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

Approve Prediction of Multisequence Learning

Saad Jamil
jamil.saad@stud.fra-uas.de

Nabeela Maham
nabeela.maham@stud.fra-uas.de

Tanvir Ahmed
tanvir.ahmed@stud.fra-uas.de

***Abstract*— This paper discusses Multisequence Learning, a strategy for learning and predicting sequences. The current Multisequence Learning implementation is analyzed to better understand how sequences are learned and predicted. The study then suggests a novel approach that improves on the current implementation. The novel method automates the procedure of reading learning sequences from one file and testing subsequences from another to determine the % prediction accuracy, which is then saved in a result file. This improves efficiency and reduces the likelihood of errors when compared to manually inserting sequences. The approach can be used for a variety of industrial applications, including recognizing music and classifying disease peptides.**

The paper emphasizes the relevance of sequence learning and prediction in a variety of businesses, as well as the proposed method's ability to reliably predict sequences. The findings indicate that the suggested approach can be utilized to improve a variety of applications involving sequence learning and prediction. Overall, the paper discusses Multisequence Learning and introduces a new method for improving the efficiency of sequence training and prediction, as well as storing prediction accuracy in a file.

Keywords*— *Hierarchical Temporal Memory, Spatial Pooler, Temporal Memory, encoder, sparse distributed representation, AI, ML, Local Area Density, Potential Radius.

I. INTRODUCTION

Multi-sequence learning with HTM entails teaching the algorithm to detect and anticipate patterns in multiple input sequences. To perform multi-sequence learning using HTM, we must first encode the input information into Sparse Distributed Representations (SDRs), which can be accomplished with a scalar encoder. After the data has been encoded, the spatial pooler generates sparse illustrations of the input sequences that are supplied into the temporal memory element for learning and prediction. Multisequence learning using HTM is an effective method for identifying and predicting patterns across numerous input sequences.

In this project, we have tried to implement new methods along the Multisequence Learning algorithm [1]. The new algorithms automatically read the dataset from the provided file; however, we also have test data in another file that must be read for later validating the subsequence. Multisequence Learning uses many sequences and test subsequences to learn. After the learning process, the predicted element's predicted element's accuracy is calculated.

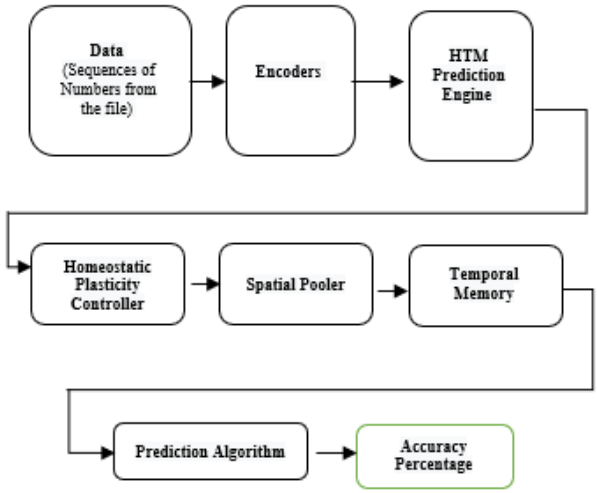

Figure 1: Overview of Sequence Learning [6]

II. LITERATURE SURVEY

A. *Hierarchical Temporal Memory*

HTM aims to mimic the brain's hierarchical structure as well as the learning process. It comprises a hierarchically arranged network of nodes, each representing a group of neurotransmitters in the neocortex. These nodes learn to find patterns in sensory data and make forecasts based on their previous experiences [1]. The predictions are then compared to the input data to fine-tune the node's models and improve predictive precision.

According to Hawkins, the neocortex teaches and generates predictions by constructing an ordered set of columns, each comprising a group of neurons that detect patterns in sensory data. These columns interact in a hierarchical structure, with higher-level columns reflecting abstract concepts [3].

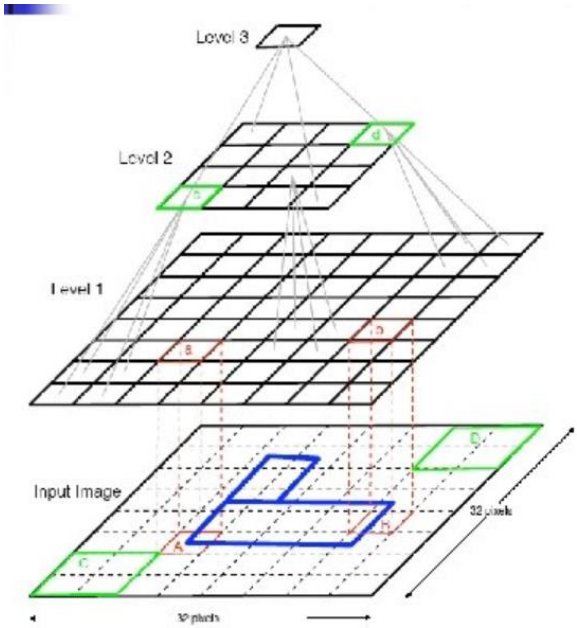

Figure 2: Hierarchical Temporal Memory Structure [8]

B. Proximal Dendrite Segments

A proximal dendrite joins cells in a column, with synapses denoted by little black circles. A solid circle indicates a valid synaptic link with a persistence value that exceeds the connection threshold. An unfilled circle represents a putative synaptic connection with a persistence value lower than the connection threshold. If enough valid synapses are associated with active input bits, feedback input activates a column following a local inhibitory step.[4]

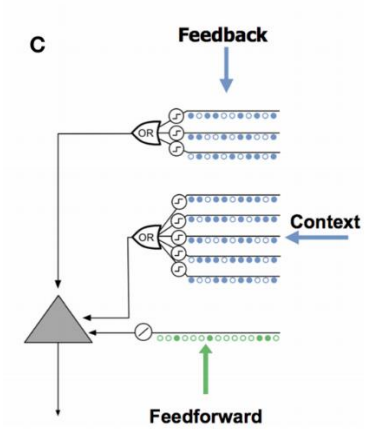

Figure 3: Proximal Dendrite Representation [4]

C. Distal Dendrite Segments

Each cell in the brain normally has about 130 distal dendrite categories, each with approximately 40 synapses. Additionally, it has a single proximal dendritic segment. Nearby cells offer lateral input to the distal segments, and within a certain "learning radius," a set of synapses can link to a subset of other cells.[6] The dendritic segment connects with previously active cells, allowing it to recall surrounding cells' activation states. If a dendritic segment identifies the same cellular activation pattern, that is, if the number of active synaptic on any segment exceeds a certain threshold, the cell enters an anticipatory state. So, signaling that feedforward input will most likely result in column activation. Cell activity is maintained by either feedforward input using the middle dendrite or lateral links through the distal branch segments.[7]

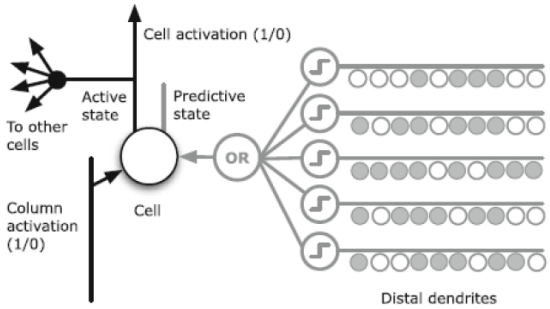

Figure 4: Distal Dendrite Segment at the Cellular Level [3]

D. Neocortex

The neocortex, an area of the cerebral cortex, is accountable for a variety of mental tasks in humans. With billions of cells and hundreds of thousands of meters, it is a complicated and intricate structure.[8] The cells are organized in layers, with distinct areas dedicated to various activities such as hearing, sight, touch, movement, and sensory balance.

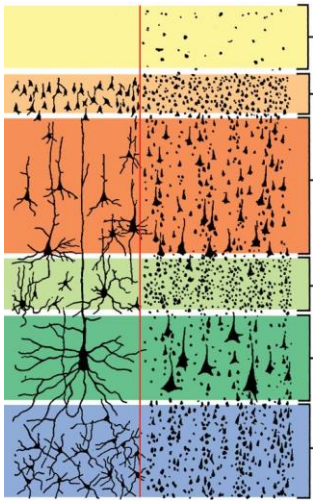

Figure 5: Neocortex Layers [9]

E. Prediction

It is mainly responsible for estimation, which is the foundation of intelligence. It uses static representations and new incoming data to make predictions about the real world. [7]

F. Memory

Parallel processors and the cortex are not the same. Although parallel computers do many computations on input patterns simultaneously to produce separate output patterns, the brain uses this method to retrieve output from its massive memory more quickly.[9] The cortex naturally accumulates and links sequential and regular patterns in a

hierarchical order. These connected memories can recover entire patterns from partial data entries in both as well as temporal memory.

G. Connection

The HTM neuron model is inspired by cortical neurons and differs from the standard ANN neuron model shown in Fig 6 [7], which requires a weighted average of inputs followed by a non-linear operation on the sum. Recent neuroscience research indicates that biological neurons execute more sophisticated roles and communicate using both electrical and chemical signals, which provide the foundation of retention and recall in the brain.[8]

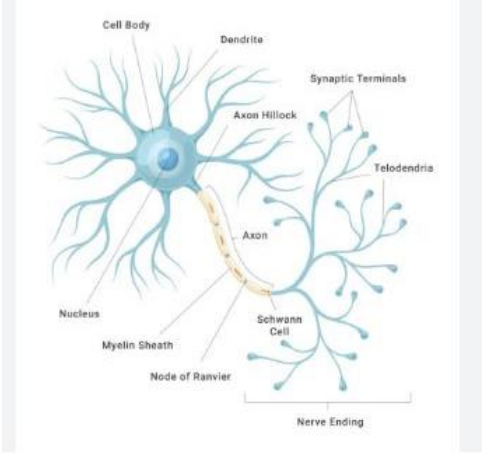

Figure 6: Neuron Structure [7]

The communication process is similar: whenever Neuron A gets an organic signal from another neuron, it becomes electrically charged with the fluid surrounding its membrane.[2] The electric field moves down the axon and away from A's soma. Vesicles are a collection of storage sites within the synapse that contain soma-produced chemicals. When an electrical charge reaches the point of contact, these vesicles fuse with its cell membrane, releasing neurotransmitters into the cleft between neurons. Neurotransmitters pass across the synaptic cleft to one of neuron B's dendrites, where they link to membrane receptors. Neuron B generates an electrical charge that travels down its axon before repeating the process.[4]

H. Temporal Memory

Temporal Memory functions by storing a collection of active cells that represent the current context of the input data. When new input patterns are received, the active cells change depending on how similar the input is to the present context. In addition, the program handles a set of cell connections that represent the series of patterns encountered before [6].

Using these connections, Temporal Memory can forecast the next most likely input pattern depending on the current context. If the forecast is true, the algorithm strengthens connections between cells that were active during the projected sequence [9]. If the forecast is inaccurate, the algorithm modifies the connections to lower the possibility of the sequence occurring again in the future.

I. Multisequence Learning

Multisequence learning is an HTM-based system that learns and predicts numerous sequences of patterns at the same time [10]. Multisequence learning involves learning and predicting each sequence of patterns using distinct Temporal Memory modules.

Ahmad's [7] technique is based on the utilization of a hierarchical node structure to learn and anticipate sequences of features at various levels of abstraction. At the most basic level, each Temporal Memory unit evaluates and anticipates the raw sensory input from just one modality. At greater levels, the nodes understand and anticipate sequences of patterns that incorporate data from many modalities.

III. METHODOLOGY

Our primary goal was to automate a couple of processes used outside of Multisequence Training. One of them was to read data from a file. Retrieve the subsequence from the file. Test the next step on the model that was trained and calculate its accuracy. In addition to all of this, we made the test more dynamic to generate a synthetic dataset. This approach automates the process of producing several sequences for training purposes.

A. Sequence

A sequence is a data format composed of a sequence identifier and a numeric sequence. The component of the list of sequencing models of data is just a collection of sequences. The subsequence is a portion of the sequence; hence it uses the same data structure.[2]

```
public class Sequence
{ public String name { get; set; }
   public int[] data { get; set; }}
```

Source code 1: Data model of Sequence

```
[{"name": "S1",
   "data": [ 0, 2, 5, 6, 7, 8, 10, 11, 13 ]},
  {"name": "S2",
   "data": [ 1, 2, 3, 4, 6, 11, 12, 13, 14 ]},
  {"name": "S3",
   "data": [ 1, 2, 3, 4, 7, 8, 10, 12, 14 ]}]
```

Dataset 1: Sample dataset

B. Encoder

The encoder accepts raw data inputs and converts them into SDRs that describe the input data's essential properties. The encoding process consists of numerous processes, including reduction of dimensionality, noise reduction, and equalization. The encoder also learns to identify patterns in input data and adjusts to variations in input concentration over time.[3]

The SP encoder is intended to handle temporal data and employ a sliding window method to detect temporal patterns in the information that is input. It first transforms the input data into an ongoing flow of binary numbers, which are then supplied into the HTM network as a series of SDRs [5].

C. Spatial Pooler

The Spatial Pooler is an unsupervised learning algorithm that converts high-dimensional input data into a lower-dimensional SDR which reflects the input data's essential properties.[12] It accomplishes this by first assigning an arbitrary number of weights to each input characteristic and then calculating the overlap between the input and the weights. The characteristics with the largest overlap are then chosen and incorporated into the SDR [6].

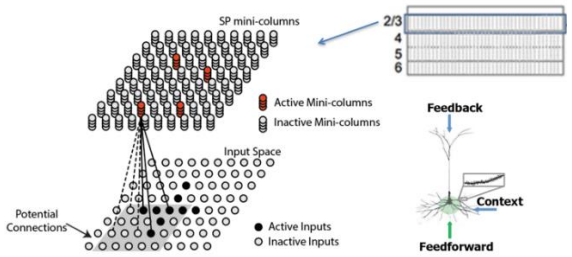

Figure 7: Representation of spatial pooler[12]

D. Sparse Distributed Representation

In HTM, SDRs are used for expressing network activity patterns. Each input to the network is converted into an SDR, which is subsequently handled by the network's node hierarchy to anticipate future input.[5]
Hawkins and Ahmad proposed SDRs, which are binary sequences containing a small number of active bits (ones) out of a vast number of total bits, as a logical way to describe sparse, scattered activation patterns in the neocortex [4].In HTM, it is critical to choose proper parameters for the various approaches, and the table lists Spatial Pooler settings with default values that are widely used in HTM investigations. Each of these criteria has a distinct impact on HTM's performance. However, we will concentrate on the effects of specific characteristics, like potential radius and local area volume, global/local inhibition, and the variety of active columns per area.[6]
The initial step in utilizing any HTM configuration is to define numerous parameters using the htmconfig class. This is a critical step in the process. The table below lists all the HTM parameters that affect image classification.

Parameters	Default value
input bits	100
numColumns	1024
CellsPerColumn	25
GlobalInhibition	true
LocalAreaDensity	-1
NumActiveColumnsPerInhArea	0.02 * numColumns
PotentialRadius	0.15 * inputBits
MaxBoost	10.0
InhibitionRadius	15
DutyCyclePeriod	25
MinPctOverlapDutyCycles	0.75
MaxSynapsesPerSegment	0.02 * numColumns
Activation Threshold	15
Connected Permanence	0.5
Permanence Decrement	0.25
Permanence Increment	0.15
Predicted Segment Decrement	0.1

Table 1: HTM Config parameters

I. IMPLEMENTATION

This section describes how the Multi Sequence Learning Experiment was carried out. We investigated how multi-sequence learning for a sequence of numbers works and improved the accuracy of the HTM prediction engine. Furthermore, we created Sequence Learning, which automates the process of reading learning sequences from one file and testing subsequences from another to determine the % prediction accuracy and then saving the results in a result file at the end of the program.[7]

A. Learning Phase

The learning phase includes fetching Datasets from the solution directory and Train using a spatial pooler using a Homeostatic Plasticity Controller for stability. Training of datasets for Multi Sequence Learning is explained as follows.

Training of Multi-Sequence of Numbers:

The process of learning an ordered set of numbers includes the initialization of datasets, including the label and the sequence. The Sequence is then used to train the spatial pooler using HTM setup parameters across multiple iterations. After numerous repetitions, the spatial pooler reaches a stable state. The figure below shows how the Multi Sequence Model performs Sequence of Numbers training.

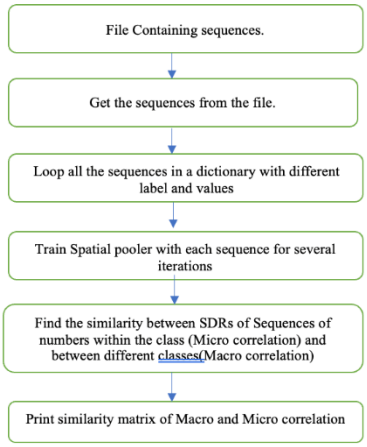

Figure. 8: Training Model – Sequences of Numbers [2]

B. Prediction Method

When the learning/training stage is complete, the model generates a similarity matrix for each class. The SDRs calculated for the input numbers are then compared to the SDRs of the appropriate sequence learned during training to determine accuracy based on the total number of hits and sequence count.[8] In addition, the input sequence is transformed into an SDR and compared to each of the learned sequences' SDRs during training. The correlation matrix is then utilized to find the best match, and the estimated sequence is allocated an observation class (label) based on its accuracy and classification, with the prediction Engine displaying the percentage accuracy of the sequences it belongs to.[11]

IV. RESULTS

In this project, we have used a sequence of Numbers for Multi-sequence learning of Numbers Sequence for Multi Sequence.

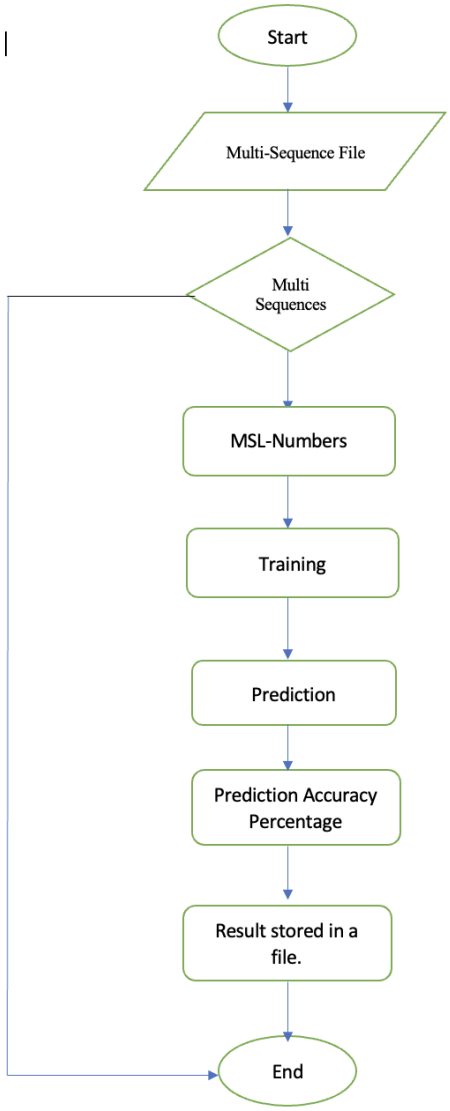

Figure. 9: T shows the flow chart for multi-sequence learning for the experiment carried out.[13]

We have run the experiment max possible number of times with different datasets. We have tried to keep the size of the dataset small and the number of sequences also small due to the large time in execution.

```
Predicted Sequence: S1 - Predicted next element: 8
------------------------------
Accuracy for T1 sequence: 100%
------------------------------
Using test sequence: T2
------------------------------
Input: 6
Predicted Sequence: S1 - Predicted next element: 7
Input: 11
Predicted Sequence: S2 - Predicted next element: 12
Input: 12
Nothing predicted :(
------------------------------
Accuracy for T2 sequence: 33.33333333333333%
------------------------------
Using test sequence: T3
------------------------------
Input: 1
Predicted Sequence: S2 - Predicted next element: 2
Input: 2
Predicted Sequence: S2 - Predicted next element: 3
Input: 3
Predicted Sequence: S2 - Predicted next element: 4
------------------------------
Accuracy for T3 sequence: 100%
------------------------------
Using test sequence: T4
------------------------------
Input: 3
Predicted Sequence: S2 - Predicted next element: 4
Input: 4
Predicted Sequence: S3 - Predicted next element: 7
Input: 7
Predicted Sequence: S3 - Predicted next element: 8
Input: 8
Predicted Sequence: S3 - Predicted next element: 10
------------------------------
Accuracy for T4 sequence: 100%
Done...
```

Figure 10: Prediction and calculation of accuracy on subsequence.

V. CONCLUSION

The Neocortex API's multi-sequence learning model of reference was used to create a solution for multi-sequence learning of a sequence of numbers. The HTM Prediction Engine was configured with numerous settings to suit the training procedure.[8] The Multi-Sequence of Numbers was saved, turned into an encoded value, and stored in a dictionary for training utilizing an encoder and SDR input.[2] The existing technique was upgraded to predict trained sequences by extracting from a file and comparing the resulting similarity matrix to each of the learned Sequence's SDRs from the training phase. The whole sequence was then predicted using the accuracy and observation class (Label), and the accuracy % of the predicted sequences was determined and saved to a file.[6] We used Multi Sequence Learning to predict a sequence of data points with high accuracy. The experiments allowed us to learn about multiple facets of the Neocortex API,[11] such as how encoders work, how the Spatial Pooler produces SDR inputs and works the learning phase, and how the Homeostatic Plasticity controller helps to stabilize the learning process.[14]

To enhance further, we can produce test data as a subset of the synthetic dataset. This ensures that the test dataset is completely a subsequence and matches the accuracy. Run the experiment on the cloud with more inputs to increase the number of subsequences.[16]

VI. References

[1] "NeoCortexApi: https://github.com/ddobric/neocortexapi".

[2] “Numenta https://www.numenta.com”.

[3] Jeff Hawkins, "On Intelligence: How a New Understanding of the Brain Will Lead to the Creation of Truly Intelligent Machines", 2004

[4] Jeff Hawkins and Subutai Ahmad, "Why Neurons Have Thousands of Synapses, A Theory of Sequence Memory in Neocortex", 2016

[5] Subutai Ahmad, "Hierarchical Temporal Memory" - Scholarpedia, 2012

[6] Jeff Hawkins and Dileep George, "Hierarchical Temporal Memory", 2009

[7] Subutai Ahmad, "Real-Time Multimodal Integration in a Hierarchical Temporal Memory Network", 2015

[8] Casarella, J. M. (2011). The Application of Hierarchical Temporal Memory to the ECG Waveforms. Proceedings of Student/Faculty Research Day, Ivan G. Seidenberg School of CSIS, Pace University.

[9] Y. Usui, R. Niiyama and Y. Kuniyoshi, "Anthropomorphic Face Robot having Soft Mouth Mechanism with Embedded Artificial Facial Muscles," 2019 International Symposium on Micro-NanoMechatronics and Human Science, Nagoya, Japan, 2019.

[10] A. Esfandbod, Z. Rokhi, A. Taheri, M. Alemi and A. Meghdari, "Human-Robot Interaction based on Facial Expression Imitation," 2019 7th International Conference on Robotics and Mechatronics, Tehran, Iran, 2019.

[11] D. Fu, F. Abawi and S. Wermter, "The Robot in the Room: Influence of Robot Facial Expressions and Gaze on Human-Human-Robot Collaboration," 2023 32nd IEEE International Conference on Robot and Human Interactive Communication, Busan, Korea, Republic of, 2023.

[12] N. Calvo-Barajas, G. Perugia and G. Castellano, "The Effects of Robot’s Facial Expressions on Children’s First Impressions of Trustworthiness," 2020 29th IEEE International Conference on Robot and Human Interactive Communication, Naples, Italy, 2020.

[13] Chyi-Yeu Lin, Thi Thoa Mac, Li-Wen Chuang, "Real-time artistic human face portrait by humanoid robot", 2009 IEEE Control Applications, (CCA) & Intelligent Control, (ISIC), 2009.

[14] F. Vannucci, G. Di Cesare, F. Rea, G. Sandini, A. Sciutti, "A Robot with Style: Can Robotic Attitudes Influence Human Actions?", 2018 IEEE-RAS 18th International Conference on Humanoid Robots, 2018.

[15] Bishop, L. M., van Maris, A., Dogramadzi, S., & Zook, N. (2019). "Social robots: The influence of human and robot characteristics on acceptance," Paladyn: Journal of Behavioral Robotics, 10(1), 2019.

[16] F. Hara, "Artificial Emotion of Face Robot through Learning in Communicative Interactions with Human," Proceedings of the 13th IEEE International Workshop on Robot and Human Interactive Communication, 2004.