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

Implementation of Scalar Encoder in HTM

|  |  |  |
| --- | --- | --- |
| Aqib Javed | Haris Abbas Qureshi | Saad Jamil |
| aqib.javed@stud.fra-uas.de | [haris.abbas@stud.fra-uas.de](mailto:haris.abbas@stud.fra-uas.de) | [saad.jamil@stud.fra-uas.de](mailto:saad.jamil@stud.fra-uas.de) |

***Abstract*— Scalar Encoder is one of the encoding techniques and is a part of Hierarchical Temporal Memory (HTM). The Scalar Encoder's main purpose is to convert scalar values (such as numeric or floating-point values) into a distributed representation that can be used by HTM algorithms. It does this by transforming the input value into a binary array of bits, where each bit position corresponds to a different feature or dimension of the input value. The Scalar Encoder generates a sparse distributed representation, where only a small subset of bits in the array are active (i.e., set to 1) for any given input value. The active bits form a contiguous block of 1's, whose location within the array varies continuously depending on the input value.**

***Keywords- HTM, neocortex, numeric, array, block***

1. **INTRODUCTION**

In Hierarchical Temporal Memory (HTM) systems, Sparse Distributed Representations (SDRs) are used to represent input data. SDRs consist of a large binary array of bits, where most of the bits are set to 0 and only a small subset of bits is set to 1. To convert raw input data into SDRs, various encoding techniques are used, including Scalar Encoder, Category Encoder, Date/Time Encoder, and others. These encoders map the input data to SDRs in a way that preserves the important semantic characteristics of the data. When multiple input values are encoded into SDRs, those SDRs are likely to overlap with each other, particularly if the inputs are similar. This overlap between SDRs is a desirable property, as it allows the HTM system to recognize patterns and make predictions based on the input data.

In HTM systems, SDRs are also used to represent the output of the system. For example, if the system is trained to classify input data into different categories, the output will be an SDR representing the most likely category for the input. This SDR can then be compared to other SDRs representing known categories to determine which category the input belongs to.

# METHODOLOGY

Encoder should create SDRs, no matter what it represents, that have a fixed number of bits ‘N’ and fixed number of active (1’s) bits ‘W’. What do you know which values are perfect for N and W?

We cannot be a big fraction of N, to preserve the properties that come from sparsity. But if W is too small then we lose the properties resulting from a distributed representation.

There are a number of specific aspects to consider when encoding the data:

1. Choosing appropriate values for N and W: N should be large enough to allow for a detailed representation of the input data, while W should be a small fraction of N to preserve sparsity. A common range for N is between 100 and 1000, while W is typically around 2-5% of N.

2. Preserving semantic relationships: The encoder should be designed to capture semantically related data, so that inputs with overlapping characteristics will generate overlapping SDRs with active bits in common.

3. Deterministic output: The encoder should always generate the same output SDR for the same input, to avoid redundancy in the learned sequence in HTM systems.

4. Fixed output dimensionality: The encoder's output should always have the same number of bits, regardless of the input, to enable comparisons and other operations.

5. Robustness to noise and subsampling: The encoder should include enough one-bits to accommodate noise and subsampling, with a general rule of thumb of at least 20-25 one-bits per SDR.

When we construct an encoder implementation, we first divide the range of values into buckets and then map the buckets into a collection of active cells.

The index i is calculated based on the input value, using the formula i = (input value - MinVal) \* (N - W) / Range.

The W bits starting from index i are set to active to represent the input value.

The remaining bits in the output are left unset.

This process is repeated for each input value, resulting in an encoded representation for the entire dataset. The resulting output will be a binary vector of length N, with W bits set to 1 in each segment that represents a particular input value. The encoder is deterministic, so the same input value will always result in the same binary vector representation. This allows for efficient comparison and processing of the encoded data in downstream algorithms such as HTM.

The bucketing approach used in this encoder implementation helps to group similar input values into the same collection of active cells, making it easier to identify patterns and similarities in the data. The choice of MinVal, MaxVal, W, and N will depend on the specific characteristics of the input data, such as the range of values, resolution required, and the desired level of noise tolerance. It's important to strike a balance between having enough bits to represent the input values accurately and

efficiently, while also ensuring that the resulting binary vector is sparse enough to avoid overlap and maintain distinguishability between different input.