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

Approve Prediction of Multisequence Learning

MD ASHIK IQBAL
ashik.iqbal@stud.fra-uas.de

MAHIDUL ISLAM RANA
mahidul.rana@stud.fra-uas.de

MD FAZLEY RABBE
fazley.rabbe@stud.fra-uas.de

Abstract*— In our research work, we demonstrate the utilization of Multisequence Learning methods to comprehend sequences effectively. This approach contrasts with the prevalent dense arguments employed in current artificial networks, as it aligns more closely with the sparse information processing characteristic of natural networks. Specifically, we employ Hierarchical Temporal Memory, a variant of the Thousands Brain Function Theory, to generate motor instructions facilitating communication with the environment and validation of predictions. To assess the accuracy of our methodology, we focus on analyzing subsequences and developing a robust method for accuracy determination. In support of our findings, we have automated several key operations, including the creation of synthetic datasets tailored to specific settings, storage of datasets to file, dataset scanning, and publication of results to file.

Keywords*— Hierarchical Temporal Memory, Spatial Pooler, Temporal Memory, encoder, sparse distributed representation, Proximal Dendrite Segments, Distal Dendrite Segments

I. INTRODUCTION

Multisequence learning has a broad range of applications across multiple disciplines, such as finance, bioinformatics, natural language processing, and sensor networks. For example, MSL methods have proven crucial in computational biology to understand the intricate regulatory systems controlling genomic sequences, protein interactions, and gene expression. Similar to this, MSL models in financial markets can predict stock prices, spot trading opportunities, and better manage portfolio risks by taking use of the dependencies between various financial time series. In this project, we provide a comprehensive overview of Multisequence Learning, covering its theoretical foundations, algorithmic methodologies, and practical applications. We discuss the key challenges associated with learning from multiple sequences and highlight the unique opportunities afforded by this paradigm. Our Project can learn the previous sequence and predict the future sequence. We have trained the project and got some output It gives good accuracy

II. LITERATURE SURVEY

Hierarchical temporal memory: Hierarchical temporal memory [1] is a machine learning algorithm that simulates the structure and biological functionality of the neocortex and is particularly suitable for sequence learning and prediction. It not only advances our understanding of how the brain may solve the sequence learning problems but also has been applied to various practical implications, such as anomaly detection [2], discrete [3] and continuous sequence modelling [4], face classification [5], handwritten digits recognition [6], and sequence prediction [7]

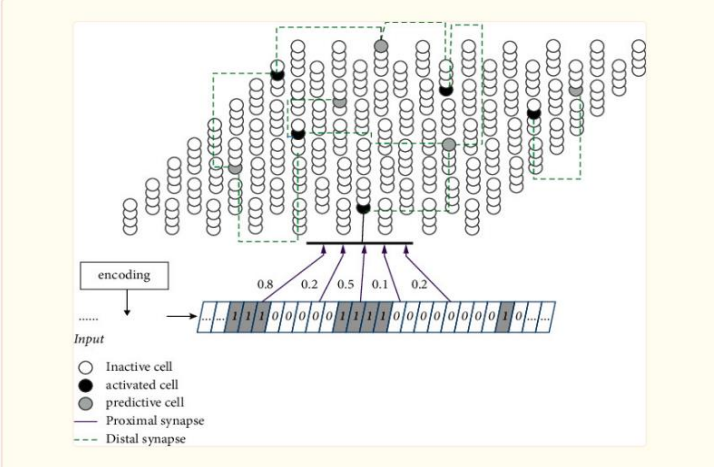


Figure: Hierarchical temporal memory Structure

Proximal Dendrite Segments: The synapses on the proximal dendrites (typically several hundred) have a relatively large effect at the soma and therefore are best situated to define the basic receptive field response of the neuron [8]. If the coincident activation of a subset of the proximal synapses is sufficient to generate a somatic action potential and if the inputs to the proximal synapses are sparsely active, then the proximal synapses will recognize multiple unique feedforward patterns in the same manner as discussed earlier. Therefore, the feedforward receptive field of a cell can be thought of as a union of feedforward patterns.

Distal Dendrite Segments: The basal dendrites of a neuron recognize patterns of cell activity that precede the neuron firing, in this way the basal dendrites learn and store transitions between activity patterns. When a pattern is recognized on a basal dendrite it generates an NMDA spike. The depolarization due to an NMDA spike attenuates in amplitude by the time it reaches the soma, therefore when a basal dendrite recognizes a pattern it will depolarize the soma but not enough to generate a somatic action potential [9,10]. We propose this sub-threshold depolarization is an important state of the cell. It represents a prediction that the cell will become active shortly and plays an important role in network behavior. A slightly depolarized cell fires earlier than it would otherwise if it subsequently receives sufficient feedforward input. By firing earlier it inhibits neighboring cells, creating highly sparse patterns of activity for correctly predicted inputs. We will explain this mechanism more fully in a later section.

Neocortex: Hawkins presented [11] a theoretical framework describing the processes in the neocortex. He describes the neocortex as a highly efficient pattern matching device [12] – as opposed to a computing engine. The brain learns by storing patterns and recognizes by matching incoming sensory data with learned patterns. It can recognize the same pattern under different conditions (invariance) much more efficiently than existing computer based systems. One example is the ability of the brain to recognize a face under different lighting conditions, or from different angles.

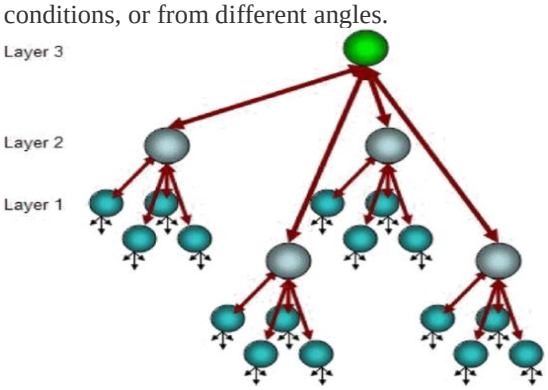
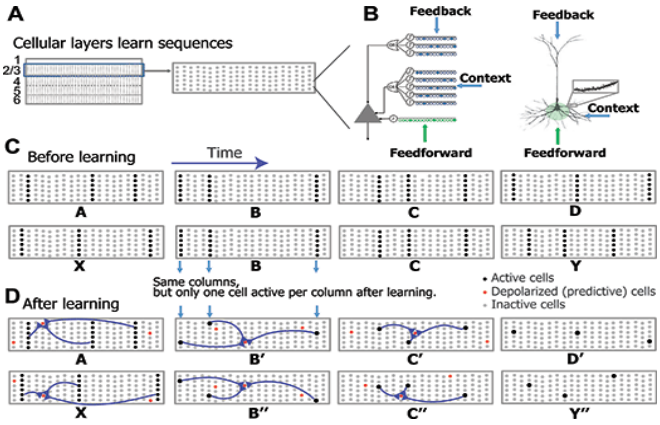


Figure: Neocortex layers

Prediction: After the model is trained, it can be used to make predictions on new or unseen sequences of data. Given a set of input sequences, the model processes the data through its learned architecture and produces predictions for future events or states. The predicted outcomes may include numerical values, class labels, probability distributions, or other relevant representations depending on the specific prediction task.

Memory: Memory mechanisms are incorporated into the model architecture to retain information from past sequences. This memory can take various forms, such as recurrent connections in recurrent neural networks (RNNs), memory cells in long short-term memory networks (LSTMs), or attention mechanisms that selectively focus on relevant past information. These mechanisms enable the model to capture long-term dependencies and contextual information across multiple sequences.

Connection: The cortex is organized into six cellular layers. Each cellular layer consists of a set of minicolumns, with each minicolumn containing multiple cells. (B) An HTM neuron (left) has three distinct dendritic integration zones, corresponding to different parts of the dendritic tree of pyramidal neurons (right). An HTM neuron models dendrites and NMDA spikes as an array of coincident detectors each with a set of synapses. The coactivation of a set of synapses on a distal dendrite will cause an NMDA spike and depolarize the soma (predicted state). (C, D) Learning high-order Markov sequences with shared sub-sequences (ABCD versus XBCY). Each sequence element invokes a sparse set of minicolumns due to intercolumn inhibition. (C) Prior to learning the sequences all the cells in a minicolumn become active. (D) After learning, cells that are depolarized through lateral connections become active faster and prevent other cells in the same column from firing through intracolumn inhibition. The model maintains two simultaneous representations: one at the minicolumn level representing the current feedforward input and the other at the individual cell level representing the context of the input. Because different cells respond to C in the two sequences (C' and C"), they can invoke the correct high-order prediction of either D or Y [13]



Temporal Pooler: The TP aims to discover common sequences of region activity by having each column learn to predict when it will become active, based on the region's activity in preceding time steps. The SDR from the SP represents the sparse features that are currently present in the region's input and this is placed in a temporal context by the TP selecting a combination of cells within active columns to also become active. This allows the TP to distinguish the *same* pattern of active columns in *different* contexts, i.e. a set of n active columns each containing m cells can represent the same pattern in nm different contexts. Active cells then remember the pattern of activity in the previous time step, thus allowing them to predict their own activity and enabling the TP to learn sequences.

Multisequence Learning: An HTM-based system called multisequence learning may simultaneously learn and predict many pattern sequences [14]. Learning is involved in multisequence learning. and making predictions for every pattern sequence utilising different Temporal Memory modules. Ahmad's [15] method relies on learning and anticipating feature sequences at several levels of abstraction through the use of a hierarchical node structure. Each Temporal Memory unit initially assesses and predicts the raw sensory input from a single modality. At higher levels, the nodes comprehend and anticipate patterns that combine information from several modalities.

III. METHODOLOGY

The methodology for prediction using multisequence learning involves a systematic approach to analyzing multiple sequences of data to make accurate predictions. 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

Sequence: A sequence is a type of data format made up of two numbers and a sequence identifier. The list of data sequencing models' components is merely an assortment of

sequences. The subsequence utilizes the same data structure because it is a segment of the sequence. Here is our Sequence below:

```
namespace MultiSequenceLearning
{
    17 references
    public class Sequence
    {
        8 references
        public String name { get; set; }
        12 references
        public int[] data { get; set; }
    }
}
```

Figure: Data model of sequence

```
[
  {
    "name": "S1",
    "data": [ 0, 2, 4, 5, 6, 7, 8, 9, 11, 13, 14, 15, 16, 19 ]
  },
  {
    "name": "S2",
    "data": [ 0, 1, 2, 4, 5, 10, 12, 13, 14, 15, 16, 17, 18, 19 ]
  },
  {
    "name": "S3",
    "data": [ 0, 1, 3, 4, 6, 7, 8, 10, 11, 13, 16, 17, 18, 19 ]
  }
]
```

Figure: Dataset

Encoder: The encoder receives multiple sequences of data as input. These sequences could represent time series data, sequences of events, or any other sequential data relevant to the prediction task. In addition, the encoder consists of one or more layers responsible for encoding the input sequences into a compact representation. These layers could be recurrent neural network (RNN) layers, long short-term memory (LSTM) layers, convolutional neural network (CNN) layers, or transformer-based layers, depending on the characteristics of the data and the task requirements. Each layer captures different aspects of the input sequences, such as temporal dependencies, spatial patterns, or hierarchical structures.

Spatial Pooler: The SP is a sparse feature detector that learns the common spatial patterns in a region's input and, at every time step, produces a sparse distributed representation (SDR) [16] of the current input. It has been extended from the original algorithm [17] into the Augmented Spatial Pooler (ASP) [14] that is capable of handling non-binary input, multiple channels, and is designed to converge onto a stable set of codes without learning being artificially terminated.

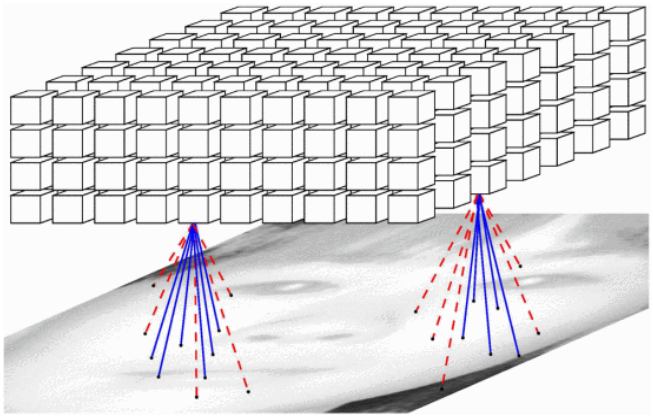


Figure: Spatial Pooler

Sparse Distributed Representation: The HTM network consists of a layer of HTM neurons organized into a set of columns (Fig. 1A). The network represents high-order sequences using a composition of two separate sparse representations. The first representation is at the column level. All neurons within a column detect identical feed-forward input patterns. Each input element is encoded as a sparse distributed activation of columns at any point in time (Fig. 1B, blue arrows). This sparse set of columns is chosen according to a spatial competitive learning rule [15].
The second representation is at the level of individual cells within these columns. After sequence learning (Fig. 1B, bottom), at any given time a distinct subset of cells in the active columns will represent information regarding the learned temporal context of the current pattern. These cells in turn lead to predictions of the upcoming input through lateral projections to other cells within the same network. The predictive state of a cell controls inhibition within a column. If a cell is in the predicted state and later receives sufficient feed-forward input, it will become active faster and inhibit other cells within that column. If there were no cells in the predicted state for an active column, all cells within the column become active.

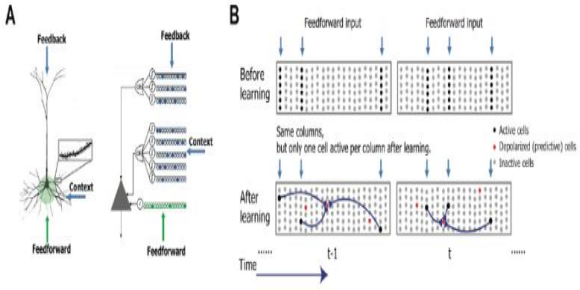


Figure: Sparse Distributed Representation

IV. Implementation

This section will describe how my project learns sequences and predicts any random sequence using the HTM method. It

will automatically read the dataset and predict future elements and give their prediction percentage accuracy.

Data Preparation: Gather your multisequence data. This could be sequences of numbers representing different time-series data or any other form of multisequence data. Preprocess the data as necessary. This may involve normalization, scaling, or encoding the data into a format suitable for HTM processing, such as sparse distributed representations (SDRs).

Model Configuration: Describe your HTM model's architecture. The number of layers, the size of each layer, and the connectivity between levels must all be specified. Select the right values for the HTM model's spatial pooler and temporal memory components. The model's capacity to identify and learn from patterns in the input data will be influenced by these factors.

Initialization: Set the HTM model's other parameters and synaptic connections to their initial values. Random initialization or initialization based on past knowledge of the data may be used in this step.

Training Loop: Run through your multisequence data while the training loop is running For every dataset sequence: Transform the sequence into an input representation that the HTM model can understand. Provide the HTM model with the encoded input. Permit the model to update its internal state as it processes the input sequence. You can also use reinforcement cues or feedback to help direct the learning process. Adjust the model's parameters in light of any feedback signals and the observed input sequences.

Evaluation: Continually assess the HTM model's performance during training. Calculating metrics like forecast accuracy, anomaly detection rates, or any other pertinent measurements might be part of this. Verify whether the model can adjust to changes in the input distribution over time or generalize to sequences that have never been seen before.

Hyperparameter Tuning: Adjust the HTM model's parameters in light of empirical findings and learnings via trial and error. This could entail modifying model architectural parameters, learning algorithms, or regularization strategies.

Monitoring and Debugging: Observe how the HTM model behaves when it is being trained. This entails assessing its forecasts, monitoring alterations in internal states, and identifying any problems that may emerge. Debug any errors that arise during training, such as convergence problems, bursting gradients, or vanishing gradients.

Iterative Improvement: Refine the training procedure in light of empirical findings and learnings from trial and error. Improve the HTM model's performance on multisequence learning tasks by continuously refining both the model and the training process.

V. Results

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.

```
Input: 5
Predicted Sequence: S2_12 - Predicted next element: 10
Input: 6
Predicted Sequence: S3_8 - Predicted next element: 7
Input: 7
Predicted Sequence: S3_10 - Predicted next element: 8
Accuracy for T1 sequence: 66.66666666666666%
Input: 6
Predicted Sequence: S3_8 - Predicted next element: 7
Input: 11
Predicted Sequence: S3_16 - Predicted next element: 13
Input: 12
No Prediction
Accuracy for T2 sequence: 0%
Input: 1
Predicted Sequence: S3_4 - Predicted next element: 3
Input: 2
No Prediction
Input: 3
Predicted Sequence: S2_5 - Predicted next element: 4
Accuracy for T3 sequence: 33.33333333333333%
Input: 3
Predicted Sequence: S2_5 - Predicted next element: 4
Input: 4
Predicted Sequence: S3_7 - Predicted next element: 6
Input: 7
Predicted Sequence: S1_9 - Predicted next element: 8
Input: 8
Predicted Sequence: S3_11 - Predicted next element: 10
Accuracy for T4 sequence: 75%
```

VI. Discussion

Future research and improvements will target the exploration of Hierarchical Temporal Memory (HTM) system scalability and efficiency in handling more complex sequence predictions and larger datasets [18]. Refinement of the algorithm aims to enhance generalization across various data types, including highly challenging unstructured data, by leveraging more universally applicable methods known to the field. An intriguing area of further exploration involves integrating HTM with other machine learning models to create hybrid systems that exploit the strengths of both methods, ultimately achieving superior performance [19].

Additionally, deriving new encoding strategies to effectively represent a wide range of data types within HTM systems is crucial. The application of HTM in real-world scenarios such as predictive maintenance, natural language processing, and anomaly detection in network security will also be investigated [20]. These endeavors will involve defining benchmarks and evaluating the framework of HTM systems against current state-of-the-art models within their respective domains. To further enhance research efforts, generating test data as a subset of synthetic datasets ensures that the test dataset is a complete subsequence, maintaining accuracy [21]. Experimentation can be conducted on cloud platforms with increased inputs to expand the number of subsequences.

VII. References

1. Chen X., Wang W., Li W. An overview of Hierarchical Temporal Memory: a new neocortex algorithm. Proceedings of the In 2012 Proceedings of International Conference on Modelling, Identification and Control; January 2012; Wuhan, China. IEEE; pp. 1004–1010
2. aylidyonok I., Frolenkova A., Panov A. I. *Biologically Inspired Cognitive Architectures Meeting* . Cham: Springer; 2018. Extended hierarchical temporal memory for motion anomaly detection; pp. 69–81.
3. Irmanova A., James A. P. HTM sequence memory for language processing. Proceedings of the In Poster Session Presented at IEEE International

Conference on Rebooting Computing (ICRC); November 2017; Washington, DC, USA.

4. al K., Bhattacharya S., Dey S., Mukherjee A. Modelling HTM learning and prediction for robotic path-learning. Proceedings of the In 2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob); August 2018; Enschede, Netherlands. IEEE; pp. 839–844.
5. Krestinskaya O., Ibrayev T., James A. P. Hierarchical temporal memory features with memristor logic circuits for pattern recognition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* . 2017;**37**(6):1143–1156.
6. Fan D., Sharad M., Sengupta A., Roy K. Hierarchical temporal memory based on spin-neurons and resistive memory for energy-efficient brain-inspired computing. *IEEE Transactions on Neural Networks and Learning Systems* . 2015;**27**(9):1907–1919.
7. Struye J., Latré S. Hierarchical temporal memory and recurrent neural networks for time series prediction: an empirical validation and reduction to multilayer perceptrons. *Neurocomputing* . 2020;**396**:291–301.
8. Spruston, N. (2008). Pyramidal neurons: dendritic structure and synaptic integration. *Nat. Rev. Neurosci.* 9, 206–221.
9. Antic, S. D., Zhou, W. L., Moore, A. R., Short, S. M., and Ikonomu, K. D. (2010). The decade of the dendritic NMDA spike. *J. Neurosci. Res.* 88, 2991–3001.
10. Major, G., Larkum, M. E., and Schiller, J. (2013). Active properties of neocortical pyramidal neuron dendrites. *Annu. Rev. Neurosci.* 36, 1–24. doi: 10.1146/annurev-neuro-062111-150343
11. J. Hawkins and S. Blakeslee. "On Intelligence," Times Books, Henry Holt and Company, New York, NY 10011, Sept 2004
12. J. Hawkins. "Learn Like A Human." in IEEE Spectrum, April 2007.
13. Yuwei Cui, Subutai Ahmad, Jeff Hawkins; Continuous Online Sequence Learning with an Unsupervised Neural Network Model. *Neural Comput* 2016; 28 (11): 2474–2504.
14. J. Thornton and A. Srbic, "Spatial pooling for greyscale images", *International Journal of Machine Learning and Cybernetics*, vol. 4, no. 3, pp. 207-216, 2013.
15. Subutai Ahmad, "Real-Time Multimodal Integration in a Hierarchical Temporal Memory Network", 2015
16. B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by VI?", *Vision Research*, vol. 37, no. 23, pp. 3311-3325, 1997.
17. J. Hawkins, S. Ahmad and D. Dubinsky, "Hierarchical temporal memory including HTM cortical learning algorithms", *Tech. Rep.*, 2011
18. 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.
19. 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.
20. 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.
21. 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.