HTM Model Code Documentation

September 13, 2017

Objective The larger aim is to build a model to explain how a song bird learns songs (a sequence of syllables) from a tutor adult. The goal of the current goal is to build a system that can learn a given set of sequences of varying length.

1 Network Structure

The model mainly works with a layer of cells. The layer of cells is in the form of m rows and n columns. Each cell has a set of d distal segments with potentially s synapses connecting to any other cell in the layer.

If the synaptic weight cw of a particular synapse is above the connection threshold $beta$, the synapse between the two cells is considered connected. If more than $theta$ synapses in a distal segment are connected to active cells, the segment is considered to be active.

2 Sequences

The model is trained on sequences of the type $[A, B, C, D]$. The model, currently, supports learning of a sequence of upto 7 syllables, though it is potentially scalable. It can also learn multiple sequences simulataneously. Specially, it is capable of remembering the context while predicting. For example, it can predict sequences of the following types accurately, when only the first syllable is provided as feed-forward input :-

Seq #1: $[A, B, C, D, E]$
Seq #2: $[F, B, C, D, G]$

3 Syllable Encoding

Each syllable is encoded as a unique set of n bits (corresponding to each column). For each syllable, an encoder picks b random bits out of n bits to represent it. Currently, it is constrained to a non-overlapping representation, i.e., the encoding of two syllables will not

have any common bits. The model hasn't been tested for an overlapping representation, yet. A syllable is considered predicted, if for each of these b corresponding columns, at least 1 of the m cells is activated.
Eg. The letter 'C' can be encoded as shown below, for $n = 21$. For this encoding, if a prediction matrix of the following type is observed, the syllable 'C' can be considered predicted.

```
'C' :   [0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0]

        - - - - - - - + - - - - - - - - - - - - -
        - - - - - - - - - - - - - - - - - - - - -
        - - - - - - - - - - - - - - - - - - - - -
        - - - - - - - - - - - - - - - - - - - - -
        - - - - - - - - - - - - - - - - - - - - -
        - - - - - - + - + - - - - - - - - - - - -
```

4 Data Structures

We have majorly 4 elements :-

1. Feed-forward input **W**: A binary vector denoting the columns that are required to be active for the corresponding syllable.

2. Activation matrix **A**: A binary $m * n$ matrix, each for the previous timestep $[t-1]$ and for the current timestep $[t]$, where an 'ON' bit implies the i^{th} cell on the j^{th} column is active, and an 'OFF' bit implies the corresponding cell is inactive.

3. Prediction matrix **P**: A binary $m * n$ matrix, each for the previous timestep $[t-1]$ and for the current timestep $[t]$, where an 'ON' bit implies the corresponding cell will become active in the next time step, if the cell is in a winning column[1].

4. Segment matrix **D**: A float $m * n * d$ matrix, which stores information regarding the s synapses from each segment. It contains tuples denoting the cell that a synapse is connected to and the weight cw $(\epsilon[0, 1])$ of the synapse.

5 Parameters

A list of parameters that are being used currently are presented in Table **??**.

[1]Winning column denotes a column selected by the feed-forward input, dictated by the corresponding syllable in the sequence.

6 Algorithm

6.1 Initialisation

1. The activation matrices, $A[t]$ and $A[t-1]$ are initialised to zero, implying no cell is active at the beginning.

2. The prediction matrices, $P[t]$ and $P[t-1]$ are also initialised to zero, depicting no cell in the predictive state.

3. Each synapse in the distal segments of the cells is initialised with a connection to a random cell (other than itself) in the layer, with a random connection weight between 0 and $initialisingLimit$ ($<= beta$).

6.2 Learning

Each sequence is trained for a few consecutive trials, before giving way for the next sequence to be trained. This is done in a circular fashion. While training a particular sequence, the feed-forward input vector, W, is initialised with the encoding of each syllable, turn by turn. This vector W^t holds information about the winning columns for this timestep.

Table 1: Parameters with values being used

Type	Parameter	Symbol	Value	Remarks
Layer	No. of columns	n	21	Scalable
	No. of rows	m	6	
	No. of segments per cell	d	1	Not scalable
	No. of synapses per segment	s	5	Works for higher
Synapse	Synaptic connection threshold	$beta$	0.5	
	Segment activation threshold	$theta$	3	Works for higher
	No. of rows	$initialisingLimit$	0.5	Must be $<= \beta$
Learning	Long term potentiation	$pPos$	0.2	
	Long term depression	$pNeg$	0.2	
	Decay	$pDec$	0.02	
	No. of trials in training	$nTrainingTrials$	200	
	No. of consecutive trials for a sequence	$nRepConsecutive$	5	
	Choose best segment acc. to algo	$chooseMax$	0	=> Random
	Selected according to	$maxCondition$	0	Immaterial
	Synapse replacement	$replaceSynapses$	1	=> ON
Result	Representation in json file	$repr\_matrix\_form$	0	printing format
Testing	Type of testing	$testFreeFlag$	1	Constrained/Free

For each winning column, it is checked if there are cells in the predictive state. If this is the case, the distal segments, which are responsible for this prediction, are reinforced, similar to Hebbian learning. We choose segments which satisfy the following condition:

$$\forall j : W^t[j] > 0, P_{ij}^{t-1} > 0 and ||\tilde{D}_{ij}^d \circ A^{t-1}||_1 > \theta \tag{1}$$

where $\tilde{D}$ is a matrix containing only connected synapses. All such segments, belonging to an active cell, are reinforced where the no. of synapses connected to active cells from the previous timestep is above the activation threshold. These segments are reinforced as follows:

$$\Delta D_{ij}^d = p^+(\dot{D}_{ij}^d \circ A^{t-1}) - p^- \dot{D}_{ij}^d \tag{2}$$

where $\dot{D}$ denotes a matrix containing synapses with positive weights only.

If no cell in the winning column is in a predictive state, a random cell is chosen from the column to represent it. A random segment belonging to this column is reinforced, as described above. Furthermore, one of the synapses of this chosen segment, which is not connected to an active cell from the previous timestep, is replaced by a connection to a random active cell from the previous timestep.

In addition, a decay is applied to active segments of cells which did not become active, as follows:

$$\Delta D_{ij}^d = p^{--} \dot{D}_{ij}^d \text{ where } A_{ij}^t = 0 \text{ and } ||\tilde{D}_{ij}^d \circ A^{t-1}||_1 > \theta \tag{3}$$

The D matrix is updated with ΔD matrices at each time step.

6.3 Computing cell states

Activation matrix A cell is activated if it was in predictive state and is in a winning column for the current timestep. If no cell is in predictive state in a particular winning column, then all the cells in that column are activated, as below.

$$A_{ij}^t = 1 \text{ if } \forall j \epsilon W^t, P_{ij}^{t-1} = 1 \text{ or } \Sigma_i P_{ij}^{t-1} = 0 \tag{4}$$

All these active cells are stored in a list, for reference in the next timestep, while replacing synapses.

Prediction matrix A cell goes into predictive state if at least 1 distal segment of the cell crosses the activation threshold, i.e. has more than θ synapses connected to a currently active cell.

$$P_{ij}^t = 1 \text{ if } \exists D_d : ||\tilde{D}_{ij}^d \circ A^t||_1 > theta \tag{5}$$

7 Testing

In order to test a particular sequence, a syllable is provided as feed-forward input and the predictive state of that timestep is computed. The next syllable is considered predicted, if there exists at least one cell in the predictive state from every column required to represent that syllable. Then, this predictive state is used to compute the next activation matrix without loss of context information. This is repeated for every syllable in the sequence, except the last.

The model was also tested in a less controlled environment. Only the first syllable is provided as feed-forward input and a prediction is computed. The prediction of this timestep is the sole determinant of the activation in the next timestep. This continues for the rest of the timesteps in the sequence. One can toggle between these two forms of testing flipping the $testFreeFlag$.

8 Results

1. The model is able to learn sequences of length 2-7 within 50-100 trials, independently. Longer sequences require the layer to be expanded. On being a presented an intermediate syllable, the model is able to predict the remaining sequence. [Results in folder TestLength]
   $[A, B]$; $[A, B, C]$; $[A, B, C, D]$; $[A, B, C, D, E]$; $[A, B, C, D, E, F]$; $[A, B, C, D, E, F, G]$

2. The model is able to learn upto 4 sequences consecutively within 250 trials. Testing longer sequences will require the layer to be expanded. As is the case above, the model is able to predict the sequence starting from any position. [Results in folder TestConsecutive]
   $[A, B]$, $[C, D]$, $[E, F]$;
   $[A, B, C]$, $[C, D, E]$, $[E, F, G]$;
   $[A, B]$, $[B, C]$, $[C, D]$, $[D, E]$;
   $[A, B]$, $[C, B]$, $[D, E]$, $[F, E]$, $[G, A]$

3. It is able to correctly predict the sequences while remembering the context, within 250 trials, when trained simultaneously with 1 distal segment per cell. [Results in folder TestContext]
   $[A, B, C]$, $[D, B, E]$, $[F, B, G]$;
   $[A, B, C, D]$, $[E, B, C, F]$;
   $[A, B, C, D, E]$, $[F, B, C, D, G]$

4. In a version of the model, where segments have 1 synapse each to every cell in the layer, the sequences of the above type can be learnt with multiple segments per cell, as well. [Results in folder HTM_distinct]

8.1 Sample Result

For instance, shown below is the learning of the sequences $[A, B, C]$ and $[D, B, E]$.

- The first syllables in both the sequences, 'A' and 'D' can never be predicted. Hence, all the cells, in the columns that represent their encodings, become active.
- The second syllable is 'B' in both the sequences. However, the cells active are different for the 2 sequences.
- The predictions for the last syllable in the 2 sequences are different. This happens as the the active cells for the syllable 'B' are different, depending on their context.

```
Timestep #1
'A':                                                    'D':
[1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  [0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0]
 + + + - - - - - - - - - - - - - - - - - - - - -    - - - - - - - - - - + + + - - - - - - - - -
 + + + - - - - - - - - - - - - - - - - - - - - -    - - - - - - - - - - + + + - - - - - - - - -
 + + + - - - - - - - - - - - - - - - - - - - - -    - - - - - - - - - - + + + - - - - - - - - -
 + + + - - - - - - - - - - - - - - - - - - - - -    - - - - - - - - - - + + + - - - - - - - - -
 + + + - - - - - - - - - - - - - - - - - - - - -    - - - - - - - - - - + + + - - - - - - - - -
 + + + - - - - - - - - - - - - - - - - - - - - -    - - - - - - - - - - + + + - - - - - - - - -

Timestep #2
'B':                                                    'B':
[0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  [0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
 - - - - - - - - - - - - - - - - - - - - - - - -    - - - - - + - - - - - - - - - - - - - - - -
 - - - - - - - - - - - - - - - - - - - - - - - -    - - - - - - - - - - - - - - - - - - - - - -
 - - - - + - - - - - - - - - - - - - - - - - - -    - - - + - - - - - - - - - - - - - - - - - -
 - - - - - + - - - - - - - - - - - - - - - - - -    - - - - - - - - - - - - - - - - - - - - - -
 - - - - - - - - - - - - - - - - - - - - - - - -    - - - - + - - - - - - - - - - - - - - - - -
 - - - + - - - - - - - - - - - - - - - - - - - -    - - - - - - - - - - - - - - - - - - - - - -

Timestep #3
'C':                                                    'E':
[0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  [0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0]
 - - - - - - - - - - - - - - - - - - - - - - - -    - - - - - - - - - - - - - + - + - - - - - -
 - - - - - - - - - - - - - - - - - - - - - - - -    - - - - - - - - - - - - - - + - - - - - - -
 - - - - - - - - - - - - - - - - - - - - - - - -    - - - - - - - - - - - - - - - - - - - - - -
 - - - - - - + - - - - - - - - - - - - - - - - -    - - - - - - - - - - - - - - - - - - - - - -
 - - - - - - - - - - - - - - - - - - - - - - - -    - - - - - - - - - - - - - - - - - - - - - -
 - - - - - - - + + - - - - - - - - - - - - - - -    - - - - - - - - - - - - - - - - - - - - - -
```

The model was tested by depriving it of the context information, i.e. the sequences were prompted by an overlapping syllable as the first syllable. This lead to all possibilities being predicted, as follows.

```
Timestep #1
'B': [0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      - - - + + + - - - - - - - - - - - - - - - -
      - - - + + + - - - - - - - - - - - - - - - -
      - - - + + + - - - - - - - - - - - - - - - -
      - - - + + + - - - - - - - - - - - - - - - -
      - - - + + + - - - - - - - - - - - - - - - -
      - - - + + + - - - - - - - - - - - - - - - -

Timestep #2
'C': [0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0]
'E': [0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0]
      - - - - - - + - - - - - - - - - - - - - - -
      - - - - - - - + + - - - + - + - - - - - - -
      - - - - - - - - - - - - - + - - - - - - - -
      - - - - - - + - + - - - - - - - - - - - - -
      - - - - - - - - - - - - - - - - - - - - - -
      - - - - - - - + + - - - - + - - - - - - - -
```

9 Versions

1. This is the main version that has been described in this document.
   Link: `https://github.com/rsankar9/HTM_model`

2. This version has no differences apart from the representation. The activation and prediction matrices are stored using n integers, instead of n columns, having m rows each. Each bit of an integer represents the corresponding row in the column. This has been discarded as no significant optimisation was achieved.
   Link: `https://github.com/rsankar9/HTM_model/tree/rsankar9-patch-1`

3. This version has fully connected segments, i.e. each segment has 1 synapse each to every other cell in the layer. This synapse has a connection weight between 0 and 1. Note that a segment can't have multiple synapses to the same cell, unlike the main version. Furthermore, it is significantly slower than the other versions.
   Link: `https://github.com/rsankar9/HTM_model/tree/rsankar9-patch-2`