Understanding HTM Sequence Memory

Hugo Pristauz, Walter Eder – Oct. 2023 (Draft v0.1)

Abstract

This paper targets to give aid for people who aim to jump start with Numenta's HTM (Hierarchical Temporal Memory) sequence memory algorithm. Sequence memory is capable of learning sequences, and to predict the next item, when part of a sequence is presented to the sequence memory.

The first three targets of this work are to setup some prerequisites: 1) to set up a formal framework HTM sequence memory and a detailed formulation of the HTM algorithm, which can be efficiently execute in Matlab or Python. 2) Harmonizing the algorithm with the standard representation of nonlinear discrete time state representation, commonly used in system and control theory. 3) Running a toy network comprising a total of 40 neurons with an investigation characteristic input sequencies.

The main target is 4) a performance comparison of HTM sequence memory with transformer based neural network alternatives by predicting the continuation of text sequences from "Tiny Shakespeare"

Introduction

In his revolutionary book *On Intelligence* [1] Jeff Hawkins, with the help of Sandra Blakeslee, describes "how a new understanding of the brain will lead to the creation of truly intelligent machines", suggesting that current "machine intelligence" is still far away from human intelligence. In his later book *A Thousand Brains* [2] Jeff Hawkins suggests a new *definition of intelligence* for beings (humans, animals, machines) on base of the capability to leveraging four abilities, which can be definitely found in the human brain:

- Continuous learning: the being must have the ability of continuous (on-line) learning, which is important for continuous adopting to a continuously changing world. This contrasts with current neural networks, which must be pre-trained before being utilized in an operational mode.
- Creating a model of the world by physical (or mental) movement through the world, with the ability to build an understanding of the world by sensing sequential information of the real word.
- Simultaneous creation of multiple models of the world in order to make accurate predictions. The human brain is mainly a prediction machine, which benefits from the possibility to select the best prediction model of a pool of simultaneously entertained prediction models.
- Utilization of Reference frames for any kind of knowledge storage. Reference frames can be dynamically assembled, which, in consequence allows to "assemble" knowledge which is stored with respect to reference frames.

Capability 2 of this list (world understanding based on sequential sensor information) emphasizes the importance of *dealing with sequential information* (with both spatial and temporal attributes). In this sense *sequence memory*, as it is under focus in this paper, might be a powerful building block for required skill sets, as described above. In addition, capability 3 to predict is not of less importance.

Biological Background

HTM (Hierarchical Temporal Memory) Sequence memory is a computer model of a biological neuron, which is based on a modern neuro-physical understanding of cortical neuron functionality [3,4].

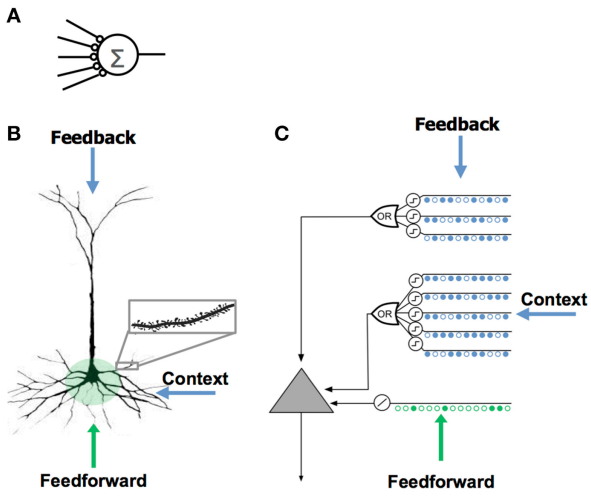

Figure 1: Comparison of neuron models

The neuron model used in most artificial neural networks is known as the perceptron [5], proposed by Frank Rosenblatt in 1957. It has few synapses and no modelling of dendrites (figure 1/A). The perceptron attempts to model the proximal area of the neuron and represents de-facto a mapping of input information arriving at the proximal synapses to the neuron's *output,* as the proximal synapses, those closest to the soma (cell body), have a large effect on the likelihood of a cell generating an *action potential,* which is bursting along the neuron's axon to synapses of other neurons of the network. It is important to realize that the perceptron model is a pure mapping model without any memory.

In contrast, an activation of a distal (non-proximal) synapse has little effect at the soma (cell body). For this reason, it was hard to understand how the thousands of distal synapses can play a role for the cell's responses [4].

Powered by advanced research methods [3], researchers could, however, unlock the secrets, that an activation of neighbored distal synapses within a short time interval leads to a local dendritic NMDA (N-methyl-D-aspartate) spike which causes a depolarization of the soma. Such depolarization subsequently enables the neuron to fire earlier than neighbor neurons with comparable proximal excitation, which gives such neuron the benefit to inhibit neighbored neurons.

Since NMDA spikes have a much longer duration than the action potential spikes of the soma, this mechanism acts like a memory functionality related to the depolarized state, in addition to the (perceptron like) mapping functionality of the neuron. Availability of memory in a neural model offers both the ability of *prediction*, and to implement *sequence state*, as will be shown in further sections.

Thus, compared to the perceptron model of figure 1/A, which generates the output signal according to a mapping of input signals at the proximal synapses (feedforward information) an extended neural model (like the HTM neuron model) deals additionally with context information arriving at distal synapses of several dendritic segments, which causes a state change of the neural model, influencing the neuron's behavior in the near future.

Synaptic Model

In Neurobiology a synapse forms a connection between a neuron's dendrite and an axon of some other neuron.

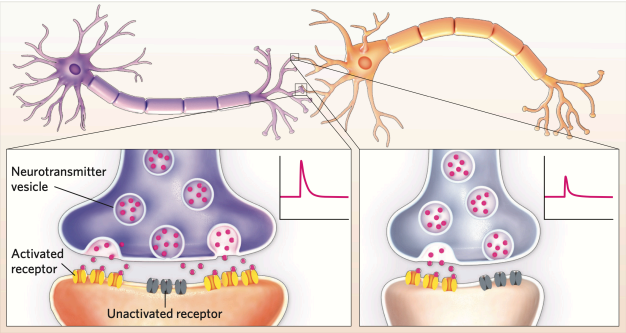

Figure 2: axon/dendrite connections via synapses

In formal neuron models synapses are usually assigned with a weight, influencing the transmission efficacy of the synapse, which is changed during the learning process. In the HTM approach there are two fundamental differences compared to the learning rules of most neural models.

- First, learning occurs by growing and removing synapses from a pool of "potential" synapses [7].
- Second, Hebbian learning and synaptic change occur at the level of a dendritic segment, not the entire Neuron [8].

Learning in an HTM neuron is modeled by the growth of new synapses from a set of potential synapses. A "permanence" value is assigned to each potential synapse and represents its growth. Learning occurs by incrementing or decrementing the permanence values. The synaptic weight is a binary value set to 1 if the permanence is above a threshold (figure 3).

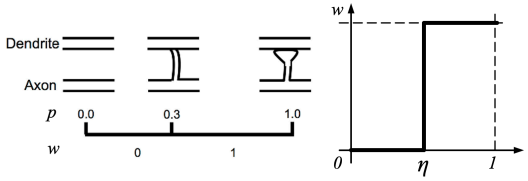

Figure 3: permanence p and weight of a synapse

Since the pre-synaptic signal is also binary, the synapse acts as a logical gate. For sufficient high permanence $p \geq \eta$, the synapse is called *connected* and allows a pre-synaptic signal to cause a post-synaptic effect *(q=z)*, while otherwise the synapse behaves as *unconnected* with no post-synaptic effect, regardless of the value of the pre-synaptic signal *(q=0)*.

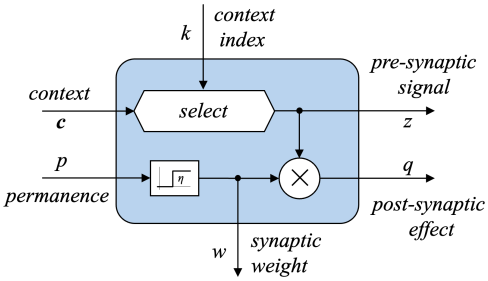

Figure 4: Block diagram for a single HTM neuron

This is shown in figure 4, where the (binary) pre-synaptic signal z is gated by the binary *synaptic weight* $w = (p \geq \eta)$, the Boolean result of the comparison of permanence $p \in [0,1]$ with synaptic threshold η (a typical setting is $\eta = 0.5$). The other block input is context $\boldsymbol{c} \in \{0,1\}^L$, typically a high dimensional binary vector collecting neuron outputs from a large neuron context set. The *select* function block in figure 5 means to pick the k-th element of *context* $\boldsymbol{c}$ for the pre-synaptic signal z. In this sense index k contributes to the definition of the neural network topology. The formal model for an HTM-synapse is as follows:

$$w = (p \geq \eta) \quad w \in \{0,1\}, \text{ synaptic weight} \tag{1a}$$
$$z = \boldsymbol{i}_k^T \cdot \boldsymbol{c} = c_k \quad z \in \{0,1\}, \text{ pre-synaptic signal} \tag{1b}$$
$$q = w \cdot z \quad q \in \{0,1\}, \text{ post-synaptic effect} \tag{1c}$$

with $\boldsymbol{i}_k^T$ denoting the *k-th* unit vector *of* $\{0,1\}^L$, and ^{T} used for transpose.

Synaptic Regions

So far (1a-1c) tells us whether a given context $\boldsymbol{c}$ presented to a synapse will cause a post-synaptic effect q with respect to the current synaptic permanence p, which is a sort of synaptic state.

A formal learning rule, however, is still missing, and we postulated for our model, that *learning* and *synaptic change* shall occur on the level of dendritic segments and not the entire neuron. This leads us to the concept of *synaptic regions* where groups of synapses collaborate with each other in order to generate some group effect.

As an example, the synapses of all proximal dendrites (receiving feedforward signals - see figure 1) collaborate with each other in order to cause occasionally the soma to generate an action potential. In this case we speak about the *proximal synaptic region*.

In contrast we have several *distal synaptic regions* formed by a collection of all synapses of distal dendritic segments. If those synapses generate sufficient collective post-synaptic effect, the distal dendritic segment can generate an NMDA spike, which can put the neuron into a so called *predictive state* by depolarization.

While we do not cover the proximal synaptic region in this paper, we will deal with the formal HTM-model for the distal synaptic regions related to distal dendritic segments. We will start with a generic formal model for a synaptic region (figure 6).

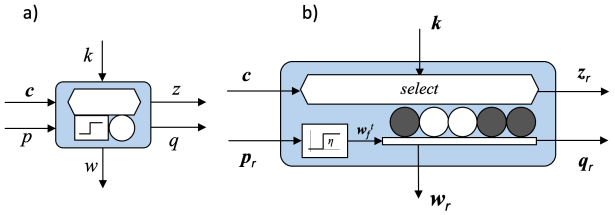

Figure 5: block symbol for a) single synapse, b) synaptic region

While in figure 5/a input/output signals of a single neuron's model are scalars (except context $\boldsymbol{c}$), we extend the single-synapse model in figure 5/b to model a synaptic region with a total number of s synapses. The individual synapses are still considered to work independent from each other according to (1a-1c), except the s instances of signals p, w, q, z, and k are now grouped in terms of vectors $\boldsymbol{p}_r$, $\boldsymbol{w}_r$, $\boldsymbol{q}_r$, $\boldsymbol{z}_r$ augmented with index r to refer to synaptic region $r \in \{1, \ldots, d\}$, d denoting the total number of *synaptic regions.*

The formal model is straight forward,

$$\boldsymbol{w}_r = (\boldsymbol{p}_r \geq \eta) \quad \boldsymbol{w}_r \in \{0,1\}^d, \text{ synaptic weights} \quad (1a)$$
$$\boldsymbol{z}_r = \boldsymbol{c}(\boldsymbol{k}_r) \quad \boldsymbol{z}_r \in \{0,1\}^d, \textit{ pre-synaptic signals} \quad (1b)$$
$$\boldsymbol{q}_r = \boldsymbol{w}_r \circ \boldsymbol{z}_r \quad \boldsymbol{q}_r \in \{0,1\}^d, \textit{ post-synaptic effects} \quad (1c)$$

with $\circ$ denoting elementwise product, and $\boldsymbol{c}(\boldsymbol{k}_r)$ meaning to pick d elements from $\boldsymbol{c}$ relating to the indices given by $\boldsymbol{k}_r$.

Mathematical Model of HTM Neuron

In HTM terminology a single HTM neuron is called a *cell*, which can be treated in the first approach as a black box with the input/output model shown in figure 3.

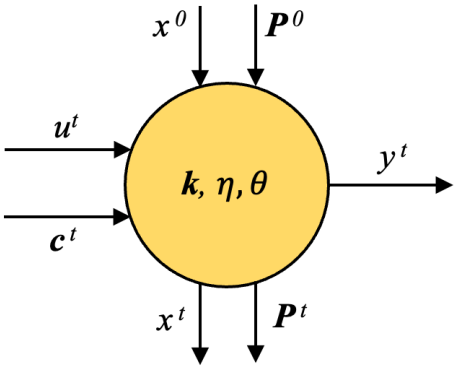

Figure 6: input/output model of a cell (HTM neuron)

All quantities shown in figure 3 are related to a discrete time index t ($t = 0, 1, 2, \ldots$). The HTM neuron presented in this paper has two kinds of inputs:

- the (scalar) binary *feedforward input* $u^t \in \{0,1\}$
- the (vectorial) binary *context input* $\boldsymbol{c}^t \in \{0,1\}^L$.

Further has the HTM neuron a state comprising

- the (scalar) binary cell state $x^t \in \{0,1\}$
- the permanenc matrix $\boldsymbol{P}^t \in [0,1]^{s \times d}$

with [0,1] denoting the closed interval $\{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$. Since the tuple $(x^t, \boldsymbol{P}^t)$ represents a state of the cell, which will be sequentially updated at each time step t, we need to provide

- initial cell state $x^0 \in \{0,1\}$
- initial permanence matrix $\boldsymbol{P}^0 \in [0,1]^M$,

which is sketched in figure 3 by the two input signals at the top. Finally, we have

- the (scalar) binary cell output $y^t \in \{0,1\}$.

In addition to inputs and outputs we have some cell parameters $\boldsymbol{k} \in (1,2,\ldots,N)^M$ and threshold values $\eta \in [0,1]$ and $\Theta \in \{0,1,\ldots,N\}$, which will be explained in more detail later.

Depending on the binary values of their input, state and output we will call a cell in the following as

- *excited*, when its input is activated ($u^t = 1$)
- *predictive*, when its state is activated ($x^t = 1$)
- *active*, when its output is activated ($y^t = 1$)

HTM Neuron Layer

To implement HTM sequence memory we need many *cells*, and we arrange them in terms of an *mn* matrix with n columns, each containing *m* cells (figure 4), resulting in a layer of total $N := mn$ cells.

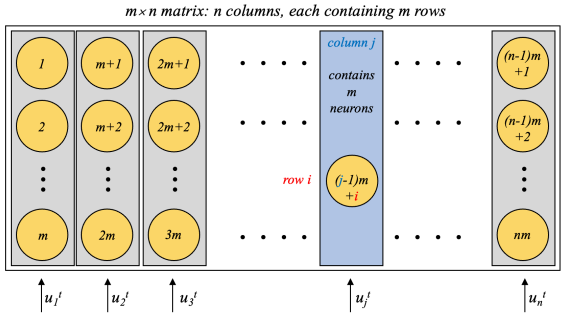

Figure 4: HTM layer with a matrix of *m n* neurons

In HTM terminology each column of the layer matrix is called a *minicolumn*, and each of them is attached with a scalar binary input $u_j^t \in \{0,1\}$. In total the HTM layer receives *n* scalar inputs u_j^t, which are combined as the (binary) *input vector*

$$\boldsymbol{u}^t = [\, u_1^t, u_1^t, \ldots, u_n^t \,]^T \in \{0,1\}^n . \qquad (1)$$

As figure 4 suggests, each cell of a particular minicolumn *j* receives the *same* binary input u_j^t. In a similar way we provide quantities x^t, y^t and $\boldsymbol{d}^t$ of figure 3 with cell index $k \in \{1, \ldots, N\}$ and combine them to (binary) vectors $\boldsymbol{x}^t$, $\boldsymbol{y}^t$ and (non-binary) matrix $\boldsymbol{D}^t$.

$$\boldsymbol{x}^t = [\, x_1^t, x_2^t, \ldots, x_N^t \,]^T \in \{0,1\}^N \qquad (2a)$$
$$\boldsymbol{y}^t = [\, y_1^t, y_2^t, \ldots, y_N^t \,]^T \in \{0,1\}^N \qquad (2b)$$
$$\boldsymbol{D}^t = [\, \boldsymbol{d}_1^t, \boldsymbol{d}_2^t, \ldots, \boldsymbol{d}_N^t \,] \in [0,1]^{M \times N} \qquad (2c)$$

Note that the binary context input $\boldsymbol{v}^t$ collects all cell outputs $\boldsymbol{y}^t$ and is applied as a common context input to all cells (thus needs no augmentation). Formally we write $\boldsymbol{v}^t = \boldsymbol{y}^t$, which introduces a contextual feedback loop to the cell layer, as it is shown in figure 5.

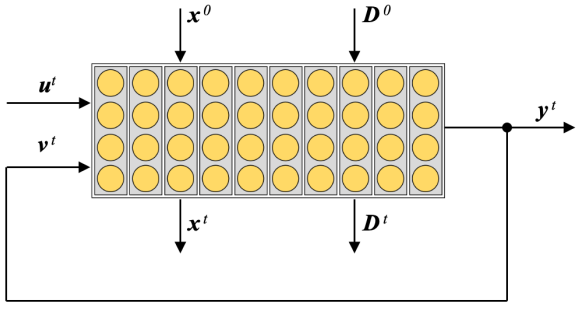

Figure 5: input/output model of cell layer with contextual feedback loop

Input Sequences

Let us now talk in more detail about how input sequences are presented to the cell layer of figure 5. Let us consider two sentences “Mary likes to sing.” and “John likes to dance.” To formally deal with these sequences, we define a vocabulary

$$V = \{\text{'Mary'}, \text{'John'}, \text{'likes'}, \text{'to'}, \text{'sing'}, \text{'dance'}, \text{'.'}\}$$

and related sequences

$$S_1 = (\text{'Mary'}, \text{'likes'}, \text{'to'}, \text{'sing'}, \text{'.'})$$
$$S_2 = (\text{'John'}, \text{'likes'}, \text{'to'}, \text{'dance'}, \text{'.'}) .$$

Since each input at time instance *t* to our cell layer must be a binary vector $\boldsymbol{u}^t \in \{0,1\}^N$, we must map each symbol of sequence S_1 and S_2 to the binary vector space $\{0,1\}^n$, and we call this process tokenizing, and talk about the binary vectors $\boldsymbol{u}^0, \boldsymbol{u}^1, \boldsymbol{u}^2, \ldots$ as *tokens,* which are sequentially presented as input vectors $\boldsymbol{u}^t$ to the cell layer at time instances $t = 0, 1, 2, \ldots$.

In this sense we can define a tokenizer as a function $\Phi: V \rightarrow \{1,0\}^n$ mapping from a vocabulary *V* to the binary input space $\{1,0\}^n$.

Example 1: 4 x 10 Toy Layer with 5 x 2 synapses per cell

Let us define a 410 cell layer ($m=4$, $n=10$) where each cell supports 2 dendritic segments ($d=2$) with 5 synapses per segment ($s=5$). Thus, the layer has $n = 10$ minicolumns with a total of $N = m \cdot n = 40$ cells, where each cell entertains $M = s \cdot d = 10$ synapses in total. Referring to vocabulary

$$V = \{\text{'Mary'}, \text{'John'}, \text{'likes'}, \text{'to'}, \text{'sing'}, \text{'dance'}, \text{'.'}\}$$

we could define a tokenizer $\Phi: V \rightarrow \{1,0\}^{10}$ with the mapping

'Mary' ↦ $[0\,0\,1\,0\,0\,1\,1\,1\,0\,0]^T$
'John' ↦ $[1\,0\,0\,1\,0\,1\,1\,0\,0\,0]^T$
'likes' ↦ $[0\,1\,0\,0\,0\,0\,0\,1\,1\,1]^T$
'to' ↦ $[0\,1\,0\,1\,1\,0\,0\,0\,0\,1]^T$
'sing' ↦ $[1\,1\,0\,0\,1\,0\,1\,0\,0\,0]^T$
'dance' ↦ $[1\,0\,0\,1\,1\,0\,0\,0\,0\,1]^T$

In this sense the tokenizer maps the sequence

$$S_1 = (\text{'Mary'}, \text{'likes'}, \text{'to'}, \text{'sing'}, \text{'.'})$$

to the input token sequence

$$U_1 = \{\boldsymbol{u}^0, \boldsymbol{u}^1, \boldsymbol{u}^2, \boldsymbol{u}^3, \boldsymbol{u}^4\} = \left\{ \begin{bmatrix}0\\0\\1\\0\\0\\1\\1\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\\0\\0\\1\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\1\\1\\0\\0\\0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\\0\\1\\0\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\1\\0\\0\\1\\1\\0\\0\end{bmatrix} \right\},$$

and sequence

S_2 = ('John', 'likes', 'to', 'dance', '.') .

To the input token sequence

$$U_2 = \{\boldsymbol{u}^0, \boldsymbol{u}^1, \boldsymbol{u}^2, \boldsymbol{u}^3, \boldsymbol{u}^4\} = \left\{\begin{bmatrix}1\\0\\0\\1\\0\\1\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\\0\\0\\1\\1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\1\\1\\0\\0\\0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\\1\\1\\0\\0\\0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\\1\\0\\0\\1\\1\\0\\0\end{bmatrix}\right\}.$$

Distal Dendritic Segments

Let us next investigate the distal dendritic segments of a cell and consult figure 5, which shows a neuron with two (d=2) distal dendritic segments, each of them entertaining five synapses (s=5).

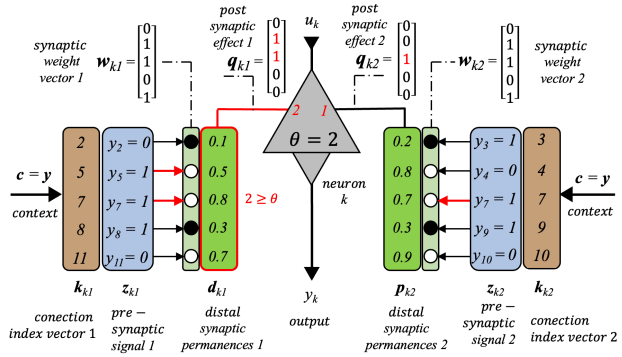

Figure 5: neuron with *2* distal dendritic segments and *5* synapses per segment (s = 5, d = 2, time index has been omitted)

According to figure 3 the model for cell k (HTM neuron k) provides both a *parameter vector* $\boldsymbol{k}_k$ as well as a *distal permanence vector* $\boldsymbol{p}_k^t$ for the total set of distal segments. Broken down to a single distal dendritic segment these vectors can be split up into partial parameter vectors $\boldsymbol{k}_{k1}$, $\boldsymbol{k}_{k2}$, …, $\boldsymbol{k}_{kd}$ and partial distal permanence vectors $\boldsymbol{p}_{k1}^t$, $\boldsymbol{p}_{k2}^t$, …, $\boldsymbol{p}_{kd}^t$ according to

$$\boldsymbol{k}_k = [\boldsymbol{k}_{k1}^T, \boldsymbol{k}_{k2}^T, \ldots, \boldsymbol{k}_{kd}^T]^T \in \{i \mid 0 \le i \le N \wedge i \ne k\}^N \quad (3a)$$

$$\boldsymbol{p}_k^t = [\boldsymbol{p}_{k1}^{tT}, \boldsymbol{p}_{k2}^{tT}, \ldots, \boldsymbol{p}_{kd}^{tT}]^T \in \{i \mid 0 \le i \le N \wedge i \ne k\}^N \quad (3b)$$

Referring to figure 5 we have for example

$$\boldsymbol{k}_k = [\,2\ \ 5\ \ 7\ \ 8\ \ 11\,|\,3\ \ 4\ \ 7\ \ 9\ \ 10\,]^T$$
$$\boldsymbol{p}_k^t = [0.1\ \ 0.5\ \ 0.8\ \ 0.3\ \ 0.7\,|\,0.2\ \ 0.8\ \ 0.7\ \ 0.3\ \ 0.9]^T$$

with partial vectors

$$\boldsymbol{k}_{k1} = [\,2\ \ 5\ \ 7\ \ 8\ \ 11\,]^T, \boldsymbol{k}_{k2} = [\,3\ \ 4\ \ 7\ \ 9\ \ 10\,]^T$$
$$\boldsymbol{p}_{k1}^t = [0.1\ \ 0.5\ \ 0.8\ \ 0.3\ \ 0.7]^T, \boldsymbol{p}_{k2}^t = [0.2\ \ 0.8\ \ 0.7\ \ 0.3\ \ 0.9]^T$$

The connection index vectors $\boldsymbol{k}_{k\mu}$ are telling us which other *cell outputs* y_i are connected to synapses *of the distal dendritic segment* μ. As figure 5 demonstrates, segment 1 has *5* synapses which connect to cell outputs y_2, y_5, y_7, y_8, and y_{11}, according to the index values contained in $\boldsymbol{k}_{k1}$. In a similar way the *5* synapses of segment 2 connect to cell outputs y_3, y_4, y_7, y_9, and y_{10}, related to the index values contained in $\boldsymbol{k}_{k2}$.

Formally the connection vector $\boldsymbol{k}_{k\mu}$ defines a projection $P_{k\mu}$: $\{0,1\}^N \rightarrow \{0,1\}^s$ given $\boldsymbol{v}^t \mapsto \boldsymbol{v}^t(\boldsymbol{k}_{k\mu}) =: z_{k\mu}{}^t$. In other words, from the total context $\boldsymbol{v}^t$ a distal dendritic segment μ of cell k "sees" only the s values of the projected vector $z_{k\mu}{}^t$, which are picked according to the s index values defined by $\boldsymbol{k}_{k\mu}$. Since the context vector $\boldsymbol{v}^t$ is identical with layer output vector $\boldsymbol{y}^t$, we have also:

$$z_{k\mu} = \boldsymbol{y}^t(\boldsymbol{k}_{k\mu}) \qquad \text{for all segments } \mu \text{ in cell } k$$

If we continue the example shown in figure 5 by assuming

$$\boldsymbol{c}^t = [0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ \ldots]$$

then we get

$$z_{k1}{}^t = \boldsymbol{c}^t(\boldsymbol{k}_{k1}) = \boldsymbol{c}^t([2, 5, 7, 8, 11]) = [0\ 1\ 1\ 1\ 0]^T$$
$$z_{k2}{}^t = \boldsymbol{c}^t(\boldsymbol{k}_{k2}) = \boldsymbol{c}^t([3, 4, 7, 9, 10]) = [1\ 0\ 1\ 1\ 0]^T$$

for the projected context. Whether these projected cell outputs will all contribute to the excitation of the distal dendritic segment, depends on the following *synaptic rule:*

- A synapse contributes to the excitation of a distal dendritic segment, if, and only if

  (i) the synapse is active (a binary value), which is given if permanence value of the synapse is greater than or equal the synaptic threshold η
  (ii) the synapse is connected to an active cell output, i.e., $y_l^t = 1$

Formally this can be modelled by introducing the binary synaptic weight vectors

$$\boldsymbol{w}_{k\mu}{}^t = (\boldsymbol{p}_{k\mu}{}^t \ge \eta) \qquad \boldsymbol{s}_{k\mu}{}^t \in \{0,1\}^s \qquad (4)$$

which has components equal to 1, if the relating component of permanence vector $\boldsymbol{p}_{k\mu}$ has a value greater or equal to synaptic threshold η, and zero otherwise.

Based on the concept of the pre-synaptic signal $z_{k\mu}$ and the synaptic vector $\boldsymbol{s}_{k\mu}$ we can define now the vector of post-synaptic effects for distal dendritic segment μ

$$\boldsymbol{q}_{k\mu}{}^t = z_{k\mu}{}^t \circ \boldsymbol{w}_{k\mu}{}^t, \qquad \boldsymbol{q}_{k\mu}{}^t \in \{0,1\}^s \qquad (5)$$

with $\circ$ denoting elementwise multiplication. If an element of the post-synaptic effect is 1, we know that both the permanence value of the related synapse is greater than or equal to synaptic threshold η, and we also know that the specific context variable relating to the synapse is also active (having value 1). Otherwise, one of the two conditions is violated.

In the next section, where we present the full HTM algorithm for the cell layer in figure 5, we will see, that the

segment activation vectors $\{ \boldsymbol{q}_{k1}, \boldsymbol{q}_{k2}, \dots , \boldsymbol{q}_{kd}\}$, which can be summarized as the *segment coincidence* matrix of cell *k*.

$$\boldsymbol{Q}_k{}^t = [\, \boldsymbol{q}_{k1}{}^t, \boldsymbol{q}_{k2}{}^t, \dots , \boldsymbol{q}_{kd}{}^t \,] , \quad \boldsymbol{Q}_k{}^t \in \{0,1\}^{sxd} \qquad (6)$$

It will play a key role for both state updates of a cell and reinforced learning, i.e., adaption of the synaptic permanence values (figure 6). For a compact formulation of the HTM algorithm we also define the *layer coincidence matrix* as

$$\boldsymbol{Q}^t = [\, \boldsymbol{Q}_1{}^t(:), \boldsymbol{Q}_2{}^t(:), \dots , \boldsymbol{Q}_d{}^t(:) \,] , \quad \boldsymbol{Q}^t \in \{0,1\}^{MxN} \qquad (7)$$

Before we tackle the HTM algorithm, let us complete the sample description of figure 5. First we calculate the synaptic vectors for each segment μ, assuming the common setting of $\eta = 0.5$.

$$\boldsymbol{s}_{k1}{}^t = (\boldsymbol{d}_{k1}{}^t \geq \eta) = ([0.1\ 0.5\ 0.8\ 0.3\ 0.7] \geq 0.5) = [0\ 1\ 1\ 0\ 1]$$
$$\boldsymbol{s}_{k2}{}^t = (\boldsymbol{d}_{k2}{}^t \geq \eta) = ([0.2\ 0.8\ 0.7\ 0.3\ 0.9] \geq 0.5) = [0\ 1\ 1\ 0\ 1]$$

Knowing now which synapses are currently in action, we can calculate the segment excitation.

$$\boldsymbol{q}_{k1}{}^t = \boldsymbol{z}_{k1}{}^t \circ \boldsymbol{s}_{k1}{}^t = [0\ 1\ 1\ 1\ 0]^T \circ [0\ 1\ 1\ 0\ 1]^T = [0\ 1\ 1\ 0\ 0]^T$$
$$\boldsymbol{q}_{k2}{}^t = \boldsymbol{z}_{k2}{}^t \circ \boldsymbol{s}_{k2}{}^t = [1\ 0\ 1\ 1\ 0]^T \circ [0\ 1\ 1\ 0\ 1]^T = [0\ 0\ 1\ 0\ 0]^T$$

$$\boldsymbol{Q}_k{}^t = [\, \boldsymbol{q}_{k1}{}^t, \boldsymbol{q}_{k2}] = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The HTM Algorithm

We are ready now to formulate the coarse form of the HTM algorithm for sequence memory. The original form is of this algorithm has been presented in [6]. The form which is used here makes more use of vector/matrix operations and less use of if-branches, which ends up in an algorithm with sequential, straight forward flow.

The computational scheme of the HTM algorithm for a sequence memory layer (as shown in figure 5) is outlined in figure 6.

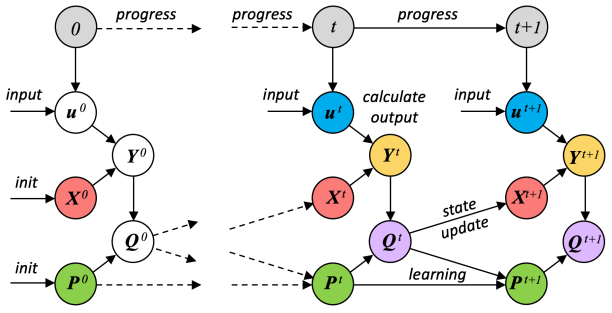

Figure 6: computational scheme for the HTM algorithm

Besides of the quantities $\boldsymbol{u}^t, \boldsymbol{x}^t, \boldsymbol{D}^t$ and $\boldsymbol{y}^t$, which are shown in figure 5, the *coincidence matrix* $\boldsymbol{Q}^t$ introduced in the previous section plays an important computational role as an intermediate quantity. Figure 6 can be interpreted as follows: After initializing the layer state $(\boldsymbol{x}^0, \boldsymbol{D}^0)$ at time $t=0$ a sequence of updating iterations is performed repeatedly for the discrete time sequence t = 0, 1, 2 ,...

initialize $\boldsymbol{x}^0, \boldsymbol{D}^0$
for $t = 0, 1, 2, \dots :$
  input $\boldsymbol{u}^t$
  calculate $\boldsymbol{y}^t$ from $\boldsymbol{x}^t, \boldsymbol{u}^t$ (output activation)
  calculate $\boldsymbol{Q}^t$ from $\boldsymbol{D}^t, \boldsymbol{y}^t$ (coincidence matrix)
  update $\boldsymbol{x}^{t+1}$ from $\boldsymbol{Q}^t$ (cell state update)
  update $\boldsymbol{D}^{t+1}$ from $\boldsymbol{D}^t, \boldsymbol{Q}^t$ (cell state update)

Before diving into the algorithmic details let us recap that we call a cell *k* with activated input *excited* ($u_k{}^t = 1$). In case of activated output ($x_k{}^t = 1$) we call the *active*, and we talk about a *predictive* cell when its cell state is set ($x_k{}^t = 1$). Also keep in mind that all cells belonging to the same minicolumn *j* share the same input $u_j{}^t$. Thus, an excitation of one cell in a minicolumn implies an excitation of all cells in the minicolumn, which suggests calling the whole minicolumn *excited*, whenever one cell (which implies: all cells) of the minicolumn is excited.

In addition, we call a minicolumn predictive, if it contains at least one predictive cell, otherwise the minicolumn is called non-predictive. An excited minicolumn with no predictive cells is called a *bursting minicolumn*.

Finally keep in mind that *cell excitation* (referring to input $u_k{}^t$) and *segment excitation* (referring to distal dendritic segment excitation $\boldsymbol{Q}_k{}^t$) are completely different concepts, and neither *cell excitation* implies necessarily *segment excitation*, or vice versa.

Step 1) Output Activation

The rules for output activation, where calculate $\boldsymbol{y}^t$ is calculated from $\boldsymbol{x}^t, \boldsymbol{u}^t$, are as follows:

- excited predictive cells get active
- cells of bursting minicolumns get active
- any other cell gets inactive

The first rule is biologically inspired by the capability of pyramidal cells to transition into a mid-term (> 100ms) depolarized state caused by distal dendritic NMDA spikes [3,4]. Such depolarized neurons, which play a predictive role, are easier to activate and will start to fire earlier than non-depolarized neurons [6].

The second rule covers just the default, where there are no depolarized neurons in a minicolumn. In this case no neuron has an activation advantage against other neurons in the minicolumn, and all cells of the bursting minicolumn (which is by definition *excited*) become *active* [6].
The calculations are as follows:

$$\boldsymbol{B}^t = \boldsymbol{1} - \boldsymbol{1}^m \cdot \boldsymbol{max}(\boldsymbol{X}^t) \quad \text{burst matrix } \boldsymbol{B}^t \in \{0,1\}^{m \times n} \quad (8a)$$
$$\boldsymbol{C}^t = \boldsymbol{X}^t + \boldsymbol{B}^t \quad \text{candidates } \boldsymbol{C}^t \in \{0,1\}^{m \times n} \quad (8b)$$
$$\boldsymbol{Y}^t = \boldsymbol{C}^t \circ (\boldsymbol{1}^m \cdot \boldsymbol{u}^{tT}) \quad \text{output activation} \quad (8c)$$

Even this kind of formulation looks a bit unusual, the chosen form has the advantage of generating a straight-forward sequence of intermediate matrix results

$$\boldsymbol{x}^t \rightarrow \boldsymbol{X}^t \rightarrow \boldsymbol{B}^t \rightarrow \boldsymbol{C}^t \rightarrow \boldsymbol{y}^t$$

with helpful interpretations, as will be demonstrated in the subsequent example. In the consolidated form (9) we clearly see the dependency of $\boldsymbol{y}^t$ from $\boldsymbol{x}^t$ and $\boldsymbol{u}^t$.

$$\boldsymbol{Y}^t = (\boldsymbol{X}^t + (\boldsymbol{1} - \boldsymbol{1}^m \cdot \boldsymbol{max}(\boldsymbol{X}^t)) \circ (\boldsymbol{1}^m \cdot \boldsymbol{u}^{tT}) \quad (9)$$

Step 2) Calculate Coincidence Matrix

In this step the *layer's coincidence matrix* $\boldsymbol{Q}^t$ is calculated from the permanence state $\boldsymbol{D}^t$ and the layer output $\boldsymbol{y}^t$. The rationale behind is that each cell entertains d distal dendritic segments which act as coincidence detectors. Any element of the binary matrix $\boldsymbol{Q}^t$, which equals 1, tells about a coincidence of a related *enabled* distal synapse with a connected active cell. In contrast, a value of 0 means that either the synapse is disabled, or the connected cell is inactive, or both.

The details for the calculation of the coincidence matrix have been explained previously in detail. Thus, we only have to summarize the required calculations of this step.

$$\boldsymbol{S}^t = \sigma(\boldsymbol{D}^t - \eta) \quad \text{synaptic matrix} \quad (10a)$$
$$\boldsymbol{Z}^t = \boldsymbol{y}^t(\boldsymbol{K}) \quad \text{projected context} \quad (10b)$$
$$\boldsymbol{Q}^t = \boldsymbol{S}^t \circ \boldsymbol{Z}^t \quad \text{coincidence matrix} \quad (10c)$$

Again (10a-c) can be consolidated to a single operation:

$$\boldsymbol{Q}^t = \sigma(\boldsymbol{D}^t - \eta) \circ \boldsymbol{y}^t(\boldsymbol{K}) \quad (11)$$

Each Boolean element of the *synaptic matrix* $\boldsymbol{S}^t$ in (10a) tells us, whether the related synapse is enabled or disabled, depending on whether its permanence value exceeds threshold η or not. In (10b) the layer output $\boldsymbol{y}^t$ is projected into a subspace to yield $\boldsymbol{Z}^t$ according to the index values of **K** in order to achieve compatible matching with the synaptic matrix $\boldsymbol{S}^t$. Finally $\boldsymbol{S}^t$ and $\boldsymbol{Z}^t$ are element-wise multiplied to establish the coincidence values.

Step 3) Cell State Transition

The aim of this step is to update *the cell state* by calculating $\boldsymbol{x}^{t+1}$ from the *coincidence matrix* $\boldsymbol{Q}^t$.

$$\boldsymbol{p}^t = [\, ||\, \boldsymbol{Q}_1{}^t\, ||_1, ||\, \boldsymbol{Q}_2{}^t\, ||_1, \ldots, ||\, \boldsymbol{Q}_n{}^t\, ||_1]^T \in \{0 \ldots s\}^n \quad (12a)$$
$$\boldsymbol{x}^{t+1} = (\boldsymbol{p}^t \geq \theta) \quad (12b)$$

Equation (12) means that the given *coincidence matrix* $\boldsymbol{Q}^t$ has to be decomposed according to (7) into its parts $\boldsymbol{Q}_k{}^t$. Then, according to (6), for each $\boldsymbol{Q}_k^t$ we calculate the coincidence number

$$p_k{}^t = ||\, \boldsymbol{Q}_k{}^t\, ||_1 = max(||\boldsymbol{q}_{k1}{}^t||_1, ||\boldsymbol{q}_{k2}{}^t||_1, \ldots, ||\boldsymbol{q}_{kd}{}^t||_1)$$

as the largest column sum, and form vector

$$\boldsymbol{p}^t = [p_1{}^t, p_2{}^t, \ldots, p_n{}^t\,]^{T.} \in \{0 \ldots s\}^n$$

In (12b), finally, we set cell state $x_k{}^{t+1} = 1$ when the related coincidence number exceeds a threshold $(p_k{}^t \geq \theta)$, otherwise we set $x_k{}^{t+1} = 0$. Again (12a,12b) can be replaced by the consolidated form (13), which expresses the dependency of $\boldsymbol{x}^{t+1}$ from $\boldsymbol{Q}^t$

$$\boldsymbol{X}^{t+1} = ([||\boldsymbol{Q}_1{}^t||_1, ||\boldsymbol{Q}_2{}^t||_1, \ldots, ||\boldsymbol{Q}_n{}^t||_1]^T \geq \theta)(m \times n) \quad (13)$$

Step 4) Learning

In the last state the permanence values of the distal dendritic segments are updated, which is called learning. The following rules are applied:

- synapses of active cells with coincidence increase their permanence values
- synapses of active cells with no coincidence decrease their permanence values
- inactive cells do not change their permanence values

Formally we have

$$\Delta\boldsymbol{D}^t = (\delta^+\boldsymbol{Q}^t - \delta^-) \circ \boldsymbol{Y}^t \quad (14a)$$
$$\boldsymbol{D}^{t+1} = \boldsymbol{D}^t + \Delta\boldsymbol{D}^t \quad (14b)$$

with $\delta^+ > \delta^-$. The consolidated form is:

$$\boldsymbol{D}^{t+1} = \boldsymbol{D}^t + (\delta^+\boldsymbol{Q}^t - \delta^-) \circ \boldsymbol{Y}^t \quad (15)$$

Summary: HTM Algorithm

Taking (9,) the HTM algorithm looks in its compact form:

configure $\boldsymbol{K}$, init $\boldsymbol{X}^0$, $\boldsymbol{P}^0$
repeat for $t = 0,1,2,3$
  input $\boldsymbol{u}^t$
  $\boldsymbol{Y}^t = (\boldsymbol{X}^t + (\boldsymbol{1} - \boldsymbol{1}^m \cdot \boldsymbol{max}(\boldsymbol{X}^t)) \circ (\boldsymbol{1}^m \cdot \boldsymbol{u}^{tT})$
  $\boldsymbol{Q}^t = \sigma(\boldsymbol{D}^t - \eta) \circ \boldsymbol{y}^t(\boldsymbol{K})$
  $\boldsymbol{X}^{t+1} = ([||\boldsymbol{Q}_1{}^t||_1, ||\boldsymbol{Q}_2{}^t||_1, \ldots, ||\boldsymbol{Q}_n{}^t||_1]^T \geq \theta)(m \times n)$
  $\boldsymbol{D}^{t+1} = \boldsymbol{D}^t + (\delta^+\boldsymbol{Q}^t - \delta^-) \circ \boldsymbol{Y}^t$

References

[1] Hawkins J., Blakeslee S.: “On Intelligence”; Owl Books (2005).

[2] Hawkins J.: “A Thousand Brains”; Basic Books, New York (2022).

[3] Antic, S. D.,et al.: “The decade of the dendritic NMDA spike”; *J. Neurosci. Res.* 88, 2991–3001. doi: 10.1002/jnr.22444 (2010).

[4] Major, G., Larkum, M. E., and Schiller, J.: “Active properties of neocortical pyramidal neuron dendrites”; *Annu. Rev. Neurosci.* 36, 1–24. doi: 10.1146/annurev-neuro-062111-150343 (2013)

[5] Rosenblatt, F.: „The perceptron. A probabilistic model for information storage and organization in the brain“; *Psychological Reviews*, 65 (1958): S. 386–408.

[6] Hawkins J., Subutai A.: „Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex“; *frontiers in Neural Circuits, 2016*.

[7] Hawkins J., Subutai A., Cui Y.: „A Theory of How Columns in the Neocortex Enable Learning the Structure of the World“; *frontiers in Neural Circuits, 2017*.

[7] Chklovskii et al.: (2004)

[8] Stuart, Häusser: (2001)

[9] Berlyand L., Jabin P.E.: „Mathematics of Deep Learning; An Introduction“; *de Gruyter Textbook, 2023*.