# HIERARCHISCHER TEMPORALER SPEICHER

**und**

## HTM-basierte kortikale Lernalgorithmen

**VERSION 0.2.1, SEPTEMBER 12, 2011**



**Deutsche Übersetzung von Ingmar Baetge**
**ingmar (dot) baetge (at) googlemail (dot) com**

**Bitte zuerst lesen!**
Dies ist eine Entwurfsversion dieses Dokuments. Es sei erwähnt, dass einige Teile noch fehlen.

**In diesem Dokument ENTHALTEN:**
Dieses Dokument beschreibt detailliert die neuen Algorithmen für Lernen und Vorhersage, die von Numenta entwickelt wurden. Die Algorithmen sind ausreichend beschrieben, so dass ein Entwickler sie verstehen und umsetzen kann, falls gewünscht. Das Dokument beginnt mit einem einführenden Kapitel. Wenn Sie die Arbeit von Numenta bereits kennen und einige unserer *white paper* gelesen haben, dann werden ihnen die Inhalte des Einführungskapitels vertraut sein. Die übrigen Inhalte sind neu.

**NICHT in diesem Dokument ENTHALTEN:**
Es gibt einige Themen im Zusammenhang mit der Implementierung der neuen Algorithmen, die noch nicht in diesen frühen Entwurf aufgenommen wurden.

- Obwohl die meisten Aspekte der Algorithmen in Form von Software umgesetzt und getestet wurden, sind keine Ergebnisse dieser Tests enthalten

- Es gibt keine Beschreibung, wie diese Algorithmen genutzt werden können, um Probleme aus der Praxis zu lösen. Es fehlt eine Beschreibung, wie Daten von einem Sensor oder einer Datenbank in eine für die Algorithmen geeignete verteilte Repräsentation umgewandelt werden könnten

- Die Algorithmen sind in der Lage, „online“ zu lernen. Einige Details, die nötig sind, um das „Online“-Lernen in einigen selteneren Fällen vollständig umzusetzen, sind nicht beschrieben

- Weitere geplante Erweiterungen beinhalten eine Diskussion der Eigenschaften von spärlichen verteilten Repräsentationen, eine Beschreibung von Anwendungen und Beispielen, und Literaturreferenzen in den Anhängen

Wir veröffentlichen dieses Dokument in seiner derzeitigen Form, da wir der Meinung sind, dass die Algorithmen für andere interessant sind. Die fehlenden Teile des Dokuments sollten das Verständnis und das Experimentieren mit den Algorithmen durch motivierte Wissenschaftler nicht beeinträchtigen. Wir werden dieses Dokument regelmäßig überarbeiten, so dass es unseren Fortschritt widerspiegelt.

**Numenta Translation License**



The text, algorithms, sample code, pseudo code and other work included herein are based upon or translated from certain works related to hierarchical temporal memory ("HTM") technology published by Numenta Inc. Numenta holds the copyright in the original works and patent rights related to HTM and the algorithms translated herein. Numenta has agreed not to assert its patent rights against development or use of independent HTM systems, as long as such development or use is for research purposes only, and not for any commercial or production use. Any commercial or production use of HTM technology that infringes on Numenta's patents will require a commercial license from Numenta.

Based on the foregoing, Numenta grants you a license to use these algorithms and works for research purposes only and not for any commercial or production use. For purposes of this license, "commercial or production use" includes training an HTM network with the intent of later deploying the trained network or application for commercial or production purposes, and using or permitting others to use the output from HTM technology for commercial or production purposes. Any distribution, publication, or copying of this work must include the full text of this Translation License in both English and the target language.

NO EXPRESS OR IMPLIED LICENSES TO ANY PATENT RIGHTS ARE GRANTED BY THIS LICENSE. NUMENTA SPECIFICALLY DISCLAIMS ANY LIABILITY OR RESPONSIBILITY FOR THE QUALITY OR ACCURACY OF ANY TRANSLATIONS LICENSED HEREUNDER.

**Anmerkungen zur deutschen Übersetzung:**

Die folgende deutsche Übersetzung wurde in freiwilliger Arbeit ohne Hilfe eines professionellen Übersetzers vorgenommen. Das vorliegende Dokument stellt aufgrund der gemischten Terminologie aus Informatik, Neurowissenschaft, Medizin und Biologie einige Herausforderungen an eine Übersetzung dar, auch weil einige Begriffe im Kontext der HTM-Theorie eine leicht abgewandelte Bedeutung erhalten. Ich bin daher sehr dankbar für Korrekturen und Hinweise per E-Mail.

Es wurde versucht, nach Möglichkeit deutsche Fachbegriffe zu nutzen, zum Teil wurden die englischen Begriffe in Klammern ergänzt, um einen engen Bezug zum englischen Originaltext zu behalten.

Auch wenn es in wissenschaftlichen Texten deutscher Sprache nicht gern gesehen wird, die „wir"-Form zu verwenden, habe ich sie an manchen Stellen dieses Textes genutzt, da ich der Meinung bin, dass damit die lesernahe Sprache des amerikanischen Originals erhalten bleibt.

Ich hoffe, dass ich mit dieser Übersetzung einen kleinen Beitrag dazu leisten kann, die von Jeff Hawkins und seinen Kollegen und Kolleginnen entwickelten Ideen über die Funktionsweise des menschlichen Neokortex auch im deutschsprachigen Raum etwas bekannter zu machen.

Ingmar Baetge

Frankfurt am Main,
im Januar 2013

ingmar (dot) baetge (at) googlemail (dot) com

Inhaltsverzeichnis

# Vorwort

Es gibt viele Dinge, die für den Menschen einfach sind, zu denen Computer derzeit nicht fähig sind. Aufgabenstellungen wie zum Beispiel Mustererkennung, das Verstehen natürlicher Sprache, Erkennen und Manipulieren von Objekten durch Berührung sind für den Menschen einfache Aufgaben. Doch trotz jahrzehntelanger Forschung gibt es wenige brauchbare Algorithmen, die Leistungen mit einem Computer erbringen können, welche denen des Menschen ähnlich sind.

Solche Fähigkeiten werden beim Menschen größtenteils durch den Neokortex hervorgebracht. Hierarchical Temporal Memory (übersetzt: hierarchischer temporaler Speicher, im Englischen abgekürzt HTM) ist eine Technologie, die modelliert, wie der Neokortex diese Fähigkeiten ausführt. HTM bietet die Hoffnung auf Maschinen, die bei vielen kognitiven Aufgaben menschliches Niveau erreichen oder übertreffen.

Dieses Dokument beschreibt die HTM-Technologie. Kapitel 1 bietet einen breiten Überblick über HTM, behandelt den Stellenwert hierarchischer Organisation, spärlich besetzter Repräsentationen und des Lernens von zeitbasierten Übergängen. Kapitel 2 beschreibt die HTM-basierten kortikalen Lernalgorithmen im Detail. Kapitel 3 und 4 liefern Pseudocode für die kortikalen Lernalgorithmen aufgeteilt in zwei Teile, auf Englisch genannt der „spatial pooler“ (übersetzt: räumliche Musterzusammenfassung) und der „temporal pooler“ (übersetzt: temporale Musterzusammenfassung). Nach dem Lesen von Kapitel 2 bis 4 sollten erfahrene Softwareentwickler in der Lage sein, die Algorithmen nach zu implementieren und mit diesen zu experimentieren. Wir hoffen, dass einige Leser darüber hinaus unsere bisherige Arbeit erweitern.

## Zielgruppe

Dieses Dokument richtet sich an ein technisch ausgebildetes Publikum. Es wird zwar kein Vorwissen im Bereich der Neurowissenschaften vorausgesetzt, aber es wird davon ausgegangen, dass Konzepte aus dem Bereich der Mathematik und Informatik verstanden werden können. Das Dokument wurde so geschrieben, dass es als Lektüre im Rahmen einer Lehrveranstaltung genutzt werden könnte. Unserer Vorstellung nach ist ein typischer Leser Student der Informatik oder der Kognitionswissenschaften, oder ein Softwareentwickler, der an der Entwicklung von künstlichen kognitiven Systemen interessiert ist, die auf denselben Prinzipien aufbauen wie das menschliche Gehirn.

Leser ohne technischen Hintergrund können dennoch von bestimmten Teilen dieses Dokuments profitieren, insbesondere von Kapitel 1.

## Zusammenhang mit früheren Dokumenten

Teile der HTM-Theorie sind in dem 2004 erschienenen Buch *On Intelligence* (deutscher Titel: *Die Zukunft der Intelligenz*) beschrieben, in den *white papers*, die von Numenta veröffentlicht wurden, und in den von Mitarbeitern von Numenta veröffentlichten wissenschaftlichen Aufsätzen, die einem Peer-Review unterzogen wurden. Wir gehen nicht davon aus, dass Sie dieses vorangehende Material gelesen haben, vieles davon ist in diesem Dokument enthalten und aktualisiert. Man beachte, dass die in den Kapiteln 2-4 beschriebenen HTM-basierten Lernalgorithmen zuvor nicht veröffentlicht wurden. Die neuen Algorithmen ersetzen die erste Generation unserer Algorithmen, genannt Zeta 1. Für eine kurze Zeit nannten wir diese Algorithmen „Fixed-density Distributed Representations", oder „FDR", aber wir verwenden diese Bezeichnungen heute nicht mehr. Wir nennen die neuen Algorithmen „HTM Cortical Learning Algorithms" (übersetzt: HTM-basierte kortikale Lernalgorithmen), oder manchmal einfach HTM-Lernalgorithmen.

Wir empfehlen die Lektüre von *On Intelligence*, verfasst von Numenta-Co-Gründer Jeff Hawkins gemeinsam mit Sandra Blakeslee. Obwohl das Buch HTM nicht unter dieser Bezeichnung nennt, bietet es eine einfach zugängliche, nicht-technische Erläuterung der HTM-Theorie und der neurowissenschaftlichen Grundlagen. Als *On Intelligence* verfasst wurde, hatten wir die grundlegenden Prinzipien von HTM verstanden, aber wir wussten nicht, wie diese Prinzipien algorithmisch umgesetzt werden können. Man kann das vorliegende Dokument als Fortsetzung der in *On Intelligence* begonnenen Arbeit verstehen.

## Über Numenta

Numenta, Inc. (www.numenta.com) wurde 2005 mit dem Ziel gegründet, HTM-Technologie sowohl für kommerzielle als auch für wissenschaftliche Zwecke zu entwickeln. Um dieses Ziel zu erreichen, werden wir unsere Fortschritte und Entdeckungen vollständig dokumentieren. Wir werden außerdem unsere Software in einer Form veröffentlichen, die es ermöglicht, sie sowohl zu Forschungszwecken als auch für kommerzielle Zwecke nutzen können. Wir haben unserer Software so aufgebaut, dass sie das Entstehen einer unabhängigen Gemeinschaft von Anwendungsentwicklern unterstützt. Die Nutzung der Software des geistigen Eigentums von Numenta ist für wissenschaftliche Nutzung kostenfrei. Wir werden Einnahmen durch den Verkauf von Support, Softwarelizenzen und Lizenzen über die Nutzung des geistigen Eigentums für kommerzielle Nutzung erzielen. Wir sind immer am Erfolg unserer Entwicklungspartner interessiert, so wie wir auch an unserem eigenen Erfolg interessiert sind.

Numenta hat seinen Sitz in Redwood City, Kalifornien. Es wird aus privaten Mitteln finanziert.

## Über die Autoren

Dieses Dokument ist eine Gemeinschaftsleistung von Mitarbeitern bei Numenta. Die Namen der Hauptautoren jedes Abschnitts werden im Revisionsverlauf aufgezählt.

## Revisionsverlauf

In dieser Tabelle vermerken wir größere Änderungen zwischen Versionen. Kleinere Änderungen, so wie kurze Erläuterungen oder Formatänderungen werden nicht vermerkt.

| Version | Datum | Änderungen | Hauptautoren |
|---|---|---|---|
| 0.1 | Nov 9, 2010 | 1. Erstveröffentlichung des Vorworts, Kapitel 1,2,3,4 und Glossar. | Jeff Hawkins, Subutai Ahmad, Donna Dubinsky |
| 0.1.1 | Nov 23, 2010 | 1. Im ersten Kapitel wurde der Abschnitt über Regionen überarbeitet, um die Terminologie zu klären, wie z.B. Ebenen, Kolumnen und Schichten<br>2. Erstveröffentlichung Anhang A. | Hawkins & Dubinsky<br><br>Hawkins |
| 0.2 | Dec 10, 2010 | 1. Kapitel 2: Verschiedene Erläuterungen<br>2. Kapitel 4: Zeilenreferenzen aktualisiert; Code wurde in den Zeilen 37 und 39 verändert.<br>3. Erstveröffentlichung Anhang B | Hawkins<br><br>Ahmad<br><br>Hawkins |
| 0.2.1 | Sep 12, 2011 | 1. Bitte zuerst lesen: Bezug zu 2010 entfernt<br>2. Vorwort: Abschnitt bzgl. Software-Veröffentlichung entfernt | |

# Kapitel 1: Überblick über HTM

Hierarchical Temporal Memory (hierarchischer temporaler Speicher, engl. Abkürzung HTM) ist eine Technologie aus dem Bereich des maschinellen Lernens, welche versucht, die strukturellen und algorithmischen Eigenschaften des Neokortex zu erfassen.

Der Neokortex ist der Ort des intelligenten Denkens im Gehirn von Säugetieren. Höhere Funktionen wie Sehen, Hören, Tasten, Bewegen, Sprechen und Planen sind alle Aufgaben des Neokortex. Man könnte erwarten, dass eine solch vielfältige Folge von kognitiven Funktionen durch eine ebenso vielfältige Ansammlung von spezialisierten neuronalen Algorithmen umgesetzt wird. Dies ist nicht der Fall. Der Neokortex besteht aus einem bemerkenswert einheitlichen Muster neuronaler Verschaltungen. Die biologischen Anhaltspunkte lassen vermuten, dass der Neokortex eine allgemeine Menge von Algorithmen umsetzt, um viele unterschiedliche kognitive Funktionen zu erfüllen.

HTM bietet einen theoretischen Rahmen, um den Neocortex und seine vielen Fähigkeiten zu verstehen. Bis heute haben wir einen kleinen Teil dieses theoretischen Rahmens umgesetzt. Mit der Zeit wird mehr und mehr dieser Theorie implementiert werden. Heute sind wir der Meinung, dass der von uns umgesetzte Teil des Verhaltens des Neokortex groß genug ist, um kommerziellen und wissenschaftlichen Wert zu haben.

HTM zu programmieren unterscheidet sich vom Programmieren traditioneller Computer. Mit heutigen Computern erstellen Programmierer spezifische Programme für spezifische Problemstellungen. Im Gegensatz dazu werden HTMs durch einströmende sensorische Daten trainiert. Die Fähigkeiten von HTMs werden zum großen Teil durch die Informationen bestimmt, mit welchen das System in Kontakt kam.

HTM kann als bestimmter Typ neuronaler Netze betrachtet werden. Per Definition sind all jene Systeme neuronale Netze, die den detaillierten Aufbau des Neokortex nachbilden. Dennoch ist der Begriff „Neuronales Netz“ für sich gestellt nicht besonders nützlich, da er für eine große Bandbreite verschiedener Systeme genutzt wird. Das Modell von HTM verwendet Neuronen (im Kontext von HTM genannt „cells“, Zellen), welche in Kolumnen (engl. „columns“), Schichten und Ebenen („layers“), Regionen („regions“) und in einer Hierarchie angeordnet werden. Die Details dieser Anordnung spielen dabei eine Rolle, und in dieser Hinsicht ist HTM eine neue Form von neuronalen Netzen.

Wie der Name vorgibt, sind HTMs grundsätzlich speicherbasierte Systeme. HTM-Netze werden mit vielen zeitvarianten Daten trainiert und stützen sich auf das Speichern einer großen Anzahl von Mustern und Sequenzen. Die Art der Speicherung und des Abrufs von Daten folgt einer anderen Logik als das

Standardmodell, welches von Programmieren heute verwendet wird. Klassischer Speicher in einem Computer hat eine flache Organisationsstruktur und unterstützt keinen eigenen Zeitbegriff. Ein Programmierer kann auf Basis des flachen Speichermodells beliebige Arten der Datenorganisation und –struktur erstellen. Sie haben Kontrolle darüber, wie und wo Informationen gespeichert werden. HTM-Speicher ist restriktiver – er hat eine hierarchische Organisation und ist naturgemäß zeitbasiert. Information wird immer auf eine verteilte Art gespeichert. Ein Nutzer eines HTM gibt an, welche Größe die Hierarchie haben soll und mit welchen Daten das System trainiert, aber der Speicher selbst kontrolliert, wo und wie Informationen gespeichert werden.
Obwohl sich HTM-Netze wesentlich vom klassischen Berechnungsmodell unterscheiden, können allgemeine Computer genutzt werden, um diese umzusetzen, sofern die Schlüsselfunktionen der Hierarchie, Zeit und spärlichen Repräsentationen (Details dazu später) aufgenommen werden.

In diesem Dokument werden Eigenschaften und Prinzipien von HTM häufig bildhaft dargestellt anhand von Beispielen, die vom menschlichen Sehen, Tasten, Hören, Sprechen oder Verhalten abgeleitet wurden. Solche Beispiele sind nützlich, da sie intuitiv und einfach verständlich sind. Dennoch ist es wichtig zu beachten, dass die Fähigkeiten von HTMs allgemein sind. Sie können genauso einfach genutzt werden, um sie auf nicht-menschliche sensorische Datenströme anzuwenden, so wie Radar- oder Infrarot-Daten, oder auf Datenströme mit reinem Informationscharakter, wie z.B. Daten aus Finanzmärkten, Wetterdaten, Nutzungsdaten von Websites oder Textdaten. HTMs sind Lern- und Vorhersagemaschinen, die auf viele Arten von Problemen angewendet werden können.

## HTM-Prinzipien

In diesem Abschnitt werden einige Kernprinzipien von HTM behandelt, es wird erläutert, warum die hierarchische Organisation wichtig ist, wie HTM-Regionen strukturiert sind, warum Daten als spärliche verteilte Repräsentationen gespeichert werden, und warum zeitbasierte Information entscheidend ist.

### Hierarchie

Ein HTM-Netzwerk besteht aus Regionen, die in einer Hierarchie angeordnet sind. Eine Region ist die Haupteinheit zur Speicherung und Vorhersage in einem HTM, und wird im nächsten Abschnitt im Detail diskutiert werden. Typischerweise stellt jede HTM-Region eine Ebene einer Hierarchie dar. Wenn man die Hierarchie aufsteigt, findet sich dabei immer Konvergenz, mehrere Elemente in einer unteren Region konvergieren zu einem Element in einer höheren Region. Allerdings divergiert Information auch wieder beim Abstieg in der Hierarchie aufgrund von Feedback-Verbindungen (die Begriffe „Region“ und „Ebene“ werden fast synonym verwendet: Der Begriff „Region“ wird verwendet, wenn die interne Funktionsweise

einer Region beschrieben wird, während hingegen der Begriff „Ebene“ verwendet wird, wenn es um die Rolle einer Region innerhalb der Hierarchie geht).

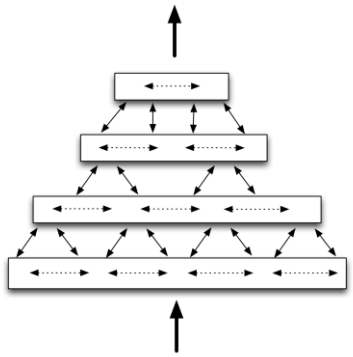


Abbildung 1.1: Vereinfachtes Diagramm von vier HTM-Regionen, die in einer Hierarchie aus vier Ebenen angeordnet sind. Information wird innerhalb der Ebenen, zwischen Ebenen und nach außen/von außen kommuniziert

Es ist möglich, mehrere HTM-Netzwerke zu kombinieren. Eine solche Struktur ist sinnvoll, wenn man Daten aus mehreren Quellen oder Sinneskanälen erhält. Zum Beispiel könnte ein Netzwerk auditive Informationen und ein anderes visuelle Informationen verarbeiten. Innerhalb jedes dieser getrennten Netzwerke gibt es Konvergenz, die getrennten Pfade konvergieren aber erst im obersten Teil.

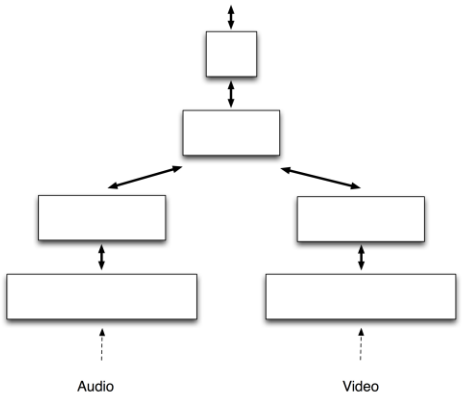



Abbildung 1.2: Zusammenfließende Netzwerkströme verschiedener Sensoren

Der Vorteil der hierarchischen Organisation ist Effizienz. Trainingsdauer und Speicherverbrauch werden erheblich verringert, wenn die Muster, die jeweils auf einer Ebene erlernt wurden, wiederverwendet werden, wenn man sie auf neue Arten in höheren Ebenen kombiniert. Zu Veranschaulichung soll ein Beispiel aus der visuellen Wahrnehmung betrachtet werden: Auf der niedrigsten Ebene speichert das Gehirn Informationen über winzige Abschnitte des visuellen Feldes, etwa Kanten und Ecken. Eine Kante ist ein fundamentaler Bestandteil vieler Objekte aus

der wahrnehmbaren Welt. Solch elementare Muster der untersten Ebene werden in den mittleren Ebenen neu kombiniert und bilden komplexere Komponenten, z.B. Kurvenzüge und Texturen. Ein Bogen kann dabei die Kante eines Ohrs sein, der obere Teil eines Lenkrades oder der Rand einer Kaffeetasse. Die Muster in den mittleren Ebenen werden weiter kombiniert, um komplexe Objekte zu repräsentieren, z.B. den Kopf eines Menschen, ein Fahrzeug oder ein Haus. Um Objekte in einer höheren Ebene zu erlernen, müssen die einzelnen Bestandteile nicht neu erlernt werden.

Als weiteres Beispiel dient das Erlernen neuer Wörter. Dabei müssen Buchstaben, Silben oder Phoneme nicht neu erlernt werden.

Die gemeinsame Nutzung von Repräsentationen in einer Hierarchie führt außerdem zu einer Generalisierung von erwartetem Verhalten. Wenn ein Mensch ein zuvor unbekanntes Tier mit einem Maul und Zähnen sieht, dann wird er annehmen, dass dieses Tier mit dem Maul frisst und daher beißen könnte. Die Hierarchie erlaubt es, dass neu wahrgenommene Objekte Eigenschaften von bereits bekannten Teilkomponenten erben.

Wie viel kann eine einzelne Ebene in einer HTM-Hierarchie lernen? Oder, anders formuliert, wie viele Ebenen in der Hierarchie sind erforderlich? Es gibt eine wechselseitige Beziehung zwischen der Größe des Speichers pro Ebene und der Anzahl der Ebenen. Erfreulicherweise erlernen HTM-Ebenen automatisch die besten Repräsentationen pro Ebene aufgrund der statistischen Eigenschaften der Eingangssignale und der Größe des zur Verfügung stehenden Speichers.
Wird mehr Speicher bereitgestellt, dann wird eine Ebene größere und komplexere Repräsentationen bilden, was dazu führt, dass eventuell weniger Ebenen benötigt werden. Wird weniger Speicher bereitgestellt, dann bildet eine Ebene kleinere und einfachere Repräsentationen, und es werden eventuell mehr Ebenen benötigt.

Bis jetzt wurden komplexe Probleme wie die visuelle Inferenz beschrieben (Inferenz im Sinne von Mustererkennung). Viele wertvolle Probleme sind allerdings einfacher als die visuelle Wahrnehmung, und eine einzelne HTM-Ebene könnte bereits ausreichend sein. Beispielsweise wurde ein HTM genutzt, um eine Vorhersage zu treffen, wohin eine Person auf einer Webseite mit hoher Wahrscheinlichkeit als Nächstes klickt. Zur Lösung dieses Problems wurden Datenströme von Klicks auf Webseiten in ein HTM geleitet. Dabei fanden sich kaum oder keine Hierarchien, zur Lösung mussten hauptsächlich zeitliche Verteilungen erkannt werden, d.h. die Vorhersage wird ermöglicht durch das Erkennen typischer Klickfolgen von Nutzern. Der temporale Lernalgorithmus in HTM-Netzen ist ideal für solche Problemstellungen.

Zusammenfassend reduziert Hierarchie den Trainingsaufwand, Speicherverbrauch und ermöglicht eine Art von Generalisierung. Dennoch können viele einfachere Vorhersageprobleme mit einer einzelnen HTM-Ebene gelöst werden.

## Regionen

Der Begriff der Regionen, die in einer Hierarchie miteinander verbunden sind, stammt aus der Biologie. Der Neokortex ist eine große Fläche aus neuronalem Gewebe mit einer Dicke von ca. 2mm. Biologen unterteilen den Neokortex in verschiedene Teile oder Regionen, in erster Linie nach der Art, wie verschiedene Regionen miteinander verbunden sind. Einige Regionen erhalten ihre Eingabeinformation direkt von der Sensorik, und andere Regionen erhalten nur Eingabeinformationen, die bereits mehrere andere Regionen durchlaufen haben. Es ist diese Verknüpfung zwischen verschiedenen Regionen, die die Hierarchie definiert.

Im Detail sehen alle Regionen im Neokortex ähnlich aus. Sie variieren in ihrer Größe und ihrer Position in der Hierarchie, aber ansonsten sind sie ähnlich. Wenn man einen Querschnitt durch das 2mm dicke Gewebe einer Region des Neokortex schneidet, dann sieht man sechs Schichten, fünf Schichten mit Zellen und eine nicht-zelluläre Schicht (es gibt einige Ausnahmen, aber dies ist die Regel). Jede Schicht in einer Region im Neokortex hat viele miteinander verbundene Zellen, die in Kolumnen angeordnet sind. HTM-Regionen sind ebenfalls aufgebaut aus einer Fläche von hoch vernetzten Zellen, die ebenfalls in Kolumnen angeordnet sind. Die Schicht 3 im Neokortex ist eine der Hauptschichten für „feed-forward“-Neuronen. Die Zellen in einer HTM-Region entsprechen grob gefasst den Zellen der Schicht 3 in einer Region im Neokortex.

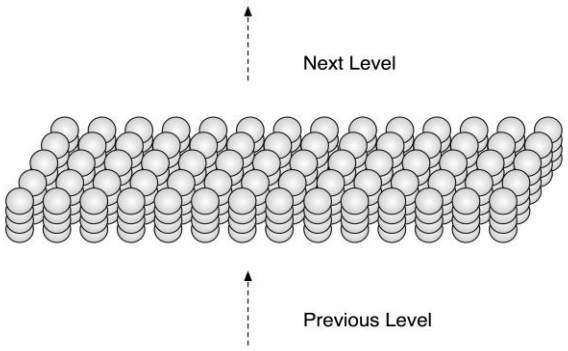



Abbildung 1.3: Ein Teil einer HTM-Region. HTM-Regionen sind aus vielen Zellen aufgebaut. Die Zellen sind als zweidimensionale Matrix von Kolumnen angeordnet. Diese Abbildung zeigt einen kleinen Ausschnitt aus einer HTM-Region mit vier Zellen pro Kolumne. Jede Kolumne ist mit einer Teilmenge der Eingabeebene verbunden, und jede Zelle verbindet sich mit anderen Zellen dieser Region (die Verbindungen sind nicht dargestellt). Man beachte, dass diese HTM-Region mit samt der Kolumnenstruktur einer Schicht von Neuronen einer Region aus dem Neokortex entspricht.

Obwohl eine HTM-Region nur einem Teil einer Region des Neokortex entspricht, kann sie bereits Schlussfolgerungen und Vorhersagen auf komplexen Datenströmen durchführen, sie ist daher bereits nützlich zur Lösung von vielen Problemen.

## Spärliche verteilte Repräsentationen

Obwohl die Neuronen im Neokortex miteinander hoch vernetzt sind, garantieren hemmende Neuronen, dass nur ein kleiner Prozentsatz der Neuronen gleichzeitig aktiv wird. Daher wird Information im Gehirn immer von einem kleinen Prozentsatz aktiver Neuronen aus einer großen Gesamtpopulation von Neuronen repräsentiert. Diese Art der Kodierung nennt man „spärliche verteilte Repräsentation". „Spärlich" bedeutet, dass nur eine kleine Anzahl Neuronen gleichzeitig aktiv sind. „Verteilt" bedeutet, dass viele Neuronen gleichzeitig aktiv sein müssen, um etwas zu repräsentieren. Ein einzelnes aktives Neuron trägt ein wenig Bedeutung, aber es muss im Kontext einer Population von Neuronen interpretiert werden, um die vollständige Bedeutung zu erhalten.

HTM-Regionen nutzen ebenfalls spärliche verteilte Repräsentationen. Tatsächlich sind die Speichermechanismen innerhalb einer HTM-Region davon abhängig, spärliche verteilte Repräsentationen zu nutzen, und sie würden sonst nicht funktionieren. Die Eingabe einer HTM-Region ist immer eine verteilte Repräsentation, aber es könnte auch eine nicht-spärliche sein, daher ist die erste Aufgabe einer HTM-Region die Umwandlung der Eingabe in eine spärliche verteilte Repräsentation.

Ein Beispiel: Eine Region erhält eine 20000 Bit große Eingabe. Der Prozentsatz der Eingabebits, die 1 oder 0 sind, variiert je nach Zeitpunkt stark. Zu einem Zeitpunkt könnten 5000 „1" Bits vorhanden sein, zu einem anderen 9000 „1" Bits. Die HTM-Region könnte diese Eingaben in einer interne Repräsentation aus 10000 Bits umwandeln, bei der 2% (200 Bits) gleichzeitig aktiv sind, unabhängig von der Zahl der aktiven Eingabebits. Da die Eingabe sich mit der Zeit ändert, wird sich auch die interne Repräsentation ändern, aber es werden immer etwa 200 von den 10000 Bits aktiv sein.

Es mag den Anschein haben, als ob dieser Prozess einen großen Verlust an Information darstellt, da die Zahl an möglichen Eingabemustern viel höher ist als die Zahl der möglichen Repräsentationen in der Region. Tatsächlich sind aber beide Zahlen überaus groß. Die tatsächlichen Eingabemuster einer Region sind ein winziger Bruchteil aller möglichen Eingabemuster. Später wird beschrieben, wie eine Region eine spärliche verteilte Repräsentation aus der Eingabe erzeugt. Der theoretische Verlust an Information wird keine praktischen Auswirkungen haben.

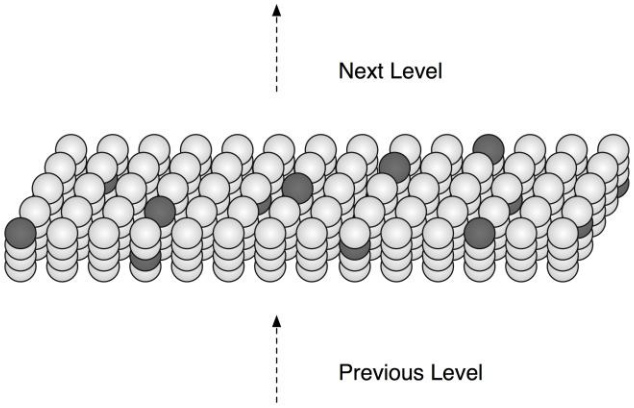



Abbildung 1.4: Eine HTM-Region mit einer spärlichen verteilten Zellaktivierung

Spärliche verteilte Repräsentationen haben einige wünschenswerte Eigenschaften und sind ein integraler Bestandteil der Funktionsweise von HTMs. Sie werden später erneut betrachtet werden.

**Die Rolle der Zeit**

Zeit spielt eine entscheidende Rolle beim Lernen, Schlussfolgern (Inferenz) und Vorhersagen.

Man betrachte zunächst die Inferenz: Ohne Zeitbegriff können wir fast nichts aus den Informationen unseres Tast- oder auditiven Sinnes schließen. Wenn wir beispielsweise unsere Augen verbinden und man uns einen Apfel in die Hand gibt, dann können wir ihn binnen Sekunden durch Tasten erkennen. Obwohl die Informationen unserer Tastsinne sich beim Berühren des Apfels ständig ändern, bleibt das Konzept des „Apfels“ auf einer höheren Ebene konstant. Wenn wir aber einen Apfel auf unsere Handfläche gelegt bekommen und wir unsere Hand oder unsere Finger nicht bewegen dürfen, dann fällt es uns schwer, zu unterscheiden, ob es sich um einen Apfel oder eine Zitrone handelt.

Dieselbe Überlegung gilt für das Hören. Ein statischer Klang trägt wenig Bedeutung. Ein Wort wie „Apfel“ oder das knirschende Geräusch eines Biss in einen Apfel kann nur durch dutzende oder hunderte schneller sequentieller Änderungen im Klangspektrum erkannt werden.

Visuelle Wahrnehmung hingegen ist eine Mischform. Im Gegensatz zu Berührung und Hören können Menschen Bilder erkennen, wenn sie so schnell vor ihnen aufblitzen, dass das Auge keine Möglichkeit hat, sich zu bewegen. Also erfordert visuelle Wahrnehmung nicht immer Eingaben, die sich mit der Zeit ändern. Dennoch bewegen wir beim alltäglichen Sehen ständig unsere Augen, unsere Köpfe und unseren Körper, und die Objekte in der Welt um uns herum bewegen sich ebenfalls. Unsere Fähigkeit, etwas aus einem sehr kurzen visuellen Eindruck zu

erkennen, wird ermöglicht durch die statistischen Eigenschaften visueller Wahrnehmung und dem jahrelangen Training. Im allgemeinen Fall erfordern Sehen, Hören und Tasten sich über die Zeit ändernde Eingaben.

Nachdem der allgemeine Fall der Inferenz und der spezielle Fall der visuellen Wahrnehmung statischer Bilder behandelt wurde, wird jetzt das Lernen betrachtet. Um zu lernen, müssen alle HTM-Systeme mit zeitvarianten Eingabedaten in Berührung kommen. Selbst bei der visuellen Wahrnehmung, bei der das Erkennen statischer Bilder in manchen Fällen möglich ist, müssen wir sich ändernde Bilder von Objekten sehen, um zu erlernen, wie ein Objekt aussieht. Stellen Sie sich beispielsweise einen Hund vor, der auf Sie zukommt. Zu jedem Zeitpunkt entsteht auf Ihrer Retina ein bestimmtes Aktivitätsmuster als Abbild des Hundes. Sie erleben diese Muster als verschiedene Ansichten desselben Hundes, aber mathematisch gesehen handelt es sich jeweils um völlig unterschiedliche Muster. Das Gehirn lernt, dass diese Muster dasselbe bedeuten, in dem es diese Muster als Sequenz beobachtet. Die Zeit lehrt uns, welche räumlichen Muster zusammengehören.

Man beachte, dass es nicht ausreicht, wenn die wahrgenommenen Muster sich ändern. Eine Abfolge unzusammenhängender sensorischer Muster würde nur zu Verwirrung führen. Die sich mit der Zeit ändernden Eingabemuster müssen einer gemeinsamen Quelle der Umgebung entstammen. Obwohl hier Beispiele aus dem Bereich der menschlichen Wahrnehmung angeführt werden, sei festgestellt, dass der allgemeine Fall auch auf nicht-menschliche Sensoren angewendet werden kann. Wenn ein HTM trainiert werden soll, Muster bei Temperatur, bei Vibrationen oder Geräuschen eines Kraftwerks zu erkennen, dann muss das HTM mit Daten dieser Sensoren in ihrer zeitlichen Abfolge trainiert werden.

Typischerweise muss ein HTM mit sehr vielen Daten trainiert werden. Sie haben gelernt, Hunde zu erkennen, indem Sie sehr viele verschiedene Vorkommen von verschiedenen Rassen von Hunden gesehen haben, nicht nur durch einen einzelnen Blickwinkel auf einen einzelnen Hund. Die Aufgabe des HTM-Algorithmus ist es, temporale Muster aus einem Datenstrom zu erlernen, d.h. zu modellieren, wie die verschiedenen Muster aufeinander folgen. Diese Aufgabe ist schwierig, da nicht klar ist, wann eine Folge von Mustern beginnt oder endet, ferner können Sequenzen sich überlappen, das Lernen muss fortlaufend erfolgen und das Lernen muss trotz Störsignalen erfolgen.

Lernen und Mustererkennung sind die Voraussetzung für das Erstellen von Vorhersagen. Wenn ein HTM gelernt hat, welche Muster voraussichtlich anderen Mustern folgen, dann können die naheliegenden Folgemuster auf Grundlage der aktuellen und gerade vergangenen Eingaben vorhergesagt werden. Die Vorhersage wird später detaillierter behandelt.

Im Folgenden werden die vier elementaren Aufgaben eines HTM betrachtet: Lernen, Inferenz, Vorhersage und Verhalten. Jede HTM-Region führt dabei die ersten drei Aufgaben durch: Lernen, Inferenz und Vorhersage. Verhalten hingegen

unterscheidet sich davon. Es ist aus der Biologie bekannt, dass die meisten Regionen im Neokortex bei der Entstehung von Verhalten eine Rolle spielen, aber wir glauben, dass es für viele interessante Anwendungen nicht wichtig ist. Daher wurde Verhalten nicht in die aktuelle Implementierung von HTM aufgenommen. Dieser Umstand sei der Vollständigkeit halber hier erwähnt.

## Lernen

Eine HTM-Region lernt etwas über die Umgebung, in dem Muster und danach Sequenzen von Mustern in den sensorischen Daten gefunden werden. Die Region „weiß“ aber nicht, was die Eingabemuster bedeuten. Sie arbeitet nur auf dem Gebiet der Statistik. Sie sucht nach Kombinationen von Eingabebits, die häufig gemeinsam auftreten, diese werden dann räumliche Muster genannt. Anschließend sucht sie danach, in welcher Reihenfolge diese räumlichen Muster im Verlauf auftreten, diese Sequenzen werden temporale Muster oder Sequenzen genannt.

Wenn die Eingabe einer Region beispielsweise sensorische Daten eines Gebäudes repräsentiert, dann könnte die Region entdecken, dass bestimme Kombinationen von Temperatur und Luftfeuchtigkeit häufig an der Nordseite des Gebäudes auftreten, und dass andere Kombinationen an der Südseite auftreten. Danach könnte sie erlernen, dass Sequenzen dieser Kombinationen im Verlauf jeden Tages auftreten.

Wenn die Eingabe einer Region Informationen über die Verkäufe eines Ladens repräsentiert, dann könnte eine HTM-Region entdecken, dass bestimmte Arten von Artikeln am Wochenende gekauft werden, oder dass bestimmte Preisspannen am Abend bei kaltem Wetter favorisiert werden. Dann könnte sie erlernen, dass unterschiedliche Individuen ähnlichen sequenziellen Einkaufsmustern folgen.

Eine einzelne HTM-Region hat eine begrenzte Lernkapazität. Eine Region passt sich in ihrem Lernverhalten automatisch an den gegebenen Speicher und die Komplexität der Eingaben an. Die räumlichen Muster, die von einer Region erlernt werden können, müssen zwangsweise einfacher werden, wenn weniger Speicher zur Verfügung gestellt wird. Sie können aber auch komplexer werden, wenn der zur Verfügung stehende Speicher erhöht wird. Wenn die erlernten räumlichen Muster einer Region einfach sind, dann könnte eine Hierarchie von Regionen erforderlich sein, um komplexe Bilder zu verstehen. Wir finden diese Charakteristik im menschlichen visuellen Wahrnehmungssystem, in dem die Region des Neokortex, die ihre Eingabe direkt von der Retina erhält, räumliche Muster für kleine Teile des gesamten visuellen Raums erlernt. Erst nach mehreren Ebenen einer Hierarchie werden die räumlichen Muster kombiniert und repräsentieren einen Großteil oder die Gesamtheit des visuellen Raums.

Wie ein biologisches System ist der Lernalgorithmus in einer HTM-Region in der Lage, „online“ zu lernen, d.h. kontinuierlich von jeder neuen Eingabe zu lernen. Es besteht keine Notwendigkeit für eine Lernphase getrennt von einer Inferenzphase, obwohl sich die Inferenz nach zusätzlichem Lernen verbessert. Wenn sich die Muster der Eingaben ändern, wird sich nach und nach auch die HTM-Region ändern.

Nach einem anfänglichen Training kann eine HTM-Region weiterhin lernen, oder das Lernen kann nach einer Trainingsphase abgeschaltet werden. Eine andere Option ist es, das Lernen nur in der untersten Hierarchieebene zu deaktivieren, auf höheren Ebenen aber noch weiter zu lernen. Sobald ein HTM die grundlegende statistische Struktur der es umgebenden Welt erlernt, findet neues Lernen hauptsächlich auf den höheren Ebenen statt. Wenn ein HTM neuen Mustern einer bisher ungekannten Struktur auf der tieferen Ebene ausgesetzt wird, dann dauert das Erlernen dieser Muster länger. Wir sehen diese Eigenschaft am Menschen. Neue Wörter einer Sprache zu erlernen, die wir schon kennen, ist relativ einfach. Wenn wir aber versuchen, neue Wörter einer Fremdsprache mit ungewohntem Klang zu erlernen, dann erweist sich dies als schwieriger, da wir die Klänge auf unteren Ebenen noch nicht kennen.

Schlicht das Erlernen von Mustern ist bereits eine potentiell wertvolle Fähigkeit. Die Muster in Marktbewegungen, in der Verbreitung von Krankheiten, des Wetters, der Herstellungskapazität oder beim Versagen komplexer Systeme, z.B. von Stromnetzen, zu verstehen ist bereits wertvoll. Darüber hinaus ist das Erlernen räumlicher und temporaler Muster meist eine Voraussetzung für Inferenz und Vorhersage.

## Inferenz

Nachdem ein HTM die Muster seiner Welt erlernt hat, kann es Inferenz auf neuen Eingaben durchführen. Wenn ein HTM Eingaben erhält, wird es die erlernten räumlichen und temporalen Muster vergleichen. Muster, die zu zuvor gespeicherten Sequenzen passen, sind der Kern der Inferenz und der Mustererkennung.

Man betrachte, wie eine Melodie erkannt wird. Am Klang der ersten Note kann man fast nichts ableiten. Die zweite Note schränkt die Anzahl der Möglichkeiten deutlich ein, aber es könnte dennoch noch nicht ausreichend sein. Üblicherweise benötigt man drei, vier oder mehr Töne bevor man eine Melodie erkennt. Inferenz in einer HTM-Region ist ähnlich. Sie betrachtet ständig die einfließenden Eingaben und vergleicht sie mit den bisher erlernten Sequenzen. Eine HTM-Region kann Übereinstimmungen am Beginn einer Sequenz erkennen, aber meist ist das Erkennen fließender und unabhängig davon, an welcher Stelle man beginnt - analog zu der Fähigkeit des Menschen, eine Melodie zu erkennen. Da HTM-Regionen verteilte Repräsentationen nutzen, ist die Verwendung des Sequenzspeichers und der Inferenz komplizierter, als es das Beispiel der Melodie nahelegt, aber das Beispiel vermittelt die grundlegende Idee der Funktionsweise.

Es mag nicht sofort ersichtlich sein, aber jede menschliche sensorische Empfindung ist eine neue Empfindung, dennoch können Menschen leicht bekannte Muster in dieser neuen Empfindung erkennen. Beispielsweise kann man das Wort „Frühstück“ von jedem beliebigen Sprecher verstehen, unabhängig davon ob die Person jung oder alt, männlich oder weiblich ist, schnell oder langsam spricht oder einen starken Akzent hat. Selbst wenn eine und dieselbe Person das Wort „Frühstück“ einhundert Mal ausspricht, der Klang würde die Cochlea (der Sitz der auditiven Rezeptoren) nie exakt auf dieselbe Art stimulieren.

Eine HTM-Region ist mit demselben Problem wie unser Gehirn konfrontiert: Eingaben könnten sich nie exakt wiederholen. Daher muss eine HTM-Region wie unser Gehirn neue Eingaben bei Inferenz und beim Lernen verarbeiten. Eine Art, wie eine HTM-Region mit neuen Eingaben umgeht, ist die Verwendung von spärlichen verteilten Repräsentationen. Eine Schlüsseleigenschaft von spärlichen verteilten Repräsentationen ist, dass nur ein Teil eines Musters passen muss, um sicher zu sein, dass es sich um ein bestimmtes Muster handelt.

## Vorhersage

Jede Region eines HTM speichert Sequenzen von Mustern. Indem die gespeicherten Sequenzen mit der aktuellen Eingabe verglichen werden, kann eine Region eine Vorhersage darüber treffen, welche Eingaben mit hoher Wahrscheinlichkeit als nächstes erwartet werden. HTM-Regionen speichern tatsächlich Übergänge zwischen spärlichen verteilten Repräsentationen. In manchen Fällen sehen diese Übergänge wie lineare Sequenzen aus, wie die Töne einer Melodie, aber im allgemeinen Fall können viele zukünftige Eingaben gleichzeitig vorhergesagt werden. Eine HTM-Region wird unterschiedliche Vorhersagen aufgrund des aktuellen Kontextes treffen, der bis weit in die Vergangenheit reichen kann. Der Hauptanteil des Speichers ist dem Speichern von Sequenzen gewidmet, oder anders gesagt dem Speichern von Übergängen zwischen räumlichen Mustern.

Im Folgenden werden einige Schlüsseleigenschaften von HTM-Vorhersagen vorgestellt.

### 1) Vorhersage ist kontinuierlich

Wir sind uns nicht darüber bewusst, aber wir treffen ständig Vorhersagen. HTMs machen dasselbe. Wenn wir einem Lied zuhören, dann treffen wir Vorhersagen über den nächsten Ton. Wenn wir Treppen hinab laufen, dann treffen wir eine Vorhersage darüber, wann unser Fuß die nächste Stufe berühren wird. Wenn wir einen Baseballspieler beim Wurf beobachten, dann treffen wir die Vorhersage, dass der Ball in Richtung Fänger fliegen wird. In einer HTM-Region ist Inferenz und Vorhersage fast dasselbe. Vorhersage ist kein separater Schritt, sondern ein integraler Bestandteil der Funktionsweise einer HTM-Region.

**2) Vorhersage findet in jeder Region und auf jeder Hierarchiestufe statt.**
In einer Hierarchie von HTM-Regionen findet Vorhersage auf jeder Ebene statt. Regionen treffen Vorhersagen über die von ihnen erlernten Muster. In einem Sprachbeispiel treffen niedrigere Regionen Vorhersagen über die nächsten Phoneme, höhere Ebenen treffen Vorhersagen über Wörter oder Phrasen bzw. Satzteile.

**3) Vorhersagen sind von ihrem Kontext abhängig.**
Vorhersagen beruhen sowohl auf dem, was in der Vergangenheit geschah, als auch darauf, was im Moment geschieht. Daher wird eine Eingabe, jeweils in Abhängigkeit von ihrem Kontext, zu unterschiedlichen Vorhersagen führen. Eine HTM-Region erlernt, soviel vorherigen Kontext zu nutzen wie nötig, und es kann diesen Kontext über sowohl kurze als auch längere Zeitabstände behalten. Diese Fähigkeit wird Speicher variabler Ordnung genannt. Man betrachte eine auswendig gelernte Rede, zum Beispiel Abraham Lincolns Rede von Gettysburg („Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal. […]"). Um das nächste Wort vorherzusagen, reicht die Kenntnis des aktuellen Wortes kaum aus; das Wort „and" wird gefolgt von „seven", und später von „dedicated", und das schon im ersten Satz. Manchmal hilft nur ein wenig Kontext, um eine Vorhersage zu treffen. Kennt man den Anfang „Four score and", dann kann man „seven" vorhersagen. In anderen Fällen gibt es wiederholte Phrasen, dann müsste man den Kontext eines deutlich größeren Zeitabschnitts nutzen, um zu wissen, an welcher Stelle der Rede man sich gerade befindet und was als Nächstes folgt.

**4) Vorhersage führt zu Stabilität.**
Die Ausgabe einer Region ist ihre Vorhersage. Eine der Eigenschaften von HTM besteht darin, dass die Ausgaben stabiler werden – d.h. sie verändern sich langsamer und bleiben länger bestehen, je höher sie sich in der Hierarchie befinden. Diese Eigenschaft folgt aus der Art und Weise, wie Regionen Vorhersagen treffen. Eine Region sagt nicht nur den unmittelbar nächsten Schritt voraus. Wenn möglich, wird eine Vorhersage für mehrere Schritte in der Zukunft getroffen. Man nehme an, eine Region kann Vorhersagen für die nächsten 5 Schritte in der Zukunft treffen. Wenn eine neue Eingabe eintrifft, wird sich die Vorhersage für den neu vorhergesagten Schritt ändern, aber die Vorhersage für die vier zuvor vorhergesagten Schritte ändert sich vielleicht nicht. Also wird sich nur ein Teil der Ausgabe ändern, obwohl die Eingabe sich komplett unterscheidet, dadurch wird die Ausgabe eine Region stabiler als ihre Eingabe. Diese Charakteristik spiegelt unsere Wahrnehmung der uns umgebenden Welt, in der Konzepte auf höheren Ebenen – zum Beispiel der Name eines Lieds – sich langsamer ändern als Konzepte auf niedrigeren Ebenen, z.B. die einzelnen Noten eines Lieds.

**5) Eine Vorhersage verrät, ob eine neue Eingabe erwartet oder unerwartet ist.**
Jede HTM-Region kann neue Muster erkennen". Da jede Region vorhersagt, was als nächstes passieren wird, „weiß" sie, wenn etwas Unerwartetes eintritt. HTM-

Regionen können viele mögliche folgende Eingaben gleichzeitig vorhersagen, nicht nur eine. Daher kann sie oft nicht exakt vorhersagen, was als nächstes passiert, aber wenn die Eingabe keiner der Vorhersagen gleicht, dann weiß die HTM-Region, dass es sich um eine Anomalie handelt.

**6) Vorhersage führt zu Stabilität gegenüber Rauschen.**
Wenn eine HTM-Region vorhersagt, was wahrscheinlich als Nächstes passiert, dann bildet die Vorhersage eine Tendenz bezüglich der Inferenz der nächsten Vorhersage. Wird beispielsweise gesprochene Sprache verarbeitet, dann würde die Region vorhersagen, welche Geräusche, Wörter oder Gedanken vermutlich als nächstes gesprochen werden. Diese Vorhersage hilft dem System, fehlende Daten zu ergänzen. Wenn ein mehrdeutiges Geräusch eintritt, dann wird die HTM-Region das Geräusch im Kontext der Vorhersage interpretieren, und damit die Inferenz in Gegenwart von Rauschen erleichtern.

In einer HTM-Region sind Sequenz-Speicher, Inferenz und Vorhersage eng miteinander verbunden. Sie sind die Kernfunktionen einer Region.

## Verhalten

Unser Verhalten beeinflusst was wir wahrnehmen. Während wir unsere Augen bewegen, erhält unsere Retina sich verändernde Eingaben. Wenn wir unsere Glieder und Finger bewegen, erhält unser Gehirn wechselnde Berührungsempfindungen. Fast all unser Handeln verändert unsere Wahrnehmung. Sensorische Eingabe und motorisches Verhalten sind eng miteinander verknüpft.

Jahrzehntelang war es vorherrschende Meinung, dass eine einzelne Region im Neokortex, die „primary motor region“, der Ursprung aller motorischen Anweisungen im Neocortex war. Mit der Zeit wurde entdeckt, dass fast alle Regionen im Neokortex motorische Ausgaben haben, selbst niedrige sensorische Regionen. Es scheint, dass alle kortikalen Regionen sensorische und motorische Funktionen zusammenfassen.

Wir erwarten, dass motorische Ausgaben zu dem bestehenden Framework hinzugefügt werden können, da das Erzeugen motorischer Signale ähnlich dem Treffen von Vorhersagen ist. Dennoch sind alle bisherigen Implementierungen von HTM-Regionen bis jetzt rein sensorisch, ohne eine motorische Komponente.

## Fortschritt in Richtung einer HTM-Implementierung

Wir haben substantiellen Fortschritt darin gemacht, das theoretische Rahmenwerk in eine Anwendungstechnologie zu wandeln. Wir haben mehrere Versionen des kortikalen Lernalgorithmus einer HTM-Region implementiert und haben festgestellt, dass die Grundarchitektur tragfähig ist. Wenn wir die Algorithmen mit neuen Daten testen, werden wir die Algorithmen anpassen und fehlende Teile ergänzen. Wir werden dieses Dokument damit aktualisieren. Die nächsten drei Kapitel beschreiben den aktuellen Zustand der Algorithmen.

Es gibt viele Bestandteile der Theorie, die noch nicht umgesetzt wurden, darunter Aufmerksamkeit, Rückkopplung zwischen Regionen, spezifische Zeitaspekte und die Integration von Motorik und Sensorik. Diese fehlenden Teile sollen in das bisher entwickelte Framework passen.

# Kapitel 2: HTM-basierte kortikale Lernalgorithmen

Dieses Kapitel beschreibt die Lernalgorithmen, die innerhalb einer HTM Region arbeiten. Kapitel 3 und 4 beschreiben die Implementierung dieser Algorithmen mittels Pseudocode, während dieses Kapitel eher konzeptuell angelegt ist.

## Terminologie

Bevor wir beginnen, könnte ein Hinweis zur Terminologie hilfreich sein. Wir verwenden neurowissenschaftliche Begriffe, um die HTM-Lernalgorithmen zu beschreiben. Begriffe wie Zellen, Synapsen, potentielle Synapsen, Dendriten und dendritische Segmente und Kolumnen werden durchgehend verwendet. Diese Terminologie ist logisch, da die Algorithmen zum großen Teil entwickelt wurden, indem neurowissenschaftliche Details und Anforderungen aus der Theorie in Zusammenhang gebracht wurden. Dennoch begegneten wir während der Implementierung der Algorithmen Geschwindigkeitsproblemen, daher suchten wir nach Wegen, Verarbeitungsschritte zu beschleunigen, wenn wir das Gefühl hatten, dass wir sie ausreichend verstanden hatten. Dies bedeutete oft, dass wir von der strikten Einhaltung von biologischen Details abgewichen sind, so lange die Berechnungsergebnisse erhalten blieben. Wenn Sie als Leser Einsteiger im Gebiet der Neurowissenschaft sind, dann stellt dies kein Problem dar. Wenn Ihnen aber neurowissenschaftliche Begriffe bekannt sind, dann könnte dies an manchen Stellen zu Verwirrung führen, da die Verwendung unserer Begriffe von Ihrer Erwartung abweichen kann. Die Anhänge zur Biologie diskutieren die verschiedenen Unterschiede und Ähnlichkeiten zwischen den HTM-Lernalgorithmen und ihren neurobiologischen Entsprechungen im Detail. An dieser Stelle werden wir einige Abweichungen erwähnen, die wahrscheinlich die meisten Verwirrungen erzeugen.

**Zustände von Zellen**
HTM-Zellen haben drei verschiedene Ausgabezustände, sie können aktiv sein durch „feed-forward"-Eingaben, aktiv durch laterale Eingaben (dies entspricht einer Vorhersage) oder sie können inaktiv sein. Der erste Ausgabezustand entspricht dem kurzen Feuern von Aktionspotentialen in einem Neuron. Der zweite Ausgabezustand entspricht einer langsameren, beständigeren Rate von Aktionspotentialen in einem Neuron. Wir haben keine Notwendigkeit gefunden, individuelle Aktionspotentiale oder auch nur Feuerraten von Aktivität außer diesen zwei Zuständen zu modellieren. Die Verwendung von verteilten Repräsentationen scheint die Notwendigkeit der Modellierung von Aktivitätsraten in Zellen überflüssig zu machen.

**Dendritische Segmente**
HTM-Zellen haben ein relativ realistisches (und daher komplexes) Model von Dendriten. Laut Theorie hat jede Zelle eines HTM ein proximales (nahes) Dendritensegment und ein oder zwei Dutzend distale (ferne) Dendritensegmente.

Das proximale Dendritensegment erhält „feed-forward“-Eingaben und die distalen Dendritensegmente erhalten laterale Eingaben von nahegelegenen Zellen. Eine Klasse von hemmenden Zellen erzwingen alle Zellen einer Kolumne auf gleiche „feed-forward“-Signale zu reagieren. Zur Vereinfachung haben wir die proximalen Dendritensegmente von jeder einzelnen Zelle entfernt und mit einem gemeinsamen Dendritensegment ersetzt, welches sich alle Zellen einer Kolumne teilen. Die räumliche Musterzusammenfassung arbeitet auf den gemeinsamen Dendritensegmenten, auf der Ebene der Kolumnen. Die temporale Musterzusammenfassung arbeitet auf den distalen Dendritensegmenten, auf der Ebene der einzelnen Zellen in den Kolumnen. Diese Vereinfachung erreicht dieselbe Funktionalität, obwohl es kein biologisches Äquivalent zu einem zu einer Kolumne zugeordneten Dendritensegment gibt.

**Synapsen**
HTM-Synapsen haben binäre Gewichte. Biologische Synapsen haben variierende Gewichte, aber sie sind auch zum Teil zufällig, dies legt nahe, dass biologische Neuronen sich nicht auf genaue synaptische Gewichte verlassen können. Die Verwendung von verteilten Repräsentationen in HTMs zusammen mit unserem Modell der dendritischen Operationen erlaubt es, ohne nachteilige Effekte binäre Gewichte bei HTM-Synapsen zu verwenden. Um das Bilden und Auflösen von synaptischen Verbindungen zu modellieren, verwenden wir zwei weitere Konzepte aus der Neurowissenschaften, die Ihnen nicht bekannt sein könnten. Eines dieser Konzepte sind die sogenannten potentiellen Synapsen. Dies repräsentiert alle Axone, die nah genug an einem Dendriten vorbeilaufen, so dass sie eine synaptische Verbindung ausformen könnten. Das zweite Konzept wird als Permanenz bezeichnet. Dies ist ein skalarer Wert, der jeder potentiellen Synapse zugewiesen wird. Die Permanenz beschreibt den Grad der synaptischen Verbindung zwischen einem Axon und einem Dendriten. Aus Sicht der Biologie würde die Permanenz folgende Stadien durchlaufen: Vollständig unverbunden, beginnendes Bilden einer Synapse ohne Verbindung, Synapse mit minimalem Kontakt bis zu einer ausgewachsenen großen Synapse. Die Permanenz einer Synapse in HTMs ist ein Wert zwischen 0.0 und 1.0. Lernen beinhaltet das Erhöhen oder Vermindern des Werts der Permanenz. Wenn der Wert der synaptischen Permanenz einen bestimmten Schwellwert übersteigt, dann ist die Synapse mit einem Gewicht von 1 verbunden. Liegt die Permanenz unterhalb des Schwellwerts, dann ist die Synapse unverbunden mit einem Wert von 0.

## Übersicht

Versetzen Sie sich in die Lage einer HTM Region. Ihre Eingabe besteht aus tausenden oder zehntausenden von Bits. Diese Bits repräsentieren sensorische Daten oder sie kommen aus einer anderen Region aus tieferen Ebenen der Hierarchie. Sie schalten sich ein und aus, in sehr komplexen Mustern. Was kann man mit einer solchen Eingabe anfangen?

Die einfachste Form der Antwort haben wir bereits diskutiert. Jede HTM-Region sucht nach häufigen Mustern in ihrer Eingabe und lernt danach Sequenzen dieser Muster. Aus ihrem Speicher dieser Sequenzen bildet die Region Vorhersagen. Diese allgemeine Beschreibung lässt den Prozess einfach erscheinen, aber tatsächlich ist dies ein sehr vielschichtiger und komplexer Vorgang. Im Folgenden soll dieser Prozess in die folgenden drei kleineren Schritte unterteilt werden:

1) Bilden einer spärlichen verteilten Repräsentation der Eingabe
2) Bilden einer Repräsentation im Kontext vorheriger Eingaben
3) Bilden einer Vorhersage im Kontext der aktuellen und der vorherigen Eingaben

Jeder dieser Schritte wird im Folgenden im Detail besprochen.

**1) Bilden einer spärlichen verteilten Repräsentation der Eingabe**

Man stelle sich die Eingabe einer Region als große Anzahl von Bits vor. Im Gehirn wären dies die Axone von Neuronen. Zu jeden Zeitpunkt werden einige dieser Bits aktiv (1) sein und andere inaktiv (0). Der Prozentsatz der aktiven Eingabebits variiert, beispielsweise zwischen 0% und 60%. Der erste Schritt einer HTM-Region ist die Umwandlung dieser Eingabe in eine spärliche Repräsentation. Sind beispielsweise 40% der Eingabebits eingeschaltet, dann sind es in der neuen Repräsentation nur noch 2%.

Eine HTM-Region ist logisch unterteilt in eine Menge von Kolumnen. Jede Kolumne besteht aus einer oder mehreren Zellen. Kolumnen können logisch in einem 2D-Array angeordnet werden, aber dies ist keine Notwendigkeit. Jede Kolumne einer Region ist mit einer eindeutigen Teilmenge von Eingabebits verbunden (oft überlappen sie mit den Eingabebits anderer Kolumnen, sind aber nie exakt gleich). Daraus folgt, dass unterschiedliche Eingaben zu unterschiedlichen Aktivierungsstärken der Kolumnen führen. Die Kolumnen mit der stärksten Aktivierung hemmen oder deaktivieren Kolumnen mit schwächerer Aktivierung. (Hemmung tritt innerhalb eines Radius auf, der von sehr lokal bis regionsübergreifend reichen kann.). Die spärliche Repräsentation der Eingabe ist durch die aktiven und durch Hemmung deaktivierten Kolumnen codiert. Die Hemmungsfunktion ist so definiert, dass eine relativ konstante Anzahl Kolumnen aktiv sind, selbst wenn die Anzahl der aktiven Eingabebits stark variiert.

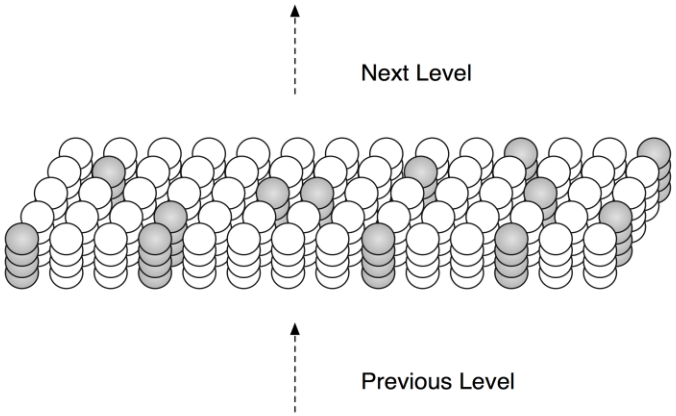



Abbildung 2.1: Eine HTM-Region besteht aus Kolumnen von Zellen. Nur ein kleiner Teil der Region ist hier abgebildet. Jede Kolumne erhält ihre Aktivierung von einer eindeutigen Teilmenge der Eingabe. Kolumnen mit der stärksten Aktivierung hemmen Kolumnen mit schwächerer Aktivierung. Das Ergebnis ist eine spärliche verteilte Repräsentation der Eingabe (aktive Kolumnen sind grau gefärbt).

Man stelle sich nun vor, dass die Eingabe sich verändert. Wenn sich nur wenige Eingabebits ändern, dann werden einige Kolumnen in eingeschaltetem Zustand ein paar mehr oder ein paar weniger Eingaben aufweisen, aber die Menge der aktiven Kolumnen wird sich wahrscheinlich kaum ändern. Daher werden ähnliche Eingabemuster (solche, die in einer signifikanten Anzahl von aktiven Bits übereinstimmen) einer relativ stabilen Menge an aktiven Kolumnen entsprechen. Wie stabil die Codierung ist, hängt zum großen Teil davon ab, mit welchen Eingaben die jeweilige Kolumne verbunden ist. Das Vorgehen zum Erlernen dieser Verbindungen wird später erläutert.

All diese Schritte (das Erlernen der Eingaben jeder Kolumne von einer Teilmenge der Eingabe, das Bestimmen der Größe der Eingabe einer Kolumne und das Nutzen von Hemmung, um eine spärliche Repräsentation zu erhalten) werden als räumliche Musterzusammenfassung (engl. „spatial pooler“) bezeichnet. Der Begriff meint, dass Muster, die einander „räumlich“ ähnlich sind (in dem Sinne, dass sie eine große Zahl gemeinsamer Bits haben) zusammengefasst werden (d.h. sie werden als eine gemeinsame Repräsentation gespeichert).

### 2) Bilden einer Repräsentation im Kontext vorheriger Eingaben

Die nächste Aufgabe, die von einer Region ausgeführt wird, ist das Umwandeln einer kolumnaren Repräsentation der Eingabe in eine neue Repräsentation, welche den Zustand oder Kontext der Vergangenheit beinhaltet. Die neue Repräsentation wird durch das Aktivieren einer Teilmenge der Zellen innerhalb der Kolumnen gebildet, typischerweise durch eine einzelne Zelle pro Kolumne (Abbildung 2.2).

Man betrachte das Hören zweier englischer Sätze, „I ate a pear“ und „I have eight pears“. Die Wörter „ate“ und „eight“ in Homonyme; sie klingen identisch. Wir

können sicher sein, dass es an irgendeiner Stelle im Gehirn Neuronen gibt, die auf die Wörter „ate“ und „eight“ identisch reagieren. Schließlich erreichen zwei identische Lautfolgen das Ohr. Dennoch können wir ebenfalls davon ausgehen, dass an einer anderen Stelle im Gehirn Neuron unterschiedlich auf diese Eingaben reagieren, wenn die Eingaben in unterschiedlichem Kontext vorkommen. Die Repräsentationen für den Klang des Wortes „ate“ wird unterschiedlich sein, je nachdem ob man „I ate“ oder „I have eight“ gehört hat. Man stelle sich vor, man hätte sich die beiden Sätze „I ate a pear“ und „I have eight pears“ auswendig gelernt. Wenn man „I ate…“ hört, dann wird dies zu einer anderen Vorhersage führen, als wenn man „I have eight...“ hört. Es müssen unterschiedliche interne Repräsentationen nach dem Hören von „I ate“ und „I have eight“ vorliegen.

Das Prinzip, eine Eingabe in unterschiedlichen Kontexten unterschiedlich zu kodieren ist eine universale Eigenschaft von Wahrnehmung und Handeln, und ist eine der wichtigsten Funktionen einer HTM-Region. Die Wichtigkeit dieser Fähigkeit kann kaum genug betont werden.

Jede Kolumne in einer HTM-Region besteht aus mehreren Zellen. Alle Zellen in einer Kolumne erhalten dieselbe „feed-forward“-Eingabe. Jede Zelle in einer Kolumne kann aktiv oder inaktiv sein. In dem man unterschiedliche aktive Zellen in jeder aktiven Kolumne wählt, kann man dieselbe Eingabe in unterschiedlichen Kontexten repräsentieren. Ein Beispiel mit Größenangaben kann helfen. Nehmen wir an, jede Kolumne hat 4 Zellen und die Repräsentation jeder Eingabe besteht aus 100 aktiven Kolumnen. Wenn immer nur eine Zelle pro Kolumne aktiv ist, dann gibt es 4^100 Arten, dieselbe Eingabe zu repräsentieren. Dieselbe Eingabe wird immer dazu führen, dass dieselben 100 Kolumnen aktiv sind, aber in unterschiedlichen Kontexten werden unterschiedliche Zellen innerhalb dieser Kolumnen aktiv. Jetzt ist es möglich, dieselbe Eingabe in einer großen Anzahl von Kontexten darzustellen, aber wie eindeutig werden diese verschiedenen Repräsentationen sein? Wenn man zufällig zwei Paare aus den 4^100 möglichen Mustern wählt, werden sie fast immer ungefähr 25 Zellen gemeinsam haben. Das bedeutet, dass zwei Repräsentationen derselben Eingabe in unterschiedlichen Kontexten bei etwa 25 Zellen übereinstimmen und sich bei 75 Zellen unterscheiden, daher können sie einfach auseinander gehalten werden.

Folgende allgemeine Regel wird von einer HTM-Region verwendet: Wenn eine Kolumne aktiv wird, dann werden alle Zellen innerhalb der Kolumne untersucht. Wenn eine oder mehrere Zellen in der Kolumne sich bereits im Vorhersagezustand befinden, dann werden nur diese aktiv. Wenn keine der Zellen der Kolumne im Vorhersagezustand sind, dann werden alle Zellen der Kolumne aktiv. Man kann sich dieses Vorgehen wie folgt vorstellen: Wenn ein Eingabemuster erwartet wurde, dann bestätigt das System diese Erwartung, indem nur die Zellen im vorhersagenden Zustand aktiviert werden. Wenn ein Eingabemuster nicht erwartet wurde, dann wird das System alle Zellen aktivieren, was in etwa das folgende bedeutet: „Die Eingabe war nicht erwartet, daher sind alle möglichen Interpretationen gültig.“

Wenn es keinen vorherigen Zustand gibt, und daher kein Kontext und keine Vorhersage vorhanden sind, dann werden alle Zellen in einer Kolumne aktiv, sobald die Kolumne aktiv wird. Dieses Szenario entspricht dem Hören der ersten Note eines Lieds. Ohne Kontext ist es üblicherweise nicht möglich, vorherzusagen, was als nächstes eintreten wird; alle Möglichkeiten sind offen. Wenn es einen vorherigen Zustand gibt, aber die Eingabe nicht der Erwartung entspricht, dann werden alle Zellen in der aktiven Kolumne aktiv. Diese Entscheidung wird für jede Kolumne einzeln getroffen, d.h. eine erfolgreiche oder erfolglose Vorhersage ist nie ein „Ganz-oder-Gar-nicht"-Ereignis.

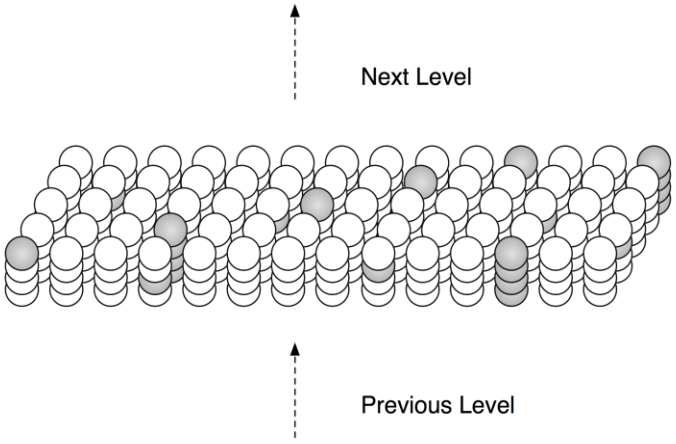



Abbildung 2.2: Indem eine Teilmenge der Zellen der Kolumnen aktiviert wird, kann eine HTM-Region dieselbe Eingabe in vielen verschiedenen Kontexten repräsentieren. Kolumnen aktivieren nur Zellen, die eine Vorhersage getroffen haben. Kolumnen, die keine Zellen mit Vorhersagen beinhalten, aktivieren alle Zellen der Kolumne. Die Abbildung zeigt einige Kolumnen mit einer aktiven Zelle und einige Kolumnen, in denen alle Zellen aktiv sind.

Im Abschnitt zur Terminologie wurde erwähnt, dass HTM-Zellen sich in einem von drei Zuständen befinden können. Wenn eine Zelle aktiv ist durch eine „feed-forward"-Eingabe, dann verwenden wir einfach den Begriff „aktiv". Wenn eine Zelle durch laterale Verbindungen zu anderen Zellen der Umgebung aktiv wird, dann sprechen wir von der Zelle als „in vorhersagendem Zustand" oder „im Vorhersagezustand" (siehe Abbildung 2.3).

**3) Bilden einer Vorhersage im Kontext der aktuellen und vorherigen Eingaben**
Der letzte Schritt in unserer Region ist das Treffen einer Vorhersage über den wahrscheinlichen nächsten Zustand. Die Vorhersage basiert auf der Repräsentation, die im zweiten Schritt entstand, welche den Kontext aus allen vorherigen Eingaben enthält.

Wenn eine Region eine Vorhersage trifft, dann werden alle Zellen in den Vorhersagezustand gesetzt, die voraussichtlich in Zukunft durch „feed-forward"-Signale aktiv werden. Da die Repräsentationen in Regionen spärlich besetz sind, können mehrere Vorhersagen gleichzeitig getroffen werden. Wenn beispielsweise

2% aller Kolumnen durch eine Eingabe aktiv sind, dann kann man erwarten, dass zehn verschiedene Vorhersagen dazu führen, dass etwa 20% aller Kolumnen eine Zelle im Vorhersagezustand beinhalten. Oder aber zwanzig verschiedene Vorhersagen könnten getroffen werden, was dazu führt, dass 40% der Kolumnen eine Zelle im Vorhersagezustand haben. Wenn jede Kolumne vier Zellen beinhaltet, dann würden 10 % aller Zellen im vorhersagenden Zustand sein.

Ein zukünftiges Kapitel über spärlich verteilte Repräsentationen wird zeigen, dass eine Region mit großer Sicherheit bestimmen kann, ob eine bestimmte Eingabe vorhergesagt wurde, obwohl die Repräsentationen überlagert wurden.

Wie trifft eine Region eine Vorhersage? Wenn Eingabemuster sich zeitlich ändern, werden unterschiedliche Mengen von Kolumnen und Zellen in bestimmter Reihenfolge aktiv. Wenn eine Zelle aktiv wird, dann verbindet sie sich mit einer Teilmenge der Zellen in der Umgebung, die unmittelbar zuvor aktiv waren. Diese Verbindungen können sich schnell oder langsam bilden, je nachdem, welche Lernrate für die jeweilige Anwendung nötig ist. Später muss eine Zelle nur diese Verbindungen betrachten, um koinzidente Aktivität auszulösen. Wenn die Verbindungen aktiv werden, dann kann eine Zelle erwarten, dass sie in Kürze selbst aktiviert wird und geht in den Vorhersagezustand über. Daher wird die Weiterleitung der Aktivierung einer Menge von Zellen zum Vorhersagezustand einer anderen Zellmenge führen, die typischerweise im Anschluss aktiv wird. Man kann sich dies wie den Moment vorstellen, in dem man ein Lied erkannt hat und damit beginnt, die nächsten Noten vorherzusagen.

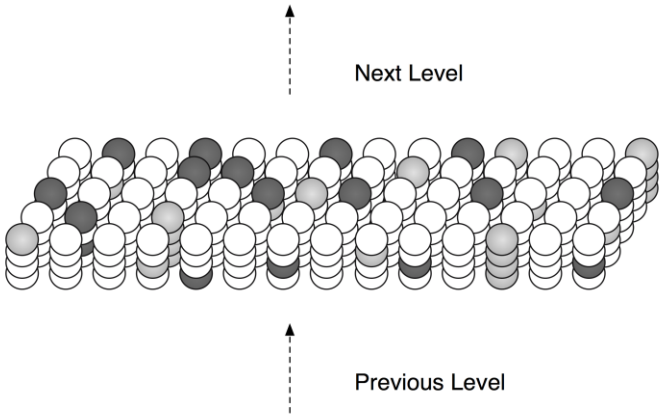



Abbildung 2.3: Zu jedem Zeitpunkt sind einige Zellen durch „feed-forward“-Eingaben aktiv (dargestellt in hellem grau). Andere Zellen, die laterale Eingaben von aktiven Zellen erhalten, werden im vorhersagenden Zustand sein (dargestellt in dunklem Grau)

Zusammenfassend betrachtet führt eine neue Eingabe zu einer spärlichen Menge aktivierter Kolumnen. Eine oder mehrere der Zellen in diesen Kolumnen werden aktiv, diese wiederum führen dazu, dass andere Zellen mittels erlernter Verbindungen einen vorhersagenden Zustand einnehmen. Die Zellen, die durch die

erlernten Verbindungen aktiv werden, stellen eine Vorhersage dar. Wenn die nächste „feed-forward“-Eingabe eintrifft, wird eine andere spärlich besetzte Menge an Kolumnen aktiv. Wenn eine der neu aktivierten Kolumnen unerwartet war, d.h. keine der Zellen im Vorhersagezustand war, dann werden alle Zellen der Kolumne aktiv. Die Ausgabe eine Region ist die Aktivität aller Zellen der Region, inklusive der Zellen, die durch „feed-forward“-Eingaben aktiv wurden und der Zellen, die sich im Vorhersagezustand befinden.

Wie zuvor erwähnt bezieht sich die Vorhersage nicht nur auf den nächsten Zeitpunkt. Vorhersagen in einer HTM-Region können sich einen Zeitpunkt beziehen, der mehrere Schritte in der Zukunft liegt. Betrachtet man Melodien als Beispiel, dann würde eine HTM-Region nicht nur die nächste Note einer Melodie vorhersagen, sondern die nächsten 4 Noten. Dies führt zu einer wünschenswerten Eigenschaft. Die Ausgabe einer Region (also die Vereinigung aller aktiven und vorhergesagten Zellen der Region) verändert sich langsamer als die Eingabe. Man stelle sich vor, eine Region sagt die nächsten 4 Noten einer Melodie voraus. Wir stellen diese Melodie durch die Buchstabensequenz A, B, C, D, E, F, G dar. Nach dem Hören der ersten beiden Töne erkennt die Region die Sequenz und beginnt mit der Vorhersage. Die Vorhersage der Region sind die Noten C, D, E, F. Die Zellen, die B repräsentieren sind bereits aktiv, das bedeutet die Zellen, die B, C, D, E, F beschreiben sind alle in einem der beiden Aktivitätszuständen. Als Nächstes hört die Region „C“. Die Menge der aktiven und vorhersagenden Zellen repräsentiert jetzt die Noten „C, D, E, F, G“. Man kann erkennen, dass das Eingabemuster sich vollständig von „B“ zu „C“ verändert, aber nur 20% der Zellen haben sich verändert.

Da die Ausgabe einer HTM-Region ein Vektor ist, der die Aktivität aller Zellen der Region repräsentiert, ist die Ausgabe 5-mal so stabil wie die Eingabe. In einer hierarchischen Organisation von Regionen kann man also einen Anstieg an temporaler Stabilität beobachten, je weiter man der Hierarchie nach oben folgt.

Wir bezeichnen die beiden Schritte des Hinzufügens von Kontext zu einer Repräsentation und die Vorhersage als temporale Musterzusammenfassung (engl. „temporal pooler“). Durch das Erzeugen von sich langsam verändernden Ausgaben für Sequenzen von Mustern werden im Wesentlichen Muster zusammengefasst, die zeitlich aufeinander folgen.

Im Folgenden werden wir eine detaillierte Betrachtung durchführen. Wir beginnen mit den Konzepten, die die räumliche und die temporale Musterzusammenfassung gemeinsam haben. Anschließend diskutieren wir Konzepte und Details, die nur die räumliche Musterzusammenfassung betreffen, gefolgt von den Konzepten und Details, die die temporale Musterzusammenfassung betreffen.

## Übergreifende Konzepte

Lernen bei der räumlichen und temporalen Musterzusammenfassung sind einander ähnlich. Lernen beinhaltet in beiden Fällen das Erzeugen von Verbindungen zwischen Zellen. Die temporale Musterzusammenfassung erlernt Verbindungen zwischen Zellen aus derselben Region. Die räumliche Zusammenfassung erlernt „feed-forward"-Verbindungen zwischen Eingabebits und Kolumnen.

**Binäre Gewichtung**
Synapsen in HTM-Netzen haben nur die Werte 0 oder 1; Ihre „Gewichtung" ist binär, eine Eigenschaft, die einen Unterschied zu vielen Modellen neuronaler Netze darstellt, die skalare variable Werte zwischen 0 und 1 verwenden.

**Permanenz**
Synapsen entstehen und trennen sich ständig während des Lernens. Wie zuvor erwähnt weisen wir jeder Synapse einen skalaren Wert (zwischen 0.0 und 1.0) zu, der beschreibt, wie permanent die Verbindung ist. Wenn eine Verbindung verstärkt wird, dann wird der Wert der Permanenz erhöht. Unter anderen Umständen kann die Permanenz verringert werden. Wenn die Permanenz über einem Schwellwert liegt (z.B. 0.2), dann wird die Synapse als verbunden betrachtet. Wenn die Permanenz unterhalb des Schwellwertes liegt, dann hat die Synapse keinen Einfluss.

**Dendritische Segmente**
Synapsen verbinden sich mit dendritischen Segmenten. Es gibt zwei verschiedene dendritische Segmente, proximale und distale.
- Ein proximales dendritisches Segment bildet Synapsen mit „feed-forward"-Eingaben. Die aktiven Synapsen aus dieser Art Segmenten werden linear summiert und bestimmen, ob eine Kolumne durch Eingabesignale aktiv wird.
- Ein distales dendritisches Segment bildet Synapsen mit Zellen innerhalb der Region. Jede Zelle hat mehrere distale dendritische Segmente. Wenn die Summe der aktiven Synapsen eines distalen Segments einen Schwellwert überschreitet, dann wird die damit verbundene Zelle vorhersagend aktiv. Da es mehrere distale dendritische Segmente pro Zelle gibt, ergibt sich der Vorhersagezustand einer Zelle aus dem logischen Oder mehrerer einzelner Schwellwert-Detektoren.

**Potentielle Synapsen**
Wie zuvor erwähnt hat jedes dendritische Segment eine Liste von potentiellen Synapsen. Alle potentiellen Synapsen erhalten einen Permanenzwert und können verbundene Synapsen werden, wenn ihr Permanenzwert einen Schwellwert übersteigt.

**Lernen**
Lernen beinhaltet das Erhöhen oder Verringern des Permanenzwertes der potentiellen Synapsen eines dendritischen Segments. Die Regel, die Synapsen mehr oder weniger permanent werden lässt, ist ähnlich der Hebbschen Lernregel. Wenn

beispielsweise eine postsynaptische Zelle aktiv wird, da eines ihrer dendritischen Segmente Eingaben erhält, die dazu führen, dass das Segment den Schwellwert übersteigt, dann werden die Permanenzwerte dieses Segments verändert. Der Permanenzwert der Synapsen, die aktiv waren und damit zur Aktivierung der Zelle beigetragen haben, wird erhöht. Der Permanenzwert der Synapsen, die inaktiv waren und daher nicht zur Aktivierung beigetragen haben, wird verringert. Die genauen Bedingungen, unter denen die synaptischen Permanenzwerte verändert werden unterscheiden sich bei der räumlichen oder temporalen Musterzusammenfassung. Die Details dazu werden im Folgenden beschrieben.

Jetzt wollen wir uns den Konzepten zuwenden, die spezifisch für die Funktionsweise der räumlichen oder temporalen Musterzusammenfassung sind.

## Räumliche Musterzusammenfassung

Die grundlegende Aufgabe der räumlichen Musterzusammenfassung ist die Umwandlung der Eingabe einer Region in ein spärlich besetztes Muster. Diese Funktion ist wichtig, da der Mechanismus, der Sequenzen erlernt und Vorhersagen trifft, zu Beginn spärlich besetzte Muster erfordert.

Es gibt mehrere einander beeinflussende Ziele für die räumliche Musterzusammenfassung, die festlegen, wie die räumliche Musterzusammenfassung arbeitet und lernt.

**1) Nutzung aller Kolumnen**
Eine Region hat eine feste Anzahl Kolumnen, welche erlernen, wie typische Muster der Eingabe repräsentiert werden. Ein Ziel ist es, zu sichern, dass alle Kolumnen erlernen, etwas Nützliches zu erlernen, unabhängig davon, wie viele Kolumnen zur Verfügung stehen. Es ist nicht erwünscht, dass einzelne Kolumnen nie aktiv werden. Um dies zu verhindern, wird mitgeführt, wie oft eine Kolumne im Verhältnis zu ihren Nachbarn aktiv war. Ist die relative Aktivität zu niedrig, dann wird ihr Eingabesignal verstärkt, bis sie beginnt, Teil von aktiven Mengen von Kolumnen zu werden. Im Wesentlichen konkurrieren alle Kolumnen mit ihren Nachbarn darum, Teil einer Repräsentation einer Eingabe zu werden. Wenn eine Kolumne nicht sehr aktiv ist, wird sie aggressiver. Im Zuge dessen werden andere Kolumnen dazu gezwungen, ihre Eingabe zu verändern und damit zu beginnen, leicht veränderte Eingabemuster zu repräsentieren.

**2) Erhalt der gewünschten Dichte**
Eine Region muss eine spärliche besetzte Repräsentation ihrer Eingabe erzeugen. Kolumnen mit den meisten Eingaben hemmen ihre Nachbarn. Es gibt einen Hemmungsradius, der proportional zur Größe des rezeptiven Feldes der Kolumnen ist (und daher von sehr kleiner Größe bis zur Größe der Gesamtregion reichen kann). Innerhalb des Hemmungsradius ist nur ein Prozentsatz der aktivsten Kolumnen als „Gewinner“ erlaubt. Die übrigen Kolumnen werden abgeschaltet (ein

Hemmungsradius erfordert eine zweidimensionale Anordnung der Kolumnen, aber das Konzept kann an andere Topologien angepasst werden).

**3) Vermeidung trivialer Muster**
Es ist erwünscht, dass alle Kolumnen nicht-triviale Teilmuster der Eingabe repräsentieren. Dieses Ziel kann erreicht werden, in dem ein Minimalschwellwert an Eingaben vorgegeben wird, der mindestens erreicht sein muss, damit die Kolumne aktiv wird. Setzt man den Schwellwert beispielsweise auf 50, so bedeutet dies, dass mindestens 50 Synapsen des dendritischen Segments einer Kolumne aktiv sein müssen, damit die Kolumne aktiv wird. Dies garantiert ein gewisses Mindestmaß an Komplexität bezüglich der Muster, die eine Kolumne repräsentiert.

**4) Vermeidung zusätzlicher Verbindungen**
Wenn nicht vorsichtig vorgegangen wird, dann könnte eine Kolumne eine große Anzahl gültiger Synapsen ausbilden. Sie würde dann stark auf eine große Zahl sehr unterschiedlicher Eingabemuster reagieren. Verschiedene Teilmengen der Synapsen würden auf unterschiedliche Muster reagieren. Um dieses Problem zu umgehen, wird der Permanenzwert jeder Synapse, die derzeit nicht zu einer gewinnenden Kolumne beiträgt, verringert. Durch diese Bestrafung nicht beitragender Synapsen wird sichergestellt, dass eine Kolumne nur eine eingeschränkte Zahl an Mustern repräsentiert, manchmal nur eines.

**5) Selbstorganisierende rezeptive Felder**
Echte Gehirne sind in hohem Maße plastisch; Regionen im Neocortex können als Reaktion auf verschiedene Veränderungen lernen, völlig andere Dinge zu repräsentieren. Wenn ein Teil des Neokortex beschäftigt wird, dann werden andere Teile sich anpassen, um das zu repräsentieren, was zuvor der beschädigte Teil repräsentiert hat. Wenn ein sensorisches Organ beschädigt wird oder sich verändert, dann wird der damit verbundene Teil des Neokortex beginnen, etwas anderes zu repräsentieren. Das System ist selbst-organisierend

Es ist erwünscht, dass HTM-Regionen dieselbe Flexibilität aufweisen. Wenn beispielsweise 10000 Kolumnen für eine Region bereitgestellt werden, dann sollte die Region erlernen, wie die Eingaben am besten mit 10000 Kolumnen repräsentiert werden können. Wenn 20000 Kolumnen genutzt werden, dann sollte auch diese Anzahl bestmöglich genutzt werden. Wenn sich die statischen Eigenschaften der Eingabe ändern, dann sollten sich die Kolumnen umorganisieren um die neuen Gegebenheiten bestmöglich zu repräsentieren. Zusammenfassend sollte ein Entwickler eines HTMs in der Lage sein, beliebige Ressourcen für eine Region zu nutzen, und die Region sollte die bestmögliche Repräsentation der Eingabe auf Basis der gegebenen Kolumnen und der Eingabestatistik finden. Im Allgemeinen werden in einer Region mit mehr Kolumnen die einzelnen Kolumnen größere und detailreichere Teilmuster der Eingabe repräsentieren. Typischerweise werden die Kolumnen auch seltener aktiv sein, dennoch wird ein relativ konstanter Spärlichkeitsgrad erhalten bleiben.

Es sind keine neuen Lernregeln nötig um dieses höchst wünschenswerte Ziel zu erreichen. Das Verstärken inaktiver Kolumnen, das Hemmen benachbarter Kolumnen zur Erhaltung der Spärlichkeit, das Einführen von Minimalschwellwerten und das Verwalten einer großen Zahl potentieller Synapsen und das Hinzufügen und Entfernen von Synapsen je nach ihrem Beitrag sind die Mechanismen, die dazu führen, dass das Ensemble der Kolumnen sich dynamisch anpasst um die gewünschte Funktion zu erzielen.

## Details der räumlichen Musterzusammenfassung

Wir können jetzt den Gesamtablauf der räumlichen Musterzusammenfassung beschreiben.

1) Man beginnt mit einer Eingabe, die aus einer festen Anzahl Bits besteht. Diese Bits können sensorische Daten repräsentieren oder aus einer Region einer tieferen Hierarchieebene stammen.

2) Man wählt eine feste Anzahl Kolumnen in der Region, die diese Eingabe empfängt. Jede Kolumne hat ein dazugehöriges dendritisches Segment. Jedes dendritische Segment hat eine Menge potentieller Synapsen die eine Teilmenge der Eingabebits repräsentieren. Jede potentielle Synapse hat einen Permanenzwert. Aufgrund ihrer Permanenzwerte werden einige Synapsen gültig.

3) Für jede Eingabe wird die Anzahl der gültigen Synapsen von jeder Kolumne bestimmt, die mit aktiven Bits verbunden sind.

4) Die Zahl der aktiven Synapsen wird mit einem verstärkenden Faktor multipliziert.

5) Die Kolumnen mit der höchsten Aktivierung nach der Verstärkung hemmen alle Kolumnen der Hemmungsumgebung bis auf einen festen Prozentsatz. Der Radius der Hemmung wird selbst durch die Streuung der Eingabebits bestimmt. Jetzt ist eine spärlich besetzte Menge aktiver Kolumnen entstanden.

6) Für jede aktive Kolumne wird der Permanenzwert der Synapsen angepasst. Der Permanenzwert der Synapsen, die mit aktiven Eingabebits verbunden sind, wird erhöht. Der Permanenzwert der Synapsen, die mit inaktiven Eingabebits verbunden sind, wird verringert. Die Veränderungen an den Permanenzwerten können dazu führen, dass einige gültige Synapsen ungültig werden, und umgekehrt.

## Konzepte der temporalen Musterzusammenfassung

Zu Erinnerung: Die temporale Musterzusammenfassung erlernt Sequenzen und trifft Vorhersagen. Die grundlegende Methode besteht darin, dass eine Zelle, die

aktiv wird, Verbindungen mit Zellen bildet, die kurz zuvor aktiv waren. Zellen können Vorhersagen treffen, indem sie ihre Verbindungen untersuchen. Wenn alle Zellen dies durchführen, dann können sie gemeinsam Muster speichern und abrufen, und sie können vorhersagen, welches Muster wahrscheinlich als nächstes auftritt. Es gibt keine zentrale Speicherung für eine Sequenz von Mustern; stattdessen ist die Speicherung über viele einzelne Zellen verteilt. Da die Speicherung verteilt ist, verhält sich das System gegenüber Rauschen und Fehlern robust. Einzelne Zellen können ausfallen, dies hat meist kaum oder keinen messbaren Effekt.

Es werden im Folgenden einige wichtige Eigenschaften von spärlichen verteilten Repräsentationen erwähnt, die bei der temporalen Musterzusammenfassung genutzt werden.

Betrachten wir eine hypothetische Region, die fortlaufend Repräsentationen bildet, indem sie 200 von 10000 Zellen aktiviert (2% aller Zellen sind also zu einem beliebigen Zeitpunkt aktiv). Wie könnte man sich ein Muster aus 200 Zellen merken und ein bestimmtes Muster erkennen? Ein einfacher Weg besteht darin, eine Liste der für uns interessanten 200 aktiven Zellen zu speichern. Wenn wir dieselben 200 Zellen erneut sehen, können wir das Muster wiedererkennen. Was wäre aber, wenn wir nur eine Liste mit 20 der 200 aktiven Zellen speichern würden und die anderen 180 ignorieren würden? Man könnte vermuten, dass das Merken von nur 20 Zellen zu vielen Fehlern führen würde, da diese 20 Zellen in vielen anderen Mustern aktiv sein könnten. Da die Muster aber groß und spärlich sind (in diesem Beispiel 200 aus 10000 Zellen), ist das Merken von 20 aus 200 Zellen beinah genauso gut wie das Merken aller 200 Zellen. Die Wahrscheinlichkeit für einen Fehler in einem umgesetzten System ist äußerst klein, und wir können den Speicherbedarf damit erheblich verringern.

Die Zellen einer HTM-Region nutzen diese Eigenschaft aus. Jedes dendritische Segment einer Zelle hat eine Menge Verbindungen zu anderen Zellen in der Region. Ein dendritisches Segment knüpft diese Verbindungen als Erkennungsmerkmal des Netzzustands zu einem bestimmten Zeitpunkt. Es könnten sich hunderte oder tausende aktiver Zellen in der Nähe befinden, aber das dendritische Segment muss sich lediglich mit 15 oder 20 davon verbinden. Wenn das dendritische Segment 15 von diesen aktiven Zellen erkennt, kann es ziemlich sicher sein, dass das größere Gesamtmuster gerade auftritt. Diese Technik nennt sich „sub-sampling“ und wird in allen Teilen des HTM-Algorithmus verwendet.

Jede Zelle ist Teil von vielen unterschiedlichen verteilten Mustern und von vielen unterschiedlichen Sequenzen. Eine einzelne Zelle könnte Teil von Dutzenden oder Hunderten von temporalen Übergängen sein. Daher besitzt jede Zelle mehrere dendritische Segmente, nicht nur eines. Idealerweise würde eine Zelle ein dendritisches Segment für jedes zu erkennende Aktivitätsmuster haben. In der Praxis kann ein dendritisches Segment jedoch Verbindungen für mehrere komplett unterschiedliche Muster erlernen, ohne dass die Funktion beeinträchtigt wird.

Beispielsweise könnte ein Segment je 20 Verbindungen für 4 verschiedene Muster erlernen, insgesamt also 80 Verbindungen. Wir setzen jetzt einen Schwellwert, so dass das dendritische Segment aktiv wird, wenn 15 dieser Verbindungen aktiv werden. Dadurch entsteht die Möglichkeit von Fehlern. Es ist mit einer gewissen Wahrscheinlichkeit möglich, dass das dendritische Segment den Schwellwert von 15 aktiven Verbindungen übersteigt, indem verschiedene Teile aus verschiedenen Mustern vermischt werden. Allerdings ist diese Art von Fehler sehr unwahrscheinlich, Grund dafür ist erneut die Spärlichkeit der Repräsentation.

Jetzt kann man erkennen, wie eine Zelle mit ein oder zwei Dutzend dendritischer Segmente und einigen tausend Synapsen hunderte verschiedener Zellzustände erkennen kann.

## Details der temporalen Musterzusammenfassung

Im Folgenden werden die einzelnen Schritte der temporalen Musterzusammenfassung aufgezählt. Wir beginnen mit dem Abschluss der räumlichen Musterzusammenfassung, mit einer Menge von aktiven Kolumnen, die eine „feed-forward"-Eingabe repräsentieren.

1) Wir überprüfen bei jeder aktiven Kolumne, ob Zellen in der Kolumne im vorhersagenden Zustand sind, und falls ja aktivieren wir diese. Falls keine Zellen im vorhersagenden Zustand waren, dann werden alle Zellen der Kolumne aktiviert. Die sich daraus ergebende Menge aktiver Zellen ist die Repräsentation der Eingabe im Kontext vorheriger Eingaben.

2) Wir zählen bei allen dendritischen Segmenten aller Zellen die Anzahl der verbundenen Synapsen, die mit einer aktiven Zelle verbunden sind. Falls die Anzahl den Schwellwert übersteigt, markieren wir das dendritische Segment als aktiv. Zellen mit aktiven dendritischen Segmenten werden in den vorhersagenden Zustand versetzt, es sei denn, sie sind bereits durch „feed-forward"-Eingaben aktiviert worden. Zellen ohne aktive Dendriten, die auch nicht durch die Eingabe aktiviert wurden, bleiben oder werden inaktiv. Die Menge der Zellen im vorhersagenden Zustand bildet die Vorhersage der Region.

3) Wenn ein dendritisches Segment aktiv wird, dann werden die Permanenzwerte aller mit diesem Segment verbundenen Synapsen verändert. Die Permanenz jeder potentiellen Synapse des aktiven Segments wird erhöht, wenn die Synapse mit einer aktiven Zelle verbunden ist, und verringert, wenn die Synapse mit einer inaktiven Zelle verbunden ist. Diese Änderungen an der Permanenz werden als vorläufig markiert.

Dadurch werden jene Synapsen auf Segmenten verändert, die bereits ausreihend trainiert wurden, um das Segment zu aktivieren, um damit zu einer Vorhersage zu

führen. Allerdings sollten wir stets die Vorhersagen zeitlich soweit wie möglich in die Vergangenheit erweitern. Dazu wählen wir ein zweites dendritisches Segment der gleichen Zelle für weiteres Training. Als zweites dendritisches Segment wählen wir eines, welches den Zustand des vorherigen Zeitpunkts am besten repräsentiert. Auch für dieses Segment erhöhen wir die Permanenz der Synapsen, die mit aktiven Zellen verbunden sind, und verringern die Permanenz der Synapsen, die mit inaktiven Zellen verbunden sind, jeweils unter Verwendung des vorherigen Zustands. Diese Änderungen an der Permanenz der Synapsen werden als vorläufig markiert.

4) Wenn eine Zelle durch „feed-forward"-Eingabesignale von inaktiv auf aktiv geschaltet wird, dann besuchen wir alle mit der Zelle verbundenen Synapsen und entfernen die Markierungen der vorläufigen Änderungen. Dadurch wird die Permanenz der Synapsen nur dann geändert, wenn sie die korrekte Aktivierung einer Zelle durch „feed-forward"-Eingabesignale vorhergesagt hat.

5) Wenn eine Zelle von einem aktiven Zustand auf einen inaktiven Zustand schaltet, dann werden die als vorläufig markierten Änderungen an der Permanenz der Synapsen dieser Zelle rückgängig gemacht. Wir wollen die Permanenz von Synapsen, die fälschlicherweise eine Zellaktivierung vorhergesagt haben, nicht verstärken.

Man beachte, dass nur Zellen, die durch „feed-forward"-Eingabesignale aktiv wurden, ihre Aktivität innerhalb der Region weiterreichen, denn andernfalls würden Vorhersagen zu weiteren Vorhersagen führen. Aber alle aktiven Zellen (sowohl die, die durch „feed-forward"-Eingabe aktiv wurden, als auch die, die sich im Vorhersagezustand befinden) bilden die Ausgabe einer Region und werden zur nächsten Region der Hierarchie weitergeleitet.

## Sequenzen erster Ordnung vs. variabler Ordnung und Vorhersage

Es gibt noch einen weiteren Hauptaspekt, der diskutiert werden soll, bevor wir unsere Diskussion der räumlichen und temporalen Musterzusammenfassung abschließen. Dieser Aspekt ist vielleicht nicht für alle Leser interessant und ist nicht nötig, um die Kapitel 3 und 4 zu verstehen.

Welche Auswirkungen hat es, wenn eine Kolumne mehr oder weniger Zellen hat? Was passiert genau, wenn nur eine Zelle pro Kolumne existiert? In dem zuvor benutzen Beispiel haben wir gezeigt, dass eine Repräsentation einer Eingabe, die aus 100 aktiven Kolumnen besteht, mit 4 Zellen pro Kolumne auf 4^100 verschiedene Arten codiert werden kann. Also kann dieselbe Eingabe in vielen verschiedenen Kontexten vorkommen, ohne dass Verwechslungen auftreten. Wenn die Eingabemuster beispielsweise Wörter repräsentieren, dann kann eine Region

viele Sätze speichern, die dieselben Wörter wieder und wieder verwenden, ohne dass Verwechslungen auftreten. Ein Wort wie „Hund“ hat eine eindeutige Repräsentation in verschiedenen Kontexten. Diese Fähigkeit erlaubt es HTM Regionen, Vorhersagen sogenannter variabler Ordnung zu treffen. Eine Vorhersage variabler Ordnung basiert nicht rein darauf, was gerade in diesem Moment passiert, sondern auf Kontextanteilen variabler Größe.

Wenn wir die Zahl der Zellen pro Kolumne auf 5 vergrößern, dann steigt die Anzahl der Codierungen für eine bestimmte Eingabe auf 5^100. Aber beide Zahlen sind so groß, dass eine Erhöhung für viele praktische Anwendungen nicht nützlich ist.

Eine deutliche Verringerung der Zahl der Zellen pro Kolumne macht allerdings einen großen Unterschied.

Wenn wir bis auf eine Zelle pro Kolumne reduzieren, dann verlieren wir die Fähigkeit, Kontext in unseren Repräsentationen zu codieren. Die Eingabe einer Region führt dann immer zu derselben Vorhersage, unabhängig von der vorherigen Aktivität. Mit einer Zelle pro Kolumne ist die HTM-Region ein Speicher erster Ordnung; Vorhersagen basieren nur auf der aktuellen Eingabe.

Die Vorhersage erster Ordnung ist ideal geeignet für eine Art von Problemen, die das Gehirn löst: Statische räumliche Inferenz. Wie zuvor erwähnt kann ein Mensch ein Bild erkennen, auch wenn es ihm so kurz gezeigt wird, dass die Augen sich in dieser Zeit nicht bewegen können. Beim Hören muss man immer eine Sequenz von Mustern hören, um sie zu erkennen. Für gewöhnlich verhält es sich mit der visuellen Wahrnehmung genauso, d.h. üblicherweise verarbeiten wir Sequenzen visueller Bilder. Aber unter gewissen Umständen kann man ein Bild mit nur einem einzigen Sinneseindruck erkennen.

Temporale und statische Erkennung scheint völlig unterschiedliche Inferenzmechanismen zu erfordern. Die eine erfordert das Erkennen von Mustersequenzen und das Treffen von Vorhersagen auf Basis von Kontextinformationen variabler Länge. Die andere erfordert das Erkennen eines statischen räumlichen Musters ohne temporalen Kontext. Eine HTM-Region mit mehreren Zellen pro Kolumne ist ideal geeignet um zeit-basierte Sequenzen zu erkennen, und eine HTM-Region mit einer Zelle pro Kolumne ist ideal geeignet, um statische Bilder zu erkennen. Bei Numenta haben wir viele Experimente durchgeführt, bei denen wir Regionen mit je einer Zelle pro Kolumne auf Bildverarbeitungsprobleme angewendet haben. Die Details dieser Experimente sprengen den Rahmen dieses Kapitels; dennoch werden wir die wichtigen Konzepte behandeln.

Wenn wir Bilder als Eingabe für eine HTM-Region nutzen, dann lernen die Kolumnen der Region, typische räumliche Anordnungen von Pixeln zu repräsentieren. Die Art der Muster, die erlernt werden, sind ähnlich zu dem, was in der Region V1 des Neocortex beobachtet werden kann (eine Region, die in der

Biologie ausführlich untersucht wurde), typischerweise Linien und Ecken verschiedener Ausrichtungen. Wenn man mit bewegten Bildern trainiert, dann erlernt die HTM Region Übergänge zwischen diesen Basisformen. Beispielsweise wird eine vertikale Linie an einer bestimmten Position häufig von einer weiteren vertikalen Linie gefolgt, die nach links oder rechts verschoben wurde. Alle typischen beobachteten Musterübergänge werden von der HTM-Region gespeichert.

Was passiert nun, wenn wir einer Region ein Bild einer nach rechts wandernden Linie zeigen? Wenn in unserer Region nur eine Zelle pro Kolumne existiert, dann wird sie vorhersagen, dass die Linie nach rechts oder nach links wandert. Sie kann keinen Kontext verwenden um zu wissen, wo die Linie in der Vergangenheit war und daher wissen, ob sie sich nach links oder rechts bewegt. Es zeigt sich, dass diese Zellen, die Bestandteil einer einzelligen Kolumne sind, sich wie „komplexe Zellen" im Neokortex verhalten. Die vorhersagende Ausgabe einer solchen Zelle wird aktiv für vertikale Linien an verschiedenen Positionen, unabhängig davon, ob die Linie sich nach rechts, nach links oder überhaupt nicht bewegt. Wir haben darüber hinaus beobachtet, dass eine solche Region sich stabil gegenüber Translation, Skalierung o.ä. verhält, wobei sie die Fähigkeit behält, zwischen verschiedenen Bildern zu unterscheiden. Dieses Verhalten ist nötig für räumliche Invarianz (die Erkennung desselben Musters an verschiedenen Orten im Bild).

Wenn wir dasselbe Experiment mit einer HTM-Region mit mehreren Zellen pro Kolumne durchführen, dann stellen wir fest, dass die Zellen sich wie „richtungsabhängige komplexe Zellen" im Neokortex verhalten. Die vorhersagende Ausgabe einer Zelle wird aktiv für eine Linie, die nach links oder nach rechts wandert, aber nicht für beide.

Wir fassen alle diese Beobachtungen zusammen und stellen die folgende Hypothese auf. Der Neokortex muss sowohl Inferenz und Vorhersage erster Ordnung und variabler Ordnung durchführen. Es gibt in jeder Region des Neokortex 4 oder 5 Schichten von Zellen. Die Schichten unterscheiden sich auf verschiedene Weisen, aber sie alle haben gemeinsame kolumnare Antwortverhalten und eine hohen horizontalen Verbindungsgrad innerhalb ihrer Ebene. Wir spekulieren dass jede Zellschicht im Neokortex eine Variation der hier in diesem Kapitel dargestellten Inferenz- und Lernregeln ausführt. Die verschiedenen Zellschichten spielen dabei unterschiedliche Rollen. Es ist beispielsweise aus anatomischen Studien bekannt, dass Zellen in Schicht 6 Feedback in der Hierarchie erzeugten, und dass die Schicht 5 an motorischem Verhalten beteiligt ist. Die beiden Hauptschichten, in die die Eingabe weitergeleitet wird, sind die Schichten 4 und 3. Wir spekulieren, dass die Zellen in Schicht 4 unabhängig arbeiten, mit einer Zelle pro Kolumne, während hingegen die Zellen in Schicht 3 mit mehreren Zellen pro Kolumne arbeiten. Das bedeutet, dass Regionen in der Nähe von sensorischer Eingabe sowohl Speicher erster Ordnung und Speicher variabler Ordnung haben. Die Speicherung von Sequenzen erster Ordnung (entspricht grob den Neuronen in Schicht 4) ist nützlich zur Erzeugung von Repräsentationen, die invariant gegenüber räumlicher Veränderungen sind. Die Speicherung von Sequenzen variabler Ordnung (entspricht

grob den Neuronen in Schicht 3) ist nützlich für die Inferenz und Vorhersage bewegter Bilder.

Zusammenfassend stellen wir die Hypothese auf, dass ähnliche Algorithmen wie die, die in diesem Kapitel beschrieben wurden, in allen Schichten von Neuronen des Neokortex arbeiten. Die Schichten unterscheiden sich in wichtigen Einzelheiten, wodurch sie unterschiedliche Rollen bezüglich Signalweiterleitung versus Feedback, Aufmerksamkeit und motorischem Verhalten spielen. In Regionen nahe der sensorischen Eingabe ist es nützlich eine Neuronen-Schicht zu haben, die eine Speicherung erster Ordnung durchführt, da dies zu räumlicher Invarianz führt. Bei Numenta haben wir mit HTM-Regionen erster Ordnung (eine Zelle pro Kolumne) mit Bildverarbeitungsproblemen experimentiert. Wir haben ferner mit der Speicherung variabler Ordnung (mehrere Zellen pro Kolumne) zur Erkennung und Vorhersage von Sequenzen variabler Reihenfolge experimentiert. In Zukunft wäre es folgerichtig, zu versuchen, diese beiden Algorithmen in einer einzelnen Region zu kombinieren und die Algorithmen für weitere Zwecke zu erweitern. Dennoch glauben wir, dass viele interessante Problemstellungen gemäß einem Äquivalent der einschichtigen Regionen mit mehreren Zellen pro Kolumne behandelt werden können, entweder allein oder in einer Hierarchie.

# Kapitel 3: Implementierung der räumlichen Musterzusammenfassung und Pseudocode

Dieses Kapitel beinhaltet detaillierten Pseudocode für eine erste Implementierung der räumlichen Musterzusammenfassung. Die Eingabe dieses Codes ist ein Array von binären Eingaben aus einer tieferen Ebene, entweder von sensorischen Daten oder einer tieferen Ebene der Hierarchie. Der Code berechnet die Funktion activeColumns(t) – eine Liste der Kolumnen, die durch die Eingabe zum Zeitpunkt t gewinnen. Diese Liste wird als Eingabe an die temporale Musterzusammenfassung gesendet, die im nächsten Kapitel beschrieben wird, d.h. activeColumns(t) ist die Ausgabe der räumlichen Musterzusammenfassung.

Der Pseudocode gliedert sich in 3 getrennte Phasen, die nacheinander ausgeführt werden:

Phase 1: Es wird die Überdeckung jeder Kolumne mit der Eingabe berechnet
Phase 2: Es werden die Gewinnerkolumnen nach Hemmung berechnet
Phase 3: Die Permanenz der Synapsen und interne Variablen werden angepasst

Obwohl das Lernen der räumlichen Musterzusammenfassung naturgemäß online, also während der Verarbeitung passiert, kann man das Lernen abgeschaltet werden, indem Phase 3 einfach übersprungen wird.

Der Rest des Kapitels beinhaltet den Pseudocode für jeden der drei Schritte. Die verschiedenen Datenstrukturen und Hilfsroutinen, die im Code verwendet werden, sind am Ende des Kapitels definiert.

**Initialisierung**

Bevor die Region Eingaben erhält, wird für jede Kolumne eine Liste von potentiellen Synapsen berechnet. Diese beinhaltet eine zufällige Menge von Eingaben aus dem Eingaberaum. Jede Eingabe wird durch eine Synapse repräsentiert und ihr wird ein zufälliger Permanenzwert zugewiesen. Die zufälligen Permanenzwerte werden nach zwei Kriterien gewählt. Zunächst werden die Werte in einem kleinen Bereich um den connectedPerm gewählt (der minimale Permanenzwert, bei dem eine Synapse als verbunden betrachtet wird). Dies erlaubt es potentiellen Synapsen, nach wenigen Trainingsiterationen verbunden (oder getrennt) zu werden. Weiterhin hat jede Kolumne ein natürliches Zentrum über der Eingaberegion, und die Permanenzwerte tendieren zu diesem Zentrum (sie haben höhere Werte in der Nähe des Zentrums).

### Phase 1: Überlappung

Die erste Phase berechnet für einen gegebenen Eingabevektor die Überlappung jeder Kolumne mit diesem Vektor. Die Überlappung ist schlicht die Anzahl der verbundenen Synapsen mit aktiver Eingabe, multipliziert mit ihrem Boost-Wert. Ist dieser Wert unterhalb des Wertes minOverlap, dann wird die Punktzahl der Überlappung auf 0 gesetzt.

```
1.  for c in columns
2.
3.          overlap(c) = 0
4.          for s in connectedSynapses(c)
5.                  overlap(c) = overlap(c) + input(t, s.sourceInput)
6.
7.          if overlap(c) < minOverlap then
8.                  overlap(c) = 0
9.          else
10.         overlap(c) = overlap(c) * boost(c)
```

### Phase 2: Hemmung

Die zweite Phase berechnet, welche Kolumnen nach dem Hemmungsschritt gewinnen. desiredLocalActivity ist ein Parameter, der bestimmt, wie viele Kolumnen gewinnen. Wenn der Wert des Parameters desiredLocalActivity beispielsweise 10 ist, dann gewinnt eine Kolumne, wenn ihre Überlappungszahl größer als die der Kolumne mit der zehntgrößten Punktzahl innerhalb des Hemmungsradius ist.

```
11. for c in columns
12.
13.     minLocalActivity = kthScore(neighbors(c), desiredLocalActivity)
14.
15.     if overlap(c) > 0 and overlap(c) ≥ minLocalActivity then
16.             activeColumns(t).append(c)
17.
```

**Phase 3: Lernen**

Die dritte Phase führt das Lernen durch; sie verändert die Permanenzwerte aller Synapsen sofern nötig, und auch den Boost-Wert und den Hemmungsradius.

Die Hauptlernregel ist in den Zeilen 20-26 implementiert. Für gewinnende Kolumnen wird für aktive Synapsen der Permanenzwert erhöht, sonst verringert. Permanenzwerte liegen zwischen 0 und 1.

Die Zeilen 28-36 implementieren das Boosting. Es gibt zwei separate Boosting-Mechanismen, die eine Kolumne helfen, Verbindungen zu erlernen. Wenn eine Kolumne nicht oft genug gewinnt (gemessen durch activeDutyCycle), dann wird der allgemeine Boost-Wert erhöht (Zeile 30-32). Alternativ werden die Permanenzwerte erhöht, wenn die verbundenen Synapsen einer Kolumne nicht oft genug mit Eingaben überlappen (Zeilen 34-36). Bemerkung: Wenn das Lernen deaktiviert wird, wird der Wert von boost(c) festgesetzt.

Schließlich wird am Ende von Phase 3 der Hemmungsradius neu berechnet (Zeile 38).

```
18. for c in activeColumns(t)
19.
20.       for s in potentialSynapses(c)
21.             if active(s) then
22.                   s.permanence += permanenceInc
23.                   s.permanence = min(1.0, s.permanence)
24.             else
25.                   s.permanence -= permanenceDec
26.                   s.permanence = max(0.0, s.permanence)
27.
28. for c in columns:
29.
30.       minDutyCycle(c) = 0.01 * maxDutyCycle(neighbors(c))
31.       activeDutyCycle(c) = updateActiveDutyCycle(c)
32.       boost(c) = boostFunction(activeDutyCycle(c), minDutyCycle(c))
33.
34.       overlapDutyCycle(c) = updateOverlapDutyCycle(c)
35.       if overlapDutyCycle(c) < minDutyCycle(c) then
36.             increasePermanences(c, 0.1*connectedPerm)
37.
38. inhibitionRadius = averageReceptiveFieldSize()
39.
```

**Unterstützende Datenstrukturen und Routinen**

Die folgenden Variablen und Datenstrukturen werden im Pseudocode verwendet:

| | |
|---|---|
| columns | Liste aller Kolumnen |
| input(t,j) | Die Eingabe dieser Ebene zum Zeitpunkt t. input(t,j) ist 1, falls die j-te Eingabe 1 ist |
| overlap(*c*) | Die Überlappung der Kolumne c mit dem Eingabemuster |
| activeColumns(*t*) | Die Liste der Indizes der Kolumnen, die durch die Eingabe der tieferen Ebene gewinnen |
| desiredLocalActivity | Ein Parameter der bestimmt, wie viele Kolumnen nach dem Hemmungsschritt aktiv sind |
| inhibitionRadius | Durchschnittliche Größe des rezeptiven Feldes der Kolumnen |
| neighbors(c) | Liste aller Kolumnen innerhalb des Hemmungsradius inhibitionRadius der Kolumne c |
| minOverlap | Die minimale Zahl der Eingaben, die aktiv sein müssen, damit eine Kolumne beim Hemmungsschritt berücksichtigt wird |
| boost(c) | Der Boost-Wert für die Kolumne c, der während des Lernens berechnet wird – er wird genutzt, um die Überlappungszahl für inaktive Kolumnen zu erhöhen |
| synapse | Eine Datenstruktur, welche eine Synapse repräsentiert. Sie enthält einen Permanenzwert und den Index einer Eingabe |
| connectedPerm | Wenn der Permanenzwert einer Synapse größer als dieser Wert ist, dann gilt die Synapse als verbunden |
| potentialSynapses(c) | Eine Liste potentieller Synapsen und ihrer Permanenzwerte |
| connectedSynapses(c) | Eine Teilmenge an Synapsen aus potentialSynapses(c), bei denen der Permanenzwert größer connectedPerm ist. Dies sind die Synapsen die derzeit mit den Eingaben einer tieferen Ebene verbunden sind |
| permanenceInc | Der Wert, um den die Permanenzwerte von Synapsen beim Lernen erhöht werden |

| | |
|---|---|
| permanenceDec | Der Wert, um den die Permanenzwerte von Synapsen beim Lernen verringert werden |
| activeDutyCycle(c) | Ein gleitender Durchschnitt, der repräsentiert, wie oft die Kolumne c aktiv war (z. B. in den letzten 1000 Iterationen) |
| overlapDutyCycle(c) | Ein gleitender Durchschnitt, der repräsentiert, wie oft die Kolumne c signifikant mit der Eingabe überlappt hat (also der Überlappungswert größer als minOverlap war). |
| minDutyCycle(c) | Eine Variable, die die minimal erwünschte Feuerrate einer Zelle repräsentiert. Wenn die Feuerrate einer Zelle unter diesen Wert fällt, dann wird sie mittels des Boosts verstärkt. Dieser Wert ergibt sich als 1 Prozent der maximalen Feuerrate der Nachbarkolumnen. |

Die folgenden Hilfsroutinen werden im oben abgebildeten Code verwendet:

kthScore(cols, k)
: Ermittelt aus einer Liste von Kolumnen den k-höchsten Überlappungswert.

updateActiveDutyCycle(c)
: Berechnet den gleitenden Durchschnitt, wie oft die Kolumne c nach dem Hemmungsschritt aktiv war.

updateOverlapDutyCycle(c)
: Berechnet einen gleitenden Durchschnitt, wie oft eine Kolumne einen Überlappungswert größer als minOverlap hatte.

averageReceptiveFieldSize()
: Der durchschnittliche Radius des verbundenen rezeptiven Feldes aller Kolumnen. Das verbundene rezeptive Feld einer Kolumne enthält nur die verbundenen Synapsen (jene mit einem Permanenzwert größer als connectedPerm). Dies wird verwendet, um die Ausbreitung der lateralen Hemmung zwischen Kolumnen zu bestimmen.

maxDutyCycle(cols)
: Liefert den höchsten Wert activeDutyCycle(c) der gegebenen Kolumnen.

increasePermanences(c, s)
: Erhöht den Permanenzwert jeder Synapse der Kolumne c um den Skalierungsfaktor s.

boostFunction(c)

Liefert den Boost-Wert einer Kolumne zurück. Der Boost-Wert ist ein skalarer Wert >= 1. Wenn activeDutyCyle(c) über dem Wert von minDutyCycle(c) liegt, dann ist der Boost-Wert 1. Der Boost-Wert erhöht sich linear, sobald der activeDutyCyle-Wert einer Kolumne unter den minDutyCycle-Wert fällt.

# Kapitel 4: Implementierung der temporalen Musterzusammenfassung und Pseudocode

Dieses Kapitel beinhaltet den detaillierten Pseudocode für eine erste Implementierung der temporalen Musterzusammenfassung. Die Eingabe dieses Codes ist activeColumns(t), wie von der räumlichen Musterzusammenfassung berechnet. De Code berechnet den Aktivitäts- und den Vorhersagezustand für jede Zelle zum aktuellen Zeitpunkt t. Das boolesche Oder der Aktivitäts- und Vorhersagezustände bildet die Ausgabe der temporalen Musterzusammenfassung für die nächste Ebene.

Der Pseudocode ist in drei getrennte Phasen geteilt, welche nacheinander auftreten:

> Phase 1: berechne den Aktivitätszustand, activeState(t), für jede Zelle
> Phase 2: berechne den Vorhersagezustand, predictiveState(t), für jede Zelle
> Phase 3: aktualisiere die Synapsen

Phase 3 wird nur für das Lernen benötigt. Dennoch beinhalten die Phasen 1 und 2 im Gegensatz zur räumlichen Musterzusammenfassung einige lernspezifische Operationen, wenn das Lernen eingeschaltet wurde. Da die temporale Musterzusammenfassung deutlich komplizierter ist als die räumliche Musterzusammenfassung, werden wir zunächst eine Version der temporalen Musterzusammenfassung angeben, die nur Inferenz durchführt, gefolgt von einer Version, die Inferenz und Lernen vereinigt. Beschreibungen der Implementierungsdetails, Terminologie und Hilfsroutinen finden sich am Ende des Kapitels nach dem Pseudocode.

## Pseudocode für die temporale Musterzusammenfassung – nur Inferenz

### Phase 1

Die erste Phase berechnet den Aktivitätszustand für jede Zelle. Für jede Gewinnerkolumne legen wir fest, welche Zellen aktiv sein sollten. Wenn die „feed-forward"-Eingabe von einer Zelle vorhergesagt wurde (d.h. ihr prediciveState den Wert 1 durch ein Sequenz-Segment im vorhergehenden Schritt erhalten hat), dann werden diese Zellen aktiv (Zeilen 4-9). Wenn die „feed-forward"-Eingabe unerwartet war (d.h. keine Zellen sich im vorhersagenden Zustand befanden), dann wird jede Zelle der Kolumne aktiv (Zeilen 11-13).

```
1.  for c in activeColumns(t)
2.
3.      buPredicted = false
4.      for i = 0 to cellsPerColumn - 1
5.          if predictiveState(c, i, t-1) == true then
6.              s = getActiveSegment(c, i, t-1, activeState)
7.              if s.sequenceSegment == true then
8.                  buPredicted = true
9.                  activeState(c, i, t) = 1
10.
11.     if buPredicted == false then
12.         for i = 0 to cellsPerColumn - 1
13.             activeState(c, i, t) = 1
```

### Phase 2

Die zweite Phase berechnet den Vorhersagezustand jeder Zelle. Eine Zelle wird ihren Vorhersagezustand aktivieren, wenn eines ihrer Segmente aktiv wird, d.h. wenn genügend horizontale Verbindungen durch „feed-forward"-Eingabe feuern.

```
14. for c, i in cells
15.     for s in segments(c, i)
16.         if segmentActive(c, i, s, t) then
17.             predictiveState(c, i, t) = 1
```

## Pseudocode für die temporale Musterzusammenfassung – kombinierte Inferenz und Lernen

### Phase 1

Die erste Phase berechnet den Aktivitätszustand für jede Zelle, die ein einer Gewinnerkolumne liegt. Aus diesen Zellen wird darüber hinaus eine Zelle pro Kolumne als Lernzelle (lernState) ausgewählt. Dieser kann man wie folgt begründen: Wenn die „feed-forward"-Eingabe von einer Zelle vorhergesagt wurde (d.h. ihr prediciveState den Wert 1 durch ein Sequenz-Segment im vorhergehenden Schritt erhalten hat), dann werden diese Zellen aktiv (Zeilen 23-27). Wenn dieses Segment durch Zellen aktiviert wurde, die selbst den lernState aktiviert hatten, dann wird auch diese Zelle als Lernzelle ausgewählt (Zeilen 28-30). Wenn die „feed-forward"- Eingabe unerwartet, dann wird jede Zelle der Kolumne aktiv (Zeilen 32-24). Zusätzlich wird die am besten passende Zelle als Lernzelle ausgewählt (Zeilen 36-41), und dieser Zelle wird ein neues Segment hinzugefügt.

```
18. for c in activeColumns(t)
19.
20.     buPredicted = false
21.     lcChosen = false
22.     for i = 0 to cellsPerColumn - 1
23.         if predictiveState(c, i, t-1) == true then
24.             s = getActiveSegment(c, i, t-1, activeState)
25.             if s.sequenceSegment == true then
26.                 buPredicted = true
27.                 activeState(c, i, t) = 1
28.                 if segmentActive(s, t-1, learnState) then
29.                     lcChosen = true
30.                     learnState(c, i, t) = 1
31.
32.     if buPredicted == false then
33.         for i = 0 to cellsPerColumn - 1
34.             activeState(c, i, t) = 1
35.
36.     if lcChosen == false then
37.         i = getBestMatchingCell(c, t-1)
38.         learnState(c, i, t) = 1
39.         sUpdate = getSegmentActiveSynapses (c, i, -1, t-1, true)
40.         sUpdate.sequenceSegment = true
41.         segmentUpdateList.add(sUpdate)
```

**Phase 2**

Die zweite Phase berechnet den Vorhersagezustand jeder Zelle. Eine Zelle wird ihren Vorhersagezustand aktivieren, wenn eines ihrer Segmente aktiv wird, d.h. wenn genügend horizontale Verbindungen durch die „feed-forward"-Eingabe feuern. In diesem Fall werden die folgenden Änderung in einer Warteliste eingetragen: a) Verstärkung der aktiven Segmente (Zeilen 47-48) und b) Verstärkung eines Segments, welches die aktuelle Aktivierung hätte vorhersagen können, d.h. ein Segment welches eine (eventuell schwache) Übereinstimmung mit der Aktivität des vorherigen Zeitpunkts hat (Zeilen 50-53).

```
42. for c, i in cells
43.     for s in segments(c, i)
44.         if segmentActive(s, t, activeState) then
45.             predictiveState(c, i, t) = 1
46.
47.             activeUpdate = getSegmentActiveSynapses (c, i, s, t, false)
48.             segmentUpdateList.add(activeUpdate)
49.
50.             predSegment = getBestMatchingSegment(c, i, t-1)
51.             predUpdate = getSegmentActiveSynapses(
52.                                 c, i, predSegment, t-1, true)
53.             segmentUpdateList.add(predUpdate)
```

**Phase 3**

Die dritte und letzte Phase führt das eigentliche Lernen durch. In dieser Phase werden die Segmentveränderungen aus den Wartelisten tatsächlich umgesetzt, sobald „feed-forward"-Eingaben erhalten werden und die Zelle als Lernzelle gewählt wird (Zeilen 56-57). Andernfalls, falls die Zelle den Vorhersagezustand verlässt, werden die Segmente negativ verstärkt (Zeilen 58-60).

```
54. for c, i in cells
55.     if learnState(s, i, t) == 1 then
56.         adaptSegments (segmentUpdateList(c, i), true)
57.         segmentUpdateList(c, i).delete()
58.     else if predictiveState(c, i, t) == 0 and predictiveState(c, i, t-1)==1 then
59.         adaptSegments (segmentUpdateList(c,i), false)
60.         segmentUpdateList(c, i).delete()
61.
```

## Details zur Implementierung und Terminologie

In diesem Abschnitt beschreiben wir einige Details unserer Implementierung der temporalen Musterzusammenfassung und der Terminologie. Jede Zelle ist über zwei Zahlen indiziert: ein Index einer Kolumne c und ein Zellenindex i. Zellen halten eine Liste von dendritischen Segmenten vor, wobei jedes Segment eine Liste von Synapsen zusammen mit ihrem Permanenzwert enthält. Veränderungen an den Synapsen eine Zelle werden als vorläufig markiert, bis die Zelle durch „feed-forward"-Eingaben aktiviert wird. Diese vorläufigen Änderungen werden in einer Liste namens segmentUpdateList verwaltet. Jedes Segment enthält außerdem eine boolesche Markierung, sequenceSegment, welches festlegt, ob dieses Segment Vorhersagen über „feed-forward"-Eingaben zum nächsten Zeitpunkt trifft.

Die Implementierung der potentiellen Synapsen unterscheidet sich von der Implementierung bei der räumlichen Musterzusammenfassung. Bei der räumlichen Musterzusammenfassung wird eine vollständige Liste aller potentiellen Synapsen durch eine explizite Liste repräsentiert. Bei der temporalen Musterzusammenfassung kann jedes Segment seine eigene (möglicherweise große) Liste potentieller Synapsen haben. Bei praktischer Umsetzung ist die Verwaltung einer langen Liste an potentiellen Synapsen rechenintensiv und verbraucht viel Speicher. Daher werden bei der temporalen Musterzusammenfassung beim Lernen zufällig aktive Synapsen zu jedem Segment (beeinflusst durch den Parameter newSynapseCount). Diese Optimierung hat eine ähnliche Wirkung wie das Verwalten einer vollständigen Liste potentieller Synapsen, aber die Liste pro Segment ist deutlich kleiner, gleichzeitig wird die Fähigkeit zum Erlernen neuer temporaler Muster erhalten.

Der Pseudocode nutzt außerdem eine kleine Zustandsmaschine, um die Zellzustände über den Zeitverlauf zu verfolgen. Wir speichern drei verschiedene Zustände für jede Zelle. Die Arrays activeState und predictiveState speichern die Aktivitäts- und Vorhersagezustände jeder Zelle zu jedem Zeitpunkt. Das Array learnState entscheidet, welche Zellausgaben beim Lernen verwendet werden. Wenn eine Eingabe unerwartet war, dann werden alle Zellen einer bestimmten Kolumne zum gleichen Zeitpunkt aktiv. Nur bei einer dieser Zellen (bei derjenigen, die am besten zur Eingabe passt) wird der Lernzustand learnState aktiviert. Wir fügen nur Synapsen zu Zellen hinzu, deren Lernzustand aktiviert ist (dies verhindert die Überrepräsentation von vollständig aktivierten Kolumnen in dendritischen Segmenten).

Die folgenden Datenstrukturen werden im Pseudocode für die temporale Musterzusammenfassung verwendet:

| | |
|---|---|
| cell(c,i) | Eine Liste aller Zellen, indiziert durch *i* und *c* |
| cellsPerColumn | Anzahl Zellen pro Kolumne |
| activeColumns(*t*) | Liste der Indizes aller Gewinnerkolumnen durch „feed-forward"-Eingaben (dies ist die Ausgabe der räumlichen Musterzusammenfassung) |
| activeState(c, i, t) | Ein boolescher Vektor mit einer Zahl pro Zelle. Er repräsentiert den Aktivitätszustand der i-ten Zelle in der Kolumne c zum Zeitpunkt t bei gegebener Eingabe und Berücksichtigung des vorherigen temporalen Kontext. activeState(c, i, t) ist der Beitrag der i-ten Zelle der Kolumne c zum Zeitpunkt t. Ist der Wert 1, dann hat die Zelle sowohl Eingabe als auch einen passenden temporalen Kontext |
| predictiveState(c, i, t) | Ein boolescher Vektor mit einer Zahl pro Zelle. Er repräsentiert die Vorhersage der i-ten Zelle der Kolumne c zum Zeitpunkt t. activeState(c, i, t) ist der Beitrag der i-ten Zelle der Kolumne c zum Zeitpunkt t. Ist der Wert 1, dann sagt die Zelle eine zukünftige Eingabe im derzeitigen temporalen Kontext voraus |
| learnState(c, i, t) | Ein boolescher Wert, der bestimmt, ob die Zelle i der Kolumne c als Lernzelle gewählt wurde. |
| activationThreshold | Schwellwert für die Aktivierung von Segmenten. Wenn die Anzahl der verbundenen Synapsen in einem Segment größer als activationThreshold ist, dann wird das Segment als aktiv bezeichnet |
| learningRadius | Die Fläche um eine Zelle bei der temporalen Musterzusammenfassung, aus der die Zelle laterale Eingabe erhalten kann |
| initialPerm | Der anfängliche Permanenzwert einer Synapse |
| connectedPerm | Ist der Permanenzwert einer Synapse größer als dieser Wert, dann gilt die Synapse als verbunden |
| minThreshold | Minimale Aktivität in einem Segment für das Lernen |
| newSynapseCount | Maximale Anzahl Synapsen, die während des Lernens zu einem Segment hinzugefügt werden können. |

| | |
|---|---|
| permanenceInc | Wert, um den die Permanenzwerte von Synapsen erhöht werden, wenn aktivitätsbasiertes Lernen aktiviert wurde |
| permanenceDec | Wert, um den die Permanenzwerte von Synapsen verringert werden, wenn aktivitätsbasiertes Lernen aktiviert wurde |
| segmentUpdate | Eine Datenstruktur, die drei Informationen enthält: a) ein Segmentindex (-1 steht dabei für ein neues Segment), b) eine Liste von bestehenden aktiven Synapsen und c) eine Markierung, ob dieses Segment als Sequenzsegment markiert werden soll (Voreinstellung ist dabei **false**) |
| segmentUpdateList | Eine Liste von segmentUpdate-Strukturen. segmentUpdateList(c, i) enthält die Liste aller Änderungen für der Zelle i in der Kolumne c. |

Die folgenden unterstützenden Routinen werden im oben abgebildeten Code verwendet:

segmentActive(s, t, state)

Diese Routine ergibt **true**, falls die Anzahl der verbundenen Synapsen im Segment s, welche durch den gegebenen Zustand state zum Zeitpunkt t aktiviert wurden größer ist als der Wert activationThreshold. Der Parameter state kann dabei die Werte activeState oder learnState einnehmen.

getActiveSegment(c, i, t, state)

Für eine Zelle i in der einer Kolumne c wird ein Segmentindex zurückgeliefert, so dass segmentActive(s, t, state) **true** ergibt. Wenn mehrere Segmente aktiv sind, dann werden Sequenzsegmente bevorzugt. Ansonsten werden die Segmente mit der meisten Aktivität bevorzugt

getBestMatchingSegment(c, i, t)

Für eine gegebene Zelle i einer Kolumne c wird das Segment mit der größten Anzahl aktiver Synapsen gesucht. Diese Routine versucht aggressiv, die beste Übereinstimmung zu finden. Die Permanenzwerte der Synapsen dürfen dabei unterhalb der Aktivierungsschwelle connectedPerm liegen. Die Anzahl der aktiven Synapsen darf unterhalb der Aktivierungsschwelle activationThreshold liegen, aber sie muss oberhalb der Schwelle minThreshold liegen. Die Routine liefert den Segmentindex zurück. Wenn kein Segment gefunden wurde, liefert sie -1 zurück.

getBestMatchingCell(c)

Für eine gegebene Kolumne wird die Zelle mit dem am besten übereinstimmenden Segment (wie zuvor definiert) .zurückgeliefert. Wenn keine Zelle ein übereinstimmendes Segment hat, dann wird die Zelle mit der geringsten Anzahl an Segmenten zurückgeliefert.

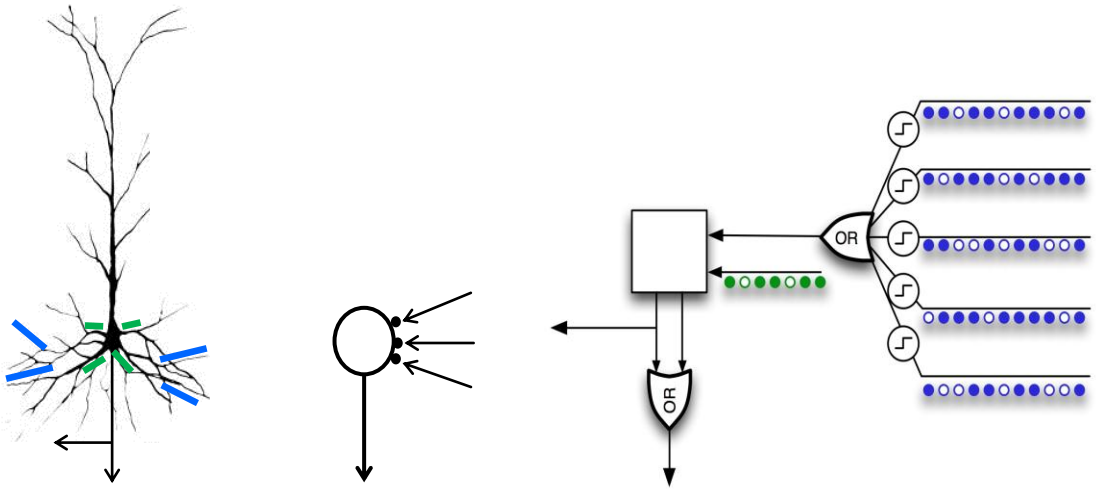
getSegmentActiveSynapses(c, i, t, s, newSynapses= **false**)

Liefert eine SegmentUpdate-Datenstruktur zurück, die eine Liste von Änderungsvorschlägen für das Segment s enthält. Sei activeSynapses die Liste der aktiven Synapsen, bei denen die Ursprungszellen den Zustand activeState = 1 zum Zeitpunkt t haben (diese Liste ist leer, wenn s = -1 ist, da es dann kein Segment gibt). newSynapses ist dabei ein optionaler Parameter, der als Standardwert **false** enthält. Enthält newSynapses den Wert **true**, dann werden newSynapsesCount – Anzahl(activeSynapses) Synapsen zur Liste activeSynapses hinzugefügt. Diese Synapsen werden zufällig aus der Liste der Zellen ausgewählt, die zum Zeitpunkt t den Zustand learnState = 1 haben.

adaptSegments(segmentList, positiveReinforcement)

Diese Funktion iteriert durch eine Liste von SegmentUpdate-Strukturen und verstärkt jedes Segment. Mit jedem SegmentUpdate-Element werden die folgenden Änderungen durchgeführt: Wenn der Parameter positiveReinforcement den Wert **true** hat, dann werden bei allen Synapsen der Aktiv-Liste die Permanenzwerte um permanenceInc erhöht. Bei allen anderen Synapsen werden die Permanenzwerte um permanenceDec verringert. Wenn der Parameter positiveReinforcement den Wert **false** enthält, dann werden bei allen Synapsen der Aktiv-Liste die Permanenzwerte um permanenceDec verringert. Nach diesem Schritt werden alle weiteren Synapsen mit einem Permanenzwert von initialPerm hinzugefügt.

# Appendix A: Vergleich biologischer Neuronen mit HTM-Zellen




Das obige Bild zeigt eine Abbildung eines biologischen Neurons auf der linken Seite, ein einfaches künstliches Neuron in der Mitte und ein HTM-Neuron oder „Zelle“ auf der rechten Seite. Der Zweck dieses Anhangs ist es, ein besseres Verständnis über HTM-Zellen und ihre Funktionsweise zu ermöglichen, indem man sie mit echten Neuronen und einfacheren künstlichen Neuronen vergleicht.

Echte Neuronen sind ungemein kompliziert und vielfältig. Wir werden uns auf die allgemeinsten Prinzipien konzentrieren und auch nur auf jene, die für unsere Modelle eine Rolle spielen. Obwohl wir viele Details realer Neuronen ignorieren, sind die Zellen, die in den HTM-basierten kortikalen Lernalgorithmen verwendet werden, wesentlich realistischer als die künstlichen Neuronen, die in den meisten neuronalen Netzen Verwendung finden. Alle in den HTM-Zellen genutzten Elemente sind notwendig für das Funktionieren einer HTM-Region.

## Biologische Neuronen

Neuronen sind die informationstragenden Zellen im Gehirn. Das obere linke Bild zeigt ein typisches exzitatorisches Neuron. Das visuelle Erscheinungsbild des Neurons ist geprägt durch die verzweigten Dendriten. Alle exzitatorischen Eingaben eines Neurons finden über Synapsen statt, die an diesen Dendriten angeordnet sind. In den letzten Jahren hat unser Wissen über Neuronen deutliche Fortschritte gemacht. Die größte Veränderung war die Erkenntnis, dass die Dendriten eines Neurons nicht nur Leitungen sind, die Signale zum Zellkörper transportieren. Wir wissen jetzt, dass Dendriten selbst komplexe nichtlineare Verarbeitungselemente sind. Die HTM-basierten kortikalen Lernalgorithmen nutzen diese nicht-linearen Eigenschaften.

Neuronen bestehen aus mehreren Teilen:

**Zellkörper:**
Der Zellkörper ist der kleine Raum im Zentrum des Neurons. Die Ausgabe der Zelle, das Axon, entspringt am Zellkörper. Die Eingaben der Zelle sind die Synapsen, die entlang der Dendriten angebracht sind, welche zum Zellkörper leiten.

**Proximale Dendriten**
Die dendritischen Äste, die am nächsten beim Zellkörper liegen, werden proximale Dendriten genannt. Im Diagramm sind einige proximale Dendriten mit grünen Linien gekennzeichnet.

Mehrere aktive Synapsen auf proximalen Dendriten haben grob einen additiven Effekt auf den Zellkörper. Fünf aktive Synapsen werden ungefähr zur fünffachen Depolarisation am Zellkörper führen, im Vergleich zu einer aktiven Synapse. Im Gegensatz dazu wird eine Synapse, die durch eine schnelle Folge von Aktionspotentialen wiederholt aktiviert wird, mit dem zweiten, dritten oder weiteren Aktionspotential einen deutlich geringeren Effekt auf den Zellkörper auslösen als mit dem ersten.

Daher können wir sagen, dass die Eingaben an proximalen Dendriten sich im Zellkörper linear aufsummieren und dass schnelle Spikes, die an einer einzelnen Synapse ankommen, nur einen geringfügig größeren Effekt als ein einzelner Spike haben.

Weiterleitende Verbindungen zu einer Region im Neokortex verbinden sich bevorzugt mit proximalen Dendriten. Dies wurde zumindest für Neuronen aus der Schicht 4 berichtet, die primäre Eingabeschicht von Neuronen in einer Region.

**Distale Dendriten**
Die Dendritenäste, die weiter entfernt vom Zellkörper liegen, werden distale Dendriten genannt. In der Abbildung sind einige distale Dendriten mit blauen Linien markiert.

Distale Dendriten sind dünner als proximale Dendriten. Sie verbinden sich mit anderen Dendriten an Verzweigungen im dendritischen Baum und verbinden sich nicht direkt mit dem Zellkörper. Diese Unterschiede geben distalen Dendriten besondere elektrische und chemische Eigenschaften. Wenn eine einzelne Synapse eines distalen Dendriten aktiviert wird, dann ist der Effekt auf den Zellkörper minimal. Die lokale Depolarisation, die lokal an der Synapse auftritt, schwächt bis zum Erreichen des Zellkörpers ab. Viele Jahre lang wurde dies als Rätsel gesehen. Es schien, als könnten die distalen Synapsen, die einen Großteil der Synapsen eines Neurons ausmachen, nicht viel ausrichten.

Heute wissen wir, dass Abschnitte der distalen Dendriten als halb unabhängige Verarbeitungseinheiten dienen. Wenn ausreichend viele Synapsen zur selben Zeit

innerhalb einer kurzen Distanz entlang des Dendriten aktiv werden, dann können sie einen dendritischen Spike auslösen, der zum Zellkörper geleitet wird und großen Effekt hat. Beispielsweise lösen zwanzig aktive Synapsen im Abstand von 40 um voneinander einen dendritischen Spike aus.

Daher können wir sagen, dass die distalen Dendriten sich verhalten wie eine Menge an Koinzidenzdetektoren mit einem Schwellwert.

Die Synapsen, die sich an distalen Dendriten bilden, sind überwiegend Verbindungen zu anderen Zellen nahe in der Region.

Das Bild zeigt einen großen dendritischen Ast, der sich nach oben erstreckt, dieser wird apikaler Dendrit genannt. Eine Theorie besagt, dass diese Struktur es einem Neuron erlaubt, mehrere distale Dendriten in einem Gebiet zu platzieren, in dem es leichter Verbindungen zu entlanglaufenden Axonen eingehen kann. In dieser Interpretation agiert der apikale Dendrit als Verlängerung der Zelle.

**Synapsen**
Ein typisches Neuron könnte mehrere tausend Synapsen haben. Die große Mehrheit davon (vielleicht 90%) befindet sich an distalen Dendriten, der Rest an proximalen Dendriten.

Viele Jahre lang wurde angenommen, dass Lernen etwas mit dem Stärken und Schwächen der Auswirkung oder dem „Gewicht“ einer Synapse zu tun hat. Obwohl dieser Effekt beobachtet werden konnte, verhält sich jede Synapse einigermaßen stochastisch. Wenn sie aktiviert wird, dann wird sie unbedingt zuverlässig Neurotransmitter ausschütten. Daher können die Algorithmen im Gehirn nicht von der Präzision oder Genauigkeit einzelner synaptischer Gewichte abhängen.

Darüber hinaus weiß man heute, dass ganze Synapsen sich schnell bilden oder auflösen können. Diese Flexibilität repräsentiert eine mächtige Art des Lernens und erklärt die schnelle Aneignung von Wissen auf bessere Weise. Eine Synapse kann sich nur bilden, wenn ein Axon und ein Dendrit innerhalb einer gewissen Distanz liegen, dies führt zum Konzept einer „potentiellen“ Synapse. Auf Basis dieser Annahmen erfolgt Lernen zum großen Teil durch Bildung von tatsächlichen Synapsen aus potentiellen Synapsen.

**Neuronale Ausgaben**
Die Ausgabe eines Neurons ist ein Spike, oder „Aktions-Potential“, der sich entlang des Axons ausbreitet. Das Axon entspringt aus dem Zellkörper und teilt sich fast immer in zwei Teile. Ein Ast verläuft horizontal und knüpft viele Verbindungen mit anderen in der Nähe gelegenen Zellen. Der andere Ast projiziert in andere Schichten oder andernorts im Gehirn. Im Bild des Neurons weiter oben war das Axon nicht sichtbar. Wir haben eine Linie und zwei Pfeile hinzugefügt, die das Axon repräsentieren.

Obwohl die tatsächliche Ausgabe eines Neurons immer ein Spike ist, gibt es unterschiedliche Sichtweisen, wie dies zu interpretieren ist. Die vorherrschende Sichtweise (besonders im Hinblick auf den Neokortex) ist es, dass die Spikerate entscheidend ist. Daher kann die Ausgabe einer Zelle als skalarer Werte betrachtet werden.

Einige Neuronen zeigen ferner ein bündelndes Verhalten, eine kurze und schnelle Serie einiger Spikes, die sich von den herkömmlichen Spikemustern unterscheiden.

Die obige Beschreibung eines Neurons soll eine kurze Einführung zu Neuronen geben. Sie konzentriert sich auf Eigenschaften, die mit Merkmalen von HTM Zellen zusammenhängen, viele Details wurden dabei weggelassen. Nicht alle der eben beschriebenen Merkmale sind allgemein anerkannt. Wir führen diese hier auf, da sie für unsere Modelle notwendig sind. Das Wissen über Neuronen könnte ohne weiteres mehrere Bücher füllen, und die wissenschaftliche Untersuchung von Neuronen wird auch weiterhin sehr aktiv bleiben.

## Einfache künstliche Neuronen

Das mittlere Bild zu Beginn dieses Anhangs zeigt ein Neuronen-ähnliches Element, wie es in vielen klassischen künstlichen neuronalen Netzmodellen verwendet wird. Diese künstlichen Neuronen haben eine Menge von Synapsen, die jeweils mit einer Gewichtung versehen sind. Jede Synapse erhält eine skalare Aktivierung, die mit dem synaptischen Gewicht multipliziert wird. Die Ausgabe aller Synapsen wird auf nicht-lineare Weise summiert, um eine Ausgabe des künstlichen Neurons zu berechnen. Lernen wird durch Anpassung der synaptischen Gewichte und eventuell durch das Anpassen der nichtlinearen Funktion ermöglicht.

Diese Art von künstlichen Neuronen und ihre Variationen haben ihre Nützlichkeit als wertvolles Berechnungshilfsmittel in vielen Anwendungen erwiesen. Dennoch erfassen sie nicht viel der Komplexität und Berechnungsstärke biologischer Neuronen. Wenn wir verstehen und modellieren wollen, wie eine Gruppe realer Neurone im Gehirn funktioniert,

## HTM-Zellen

Das linke Bild unserer Abbildung stellt eine Zelle dar, wie sie in den HTM kortikalen Lernalgorithmen verwendet wird. Eine HTM Zelle umfasst viele wichtige Fähigkeiten realer Neurone, aber es werden auch einige Vereinfachungen getroffen.

**Proximale Dendriten**
Jede HTM Zelle besitzt einen einzelnen proximalen Dendriten. Alle „feed-forward“-Eingaben zur Zelle werden über Synapsen geleitet (dargestellt als grüne Punkte). Die Aktivierung der Synapsen wird linear aufsummiert um eine weiterleitende Aktivierung der Zelle zu erreichen.

Wir fordern, dass alle Zellen eine Kolumne das gleiche Reaktionsverhalten bezüglich „feed-forward“-Eingaben zeigen. Mit realen Neuronen würde dies wahrscheinlich mit einer Art hemmender Zelle erreicht werden. In HTMs zwingen wir einfach alle Zellen einer Kolumne, einen einzelnen proximalen Dendriten gemeinsam zu verwenden.

Um zu vermeiden, dass es Zellen gibt, die nie den Wettkampf mit ihren Nachbarzellen gewinnen, wird eine HTM Zelle die Aktivität ihrer Aktivierung durch „feed-forward“-Signale verstärken, wenn sie in Relation zu ihren Nachbarn nicht häufig genug gewinnt. Also gibt es zwischen den Zellen einen beständigen Wettkampf. Erneut wird dies in HTM als Wettkampf zwischen Kolumnen, nicht zwischen Zellen modelliert. Dieser Wettkampf ist nicht im Diagramm illustriert.

Zuletzt hat der proximale Dendrit eine Menge potentieller Synapsen, die eine Teilmenge aller möglichen Eingaben der Region ist. Beim Lernen wird der Permanenzwert der aller potentiellen Synapsen des proximalen Dendriten erhöht oder verringert. Nur jene Synapsen mit einem Permanenzwert über dem Schwellwert sind gültige Synapsen. Wie zuvor beschrieben stammt das Konzept der potentiellen Synapsen aus der Biologie, es bezeichnet Axone und Dendriten, die nahe genug beieinander liegen, dass eine Synapse ausgeformt werden könnte. Wir erweitern dieses Konzept, in dem wir für HTM Zellen große Mengen an potentiellen Synapsen wählen. Dendriten und Axone von biologischen Neuronen können während des Lernens wachsen und sich zurückziehen, daher kann sich die Menge der potentiellen Synapsen mit dem Wachstum ändern. In dem wir die Menge der potentiellen Synapsen einer HTM Zelle groß wählen, erzielen wir ungefähr das gleiche Ergebnis wie durch axonales und dendritisches Wachstum. Die Menge der potentiellen Synapsen ist nicht abgebildet.

Die Kombination aus Wettkampf zwischen Kolumnen, Lernen aus einer Menge von potentiellen Synapsen und das Verstärken weniger genutzter Kolumnen verleiht einer Region aus HTM Neuronen eine mächtige Plastizität, wie man sie auch im Gehirn beobachten kann. Eine HTM Region wird automatisch anpassen, was eine Kolumne repräsentiert (durch Änderungen der Synapsen des proximalen Dendriten), wenn sich die Eingabe ändert oder die Anzahl der Kolumnen erhöht oder verringert wird.

**Distale Dendriten**

Jede HTM-Zelle führt eine Liste von distalen dendritischen Segmenten. Jedes Segment agiert dabei als Schwellwert-Detektor. Wenn die Anzahl der aktiven Synapsen auf einem beliebigen Segment größer als der Schwellwert wird, dann wird das Segment aktiv, und die damit verbundene Zelle wechselt in den vorhersagenden Zustand. Der Vorhersagezustand einer Zelle ist also die Verknüpfung aller aktiven Segmente mit dem logischen Oder.

Ein dendritisches Segment speichert den Zustand der Region, indem es Verbindungen zu Zellen aufbaut, die zu einem Zeitpunkt gemeinsam aktiv waren. Das Segment sucht also nach Zuständen, der vorhersagt, dass die Zelle aktiv wird. Ein typischer Schwellwert für ein dendritisches Segment ist 15. Wenn 15 gültige Synapsen eines Segments gleichzeitig aktiv sind, dann wird der Dendrit aktiv. Es könnten hunderte oder tausende von Zellen in der Nachbarschaft aktiv sein, aber die Verbindung mit 15 davon ist ausreichend, um ein größeres Muster zu erkennen.

Jedes distale dendritische Segment hat auch eine Menge potentieller Synapsen. Die Menge der potentiellen Synapsen ist eine Teilmenge aller Zellen der Region. Beim Lernen werden die Permanenzwerte aller potentiellen Synapsen eines Segments erhöht oder verringert. Nur jene potentiellen Synapsen mit einem Permanenzwert oberhalb des Schwellwerts sind gültige Synapsen.

In einer Implementierung nutzen wir eine feste Anzahl dendritischer Segmente pro Zelle. In einer anderen Implementierung werden Segmente während des Trainings gelöscht und hinzugefügt. Beide Methoden können funktionieren. Wenn wir eine feste Anzahl dendritischer Segmente pro Zelle haben, dann kann man mehrere unterschiedliche Mengen an Synapsen auf demselben Segment speichern. Man nehme an, es befinden sich beispielsweise 20 gültige Synapsen auf einem Segment und der Schwellwert beträgt 15 (im Allgemeinen wählt man den Schwellwert kleiner als die Anzahl der Synapsen, um die Anfälligkeit gegenüber Rauschen zu verringern). Das Segment kann jetzt einen bestimmten Zustand der Zellen in der Umgebung erkennen. Was würde passieren, wenn wir 20 weitere Synapsen zu demselben Segment hinzufügen, welche einen ganz anderen Zustand der umgebenden Zellen repräsentieren? Dadurch können Fehler entstehen, denn das Segment könnte fälschlicherweise aktiv werden, da 8 Synapsen aus dem einen Muster und 7 Synapsen aus dem anderen Muster aktiv werden könnten. Wir haben experimentell beobachtet, dass man bis zu 20 Muster auf einem Segment speichern kann, bevor Fehler auftreten. Daher kann eine HTM Zelle mit einem Dutzend dendritischer Segmente Teil von vielen verschiedenen Vorhersagen sein.

**Synapsen**

Synapsen bei HTM-Zellen haben eine binäre Gewichtung. Skalare synaptische Gewichte werden durch das HTM Modell nicht ausgeschlossen, aber durch die Verwendung von spärlichen verteilten Mustern haben wir bisher keine Notwendigkeit für die Verwendung von skalaren Gewichten gesehen.

Allerdings haben Synapsen in HTM Zellen einen skalaren Wert, der als Permanenz bezeichnet wird und der beim Lernen angepasst wird. Der Permanenzwert 0.0 steht für eine potentielle Synapse, die nicht gültig ist und keinen Fortschritt in Richtung der Ausbildung einer gültigen Synapse gemacht hat. Ein Permanenzwert über einem Schwellwert (typischerweise 0.2) steht für eine Synapse, die sich gerade verbunden hat, aber leicht wieder getrennt werden könnte. Ein hoher Permanenzwert, zum Beispiel 0.9, steht für eine Synapse, die verbunden ist und nicht einfach wieder getrennt werden kann.

Die Anzahl der gültigen Synapsen auf den proximalen und distalen Dendriten ist nicht fest. Sie ändert sich, wenn die Zelle mit verschiedenen Mustern in Kontakt kommt. Beispielsweise ist die Anzahl gültiger Synapsen auf den distalen Dendriten abhängig von der temporalen Struktur der Daten. Wenn es keine bleibenden temporalen Muster in der Eingabe der Region gibt, dann haben alle Synapsen auf distalen Segmenten niedrige Permanenzwerte und nur sehr wenige Synapsen wären gültig. Gibt es viel temporale Struktur im Eingabestrom, dann finden wir viele gültige Synapsen mit hohen Permanenzwerten.

**Ausgabe von Zellen**
Eine HTM Zelle hat zwei verschiedene binäre Ausgaben: 1) die Zelle wird aktiv durch „feed-forward"-Eingaben (über den proximalen Dendriten) und 2) die Zelle ist aktiv durch laterale Verbindungen (über die distalen dendritischen Segmente). Ersteres wird „aktiver Zustand", letzteres „vorhersagender Zustand" genannt.

In dem zuvor abgebildeten Diagramm sind die beiden Ausgaben durch zwei Linien dargestellt, die den rechteckigen Zellkörper verlassen. Die linke Linie zeigt den aktiven Zustand durch „feed-forward"-Signale, während die rechte den vorhersagenden Zustand darstellt.

Nur der aktive Zustand durch „feed-forward"-Signale ist mit anderen Zellen der Region verbunden, um sicherzustellen, dass Vorhersagen immer auf der aktuellen Eingabe und ihrem Kontext beruhen. Es soll vermieden werden, dass Vorhersagen aufgrund anderer Vorhersagen getroffen werden. Würde dies ermöglicht, wären nach wenigen Iterationen alle Zellen der Region im vorhersagenden Zustand.

Die Ausgabe einer Region ist ein Vektor, der die Zustände aller Zellen repräsentiert. Dieser Vektor wird zur Eingabe der nächsten Region der Hierarchie, falls es eine gibt. Die Ausgabe ist die logische Oder-Verknüpfung des aktiven und des vorhersagenden Zustands. Durch das Kombinieren des aktiven und des vorhersagenden Zustands wird die Ausgabe der Region stabiler (sie ändert sich langsamer) als die Eingabe. Solch eine Stabilität ist eine wichtige Eigenschaft der Inferenz in einer Region.

## Empfohlene Literatur

Wir werden oft gebeten, Literatur zur Vertiefung des Wissens über Neurowissenschaften zu empfehlen. Das Gebiet der Neurowissenschaften ist so umfangreich, dass eine allgemeine Einführung es erfordert, viele unterschiedliche Quellen heranzuziehen. Neue Ergebnisse werden in wissenschaftlichen Zeitschriften veröffentlicht, die sowohl schwer verständlich als auch schwer erhältlich sind, wenn man keiner Universität angehört.

Hier sind zwei erhältliche Bücher, die bezüglich der in diesem Anhang besprochenen Themen relevant sind.

Stuart, Greg, Spruston, Nelson, Hauser, Michael, *Dendrites, second edition* (New York: Oxford University Press, 2008)

Dieses Buch ist eine gute Quelle für alle Themen rund um Dendriten. Kapitel 16 bespricht die nicht-linearen Eigenschaften von dendritischen Segmenten, wie sie in den HTM kortikalen Lernalgorithmen verwendet werden. Es wurde von Bartlett Mel verfasst, der viel in diesem Gebiet gearbeitet hat.

Mountcastle, Vernon B. Perceptual Neuroscience: *The Cerebral Cortex* (Cambridge, Mass.: Harvard University Press, 1998)

Dieses Buch ist eine gute Einführung in alle Themen rund um den Neokortex. Mehrere Kapitel besprechen die Zelltypen und ihre Verbindungen. Man kann einen klaren Überblick über kortikale Neuronen und ihre Verbindungen erhalten, allerdings ist es etwas älter und enthält daher nicht die aktuellen Fortschritte über Eigenschaften von Dendriten.

# Appendix B: Vergleich der Schichten im Neokortex und einer HTM-Region

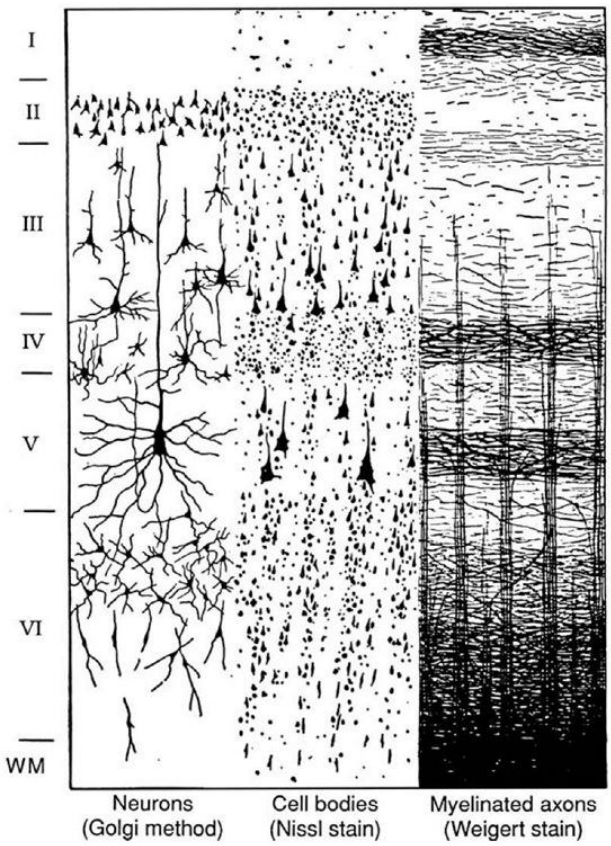
Dieser Anhang beschreibt die Beziehung zwischen einer HTM-Region und einer Region in einem biologischen Neokortex.

Im speziellen behandelt dieser Anhang, wie sich der HTM-basierte kortikale Lernalgorithmus mit seinen Kolumnen und Zellen auf die schichtbasierte und kolumnare Architektur des Neokortex abbilden lässt. Viele sind verwirrt von dem Konzept der „Schicht“ im Neokortex, und welcher Zusammenhang zu einer Schicht in einem HTM besteht. Wir hoffen, dass dieser Anhang Klarheit und ein besseres Verständnis der biologischen Grundlagen der HTM-basierten kortikalen Lernalgorithmen bringen wird.

## Schaltkreise im Neokortex

Der menschliche Neokortex ist ein neuronales Gewebe etwa der Fläche von 1000 $cm^2$ und einer Dicke von 2mm. Um diese Fläche zu visualisieren, kann man sich eine Stoffserviette vorstellen, eine vernünftige Annäherung an die Fläche und Dicke des Neokortex. Der Neokortex ist in Dutzende funktionaler Regionen unterteilt, einige davon betreffen das Sehen, andere das Hören, andere Sprachverarbeitung usw. Unter einem Mikroskop sehen sich die verschiedenen Regionen erstaunlich ähnlich.

Mehrere Organisationsprinzipien können in allen Regionen des Neokortex beobachtet werden.




**Schichten**

Im Allgemeinen unterteilt man den Neokortex in sechs Schichten. Fünf der Schichten enthalten Zellen, und eine hauptsächlich Verbindungen. Die Schichten wurden vor über hundert Jahren entdeckt, als die ersten Färbungstechniken aufkamen. Das obige Bild (von Cajal) zeigt einen kleinen Schnitt aus dem Neokortex, welcher mit drei verschiedenen Färbungen behandelt wurde. Die vertikale Achse entspricht der Dicke des Neokortex, ungefähr 2mm. Die linke Seite des Bildes beschreibt die sechs Schichten. Schicht 1, ganz oben, ist die Schicht ohne Zellen. Die Bezeichnung „WM“ am unteren Ende kennzeichnet den Beginn der „white matter“, der weißen Gehirnsubstanz, der Bereich, in dem Axone von Zellen zu anderen Zeilen des Neokortex und zu anderen Teilen des Gehirns verlaufen.

Der rechte Teil des Bilds zeigt eine Färbung, die nur myelinisierte Nervenfasern sichtbar werden lässt (Myelin ist eine fettige Isolation, die einige, aber nicht alle Nervenfasern umhüllt). In diesem Teil des Bildes kann man zwei der Hauptorganisationsprinzipien des Neokortex erkennen, Schichten und Kolumnen. Die meisten Axone teilen sich direkt nach Austritt aus dem Zellkörper in zwei Äste. Ein Ast verläuft hauptsächlich horizontal und der andere Ast hauptsächlich vertikal. Der horizontale Ast knüpft eine große Zahl Verbindungen mit nahegelegenen Zellen derselben oder einer benachbarten Schicht, daher werden die Schichten in Färbungen wie dieser sichtbar. Man mache sich klar, dass es sich hier um einen

Schnitt des Neokortex handelt. Die meisten Axone verlaufen in die Bildebene hinein oder aus hier heraus, Axone sind also länger, als sie auf diesem Bild erscheinen. Schätzungen gehen davon aus, dass in jedem Kubikmillimeter des Neokortex zwischen 2 und 4 Kilometer Axone und Dendriten verlaufen.

Im mittleren Teil des Bildes findet sich eine Färbung, die nur Zellkörper einfärbt, aber keine Axone oder Dendriten zeigt. Man erkennt, dass die Größe und Häufigkeit der Zellen ebenfalls je nach Schicht variiert. In diesem Bild gibt es nur wenig Hinweise auf Kolumnen. Man kann sehen, dass es ein paar Neuronen in der Schicht 1 gibt. Die Anzahl der Neuronen in Schicht 1 ist aber so klein, dass man sie immer noch als Schicht ohne Zellen einordnet. Neurowissenschaftler schätzen, dass sich ca. 100000 Neuronen in jedem Kubikmillimeter des Neokortex befinden.

Der linke Teil des Bildes zeigt eine Färbung, die die Zellkörper, Axone und Dendriten einiger weniger Neuronen darstellt. Hier zeigt sich, dass die Größe der dendritischen „Bäume" sich deutlich je nach Schicht unterscheidet. Ebenfalls sichtbar sind einige „apikale Dendriten", die von einem Zellkörper ausgehend Verbindungen mit anderen Schichten herstellen. Das Vorhandensein und die Zielrichtung der apikalen Dendriten sind in jeder Schicht unterschiedlich.

Zusammenfassend zeigt sich die Schicht- und Kolumnenorganisation, wenn man neuronales Gewebe einfärbt und mit einem Mikroskop untersucht.

**Variationen der Schichten in verschiedenen Regionen**

Es gibt einige Unterschiede bezüglich der Dicke der Schichten in unterschiedlichen Regionen des Neokortex und einige Meinungsverschiedenheiten bezüglich der Anzahl der Schichten. Die Variationen sind abhängig davon, welches Tier man untersucht, welche Region man betrachtet und wer die Untersuchung durchführt. Beispielsweise können in dem oben gezeigten Bild die Schichten 2 und 3 einfach auseinandergehalten werden, aber dies ist im Allgemeinen nicht der Fall. Einige Wissenschaftler berichten, dass sie die Schichten in den von ihnen untersuchten Regionen nicht unterscheiden können, daher werden die Schicht 2 und 3 oft zusammengefasst und als „Schicht 2/3" bezeichnet. Andere Wissenschaftler beobachten Gegenteiliges und definieren Unterteilungen wie 3A und 3B.

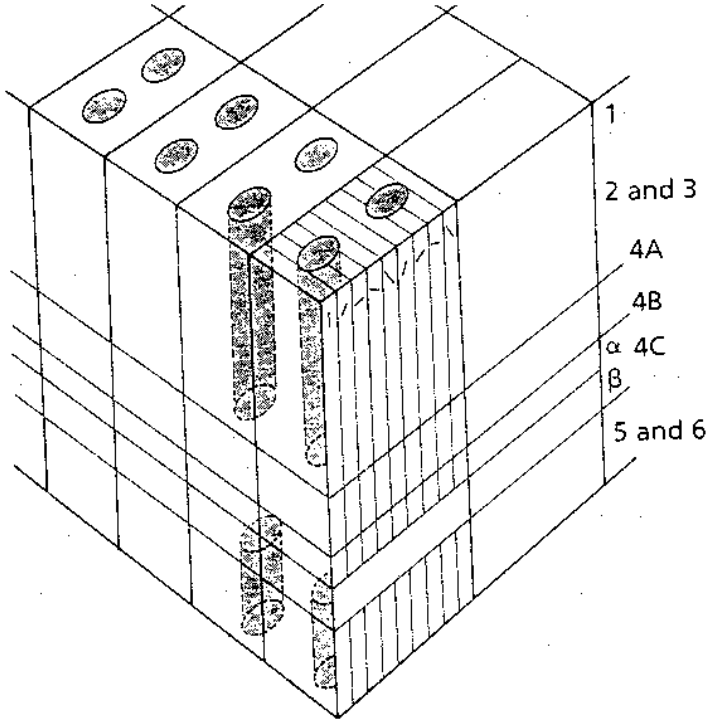
Die Schicht 4 ist am deutlichsten ausprägt in solchen Regionen, die am nächsten an sinnesverarbeitenden Organen gelegen sind. Bei einigen Tieren (zum Beispiel bei Affen und beim Menschen) ist die Schicht 4 der ersten Region der Visuellen Verarbeitung deutlich unterteilt. Bei anderen Tieren findet sich diese Unterteilung nicht. Die Schicht 4 verschwindet meist in Regionen, die hierarchisch weit von den Sinnesorganen gelegen sind.

**Kolumnen**

Das zweite Hauptorganisationsprinzip des Neokortex sind Kolumnen. Zum Teil kann die kolumnare Struktur in gefärbten Schnitten erkannt werden, aber die meisten

Hinweise findet man aufgrund des Antwortverhaltens der Zellen auf verschiedene Eingaben.

Wenn Wissenschaftler Sonden verwenden, um zu untersuchen, welche Neuronen aktiviert werden, dann stellen sie fest, dass Zellen auf einer gemeinsamen vertikalen Linie über mehrere Schichten grob auf dieselbe Eingabe reagieren.




Diese Zeichnung illustriert einige Antworteigenschaften der Zellen in V1, der ersten kortikalen Region, die Information von der Retina verarbeitet.

Eine der ersten Entdeckungen bestand darin, dass die meisten Zellen in V1 auf Linien oder Kanten verschiedener Ausrichtungen in verschiedenen Bereichen der Retina reagieren. Zellen, die kolumnar angeordnet sind, sich also vertikal auf einer Linie befinden, reagieren alle auf Kanten gleicher Ausrichtung. Wenn man genau hinsieht, erkennt man einige kleine Linien verschiedener Ausrichtung, die in einer Reihe am oberen Ende der Abschnitte angebracht sind. Diese Linien zeigen an, auf welche Ausrichtung die dort gelegenen Zellen reagieren. Zellen die innerhalb des dünnen vertikalen Bandes liegen, reagieren auf Linien mit derselben Ausrichtung.

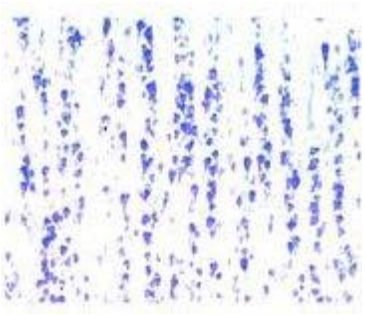
Es gibt mehrere weitere kolumnare Eigenschaften, die man in V1 beobachten kann, zwei davon sind in der Skizze dargestellt. Es gibt Kolumnen mit „okularer Dominanz“, die auf ähnliche Kombinationen aus rechtem und linkem Auge reagieren. Und es gibt „Kleckse“, in denen Zellen hauptsächlich auf Farbe reagieren.

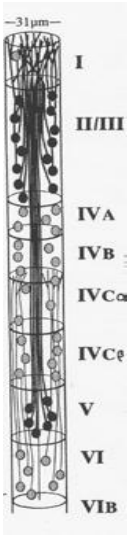
Die Kolumnen mit okularer Dominanz sind als größere Blöcke im Diagramm dargestellt. Die „Kleckse“ sind als dunkle Ovale dargestellt.

Die allgemeine Regel im Neokortex ist es, dass verschiedene Antwortverhalten miteinander überlagert werden, zum Beispiel Ausrichtung und okulare Dominanz. Bewegt man sich horizontal über die kortikale Oberfläche, dann verändert sich die Kombination des Antwortverhaltens. Dennoch haben Neuronen, die vertikal auf einer Linie stehen, dasselbe Antwortverhalten. Dies gilt für Region der auditiven, visuellen und somatosensorischen Verarbeitung. Es besteht eine Debatte unter Eurowissenschaftlern, ob dies überall im Neokortex der Fall ist, es scheint in fast allen, wenn nicht allen der Fall zu sein.

**Mini-Kolumnen**

Die kleinste kolumnare Struktur im Neokortex ist die sogenannte Mini-Kolumne. Mini-Kolumnen haben einen Durchmesser von ungefähr 30um und enthalten etwa 80-100 Neuronen über alle 5 Schichten verteilt. Der gesamte Neokortex ist aus Mini-Kolumnen zusammengesetzt. Man kann sie vorstellen wie kleine Spaghettistücke vorstellen, die dicht gepackt aneinander liegen. Es gibt kleine Lücken mit wenigen Zellen zwischen den Mini-Kolumnen, die sie manchmal in gefärbten Schnitten sichtbar werden lassen.






Auf der linken Seite ist ein gefärbtes Bild, welches Zellkörper aus einem Schnitt des Neokortex. Die vertikale Struktur der Mini-Kolumnen ist in diesem Bild klar erkennbar. Auf der rechten Seite sieht man eine konzeptuelle Zeichnung einer Mini-Kolumne (von Peters und Yilmez). Echte Kolumnen sind dünner als diese Darstellung. Man bemerke dass es mehrere Neuronen in jeder Schicht der Kolumne gibt. Alle Neuronen in einer Mini-Kolumne reagieren auf dieselbe Eingabe.

In der zuvor gezeigten Darstellung des Ausschnitts aus V1 beispielsweise wird eine Mini-Kolumne Zellen enthalten, die auf Linien einer bestimmten Ausrichtung und

einer bestimmten okularen Dominanz reagieren. Die Zellen einer benachbarten Mini-Kolumne reagieren auf eine leicht unterschiedliche Ausrichtung oder eine unterschiedliche okulare Dominanz.

Hemmende Neuronen spielen eine wesentliche Rolle zur Definition von Mini-Kolumnen. Sie sind in dem Bild oder der Zeichnung nicht sichtbar, aber die Axone hemmender Neuronen verlaufen in geraden Linien zwischen Mini-Kolumnen, was zum Teil zur physischen Trennung beiträgt. Man nimmt an, dass die hemmenden Neuronen auch dabei helfen, alle Zellen einer Mini-Kolumne dazu zu zwingen, auf gleiche Eingaben zu reagieren.

Die Mini-Kolumne ist der Prototyp einer in den HTM-basierten kortikalen Lernalgorithmen verwendeten Kolumne.

**Eine Ausnahme bei kolumnarem Antwortverhalten**

Es gibt eine Ausnahme bezüglich des kolumnaren Antwortverhaltens, die für die basierten kortikalen Lernalgorithmen relevant ist. Üblicherweise finden Wissenschaftler heraus, worauf eine Zelle reagiert, in dem man ein Tier in einem Experiment einem einfachen Stimulus aussetzt. Beispielsweise könnte man einem Tier eine einzelne Linie in einem kleinen Abschnitt des visuellen Bereichs zeigen, um das Antwortverhalten der Zellen in V1 zu bestimmen. Wen man einfache Eingaben verwendet, dann stellt man fest, dass Zellen immer auf dieselbe Art auf dieselbe Eingabe reagieren. Wenn die einfache Eingabe allerdings in ein Video mit einer natürlichen Szene eingebettet wird, dann werden die Zellen wählerischer. Eine Zelle, die verlässlich auf eine einzelne vertikale Linie reagiert, wird nicht immer reagieren, wenn die vertikale Linie in eine komplexe bewegte natürliche Szene eingebettet wird.

In den HTM-basierten kortikalen Lernalgorithmen haben alle Zellen dasselbe Antwortverhalten bezüglich „feed-forward“-Eingaben, aber in einer erlernten temporalen Sequenz wird nur eine der Zellen einer HTM-Kolumne aktiv. Dieser Mechanismus dient zur Repräsentation von Sequenzen variabler Ordnung und ist eine Analogie zu der gerade beschriebenen Eigenschaft von Neuronen. Eine einfache Eingabe ohne Kontext wird alle Zellen einer Kolumne aktivieren. Die gleiche Eingabe als Teil einer erlernten Sequenz führt nur zur Aktivierung einer Zelle.

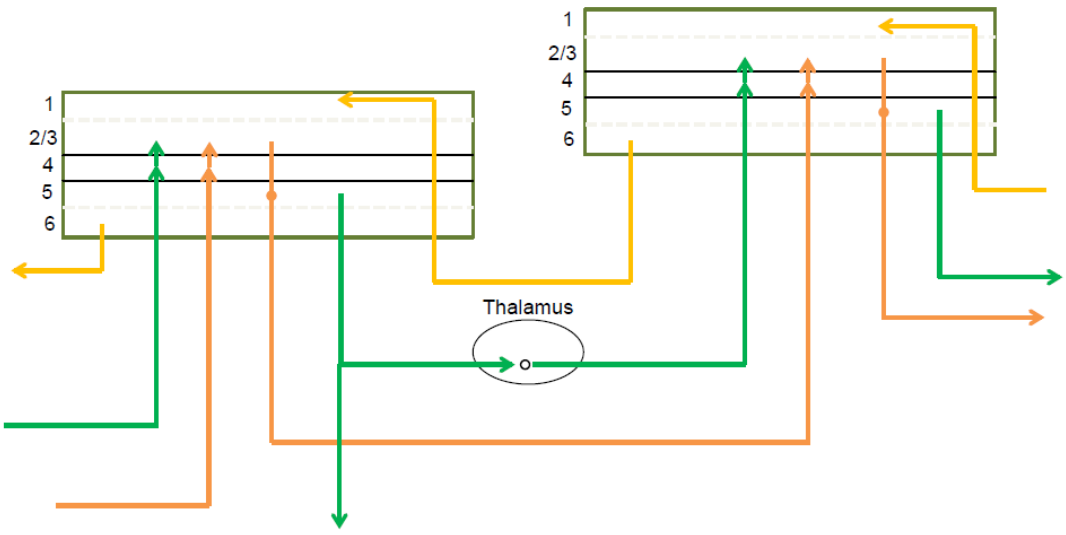
Wir wollen nicht behaupten, dass nur ein Neuron in einer Mini-Kolumne gleichzeitig aktiv wird. Die HTM-basierten kortikalen Lernalgorithmen weisen darauf hin, dass innerhalb einer Kolumne bei unvorhergesehenen Eingaben alle Neuronen einer Schicht aktiviert werden würden, und bei vorhersehbaren Eingaben eine Teilmenge der Neuronen.

## Warum gibt es Schichten und Kolumnen?

Niemand weiß sicher, warum die Schichten oder Kolumnen im Neokortex existieren. Die HTM-Theorie schlägt allerdings eine Antwort vor. Die HTM-basierten kortikalen Lernalgorithmen zeigen, dass eine Schicht von Zellen, die in Kolumnen organisiert ist, als hoch-kapazitiver Speicher von Zustandsübergängen variabler Ordnung fungieren kann. Einfacher ausgedrückt, eine Schicht von Zellen kann eine Menge Sequenzen erlernen. Kolumnen solcher Zellen, die auf „feed-forward"-Eingabe auf die gleiche Weise reagieren, sind der Schlüsselmechanismus für das Erlernen der Zustandsübergänge variabler Ordnung.

Diese Hypothese erklärt, warum Kolumnen nötig sind, aber was ist mit den fünf Schichten? Wenn eine einzelne kortikale Schicht Sequenzen erlernen und Vorhersagen treffen kann, warum finden wir fünf Schichten im Neokortex?

Wir schlagen vor, dass die verschiedenen Schichten, die man im Neokortex beobachten kann, alle jeweils Sequenzen mit denselben grundlegenden Mechanismen erlernen, aber dass die Sequenzen jeweils unterschiedlich verwendet werden. Es gibt noch viele Unklarheiten, aber wir können die grundsätzliche Idee erläutern. Zuvor kann es hilfreich sein, zu beschreiben, wie die Neuronen unterschiedlicher Schichten miteinander verbunden sind.




Das obige Diagramm zeigt zwei Regionen im Neokortex und die Hauptverbindungen dieser Regionen untereinander. Solche Verbindungen kann man überall im Neokortex finden, wo zwei Regionen aufeinander projizieren. Die Box auf der linken Seite repräsentiert eine kortikale Region, die hierarchisch gesehen niedriger angeordnet ist als die Region (Box) auf der rechten Seite, d.h. „feed-forward"-Information verläuft in dem Diagramm von links nach rechts. Der abwärts gerichtete Pfeil projiziert auf andere Regionen im Gehirn. Rückgekoppelte

Information fließt von rechts nach links. Jede Region ist in Schichten unterteilt, Schicht 2 und Schicht3 sind gemeinsam als Schicht 2/3 dargestellt.

Die farbigen Linien repräsentieren die Ausgaben der Neuronen der verschiedenen Schichten. Diese sind Axon-Bündel, die aus den Neuronen der jeweiligen Schicht entspringen. Zur Erinnerung: Diese Axone teilen sich unmittelbar in zwei Teile. Ein Teil verläuft horizontal innerhalb der Region, hauptsächlich in derselben Schicht. Also sind alle Zellen in jeder Schicht miteinander hoch vernetzt. Die Neuronen und horizontalen Verbindungen sind nicht dargestellt.

Es gibt zwei „feed-forward"-Leitungspfade, ein direkter Pfad in orange dargestellt und ein indirekter Pfad in grün. Die Schicht 4 ist die Haupteingabeschicht für „feed-forward"-Verbindungen und erhält ihre Eingabe aus beiden Pfaden. Schicht 4 projiziert auf Schicht 3.

Schicht 3 ist der auch der Ursprung des direkten „feed-forward"-Pfades. Das bedeutet, der direkte „feed-forward"-Pfad beschränkt sich auf die Ebenen 3 und 4.

Einige „feed-forward"-Verbindungen überspringen Schicht 4 und verlaufen direkt zu Schicht 3. Und, wie zuvor bemerkt, entfällt die Schicht 4 in Regionen, die weit von sensorischer Eingabe entfernt liegen. Dort verläuft der „feed-forward"-Pfad einfach von Schicht 3 zu Schicht 3 in der nächsten Region.

Der zweite „feed-forward"-Pfad (in Grün) entspringt der Schicht 5. Zellen der Schicht 3 verbinden sich mit der Schicht 5, und verlaufen daran vorbei weiter zur nächsten Region. Nach dem Austritt aus dem kortikalen Gewebe teilen sich die Axone aus der Schicht 5 erneut. Ein Ast projiziert zu sub-kortikalen Bereichen, die mit der motorischen Steuerung befasst sind. Man nimmt an, dass diese Axone motorische Befehle sind (dargestellt als abwärts gerichtete Pfeile). Der andere Ast projiziert zu einem Teil des Gehirns genannt Thalamus, der als Weiche agiert. Der Thalamus gibt Information entweder zur nächsten Region weiter oder blockiert sie.

Schließlich beginnt der primäre Rückkopplungspfad, in Gelb dargestellt, in Schicht 6 und projiziert auf die Schicht 1. Zellen in den Schichte 2,3 und 5 verbinden sich mit ihren apikalen Dendriten mit der Schicht 1 (nicht eingezeichnet). Schicht 6 erhält Eingabe von Schicht 5.

Diese Beschreibung ist eine beschränkte Zusammenfassung des Wissens über die Schicht-zu-Schicht-Verbindungen. Aber es ist ausreichend, um unsere Hypothese zu verstehen, warum es mehrere Schichten gibt, wenn doch alle Schichten Sequenzen erlernen.

## Hypothese über die Aufgaben der verschiedenen Schichten

Wir schlagen vor, dass die Schichten 3,4 und 5 alle „feed-forward“-Schichten sind und alle Sequenzen erlernen. Schicht 4 erlernt Sequenzen erster Ordnung. Schicht 3 erlernt Sequenzen variabler Ordnung. Und Schicht 5 erlernt Sequenzen variabler Ordnung mit Zeitaspekten. Wir betrachten die Schichten in größerer Genauigkeit.

*Schicht 4*
Es ist einfach, Sequenzen erster Ordnung mit dem HTM-basierten kortikalen Lernalgorithmus zu erlernen. Wenn wir keine Hemmung der Zellen einer Kolumne untereinander erzwingen, d.h. die Zellen die Eingabe nicht nach Kontext vorheriger Eingaben unterscheiden, dann tritt Lernen von Sequenzen erster Ordnung auf. Im Neokortex könnte dies wahrscheinlich erreicht werden, wenn man den hemmenden Effekt zwischen Zellen derselben Kolumne entfernt. In unseren Computermodellen der HTM-basierten kortikalen Lernalgorithmen verwenden wir einfach eine Zelle pro Kolumne, welches dasselbe Ergebnis bewirkt.

Sequenzen erster Ordnung werden benötigt, um invariante Repräsentation für räumliche Transformationen von Eingaben zu bilden. Bei der visuellen Wahrnehmung sind Translation, Skalierung und Rotation räumliche Transformationen. Wenn eine HTM-Region mit Speicher erster Ordnung mit bewegten Bildern trainiert wird, dann erlernt sie, dass unterschiedliche räumliche Muster äquivalent sind. Der daraus entstehenden HTM-Zellen verhalten sich wie sogenannte „komplexe Zellen“ im Neokortex. Die HTM-Zellen bleiben für eine ganze Bandbreite räumlicher Transformationen aktiv (und in vorhersagendem Zustand).

Bei Numenta haben wir visuelle Experimente durchgeführt, die bestätigen, dass diese Mechanismen wie erwartet arbeiten, und dass einige räumliche Invarianz innerhalb jeder Ebene erreicht werden kann. Die Details dieser Experimente sprengen den Rahmen dieses Anhangs.

Das Erlernen von Sequenzen erster Ordnung in Schicht 4 steht mit dem Fund der komplexen Zellen in Schicht 4 im Einklang, und es erklärt, warum die Schicht 4 in höheren Regionen des Neokortex verschwindet. Während man die Hierarchie nach oben aufsteigt, ist es ab einem bestimmten Punkt nicht mehr möglich, räumliche Invarianten zu erlernen, da die Repräsentationen bereits invariant sind.

*Schicht 3*
Die Schicht 3 ist am ähnlichsten zu dem HTM-basierten kortikalen Lernalgorithmus, den wir in Kapitel 2 vorgestellt haben. Sie erlernt Sequenzen variabler Ordnung und trifft Vorhersagen, die stabiler sind als ihre Eingabe. Schicht 3 projiziert immer in die nächste Region der Hierarchie und führt daher zu erhöhter temporaler Stabilität innerhalb der Hierarchie. Sequenzspeicher variabler Ordnung führt zu Neuronen, die als richtungsorientiere „komplexe Zellen“ aufgefasst werden können. Sie

unterscheiden nach temporalen Kontext, z.B. ob eine Linie nach rechts oder links wandert.

*Schicht 5*
Die letzte Schicht für „feed-forward"-Signale ist die Schicht 5. Wir schlagen vor, dass die Schicht 5 ähnlich der Schicht 3 ist, mit drei Unterschieden. Der erste Unterschied besteht darin, dass die Schicht 5 das Konzept der Zeitberechnung. Schicht 3 trifft eine Vorhersage „Was" als Nächstes passieren wird, aber sie macht keine Aussage über das „Wann". Allerdings erfordern viele Aufgaben Zeitberechnung, zum Beispiel das Erkennen gesprochener Wörter, ein Prozess, bei dem die relativen Zeitunterschiede zwischen Geräuschen eine Rolle spielen. Motorisches Verhalten ist ein weiteres Beispiel; zeitliche Koordination von Muskelaktivität ist wesentlich. Wir nehmen an, dass die Neuronen der Schicht 5 eine Vorhersage zu einem bestimmten Zeitpunkt treffen. Es gibt mehrere biologische Hinweise, die diese Hypothese stützen. Ein Hinweis darauf ist, dass die Schicht 5 der Ursprung motorischer Signale im Neokortex ist. Ein weiterer Hinweis besteht darin, dass die Schicht 5 Eingaben aus Schicht 1 erhält, die aus einem Teil des Thalamus stammen (im Diagramm nicht dargestellt). Wir nehmen an, dass diese Information eine Codierung der Zeit darstellt und über die Thalamus-Eingaben der Schicht 1 an viele Zellen verteilt werden (im Diagramm nicht dargestellt).

Der zweite Unterschied zwischen Schicht 3 und Schicht 5 besteht darin, dass die Schicht 3 Vorhersagen soweit wie möglich in die Zukunft trifft und damit temporale Stabilität erreicht. Der HTM-basierte kortikale Lernalgorithmus, der in Kapitel 2 beschrieben wurde, verhält sich auf diese Weise. Im Gegensatz dazu soll die Schicht 5 nur das nächste Muster (zu einem bestimmten Zeitpunkt) vorhersagen. Wir haben diesen Unterschied nicht modelliert, aber er würde ganz natürlich auftreten, wenn Musterübergänge zusammen mit einer Zeitangabe gespeichert werden würden.

Der dritte Unterschied zwischen Schicht 3 und Schicht 5 kann im Diagramm betrachtet werden. Die Ausgabe der Schicht 5 projiziert immer an sub-kortikale motorische Zentren, und die „feed-forward"-Signale werden durch den Thalamus gesteuert. Die Ausgabe der Schicht 5 wird also manchmal zur nächsten Region weitergeleitet und manchmal blockiert. Wir (und andere) nehmen an, dass diese Steuerung in Zusammenhang mit verdeckter Aufmerksamkeit steht (verdeckte Aufmerksamkeit bedeutet, die Aufmerksamkeit auf eine Wahrnehmung zu lenken, ohne motorisches Verhalten zu produzieren).

Zusammenfassend kombiniert die Schicht 5 genaue Zeitberechnung, Aufmerksamkeit und motorisches Verhalten. Es gibt noch viele Geheimnisse über das Zusammenspiel dieser Mechanismen. Der Punkt, der uns wichtig ist, besteht darin, dass eine Variation der HTM-basierten kortikalen Lernalgorithmen auf einfache Weise spezifische Zeitberechnung beinhalten könnte und daher die Berechtigung für eine eigene Schicht im Neokortex rechtfertigt.

*Schicht 2 und Schicht 6*
Schicht 6 ist der Ursprung der Axone, die Information in tiefere Regionen zurücksenden. Über die Schicht 2 ist deutlich weniger bekannt. Wie zuvor erwähnt, ist sogar ihre Existenz als Abgrenzung von Schicht 3 manchmal umstritten. Wir haben zum gegenwärtigen Zeitpunkt keine weiteren Anmerkungen über diese Fragen, außer festzustellen, dass Schicht 2 und Schicht 6 wie alle anderen Schichten massive horizontale Vernetzung und kolumnares Antwortverhalten zeigen, daher nehmen wir an, dass sie ebenfalls eine Variante des HTM-basierten kortikalen Lernalgorithmus ausführen.

**Was entspricht einer HTM-Region im Neokortex?**
Wir haben die HTM-basierten kortikalen Lernalgorithmen auf zwei Arten implementiert, einmal mit mehreren Zellen pro Kolumne für Sequenzen variabler Ordnung und einmal mit einer Zelle pro Kolumne für Speicher erster Ordnung. Wir glauben, diese beiden Varianten entsprechen der Schicht 3 und Schicht 4 im Neokortex.

Obwohl die HTM-basierten kortikalen Lernalgorithmen (mit mehreren Zellen pro Kolumne) am ehesten der Schicht 3 ähneln, zeigen unsere Modelle eine Flexibilität, die das Gehirn nicht hat. Beispielsweise kennen wir die Reihenfolge, in der Synapsen in dendritischen Segmenten gebildet werden. Wir können diese Information nutzen, umherauszufinden, welche Muster für den unmittelbar nächsten Zeitpunkt vorhergesagt werden und welche Muster allgemeinere Vorhersagen zukünftiger Muster darstellen. Spezifische Zeitangaben können vermutlich auf dieselbe Weise hinzugefügt werden. Daher sollte es möglich sein, eine einschichtige HTM-Region zu erzeugen, die die Funktionalität von Schicht 3 und Schicht 5 zusammenfasst.

## Zusammenfassung

Die HTM-basierten kortikalen Lernalgorithmen verkörpern nach unserer Annahme die elementaren Bausteine der neuronalen Organisation im Neokortex. Sie zeigen, wie eine Schicht horizontal vernetzter Zellen Sequenzen spärlicher verteilter Repräsentation erlernt. Variationen der HTM-basierten kortikalen Lernalgorithmen werden in den verschiedenen Schichten des Neokortex verwendet, um ähnliche, aber unterschiedliche Aufgabe zu erfüllen.

Wir nehmen an, dass „feed-forward“-Eingaben einer Region im Neokortex, unabhängig ob zu Schicht 3 oder Schicht 4, hauptsächlich auf proximale Dendriten projiziert werden, welche mit Hilfe von hemmenden Zellen eine spärliche verteilte Repräsentation der Eingabe erzeugen. Wir nehmen weiter an, dass die Zellen der Schichten 2,3,4,5 und 6 diese spärliche verteilte Repräsentation mitverwenden. Dies wird erreicht, indem alle Zellen einer schichtenübergreifenden Kolumne dazu gezwungen werden, auf gleiche Eingabe zu reagieren.

Wir nehmen an, dass Zellen aus Schicht 4, sofern vorhanden, den HTM-basierten kortikalen Lernalgorithmus nutzen, um temporale Musterübergänge erster Ordnung zu erlernen, welche zu Repräsentationen führen, die invariant gegenüber räumlicher Transformationen sind. Schicht 3 nutzt den HTM-basierten kortikalen Lernalgorithmus um temporale Musterübergänge variabler Ordnung zu erlernen und stabile Repräsentationen zu bilden, die in der kortikalen Hierarchie nach oben gereicht werden. Zellen der Schicht 5 erlernen Musterübergänge variabler Ordnung mit Zeitaspekten. Wir haben noch keine spezifischen Annahmen über Schicht 2 und Schicht 6. Allerdings deutet die typische horizontale Vernetzung darauf hin, dass auch diese Schichten eine Art Sequenzspeicher darstellen.

## Glossar

Hinweis: Die hier beschriebenen Definitionen beschreiben, wie die Begriffe in diesem Dokument verwendet werden und könnten in der allgemeinen Verwendung andere Bedeutungen tragen.

| | |
|---|---|
| Aktiver Zustand | der Zustand einer Zelle, die durch „feed-forward"-Eingabe aktiviert wurde |
| dendritisches Segment, Dendritensegment | eine zusammengefasste Einheit von Synapsen, die mit Zellen und Kolumnen in Verbindung stehen<br><br>*HTMs haben zwei verschiedene Arten von dendritischen Segmenten. Eines steht im Zusammenhang mit den lateralen Verbindungen einer Zelle. Wenn die Anzahl der aktiven Synapsen einen Schwellwert überschreitet, dann geht die verbundene Zelle in den vorhersagenden Zustand über. Die andere Art gehört zu den „feed-forward"-Verbindungen einer Kolumne. Die Anzahl der Synapsen wird aufsummiert, um eine „feed-forward"-Aktivierung zu erzeugen.* |
| Ebene | eine HTM-Region im Kontext einer Hierarchie |
| „feed-forward" | Signalrichtung weg von der Eingabeebene, von einer niedrigeren Ebene der Hierarchie zu einer höheren Ebene der Hierarchie. |
| Feedback, Rückkopplung | Signalrichtung hin zur Eingabeebene, von einer höheren Ebene der Hierarchie zu einer niedrigeren Ebene der Hierarchie. |
| gewünschte Dichte | gewünschter Prozentsatz an Kolumnen, die durch "feed-forward"-Eingaben innerhalb einer Region aktiv werden<br><br>*Der Prozentsatz gilt nur innerhalb eines Radius, der je nach Streuung der „feed-forward"-Eingaben variiert. „Gewünscht" daher, da der Prozentsatz je nach Eingabe variiert.* |
| Hemmungsradius | definiert die Fläche um eine Kolumne, in der diese aktiv andere Kolumnen hemmt |

| | |
|---|---|
| Hierarchical Temporal Memory (HTM) | eine Technologie, die einige strukturelle und algorithmische Funktionen des Neokortex nachbildet |
| Hierarchie | ein Netzwerk verbundener Elemente, in der die Verbindungen zwischen Elementen eindeutig als „feed-forward" oder Feedback bezeichnet werden |
| HTM-basierte kortikale Lernalgorithmen | Funktionalität für räumliche Musterzusammenfassung, temporale Musterzusammenfassung und Lernen/Vergessen, die eine HTM-Region bilden |
| HTM-Netzwerk | eine Hierarchie von HTM-Regionen |
| HTM-Region | eine elementare Einheit zur Speicherung und Vorhersage<br><br>*Eine HTM-Region ist aufgebaut au seiner Schicht hoch vernetzter Zellen, die in Kolumnen angeordnet sind. Eine heutige HTM-Region hat eine einzelne Schicht aus Zellen, während eine Region im Neokortex (und später auch im HTM) aus mehreren Zellschichten besteht. Wenn man eine Region im Kontext einer Hierarchie betrachtet, kann eine Region auch als Ebene bezeichnet werden.* |
| Inferenz | das Erkennen, dass ein räumliches und zeitliches Muster einem zuvor gespeicherten Muster ähnelt |
| koinzidente Aktivität | zwei oder mehrere Zellen sind gleichzeitig aktiv |
| Kolumne | eine Gruppe bestehend aus einer oder mehreren Zellen, die in einer HTM-Region als Einheit arbeiten<br><br>*Zellen in einer Kolumne repräsentieren dieselbe Eingaben, aber in unterschiedlichem Kontext* |
| Laterale Verbindungen | Verbindungen zwischen Zellen derselben Region |
| Neuron | eine informationsverarbeitende Zelle im Gehirn<br><br>*In diesem Dokument wird das Wort Neuron genutzt, um speziell biologische Zellen zu bezeichnen, und das Wort „Zelle", wenn es um eine informationsverarbeitende Einheit in einem HTM geht* |

| | |
|---|---|
| Permanenz | Ein skalarer Wert, der den Zustand einer potentiellen Synapse anzeigt<br><br>*Ein Permanenzwert unterhalb eines Schwellwerts zeigt, dass keine Synapse an einer möglichen Stelle ausbildet wurde. Ein Permanenzwert über dem Schwellwert bedeutet, dass die Synapse gültig ist. Lernen in einer HTM-Region wird erreicht, indem die Permanenzwerte potentieller Synapsen verändert werden.* |
| Potentielle Synapse | eine Teilmenge aller Zellen, die möglicherweise eine Synapse mit einem bestimmten dendritischen Segment bilden könnten<br><br>*Nur eine Teilmenge der potentiellen Synapsen wird zu einem bestimmten Zeitpunkt gültig, abhängig von ihren Permanenzwerten.* |
| Räumliche Muster-zusammenfassung | der Prozess der Bildung einer spärlichen verteilten Repräsentation der Eingabe<br><br>*Eine der Eigenschaften der räumlichen Musterzusammen-fassung ist es, dass ähnliche Muster auf dieselbe spärliche verteilte Repräsentation abbilden* |
| Rezeptives Feld | eine Menge an Eingaben, mit der eine Kolumne oder Zelle verbunden ist<br><br>*Wenn die Eingaben einer HTM-Region als 2D-Array von Bits organisiert sind, dann das rezeptive Feld als Radius innerhalb der Eingabeebene beschrieben werden.* |
| Spärliche verteilte Repräsentation | eine Repräsentation, die aus vielen Bits besteht, von denen ein kleiner Prozentsatz aktiv ist, und in der kein einzelnes Bit ausreichend Bedeutung trägt |
| Sub-Sampling | Das Erkennen eines großen verteilten Musters durch das Finden einer kleinen Teilmenge aktiver Bits im Gesamtmuster |
| Synapse | Verbindung zwischen Zellen, die beim Lernen gebildet wird |

| | |
|---|---|
| Temporale Muster-zusammenfassung | der Prozess der Bildung einer Repräsentation einer Sequenz von Eingabemustern, bei der die Ergebnisrepräsentation stabiler als die Eingabe ist |
| Vorhersage | Zellaktivierung (in den vorhersagenden Zustand), die anzeigt, welche Zellen wahrscheinlich in naher Zukunft aufgrund von „feed-forward“-Eingaben aktiv werden<br><br>*Eine HTM-Region sagt häufig viele mögliche zukünftige Eingaben gleichzeitig voraus.* |
| Vorhersage erster Ordnung | eine Vorhersage, die nur auf der aktuellen Eingabe beruht und nicht auf vorherigen Eingaben – vgl. mit Vorhersage variable Ordnung |
| Vorhersage variabler Ordnung | Eine Vorhersage, die auf unterschiedlichen Mengen vorheriger Eingaben basiert – vgl. mit Vorhersage erster Ordnung<br><br>*Die Bezeichnung “variabel” stammt daher, dass der Speicher so viel vorherigen Kontext speichert, wie nötig. Daher kann ein System, welches die Vorhersage variabler Ordnung umsetzt, Kontext aus weiter Vergangenheit nutzen ohne exponentiell viel Speicher zu benötigen.* |
| Zellen | HTM –Äquivalent eines Neuron<br><br>*Zellen sind in HTM-Regionen kolumnar organisiert.* |