

# Improve the Scalar Encoder of NeocortexApi

Bishal Gautam  
bishal.gautam@stud.fra-uas.de

**Abstract**—Neocortexapi is code implemented on .Net framework that tries to mimic what is happening in the neocortex region of brain. It is an implementation of Hierarchical Temporal Memory (HTM). HTM is the biologically inspired machine intelligence technology which tries to mimic the neocortex region of brain in terms of architecture and processing. In the neocortex, data is inputted in the form of Sparse Distributed Representations (SDRs). The SDRs are generated in the form of active pulse through millions of axons from sensory organs and from other cortical regions. Encoder is a part of neocortexapi that acts as sensory organs. Just like sensory organs which converts sensations to pulses and send that pulse to the neocortex region of brain for processing, likewise encoder encodes any input to the form of SDRs, which are then processed by the neocortexapi. In this paper, one of the encoders, Scalar Encoder's will be described and improved. Scalar Encoder is responsible for encoding numeric or floating-point values in a series of 0's and contiguous block of 1's. Current version of Scalar Encoder is tested and improved by choosing appropriate value for parameters and by improving the interface and current logic.

**Keywords**—sparse distributed representations, scalar encoder, neocortex, NeocortexApi, .NET6, APIs, HTM

## I. INTRODUCTION

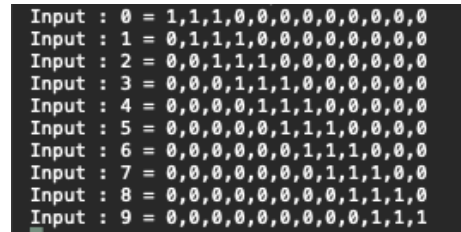
Massive amount of sensory information is processed continuously in the neocortex. Neocortex never stops processing the information and still work flawlessly. All the information is processed as a series of active bits passed down from the sensory organs to the neurons through the synaptic connections. This information is represented in neuron in the form of active pulse for a time instant. The brain, especially neocortex, can hold huge amount of information from life to death. This is the possible because of sparse representation of neurons for a particular input. In sparse representation, large array of bits is inactive(0's) and a very few are active(1's). Each bit carries some meaning. Two SDRs having more than a few overlaps would represent that two SDRs are semantically similar. Encoder in neocortexapi corresponds to sensory organs. Encoder encodes any input in the form of sparse distributed representations.

Scalar Encoder is a type of encoding technique in HTM system that encodes scalar values such as numeric or floating-point value in the series of 0's and 1's. In this type of encoder, output has 0's with adjacent block of 1's. The goal of this paper is to ameliorate the encoding of scalar values. Enhancement is done on preexisting scalar encoder by setting appropriate parameters and by adding various functionalities.

## II. MATERIALS AND METHODS

### A. Encoder Overview

First step of using neocortexapi is to transform the data to a form (SDRs) so that it can be digested for further processing (i.e. by Spatial Pooler). Encoder is responsible for converting the data to a SDRs representation. There are various types of encoders such as Geospatial encoder, Image encoder, Scalar Encoder, DateTime Encoder, etc. Geospatial encoder takes geometric coordinates values and convert them to SDRs. Two objects can be said to be close to each other geographically if their SDRs overlap over a some threshold. Similarly in Scalar Encoder two values SDRs would be similar if they are equal or close to each other.



```
Input : 0 = 1,1,1,0,0,0,0,0,0,0,0,0
Input : 1 = 0,1,1,1,0,0,0,0,0,0,0,0
Input : 2 = 0,0,1,1,1,0,0,0,0,0,0,0
Input : 3 = 0,0,0,1,1,1,0,0,0,0,0,0
Input : 4 = 0,0,0,0,1,1,1,0,0,0,0,0
Input : 5 = 0,0,0,0,0,1,1,1,0,0,0,0
Input : 6 = 0,0,0,0,0,0,1,1,1,0,0,0
Input : 7 = 0,0,0,0,0,0,0,1,1,1,0,0
Input : 8 = 0,0,0,0,0,0,0,0,1,1,1,0
Input : 9 = 0,0,0,0,0,0,0,0,0,1,1,1
```

Fig. 1. Scalar Encoding of scalar values with total number of bits 12 and 3 number of active bits

Two encodings are said to be similar if they have more overlapping active bits. As shown in figure 1, we can see that the encoding of 1 and 2 are similar to each other. Likewise, the encoding of 6 and 7 are similar to each other. But the encoding of 1 and 7 is totally different than the encoding of 1 and 2. This is due to the fact that 1 and 2 has less difference than 1 and 7 and hence 1 and 2 are more similar both semantically and .

There are some important aspects that need to be considered when encoding data. They are described as [1]:

- For data which are semantically similar, encoding should result in SDRs with overlapping active bits
- Encoding should be deterministic, meaning that same input should always produce the same SDR as output.
- The output should have the same dimensionality (total number of bits) for all inputs.

- The output of encoding should have similar sparsity for all input. Output should have enough one-bits to handle noise and subsampling.

### B. Parameters and Terms for Scalar Encoder

Before encoding scalar values, scalar encoder is initialized with parameter dictionary. The parameters which the encoder depends can be explained as :

- N: Total number of bits to represent the input data
- W: Number of active bits (1's) to represent the input data
- MinVal: Minimum Value that the SDR can represent without clipping
- MaxVal: Maximum Value that the SDR can represent without clipping. For periodic input, this value should be strictly less than input
- Periodic: Boolean Value, if true, representation is repeated after certain interval
- ClipInput : If true, value less than the minimum has SDR similar to minimum value and values more than the maximum has SDR similar to maximum value
- Resolution: For a particular value of resolution, input separated by more than or equal to resolution would have different representations. For e.g., if resolution = 0.5, then 4 and 5 will have different encoding but 4.1 and 4.4 could have similar encoding.
- Radius: For a particular value of radius, input separated by the radius would have completely different representations with non-overlapping active bits.
- Name: It is an optional string that will become a part of description

N, Resolution and Radius are three mutually exclusive parameters. Exactly one of N, Radius or Resolution should be given for the input space, rest of the two should be "0" or not set on the input space. Rest of the two parameters will be calculated internally based on the parameters supplied to the scalar encoder. Total Buckets represent the total number of spaces that can represent distinct SDRs. For example, for a particular setting if the bucket size is 14 then the encoder can encode 14 distinct SDRs. The equation to calculate the Bucket size for a encoder is:

$$\text{Total Buckets} = (N - W + 1) \quad (1)$$

## III. TEST CASES WITH RESULTS

Current version of Scalar encoder has some defects, in this part we will be discussion the pitfalls of currently implemented scalar encoder.

After finding the pitfalls, we will be experimenting to find the correct solution with a way to improve current version of scalar encoding.

### A. Test Case generating the bucket size problem

In the first experiment, we are tried to regenerate the error of bucket size problem with the predefined parameters. For the experiment, we have:

- N = 10
- W = 3
- MinVal = 0
- MaxVal = 9
- Periodic = False
- ClipInput = True

The output with the above defined parameter is shown as :

```

Terminal - MyProject
Inside the Main of ImproveScalar
Encoder Namespace
Input : 0 = 1,1,1,0,0,0,0,0,0,0
Input : 1 = 0,1,1,1,0,0,0,0,0,0
Input : 2 = 0,0,1,1,1,0,0,0,0,0
Input : 3 = 0,0,1,1,1,0,0,0,0,0
Input : 4 = 0,0,0,1,1,1,0,0,0,0
Input : 5 = 0,0,0,0,1,1,1,0,0,0
Input : 6 = 0,0,0,0,0,1,1,1,0,0
Input : 7 = 0,0,0,0,0,0,1,1,1,0
Input : 8 = 0,0,0,0,0,0,0,1,1,1
Input : 9 = 0,0,0,0,0,0,0,0,1,1

```

Fig.2. Output of Scalar Encoder with similar SDRs

The total number of buckets that our configuration can represent is  $(N - W + 1) = (10 - 3 + 1) = 8$  for a Resolution of 1. We represent data from 0 to 9, which in total should be 10 different representations. Because the number of available bucket (8) is less than the required bucket (10), this causes an overlap in SDR of input space. In the above case overlap occurs for the input of 2 and 3.

### B. Test Case generating the bucket size problem – Second

In the test case 1, the overlapping was clearly due to the low number of bucket size. So, for our second experiment, we increased the value of N to 11, we have:

- N = 11
- W = 3
- MinVal = 0
- MaxVal = 9
- Periodic = False
- ClipInput = True

The output with the above defined parameter is shown as:

```

Terminal - MyProject
Terminal - MyProject

Inside the Main of ImproveScalarEncoder Namespace
Input : 0 = 1,1,1,0,0,0,0,0,0,0,0,0
Input : 1 = 0,1,1,1,0,0,0,0,0,0,0,0
Input : 2 = 0,0,1,1,1,0,0,0,0,0,0,0
Input : 3 = 0,0,0,1,1,1,0,0,0,0,0,0
Input : 4 = 0,0,0,0,1,1,1,0,0,0,0,0
Input : 5 = 0,0,0,0,1,1,1,0,0,0,0,0
Input : 6 = 0,0,0,0,0,1,1,1,0,0,0,0
Input : 7 = 0,0,0,0,0,0,1,1,1,0,0,0
Input : 8 = 0,0,0,0,0,0,0,1,1,1,0,0
Input : 9 = 0,0,0,0,0,0,0,0,1,1,1,1

```

Fig. 3. Output of Scalar Encoder for the given parameter with similar SDRs

The total number of buckets that our configuration can represent is  $(N - W + I) = (11-3+1) = 9$ . Since we need to represent data from 0 to 9, which is in total 10. The number of available bucket (9) is less than the required bucket (10). This cause in an overlap in SDR of input space. In our case, 4 and 5 input overlapped.

### C. Test Case for the solution of the given predefined parameter

We experimented by increasing the value of N again so as to increase the number of bucket size. The parameters value is:

- N = 12
- W = 3
- MinVal = 0
- MaxVal = 9
- Periodic = False
- ClipInput = True

The output with the above defined parameter is shown as:

```

Terminal - MyProject
Terminal - MyProject

Inside the Main of ImproveScalarEncoder Namespace
Input : 0 = 1,1,1,0,0,0,0,0,0,0,0,0
Input : 1 = 0,1,1,1,0,0,0,0,0,0,0,0
Input : 2 = 0,0,1,1,1,0,0,0,0,0,0,0
Input : 3 = 0,0,0,1,1,1,0,0,0,0,0,0
Input : 4 = 0,0,0,0,1,1,1,0,0,0,0,0
Input : 5 = 0,0,0,0,0,1,1,1,0,0,0,0
Input : 6 = 0,0,0,0,0,0,1,1,1,0,0,0
Input : 7 = 0,0,0,0,0,0,0,1,1,1,0,0
Input : 8 = 0,0,0,0,0,0,0,0,1,1,1,0
Input : 9 = 0,0,0,0,0,0,0,0,0,1,1,1

```

Fig. 4. Output of the Scalar Encoder after increasing N for appropriate bucket size

The total number of buckets that our configuration can represent is  $(N - W + I) = (12 - 3 + 1) = 10$ . Since we need to represent data from 0 to 9, which is 10 different SDRs representations. The number of available buckets is equal to the number of required buckets. This causes a discrete SDR representation of different input.

### D. Test Case implementing intuitive terminal to solve similar sdr representation problem

To solve the problems faced in test case A, B and C, We implemented an intuitive command line interface that lets the user update the value of N during runtime.

```

Inside the Main of ImproveScalarEncoder Namespace
The value of N is less than required 12 for resolution of 1.This might cause some output SDRs to overlap.
Are you sure you want to proceed with the entered value of N or do you want to use recommended N ?

Enter [yes] if you would like to update N to minimum required. [no] if you don't.
no
Noted : NO!
Keeping N as it is.
Input : 0 = 1,1,1,0,0,0,0,0,0,0,0,0
Input : 1 = 0,1,1,1,0,0,0,0,0,0,0,0
Input : 2 = 0,0,1,1,1,0,0,0,0,0,0,0
Input : 3 = 0,0,0,1,1,1,0,0,0,0,0,0
Input : 4 = 0,0,0,0,1,1,1,0,0,0,0,0
Input : 5 = 0,0,0,0,1,1,1,0,0,0,0,0
Input : 6 = 0,0,0,0,0,1,1,1,0,0,0,0
Input : 7 = 0,0,0,0,0,0,1,1,1,0,0,0
Input : 8 = 0,0,0,0,0,0,0,1,1,1,0,0
Input : 9 = 0,0,0,0,0,0,0,0,1,1,1,1

```

Fig.5a. Interactive terminal to set the value of N (choosing “no” option)

```

Inside the Main of ImproveScalarEncoder Namespace
The value of N is less than required 12 for resolution of 1.This might cause some output SDRs to overlap.
Are you sure you want to proceed with the entered value of N or do you want to use recommended N ?

Enter [yes] if you would like to update N to minimum required. [no] if you don't.
yes
Noted : YES!
Updating N to appropriate value.
Input : 0 = 1,1,1,0,0,0,0,0,0,0,0,0
Input : 1 = 0,1,1,1,0,0,0,0,0,0,0,0
Input : 2 = 0,0,1,1,1,0,0,0,0,0,0,0
Input : 3 = 0,0,0,1,1,1,0,0,0,0,0,0
Input : 4 = 0,0,0,0,1,1,1,0,0,0,0,0
Input : 5 = 0,0,0,0,0,1,1,1,0,0,0,0
Input : 6 = 0,0,0,0,0,0,1,1,1,0,0,0
Input : 7 = 0,0,0,0,0,0,0,1,1,1,0,0
Input : 8 = 0,0,0,0,0,0,0,0,1,1,1,0
Input : 9 = 0,0,0,0,0,0,0,0,0,1,1,1

```

Fig. 5b. Interactive terminal to choose value of N (choosing “yes” option)

In the figure 5a, we chose the option “no”. Choosing no makes sure that N stays as it was set previously, this resulted in no updating of N to the required value to represent a distinct SDRs. Clearly as in the figure 5a, we can see that the encoding for input 4 and 5 are similar.

In the figure 5b, we chose the option “yes”. Choosing yes updated the value of N to the minimum required. Every data of the given input space was found to be distinct.

### E. Test Case generating the OverflowException

Instead of relying on N for encoding, this time we choose different values of Resolution for Encoding.

- Resolution = 1 to 10
- W = 3
- MinVal = 0
- MaxVal = 9
- Periodic = False
- ClipInput = True

While using the above parameters for encoding, we observed System.OverflowException. This is shown in the figure 6.a.

```

Terminal - MyProject

Inside the Main of ImproveScalarEncoder Namespace
-----Output for Resolution 1-----
Overflow exception at 1
-----Output for Resolution 2-----
Overflow exception at 2
-----Output for Resolution 3-----
Overflow exception at 3
-----Output for Resolution 4-----

```

Fig. 6.a. Output showing Exception for a given set of parameters

MaxVal	55
MinVal	11
N	-129
Ninternal	-131
Name	*ScalarEncod

Fig 6.b. Values of N when the Exception occurred

After observing the values of local variable using breakpoints, we found out that the value of N to be negative as shown in figure 6.b. This should be the reason for the overflow exception.

#### F. Test Case generating IndexOutOfRangeException

For this experiment, we chose Radius instead of N or Resolution for encoding.

Clearly as viewed on the figure 7, at certain value of Radius, IndexOutOfRangeException occurred. Even so, we can view the encodings at different input space.

```

Input : 7 = 0,0,0,0,0,1,1,1,0
Input : 8 = 0,0,0,0,0,0,1,1,1
Index out of range exception at 4
-----Output for Radius 5-----
Input : 0 = 1,1,1,0,0,0,0,0,0
Input : 1 = 0,1,1,1,0,0,0,0,0
Input : 2 = 0,1,1,1,0,0,0,0,0
Input : 3 = 0,0,1,1,1,0,0,0,0
Input : 4 = 0,0,1,1,1,0,0,0,0
Input : 5 = 0,0,0,1,1,1,0,0,0
Input : 6 = 0,0,0,0,1,1,1,0,0
Input : 7 = 0,0,0,0,0,1,1,1,0
Input : 8 = 0,0,0,0,0,0,1,1,1
Input : 9 = 0,0,0,0,0,0,1,1,1
-----Output for Radius 6-----
Input : 0 = 1,1,1,0,0,0,0,0,0
Input : 1 = 0,1,1,1,0,0,0,0,0
Input : 2 = 0,1,1,1,0,0,0,0,0
Input : 3 = 0,0,1,1,1,0,0,0,0
Input : 4 = 0,0,1,1,1,0,0,0,0
Input : 5 = 0,0,0,1,1,1,0,0,0
Input : 6 = 0,0,0,0,1,1,1,0,0
Input : 7 = 0,0,0,0,0,1,1,1,0
Input : 8 = 0,0,0,0,0,0,1,1,1
Index out of range exception at 6

```

Fig. 7. IndexOutOfRangeException for some input space

#### G. Test Case explaining the OverflowException

As a comment on test case E, we worked with various values of Resolution (without setting N or Radius). After experimenting and through debugging we found that System.OverflowException occurred because the default Radius configuration was negative (-1), which was set inside the EncoderBase.cs file. This resulted in negative value of N. Since the output array was initialized using the value:

$$\text{Output} = \text{int}[N] \quad (2)$$

This resulted in OverflowException.

```

public void Initialize(Dictionary<String, Object> encoderSettings)
{
    if (encoderSettings != null)
    {
        this.Properties.Clear();
        Name = this.GetType().Name;
        IsRealCortexModel = false;
        N = 0;
        Resolution = 0.0;
        Radius = 0.0;
        Periodic = false;
        ClipInput = false;
        foreach (var item in encoderSettings)
        {
            if (!this.Properties.ContainsKey(item.Key))
            {
                this.Properties.Add(item.Key, item.Value);
            }
            else
            {
                this.Properties[item.Key] = item.Value;
            }
        }
    }
    this.AfterInitialize();
}

```

Fig.8. Default values of parameters inside of EncoderBase.cs file

Setting the default value of Radius to 0, which was previously -1 inside the Initialize method of EncoderBase.cs file, N was no longer negative and OverflowException was solved. After solving this issue, IndexOutOfRangeException was observed in some input space.

#### H. Test Case explaining IndexOutOfRangeException

In the test case G, we solved the problem of OverflowException. After that we need to solve the problem of IndexOutOfRangeException. By experimenting with various inputs, we find out that the exception was due to the result of minbin or maxbin value, which denotes the starting position and ending position of the active bits.

We first find the centerbin and on adding padding to the left and right of the centerbin we get the values of minbin and maxbin. minbin represent the index position to start the encoding and maxbin represent the index position of the end of encoding for a value of N. The values of parameters used for encoding for this test case is :

- Resolution = 1 to 10
- W = 3
- MinVal = 13
- MaxVal = 29
- Periodic = False
- ClipInput = True

```

-----Input data 26-----Resolution 6-----
MinBin : 2
MaxBin : 4
N : 5
Input : 26 = 0,0,1,1,1
-----Input data 27-----Resolution 6-----
MinBin : 2
MaxBin : 4
N : 5
Input : 27 = 0,0,1,1,1
-----Input data 28-----Resolution 6-----
MinBin : 3
MaxBin : 5
N : 5
Exception thrown: 'System.IndexOutOfRangeException' in NeoCortexArrayLib.dll
Index out of range exception at 28
-----Output for Resolution 7-----
-----Input data 13-----Resolution 7-----
MinBin : 0
MaxBin : 2
N : 5
Input : 13 = 1,1,1,0,0

```

Fig. 9. Experiment showing IndexOutOfRangeException

Clearly in the above result in case of exception, minbin is 3 and maxbin equals 5. The value of N is 5 for this experiment. Minbin and maxbin represent the index position and index position starts at 0. This means that our encoder tries to set the data from N[3] to N[5]. The value of N is 5, because of this we can only go up to N[4] as indexing starts at 0 in c-sharp and most other programming language. This resulted in `IndexOutOfRangeException`.

#### I. Test Case Solving `IndexOutOfRangeException`

Experimenting with different values and by using breakpoints, we found out that there might be issue in the value of center bin or N. Center bin value after experimenting seems to be fine, logic seemed okay. So, we then looked for the value of N. Because of low value of N, the error seems to persist in some input space. To solve this, we changed the value of N in line 157 of `ScalarEncoder.cs`.

$$N = (\text{int})(n\text{Float}) \quad (3)$$

$$N = (\text{int})\text{Math.Round}(n\text{Float}) \quad (4)$$

The value of N was changed from equation 3 to equation 4. This makes sure that the decimal values are rounded to nearest whole number. This increased the value of N for some input space.

For the input parameters of test case H, the problem seems to be solved.

```
-----input data 26-----Resolution 6-----
MinBin : 2
MaxBin : 4
N : 6
Input : 26 = 0,0,1,1,1,0
-----input data 27-----Resolution 6-----
MinBin : 2
MaxBin : 4
N : 6
Input : 27 = 0,0,1,1,1,0
-----input data 28-----Resolution 6-----
MinBin : 3
MaxBin : 5
N : 6
Input : 28 = 0,0,0,1,1,1
-----input data 29-----Resolution 6-----
MinBin : 3
MaxBin : 5
N : 6
Input : 29 = 0,0,0,1,1,1
```

Fig. 10.a. Experiment solving `IndexOutOfRangeException` for the given parameter

But still the problem seems to be unsolved for some different input space. The parameters which resulted `IndexOutOfRangeException` were:

- Resolution = 1 to 10
- W = 3
- MinVal = 0
- MaxVal = 9
- Periodic = False
- ClipInput = True

Figure 10.b. shows the output of the above parameters.

```
-----input data 6-----Resolution 6-----
MinBin : 1
MaxBin : 3
N : 4
Input : 6 = 0,1,1,1
-----input data 7-----Resolution 6-----
MinBin : 1
MaxBin : 3
N : 4
Input : 7 = 0,1,1,1
-----input data 8-----Resolution 6-----
MinBin : 1
MaxBin : 3
N : 4
Input : 8 = 0,1,1,1
-----input data 9-----Resolution 6-----
MinBin : 2
MaxBin : 4
N : 4
Exception thrown: 'System.IndexOutOfRangeException' in NeoCortexArrayLib.dll
Index out of range exception at 9
```

Fig. 10.b. `IndexOutOfRangeException` still persists

After working with different settings we found out that changing the value of N in line 157 of `ScalarEncoder.cs` to equation 5 solved the issue of `IndexOutOfRangeException`.

$$N = (\text{int})\text{Math.Ceiling}(n\text{Float}) \quad (5)$$

```
-----input data 6-----Resolution 6-----
MinBin : 1
MaxBin : 3
N : 5
Input : 6 = 0,1,1,1,0
-----input data 7-----Resolution 6-----
MinBin : 1
MaxBin : 3
N : 5
Input : 7 = 0,1,1,1,0
-----input data 8-----Resolution 6-----
MinBin : 1
MaxBin : 3
N : 5
Input : 8 = 0,1,1,1,0
-----input data 9-----Resolution 6-----
MinBin : 2
MaxBin : 4
N : 5
Input : 9 = 0,0,1,1,1
```

Fig. 10.c. `IndexOutOfRangeException` solved for any input space

Using `Math.Ceiling` increased the value of N in case of any decimal value of `nFloat`. This resulted in increase in the number of positions for output bits. Due to this, calculated maxbin was always less than or equal to N.

#### J. Test Case Implementing Exception Handlers

For appropriately handling the Exception and to make sure that N, Radius and Resolution are mutually exclusive, `ArgumentException` check had been implemented. For a proper implementation of Exception handler, the default value of Resolution is changed to 0 inside the initialize method of `EncoderBase.cs` file. This facilitates proper logic check.

#### K. Test Case making `Scalar Encoder` callable using command line argument

In improved version of the `ScalarEncoder`, it can take input from both the dictionary settings and command line arguments. To work with the command line arguments, arguments passed on the Main function of our program can be directly passed to the `ScalarEncoder`. Because of function overloading, method that processes the command line arguments will be called instead of the method that takes dictionary as input for encoding.

- The command line arguments can be set on visual studio by going to the properties of our current solution and adding arguments.

- b. Command line arguments can also be passed directly from the console and the program can be run as shown below:

Listing 1. Command line code to encode the given input

```
dotnet run --project path-to-project-file --n 14 --w 3
--minval 1 --maxval 8 --periodic false --clipinput
```

In our case the path to the project file would be:

```
neocortexapi/source/ScalarEncoderImproved/Sc
alarEncoderImproved.csproj
```

```
bisalg@bisalg neocortexapi % dotnet run --project source/Samples/NeoCortexApiSample/
NeoCortexApiSample.csproj --n 14 --w 3 --minval 1 --maxval 8 --periodic false --clipin
put true
Inside the Main of ImproveScalarEncoder Namespace
Input : 1 = 1,1,1,0,0,0,0,0,0,0,0,0,0,0
Input : 1.6363636363636362 = 0,1,1,1,0,0,0,0,0,0,0,0,0,0
Input : 2.2727272727272725 = 0,0,1,1,1,0,0,0,0,0,0,0,0,0
Input : 2.9090909090909097 = 0,0,0,1,1,1,0,0,0,0,0,0,0,0
Input : 3.5454545454545455 = 0,0,0,0,1,1,1,0,0,0,0,0,0,0
Input : 4.181818181818182 = 0,0,0,0,0,1,1,1,0,0,0,0,0,0
Input : 4.818181818181818 = 0,0,0,0,0,0,1,1,1,0,0,0,0,0
Input : 5.454545454545455 = 0,0,0,0,0,0,0,1,1,1,0,0,0,0
Input : 6.090909090909092 = 0,0,0,0,0,0,0,0,1,1,1,0,0,0
Input : 6.727272727272728 = 0,0,0,0,0,0,0,0,0,1,1,1,0,0
Input : 7.363636363636365 = 0,0,0,0,0,0,0,0,0,0,1,1,1,0
bisalg@bisalg neocortexapi %
```

Fig. 11. Output of using command line for encoding

Proper exception handling is implemented if the supplied argument is not in a correct order or format.

#### IV. LINKS TO ACCESS THE PROJECT

All the implementations are implemented inside the forked repository of Neocortexapi inside ScalarEncoderImproved.

The link is :

<https://github.com/bisalg/neocortexapi/tree/master/source/ScalarEncoderImproved>

#### V. CONCLUSION

Working with the given scalar encoder, we found various exceptions and errors. We implemented proper logic and corrected the exception handling.

We made the program intuitive using the terminal. We found proper default values for the parameter so that proper check could be implemented inside the program.

Implementation of how N is calculated is changed. This makes sure that no exception is occurred. Also, interactive console prompt is implemented. This lets user decide if the existing value needs to be upgraded for discrete encoding.

#### VI. REFERENCES

- [1] Purdy, S., "Encoding Data for HTM Systems," ResearchGate, 2016.
- [2] Hawkings, J. & Ahmad, S., "Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex", <https://arxiv.org/abs/1511.00083v2>, 2015.