Information Technology Course
Module Software Engineering
by Damir Dobric / Andreas Pech

Investigation of Sequence Learning of SP/TM Layer (HTM SPARSITY)

Ghulam Mustafa
ghulam.mustafa@stud.fra-usa.de

Abdul Samad
abdul.samad@stud.fra-uas.de

Diab elmehdi
el.diab@stud.fra-uas.de

Muhammad Mubashir Ali Khan
muhammad.khan2@stud.fra-uas.de

Treesa Maria Thomas
treesa.thomas@stud.fra-uas.de

Abstract—. It is necessary for survival in natural environment to be able to identify and predict temporal sequence of sensory input. Based on numerous common properties of the cortical neurons, the theoretical framework for sequence learning in the neo cortex recently proposed hierarchical temporal memory (HTM). In this paper, we analyze the sequence learning behavior of spatial pooler and temporal memory layer in dependence on HTM Sparsity. We found the ideal value of HTM Sparsity that will have optimal learning for the given input sequence. We also showed the effect of changing Width and Input Bits on learning such that the value of HTM Sparsity remains constant. We devised a relation between HTM Sparsity and max for optimal learning of the given sequence.

Keywords— Temporal memory, Sequence learning, Spatial pooler, HTM Sparsity

I. INTRODUCTION

Hierarchical temporal memory (HTM) is a neuromorphic machine learning algorithm which resembles neocortex functions in the human brain. The HTM architecture comprises a spatial pooler (SP) and temporal memory (TM) with the sparsely, modular and hierarchically characteristically components. The SP sparsely distributes the input data while the TM is in charge of the learning process. The imitation of human brain functionality allows HTM an adaptive core algorithm for various operations such as categorization and data classification of objects. Therefore, the temporal memory is the substratum for all of the neocortical functions. The discrepancy between HTM theory and most other theories of artificial neural grids is possibly the biggest.HTM starts whit the presumption of the memory and the confirmation of sequence of patterns that all the neocortex does.

The main objective of this paper is to analyze various HTM parameters namely cells per column, column dimensions, input bits (N), Width (W), Range (Max and Min values), Boosting and HTM sparsity by working on simple sequence and complex sequence of inputs and observe how they influence the learning process. The rest of the paper is structured as follows; Section 2 covers HTM Overview. Section 3 explains the working methodology of the whole experiment. Experimental results and conclusion are presented in section 4 and 5 respectively. [1]

II. HTM OVERVIEW

The HTM network is made of regions that are arranged hierarchically as shown in the figure 1. Each of these regions has neurons known as cells. These cells are arranged vertically forming a column such that it responds to one single specific input at a time. These cells can be considered as major units of HTM. These cells also have dendrites, both distant and proximal allowing them to connect with input spaces and neighboring cells in that area.

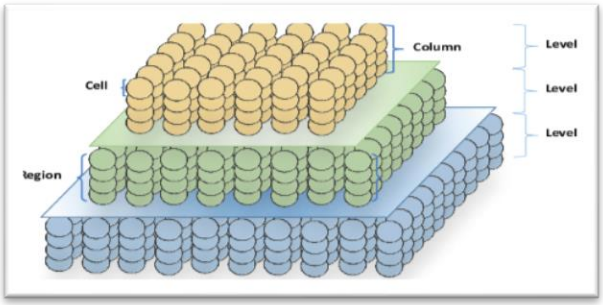


Figure 1: A representation of three layers of HTM arranged hierarchically. Each region contains columns of cells arranged vertically.

A. *Encoder*

In this part, the data taken from as input is encoded semantically as a sparse distributed representation (SDR). These arrays are then passed through the spatial pooler making them into a fixed size. This is done by Hebbian learning, by activating the inactive cells. It is also important to obtain similar SDR'S from similar inputs. In this project we have used a scalar encoder that encodes a numeric value to an array of bits. This is necessary to map our data into SDR to be fed into Spatial Pooler as a scalar input.

B. *Spatial Pooler*

The spatial pooler model encodes the binary input data into sparse distributed representations (SDR). The SDR does the job of learning, distinguishing the common traits and making the predictions. With the use of a proximal segment, consisting of many synapses, each column is linked to a specific component of the input space. If there are enough number of active synapses connected to active bits in the input, then we can say that the proximal dendritic segment is being activated and this result in representing the input with its surrounding column. And finally, the column with the most active inputs and synapses inhibits its neighbors and is active (winner). The spatial pooler three stages namely initialization, overlap, inhibition, and learning. [1]

C. *Temporal Memory Module*

Temporal memory model cycle involves the cells of the winning columns. The active cells of the winning columns produce lateral synaptic connections with the active preceding cells. This helps to predict the active state by simply analyzing the distal segments. Each cell is connected to other cell via distal segments, which has the function of keeping the status of each cell (inactive, predictive, or an active). One cell per active column is selected to be active and to learn the pattern of input. If the input is not expected, i.e. there are no cells in the predictive state, then all the active column cells must burst to be in the active state. In addition, the cell with the greatest number of active synapses, known as the best matching cell, is chosen as a learning cell for learning the sequence of inputs. [2]

III. METHODOLOGY

Following are the parameters on which we have worked to observe how they affect the learning of input sequence in SP/TM layer.

- ***Width***

The number of bits that are set to encode a single value. The "width" of the output signal restriction: w must be odd to avoid centering problems. It is a parameter for encoder. The symbol for width is W.

- ***Input Bits***

The number of bits in the output. Must be greater than or equal to ``w``. It is a parameter for encoder. The symbol for Input Bits is N.

- ***Htm Sparsity***

HTM Sparsity is the ratio of Width (W) and Input Bits (N). It is the combination of W and N, so it is also a parameter for encoder.[4]

- ***Max and Min values***

An encoder parameter, which defines the input range. The input sequence must be within that range. Range should be small just related to the input to avoid overlap between the two input values.

By varying these parameters, we will find out how they are affecting the learning process, upon changing these parameters whether the cycles required by the algorithm to learn the input sequence increases of decreases. The input sequence taken in each of the following experiment is [0, 1, 2, 3, 4, 5, 6, 7, 8, 9].

Case 1. Required learning cycles in dependence of HTM Sparsity

In this experiment we are giving different values of Htm Sparsity to check its effect on learning of the given sequence (i.e. Cycles required for 100% prediction). Find the most optimal value of Htm Sparsity for the given input parameters and sequence. Following input parameters are given to the unit test (HtmSparsity.cs). The syntax of DataRow is:

```
[DataRow(Width, InputBits, Experiment Repetitions)]

[DataRow(9, 90, 1000)] //Sparsity - 0.10
[DataRow(9, 70, 1000)] //Sparsity - 0.128
[DataRow(9, 60, 1000)] //Sparsity - 0.15
[DataRow(9, 50, 1000)] //Sparsity - 0.18
[DataRow(9, 45, 1000)] //Sparsity - 0.20
```

```
[DataRow(9, 40, 1000)] //Sparsity - 0.225
[DataRow(9, 36, 1000)] //Sparsity - 0.25
[DataRow(9, 30, 1000)] //Sparsity - 0.30
```

max = 10;

The unit test runs for 1000 times for each set of data given in data rows above, to get a more average result. As the results are fluctuating a lot and the more times, we run the unit test more averaged result we get.

Case 2. Constant HTM Sparsity with varying Width and Input Bits

In this experiment we are changing the values of width W and input bits N but keeping the value of htm sparsity same i.e. 0.18. As we found the that it's the most optimal value for the given sequence and input parameters in the experiment mentioned in case 1. So, we are finding that, is learning effected if we change the values of Width W and Input Bits N but keeping their ratio same.

```
[DataRow(Width, InputBits, Experiment Repetitions)]
[DataRow(9, 50, 5000)]   //Sparsity - 0.18
[DataRow(27, 150, 5000)] //Sparsity - 0.18
[DataRow(45, 250, 5000)] //Sparsity - 0.18
[DataRow(81, 450, 5000)] //Sparsity - 0.18
```

max =10;

The unit test runs for 5000 times for each set of data given in data rows above, to get a more average result. As the results are fluctuating a lot and the more times, we run the unit test, more averaged result we get.

Case 3. Relationship between HTM Sparsity and Max

In this experiment we are discussing relationship between Htm Sparsity and max. What changes to learning of the sequence because to Htm sparsity when we increase the value of max. Finding out should we increase or decrease the value of HTM sparsity for optimal learning of input sequence in this case [0,1,2,3,4,5,6,7,8,9] and increasing max from 10 to 15. This experiment is performed on HtmSparsityAndMax unit test. The syntax of DataRow is:

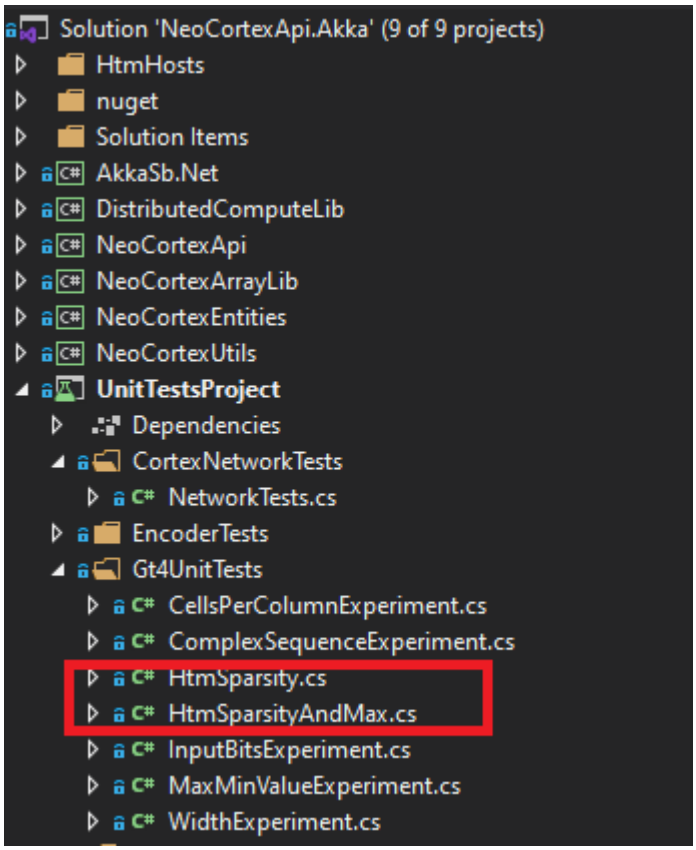
[DataRow(Width, InputBits, max, Experiment repetitions)

```
[DataRow(9, 50, 10, 1000)] //Sparsity - 0.18
[DataRow(9, 50, 15, 1000)] //Sparsity - 0.18
[DataRow(9, 40, 15, 1000)] //Sparsity - 0.225
[DataRow(9, 60, 15, 1000)] //Sparsity - 0.15
```

The unit test runs for 1000 times for each set of data given in data rows above, to get a more average result. As the results are fluctuating a lot and the more times, we run the unit test, more averaged result we get.

IV. C# CODE

The unit tests used for these experiments can be found under the solution "NeoCortexapi.Akka" in HTM folder, in "UnitTestsProject" there is a folder name Gt4UnitTests which has all the Unit Tests we worked on in these experiments. The Unit Tests are highlighted in figure below.



In the unit test "SimpleSequenceExperiment", first encoder an spatial pooler is added to the layer and their parameters are passed as this is the newborn learning stage. After training of SP, instance of TM is added to the layer. Now Spatial pooler and temporal memory are trained together (SP is pretrained on pattern). Prediction also takes place during the learning process.

The unit tests under Gt4UnitTests folder "HtmSparsity" and "HtmSparsityAndMax" contains a loop in which the base experiment "SimpleSequenceExperiment" runs for "loop" number of times. To obtain a more averaged result. Input parameters for different

experiments (Width, Input bits, Number of Experiment Repetitions and Max) is given to the unit test using Data Row. The results from the Unit tests (first cycle at which we get 100% correct prediction for the given sequence) are saved in Excel .csv file using "StreamWriter". Then the graphs are made manually using excel data to observe the result in a graphical form.

V. RESULT

Number of cycles for 100% Prediction:

It is the minimum number of cycles required to get 100% prediction for the given sequence.

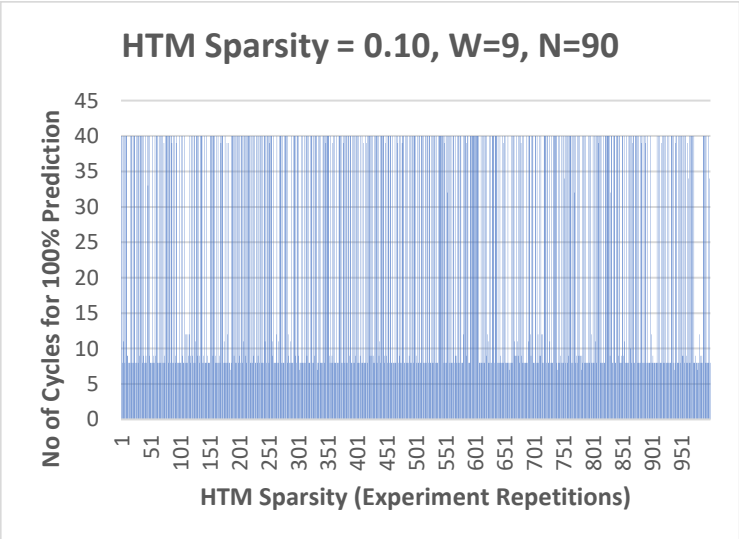
Experiment Repetitions:

It is the number of times the unit test is running. It is running for multiple times due to fluctuations in the results i.e. to obtain a well averaged result.

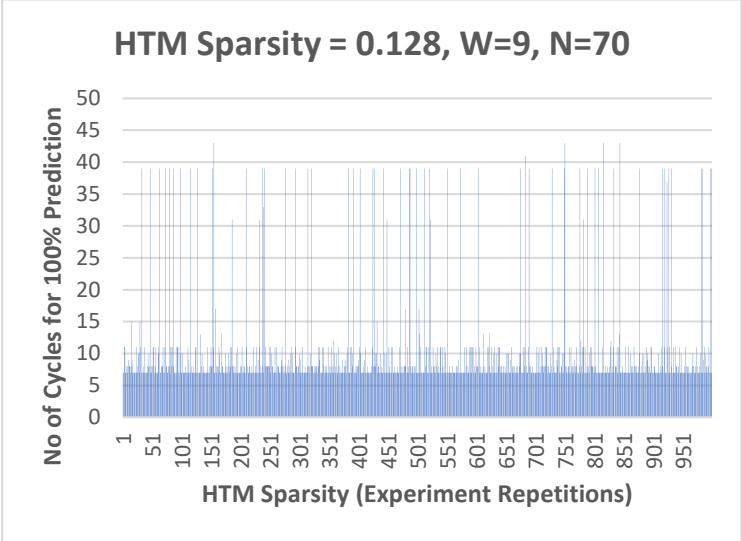
Case 1. Required learning cycles in dependence of HTM Sparsity

In this experiment we observe the number of cycles that are required to get 100 % prediction when predicting the input sequence. Lower number of cycles gives us the most optimal value of HTM sparsity as it is learning the sequence faster. We will observe this phenomenon for the following values of sparsity.

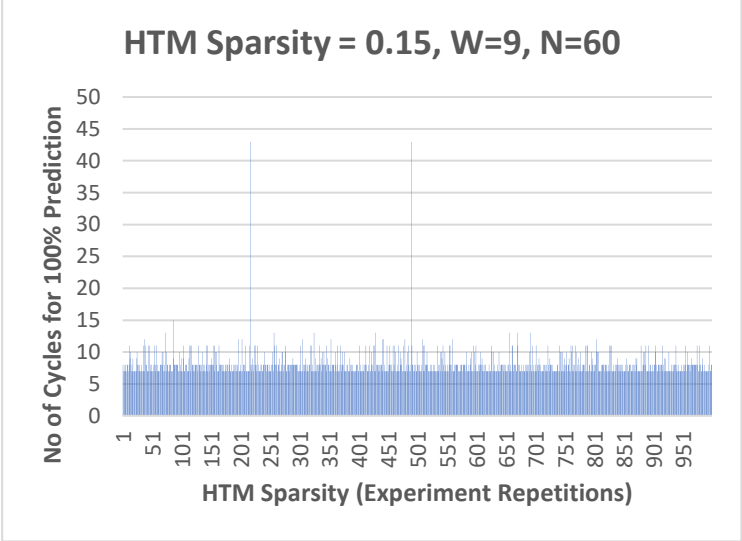
- ***HTM Sparsity = 0.10***



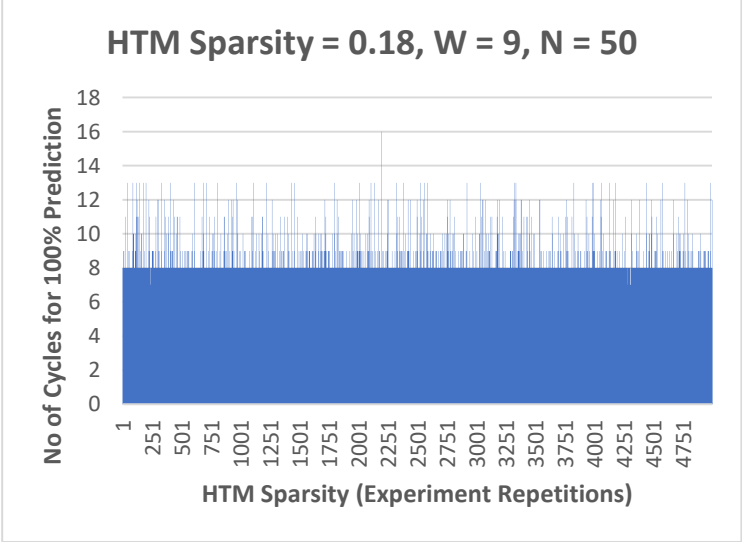
- ***HTM Sparsity = 0.128***



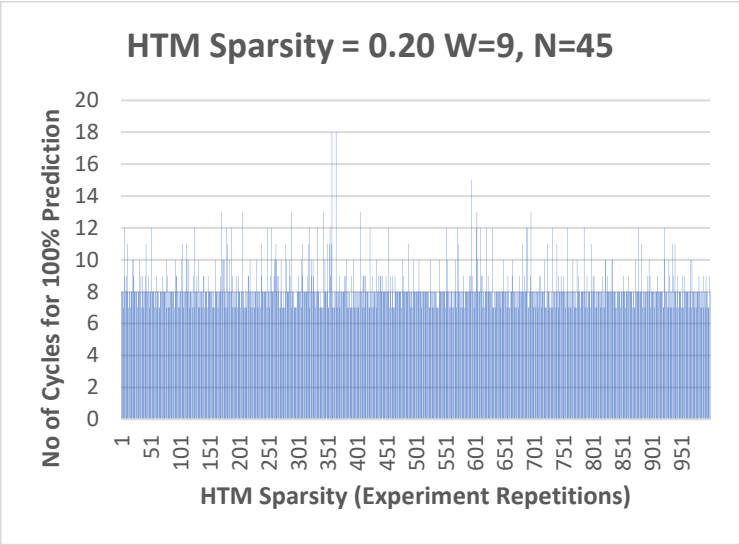
- ***HTM Sparsity = 0.15***



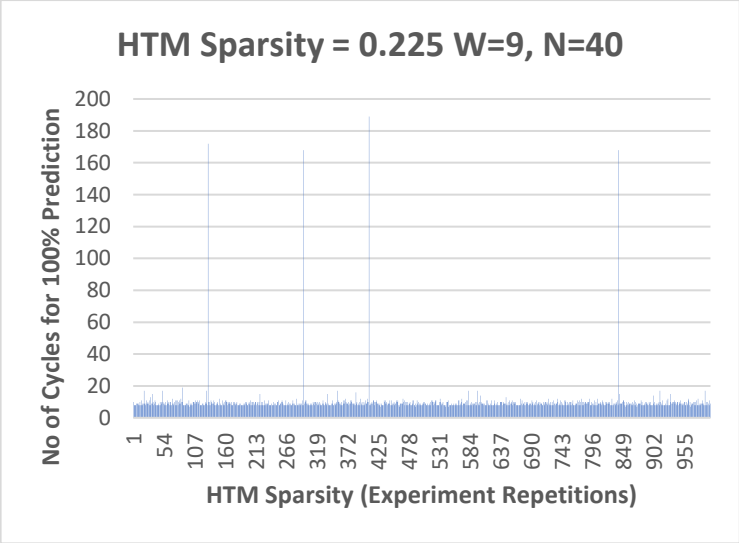
- ***HTM Sparsity = 0.18***



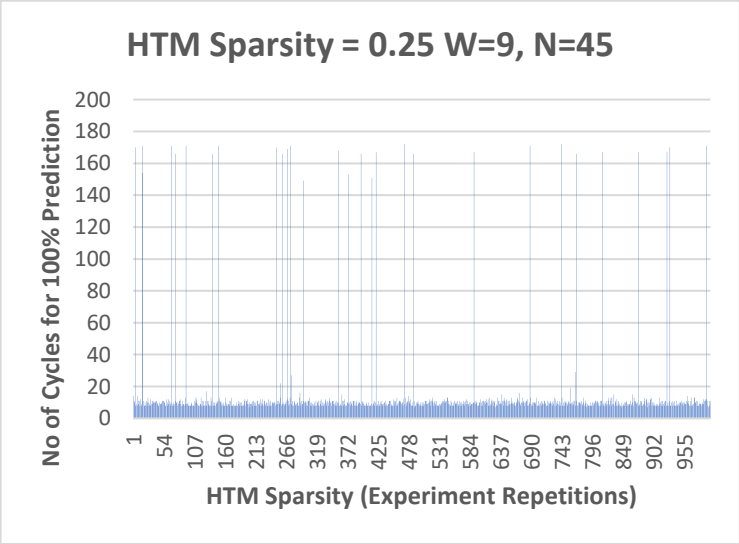
- ***HTM Sparsity = 0.20***



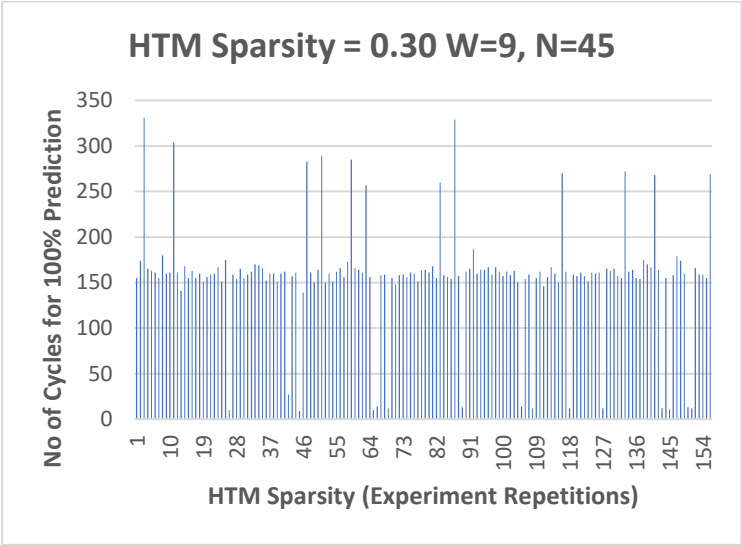
- ***HTM Sparsity = 0.225***



- ***HTM Sparsity = 0.25***



- ***HTM Sparsity = 0.30***



When we average the results at each value of HTM Sparsity, we get the following table:

HTM Sparsity	Average Number of Cycles for 100% prediction
0.1	24.428
0.128	10.002
0.15	8.103
0.18	7.86
0.2	8.224
0.225	9.816
0.25	14.224
0.3	155.6026

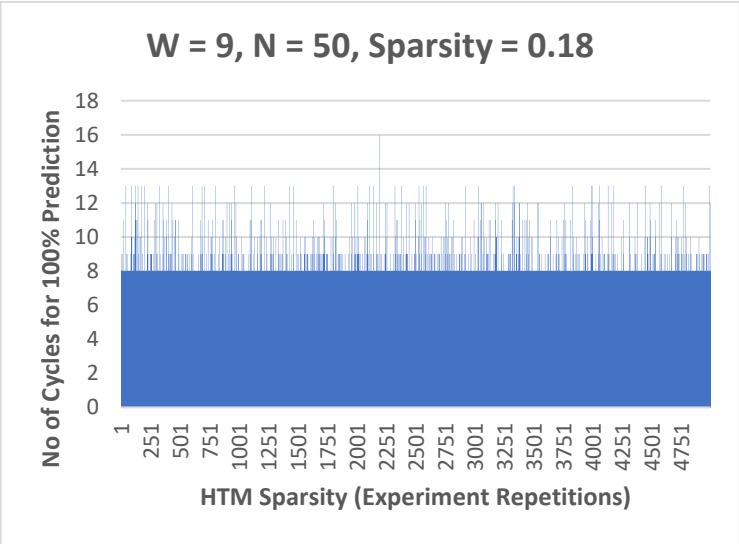
As it can be clearly seen from the above table that @ HTM Sparsity 0.18 we get the most optimal result under the given circumstances, as it is the learning and predicting the input sequence in the lowest number of cycles.

Case 2. Constant HTM Sparsity with varying Width and Input Bits

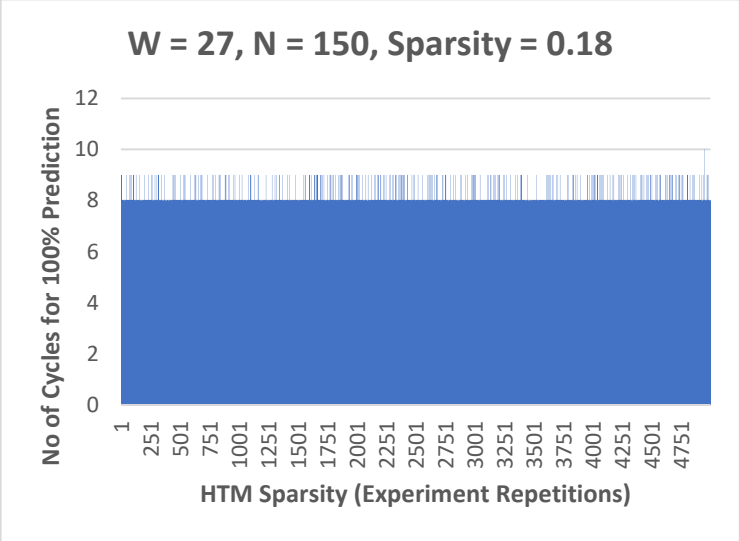
In this experiment we observe that, are the number of cycles changing if we keep the HTM Sparsity same (0.18) and change the values of Width and Input Bits. So, we have to change the values of Width and Input Bits such that their ratio remains constant. We can do that by multiplying the values of Width and Input Bits with Integers. Keeping in mind that the values of Width must be odd to avoid centering problem. We will observe this phenomenon for the following values of Width and Input Bits:

HTM Sparsity = 0.18

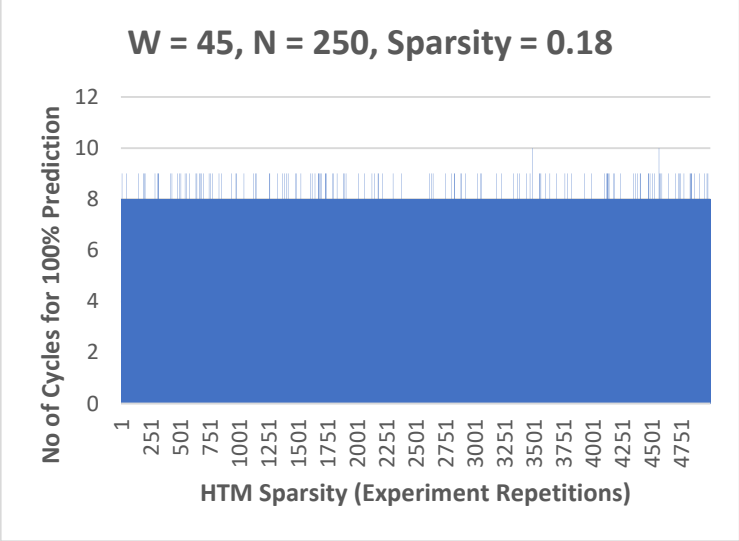
- ***Width = 9, Input Bits = 50***



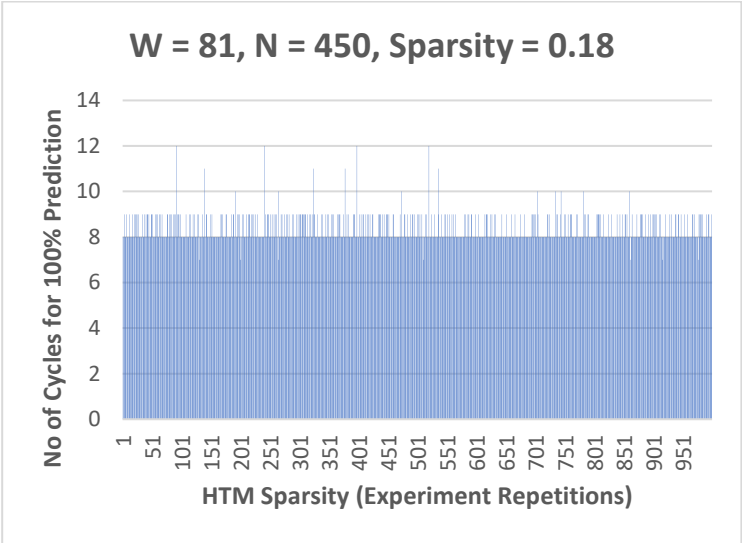
- ***Width = 27, Input Bits = 150***



- ***Width = 45, Input Bits = 250***



- ***Width = 81, Input Bits = 450***



When we average the results at each above value, we get the following table:

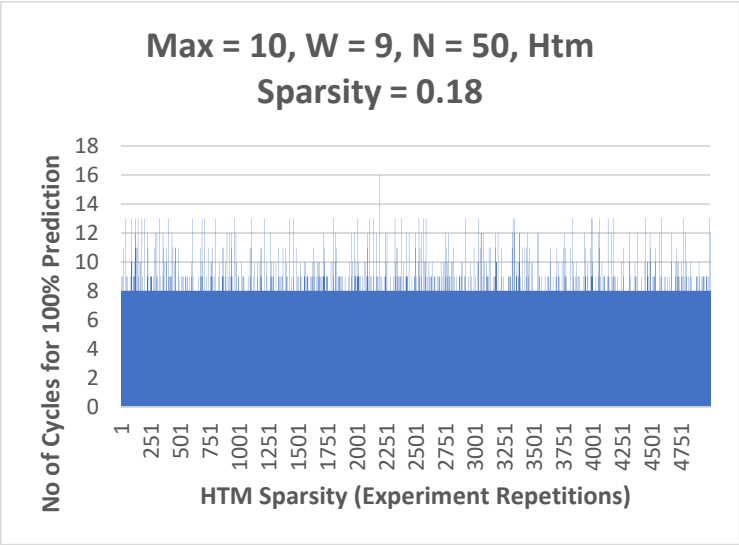
HTM Sparsity, Width, Input Bits	Average Number of Cycles for 100% prediction
0.18, 9, 50	7.97
0.18, 27, 150	7.86
0.18, 45, 250	7.96
0.18, 81, 450	8.03

From the above table, it can be clearly seen that the Number of cycles required for 100 % prediction of the input sequence remains same in each of the case (The difference can be attributed as error).

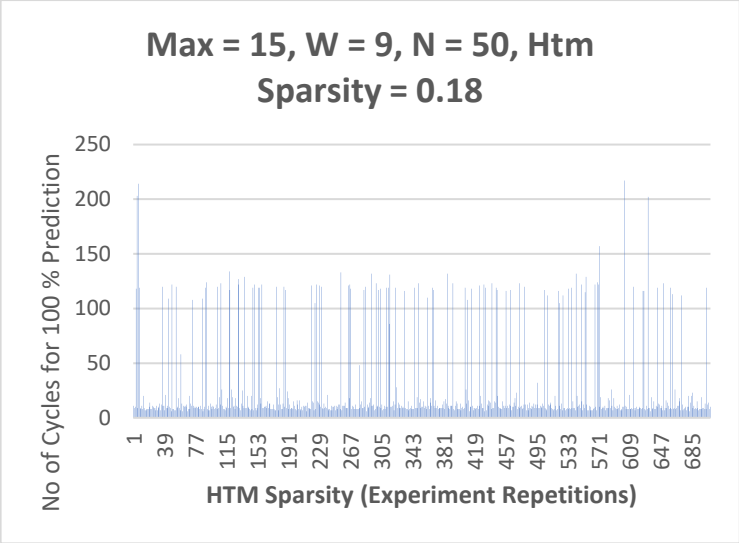
Case 3. Relationship between HTM Sparsity and Max

In this experiment we found a relation between HTM Sparsity and max for optimal learning of the given sequence. First, we increased the value of max to 15 and observe the increase in cycles for 100 % prediction, while Sparsity is same. Then we increased the value of HTM Sparsity to find out that it gave even worse result. Then we decreased the value of HTM sparsity and found that it resulted in lower cycles.

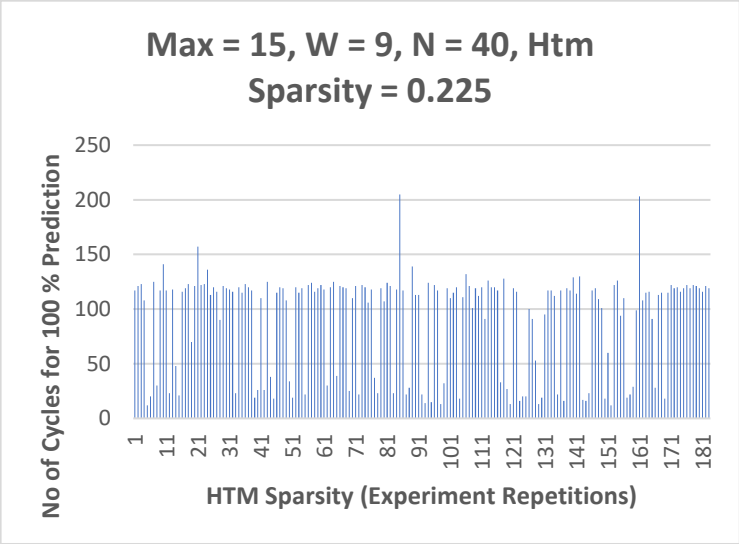
- ***Max = 10, Sparsity = 0.18***



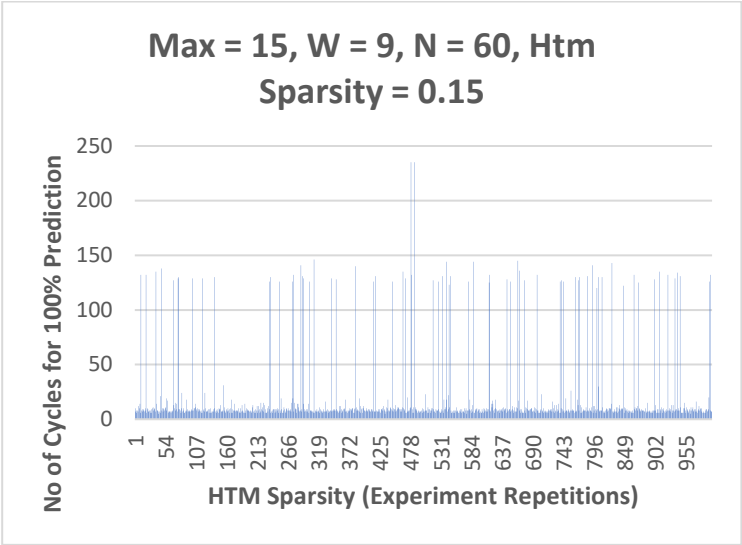
- ***Max = 15, Sparsity = 0.18***



- ***Max = 15, Sparsity = 0.225***



- ***Max = 15, Sparsity = 0.15***



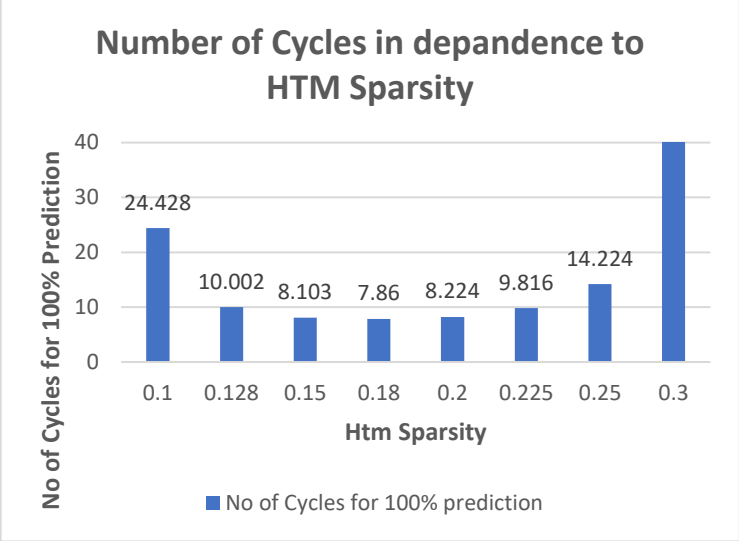
When we average the result from the above graphs, we get the following table:

Max, HTM Sparsity	Average Number of Cycles for 100% prediction
10, 0.18	8
15, 0.18	25.63
15, 0.225	91.55
15, 0.15	17.39

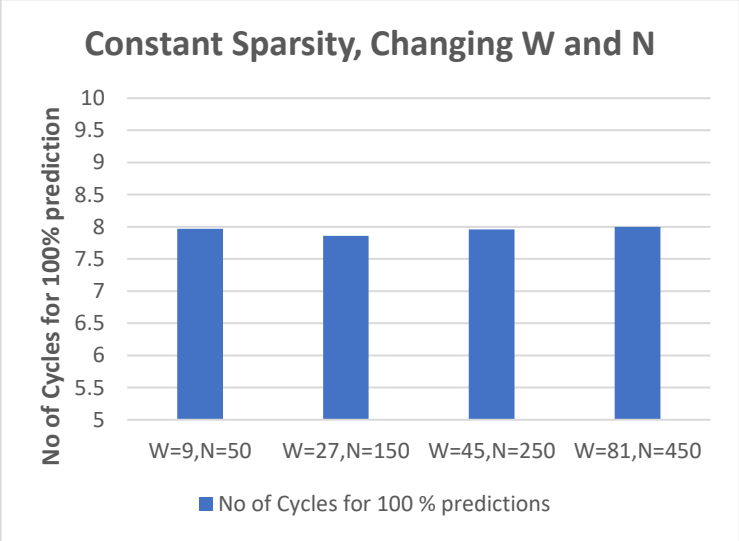
From the above table, it can be clearly seen that for faster learning and prediction we need to decrease that value of HTM Sparsity when max is increased.

VI. CONCLUSION

- In case 1, we found the optimal value for HTM sparsity to be 0.18 for the given parameters and input sequence.

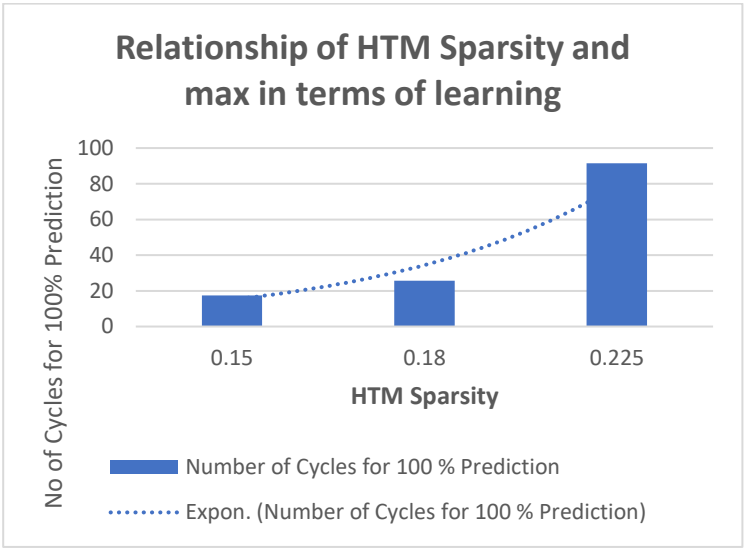


- In case 2, we found that changing width and input bits such that HTM Sparsity remains same does not affect the learning process



- In case 3, we found if we increase the value of max, we need to decrease that value of HTM Sparsity for faster learning. So, for increase in max we need to decrease HTM Sparsity i.e.

$$Max \propto \frac{1}{Htm\ Sparsity}$$



VII. REFERENCE

[1] Hawkins, J., & Ahmad, S. (2016), Why neurons have thousands of synapses, a theory of sequence memory in neocortex. Frontiers in neural circuits, 10.

[2] Zyarah, A. M., & Kudithipudi, D. (2019), Neuromorphic architecture for the hierarchical temporal memory. IEEE Transactions on Emerging Topics in Computational Intelligence, 3(1), 4-14.

[3] S. Ahmad, M. Lewis, (2017), "Temporal memory algorithm", Technical Report Version 0.5, Numenta Inc.

[4] HTM Forum, https://discourse.numenta.org/t/htm-sparsity-vs-reciprocal-of-information-density/1915