

ML19/20-5.1. Implementation of Geo-Spatial Encoder in HTM

Gouthaami Sukadore Ramesh

Lakshmi Alekya Chittem

Ravi Kumar Solanki

gouthaami.sukardoramesh@stud.fra-uas.de, lakshmi.chittem@stud.fra-uas.de, ravikumar.solanki@stud.fra-uas.de

Abstract— For implementing geospatial encoder, This application utilizes Hierarchical Temporal Memory (HTM), a biologically inspired computational theory based on the neocortex, to automatically model typical travel patterns for larger areas and smaller areas. Travel patterns are learned by encoding latitude and longitude data plus speed into Sparse Distributed Representations (SDRs). In this we describe how to encode data as SDRs for use in HTM systems. How an encoder can capture geospatial data. We explain about the scalar encoders, and we discuss about the test cases implemented in Geo spatial encoder for various locations.

Keywords—HTM, SDRs, Scalar Encoder, Latitude, Longitude

I. INTRODUCTION

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 (Hawkins and Ahmad, 2015). HTM systems require data input in the form of Sparse Distributed Representations (SDRs) (Ahmad and Hawkins, 2016). 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 0's and a few are 1's. 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. Any data that can be converted into an SDR can be used in a wide range of applications using HTM systems. Consequently, the first step of using an HTM system is to convert a data source into an SDR using what we call an encoder. The encoder converts the native format of the data into an SDR that can be fed into an HTM system. The encoder is responsible for determining which output bits should be ones, and which should be zeros, for a given input value in such a way as to capture the important semantic characteristics of the data. Similar input values should produce highly overlapping SDRs.

II. METHODOLOGY

In the concept of Encoding the data, the design of an encoder is dependent on the type of data. The encoder must capture the semantic characteristics of the data that are important for your application. Many of the encoder implementations in NuPIC take range or resolution parameters that allow them to work for a broad range of applications. There are a few important aspects that need to be considered when encoding

data:

1. Semantically similar data should result in SDRs with overlapping active bits.
2. The same input should always produce the same SDR as output.
3. The output should have the same dimensionality (total number of bits) for all inputs.
4. The output should have similar sparsity for all inputs and have enough one-bits to handle noise and subsampling.

SDR

SDR is fundamental form of information representation in the brain and in HTM system

SDR consist of thousands of bits where at any points in time a small percentage of bits 1's and the rest are 0's.

Binary vectors in SDR: for the purposes of this discussion we consider SDR as binary vectors the notation $X [b_0, \dots b_1]$ for an SDR. The value of each elements is "0" or "1" for OFF and ON respectively.

Geo-Spatial Encoder

Geospatial data comes in many forms and formats, and its structure is more complicated than tabular or even nongeographic geometric data. It is, in fact, a subset of spatial data, which is simply data that indicates where things are within a given coordinate system.

To know about how an encoder can capture geospatial data. The most obvious aspect is that locations close to each other should be considered similar while locations far apart should not be considered similar. To encode this meaning, we first have to determine the resolution that we want to encode.

For this example, we will assume that we are using GPS coordinates that are accurate to about ten feet. And we will be doing two dimensional locations, although extending the encoding to three dimensions would be straight-forward.

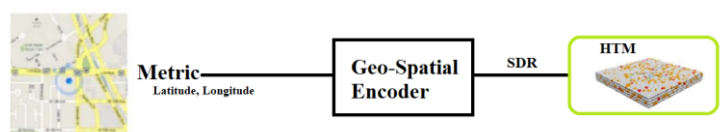


Figure.1. Flow Chart for Implementation

Classes for encoding different types into SDRs for HTM input. Like Base Encoder, Category encoder, Scalar Encoder, Date Encoder. We have opted for scalar encoder technique to encode the data into SDR's.

There is special case in co-ordinate encoder, When # of dimensions = 1, it is become a Scalar Encoder. set the radius based on the past statistics of the data you have scalar encoder with dynamic resolution. In this project we taken the simple scalar encoder parameter to generate the output of Geo-Spatial Encoder. The parameter is mention further in the following report.

Generally scalar encoder encodes a numeric (floating point) value into an array of bits. The output is 0's except for a contiguous block of 1's. The location of this contiguous block varies continuously with the input value.

The encoding is linear. If you want a nonlinear encoding, just transform the scalar (e.g. by applying a logarithm function) before encoding. It is not recommended to bin the data as a pre-processing step, e.g. "1" = \$0 - \$.20, "2" = \$.21-\$0.80, "3" = \$.81-\$1.20, etc. as this removes a lot of information and prevents nearby values from overlapping in the output. Instead, use a continuous transformation that scales the data (a piecewise transformation is fine).

Choosing the Size of an Encoder

Every encoder, no matter what it is representing, should create SDRs that have a fixed number of bits n and a fixed number of one bits w . How do you know what are good values for n and w ? To maintain the properties that come from sparsity, w can't be a large fraction of n . But if w gets too small then we lose the properties that come from having a distributed representation. As a general rule, for numerical values w should be at least 20 to properly handle noise and subsampling and n should be at least 100 to provide enough resolution to distinguish between many numbers.

Parameters

1. **w** – The number of bits that are set to encode a single value - the "width" of the output signal restriction: w must be odd to avoid centering problems.
2. **minval** – The minimum value of the input signal.
3. **maxval** – The upper bound of the input signal. (input is strictly less if periodic == True)
4. **periodic** – If true, then the input value "wraps around" such that $\text{minval} = \text{maxval}$. For a periodic value, the input must be strictly less than maxval , otherwise maxval is a true upper bound.
5. **n** – The number of bits in the output. Must be greater than or equal to w
6. **radius** – Two inputs separated by more than the radius have non-overlapping representations. Two inputs separated by less than the radius will in general overlap in at least some of their bits. You can think of this as the radius of the input.
7. **resolution** – Two inputs separated by greater than, or equal to the resolution are guaranteed to have different representations.

8. **name** – an optional string which will become part of the description
9. **clipInput** – if true, non-periodic inputs smaller than minval or greater than maxval will be clipped to $\text{minval}/\text{maxval}$
10. **forced** – if true, skip some safety checks (for compatibility reasons), default false

'n' vs 'Resolution'

It may not be natural to specify "n", especially with non-periodic data. For example, consider encoding an input with a range of 1-10 (inclusive) using an output width of 5. If you specify $\text{resolution} = 1$, this means that inputs of 1 and 2 have different outputs, though they overlap, but 1 and 1.5 might not have different outputs. This leads to a 14-bit representation like this:

```
1 -> 11111000000000 (14 bits total)
2 -> 01111100000000
...
10-> 00000000011111
[resolution = 1; n=14; radius = 5]
```

You could specify $\text{resolution} = 0.5$, which gives

```
1 -> 11111000... (22 bits total)
1.5 -> 011111....
2.0 -> 001111....
[resolution = 0.5; n=22; radius=2.5]
```

You could specify $\text{radius} = 1$, which gives

```
1 -> 111110000000.... (50 bits total)
2 -> 000001111100....
3 -> 00000000001111...
...
10 -> .....000011111
[radius = 1; resolution = 0.2; n=50]
```

An N/M encoding can also be used to encode a binary value, where we want more than one bit to represent each state. For example, we could have: $w = 5$, $\text{minval} = 0$, $\text{maxval} = 1$, $\text{radius} = 1$ (which is equivalent to $n=10$)

```
0 -> 1111100000
1 -> 0000011111
```

Implementation Details

```
range = maxval - minval
h = (w-1)/2 (half-width)
resolution = radius / w
n = w * range/radius (periodic)
n = w * range/radius + 2 * h (non-periodic)
```

Figure.2. Formulae used for Implementation

This is an implementation of the scalar encoder that adapts the min and max of the scalar encoder dynamically. This is essential to the streaming model of the online prediction framework. Initialization of an adaptive encoder using resolution or radius is not supported; it must be initialized with n . This n is kept constant while the min and max of the encoder changes.

When we create an implementation of this encoder, we first split the range of values into buckets, and then map the buckets to a set of active cells. Here are the steps for encoding a value with this approach:

1. Choose the range of values that you want to be able to represent, minVal and maxVal
2. Compute the range as $\text{range} = \text{maxVal} - \text{minVal}$
3. To Calculate the no of bits for periodic, $n = w * \text{range} / \text{radius}$
4. To Calculate the no of bits for non-periodic, $n = w * \text{range} / \text{radius} + 2 * h$ where h is half width.
5. We can calculate half width, by using the formulae $h = (w - 1) / 2$.
6. Create the encoded representation by starting with n unset bits and then set the w consecutive bits.

Note that this encoding scheme has four parameters: minimum value, maximum value, radius, and number of active bits (w). Alternatively, you may choose the total number of bits, n , rather than the number of buckets.

Here is an example, we need to encode a location as a set of n bits with w active. for latitude real time values between two countries. For Germany= 48° N and India= 20° N

1. Given, $\text{minVal} = 20$ and $\text{maxVal} = 48$.
2. Computed the $\text{range} = \text{maxVal} - \text{minVal} = 48 - 20 = 28$.
3. Here have Chooses random value for width, so $w = 21$.
4. We have considered non periodic for values, so the no of bits for non-periodic is $n = w * \text{range} / \text{radius} + 2 * h = 389$.
5. We can calculate half width, by using the formulae $h = (w - 1) / 2 = 10$.
6. The representation will be with 389 bits and with 21 consecutive active bits starting at the 20th bit:

000000....11111111111111111111000000000.....000
 ↑
 21st Bit

This encoding approach is simple and provides quite a bit of flexibility but requires that you know the appropriate range of the data. If your data falls outside the minimum and maximum values then the encoder doesn't work well. Typically, you would use the smallest w for values below the range and the largest w for values above the range. Thus, all values above the range will have the same representation, and similarly for those below the range.

The number of buckets can be chosen depending on the inherent noise for this metric in your application and the quality of the desired predictions. For a very noisy signal you might want a smaller number of buckets. This would allow the HTM to see more stable inputs but it would not be able to make very precise predictions. For a very clean signal you could have a large number of buckets. In this case the HTM would be able to make very precise predictions.

III. CODE

Under 'NeoCortexApiTests1' in HTM Folder we have implemented the Scalar encoder Test cases in the folder named Encoders. (Fig.3) Here we have used two Neo-cortex Libraries 'NeoCortexApi.Network', 'NeoCortexApi.Encoders'.

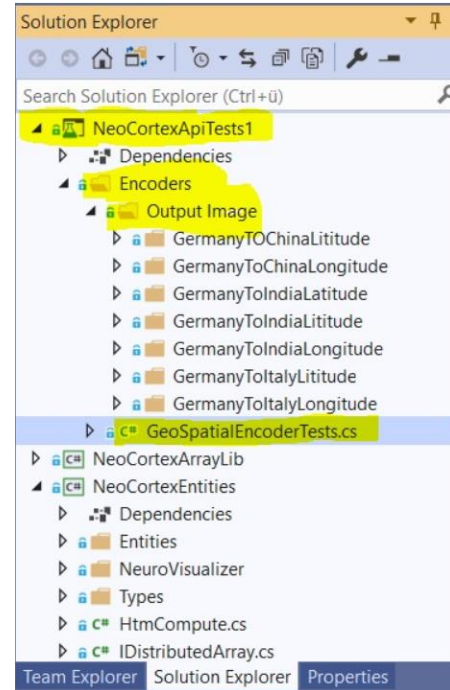


Figure.3. Geo-Spatial Encoder Implemented Code Location

To verify this implemented code under different scenarios, several tests have been conducted, as highlighted in Fig.4

Test	Duration	Traits
NeoCortexApiTests1 (6)	15 min	
UnitTestsProject.EncoderTests (..)	15 min	
GeoSpatialEncoderTests (6)	15 min	
GermanyTOChinaLatitude	1 min	
GermanyToChinaLongitude	7 min	
GermanyToIndiaLatitude	2 min	
GermanyToIndiaLongitude	5 min	
GermanyToItalyLatitude	13 sec	
GermanyToItalyLongitude	8 sec	

Test Detail Summary	
GermanyTOChinaLatitude	Source: GeoSpatialEncoderTests.cs line 212
Test has multiple result outcomes	
11 Passed	
Results	
1) GermanyTOChinaLatitude	Duration: 34 sec
2) GermanyTOChinaLatitude (20,System.Int32[])	

Figure.4. Implemented Test Cases for Different locations

IV. TEST CASES & RESULTS

The output images are located the folder as mentioned in Fig.3. Here, we have shown some output results for reference using Bit map for the following locations using latitude and longitude values:

1. Unit Test Case for Germany to India


In the following unit test cases we have used scalar encoder parameters to implement the geo-spatial data. Here we have shown test cases implementation for Germany and India.

We have considered real time values for India ("20.5937" for Latitude and "77.9629" for Longitude) and real time values for Germany ("48" for Latitude and "10" for Longitude).

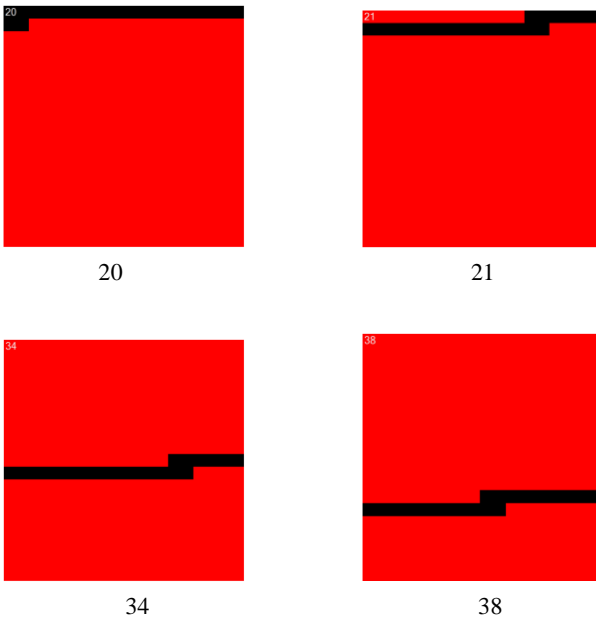
For this value we generated two different unit test cases for each metric which are Latitude and Longitude are shown below:

1.1 Unit Test Case for Germany to India (Latitude):

1. Given, minVal = 20 and maxVal = 48.
2. Computed the $range = maxVal - minVal = 48 - 20 = 28$.
3. Here have Chooses random value for width, so $w=21$.
4. We have considered non periodic for values, so the no of bits for non-periodic is
 $n = w * range / radius + 2 * h = 389$.
5. We can calculate half width, by using the formulae
 $h = (w - 1) / 2 = 10$.
6. The representation will be with 389 bits and with 21 consecutive active bits starting at the 20th bit:


000000....111111111111111111110000000000.....000000
 20th Bit

The above-mentioned output is in binary form, and we have generated the Image form using Bit Map. The same process is followed for different w values. Here we have shown output bit map images for various values we have used to generate output.

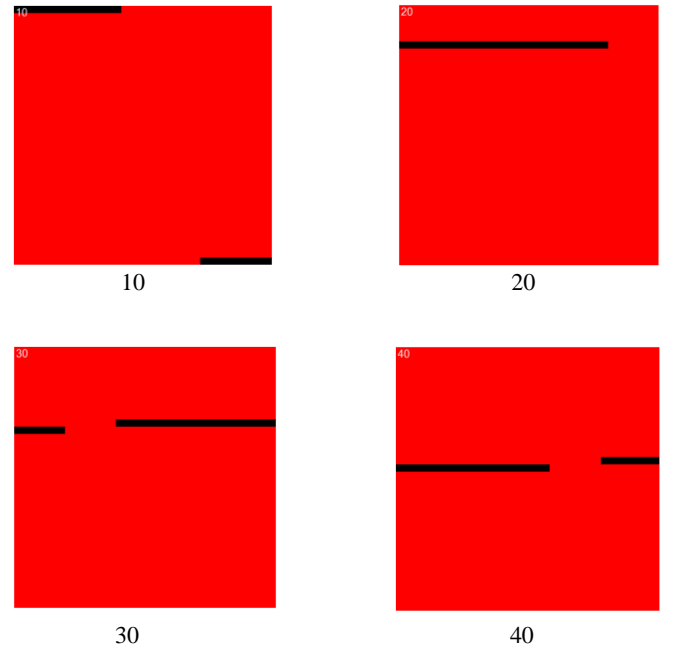


1.2 Unit Test Case for Germany to India (Longitude):

1. Given, minVal = 10 and maxVal = 77
2. Computed the $range = maxVal - minVal = 77 - 10 = 67$
3. Here have Chooses random value for width, so $w=29$.
4. We have considered periodic for values, so the no of bits for periodic is
 $n = w * range / radius = 1300$
5. The representation will be with 1300 bits and with 29 consecutive active bits starting at the 10th bit:

00000.....1111111111111111111111111111110000.....000000
 10th Bit

The above-mentioned output is in binary form, and we have generated the Image form using Bit Map. The same process is followed for different w values. Here we have shown output bit map images for various values we have used to generate output.



2. Unit Test Case for Germany to China

We have considered real time values for China ("36" for Latitude and "104" for Longitude) and real time values for Germany ("48" for Latitude and "10" for Longitude).

For this value we generated two different unit test cases for each metric which are Latitude and Longitude are shown below:

2.1 Unit Test Case for Germany to China (Latitude):

1. Given, minVal = 36 and maxVal = 48.
2. Computed the $range = maxVal - minVal = 48 - 36 = 12$.
3. Here have Chooses random value for width, so $w=21$.
4. We have considered non periodic for values, so the no of bits for non-periodic is
 $n = w * range / radius + 2 * h = 430$.
5. We can calculate half width, by using the formulae
 $h = (w - 1) / 2 = 10$.

- [illegible]

Figure 1 displays four heatmaps illustrating the evolution of the function $f(x)$ for different values of α : 36, 38, 40, and 48. The plots show the spatial distribution of $f(x)$ over a square domain, with the red region representing $f(x) = 1$ and the black region representing $f(x) = 0$. As α increases, the black region expands from the top-left corner towards the bottom-right.

1. Given, $\text{minVal} = 10$ and $\text{maxVal} = 104$
2. Computed the $\text{range} = \text{maxVal} - \text{minVal} = 104 - 10 = 94$
3. Here have Chooses random value for width, so $w=29$.
4. We have considered periodic for values, so the no of bits for periodic is

$$n = w * \text{range} / \text{radius} = 1820$$
5. The representation will be with 1820 bits and with 29 consecutive active bits starting at the 10th bit:

The above-mentioned output is in binary form, and we have

We have considered real time values for Italy (“51.82” for Latitude and “12.57” for Longitude) and real time values for Germany (“48” for Latitude and “10” for Longitude).

2.1 Unit Test Case for Germany to Italy (Latitude):

- 48th Bit

[illegible]

The above-mentioned output is in binary form, and we have

48

49

3.2 Unit Test Case for Germany to Italy (Longitude):

- [illegible]

There are many different kinds of encoders available used for different kinds of application, further research work is in progress for improvising and developing new encoders and new techniques to encode the given data set.

As per the observation made while experimenting with these parameters as the value of the resolution changes the change is observed in output representation. The change in resolution is depends the two parameters, i.e. W (Width) and R (Radius). We have experimented with the Value of W and assuming radius to be constant for all the data sets and observed the output for the same. Here we have trained the data to give similar output certain radius value. Geospatial data which are close to each other will have semantically similar representation. While encoding data sets, we have considered latitude and longitudinal values individually because this encoder is subclass of scalar encoder. Whenever the dimension equal to one then it can be considered as generalized scalar encoder. In the above 6 test cases we would say that more defined out is obtained after 2nd test case. Here we have also considered bits for noise representation and subsampling. It will be simpler to understand the encoding of Geospatial if considered as single entity and we can represent the encoded values in the same bit map or different bit map based on the what application demand. Further Geospatial encoder can be used in geospatial tracking and Anomaly detection. Considering values of latitude and longitude individually can be advantage in anomaly detection to find the error.

- [1] Hawkins, J. & Ahmad, S. (2015). Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex. arXiv, 1511.00083. Neurons and Cognition; Artificial Intelligence. Retrieved from <http://arxiv.org/abs/1511.00083>
- [2] Ahmad, S., & Hawkins, J. (2016). How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites. arXiv, 1601.00720. Neurons and Cognition; Artificial Intelligence. Retrieved from <http://arxiv.org/abs/1601.00720>
- [3] Webster D, Popper A, Fay R editors (1992). The Mammalian Auditory Pathway: Neuroanatomy. Springer handbook of auditory research, v1. Springer-Verlag New York, Inc
- [4] Schuknecht, H.F. (1974). Pathology of the Ear. Cambridge, MA: Harvard University Press.
- [5] Webber, F. D. S. (2015). Semantic Folding Theory And its Application in Semantic Fingerprinting, 57. Artificial Intelligence; Computation and Language; Neurons and Cognition. Retrieved from <http://arxiv.org/abs/1511.08855>