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

Implement Trace SDR Method

Aishin Abdulla Yoosufali
aishin.abdullayoosufali@stud.fra-uas.de

***Abstract*— Computers amalgamates the combinations as 0's and 1's to store the information. Comparably, there are neurons in our brain that use Sparse Distributed Representations (SDR) to store the data or analyze it. In the HTM research, it has found that the Spatial Pooler Learning algorithm (operating at Layer2) with the Homeostatic Plasticity Controller (HPC), which continuously generates the SDR, has an encoded input or sequence. The activation of neurons is not always the same for every sequence. This paper aims to make distinctive SDR vectors traced by an underlying algorithm, which have similar semantic attributes in the same locations of each SDRs. The same mechanism was used in this experiment. Thus, two SDR encodings have been taken, which are obtained from an existing model to investigate, along with that, implemented a new model that will manifest the dissimilarity in both the traced encodings of SDR. The dissimilarity of SDR vectors are compared using a padding text such as given specific spacing between the indices of every bit in SDR. The experiment will show a well formation of traced SDR encodings.**

Keywords*— *Hierarchical Temporal Memory, Sparse Distributed Representation, Spatial Pooler

I. INTRODUCTION

In Hierarchical Temporal Memory (HTM), the term Spatial Pooler (SP) has been used because input patterns that share a large number of co-active neurons (i.e., that are spatially similar) are grouped into a common output representation [1]. The SP learning algorithm encodes spatially similar inputs into the same or similar SDRs memorized as a population of active neurons organized in groups called micro-columns [2]. The following figure 1 [3] and figure 2 [6] show the SDR pattern of active neurons in the neocortex. The blue cells represent the inactive neurons, and similarly the yellow ones represent the active neurons.

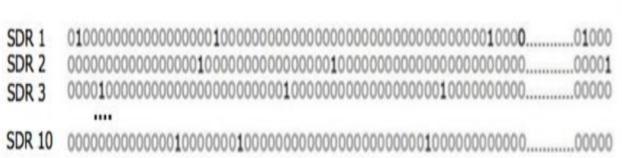

Figure 1: A union of SDRs [Source].

SDR contains thousands of bits constituted as 0's and 1's, where at some particular time only a small ratio is indicated as 1's and the rest are 0's. These bits are considered as neurons, the same as the bits in the machine world. In SDR, the bit 1 is a comparatively active neuron and bit 0 is indicated as an inactive neuron. This is what SDR is responsible for and has a meaning to it.

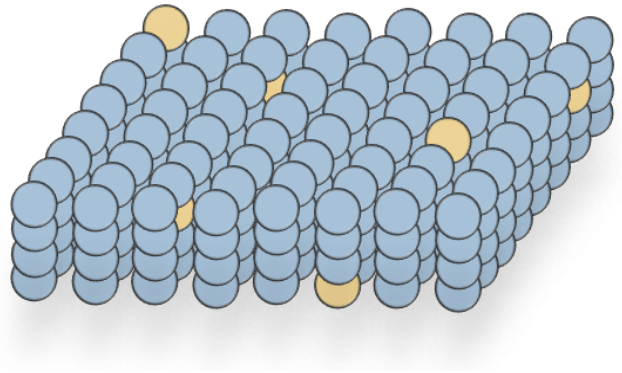

Figure 2. Sparse distributed representation of the spatial-temporal pooler. [Source]

Every bit in SDR is not designated with any value or names, but it has a semantic meaning

which is to be learned [3]. The experiment here, shows the work of the two SDR vectors which have the active bits on the similar locations and in some locations there are also dissimilar or inactive bits and both SDR sets allot the same semantic attributes because of the active bits in the same index.

The investigation proposed here, there is an overlap between both the SDRs as they have somewhere the same set of active bits, an immediate comparison between the two representations which are semantically similar and differentiate the indices of bits which are dissimilar can be shown. Within one set of neurons, an SDR at one point in time can associatively link to the next occurring SDR. In this way, sequences of SDRs are learned. Associative linking also occurs between different populations of cells (layer to layer or region to region). The meanings of the neuron encodings in one region are different from the meanings of neuron encodings in another region. [3]

In this paper, our approach shows a better format for similar SDRs which are being generated using an already proposed method called StringifyVector. Furthermore, implemented a new method using the proposed StringifyVector method to show the well formation of the two SDRs, which will show the representations (as SDRs are similar in most locations) to quickly see the difference between the two SDR encodings. This section describes the generation and tracing of SDR encodings and also focuses on our approach to well format the SDR vectors. Section II states the methods which are used to implement and show the difference between two SDR vectors, if they have a similar index of bits which are active. Section IV shows the results of our implemented model and also a Unit Test of our experiment, to make sure our method is correctly executed. Lastly, Section V concludes the approach followed for the experiment and discusses the future approaches of this model.

II. METHODOLOGY

To analyze the formation of two SDRs with similar active index bits, during the experiment created a new method that will show the outputs of SDR vectors with gaps or spaces to clearly distinguish the index of two representations. The detailed working of the experiment is described in this section.

A. Sparse Distributed Representations

In HTM, the generation of SDRs is encoded by a pattern, enrouted from the Spatial Pooler algorithm [2]. One of the properties of SDR is similarity [4]. SDRs have dispersed representations, where each bit has a valuable meaning. Consider two SDR vectors which have 1's in the same position (i.e. semantic similar attribute), such as overlapping. These representations are stored in a class as a list of strings of active bits.

The following block below shows how the SDR encodings generated from the Spatial Pooler algorithm have been stored as one of the inputs for the model in the code.

Listing 1. List of induced of active bits

```
51, 76, 87, 113,..., 224
```

The algorithm in HTM, traces out the SDR to process the information. The Stringify method [5], which is an already existing method produces the output such as two SDR vectors. The result of the above mentioned method, shows indexes of bits that are active.

The following extant method StringifyVector shows how to trace out the SDR vectors.

Listing 2. Existing StringifyVector method [5]

```
Helpers.StringifyVector(lyrOut.PredictiveCells.Sele
ct(c => c.Index).ToArray())
```

B. Overlapping

During the time of tracing of representations, discovered that there is similarity of SDR vectors, which is known as overlapping. The overlap score is simply the number of bits that are active or ON in the same locations of SDR encodings [8]. Now, the output of SDR are similar, then it is a challenge to see the difference in both the SDRs.

Hence, the new constructed method called StringifyTraceSDR [7] which will create a well formatted set of SDR vectors, which has been obtained from the output of StringifyVector method, consisting of similar active bits. Then, the output of the StringifyVector method is used as an input for the newly proposed StringifyTraceSDR method.

The following StringifyTraceSDR method [7] shows how to differentiate between two SDR vectors in a well arranged indices of active and inactive bits.

Listing 3. Newly proposed StringifyTraceSDR [7]

```
string StringifyTraceSDR(List<int[]> sdrs)
{
    var heads = new List<int>(newint[sdrs.Count]);
      var outputs = new StringBuilder[sdrs.Count];

}
```

These indices of SDR vectors are stored in a StringBuilder class [9], which represents a mutable set of characters. Thus, every index of bits that are active are provided into StringBuilder, and are then further examined to distinguish them in SDR encodings. The implied method stores every bit of the active column as a list of strings.

Listing 4. StringBuilder class [9]

```
var outputs = new StringBuilder[sdrs.Count];
```

C. Padding

Besides, to differentiate between the two SDRs which have similar encodings, created a better result, such as by adding space (i.e. or padding) at places that do not have the same or similar index in the same location of the SDRs.

Following code of block shows how to fill the space in place of missing index bits which are inactive.

Listing 5. Implementing inactive bits with Spaces [7]

```
var numOfSpaces =
minActiveColumn.ToString().Length;
            for (var j = 0; j < numOfSpaces; j++)
            {
               outputs[i].Append(" ");
            }
            outputs[i].Append(", ");
```

This code represents if there is an index which has not similar vectors in both the representations, then it will add a space or padding in the place of an inactive bit from both the SDRs.

III. RESULTS

The crux of our approach was to modify the formation of two SDR vectors. Along with that, we compared both the results using a padding or spacing in place of the indices of inactive SDR bits. The result shows how SDR encodings with correct spacing between the indices have occurred.

A. Implementation of the Model

Figure 3 manifests the result of our proposed model. It shows how the traced SDR vectors are inserted as input, the StringifyVector method [8] gives a symmetric view as the output. Here, only two SDRs are being considered to first initiate such a model. Nevertheless, an increased number of SDR vectors can be used using this model for the well arrangement of the SDRs. Also, the perspective of the model is that it can be worked for active columns which encode the SDRs, and has a number of columns more than 2048.

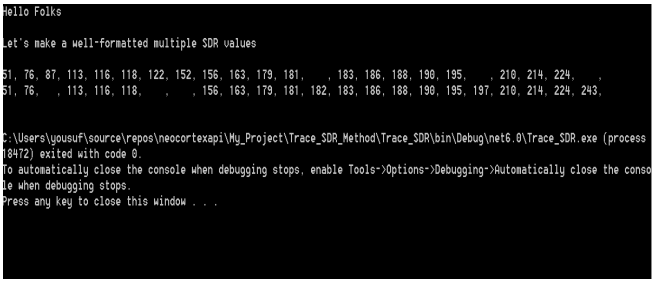

Figure 3. Result of the approached model

B. *Unit Testing*

A unit test project has been also created for the model, a test method called StringifyTest [10] shows how the proposed model and the output is correct. The following lines of code is written in the StringifyTest method.

Listing 6. Unit test StringifyTest [10]

```
public void StringifyTest()
      {

         var list = new int[] { 51, 76, 87 };

         var list1 = new int[] { 51, 76, 113 };

         var output =
Helpers.StringifyTraceSDR(new  List<int[]> { list,
list1 });

         var expectedResult = new StringBuilder();
        var sdr1 = new StringBuilder();
         sdr1.Append("51, 76, 87,    , ");

         var sdr2 = new StringBuilder();
         sdr2.Append("51, 76,   , 113, ");


expectedResult.AppendLine(sdr1.ToString());

expectedResult.AppendLine(sdr2.ToString());

        Assert.IsTrue(output ==
expectedResult.ToString());
      }
```

In this method, only three index bits of SDR vectors are considered for the testing, further the whole string of indices bits can also be tested using this same test method. The output is the final result of the proposed model. Whereas, the expectedResult is what is newly created for the test method. It was built using the same process as the outputs of SDR in the StringifyTraceSDR method generated. The expected result is also stored as a StringBuilder and later converted the integers into a list of strings. This is the same implementation of the proposed method [7] which creates the well formatted output of SDRs.

IV. CONCLUSION & FUTURE WORK

In Hierarchical Temporal Memory, the underlying algorithm tracks the SDRs. These SDRs are continuously generating encoded bits inside the mini-columns, and these SDR vectors have a sequence which are mostly identical with each other in the same locations. However, it is quite difficult for a person to differentiate a long string of SDRs, as most of the bit indexes have the same values and locations. Thus, this part of the problem is solved by building a new method applied to the model, which traces out the SDR encodings with similar semantic attributes and, moreover, creates the spacing wherever inactive bits of SDR vectors are observed. However, the proposed model can moreover be modified in the future, such as instead of spacing there can be any symbol inserted, for example a “_” or “$”, which will differentiate between the active and inactive bits. In addition to that, the active bits have two and three digit numbers. In the future or parallel implementations, it can be changed so that every bit has the same digit number.

REFERENCES.

The main source that has served this experiment model is the paper mentioned in [1] and the model mentioned in [8].
Further references [5], [7], [9] & [10] contain source code for the experiments stated above.

[1] Damir Dobric, Andreas Pech, Bogdan Ghita, Thomas Wennekers, (2021), "Improved HTM Spatial Pooler with Homeostatic Plasticity Controller". https://www.scitepress.org/Papers/2021/103142/103142.pdf

[2] Yuwei Cui, Subutai Ahmad, Jeff Hawkins, (2017), "The HTM Spatial Pooler- A Neocortical Algorithm for Online Sparse Distributed Coding", frontiers in Computational Neuroscience. https://doi.org/10.3389/fncom.2017.00111

[3] https://numenta.com/assets/pdf/biological-and-machine-intelligence/BaMI-SDR.pdf

[4] Subutai Ahmad, (2017), "Sparse Distributed Representations", HTM Forum https://discourse.numenta.org/t/sparse-distributed-representations/2150

[5] StringifyVector method- https://github.com/ddobric/neocortexapi/blob/master/source/UnitTestsProject/CortexNetworkTests/InputBitsExperimentTest.cs

[6] Abdullah M. Zyarah, (2015), "Design and analysis of a reconfigurable hierarchical temporal memory architecture", ResearchGate https://www.researchgate.net/figure/Sparse-distributed-representation-of-the-spatial-temporal-pooler-The-blue-cells_fig2_299280624

[7] StringifyTraceSDR method- https://github.com/aishincp/neocortexapi/blob/Infinity/My_Project/Trace_SDR_Method/Trace_SDR/Helpers.cs

[8] Subutai Ahmad, (2015), "Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory", ResearchGate https://www.researchgate.net/publication/274094611_Properties_of_Sparse_Distributed_Representations_and_their_Application_to_Hierarchical_Temporal_Memory

[9] StringBuilder class- https://github.com/ddobric/neocortexapi/blob/master/source/NeoCortexApi/Helpers.cs

[10] Unit Test- StringifyTest method https://github.com/aishincp/neocortexapi/blob/Infinity/My_Project/Trace_SDR_Method/Trace_SDR.UnitTests/Helpers_UnitTest.cs