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

Investigate influence of parameter MaxNewSynapseCount

Akash Shah
akash.saha@stud.fra-uas.de

Md Fathir Ahmed Shishir
md.shishir@stud.fra-uas.de

***Abstract*—Hierarchical Temporal Memory (HTM) is the consider as new era implementation of machine learning algorithm. This algorithm is based on neocortex of human brain. Our human brain is very complex system which learns and understand in different way as compared any other computation learning algorithm which have been implemented. HTM is specifically designed to learn as if brain is computing input from a sensory organ. These sensory organs take real world input and encode it in sparse distributed representation (SDR). This representation is later feed to HTM algorithm. A single SDR represents multiple properties of single input. While learning happens we do not want to give rise to problem which traditional machine learning algorithms have such as overfitting, zero filling null values, working with mean which misleads outliers in the input and hence HTM algorithm have different sort of parameter such as sparsity, learning rate, inhibition rate and so on. One of the parameters called as MaxNewSynapseCount is responsible for growth of synapses while learning takes place. We have tried to analyse the influence of MaxNewSynapseCount parameter in**

Keywords— Machine Learning, neural network, artificial intellegence, hierarchical temporal memory, sparse distributed representation, encoders, spatial pooler, sequence learing, temporal memory.

I. INTRODUCTION

In this experiment we are using HTM based algorithm called Multisequence Learning algorithm. This algorithm is implemented and taken from neocortexapi [1]. This algorithm learning the pattern of input given which is mapped to a real world object and then it can predict this object when a similar patter is given. As mentioned, Multisequence Learning is based on HTM which needs to be configured with various parameters. One of the parameters called MaxNewSynapseCount is responsible for dynamic growth of synapse during the learning process. During the experiment, we have tried to analyse the influence of this parameter and create a report.

II. LITERATURE SURVEY

A. Hierarchical Temporal Memory

Hierarchical Temporal Memory (HTM) is a modern world machine learning model. Its adaptable and biologically faithful framework makes it suitable for solving a wide range of data-driven problems, including prediction, classification, and anomaly detection. To feed data into HTM systems, Sparse Distributed Representations (SDRs) are required, which differ significantly from conventional computer representations like ASCII for text. SDRs are encoded directly into the representation and comprise mostly zeros, with only a few ones. Each one-bit holds a unique semantic meaning, and if two SDRs have several overlapping one-bits, they are considered to have similar meanings [5].

HTM can be considered as a type of neural network, with interconnected biological neurons forming layers of nodes or neurons. The output from one layer is passed on to the next layer, and the connections between them can be adjusted using a bias. However, HTM is structured hierarchically, much like the neocortex of the human brain. Each column in HTM contains several cells that can either be active or inactive, depending on the input received in the form of SDRs.

B. Sparse Distributed Representation

Sparse distributed representation (SDR) is a method of data encoding that aims to represent information using sparse patterns of activity in a high-dimensional space. In SDR, only a small subset of the elements in a vector are active, while the rest are inactive or zero. This type of encoding has been shown to be efficient, flexible, and robust, and has applications in various fields, including neuroscience, artificial intelligence, and information theory [2]. SDRs can handle missing and noisy data, and can generalize to new patterns that are similar to learned patterns.

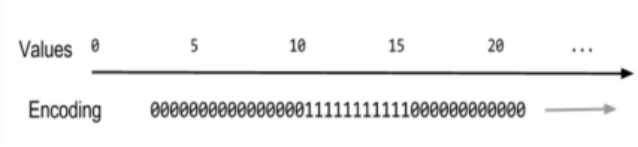

Figure 1: SDR of a number

C. Spatial Pooler

The Spatial Pooler is responsible for creating a sparse distributed representation of the input data, which is then fed into the Temporal Memory component for further processing.

The Spatial Pooler works by creating a set of columns, each of which represents a potential input feature [3]. The input data is then mapped onto these columns, with each input feature activating a subset of the columns. The Spatial Pooler then selects a small subset of the active columns to represent the input data, using a process known as inhibition.

The inhibition process involves selecting a fixed percentage of the most highly activated columns and suppressing the activation of all other columns. This results in a sparse distributed representation, where only a small subset of the columns is active at any given time.

D. Temporal Memory

The temporal memory component of HTM is responsible for learning and predicting sequences in time-varying input data.

Temporal Memory works by maintaining a set of cells that represent patterns of activity in the input data. These cells are connected to each other through synapses, which can strengthen or weaken over time based on the input data patterns. As the input data sequences are presented to the temporal memory, the cells become active in a sparse and distributed manner, forming a representation of the input sequence.

When a new input sequence is presented to the temporal memory, it compares the sequence to the patterns it has learned and predicts the next likely pattern based on the sequence's context.

E. Encoder

An encoder is a technique that converts unprocessed input data into a sparsely dispersed depiction, which can be handled by the Spatial Pooler component [4]. The encoder's task is to transform continuous input data into a collection of distinct categories, forming a representation that is resistant to noise and can adapt to novel patterns.

In HTM, various encoder types are available, which include scalar encoders, category encoders, and datetime encoders. Among these, scalar encoders are the most frequently employed type and are utilized to encode continuous data like temperature values, sensor readings, and audio signals. On the other hand, category encoders are employed to encode discrete data such as words, colors, or symbols.

III. METHODOLOGY

Most of the time in this experiment was spent on understanding and experimenting on Multisequence Learning. The Multisequence Learning is based on HTM Classifier taken from the NeoCortexApi [1].

In this algorithm, the connections of taken as per HTM Configuration. The Spatial Pooler creates a sparse representation of encoded data, while the Temporal Memory is used to remember the sparse representation. Additionally, there is a Cortex Layer and an HTM Classifier that receives the trained model and predicts the outcome.

Following is the pseudo code for Multisequence Learning:

01. Get HTM Config and initialize memory of Connections.
02. Initialize HTM Classifier and Cortex Layer
03. Initialize HomeostaticPlasticityController
04. Initialize memory for Spatial Pooler and Temporal Memory
05. Add Spatial Pooler memory to Cortex Layer
05.01 Compute the SDR of all encoded segment for multi-sequences using Spatial Pooler
05.02 Continue for maximum number of cycles.
06. Add Temporal Memory to Cortex Layer
06.01 Compute the SDR as Compute Cycle and get Active Cells
06.02 Learn the Label with Active Cells
06.03 Get the input predicted values and update the last predicted value depending upon the similarity.
06.04 Reset the Temporal Memory
06.05 Continue all above steps for sequences of multi-sequences for maximum cycles.
07. Get the trained Cortex Layer and HTM Classifier

A. Identifing use of MaxNewSynapseCount

After going through the project, we found that MaxNewSynapseCount is being stored in HTMConfig and used in TemporalMemory (TM). In TM, ActivatePredictedColumn() and BurstColumn() which in the end calls GrowSynapses() depending on the condition of Active Segment and Matching Segments.

B. Logger in Temporal Memory

Added logger string in TemporalMemory.cs and log the calls for GrowSynapses() and by which condition it is called.

```
public List<String> Logger { get; set; }
public int countLogger { get; set; }
:        :           :
:        :           :

//in GrowSynapses()
this.Logger.Add($"GrowSynapses(): calledBy: {calledBy},
requiredNewSynapses: {requiredNewSynapses},
numMissingSynapses: {numMissingSynapses}");
this.countLogger++;
//use above variable as per each cycle is changes
```

Figure 2: Adding and updating logger variable

C. *Report*

The Report is a data model which is updated per cycle of learning loop and saves sequence name with logs from temporal memory and accuracy per cycle.

D. *Analysis*

In Analysis data model we create a extracted model out of Report data model which stores the cycle sequence name and number of call made by which condition and number of synapses increased in that cycle.

IV. RESULTS

This experiment has been tried to run as many times as possible and find our optimal configuration and trace the changes.

The reports and analysis is store in base path of app domain (AppDomain.CurrentDomain.BaseDirectory) which has reports and analysis directory having all the outputs.

Furthermore,

1. MaxNewSynapseCount value changes in ActivatePredictedColumn() if we have active column and active segment and the positive difference in MaxNewSynapseCount and active potential synapse is addition of synapse

2. MaxNewSynapseCount value changes in BurstColumn() if we have active column and no active segment and the positive difference in MaxNewSynapseCount and last active potential synapse is addition of synapse

3. The accuracy of each cycle changes (goes up usually) even if we do not add any new synapses, but it does not assure that adding synapses with increase the accuracy.

A. *Based of Reports data model*

The report shows all the calls made for calling GrowSynapses() and the number of syanpses added:

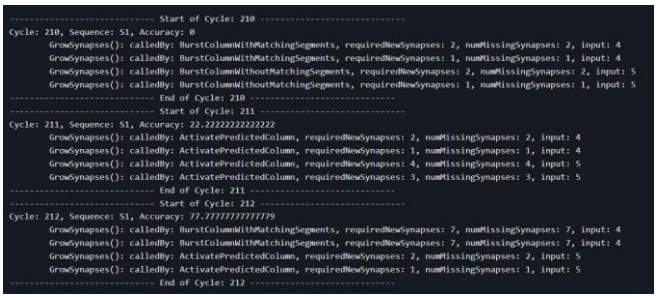

Figure 3: Slice from the report generated

1. calledBy: ActivatePredictedColumn

It means that there were some Active Segments and even though there were some Active Segments, new synapses can be added that connect previously active cells with the segment in ActivatePredictedColumn(). If the difference b/w MaxNewSynapseCount and last potential actice cells is positive, we grow that many synapses.

2. calledBy: BurstColumnWithMatchingSegments

It means that there were no Active Segments and we need to active random cell in BurstColumn(). We adapt segment and check for matching segment and if the positive difference b/w MaxNewSynapseCount and last potential active synapse is positive we grow that many synapses.

3. calledBy: BurstColumnWithoutMatchingSegments

It means that there were no Active Segments and we need to active random cell in BurstColumn(). We adapt segment and if no matching segment then we calcuate the minimum b/w MaxNewSynapseCount and previous winner cell count to grow the synapses

B. *Based of Analysis data model*

We save number of calls made by ActivatePredictedColumn or BurstColumnWithMatchingSegments or BurstColumnWithMatchingSegments and the number of synapses increased for all sequence per cycle. The following image shows visual representation of change happening for a sequence in all its cycles. These are saved as CSV files.

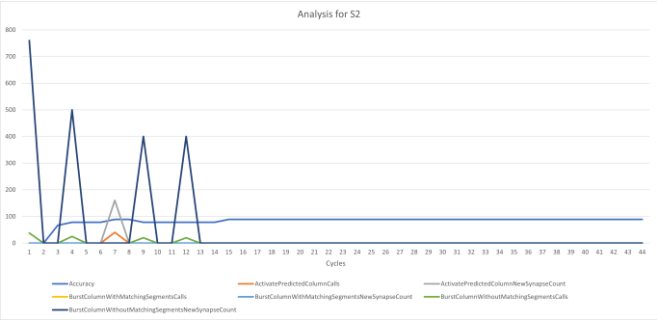

Figure 4: Analysis of a sequence

V. DISCUSSION

It is always seen that synapse are added in initial cycles of the learning later of the count goes down. Also if the algorithm is learning a new sequence if adds some new synapse but not as much as it added when it started learning in very first cycle. There is no guarantee that increasing in synapse count will increase the accuracy of upcoming cycle. Also, it is nature of the temporal memory that after couple of cycle if has already extracted the pattern.

VI. REFERENCES

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

[2] Jeff Hawkins, "On Intelligence" (2004)

[3] Subutai Ahmad and Jeff Hawkins, "A Review of Hierarchical Temporal Memory", 2016

[4] Matthew K. Graham, William C. Gross, Subutai Ahmad, and Jeff Hawkins, "Real-Time Anomaly Detection in Streaming Sensor Data Using Hierarchical Temporal Memory", 2019