FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Investigation of Hierarchical Temporal Memory Multiple Sequences Learning

Duy Nguyen
ngthanhduy7@gmail.com

***Abstract*— The Hierarchical Temporal Memory (HTM) - is an AI framework, which has some applications in practice. This paper focuses on one of its applications which is multiple sequence learning and how to optimize it. The HTM model consists of an Encoder, a Spatial Pooler, and a Temporal Memory. Of the three mentioned components, Temporal Memory will be the focus of this paper as it is where the optimization takes place. For sequence learning, the Temporal Memory is responsible for creating a connection between each element in the sequence. This allows the HTM to predict what the next input element will likely be given an input element. Experiments are set up to test if it is beneficial to keep the connection between the last element of the sequence and the first element, or if severing or resetting this connection is preferable in terms of learning performance. The results gathered from the experiment show a faster learning rate in terms of the number of cycles needed with the removal of this connection.**

Keywords*— *Temporal Memory, learn, predict, resetting, connection

I. INTRODUCTION

The HTM model typically consists of an Encoder, a Spatial Pooler, a Temporal Memory (TM), and an HTM Classifier [1]. Each element in a sequence has its Sparse Distributed Representations (SDRs). Temporal memory learns sequences of these SDRs and makes predictions of the future input SDR [1].

The output SDRs of the Spatial Pooler is a three-dimensional structure with many columns. Each column has several cells per column creating the depth of a column. The Spatial Pooler only operates at the column level. Meaning that given an array of active bits and unactive bits in a certain order as the input for the Spatial Pooler, it will activate some columns in the SDRs that correspond with the given input. Different input results in different columns being active and thus creating different SDRs [1].

The HTM Temporal Memory operates on the cell level of these columns [2]. TM learns connections between cells. when a cell becomes active, it forms connections to other cells that were active just prior. These connections are called synapses and they have their values called permanence values. During learning, some synapses will be reinforced (their values increase) until a certain threshold then they will become active and fully connected. Before surpassing the threshold, the synapses have potential but are not yet connected. Cells can then predict when they will become active by looking at their connections. Therefore, cells can predict what is likely to happen next or what probably is the next element [2].

The previous model of the HTM network keeps all connections between each element of a sequence. The current model introduces a “reset” that will cut the connections between the last element in the sequence and the first element in the sequence. With that said, in theory, when the last element is given to the HTM network, it can no longer predict the next element as opposed to the previous implementation which is likely to predict the first element in the sequence as the next element. But if the last element is truly considered the last element, then predicting the first element as the next element makes little sense. Furthermore, the speed at which the network learns will be evaluated between these two models which is the main reason for introducing this reset.

During the learning phase of the TM, prediction of the next element occurs at every element of a sequence. After the last prediction, a learning cycle

ends and the TM will learn and predict the same sequence again thus starting a new learning cycle. The preferable result is each cycle has the correct predictions for every element. So after how many cycles with correct predictions can the learning process be stopped? The previous exit condition is set to be 30 consecutive cycles with correct predictions. This experiment tries to find a smaller number of cycles used to exit the learning phase if possible.

II. METHODS

Two arbitrary sequences were chosen for this experiment:

1st sequence: {16.0, 17.0, 18.0, 19.0, 20.0, 19.0, 18.0, 17.0, 16.0, 15.0, 16.0, 17.0}

2nd sequence: {1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 19.0, 18.0, 17.0, 16.0, 15.0, 16.0, 17.0}

Each sequence is put inside a .csv file which is inside MyInput folder. The sequence(s) will be extracted by the program and will be put into the TM to learn.

Both network models (with and without reset) learn the same two sequences continuously. The pseudocode is of the following [4]:

```
Initialize maxCycles = 1000, maxMatchCnt,
isInStableState;
Initialize layer1 <- CortexLayer which
contains the Encoder, SP, and TM
Initialize HTM Classifier cls
Initialize HomeostaticPlasticityController
hpc
  IF hpc isStable
     isInStableState = isStable;
Initialize cycle = 0, matches = 0,
lastPredictedValues as List of string;
FOR i = 0 to maxCycles and isInStableState
is false
   matches = 0;
   cycle++;
       FOREACH inputs in sequences
          FOREACH input in inputs
              Compute layer1 using input;
                  IF isInStableState
                  break;
          IF isInStableState
          break;
cls.ClearState(); <- Clear all learned
patterns in the classifier.
FOREACH sequenceKeyPair in sequences
   FOR i = 0 to maxCycles
       matches = 0;
       cycle++;
   FOREACH value of  sequenceKeyPair
       Compute layer1 using value and
assign to lyrOut;
       Get activeCells from layer1;
       Get key of the sequence (Ex: key "1-
4-5")
       Learn cls with key and activeCells
<- assign SDRs with a key
        IF lastPredictedValues contains key
            matches++;
        IF there are predictiveCells in
lyrOut
GetPredictedInputValue(predictiveCells);
<- predict the next element and return the
result to predictedInputValues.
           Assign lastPredictedValues with
predictedInputValues;
        Initialize accuracy =
matches/number of value of sequenceKeyPair
* 100;
        Initialize maxMatchCnt = 0;
        IF accuracy >= 100
           maxMatchCnt++;
        Reset the TM; <- Clears any
predictions and makes sure synapses don't
grow to the currently active cells in the
next time step.
```

For consistency, The only difference between the models is the reset at the end of a sequence which is highlighted in red in the pseudocode. Model without will not clear the prediction for the next time step which is back to the first element. The exit condition for the learning phase for both models is after 1000 learning cycles.

If every prediction in a sequence in a cycle is correct. Then the accuracy reaches 100% for that cycle. The preferable outcome is that the accuracy reaches 100% for as many consecutive cycles as possible. Due to the removal of synapses' growth at the last element when using TM reset, the calculation of accuracy will be different between the two models. The detail will be discussed in the Accuracy section below.

As mentioned in the Introduction, there should be a way to find out how many consecutive cycles with 100% accuracy are needed to exit the learning phase. Consider a cycle that has 100% accuracy shall be called a stable cycle, a list of stable cycles is introduced to keep track of all of these cycles. The list first stores the order of a cycle only if the accuracy of that cycle is 100%. It will not store the order of consecutive cycles with the same 100% accuracy. But if at a certain cycle in the future, the accuracy drops down from 100% (Meaning there are incorrect predictions of one or more elements in a sequence), then the list will store the previous order of the cycle which has 100% accuracy. This creates a "starting" cycle and an "ending" cycle of consecutive cycles with 100% accuracy. In the end, the list would have some kind of "coordinates" where it stores pairs of starting order of cycle and ending order of cycle. These coordinates will show fast and how stable a network model learns. Details

about this list will be discussed in the List of stable cycle coordinates below.

A. *Accuracy*

The formulas for the accuracy are different with and without using reset:

1. Accuracy without reset = matches/input length*100.0.
2. Accuracy with reset = matches/(input length - 1)*100.0.

With matches are the number of times the predicted sequence matches with the expected sequence and input length is the length of the sequence.

Notice that with reset, the accuracy does not account for one prediction (input length – 1 highlighted in red). As mentioned before, the removal of synapses' growth at the last element prevents the TM to predict the next element (Since in theory, there is no more element after the last one). The TM will still try to predict the next element at every time step (every element) but (input length – 1) means the calculation will disregard the last prediction.

Furthermore, resetting only occurs during the learning phase and is not used afterward. After the learning phase, the user can input a number and the HTM will simply predict the next sequence without resetting. Therefore, the accuracy calculation with reset can be considered more lenient compared to without reset.

B. *List of stable cycle coordinates*

To explain this list better, consider an example: During the learning phase of a sequence, the accuracy reaches 100% at the 34th cycle. It keeps the accuracy until the 41st cycle when the accuracy drops below 100%. Then the first pair of coordinates in the list would be {34, 40}. Let's says the accuracy rises to 100% again at the 78th cycle and stays that way until the last cycle which is the 1000th cycle, the second pair of coordinate would be {78, 1000}. The list will have two pairs of coordinates in total. So, when comparing the two models, the model that has fewer pairs is preferable as the accuracy stays at 100% instead of fluctuating between 100% and below 100% during 1000 cycles of learning phase.

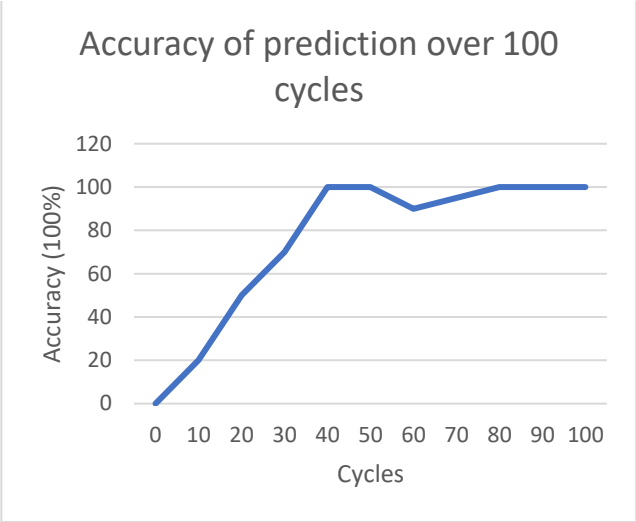


Figure 1 Accuracy of prediction over 100 cycles

Figure 2 was added to give a better visualization about the oscillation that may occur in the learning phase. The accuracy reaches and stay at 100% first from the 40th to the 50th cycles. It then drops slightly after until the 80th cycle where it rises to 100% again and stays there till the 100th cycle.

Furthermore, the model with a longer distance between the starting maximum accuracy cycle and the ending maximum accuracy cycle (Ex: {40, 1000} is longer or bigger than {234, 1000}) is the better one. The reason is that both models exit the learning phase at the 1000th cycle, so it cannot be said that the accuracy will not drop from 100% after the 1000th cycle as the experiment cannot be run forever. But at least if the number of consecutive maximum accuracy cycles is big, there is more confidence to say that the model has truly learned the sequence.

III. Result and Discussion

Data are collected from learning each sequence with and without using reset. Data from the 1st and 2nd sequences will be combined to calculate the minimum, maximum and average values for the number of stable cycle coordinates as well as the length between the starting maximum accuracy cycle and the ending maximum accuracy cycle in each coordinate. The data can be found on my GitHub repository [6].

A. ***Comparison of the number of stable cycle coordinates.***

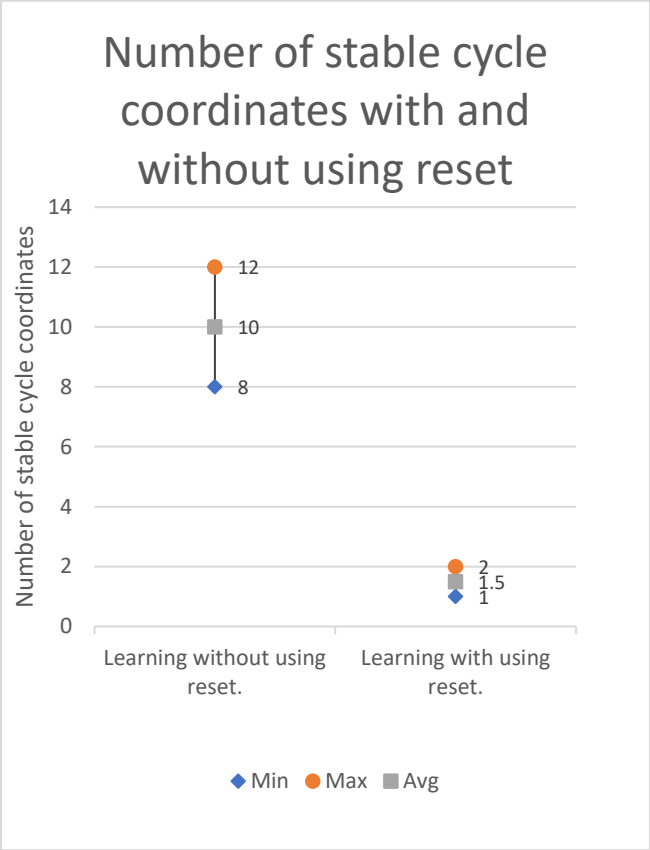
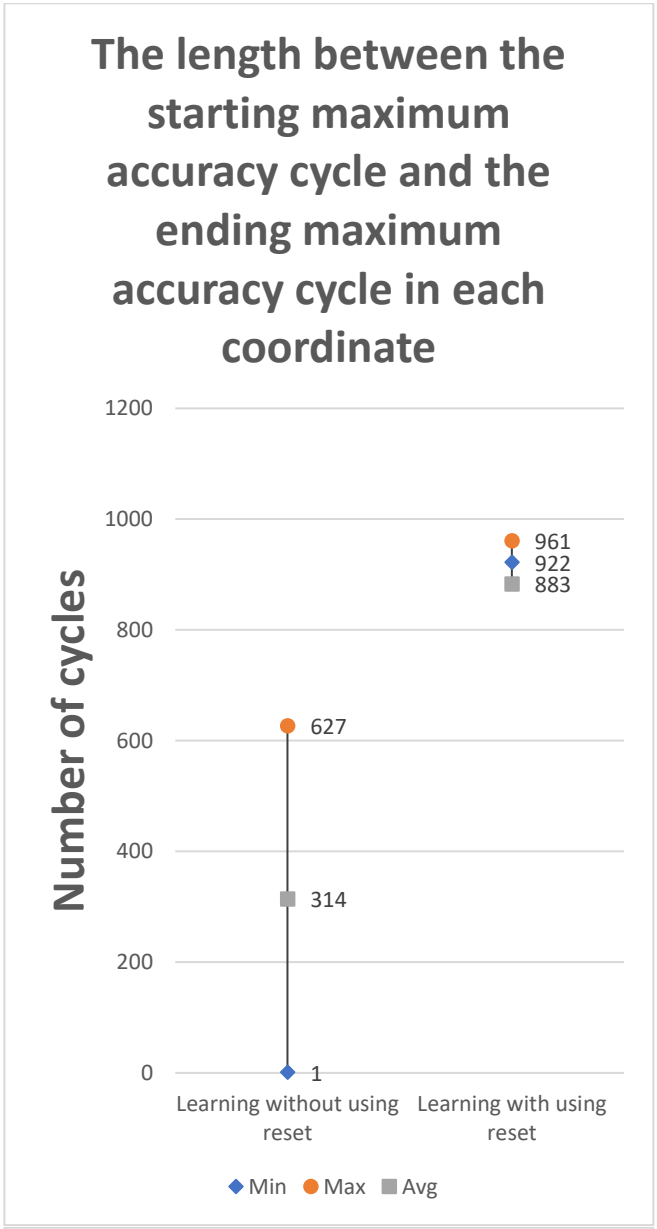
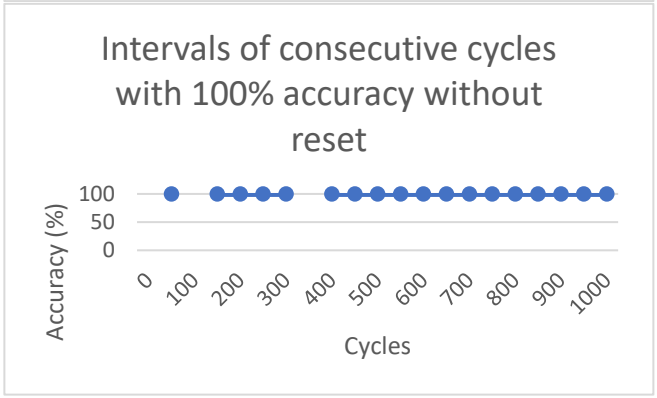


Figure 3 Number of stable cycle coordinates with and without using the reset

On average, the number of stable cycle coordinates when learning sequences without using reset is about ten times higher than with using reset. Please note that the smaller the number the better. Because the higher number of these coordinates means that during the learning phase, the accuracy could not maintain the maximum value and there are more oscillations between 100% and sub-100 % accuracy. Therefore, the model using reset is better in this case. But as mentioned before, another criterion to be considered is the length between the starting maximum accuracy cycle and the ending maximum accuracy cycle should be as big as possible. The next section will show and evaluate this length.

B. ***Comparison of the length between the starting maximum accuracy cycle and the ending maximum accuracy cycle in each pair of coordinates.***





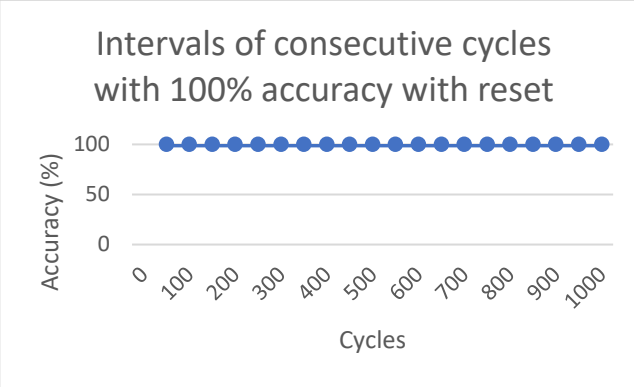


Figure 4 The length between the starting maximum accuracy cycle and the ending maximum accuracy cycle in each coordinate

As shown in Figure 3, on average, learning using reset has a much longer length of consecutive cycles with an accuracy of 100% which is 922 consecutive cycles. Compared to only 314 cycles on average when not using reset, the model with reset is about 3 times better in terms of having the longest consecutive cycles with 100% accuracy between the two. Furthermore, the difference between the minimum and the maximum length when using reset is not as big as the length without using reset. Also, when looking at the intervals of the consecutive cycles with 100% accuracy, learning with reset is more stable as it has only one interval starting from 118th cycle till the end cycle. Learning without reset, however, has several intervals where the accuracy drops down or fluctuates from 100%, discontinuing the line in the second graph of Figure 3. So learning with reset tend to be more stable.

As mentioned before, there is no telling if the accuracy will drop down from 100% after the 1000th cycle. But with the number of consecutive cycles with 100% accuracy reaching about 922 cycles over 1000 cycles. It is unlikely that the model using reset will drop its accuracy anytime soon if the experiment continues beyond 1000 cycles. And thus, the model with reset can be considered to truly learn the sequences faster than without reset.

IV. CONCLUSION

By severing and preventing the growth of synapses after reaching the last element in a sequence during the learning phase, the network was able to learn sequences faster. Also, the number of oscillations in which the accuracy fluctuates between 100% and sub-100% during learning was greatly reduced.

Furthermore, as shown in Figure 3, when using reset, the minimum number of consecutive cycles with 100% accuracy is 883 cycles over 1000 cycles which is very high. The current exit condition for the learning phase is 30 consecutive cycles with 100% accuracy. This number can be lowered to something like 10 as it is unlikely that after the first cycle in which the accuracy reaches 100%, it will drop down below 100% in the future.

REFERENCES

1] Numenta, "Numenta," 2021. [Online]. Available: https://numenta.com/resources/biological-and-machine-intelligence/temporal-memory-algorithm/.

2] Numenta, "Temporal Memory Algorithm," [Online]. Available: https://numenta.com/assets/pdf/temporal-memory-algorithm/Temporal-Memory-Algorithm-Details.pdf.

3] Damir Dobric, "GitHub ddobric/neocortexapi," [Online]. Available: https://github.com/ddobric/neocortexapi. [Accessed 2021].

4] D. Nguyen, "GitHub perfectaccountname/neocortexapi," [Online]. Available: https://github.com/perfectaccountname/neocortexapi/blob/master/source/Samples/NeoCortexApiSample/SequenceLearning_DUY.cs.

5] J. Hawkins and et al., "Sparse Distributed Representation," in *Biological and Machine Intelligence*, Release 0.4 ed., Numenta, 2016-2020.

6] D. Nguyen, "Documentation perfectaccountname/neocortexapi," GitHub, [Online]. Available: https://github.com/perfectaccountname/neocortexapi/tree/master/source/Documentation.