*Improve Scalar Encoder for NeocortexApi*

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

*Abstract*—Neocortexapi is a block of 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), 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. Encoder encodes any input in the form of SDRs, which are then processed by the neocortex. 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. Improving is done by using the appropriate value for parameters and by improving the interface.

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

# 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 spare 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 spare 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.

# Materials and methods

## Encoder Overview

First step of using neocortexapi, which is an implementation of HTM, is to transform the data to a form (SDRs) so that it can be digested by the 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 object can be said to be close to each other geographically if their SDRs overlap over a some threshold.

Important aspects that need to be considered when encoding data are :

* Semantically similar data should result in SDRs with overlapping active bits
* The 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 should have similar sparsity for all input and have enough one-bits to handle noise and subsampling.

## 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 time
* ClipInput : If true, value less than the minimum has SDR similar to minimum value and values more than the maximum has SDR similar to maximim value
* Resolution: For a particular value of resolution, input separated by more than or equal to resolution would have different representations. For eg,: 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.
* Total buckets that a SDR can represent = (N-W+1)

# 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.

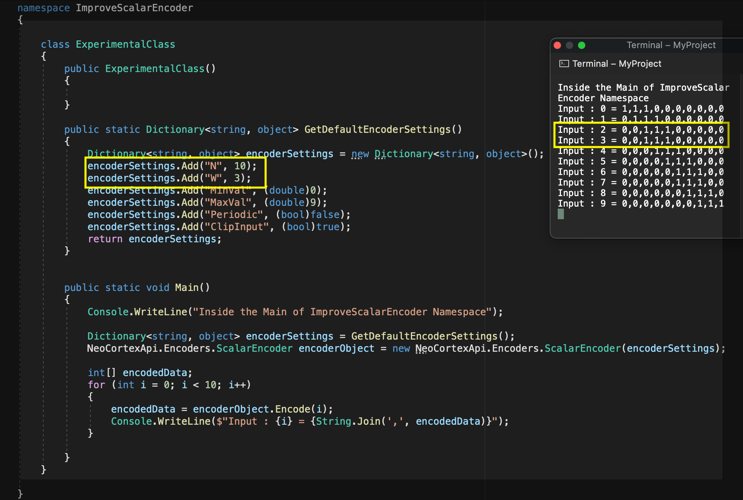
After finding the pitfalls, we will be experimenting to find the correct solution with a way to improve current version of scalar encoding.

1. Test Case generating the bucket size problem

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

1. N = 10
2. W = 3
3. MinVal = 0
4. MaxVal = 9
5. Periodic = False
6. ClipInput = True

The output with the above defined parameter is shown as :



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.

Input 2: 0011100000

Input 3: 0011100000

1. Test Case generating the bucket size problem – 2nd

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:

1. N = 10
2. W = 3
3. MinVal = 0
4. MaxVal = 9
5. Periodic = False
6. ClipInput = True

The output with the above defined parameter is shown as:

Text

Description automatically generated

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

Input 4: 0000111000

Input 5: 0000111000

1. 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 are:

1. N = 12
2. W = 3
3. MinVal = 0
4. MaxVal = 9

The output with the above defined parameter is shown as:

Text

Description automatically generated

The total number of buckets that our configuration can represent is = (12-3+1) = 10. Since we have to represent data from 0 to 9, which is 10 different SDRs represention. So the number of available bucket (10) is equal to the number of required bucket. This cause an discrete SDR representation of different input.

1. Test Case generating the OverFlow Exception

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

Graphical user interface, text

Description automatically generated

System.OverFlow exception was observed. Clearly as seen above, the value of N is negative.

1. Test Case generating IndexOutofRange Exception

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

Graphical user interface, text

Description automatically generated

Clearly as viewed on the image above, at certain value of Radius (which is random values taken – no pattern), IndexOutofRange Exception occurred. Even so, we can view the SDRs at different input space.

1. Test Case explaining the OverFlow Exception

As a comment on test case 4, we worked with various values of Resolution(Without setting N or Radius). After experimenting and throw debugging we found that System.OverFlowException was occurred because the default Radius configuration was negative (-1), which is set inside the Encoders\_Base.py file . This resulted in negative value of N. Since the output array was initialized using the value : output = int[N], this resulted in OverFlowException. Image is attached as description.

Setting the default value of Radius to 0, N was no longer negative and OverFlowException was solved. After solving this issue, IndexOutofRangeException was observed in some input space.

1. Test Case explaining IndexOutofRange Exception (TODO again)

In the test case 6, we solved the problem of OverFlow Exception. After that we need to solve the problem of IndexOutofRange Exception. By experimenting with various inputs, we find out that the exception as a result of minbin or maxbin value, which denotes the starting position and ending position of the active bits.

Text

Description automatically generated

As shown in the image above, the maxbin value was 4 whereas our array could only hold data for position up to 3.

1. Test Case Solving IndexOutofRange Exception

Experimenting with different values and using breakpoints, we found out there 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 less number 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 to N = (int)Math.Round(nFloat). This solved the problem for our given input, W = 3, MinVal = 13, MaxVal = 29, Resolution 1-10. The problem seems to be solved for the given input.

Text

Description automatically generated

But still the problem seems to be unsolved for some different input space. With certain input space (W=3, MinVal = 0, MaxVal = 9, Resolution 1-10.

Text

Description automatically generated

After working with different settings we found out that changing the value of N in line 157 of ScalarEncoder.cs to N = (int)Math.Ceiling(nFloat) solved the issue of IndexOutofRange Exception.

Text

Description automatically generated

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

1. Test Case Implementing Exception Handlers and Interactive prompt

For appropriately handling the Exception, ArgumentException check had been implemented. For a proper implementation of Exception handler, the default value of Resolution is changed to 0 in EncoderBase.cs. This facilitates the implementation of Exception Handlers.

The ScalarEncoder is made interactive by giving the User option to enter data on the console. Based on the input string by the User, the value of N was set and decisions was taken accordingly for SDR generation.

1. Test Case making Scalar Encoder callable using command line argument

The ScalarEncoder has been modified so that 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, ScalarEncoder(String[] args) will be called on our program instead of ScalarEncoder(Dictinary <..>).

1. The command line arguments can be set on visual studio by going to the properties of our current solution and adding arguments.
2. Command line arguments can also be passed directly from the console and the program can be run as shown below:

“dotnet run --project project.csproj --n 14 --w 3 --minval 1 --maxval 8 --periodic false --clipinput true”

Exception(System.FormatException) is handled if the supplied arguments are not convertible to desired type.

# Links to acces the test cases

TODO

# Conclusion

Concluison of the finds and result

##### References

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*
2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
3. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
4. K. Elissa, “Title of paper if known,” unpublished.
5. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
6. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
7. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989