Understanding HTM Sequence Memory

Hugo Pristauz, Walter Eder, Willy Ottinger – Oct. 2023 (Draft v0.4)

Abstract

This paper is for jump starters into a playground 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.

In this sense some pre-requisites are setup first: a formal framework for HTM sequence memory with a detailed formulation of the HTM algorithm, which can be efficiently execute in Matlab or Python. Focus is on modular formulation down to the HTM-neuron level with local state transition and learning rules. Algorithms are in harmony with the standard formulations of nonlinear discrete time state representations, commonly used in system and control theory.

Based on these pre-requisites the HTM-algorithm is studied by running it on a 40 neuron toy network to investigate the behavior when feeding the system with different "toy input sequences". Finally, the performance of HTM sequence memory is compared with transformer based neural network alternatives, when both systems have to predict the continuation of text sequences from "Tiny Shakespeare".

Introduction

In his 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 a skill set leveraging four abilities found in the human brain:

- *Continuous learning*: there must be the ability of continuous (on-line) learning, which is important for continuous adopting to a permanently changing world. This contrasts with current neural networks, which must be pre-trained before running in an operational mode.
- *Sequential world modelling*: Creating a world model by physical (or mental) movement through the world with sequential information sensing generated by inter-actions with the world.
- *Multi world prediction*: The brain of humans and mammals is an *adaptive prediction machine*, which benefits from the possibility to select the best of a pool of simultaneously created and entertained prediction models.
- *Reference framing*: Reference frames are used for any kind of knowledge storage. They can be dynamically assembled, which, in consequence allows hierarchical "assembly" of knowledge to be stored with respect to reference frames.

The challenges induced by this skill set emphasize the importance of *efficient dealing with sequential information* comprising both spatial and temporal attributes. In this sense *sequence memory* with its promising prediction properties seems to be a powerful building block for above listed skill set.

Biological Background

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

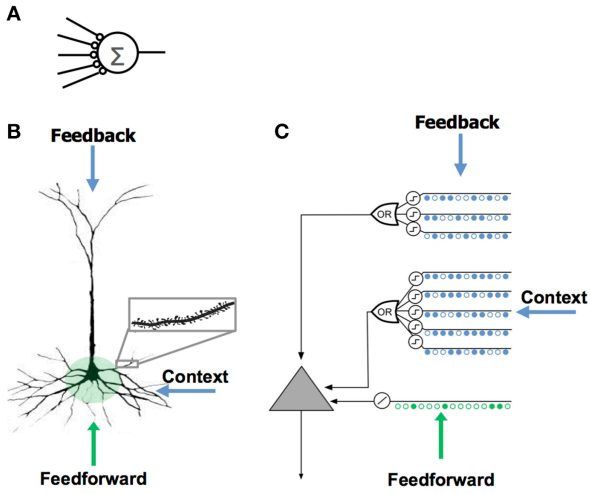

Figure 1: Comparison of neuron models

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.* There is evidence that the proximal synapses, those closest to the soma (cell body), have a large effect on the likelihood of the 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 excitation 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 1A, 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.

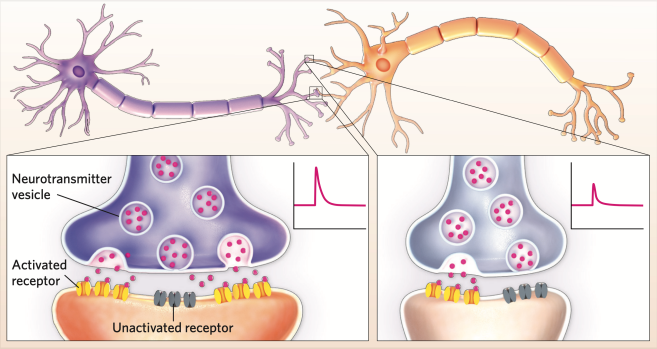

Figure 2: axon/dendrite connections via synapses

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), otherwise set to zero.

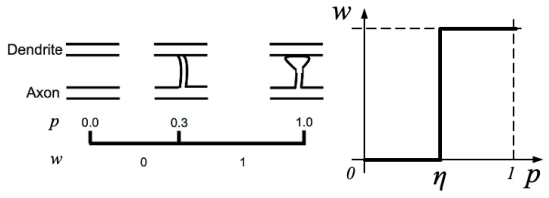

Figure 3: permanence p and weight of a synapse

Since the pre-synaptic signal is 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)*.

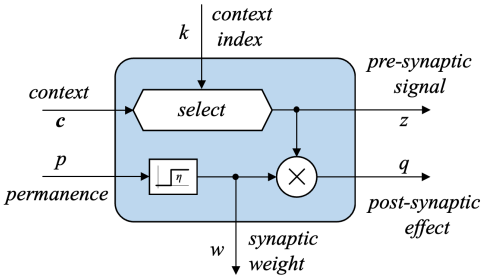

Figure 4: Block diagram for a single HTM synapse

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 neural network topology definition. The formal model for an HTM-synapse is as follows:

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

with $B = \{0,1\}$ defined as the binary set, $\boldsymbol{i}_k$ denoting the *k-th* unit vector *of* B^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 contribute to a *post-synaptic effect q* with respect to the current synaptic *permanence p*, which is a sort of synaptic state variable.

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 5).

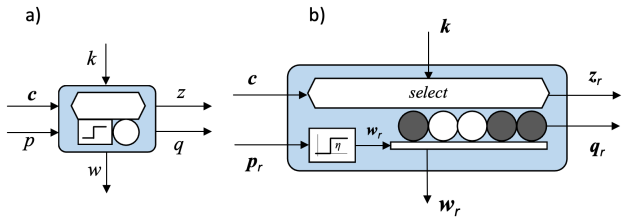

Figure 5: block symbol for a) single synapse, b) synaptic region

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

$$\boldsymbol{w}_r = (\boldsymbol{p}_r \geq \eta) \quad \boldsymbol{w}_r \in \mathcal{B}^d, \text{ synaptic weights} \qquad (2a)$$
$$\boldsymbol{z}_r = \boldsymbol{c}(\boldsymbol{k}_r) \quad \boldsymbol{z}_r \in \mathcal{B}^d, \textit{ pre-synaptic signals} \qquad (2b)$$
$$\boldsymbol{q}_r = \boldsymbol{w}_r \circ \boldsymbol{z}_r \quad \boldsymbol{q}_r \in \mathcal{B}^d, \textit{ post-synaptic effects} \qquad (2c)$$

[1] in German we say "Synaptik"

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 $\boldsymbol{k}_r$.

Synaptic Bank

The formal model of an HTM-neuron requires typically more than one synaptic regions. A set of synaptic regions can be combined to a synaptic bank, which is a *formal generic building block* used to model the behavior of a set of *distal dendritic segments*. Those play a key role in the HTM neuron for modelling soma depolarization and synaptic learning when sufficient spatially and temporally neighbored synapse activity occurs.

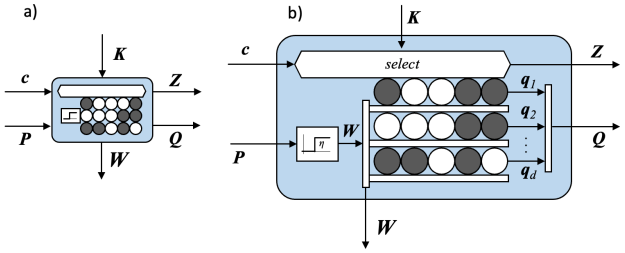

Figure 6: schematics of a *synaptic bank: a) less detailed, b) more detailed*

Figure 6a and 6b symbolize a synaptic bank with synapses arranged in a $d \times s$ matrix to model, e.g., a set of d dendritic segments with s synapses each. Given d synaptic regions indexed by r with *permanences* $\boldsymbol{p}_r$, *connection indices* $\boldsymbol{k}_r$, *synaptic weights* $\boldsymbol{w}_r$, and output signals $\boldsymbol{z}_r$, $\boldsymbol{q}_r$, we arrange these vectors as matrices.

$$\boldsymbol{P} = [\ldots, \boldsymbol{p}_r, \ldots], \ \boldsymbol{K} = [\ldots, \boldsymbol{k}_r, \ldots], \ \boldsymbol{W} = [\ldots, \boldsymbol{w}_r, \ldots]$$
$$\boldsymbol{Z} = [\ldots, \boldsymbol{z}_r, \ldots], \ \boldsymbol{Q} = [\ldots, \boldsymbol{k}_r, \ldots]$$

The representation of the listed vectors as matrix columns must not be confused with the horizontal arrangement of a synapse region in figure 6. Equations (2a-2c) give the formal description of the synapse bank functionality.

$$\boldsymbol{W} = (\boldsymbol{P} \geq \eta) \quad \boldsymbol{W} \in \mathcal{B}^{d \times s}, \text{ synaptic weights} \qquad (3a)$$
$$\boldsymbol{Z} = \boldsymbol{c}(\boldsymbol{K}) \quad \boldsymbol{Z} \in \mathcal{B}^{d \times s}, \textit{ pre-synaptic signals} \qquad (3b)$$
$$\boldsymbol{Q} = \boldsymbol{W} \circ \boldsymbol{Z} \quad \boldsymbol{Q} \in \mathcal{B}^{d \times s}, \textit{ "synaptics"} \qquad (3c)$$

Matrix $\boldsymbol{Q}$, which holds the plurality of synaptic effects of each synapse, plays such an important role that we invented the artificial name *synaptics*[1] for it, and it is worth to recap the meaning of the *synaptics* $\boldsymbol{Q}$:

- If an element $q_{\mu\nu}$ of the *synaptics* $\boldsymbol{Q}$ equals 1, it means both, that synapse μ of the synaptic field $\boldsymbol{q}_\nu$ is connected (permanence $q_{\mu\nu} \geq \eta$), and that the neuron addressed by index $k_{\mu\nu}$ (found in matrix $\boldsymbol{K}$ in row μ and column ν) fires its output signal through this synapse.

- If an element $q_{\mu\nu}$ of the *synaptics* $\boldsymbol{Q}$ equals 0, it means that at least one of the conditions above is violated.

We use the 1-norm of a vector $\| \mathbf{q} \|_1$ for the sum of absolute values of its elements, and the 1-norm of matrix $\boldsymbol{Q}$ as the maximum absolute column sum

$$\| \boldsymbol{Q} \|_1 := \max_j \{ \| \boldsymbol{q}_j \|_1 \} \qquad (4)$$

The 1-norm $||\boldsymbol{Q}||_1$ builds internally in a first step all column sums of the *synaptics*, which are the sums of post synaptic effects of each synaptic region (or distal dendritic segment), and picks in a second step the maximum of these values. In the next section, where we present the HTM cell algorithm, we will see that excited cells get predictive whenever the 1-norm of their *synaptics* $\boldsymbol{Q}$ exceeds a threshold θ.

$$\| \boldsymbol{Q} \|_1 \geq \theta \qquad (5)$$

Equation (5) means that at least one of the synaptic regions (distal dendritic segments) "sees" sufficient post-synaptic activities, i.e., can sum up the number of post-synaptic effects to a value exceeding threshold θ.

Example 2

Consider the following *synaptics* of a neuron with threshold $\theta = 2$.

$$\boldsymbol{Q} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The post synaptic effects for synaptic fields 1,2,3 are in vector-representation $\boldsymbol{q}_1 = [0\ 1\ 1\ 0\ 0]^T$, $\boldsymbol{q}_2 = [1\ 1\ 0\ 0\ 0]^T$, $\boldsymbol{q}_3 = [0\ 0\ 1\ 0\ 0]^T$. These model post-synaptic activity in the distal dendritic segments 1,2,3. The related 1-norms of the $\boldsymbol{q}_j$-vectors are:

$$\| \boldsymbol{q}_1 \|_1 = 2, \qquad \| \boldsymbol{q}_2 \|_1 = 3, \qquad \| \boldsymbol{q}_3 \|_1 = 2$$

The 1-norm of the synaptics is

$$\| \boldsymbol{Q} \|_1 = \max\{ \| \boldsymbol{q}_1 \|_1, \| \boldsymbol{q}_2 \|_1, \| \boldsymbol{q}_3 \|_1 \} = \max\{2,3,2\} = 3$$

Since with $\theta = 2$ we have $\| \boldsymbol{Q} \|_1 \geq \theta$, the neuron will enter a predictive state.

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 7.

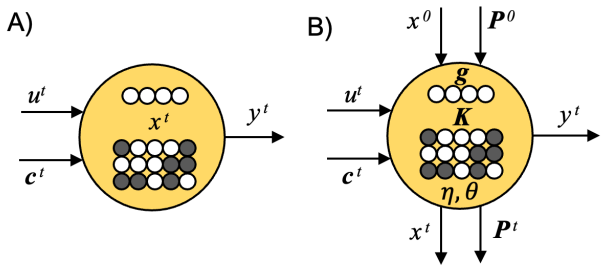

Figure 7: input/output model of a cell (HTM neuron)

Figure 7A shows a simplified version of the cell with many quantities omitted, while figure 7B shows all relevant quantities related to the formal cell model. All quantities shown in figure 7 are related to a discrete time index t ($t = 0, 1, 2, \ldots$).

Let us start with a signal description of figure 7A. It shows

- the (scalar) binary *feedforward input* $u^t \in \mathcal{B}$
- the (vectorial) binary *context input* $\boldsymbol{c}^t \in \mathcal{B}^L$.
- the (scalar) binary cell output $y^t \in \mathcal{B}$.
- the (scalar) binary (internal) cell state $x^t \in \mathcal{B}$,

The graphical symbol (see also figure 7B) suggests that the cell entertains a synaptic field, which is assigned with index vector $\boldsymbol{g} \in (1,2,\ldots,N)^m$ *to* select the outputs from context $\boldsymbol{c}$ of the neighbor cells, assigned with constant permanences equal to one.

In addition, the cell entertains a synaptic bank with $s \times d$ synapses, with related index matrix $\boldsymbol{K}$ containing the indices of the context vector elements, which connect to synapses of the synaptic bank, controlled by the time varying permanence matrix $\boldsymbol{P}^t \in [0,1]^{s \times d}$.

Since the tuple $(x^t, \boldsymbol{P}^t)$ represents the total state of the cell, which is sequentially updated at each time step t, the formal model needs to provide

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

which is sketched in figure 7B by the two input signals at the top.

HTM Cell Algorithm

We have now all pre-requisites for constructing a formal model of the HTM neuron (cell). 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 ($y^t = 1$)

Let us introduce the HTM cell algorithm first with its verbal form:

configure and initialize the cell
for each time step $t = 0, 1, 2, \ldots$

- *excited predictive* cells become *active*
- a cell recognizes its group as *bursting,* if it does not contain *excited predictive* cells
- *excited* cells in a *bursting group* become *active*
- a cell transitions into a *predictive* cell iff the norm of the current *synaptics* exceeds a threshold
- all synapses of *activated* cells learn (change permanences), all synapses of non-excited cells do not
- a synapse of an *activated* cell is *reinforced*, if the connected cell *fires through it,* otherwise it is *penalized.*

Before we dive into the formal details of the HTM cell algorithm, let us present the complete algorithm in terms of a signal processing flow chart (figure 8).

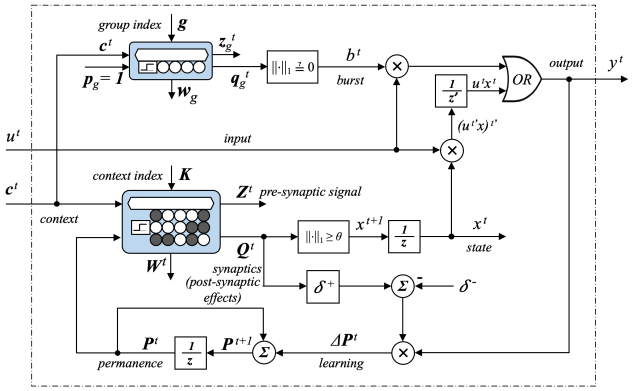

Figure 8: signal processing flow of HTM cell algorithm

If this diagram seems overwhelming, the reader is advised at this point to pay attention to a few high level aspects:

- The whole cell algorithm is local, as the diagram shows. Locality implies that a single cell can access numerous output signals of other cells (which in neurobiology are carried via axons of surrounding neurons to our neuron's synapses), but the cell cannot directly access the (predictive) states of other neurons (which in neurobiology would require an additional kind of axons). The solution is a time multiplexing mode, where sudden raise of a neighbor cell's output is reflecting the predictive state of the neighbor cell, while a delayed raise is due to the excitation of the neighbor cell from the input side.
- There is a *synaptic field* selecting the outputs of the group partner cells, to construct the so called *burst* signal b^t. The permanences $\boldsymbol{p}_g$ of this synaptic field are constantly set to one, thus, not changing over time.
- There is also a synaptic bank controlled by the time varying permanence matrix $\boldsymbol{P}^t$, which outputs the *synaptics* matrix $\boldsymbol{Q}^t$ as an internal quantity, which has major influence of *cell state transition* and *learning* (permanence matrix adaption).

The remaining detailes will be covered later. Let us investigate now the details of the formal HTM cell algorithm, starting with:

A) Configuration and Initialization phase

- Vector $\boldsymbol{g}$ is configured with the indices of all cells, with which the cell collaborates inside of a group (minicolumn).
- Matrix $\boldsymbol{K}$ is configured with the indices of the context vector $\boldsymbol{c}$ to tell the cell, which signals of the context vector connect to corresponding synapses of the cell's synaptic bank.
- State x^0 is initialized as zero.
- The permanence matrix $\boldsymbol{P}^0$ is initialized with normal distributed random numbers around mean value η.

B) Iteration Phase

Step1: input u^t

Step 2: calculate output y^t from state x^t and input u^t according to the following rule: *excited predictive cells become active.*

$$y^{t'} = x^t \cdot u^t \tag{6}$$

Step 3: update y^t in case of a *burst* condition, which is given, when the *group contains no excited predictive cells*. This requires that all other cells have already executed before execution step 2, before (outside the cell algorithm) $\boldsymbol{c}^{t'}$ is updated with cell outputs relating to progressed time t'. Otherwise, the cell has no possibility to "see" whether there are predictive cells in the group.

$$y^t = y^{t'} \vee (\|\boldsymbol{c}^{t'}(\boldsymbol{g})\|_1 = 0) \tag{7}$$

Step 4: calculate the cell's *synaptics* $\boldsymbol{Q}^t$

$$\boldsymbol{W}^t = (\boldsymbol{P}^t \geq \eta) \quad \text{weight matrix} \tag{8a}$$
$$\boldsymbol{Z}^t = \boldsymbol{c}^t(\boldsymbol{K}) \quad \text{pre-synaptic signals} \tag{8b}$$
$$\boldsymbol{Q}^t = \boldsymbol{S}^t \circ \boldsymbol{Z}^t \quad \text{synaptics} \tag{8c}$$

Step 5: perform a state transition, i.e., calculate x^{t+1} from the *synaptics* $\boldsymbol{Q}^t$

$$x^{t+1} = (\|\boldsymbol{Q}^t\|_1 \geq \Theta) \quad \text{state transition} \tag{9}$$

Step 6: Hebbian like learning; synapses of *active* cells with corresponding non-zero synaptics' elements are *reinforced* (increase their *permanences*), all other synapses of *active* cells are *penalized* (decrease their *permanences*), while inactive synapses stay unaffected (maintain their *permanences*). This can be formalized by equations (10a-b)

$$\Delta\boldsymbol{P}^t = (\delta^+\boldsymbol{Q}^t - \delta^-)y^t \qquad (\delta^+ > \delta^-) \tag{10a}$$
$$\boldsymbol{P}^{t+1} = \boldsymbol{P}^t + \Delta\boldsymbol{P}^t \tag{10b}$$

Example

... to be done ...

Text after here requires rework!

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 9), resulting in a layer of total $N := mn$ cells.

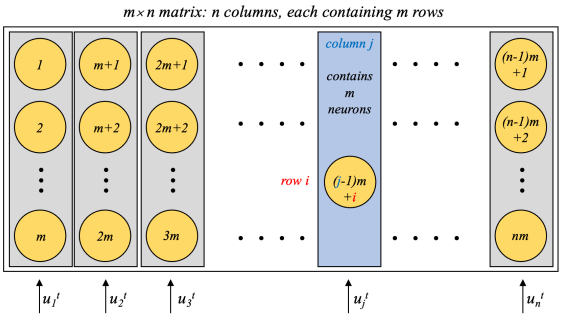

Figure 9: HTM layer with a matrix of *m n* neurons

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*

$$\boldsymbol{u}^t = [\, u_1^t,\, u_1^t,\, \ldots\, ,\, u_n^t\,]^T \in \{0,1\}^n\,. \qquad (11)$$

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 $\boldsymbol{D}^t$.

$$\boldsymbol{x}^t = [\, x_1^t,\, x_2^t,\, \ldots\, ,\, x_N^t\,]^T \in \{0,1\}^N \qquad (12a)$$
$$\boldsymbol{y}^t = [\, y_1^t,\, y_2^t,\, \ldots\, ,\, y_N^t\,]^T \in \{0,1\}^N \qquad (12b)$$
$$\boldsymbol{P}^t = [\, \boldsymbol{p}_1^t,\, \boldsymbol{p}_2^t,\, \ldots\, ,\, \boldsymbol{p}_N^t\,] \in [0,1]^{M \times N} \qquad (12c)$$

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 10.

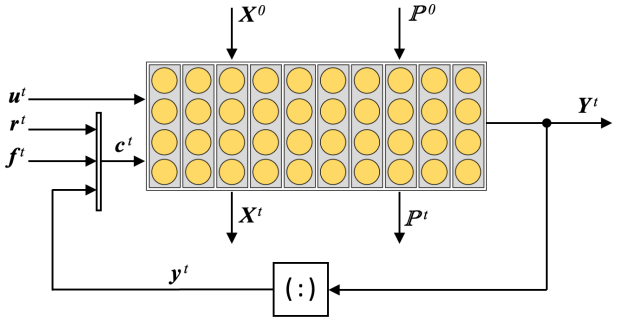

Figure 10: input/output model of cell layer with contextual feedback loop

Input Sequences

Before studying this algorithm in an example, we need another pre-requisite: covering a formal description of input sequences. Let us consider two sentences "Mary likes to sing." and "John likes to dance." , for which 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 $\boldsymbol{u}^0$, $\boldsymbol{u}^1$, $\boldsymbol{u}^2$, ... as *tokens*. These 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

Let us define a 40 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

$$\begin{aligned}
&\text{'Mary'} \;\mapsto [0\,0\,1\,0\,0\,1\,1\,1\,0\,0]^T\\
&\text{'John'} \;\mapsto [1\,0\,0\,1\,0\,1\,1\,0\,0\,0]^T\\
&\text{'likes'} \;\mapsto [0\,1\,0\,0\,0\,0\,0\,1\,1\,1]^T\\
&\text{'to'} \;\mapsto [0\,1\,0\,1\,1\,0\,0\,0\,0\,1]^T\\
&\text{'sing'} \;\mapsto [1\,1\,0\,0\,1\,0\,1\,0\,0\,0]^T\\
&\text{'dance'} \;\mapsto [1\,0\,0\,1\,1\,0\,0\,0\,0\,1]^T
\end{aligned}$$

In this sense the tokenizer maps the sequence

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

to the input token sequence

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

and sequence

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

To the input token sequence

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

The HTM Algorithm

We are ready now to formulate the coarse form of the HTM algorithm for sequence memory. The original form 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.

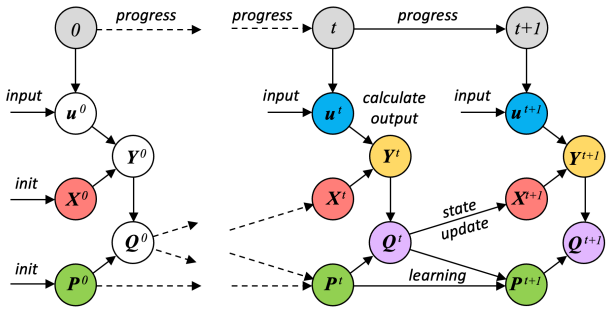

Figure 10: computational scheme for the HTM algorithm

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 $\boldsymbol{x}^0$, $\boldsymbol{P}^0$
for $t = 0, 1, 2, \ldots$:
  input $\boldsymbol{u}^t$
  calculate $\boldsymbol{y}^t$ from $\boldsymbol{x}^t$, $\boldsymbol{u}^t$ (output activation)
  calculate $\boldsymbol{Q}^t$ from $\boldsymbol{P}^t$, $\boldsymbol{y}^t$ (synaptic effects)
  update $\boldsymbol{x}^{t+1}$ from $\boldsymbol{Q}^t$ (cell state update)
  update $\boldsymbol{P}^{t+1}$ from $\boldsymbol{P}^t$, $\boldsymbol{Q}^t$ (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 it *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*.

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:

$$\boldsymbol{B}^t = \boldsymbol{1} - \boldsymbol{1}^m \cdot \boldsymbol{max}(\boldsymbol{X}^t) \quad \text{burst matrix } \boldsymbol{B}^t \in \{0,1\}^{m \times n} \quad (13a)$$
$$\boldsymbol{C}^t = \boldsymbol{X}^t + \boldsymbol{B}^t \quad \text{candidates } \boldsymbol{C}^t \in \{0,1\}^{m \times n} \quad (13b)$$
$$\boldsymbol{Y}^t = \boldsymbol{C}^t \circ (\boldsymbol{1}^m \cdot \boldsymbol{u}^{tT}) \quad \text{output activation} \quad (13c)$$

Even this kind of formulation looks a bit unusual, the chosen form has the advantage of generating a straightforward 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 (14) we clearly see the dependency of $\boldsymbol{y}^t$ from $\boldsymbol{x}^t$ and $\boldsymbol{u}^t$.

$$\boldsymbol{Y}^t = (\boldsymbol{X}^t + (\boldsymbol{1} - \boldsymbol{1}^m \cdot \boldsymbol{max}(\boldsymbol{X}^t)) \circ (\boldsymbol{1}^m \cdot \boldsymbol{u}^{tT}) \quad (14)$$

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.

$$\boldsymbol{W}^t = \sigma(\boldsymbol{D}^t - \eta) \quad \text{weight matrix} \quad (15a)$$
$$\boldsymbol{Z}^t = \boldsymbol{c}^t(\boldsymbol{K}) \quad \text{pre-synaptic signals} \quad (15b)$$
$$\boldsymbol{Q}^t = \boldsymbol{S}^t \circ \boldsymbol{Z}^t \quad \text{synaptics} \quad (15c)$$

Again (15a-c) can be consolidated to a single operation:

$$\boldsymbol{Q}^t = \sigma(\boldsymbol{D}^t - \eta) \circ \boldsymbol{y}^t(\boldsymbol{K}) \quad (16)$$

Each Boolean element of the *synaptic matrix* $\boldsymbol{S}^t$ in (15a) tells us, whether the related synapse is enabled or disabled, depending on whether its permanence value exceeds threshold η or not. In (15b) the layer output $\boldsymbol{y}^t$ is projected into a subspace to yield $\boldsymbol{Z}^t$ according to the index values of $\mathbf{K}$ 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* $\boldsymbol{Q}^t$.

$$\boldsymbol{p}^t = [\,||\,\boldsymbol{Q}_1{}^t||_1, ||\,\boldsymbol{Q}_2{}^t||_1, ..., ||\,\boldsymbol{Q}_n{}^t||_1]^T \in \{0 \dots s\}^n \quad (17a)$$
$$\boldsymbol{x}^{t+1} = (\boldsymbol{p}^t \geq \theta) \quad (17b)$$

Equation (17) means that the given *coincidence matrix* $\boldsymbol{Q}^t$ has to be decomposed according to (???) into its parts $\boldsymbol{Q}_k{}^t$. Then, according to (6), for each $\boldsymbol{Q}_k{}^t$ we calculate the coincidence number

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

as the largest column sum, and form vector

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

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 $\boldsymbol{Q}^t$

$$\boldsymbol{X}^{t+1} = ([||\boldsymbol{Q}_1{}^t||_1, ||\boldsymbol{Q}_2{}^t||_1, \dots, ||\boldsymbol{Q}_n{}^t||_1]^T \geq \theta)(m \times n) \quad (18)$$

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

$$\Delta\boldsymbol{D}^t = (\delta^+\boldsymbol{Q}^t - \delta^-) \circ \boldsymbol{Y}^t \quad (19a)$$
$$\boldsymbol{D}^{t+1} = \boldsymbol{D}^t + \Delta\boldsymbol{D}^t \quad (19b)$$

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

$$\boldsymbol{D}^{t+1} = \boldsymbol{D}^t + (\delta^+\boldsymbol{Q}^t - \delta^-) \circ \boldsymbol{Y}^t \quad (20)$$

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 $\boldsymbol{u}^t$
  $\boldsymbol{Y}^t = (\boldsymbol{X}^t + (1 - \boldsymbol{1}^m \cdot \boldsymbol{max}(\boldsymbol{X}^t)) \circ (\boldsymbol{1}^m \cdot \boldsymbol{u}^{tT})$
  $\boldsymbol{Q}^t = \sigma(\boldsymbol{D}^t - \eta) \circ \boldsymbol{y}^t(\boldsymbol{K})$
  $\boldsymbol{X}^{t+1} = ([||\boldsymbol{Q}_1{}^t||_1, ||\boldsymbol{Q}_2{}^t||_1, \dots, ||\boldsymbol{Q}_n{}^t||_1]^T \geq \theta)(m \times n)$
  $\boldsymbol{D}^{t+1} = \boldsymbol{D}^t + (\delta^+\boldsymbol{Q}^t - \delta^-) \circ \boldsymbol{Y}^t$

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*.

[8] Chklovskii et al.: Cortical rewiring and information storage. *Nature* 431, 782–788. doi: 10.1038/nature03012 (2004).

[9] Stuart, G.J., Häusser, M.:. Dendritic coincidence detection of EPSPs and action potentials. *Nat. Neurosci.* 4, 63–71. doi: 10.1038/82910 (2001)

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