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— Current artificial networks rely heavily on dense arguments, whereas natural networks rely on sparse information. Hierarchical Temporal Memory is a variant of the Thousands Brain Function Theory that produces motor instructions for communicating with the surroundings and testing predictions. In our research work, we show how to understand a sequence using Multisequence Learning methods. A subsequence is utilized for determining accuracy and the method for doing so. To support our findings, several operations have been automated, such as building a synthetic dataset based on the settings, storing the dataset to file, scanning the dataset, and publishing the results to file.

Keywords— Hierarchical Temporal Memory, Spatial Pooler, Temporal Memory, encoder, sparse distributed representation, AI, ML.

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.

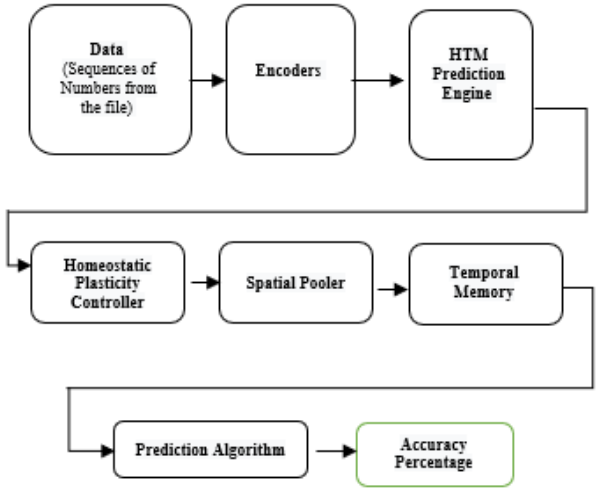

Figure 1: Overview of Sequence Learning

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. 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].

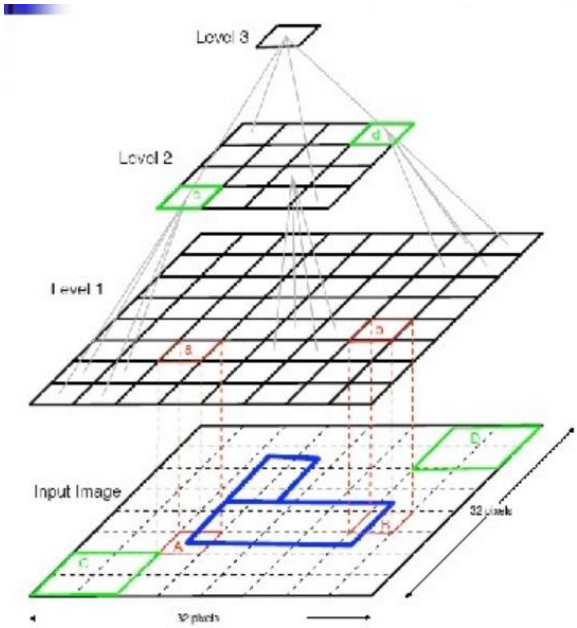

Figure 2: Hierarchical Temporal Memory Structure [8]

B. *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.

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].

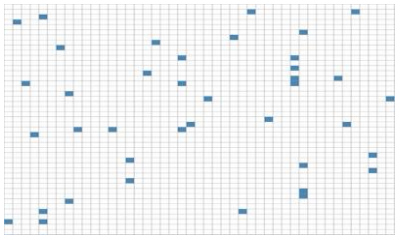

Figure 3: represents a sparse distributed representation.

C. *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.

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].

D. *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. 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].

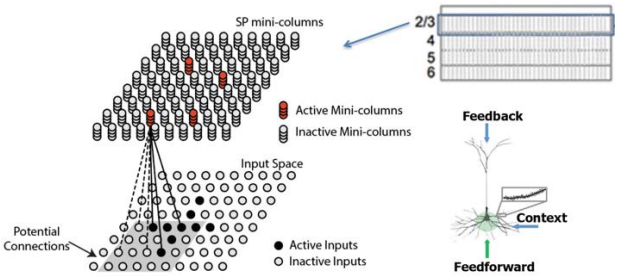

Figure 4: Representation of spatial pooler

E. *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.

F. *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.

```
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

```
[
  {
   "name": "T1",
   "data": [ 1, 2, 4 ]
  },
  {
   "name": "T2",
   "data": [ 2, 3, 4 ]
  },
  {
   "name": "T3",
   "data": [ 4, 5, 7 ]
  },
  {
   "name": "T4",
   "data": [ 5, 8, 9 ]
  }
]
```

Dataset 2: Sample subsequence

B. FetchHTMConfig method

Here we save the HTM-Config which is used for Hierarchical Temporal Memory to Connections.

```
/// <summary>
/// HTM Config for creating Connections
/// </summary>
/// <param name="inputBits">input bits</param>
/// <param name="numColumns">number of
columns</param>
/// <returns>Object of HTMConfig</returns>
public static HtmConfig FetchHTMConfig(int
inputBits, int numColumns)
{
   HtmConfig cfg = new HtmConfig(new int[] {
inputBits }, new int[] { numColumns })
   {
      Random = new ThreadSafeRandom(42),

      CellsPerColumn = 25,
      GlobalInhibition = true,
      LocalAreaDensity = -1,
      NumActiveColumnsPerInhArea = 0.02 *
numColumns,
      PotentialRadius = (int)(0.15 * inputBits),
      MaxBoost = 10.0,
      DutyCyclePeriod = 25,
      MinPctOverlapDutyCycles = 0.75,
      MaxSynapsesPerSegment = (int)(0.02 *
numColumns),
      ActivationThreshold = 15,
      ConnectedPermanence = 0.5,e.
      PermanenceDecrement = 0.25,
      PermanenceIncrement = 0.15,
      PredictedSegmentDecrement = 0.1,
   };

   return cfg;
}
```

Source code 2: FetchHTMConfig Method

This capability allows for the creation of multiple configurations and the training of various models. This allows you some flexibility in scaling up the experiment [11].

C. GetEncoder Method

We have used a Scalar Encoder [1] since we are encoding all numeric values only.

```
/// <summary>
/// Get the encoder with settings
/// </summary>
/// <param name="inputBits">input bits</param>
/// <returns>Object of EncoderBase</returns>
public static EncoderBase GetEncoder(int inputBits)
{
      double max = 20;

      Dictionary<string, object> settings = new
Dictionary<string, object>()
      {
      { "W", 15},
      { "N", inputBits},
      { "Radius", -1.0},
      { "MinVal", 0.0},
      { "Periodic", false},
      { "Name", "scalar"},
      { "ClipInput", false},
      { "MaxVal", max}
      };

      EncoderBase encoder = new
ScalarEncoder(settings);

      return encoder;
}
```

Source code 3: Implementation of Scalar Encoder method

D. ReadDataset method

When a full path is provided, the JSON file is read, and the object of the Sequence data model's list is returned.

```
/// <summary>
/// Reads dataset from the file
/// </summary>
/// <param name="path">full path of the file</param>
/// <returns>Object of list of Sequence</returns>
public static List<Sequence> ReadDataset(string path)
{
      Console.WriteLine("Reading Sequence...");
      String lines = File.ReadAllText(path);

      List<Sequence> sequence =
System.Text.Json.JsonSerializer.Deserialize<List<Sequ
ence>>(lines);

      return sequence;
}
```

Source code 4: Method to read dataset file

E. CreateDataset method

We improved the process to create datasets automatically, saving us time. We build a dataset containing parameters such as the number of sequences to be formed, the size of a sequence, the startVal (perhaps the start range), and the endVal.

```
/// <summary>
/// Creates list of Sequence as per configuration
/// </summary>
/// <returns>Object of list of Sequence</returns>
public static List<Sequence> CreateDataset()
{
      int numberOfSequence = 3;
      int size = 12;
      int startVal = 0;
      int endVal = 15;
      Console.WriteLine("Creating Sequence...");
      List<Sequence> sequence =
HelperMethods.CreateSequences(numberOfSequence,
size, startVal, endVal);

      return sequence;
}
```

Source code 5: Create a list of sequences as per configuration.

Note that endVal should be less than equal to MaxVal of Scalar Encoder used above.

F. Calculate accuracy in the PredictNextElement method

```
int matchCount = 0;
int predictions = 0;
double accuracy = 0.0;

foreach (var item in list)
{
   Predict();
   //compare current element with prediction of previous
element
   if(item == Int32.Parse(prediction.Last()))
   {
      matchCount++;
   }
   predictions++;
   accuracy = (double)matchCount / predictions * 100;
}
```

Source code 6: Calculate accuracy.

G. SaveDataset method

Saves the dataset in the dataset director of the Base Path of the application where it is running.

```
/// <summary>
/// Saves the dataset in 'dataset' folder in
BasePath of application
/// </summary>
/// <param name="sequences">Object of list
of Sequence</param>
/// <returns>Full path of the
dataset</returns>
public static string
SaveDataset(List<Sequence> sequences)
{
      string BasePath =
AppDomain.CurrentDomain.BaseDirectory;
      string reportFolder =
Path.Combine(BasePath, "dataset");
      if (!Directory.Exists(reportFolder))

Directory.CreateDirectory(reportFolder);
      string reportPath =
Path.Combine(reportFolder,
$"dataset_{DateTime.Now.Ticks}.json");

      Console.WriteLine("Saving dataset...");

      if (!File.Exists(reportPath))
      {
      using (StreamWriter sw =
File.CreateText(reportPath))
      {
```

Source code 7: Saves the dataset in JSON file.

IV. 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.

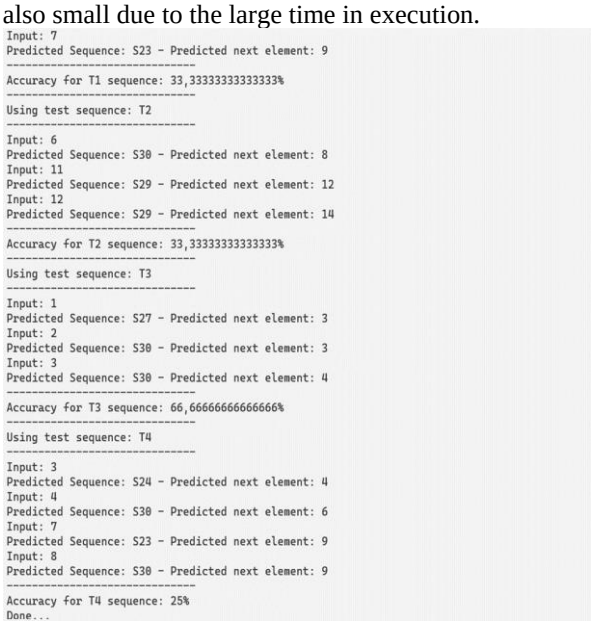

Figure 5: Prediction and calculation of accuracy on subsequence

V. DISCUSSION

The exploration of Hierarchical Temporal Memory (HTM) system scalability and efficiency in handling more complex sequence predictions and larger datasets will be the target of future research and improvements [12]. Further work will refine the algorithm with the aim of better generalizing across different types of data, including particularly highly difficult unstructured data, more generally applicable methods are known to the field. A further interesting area of work will then be the integration of HTM with other machine learning models to make the hybrid system exploit, the strengths of both methods and achieve better performance [13]. It also becomes very important to derive new strategies of encoding that properly define the representation of a wide variety of data types within HTM systems. Similarly, it will focus on investigating the application of HTM in real-world problems such as predictive maintenance, natural language processing, and anomaly detection in network security [14]. All of these will include the definition of benchmark and the framework of HTM systems evaluation versus current models in the state of the art within the context of their respective domains.

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.