Understanding HTM Sequence Memory

Hugo Pristauz, Walter Eder – Oct. 2023

Abstract

This paper is being intended as an aid for people who want to get detailed understanding of Numenta's HTM (Hierarchical Temporal Memory) sequence memory. Sequence memory is capable of learning sequences, and to predict the next item when part of a sequence is been presented to the sequence memory.

In this paper a mathematical model of the HTM sequence memory algorithm is presented, which complies to the standard representation of nonlinear discrete time state representation, as it is commonly used in system and control theory. Further a toy network comprising a total of 40 neurons is presented, which allows to investigate and study the HTM sequence memory algorithm in every detail. Further the algorithm is completely presented in MATLAB code, in order to support a jump start into a playground for HTM sequence memory.

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 definition of intelligence for beings (humans, animals, machines) by the capability of 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].

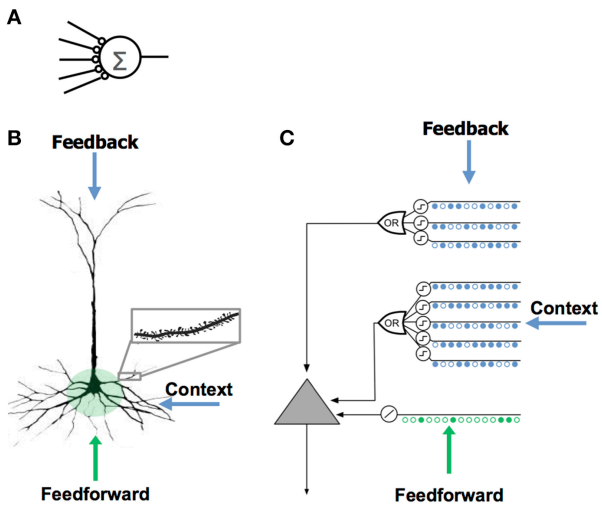

Figure 2: Comparison of neuron models

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 2/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 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 2/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) deal additionally with context information arriving at distal synapses of several dendritic segments which cause a state change of the neural model, which influences the neuron's behavior in the near future.

Mathematical Model of the 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.

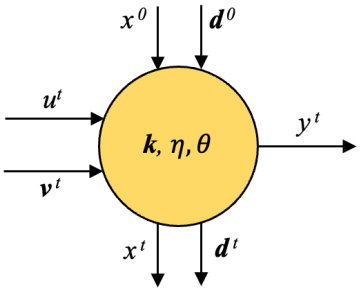

Figure 3: input/output model of a cell (HTM neuron)

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* $\boldsymbol{v}^t \in \{0,1\}^N$.

Further has the HTM neuron a state comprising

- the (scalar) binary cell state $x^t \in \{0,1\}$
- the (vectorial) dendritic weights $\boldsymbol{d}^t \in [0,1]^M$

with [0,1] denoting the closed interval $\{x \in \mathbb{R} \mid 0 \le x \le 1\}$. Since the tuple $(x^t, \boldsymbol{d}^t)$ 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 dendritic weight vector $\boldsymbol{d}^0 \in [0,1]^M$

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 $\boldsymbol{k} \in (1,2,\ldots,N)$ and threshold values $\eta \in [0,1]$ and $\Theta \in \{0,1,\ldots,N\}$, which will be explained in more detail later.

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.

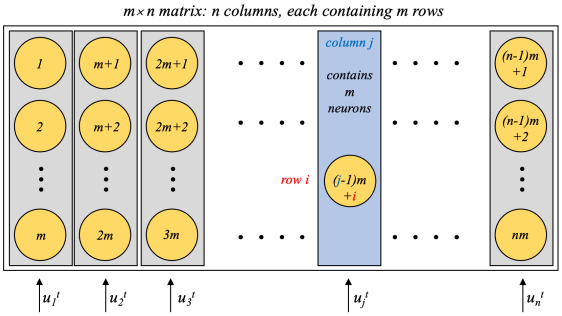

Figure 4: HTM layer with a matrix of *m n* neurons

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*

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

(with T denoting matrix transpose). 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.

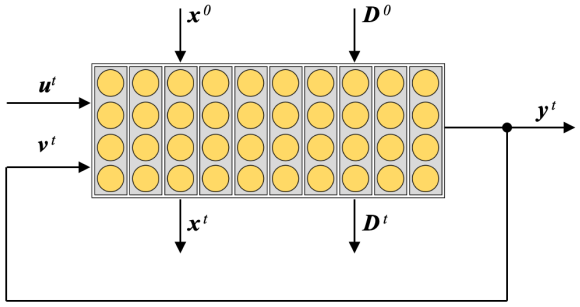

Figure 5: input/output model of cell layer with contextual feedback loop

Input Sequences

Let us now talk in more detail about how input sequences are presented to the cell layer of figure 5. Let's 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 particular 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

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

and sequence

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

To the input token sequence

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

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 ($s=5$).

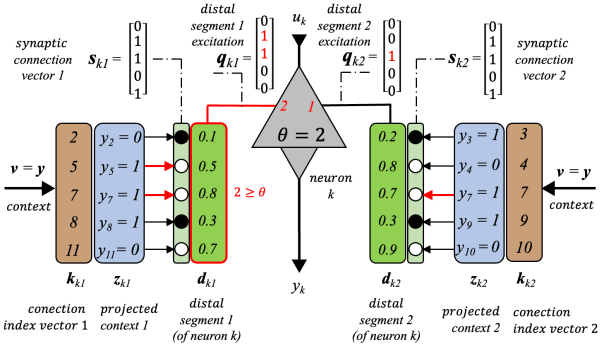

Figure 5: neuron with *2* distal dendritic segments and *5* synapses per segment (s = 5, d = 2, time index has been omitted)

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* $\boldsymbol{d}_k{}^t$ 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 $\boldsymbol{d}_{k1}{}^t$, $\boldsymbol{d}_{k2}{}^t$, ..., $\boldsymbol{d}_{kd}{}^t$ according to

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

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

Referring to figure 5 we have for example

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

with partial vectors

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

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 $\boldsymbol{z}_{k\mu}{}^t$, 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

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

then we get

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

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 vectors

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

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

Based on the concept of the projected context $z_{k\mu}$ and the synaptic vector $\boldsymbol{s}_{k\mu}$ we can define now the *segment excitation* for distal dendritic segment μ

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

with $\circ$ denoting element wise multiplication. If an element of the segment excitation vector 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 (contextual) cell output to the related 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 $\{\boldsymbol{q}_{k1}, \boldsymbol{q}_{k2}, \ldots, \boldsymbol{q}_{kd}\}$, which can be summarized as the segment activation matrix of cell k

$$\boldsymbol{Q}_k{}^t = [\boldsymbol{q}_{k1}{}^t, \boldsymbol{q}_{k2}{}^t, \ldots, \boldsymbol{q}_{kd}{}^t], \quad \boldsymbol{Q}_k{}^t \in \{0,1\}^{s \times d} \quad (6)$$

will play a key role for both state activation of a cell and reinforced learning, i.e., adaption of the synaptic permanence values.

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 $\eta = 0.5$.

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

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

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

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

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

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