird der Faktor $\gamma$ sehr klein (im Bereich von 1 – 2 ms). Ein Ansatz, um größere $\gamma$'s zu erreichen, ist in [15] geschildert. Mit mehreren zeitverzögerten Synapsen an einem Dendritenast, wird eine längere lineare Anfangsphase der PSP's am Soma erreicht. So kann der Faktor $\Delta$, und damit auch $\gamma$, vergrößert werden.

Der Einsatz des *Backpropagation* Algorithmus zum Trainieren biologischer Nervensysteme scheint unrealistisch zu sein. Das Vorhandensein eines „Lehrers“, der die richtigen Zielwerte vorgibt, ist unwahrscheinlich. Andere Lernregeln, wie z. B. die Hebb–Lernregel, können viel besser biologisch motiviert werden (siehe dazu [14, Abschnitt 4] oder [18]).
In [16] wird z. B. eine „Lernfunktion“ $L(\Delta t)$ angewendet um RBF Netzwerke zu trainieren. Die Gewichte der Synapsen werden dabei im Prinzip aufgrund der Zeitdifferenz $\Delta t = t_{pre} - t_{post}$ des Feuerns eines präsynaptischen und des postsynaptischen Neurons verändert. Da ein Aktionspotential auch durch den Dendritenbaum zurückpropagiert wird, sind an der Synapse die Informationen über beide Feuerzeiten vorhanden. Die Funktion $L$ wird dabei so gewählt, daß bei kleinen $\Delta t$'s das Gewicht der Synapse vergrößert wird und bei großen $\Delta t$'s verringert. Diese Art der Gewichtsänderungen ist biologisch viel realistischer.

Zuguterletzt soll noch auf das Problem langer Simulationszeiten hingewiesen werden. Die in den vorigen Abschnitten beschriebenen Ergebnisse wurden mit einem Pentium II 300MHz Prozessor erzielt. Für kleine Netzwerke und kleine Trainingsmengen, wie z. B. beim XOR-Problem, liegen die Trainingszeiten im Bereich von einigen Stunden. Bei größeren Netzwerken (zwischen 10 und 20 Neuronen) und größeren Trainingsmengen (ca. 100 Trainingsbeispiele) steigen die Trainingszeiten stark an. Die Zeit, um einige wenige Trainingsepochen zu simulieren, liegt dann schon im Bereich einer Stunde. Aus diesem Grund konnten größere Netzwerke nicht mit einem vernünftigen Aufwand trainiert werden.

# Anhang A

# Das Programmpaket GENESIS

Es gibt mittlerweile schon einige Softwarepakete, mit denen sich Kompartmentmodelle am Computer simulieren lassen. Die in Kapitel 5 dieser Arbeit beschriebenen Ergebnisse wurden mit dem Simulator GENESIS (GEneral NEural SImulation System) erzielt. GENESIS ist geeignet, um sowohl komplexe Modelle einzelner Neuronen, als auch große Netzwerke zu simulieren.

Das Programmpaket wurde am Caltech Institut entwickelt. Die aktuelle Version (derzeit Version 2.1 vom August 1997) wird kostenlos angeboten und ist via *anonymous ftp* vom Server *genesis.bbb.caltech.edu* erhältlich. GENESIS ist in C geschrieben und läuft auf verschiedenen Unix–Plattformen (SunOS 4.x, Sun Solaris 2.x, DECstation (Ultrix), Silicon Graphics (Irix 4.0.1 und aufwärts), x86 PC (Linux oder FreeBSD)). Für die graphische Oberfläche wird das X-Windows System (Version X11R4, X11R5, oder X11R6) benötigt. Weitere Informationen sind auf der GENESIS–Homepage *http://www.bbb.caltech.edu/GENESIS* erhältlich.

Das Buch *The Book of GENESIS* [6] ist als Einführung zur Arbeit mit GENESIS zu empfehlen. Der erste Teil des Buches behandelt Konzepte der Modellierung biologischer neuronaler Netze. Im zweiten Teil wird die Erstellung eigener Simulationen mit der GENESIS Script Sprache vorgestellt. Im Programmpaket selbst ist außerdem eine Hypertext–Hilfe inkludiert [3].

# Literaturverzeichnis


[1] Abeles B., Bergmann H., Margalit E. and Vaadia E. *Spatiotemporal firing patterns in the frontal cortex of behaving monkeys.* In J. of Neurophysiology, vol. 70, pages 1629–1638, 1993.

[2] Anderson John A., Rosenfeld Edward, eds., *Neurocomputing: Foundations of Research*, MIT Press, Cambridge, 1988.

[3] Dave Beeman and Upi Bhalla. *GENESIS Reference Manual.* Calfornia Institute of Technologie, 1997. Included in the GENESIS distribution.

[4] Bishop Christopher M. *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.

[5] Bischof Horst. *Pyramidal Neural Networks*, Lawrence Erlbaum Associates, Inc., 1995.

[6] BowerJames M. and Beeman David. *The Book of Genesis: Exploring Realistic Neural Models with the GEneral NEural SImulation System*, second edition, Springer-Verlag, Inc. Published by TELOS, New York, 1998.

[7] Hagan Martin T., Demuth Howard B., Beale Mark *Neural Network Design*, PWS Publishing Company, 1996.

[8] Haykin Simon. *Neural Networks - A Comprehensive Foundation*, Macmillan College Publishing Company Inc., New York, 1994.

[9] Johnston Daniel and Wu Samuel Miao–Sin. *Foundations of Cellular Neurophysiology*, MIT Press, 1995.

[10] Kahle W., Leonhardt H., Platzer W., *Taschenatlas der Anatomie: Nervensystem und Sinnesorgane* Band 3, Thieme Verlag, 1991.

[11] Kandel Eric R., Schwartz James H., Jessell Thomas M. *Principles Of Neural Science* third edition, Prentice-Hall International, 1991.

[12] Koch Christof. *Biophysics of Computation: Information Processing in Single Neurons*, Oxford University Press, 1999.

[13] Koch Christof and Segev Idan, editors. *Methods in Neuronal Modeling*, MIT Press, 1992.

[14] Maass W., *Fast Sigmoidal Networks via Spiking Neurons*. In Neural Computation 9, pages 279–304, 1997.

[15] Maass W. and Natschläger T. *Networks of spiking neurons can emulate arbitrary Hopfield nets in temporal coding*. In Network: Computation in Neural Systems, 8(4):355-372, 1997.

[16] Natschläger T. and Ruf B. *Spatial and temporal pattern analysis via spiking neurons*. Network: Computation in Neural Systems, 9(3):319-332, 1998.

[17] Rojas Raúl *Theorie der neuronalen Netze: Eine systematische Einführung*, Springer–Verlag, 1993.

[18] Ruf B. and Schmitt M. *Hebbian learning in networks of spiking neurons using temporal coding*. In Biological and artificial computation: From neuroscience to technology, Mira J., Moreno-Diaz R. and Cabestany J., editors, volume 1240 of Lecture Notes in Computer Science, pages 380-389. Springer, Berlin, 1997.

[19] Ruf B.. *Networks of spiking neurons can compute linear functions using action potential timing*. In Lindblad T., editor, Proc. of the VI-Dynn 98 conference. Spie, 1998.

[20] Rumelhart David E., McClelland James L. and the PDP Research Group *Parallel Distributed Processing: Explorations in the Micorstructure of Cognition*, Volume 1, MIT Press, 1986.

[21] Schmidt Robert F. (Hrsg.). *Neuro- und Sinnesphysiologie*, Springer-Verlag, Berlin Heidelberg, 1993.

[22] Sejnowski T. J. *Time for a new neural code?*. In Nature, vol. 376, pages 21–22, 1995.

[23] Sheperd Gordon M. *Neuropiology*, chapter 5. Oxford University Press, 1994.

[24] Thews Gerhard, Mutschler Ernst, Vaupel Peter. *Anatomie, Physiologie, Pathophysiologie des Menschen*, Wissenschaftliche Verlagsgesellschaft mbH, Stuttgart, 1991.

[25] Thorpe, S. J. and Imbert, M. *Biological constraints on connectionist modelling*. In Connectionism in Perspective, Pfeifer R., Schreter Z., Fogelman-Soulié F. and Steels L., editors, Elsevier (North-Holland), 1989.

[26] Thorpe, S. J. and Gautrais J. *Rapid visual processing using sipke asynchrony.* In Advances in Neural Information Processing Systems, vol. 9, pages 901–907, MIT Press, Cambridge, 1997.

[27] Werbos P.J., *Beyond regression: New tools for prediction and analysis in the behavioral sciences* Ph.D. Thesis, Harvard University, Cambridge, MA, 1974.

[28] White Halbert with Gallant A.R., Hornik K., Stinchcombe M. and Wooldridge J. *Artificial Neural Networks: Approximation and Learning Theory*, Biddles Ltd., Guildford, 1992.