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

Investigate Label Prediction using time-series sequence

Amar Arvindkumar Singh
amar.singh@stud.fra-uas.de

Chandan Dev Singh
chandan.singh@stud.fra-uas.de

Mandar Anil Patkar
mandar.patkar@stud.fra-uas.de

***Abstract*—Most artificial networks today depend upon dense representations, whereas biological networks rely on sparse representations. The term hierarchical temporal memory describes a specific realization of the thousands Brain Theory, and it generates motor commands to interact with the surroundings and test the predictions. In the time-series, datetime is converted to a fixed time frame as segment. In this paper we show how we encode segment as Sparse Distributed Representations (SDRs) for use in HTM systems. It uses the field of AI to predict the label according to the time-series sequence. The task here is to implement sequence learning and prediction code that learns the sequence. The date and time are to be encoded in order to be used in spatial pooler and HTM. The temporal memory, then learns the multi-sequence learning according to the date and time as well as the power consumed at that specific segment.**

Keywords—AI, HTM, SDRs, spatial pooler, time-series sequence.

I. INTRODUCTION

In nature there are many events which occur periodically, here we are trying to identify such time-series sequence which is associated entity/label and the end goal is to learn time-series with label and predict the label at given time.

The neocortex is the seat of intelligent thought in the mammalian brain. High level vision, hearing, touch, movement, language, and planning are all performed by the neocortex. Given such a diverse suite of cognitive functions, you might expect the neocortex to implement an equally diverse suite of specialized neural algorithms. This is not the case. The neocortex displays a remarkably uniform pattern of neural circuitry. The biological evidence suggests that the neocortex implements a common set of algorithms to perform many different intelligence functions [1].

The neocortex continually processes an endless stream of rich sensory information. It does this remarkably well, better than any existing AI computer. A wealth of empirical evidence demonstrates that cortical regions represent all information using sparse patterns of activity. To function effectively throughout a lifetime these representations must have tremendous capacity and must be extremely tolerant to noise. However, a detailed theoretical understanding of the capacity and robustness of cortical sparse representations has been missing [2] .

HTM provides a theoretical framework for understanding the neocortex and its many capabilities. To date we have implemented a small subset of this theoretical framework. Over time, more and more of the theory will be implemented. Today we believe we have implemented a sufficient subset of what the neocortex does to be of commercial and scientific value [1].

II. LITERATURE SURVEY

A. *Sequence*

A particular order in which related things follow each other.

B. *Time-series*

Sequence of event that occur in given period of time

C. *Label*

The time-series which is associated with an event is called label i.e. power consumption (in KWh). Power consumption, we are using data set of gym's power consumption on hourly basis.

D. *HTM*

Hierarchical Temporal Memory (HTM) provides a flexible and biologically accurate framework for solving prediction, classification, and anomaly detection problems for a broad range of data types. HTM systems require data input in the form of Sparse Distributed Representations (SDRs). SDRs are quite different from standard computer representations, such as ASCII for text, in that meaning is encoded directly into the representation. An SDR consists of a large array of bits of which most are zeros, and a few are ones. Each bit carries some semantic meaning so if two SDRs have more than a few overlapping one-bits, then those two SDRs have similar meanings [3].

HTMs can be viewed as a type of neural network. By definition, any system that tries to model the architectural details of the neocortex is a neural network. However, on its

own, the term “neural network” is not very useful because it has been applied to a large variety of systems. HTMs model neurons (called cells when referring to HTM), which are arranged in columns, in layers, in regions, and in a hierarchy. The details matter, and in this regard HTMs are a new form of neural network [1].

E. Sparse Distributed Representations (SDRs)

Empirical evidence shows that every region of the neocortex represents information using sparse activity patterns made up of a small percentage of active neurons, with the remaining neurons being inactive. An SDR is a set of binary vectors where a small percentage of 1s represent active neurons, and the 0s represent inactive neurons. The small percentage of 1s, denoted the *sparsity*, varies from less than one percent to several percent. SDRs are the primary data structure used in the neocortex and used everywhere in HTM systems. There is not a single type of SDRs in HTM but distinct types for various purposes [4].

While a bit position in a dense representation like ASCII has no semantic meaning, the bit positions in an SDR represent a particular property. The semantic meaning depends on what the input data represents. Some bits may represent edges or big patches of color; others might correspond to different musical notes. Figure 1 shows a somewhat contrived but illustrative example of an SDR representing parts of a zebra. If we flip a single bit in a vector from a dense representation, the vector may take an entirely different value. In an SDR, nearby bit positions represent similar properties. If we invert a bit, then the description changes but not radically [4].

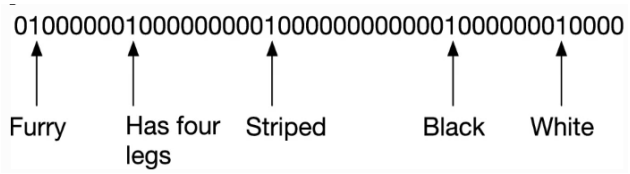

Figure 1 : Encoded data for an animal

III. METHODOLOGY

In the experiment, the data set for power consumption was refined for any error and convert to segments. The datetime segments where then encoded using Scalar Encoder. The encoded data was then trained using algorithm called Multi-sequence Learning. Detailed explanation is given below.

Two methods were used to create refined segments as part of the experiment.

A. Datatset Preprocessing

The dataset consists of two columns called timestamp and power consumption (in KWh). The data was for taken down for a time period of six months at an interval of one hour. Slice of the used data set can be seen in Figure 2. The whole dataset has total of 4416 rows.

```
1   timestamp,consumption
2   datetime,float
3   7/2/10 0:00,21.2
4   7/2/10 1:00,16.4
5   7/2/10 2:00,4.7
6   7/2/10 3:00,4.7
7   7/2/10 4:00,4.6
8   7/2/10 5:00,23.5
9   7/2/10 6:00,47.5
10  7/2/10 7:00,45.4
11  7/2/10 8:00,46.1
```

Figure 2 : Dataset

The datetime in the timestamp was not formatted and did not have proper formatting. Figure 3 show the transformation from raw data to segmentation of datetime.

Figure 3 : Segment of dataset

The segments of datetime are done as per one hour time frame as shown in Figure 3.

B. Creating multiple sequences

For the experiment, whole dataset was not taken into consideration. Only a part of data was taken which is 744 rows out of 4416 rows. There are three ways or configuration in which the time-series can be taken up and create multiple sequences which are by day, by week, by month. If considered a sequence for a single day then 24 segment create a time-series.

1) By day

When the configuration of “by day” is chosen on total segments of 744 in which each time segment is of 1 hour then results into 24 segments of each sequence and total of 31 sequences. This configuration is used for our experiment.

2) By week

When the configuration of “by week” is chosen on total segments of 4416 in which each time segment is of 1 hour then results into 168 segments of each sequence and total of 26 sequences.

3) By month

When the configuration of “by day” is chosen on total segments of 4416 in which each time segment is of 1 hour then results into 720 segments of each sequence and total of 6 sequences.

C. *Encode datetime/segment*

Encoding is done using Scalar Encoder of NeoCortexApi. For segmenting there are two methods as show below to create and reformat.

1) *Method 1*

In this method for experiment, the segmented data consists of date, month, year, and hour (as segment). Sample of this method is shown in Figure 4.

Figure 4 : Segments as per method 1

2) *Method 2*

In this method for experiment, the segmented data consists of date, month, hour, and day of the week (as segment). Sample of this method is shown in Figure 3.

Following is the encoded output for the shown datetime in Figure 5.

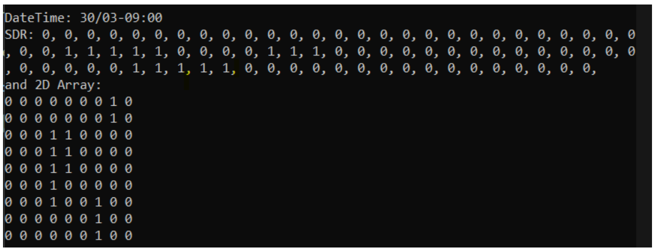

Figure 5 : Encoded using Scalar Encoder for datetime 30/03-09:00

D. *Multisequence Algorithm*

The Multisequence Learning is based on HTM Classifier taken from the NeoCortexApi. In this algorithm, the connections of taken as per HTM Configuration. These configurations are responsible for the mapping to actual cortex of human brain. Spatial Pooler is used create a sparse representation of encoded data and Temporal Memory is used to remember the sparse representation. There is also a Cortex Layer and HTM Classifier which gets trained model and predict the label.

Following is the algorithm for Multisequence Algorithm.

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

E. *Predict the model*

The object of Cortex Layer to compute the SDR (Sparse Distributed Representation). The object of HTM Classifier to predict possible value.

IV. RESULTS

In this experiment, total 40 times the project was ran to get and check on the results. On the X-axis the number of runs is mentioned. On the Y-axis there is maximum percentage of accuracy. The results are shown in Figure 6.

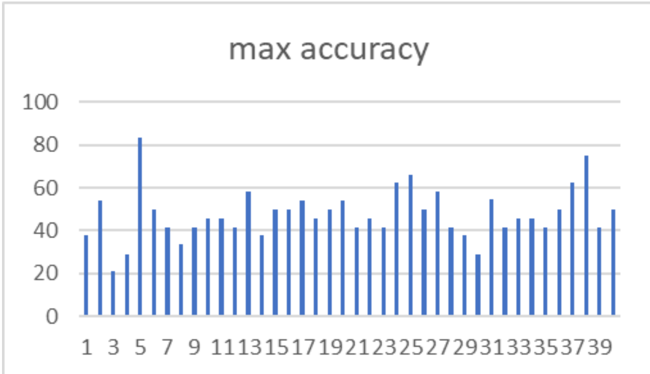

Figure 6 : Results for the experiments

Figure 7 shows the final output from one of the outputs.

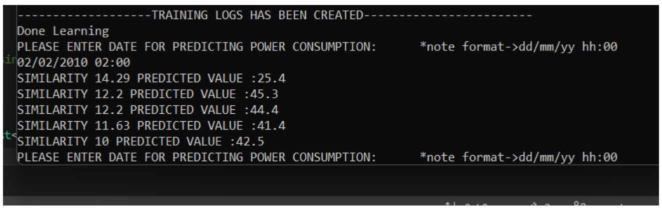

Figure 7 : Prediction output

V. DISCUSSIONS

Learning and predicting time-series have been experimented for decades and various methods have been used. Adapting an existing method for large and noisy data remains a challenge. In the experiment, HTM Classifier is used which is a recently developed neural network based on cortex of human brain and not just a single neuron model.

But there were some hardware limitations running on local machine. To solve this in better way the road of using cloud can be taken to scale up the learning process.

Low accuracy is seen due to small size of dataset has been consider in the experiment. As seen while the run 744 segment took around 35 to 38 minutes to be trained. Also, long runtime was seen when more cycles where used.

VI. REFERENCES

[1] J. Hawkins, “numenta.com,” Numenta, 12 September 2011. [Online]. Available: https://numenta.com/neuroscience-research/research-publications/papers/hierarchical-temporal-memory-white-paper/. [Accessed 22 March 2022].

[2] S. A. &. J. Hawkins, "arxiv.org," Arxiv, 25 March 2015. [Online]. Available: https://arxiv.org/abs/1503.07469. [Accessed 22 March 2022].

[3] S. purdy, "arxiv.org," Arxiv, 18 February 2016. [Online]. Available: https://arxiv.org/abs/1602.05925. [Accessed 22 March 2022].

[4] K. j. H. &. S. Ahmad, "link.springer.com," SpringerLink, 20 July 2021. [Online]. Available: https://link.springer.com/article/10.1007/s42452-021-04715-0#Sec14. [Accessed 22 March 2022].

[5] "Dataset: https://github.com/numenta/nupic/blob/master/src/nupic/datafiles/extra/hotgym/rec-center-hourly.csv".

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