Hierarchical Temporal Memories prediction performance and robustness to faults on multivariate time series

Mathieu Jégou, Pierre Chevaillier, Pierre de Loor

Hierarchical Temporal Memories prediction performance and robustness to faults on multivariate time series

Mathieu Jégou
Lab-STICC, UMR CNRS 6285
ENIB
Brest, France
jegou@enib.fr

Pierre Chevaillier
Lab-STICC, UMR CNRS 6285
ENIB
Brest, France
chevaillier@enib.fr

Pierre De Loor
Lab-STICC, UMR CNRS 6285
ENIB
Brest, France
deloor@enib.fr

***Abstract*—In this article, we evaluate the ability of Hierarchical Temporal Memories (HTM) to process values coming from sensor chains. We present a study on the impact of the HTM parameterization on its ability to predict input values and its robustness to sensor faults. The HTM is evaluated on simulated multivariate time series comprising several causal relations between variables. The results show the ability of HTM to predict future values of multivariate time series and to be robust to sensor faults. We then present which parameters most impact HTM prediction performance and its robustness to faults.**

***Keywords*-Multivariate Time Series; Prediction; Fault Robustness; Hierachical Temporal Memory**

I. Introduction

Machine learning algorithms have been extensively used to predict future values of data coming from sensor chains [10]. Robustness to faults is a critical issue in order to apply these algorithms to real world applications [10]. In these cases, data coming from one or several sensors may be abnormal. Several types of sensor faults exist [10] [1]. We will treat here the case of one or several missing sensor values. In this case, the machine learning algorithm should rely on the values of other sensors to infer the right values of faulty sensors. Several approaches have applied Bayesian networks to the automatic sensor fault detection and their recovery [10] in the case of sensor networks. These approaches divide the process into two steps. First, a fault detection algorithm is used to detect potential faults in the inputs, second, the values of the other sensors are used to guess the normal value of the sensor. For sensor fault recovery, these approaches mostly rely on hand-crafted rules to recover sensor faults [10]. Using a machine learning algorithm could avoid this specification by automatically learn to adapt to faults by exploiting the values of the other sensors. Hierarchical Temporal Memories (HTM) could solve this problem. Indeed, the unsupervised nature of HTM makes it able to automatically discover relationships between variables. Moreover, compared to batch learning models, the online nature of the HTM learning algorithm makes it able to continuously learn the relationship between variables without needing to train it again.
These two characteristics can be useful in a lot of situations where a long-term adaptation to the environment is needed while still being able to be robust to sensor faults, such as the case of unmanned aerial vehicles which need to explore different kinds of environment [7]. However, HTM also contains numerous parameters for which very little is known, because very few studies have formally analyzed the effect of the parameters on HTM behavior [9]. In addition, little is known about the predictive capabilities of HTM on multivariate data. This study aimed at answering two questions. First, what parameters most impact the prediction abilities of HTM? We answer this question by evaluating the number of time steps necessary for HTM to learn to predict the input sequence values. Second, what parameters most influence robustness to sensor faults? To answer this question, we compare the difference between the prediction of HTM when faults are inserted in the sequence to the prediction and when no fault is inserted in the dataset. In the following sections, we will present a state-of-the-art on prediction of multivariate time series and introduce HTM. We will then present the methodology used to perform our study, the results of our study and a discussion.

II. Background and related work

A. Related work

Multivariate forecasting of sensor data consists in predicting future values in time series comprising several and potentially correlated values coming from sensors. Multivariate forecasting has been extensively studied over the last decades. Models used for multivariate forecasting encompass statistical models like the Vector Autoregression Model, the multivariate bilinear model, and machine learning models like Bayesian Networks, Hidden Markov Models (HMM), Long Short Term Memory (LSTM) and Online-Sequential Extreme Learning Machine (OS-ELM). Most of these approaches are supervised and necessitates an offline learning [5].
An important property of a multivariate forecasting algorithm is its capacity to be robust to sensor faults. Several

types of sensor faults exist [1] [10]. We chose to study the robustness of HTM to data loss. In this case, one sensor is unable to deliver values during a period of time. In the case of data loss, a robust model should be able to use the values of other sensors to guess the values the faulty sensors should deliver.

HTM have been proposed by [6] as an alternative to existing machine learning models. HTM is a neuro-inspired, unsupervised and online machine learning algorithm dedicated to prediction and anomaly detection in time series. Several articles compared the performance of HTM to other machine learning techniques. [2] showed that HTM have better prediction performances than LSTM and OS-ELM on the NYC taxi passengers dataset, a dataset representing the evolution of the number of taxi passengers every 30 min in New-York during a year. Contrarily, [4] showed that, on the same dataset, LSTM had better prediction performances than HTM. The difference compared to the study of [2] lies in the implementation used for LSTM and its training method. However, they also showed that HTM was able to adapt to non-stationary time series while LSTM was not able to adapt to the changes without going through a training phase again. Finally, they found that, for this dataset, HTM and LSTM have similar performance than a multilayer perceptron. [3] compared the prediction performance of HTM to the prediction performance of a Hidden Markov Model (HMM). They showed that HTM performed better than a first-order HMM on a dataset of images representing letters. In their setup, the internal representation of data made by HTM was used by HMM to predict the next label. However, they also showed that the prediction performances of HMM was better when random noise was added to the images.

These studies show that HTM do not outperform current machine learning techniques in pure prediction performance. However, by its unsupervised and online nature, HTM is able to adapt to changes in the sequences of data while such changes require a new learning step in other algorithms.

However, the effect of HTM parameterization on its prediction performance is mostly unknown. Except works by [8] and [9] that studied the effect of parameterization on the Spatial Pooler ability to discriminate inputs, no study has analyzed the effect of parameterization on HTM ability to predict values in the input sequences. Moreover, the behavior of HTM on multivariate datasets with more than two variables is unknown.

B. HTM structure and operation

The role of HTM is to memorize sequences of data and to predict the next element of a sequence. HTM takes as an input a binary array representing the current element of the sequence to predict. Internally, HTM is composed of a set of interconnected columns of cells that are connected to the input. For the remainder of the article, N is the number of columns and N_c the number of cells within a column. As an output, HTM activates or deactivates cells. The set of activated cells represents both the internal representation of the current element of the sequence and the temporal context, that is, the previous values of the sequence. It also produces a second set of predicted cells that represents the expected value at the next time step. Figure 1 illustrates the operation of HTM.

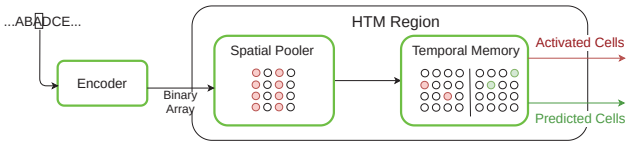

Figure 1. Illustration of HTM operation

1) Encoder: HTM encodes the current label of the sequence to a binary array representing the data. In this binary array, each bit has an equal weight relative to the others. Moreover, each value representing the same type has the same total number of bits and a fixed number of active bits. The number of active bits is called the population of the array. For the remainder of the article, we will call the total number of bits the dimension (N_i) and the number of bits with the value 1 the population w. The sparsity s is then defined as $\frac{w}{N_i}$. The way data are encoded depends on the encoder used.

2) Spatial Pooler: The Spatial Pooler takes as an input the binary array produced by the encoder and produces as output an internal representation of the input by activating columns of neurons. Moreover, the Spatial Pooler ensures that the ratio between the number of active columns and the total number of columns (i.e. the sparsity) is constant for each input. In the Spatial Pooler, each column of cells is connected to a subset of the input by a single proximal segment. The column can only have connections with the inputs when distance is less than the connection radius γ. Moreover, the segments have connections only with a subset of the input bits within γ. The ratio of input bits to which the segment can be connected to within the connection radius γ is defined by the parameter r_i. The connections between the inputs and the column can be potential or real. For real connections, the active state of a bit is transmitted to the column while it is not the case for a potential connection. Based on the connections between the column and the inputs, an activation value is computed at each time step. The activation value is proportional to the number of real connections corresponding to active inputs. More precisely the activation is computed by the equation 1.

$$a_i = b_i \sum_j W_{ij} z_j \tag{1}$$

In equation 1, a_i corresponds to the activation value of the column, W_{ij} equals to 1 if the corresponding connection is real, and z_j is the state of the bit corresponding to the connection. b_i is a boosting parameter associated with the column. The column is activated when this activation value is greater than a threshold θ_{stim}.

Next, an inhibition step ensures that only a fixed ratio r_c of columns remains activated. The inhibition can be global or local. When the inhibition is global, the inhibition is done on the entire region. When the inhibition is local, the inhibition is done on subsets of columns. In the case of local inhibition, the inhibition radius is computed by the following formula $\phi = \gamma \times N_{avg}$, N_{avg} being the mean number of columns connected to a single input bit.

The learning step consists in modifying the nature of the connections between the column and its inputs. While a column can't have connections with an input that does not initially have a connection with this column, a potential connection can become a real connection and conversely. Each connection is associated with a permanence value p. A connection is real if and only if this permanence value is greater than a threshold θ_p and potential if the connection is less than this threshold. Then, at each time step, for each activated column after the inhibition process, the permanences of the connections are modified such that the connections to active bits are increased by a value p^+ and connections to inactive bits are decreased by a value p^-. At the end of the Spatial Pooling process, the boosting parameter is adjusted in order to maximize the number of columns that contribute to the representation of input data. The boosting mechanism favorizes the activation of columns that have an activation frequency below a level, and disadvantages columns that have an activation frequency above this level. The boosting parameter is decreased or increased based on the equation 2.

$$b_i = e^{\beta(t_a - \bar{a_i})} \tag{2}$$

$\bar{a_i}$ represents the ratio of time a column has been activated over the T_b last time steps, t_a represents the desired activity ratio for T_b time steps. The boosting parameter is then less than 1 when the mean activity of a column is greater than the targeted activity and greater than 1 when the mean activity of the column is less than the targeted activity.

3) Temporal Memory: The Temporal Memory determines the activated and predicted cells based on the activated columns determined by the Spatial Pooler and the set of predicted cells determined by the Temporal Memory at the previous time step.

The Temporal Memory algorithm selects the cells being activated within a column by looking if cells have already been predicted. If the column contain cells that have been predicted, these cells remain activated and deactivate the others. Otherwise, all the cells of the column remain activated.

The next step consists in determining the predicted cells based on the active cells. For that, each cell has a maximum number of max_{seg} of distal segments, these distal segments being connected to the other cells by a set of synapses. Thus, N_{syn} is the number of synapse that cannot be greater than max_{syn}. Following the same principle as the Spatial Pooler, these connections are either potential or real. The distal segment is then activated when the number of active cells having a real connection with the distal segment is greater than a threshold θ_d. Then, if a cell has at least one activated distal segment, it becomes predicted.

Next, the Temporal Memory has a learning algorithm that is similar to the learning algorithm of the Spatial Pooler. Each synaptic connection has a permanence value p_t. When a cell has been predicted at time step $t-1$ and activated at time step t (meaning it has been correctly predicted), the synaptic connections are reinforced such that the permanence values of the connections corresponding to active cells at time step $t-1$ are increased by the value p_t^+ and the permanence values of the connections corresponding to inactive cells are decreased by the value p_t^-. When the permanence is greater than the permanence threshold θ_{p_t} the connection is considered real and if the permanence is less than this value, the connection is considered potential. Moreover, when a predicted neuron has not been activated at the next time step, the permanence value is decreased by p_t^{--}. At the beginning of the simulation, the permanence values of the Temporal Memory are initialized with the value p_i.

Finally, it is likely that, in the first time steps, no cells are predicted. In an actived column having no predicted cells, the segment having the greater activation is reinforced by increasing by p_t^+ the permanence values of its connections if this segment has an activity greater than a threshold m.

III. Method

In this section we present the methodology used to conduct our experiment. First, we present the way we generated the signal. Next, we describe the simulations we did and how we measured HTM performance on the prediction and fault robustness test. Finally, we present the parameters we used.

A. Sequence generation and encoding

The sequence generation step is in charge of determining the sequence of labels used in the study and encoding it. The sequence generated has five different categorical variables V_0, V_1, V_2, V_3, V_4. Each of these variables has three possible values. Moreover, variables have causality links. V_0 and V_1 causes V_2 and V_4 causes V_3. The values of V_2 and V_3 variables at time step $t+1$ are then entirely determined by the values of the cause variables (V_0 and V_1 for V_2 and V_4 for V_3) at the previous time step. The values of V_0, V_1 and V_4 depend only on their values on the two last time steps. This means that HTM needs to keep in

memory the last two values in order to predict the next value. The resulting sequence is composed of a total of 400 elements. Moreover, this sequence consists in 33 repetitions of the same pattern composed of 12 elements. The encoding step consists in transforming the sequence into binary array. Each variable is encoded into a binary array composed of contiguous active bits such that each value is represented by a different set of active bits. The binary arrays produced for each variable are then concatenated into one single binary array. The binary arrays representing each variable have a population $w = 43$ and a dimension $N_i = 215$. As a result, after the concatenation of each binary array, the output of the encoder has a dimension $N_i = 1075$ and a population $w = 215$.

B. Simulation and performance measurements

We ran two different simulations. The first simulation consisted in learning to predict the normal sequence of data. In this simulation, learning was activated. At each time step, the exactness of the prediction was computed by comparing the predicted cells with the active cells at the next time step. The comparison was done by the prediction score presented in equation 3. In this equation, π_{i-1} is the set of predicted cells at time step $i-1$ and a_i is the set of active cells at time step i. $dim(a_i)$ is the number of active cells and $dim(\pi_{i-1})$ is the number of predicted neurons. $\pi_{i-1} \cdot a_i$ is a scalar product that gives the total number of shared cells between the set of predicted neurons and the set of active neurons. The result is a value that represents how exactly HTM was able to predict the cells that could be active at the next time step. The value of the prediction score was then decreased when HTM predicted cells that were not activated at the next time step and did not predicted cells that were activated at the next time step.

$$r_i = \frac{2 \times \pi_{i-1} \cdot a_i}{dim(a_i) + dim(\pi_{i-1})} \tag{3}$$

At the end of the simulation, we applied a weight to the prediction score for each time step by a coefficient computed from the mean number of shared columns m_{sc} between the different representations of the input. This weight was computed according to equation 4.

$$v = \frac{1}{1 + e^{-(m_{sc} - 0.8)}} \tag{4}$$

This formula means that we strongly decrease the values of the prediction score when the mean number of shared columns between the representations is greater than 80%. Sharing more than 80% of columns between representations means that different values of the input are represented by the same columns. In this case, this means that the Spatial Pooler did not correctly learn to represent the data. Finally, we compute a learning speed metric. This metric corresponds to the time step where the prediction score definitely becomes greater than a threshold value of 0.8. We retained only simulations where the prediction score was greater than 0.8 for at least five iterations. Otherwise, we set the learning speed metrics to -1.

During the second simulation, we evaluated the robustness of HTM to sensor faults. For that, we replaced the values of the effect variables (V_2 and V_3) by a symbol indicating a fault between time step 50 and time step 100. We then evaluated the ability of HTM to continue to predict the normal values of V_2 and V_3. For that, we compute a robustness score between what the model predicted at time step t-1 and the value that should normally be present without sensor faults as shown in equation 5. This equation is similar to equation 3 except that the scalar product is done between the set of predicted neurons at time step $t-1$ π^f_{i-1} and the last set of active cells representing input at time step t without faults. In this simulation, learning was deactivated.

$$fr_i = \frac{2 \times \pi^f_{i-1} \cdot a^n_i}{dim(a^n_i) + dim(\pi^f_{i-1})} \tag{5}$$

C. Selection of the parameters

The parameters and their values we used in our simulations are described in table I. We could not test all combinations of parameters as it would necessitate to run over 55×10^9 simulations. We then decided to randomly select the parameter values before executing the simulations.

IV. Results

We ran a total of 14101 simulations with different parameter values, 1079 succeeded in predicting the inputs. Among these 1079 simulations, 961 gave results of fault robustness greater than 0.6 as computed by equation 5. The small amount of simulations that succeeded can be explained by the fact that we chose the parameters randomly.

A. Prediction performance

1) Logistic regression: We used the learning speed metrics to measure the contribution of the parameters on HTM prediction performance. We distinguished two cases. We first analyzed which parameter combinations conducted to the learning of the sequence (learning speed greater than 0) and which parameter combination led to an inability to learn (learning speed equals to -1). We thus created a binary variable which equalled 0 when HTM did not learn and 1 when it learned to predict the inputs. We then ran a logistic regression. Before applying the logistic regression, all variables were scaled by substracting their value by their mean and dividing by their standard deviation. We then divided the data into a training set and a test set. For each model tested, we ensured that the model had the same classification performance on the train and test set, showing that it did not overfit the data. Finally, as the number of simulations that

Name	Component of HTM	Description	Range of values tested
N	Spatial Pooler	Number of columns	128,256,512,1024
r_γ	Spatial Pooler	Ratio between the connexion radius γ and the size of the inputs N_i $r_\gamma = \frac{\gamma}{N_i}$	0.0,0.01,0.1,0.3,0.5,0.8,1.0
r_i	Spatial Pooler	Ratio of inputs having a connection with the column	0.0,0.01,0.1,0.3,0.5,0.8,1.0
g	Spatial Pooler	Whether the inhibition is local (0) or global (1)	0,1
r_c	Spatial Pooler	Ratio of columns selected by the inhibition mecanism	0.0,0.02,0.1,0.2,0.5,0.8,1.0
r_{p+}	Spatial Pooler	Ratio between the permanence increase and the connection threshold $\frac{p^+}{\theta_p}$	0.0,0.01,0.2,0.5,0.8,1.0
r_{p-}	Spatial Pooler	Ratio between the permanence decrease and the connection threshold $\frac{p^-}{\theta_p}$	0.0,0.01,0.2,0.5,0.8,1.0
r_s	Spatial Pooler	Ratio between the activation threshold of a column and the number of synapses $\frac{r_\gamma \times r_i}{\theta_{stim}}$	0.0,0.01,0.2,0.5,0.8,1.0
β	Spatial Pooler	The boosting strength	0.0,0.1,0.5,1.0,3.0
N_c	Temporal Memory	The number of cells in a column	3,6,8
r_{st}	Temporal Memory	Ratio between the maximum number of synapses and the activation threshold of a segment $\frac{max_{syn}}{\theta_d}$	0.0,0.01,0.1,0.3,0.6,0.8,1.0
r_{ic}	Temporal Memory	Ratio between the initial value of the permanences and the connection threshold $\frac{p_i}{\theta_{p_t}}$	0.2,0.5,1.0
r_m	Temporal Memory	Ratio between the minimal activation value for learning and the activation threshold of a segment $\frac{m}{\theta_d}$	0.0,0.01,0.1,0.3,0.5,0.8,1.0
max_{seg}	Temporal Memory	The maximum number of segments	4,8,20,40,128
r_{inc}	Temporal Memory	The ratio between the increase value of the permanence and the connection threshold $\frac{p_t^+}{\theta_{pred}}$	0.0,0.01,0.1,0.3,0.6,0.8,1.0
r_{dec}	Temporal Memory	The ratio between the decrease value of the permanence and the connection threshold $\frac{p_t^-}{\theta_{pred}}$	0.0,0.01,0.1,0.3,0.6,0.8,1.0
r_{pred}	Temporal Memory	The ratio between p_t^{--} and θ_{pred} $r_{pred} = \frac{p_t^{--}}{\theta_{pred}}$	0.0,0.01,0.1,0.3,0.6,1.0
max_{syn}	Temporal Memory	The maximum number of synapses a segment can have with other cells	4,8,16,32,64,128,256

Table I
HTM PARAMETERS AND THEIR VALUES USED IN THE SIMULATIONS

succeeded in predicting the inputs represented only 7 % of the data, we upsampled the training set in order to have the same number of elements in both classes. We tried several logistic regression model and selected the model that had the highest McFadden pseudo-R^2. The model selected is a first order model with interaction effects between parameters. A log-likelihood ratio test and a McFadden pseudo-R^2 measure showed that this regression model performed significantly better than a null model (McFadden: $R^2 = 0.72$, likelihood ratio test: $\chi^2 = 26034$, $p < 2.2 \times 10^-16$).
The results of the logistic regression are shown in table II. For each parameters, a Wald test is computed in order to assess the significance of each parameter. Moreover, as the number of predictors is large in the final regression model (1006), we chose to show only the 20 predictors with the highest coefficient absolute value.

The results of the logistic regression are shown in table II. Both the Spatial Pooler and the Temporal Memory parameters have an impact on the input learning.
Several main effect were found. The ratio of columns selected by the inhibition mechanism has the highest associated coefficient. The highest r_c the highest the probability to learn. Next, the activation threshold of the column r_s seems to decrease the learning likelihood as it grows. Similarly, the percentage of input bits a column is connected to, r_i, should remain low in order to favorize the learning. The ratio between the input radius and the input size, r_γ, has the same impact on the learning. The maximum number of synapses max_{syn} and the ratio between the permanence increase of the Temporal Memory and the connection threshold, r_{inc}, should be low in order to favorize learning. Finally, the highest the number of columns N, the highest the probability to learn.
Several interaction effects can also be observed in this table. r_i is negatively correlated with r_s and max_{syn}, the highest r_i is, the lowest r_s and max_{syn} should be. r_γ also has a negative correlation with r_s and r_i. r_c is negatively correlated with g, meaning that, when inhibition is global, r_c should remain low. Similarly a negative correlation can be observed between r_c and r_γ. The maximum number of synapses max_{syn}, the ratio between the activation threshold and the maximum number of synapses r_{st} should also be low when the inhibition is global. The boosting strength β is postively correlated with r_c. When β is low, r_c should be high and conversely. Finally, N_c has a negative correlation with max_{syn} and r_{ic}.

2) Learning speed: We selected the simulations where the learning speed was greater than 0 and ran a regression analysis to measure the effect of the parameters on the learning speed. We scaled the input parameters using the same method as the logistic regression. We tested several models and selected the model that had the best R^2. During, the

Parameter	Estimated coefficient	standard error	z-value	p-value
Intercept	-4.75	0.30	-15.84	$< 2.2 \times 10^{-16}$ ***
r_c	3.59	0.19	18.90	$< 2.2 \times 10^{-16}$ ***
r_s	-2.40	0.25	-9.72	$< 2.2 \times 10^{-16}$ ***
r_i	-2.14	0.20	-10.91	$< 2.2 \times 10^{-16}$ ***
r_γ	-2.14	0.19	-11.11	$< 2.2 \times 10^{-16}$ ***
$r_i \times r_s$	-1.86	0.19	-9.77	$< 2.2 \times 10^{-16}$ ***
$r_i \times max_{syn}$	-1.83	0.20	-9.38	$< 2.2 \times 10^{-16}$ ***
$r_\gamma \times r_s$	-1.64	0.16	-10.04	$< 2.2 \times 10^{-16}$ ***
$r_\gamma \times r_i \times r_s$	-1.45	0.08	-18.78	$< 2.2 \times 10^{-16}$ ***
$r_\gamma \times r_c$	-1.38	0.13	-10.77	$< 2.2 \times 10^{-16}$ ***
$g \times max_{syn}$	-1.34	0.21	-6.42	1.38×10^{-10} ***
max_{syn}	-1.24	0.33	-3.70	2.12×10^{-4} ***
$r_\gamma \times r_i$	-1.23	0.16	-7.73	1.07×10^{-14} ***
$g \times r_c$	-1.05	0.10	-10.07	$< 2.2 \times 10^{-16}$ ***
r_{inc}	-1.05	0.20	5.21	1.92×10^{-7} ***
$N_c \times max_{syn}$	-1.00	0.19	-5.28	1.29×10^{-7} ***
N	0.93	0.17	5.60	2.17×10^{-8} ***
$g \times r_{st}$	-0.87	0.10	-8.29	1.12×10^{-16} ***
$r_c \times \beta$	0.86	0.15	5.71	1.16×10^{-8} ***
$N_c \times r_{ic}$	-0.85	0.14	-6.14	8.18×10^{-10} ***

Table II
RESULTS OF THE LOGISTIC REGRESSION

Parameter	Estimated coefficient	Standard error	t-value	
Intercept	156.46	12.59	12.42	$< 2.2 \times 10^{-16}$ ***
β	-49.95	15.59	-3.20	1.51×10^{-3} **
max_{seg}	47.27	7.99	5.91	9.16×10^{-9} ***
$\beta \times r_{dec}$	-40.56	5.88	-6.90	3.18×10^{-11} ***
r_m	37.79	7.67	4.93	1.38×10^{-6} ***
r_{dec}	37.49	7.18	5.22	3.36×10^{-7} ***
g	-37.43	6.26	-5.98	6.32×10^{-9} ***
$r_i \times maxseg$	37.38	6.67	5.61	4.73×10^{-8} ***
$r_s \times r_{ic}$	-34.20	6.04	-5.66	3.5×10^{-8} ***
$r_\gamma \times r_{dec}$	32.80	5.80	5.65	3.69×10^{-8} ***
$max_{syn} \times r_{inc}$	31.36	5.96	5.26	2.75×10^{-7} ***
$g \times max_{seg}$	-30.79	4.71	-6.53	2.85×10^{-10} ***
$r\gamma \times r_{inc}$	29.86	5.99	4.98	1.07×10^{-6} ***
r_{p^-}	-28.87	7.47	-3.87	1.36×10^{-4} ***
$r_{st} \times r_{ic}$	28.79	6.08	4.74	3.40×10^{-6} ***
r_{ic}	28.43	11.18	2.54	1.15×10^{-2} *
$r_{inc} \times max_{seg}$	-28.31	7.17	-3.95	9.91×10^{-5} ***
β^2	-28.08	9.26	-3.03	2.63×10^{-3} **
$r_{p^+} \times max_{seg}$	-28.04	5.80	-4.84	2.13×10^{-6} ***
$r_{inc} \times r_{pred}$	27.50	5.73	4.80	2.53×10^{-6} ***

Table III
RESULTS OF THE REGRESSION ON THE LEARNING SPEED

selection process, we divided the data into a training set and a test set. Each time we tested a model, we systematically trained the model with the training set and, after, checked its prediction performance on the test set using a root mean squared error (RMSE) measure. The model learned had a residual standard error of 26.34. The prediction on the test set gave a RMSE similar to the RMSE on the training set, showing that the model did not overfit the data. Moreover the trained model had an adjusted $R^2 = 0.80$ and F-statistic showed that the model performed significantly better than a null model ($F = 7.692$, $p < 2.2e - 16$). The results are shown in table III. As the number of predictor is high (461), we chose to show only the 20 highest parameters. In this table, t-values and p-values are computed by a Student test. Several main effects where found. High boosting strength β favorizes the learning speed, as a global inhibition (g) and a high spatial pooler permanence decrease value r_{p^-}. Conversely, low minimum activation threshold r_m, low temporal memory permanence decrease value r_{dec} and low minimum activation threshold r_{ic} improve the learning speed.

Several interaction effects are also observed. β and r_{dec} are negatively correlated, as r_s and r_{ic}, r_{inc} and max_{seg}, r_{p^+} and max_{seg}. Conversely, r_i and max_{seg}, r_γ and r_{dec}, max_{syn} and r_{inc}, r_{st} and r_{ic}, r_{inc} and r_{pred} are positively

correlated.

B. Fault robustness

We ran a regression analysis in order to measure the effect of parameterization on the robustness score. The method used to run the analysis was similar as the regression analysis on the learning speed. Overall, the model selected had a residual standard error of 0.11, an adjusted $R^2 = 0.58$ and the F-statistic showed that the regression model performed better than a null model ($F = 42.83$, $p =< 2.2e-16$). The results are shown table IV. In this table, t-values and p-values are computed by a Student t-test.

The parameter that influences the most the robustness of the HTM is the inhibition parameter r_c. The higher r_c, the higher the robustness. Similarly, high values of r_{p^-} and r_s positively influence the robustness. Conversely, r_i, r_{ic}, r_{st} and r_γ should be low in order to favorize robustness.

Several interaction effects are observed. r_c and r_m, r_c and r_i, r_c and r_{st}, r_c and r_{ic}, r_γ and r_c, r_{p^-} and r_{ic} are positively correlated. Conversely, r_{st} and r_{ic}, r_i and r_m, r_{ic} and r_m, r_c and r_s, r_γ and r_m are negatively correlated.

V. Discussion

The results on the experiment show several interesting points. First, what parameters impact the most the prediction performance? According to our results, both Spatial Pooler and Temporal Memory parameters impact the learning likelihood and learning speed. For Spatial Pooler parameters, learning likelihood is favorized when each column is connected to a low number of input bits situated in its neighbourhood (r_γ and r_i are low) with an activation and an inhibition mechanism that encourage the selection of a high number of columns (low r_s and high r_c) in the Spatial Pooler. Boosting seems to favorize learning as a high boosting positively impacts both the learning likelihood and the learning speed. Global inhibition should also be preferred as it strongly increases the learning speed. However, learning likelihood tend to decrease when using global inhibiton with high r_c and high max_{syn}. This means that a tradeoff should be found between using global inhibition to increase learning speed and using high r_c and max_{syn} to favorize learning likelihood. Finally, high permanence increase and decrease values r_{p^-} and r_{p^+} tend to increase the learning speed. For Temporal Memory parameters, limiting new connections that cells can have with other cells tend to favorize both learning likelihood and speed. Indeed, the maximum number of synapses max_{syn} and the maximum number of segments max_{seg} should remain low as the permanence parameters r_{inc}, r_{dec} and r_{ic} and the number of cells per column N_c. Finally, a low minimum activation threshold for learning r_m encourages learning speed.

Second, what parameters impact the most HTM robustness to faults? For Spatial Pooler parameters, the number of inputs a column is connected to should remain low. The inhibition mechanism should encourage the activation of a low number of columns. Finally, high values of permanence decrease r_{p^-} and activation threshold r_s encourage robustness. For Temporal Memory, a low minimum activation threshold for learning r_m, a low activation threshold r_{st} and a low permanence initialization value r_{ic} postively impact robustness.

This study is a preliminary study about the effect of parameterization on the prediction and robustness to faults abilities of HTM and has several limitations. First, our study focused on only one dataset. While we hypothesize that the results is generalizable on multivariate datasets of the same type, further study should verify that the results are the same for different datasets. Such study would imply testing the HTM on datasets where causal variables are faulty, testing the HTM on datasets with circular causalities (for example $V2$ causes $V1$ that causes $V2$) and varying the number of variables. Moreover, testing HTM on continuous variables rather than categorical variables should be interesting. Finally, in our study, we considered HTM as a black box. We modified its parameters and observed the outputs without analyzing the details of HTM behavior. Observing the internal behavior of HTM could make us observe more in details how the inputs are internally represented by the Spatial Pooler and how columns and cells are interconnected.

VI. Conclusion

In this article, we presented a study on the behavior of HTM in the prediction of values from multivariate time series. We showed the influence of the parameters on the prediction ability and robustness to faults. We discovered several interesting effects of parameters on the performance of HTM performance. We plan to do further analyses to understand more in-depth how HTM behaves in the case of multivariate datasets with faults. We also plan to compare the performances of HTM with other machine learning algorithms.

Acknowledgment

This work was funded by the French Direction Générale de l'Armement (DGA) as part of the Plan d'Étude Amont "Man Machine Teaming".

References

[1] Y. Fu, C. Peng, F. Gomez, Y. Narazaki, et B. F. Spencer, "Sensor fault management techniques for wireless smart sensor networks in structural health monitoring", Struct Control Health Monit, vol. 26, no 7, p. e2362, jul. 2019.

[2] Y. Cui, C. Surpur, S. Ahmad, et J. Hawkins, "A comparative study of HTM and other neural network models for online sequence learning with streaming data", in 2016 International Joint Conference on Neural Networks (IJCNN), 2016, p. 1530–1538.

Parameter	Estimated coefficient	Standard error	t-value	p-value
(Intercept)	0.744	0.0121	61.40	$< 2.2 \times 10^{-16}$ ***
r_c	0.120	0.0093	12.87	$< 2.2 \times 10^{-16}$ ***
r_m	-0.064	0.0064	-10.08	$< 2.2 \times 10^{-16}$ ***
r_i	-0.058	0.0070	-8.25	$< 2.2 \times 10^{-16}$ ***
r_{ic}	-0.053	0.0062	-8.56	$< 2.2 \times 10^{-16}$ ***
r_{st}	-0.051	0.0062	-8.03	$< 2.2 \times 10^{-16}$ ***
r_γ	-0.029	0.0057	-5.10	4.35×10^{-7} ***
$r_c \times r_m$	0.020	0.0026	7.53	1.52×10^{-13} ***
r_{p^-}	0.020	0.0053	3.63	2.99×10^{-4} ***
r_i^2	0.018	0.0048	3.78	1.71×10^{-4} ***
r_c^2	-0.018	0.0031	-5.64	2.4410^{-8} ***
$r_i \times r_c$	0.018	0.0031	5.67	2.03×10^{-8} ***
$r_c \times r_{st}$	0.018	0.0028	6.22	8.2×10^{-10} ***
$r_{st} \times r_{ic}$	-0.017	0.0038	-4.44	1.02×10^{-5} ***
r_s	0.015	0.0052	2.91	3.68×10^{-3} **
$r_c \times r_{ic}$	0.015	0.0027	5.48	5.91×10^{-8} ***
r_γ^2	0.014	0.0043	3.25	1.2×10^{-3} **
$r_i : r_m$	-0.009	0.0036	-2.41	1.6×10^{-2} *
$r_{ic} \times r_m$	-0.009	0.0035	-2.39	1.73×10^{-2} *
$r_c \times r_s$	-0.008	0.0025	-3.23	1.2910^{-3} **
$r_{p^-} \times r_m$	0.008	0.0030	2.62	9.07×10^{-3} **
$r_\gamma \times r_c$	0.008	0.0026	2.94	3.41×10^{-3} **
$r_{p^-} \times r_{ic}$	0.007	0.0031	2.41	1.63×10^{-2} *
$r_\gamma \times r_m$	-0.007	0.0035	-2.16	3.14×10^{-2} *

Table IV
RESULTS OF THE REGRESSION ON THE FAULT ROBUSTNESS TEST

[3] D. E. Padilla, R. Brinkworth, et M. D. McDonnell, "Performance of a hierarchical temporal memory network in noisy sequence learning", in 2013 IEEE International Conference on Computational Intelligence and Cybernetics (CYBERNETICSCOM), 2013, p. 45–51.

[4] J. Struye et S. Latr, "Hierarchical temporal memory and recurrent neural networks for time series prediction: An empirical validation and reduction to multilayer perceptrons", Neurocomputing, apr. 2019.

[5] S. Makridakis, E. Spiliotis, et V. Assimakopoulos, "Statistical and Machine Learning forecasting methods: Concerns and ways forward", PLoS ONE, vol. 13, no 3, p. e0194889, mar. 2018.

[6] J. Hawkins, On intelligence, 1st ed. New York, NY: Owl Books, 2005.

[7] S. Khan, C. F. Liew, T. Yairi, et R. McWilliam, "Unsupervised anomaly detection in unmanned aerial vehicles", Applied Soft Computing, vol. 83, p. 105650, oct. 2019.

[8] M. Leake, L. Xia, K. Rocki, et W. Imaino,"A Probabilistic View of the Spatial Pooler in Hierarchical Temporal Memory", vol. 9, no 5, p. 8, 2015.

[9] J. Mnatzaganian, E. Fokou, et D. Kudithipudi, "A Mathematical Formalization of Hierarchical Temporal Memorys Spatial Pooler", Front. Robot. AI, vol. 3, 2017.

[10] F.-K. Tsai, C.-C. Chen, T.-F. Chen, et T.-J. Lin, "Sensor Abnormal Detection and Recovery Using Machine Learning for IoT Sensing Systems", in 2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA), Tokyo, Japan, 2019, p. 501505.