Analyse and Describing Boosting Algorithm

Chinmaya Nithin Dimmiti
chinmaya.dimmiti@stud.fra-uas.de

Kiran Kumar Athirala
kirankumar.athirala@stud.fra-uas.de

Sandhya Bagadi
sandhya.bagadi@stud.fra-uas.de

***Abstract* —This paper represents Analyzing and Describing of Boosting Algorithm which uses Spatial pooler and Spatial Pattern Learning. Boosting Algorithm uses Spatial Pooler and Spatial Pattern learning, this algorithm uses internally a boosting algorithm which makes sure that the unused mini columns and weak synapses periodically get boosted. The main idea behind boosting is that the SDR's of symbols uses a wider range of cells by making the most active cells less active and least active cells more active. This is done via scalar multiplication of a boosting matrix over the spatial pooler to change permeances of each cell. It may be concluded that the boosting process has an impact on the SP stability. The SP can reach a stable state, but SDRs with a drastically different number of active mini columns will result. If boosting is turned on, the SP will activate mini columns equitably, but the learning will be unstable. We also performed experiment to obtain stability of Spatial Pooler output by tuning the boost and duty cycles. We evaluated each of these parameters based on the theoretical and practical framework and summarized the results in graphical diagrams.**

Keywords—Spatial Pooler compute, Update Duty Cycles, Bump Up Weak Columns, Update Boost Factors, Update Min Duty Cycles.

I. INTRODUCTION

Hierarchical temporal memory (HTM) is a neuromorphic machine learning algorithm which resembles neocortex functions in the human brain. The HTM architecture comprises a spatial pooler (SP) and temporal memory (TM) with the sparsely, modular and hierarchically characteristical components. The SP sparsely distributes the input data while the TM is in charge of the learning process. HTM starts with the presumption of the memory and the confirmation of sequence of patterns that all the neocortex does. [1]

The spatial pooler takes the input data and translates the incoming data into active columns. Let's assume that for a given input space a spatial pooling tries to learns the sequences, to learn the sequence every mini column is connected to certain amount of synapses from the input. Then the overlap score is calculated and if the overlap score is above some permeance or threshold value the column is activated else it is not. Let's say the threshold value is 50 so all the columns with overlap score more than 50 will get activated.

In order for a column in a Spatial pooler to exist it should be a winning column i.e the overlap score should be above some threshold value while non-winning columns are inhibited from learning. Only the winner columns can update their permanence values. Boosting helps to change the overlap score before the inhibition occurs giving less active columns a better chance to express themselves and diminishing columns that seem overactive . Boosting on better enables the learning of input data i.e it improves the efficiency. In other words we can say that the columns that have low overlap score are boosted so that they can better express themselves and all the columns with higher overlap score are inhibited because they are expressing themselves too much [2]. The figure 1 shows the conversion of input bits to SDR.

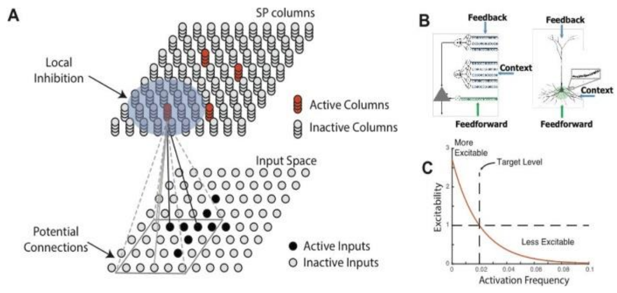


Figure 1The HTM spatial pooler converts inputs (bottom) to SDRs (top). [1]

The main objective of this paper is to analyse various Boosting parameters namely Bump Up Weak Columns, Update Boost Factors, Update Duty Cycles, Update Min Duty Cycles, MaxBoost and observe how they influence the Stability of the spacial pooler. The rest of the paper is structured as follows; Section 2 covers HTM Overview. Section 3 explains each of the methods used in HTMconfig. Section 4 contains the cases which were implimented, section 5 and 6 has the code and the unit tests of each method. Experimental results and conclusion are presented in section 7 and 8 respectively.

II. HTM OVERVIEW

The HTM network is made of regions that are arranged hierarchically as shown in the figure 2. Each of these regions has neurons known as cells. These cells are arranged vertically forming a column such that it responds to one single specific input at a time. These cells can be considered as major units of HTM. These cells also have dendrites, both distant and proximal allowing them to connect with input spaces and neighbouring cells in that particular area.

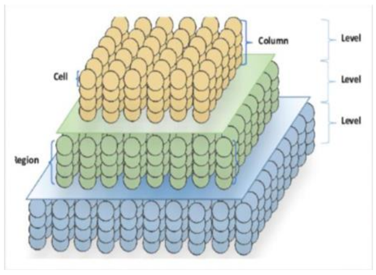


Figure 2: A representation of three layers of HTM arranged hierarchically. Each region contains columns of cells arranged vertically.

A. Sparse Distributed Representations:

Sparse Distributed Representations (SDRs) are binary representation i.e. an array consisting of large number of bits where small percentage are 1's represents an active neuron and 0 an inactive one. Each bit typically has some meaning (such as the presence of an edge at a particular location and orientation). This means that if two vectors have 1s in the same position they are semantically similar in that attribute.

B. Encoding:

The encoder converts the native format of the data into an SDR that can be fed into an HTM system, it is the incharge of identifying which output bits should be ones and which should be zeros for a particular input value in order to capture the data's significant semantic properties. SDRs with similar input values should have a high degree of overlap.

C. Spatial Pooler:

Input patterns that are spatially similar should have a common internal representation, also sparse to abide by the principles of SDR. When presented with input data, all columns in an HTM region will compute their feedforward activation. Columns that become active are allowed to inhibit neighbouring columns. For each of the active columns, permanence values of all the potential synapses are adjusted. The permanence values of synapses aligned with active input bits are increased, whereas permanence values of synapses aligned with inactive input bits are decreased [2]. Figure 3 shows a flow chart of the phases involved.

The spatial pooling operations are as follows:

Phase 0 (corresponding to initialization): Each column is randomly assigned a random set of inputs (50 percent of the input vector), which is referred to as the *potential pool* of the column. Each input within this pool is represented by a potential synapse and assigned a random permanence value.

Phase 1 (corresponding to overlap): The overlap for each column is computed as the number of connected synapses with active inputs multiplied by its boost. If this value is below a predefined threshold ("minOverlap"), the overlap score is set to 0.

Phase 2 (corresponding to inhibition): The number of winning columns in a local area of inhibition (neighbourhood of a column) is set to a predefined value, *N*. A column is a winner if its overlap score is greater than the score of the *N*th highest column within its inhibition radius. A variation of this inhibition strategy that is significantly less computationally demanding is picking the columns with the highest overlap scores for every level of the hierarchy.

Phase 3 (corresponding to learning): During this phase, For winning columns, if a synapse is active, its permanence value is incremented; if inactive, it is decremented. There are two separate boosting mechanisms in place to help a column learn connections. If a column does not win often enough, its overall boost value is increased; alternatively, if a column's connected synapses do not overlap well with any inputs often enough, its permanence values are boosted. This phase terminates with updating the inhibition. [3]

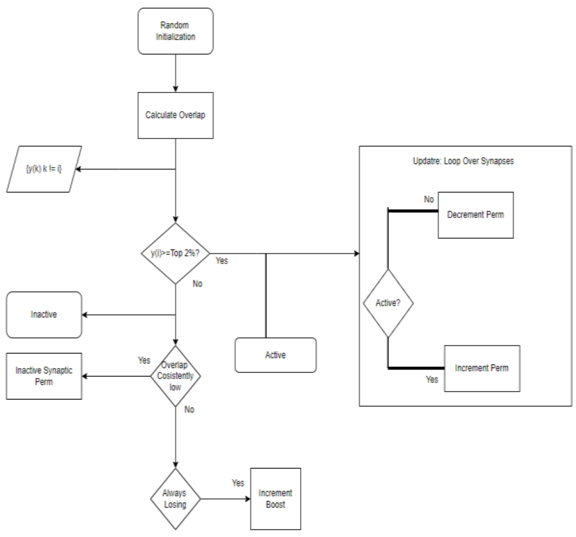


Figure 3: Spatial pooler flowchart [6]

III. Methodology

Following are the parameters on which we have worked on to observe how they affect the learning of input sequence in SP layer.
There are two types of boosting used in the spatial pooler algorithm:

Synaptic Boost of inactive mini columns:

A mini column is defined as inactive if the number of its connected synapses at the proximal dendrite segment is not sufficient in a learning cycle. If the number of connected synapses of a mini column in the cycle to the current input is less than the stimulus threshold, then permeance values of all potential synapses of a mini column will be slightly incremented by parameter 'stimulus increment'.

A. Bump Up Weak Columns:

This method ensures that each column has enough connections to input bits to allow it to become active. Since a column must have at least stimulus threshold overlaps to be considered during the inhibition phase, columns without such minimal number of connections, even if all the input bits they are connected to turn on, have no chance of obtaining the minimum threshold. For such columns, the permeance values are increased until the minimum number of connections are formed.

Uniform Activation of mini columns:

This implement makes sure that all mini columns in the HTM area become uniformly activated. The absence of this kind of plasticity leads to very different sparsity in the HTM area, which leads to incorrect prediction and inaccurate learning. To ensure the uniform participation of mini columns in the learning, the overall column overlap, and activation are considered. If the value of synaptic permeances defines energy stored in the synaptic connection, the goal here is to keep that energy uniformly distributed across the entire cortical area.

B. Update Boost Factors:

The boost factors are used to increase the overlap of active columns to improve their chances of becoming active, and hence encourage participation of more columns in the learning process. This means that columns that have been active enough have a boost factor of 1, meaning their overlap is not boosted. Columns whose active-duty cycle drops too much below that of their neighbours are boosted depending on how infrequently they have been active. The more infrequent, the more they are boosted. Boost factors are not recalculated if minimum active-duty cycles are all set on 0.

$$\text{boost} = \frac{(1-\text{maxBoost})}{\text{minDuty*activeDutyCycle} + \text{maxBoost}}.$$

C. Update Duty Cycles:

This is a helper function that is used to update several duty cycle variables in the column class such as overlap duty cycle, active-duty cycle, min Pct duty cycle and returns the updated duty cycle.

$$\text{dutyCycle} = \frac{(\text{ Period - 1}) * \text{dutyCycle} + \text{newValue}}{\text{period}}$$

D. Update Min Duty Cycles:

It updates the minimum duty cycles for SP which uses global inhibition and sets the minimum duty cycles for the overlap and activation of all columns to be a percent of the maximum in the region, specified by Min Overlap Duty Cycles and min Pct Active-Duty Cycle respectively.

E. Max Boost:

The Maximum overlap boost factor, each columns overlap gets multiplied by a boost factor before it considered for inhibition. The actual boost factor for a column is between 1.0 and the maxboost.

F. Duty Cycle:

A time duration cycle which notices that which column is active for which time after inhibition. This is also called Active duty cycle which call as a function. In the code we are setting Duty_Cycle_Period to vary this parameter.

G. Stimulus Increment:

Synapses of weak mini columns will be stimulated by the boosting mechanism, This stimulation is done by adding this increment value to the current permanence value of the synapse.

H. Stimulus Threshold:

This value is used by SP. When some permanence is under this value, it is set to zero. In this case the synapse remains the potential one and still can participate in learning.

IV. Cases

Executed the program using Spatial Pattern Learning experiment by updating/changing different boosting parameters to observe the effect of these parameters on overall stability, the cases are shown below.

Cases 1, 2 and 3 are performed by changing the values of duty cycle period and max boost while having all other values constant.

Case 1. Effect of Duty Cycle Period and max Boost (50000 & 1.0) on Spatial pattern learning. Parameters are are shown in Table1.

```
double minOctOverlapCycles = 1.0;
double maxBoost = 1.0;
int countval = 0;
int inputBits = 200;
int numColumns = 2048;
CellsPerColumn = 10,
MaxBoost = maxBoost,
count = countval,
DutyCyclePeriod = 50000,
GlobalInhibition = true,
NumActiveColumnsPerInhArea = 0.02 *     numColumns,
PotentialRadius = (int)(0.15 * inputBits),
LocalAreaDensity = -1,
MaxSynapsesPerSegment = (int)(0.01* numColumns),
Random = new ThreadSafeRandom(42),
StimulusThreshold = 10,
double max = 100;
```

Table: 1

Case 2. Effect of Duty Cycle Period and max Boost (100000 & 5.0) on Spatial pattern learning.

The values of different parameters are shown in table 2:

```
int countval = 0;
int inputBits = 200;
int numColumns = 2048;
CellsPerColumn = 10,
MaxBoost = maxBoost,
count = countval,
DutyCyclePeriod = 100000,
GlobalInhibition = true,
NumActiveColumnsPerInhArea = 0.02 *      numColumns,
PotentialRadius = (int)(0.15 * inputBits),
LocalAreaDensity = -1,
MaxSynapsesPerSegment = (int)(0.01* numColumns),
Random = new ThreadSafeRandom(42),
StimulusThreshold = 10,
double max = 100;
```

Table: 2

Case 3. Effect of Duty Cycle Period and max Boost (150000 & 10.0) on Spatial pattern learning.

The values of different parameters are shown in table 3:

```
double minOctOverlapCycles = 1.0;
double maxBoost = 10.0;
int countval = 0;
int inputBits = 200;
int numColumns = 2048;
CellsPerColumn = 10,
MaxBoost = maxBoost,
count = countval,
DutyCyclePeriod = 150000,
GlobalInhibition = true,
NumActiveColumnsPerInhArea = 0.02 * numColumns,
PotentialRadius = (int)(0.15 * inputBits),
LocalAreaDensity = -1,
MaxSynapsesPerSegment = (int)(0.01* numColumns),
Random = new ThreadSafeRandom(42),
StimulusThreshold = 10,
double max = 100;
```

Table 3

Case 4. Effect of Duty Cycle Period, max Boost and IsBumpUpWeakColumns (true) on Spatial pattern learning.

This experiment is performed by having the Is Bump Up WeakColumns parameter as true, table 4.

```
double minOctOverlapCycles = 1.0;
double maxBoost = 5.0;
int countval = 0;
int inputBits = 200;
int numColumns = 2048;
CellsPerColumn = 10,
MaxBoost = maxBoost,
count = countval,
DutyCyclePeriod = 100000,
GlobalInhibition = true,
NumActiveColumnsPerInhArea = 0.02 *      numColumns,
PotentialRadius = (int)(0.15 * inputBits),
LocalAreaDensity = -1,
//IsBumpUpWeakColumnsDisabled = true,
MaxSynapsesPerSegment = (int)(0.01* numColumns),
Random = new ThreadSafeRandom(42),
StimulusThreshold = 10,
double max = 100;
```

Table: 4

Case 5. Effect on Spatial pattern learning while Duty Cycle Period, max Boost, inputBits, numColumns, and double max are changed.

This experiment is performed by changing the values of duty cycle period, max boost, inputBits, numColumns, double max while keeping all other values constant. The values of different parameters are shown in table 5.

```
double minOctOverlapCycles = 1.0;
double maxBoost = 5.0;
int countval = 0;
int inputBits = 100;
int numColumns = 1024;
CellsPerColumn = 10,
MaxBoost = maxBoost,
count = countval,
DutyCyclePeriod = 10000,
GlobalInhibition = true,
NumActiveColumnsPerInhArea = 0.02 * numColumns,
PotentialRadius = (int)(0.15 * inputBits),
LocalAreaDensity = -1,
MaxSynapsesPerSegment = (int)(0.01* numColumns),
Random = new ThreadSafeRandom(42),
StimulusThreshold = 10,
double max = 50;
```

Table: 5

Case 6. Changed Syn Perm Below Stimulus Inc, Syn Perm Trim Threshold (0.1 & 0.08), inputBits, numColumns, double max.

This experiment is performed by changing value of Syn Perm Below Stimulus Inc, Syn Perm Trim Threshold, inputBits, numColumns, double max and keeping all other values constant, table 6.

```
double minOctOverlapCycles = 1.0;
double maxBoost = 5.0;
int countval = 0;
int inputBits = 100;
int numColumns = 1042;
CellsPerColumn = 10,
MaxBoost = maxBoost,
count = countval,
DutyCyclePeriod = 10000,
GlobalInhibition = true,
NumActiveColumnsPerInhArea = 0.02 *  numColumns,
PotentialRadius = (int)(0.15 * inputBits),
LocalAreaDensity = -1,
MaxSynapsesPerSegment = (int)(0.01* numColumns),
Random = new ThreadSafeRandom(42),
StimulusThreshold = 10,
double max = 50.
SynPermBelowStimulusInc = 0.1,
SynPermTrimThreshold = 0.08;
```

Table: 6

Case 7. Changed Syn Perm Below Stimulus Inc and Syn Perm Trim Threshold (0.5 & 0.09)

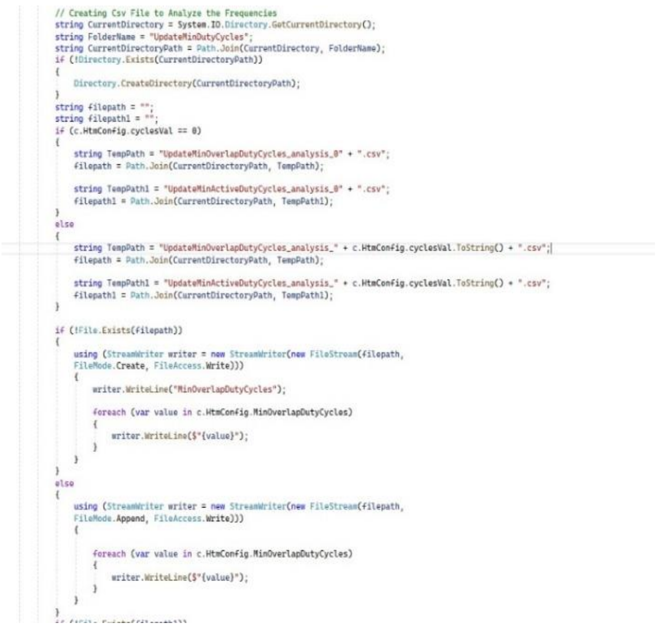
This experiment is performed by changing value of Syn Perm Below Stimulus Inc and Syn Perm Trim Threshold and keeping all other values constant, table 7.

```
double minOctOverlapCycles = 1.0;
double maxBoost = 5.0;
int countval = 0;
int inputBits = 200;
int numColumns = 2048;
CellsPerColumn = 10,
MaxBoost = maxBoost,
count = countval,
DutyCyclePeriod = 10000,
GlobalInhibition = true,
NumActiveColumnsPerInhArea = 0.02 *numColumns,
PotentialRadius = (int)(0.15 * inputBits),
LocalAreaDensity = -1,
MaxSynapsesPerSegment = (int)(0.01* numColumns),
Random = new ThreadSafeRandom(42),
StimulusThreshold = 10,
double max = 100;
SynPermBelowStimulusInc = 0.5,
SynPermTrimThreshold = 0.09;
```

Table: 7

V. CODE

We Updated the Spatial Pattern and Spatial Pooler code to generate the CSV files of all the 4 methods used for boosting in HTMconfig. The code generated to get CSV files are shown in the figure below and also in the link: [4]



```
// Creating Csv File to Analyze the Frequencies
string CurrentDirectory = System.IO.Directory.GetCurrentDirectory();
string FolderName = "UpdateMinDutyCycles";
string CurrentDirectoryPath = Path.Join(CurrentDirectory, FolderName);
if (!Directory.Exists(CurrentDirectoryPath))
{
    Directory.CreateDirectory(CurrentDirectoryPath);
}
string filepath = "";
string filepath1 = "";
if (c.HtmConfig.cyclesVal == 0)
{
    string TempPath = "UpdateMinOverlapDutyCycles_analysis_0" + ".csv";
    filepath = Path.Join(CurrentDirectoryPath, TempPath);

    string TempPath1 = "UpdateMinActiveDutyCycles_analysis_0" + ".csv";
    filepath1 = Path.Join(CurrentDirectoryPath, TempPath1);
}
else
{
    string TempPath = "UpdateMinOverlapDutyCycles_analysis_" + c.HtmConfig.cyclesVal.ToString() + ".csv";
    filepath = Path.Join(CurrentDirectoryPath, TempPath);

    string TempPath1 = "UpdateMinActiveDutyCycles_analysis_" + c.HtmConfig.cyclesVal.ToString() + ".csv";
    filepath1 = Path.Join(CurrentDirectoryPath, TempPath1);
}

if (!File.Exists(filepath))
{
    using (StreamWriter writer = new StreamWriter(new FileStream(filepath,
    FileMode.Create, FileAccess.Write)))
    {
        writer.WriteLine("MinOverlapDutyCycles");

        foreach (var value in c.HtmConfig.MinOverlapDutyCycles)
        {
            writer.WriteLine($"{value}");
        }
    }
}
else
{
    using (StreamWriter writer = new StreamWriter(new FileStream(filepath,
    FileMode.Append, FileAccess.Write)))
    {

        foreach (var value in c.HtmConfig.MinOverlapDutyCycles)
        {
            writer.WriteLine($"{value}");
        }
    }
}
```

Figure 4

VI. UNIT TESTS

The unit tests for 4 methods which we worked on can be found under the solution "NeoCortexapi" in "UnitTestsProject" The Unit Tests which we have added are highlighted in the figure 4.

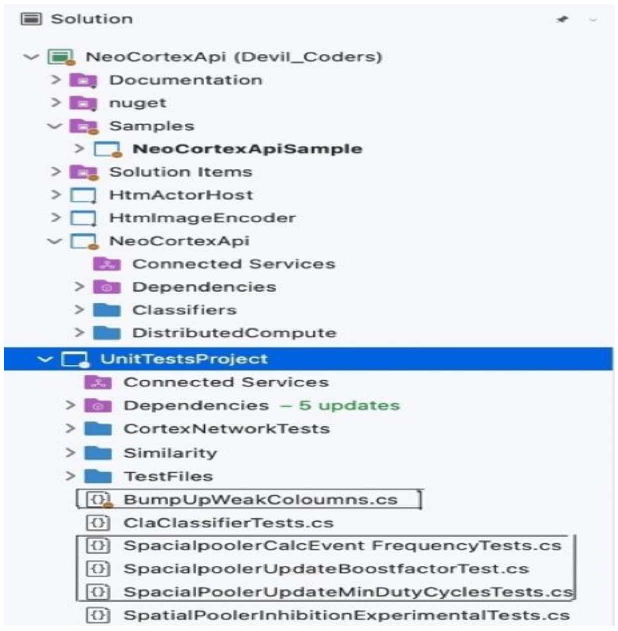


Figure 5

A. Bump Up Weak Columns

Condition: If OverlapDutyCycles MinOverlapDutyCycles the column is not considered as a weak column and hence the permanence values of synapses of columns doesn't increase.

Case 1: We tested if the permanence values are being updated as expected in Bumpupweakcoloumns method with SynPerm Below StimulusInc = 0.01 and SynPermTrimThreshold = 0.05.

Case 2: This test verifies that the permanence values dont get updated when the OverlapDutyCycles are greater than MinOverlapDutyCycles (as it is not considered as a weak column with the condition mentioned here and hence the permanence values don't get updated).

B. Update Boost Factor

Case 1: We tested if Boost Factors are updated as per the formula defined in UpdateBoostFactors method with maxboost = 10

Case 2: Also, tested if Boost Factors are updated as per the formula defined in UpdateBoostFactors method with maxboost = 1

Case 3: This test verifies that the Boost Factors are not updated when minActiveDutyCycles are 0 (A boost factor of 1.0 is used if the duty cycle is >= minOverlapDutycycle)

C. Calc Event Frequency

Calculates the normalised counter value of the frequency of an event. An event can be overlap or the activation of the column. It updates a duty cycle estimate with a new value and it is a helper function which is used to update several duty cycle variables in the Column class and returns the updated duty cycle.

*Case 1***:** Tests if the duty cycles are updated as per the formula defined in CalcEventFrequency with period=1000.

Case 2: Tests if duty cycles are updated as per the formula defined in CalcEventFrequency method with period=1.

D. Update Min Duty Cycles

Wraparound: Determines if inputs at the beginning and end of an input dimension should be considered neighbours when mapping columns to inputs.

Case1: (LOCAL) We tested if Min Duty cycles are updated as per formula defined in UpdateMinDutyCycleLocal without wrap around.

Case2: (LOCAL) We tested if Min Duty cycles are updated as per formula defined in UpdateMinDutyCycleLocal with wraparound.

Case3: (GLOBAL) Tested duty cycles are updated as per the formula defined in CalcEventFrequency method with period 1.

VII. RESULTS

Experiment Repetitions:

It is the number of the times the program is running. It is running for multiple times due to changing of boosting parameters i.e., to obtain the stability. For each case the code is run for multiple times.

Starting with SDR with randomly dispersed connections from each column to the input space, the process begins. Normally, Spatial Pooler will only have a few active columns that represent distinct inputs, or their active duty-cycles will be near to one. During the whole learning process, other inactive columns will never be active. This means that the output SDR can only explain a limited amount of information about the input set. More columns will be able to participate in expressing the input space thanks to the boosting technique. The boosting approach of Spatial Pooler allows all columns to be used consistently across all patterns. Even though the columns have previously learnt patterns, the boosting process is still active, causing the Spatial Pooler to forget the input. To address this issue, the Spatial Pooler now includes a new homeostatic plasticity controller that turn off boosting once the learning has reached a stable state. The output SDRs of the Spatial Pooler do not change over time, according to research.

For a column in a Spatial Pooler to exist it should be a winning column, the overlap score should be above some threshold value while non-winning columns are inhibited from learning. Only the winner columns can update their permanence values. Boosting helps to change the overlap score before the inhibition occurs giving fewer active columns a better chance to express themselves and diminishing columns that seem overactive.

Case 1: By changing max boost and duty cycle period the stability is attaining at cycle=0153, i=82, cols=41, s=100 and the total time taken to complete the program is 46:00 minutes Figure 5 and 6.

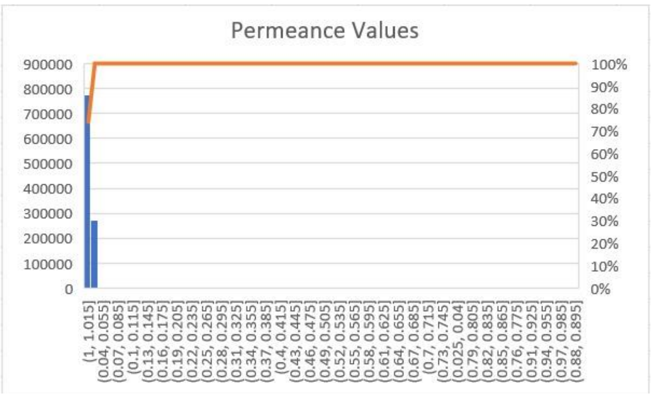


Figure 5

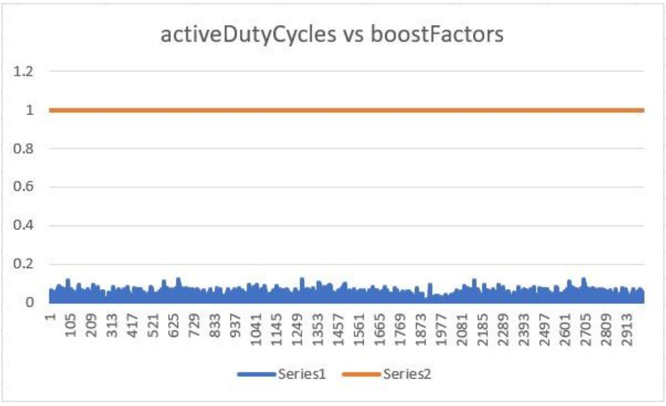


Figure 6

Case 2: By changing max boost and duty cycle period the stability is attaining at same cycle=0232, i=0, cols=40, s=100 and the total time taken to complete the program is 46:00 minutes Figure 7 and 8.

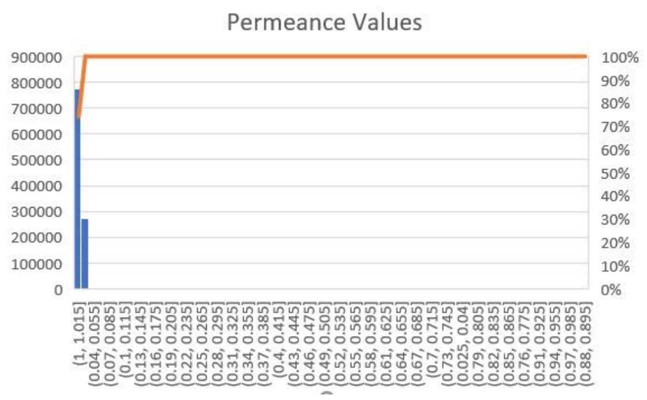


Figure 7

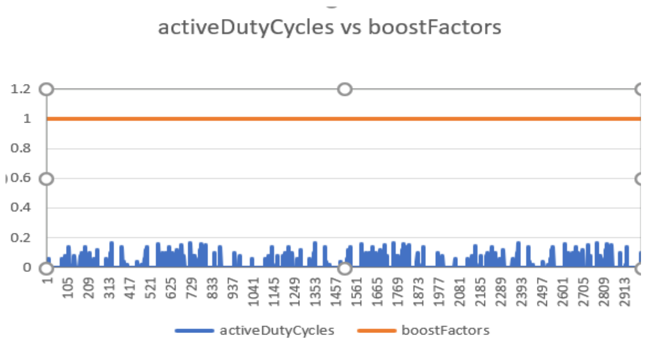


Figure 8

Case 3: By changing max boost and duty cycle period the stability is attaining at same cycle=0232, i=0, cols=40, s=100 and the total time taken to complete the program is 42:00 minutes, Figure 9 and 10.

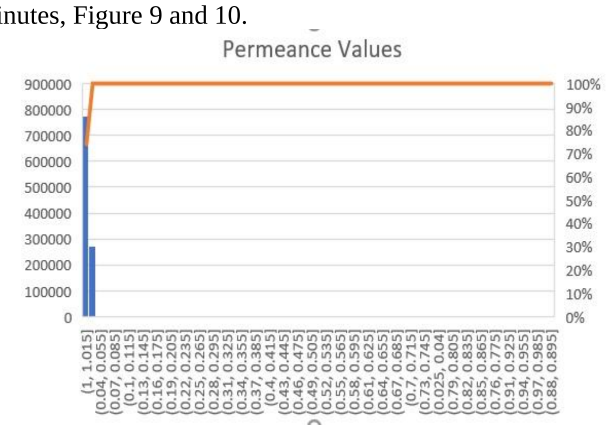


Figure 9

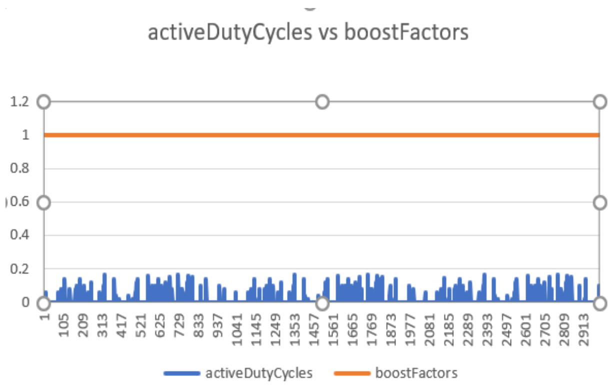


Figure 10

Case 4: By changing these values, the stability is attaining at cycle=0096, i=0, cols=41, s=100 and the total time taken to complete the program is 28:00 minutes Figure 11 and 12.

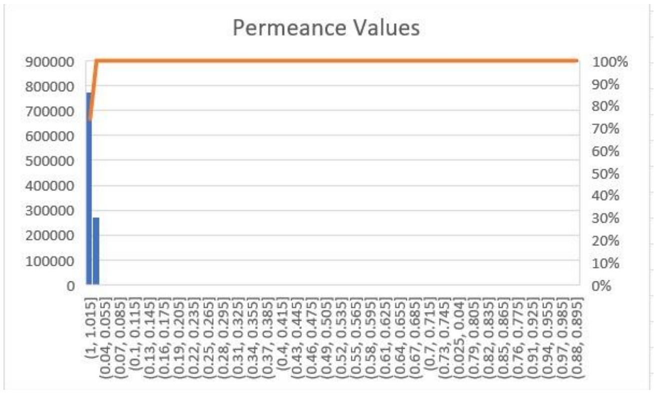


Figure 11

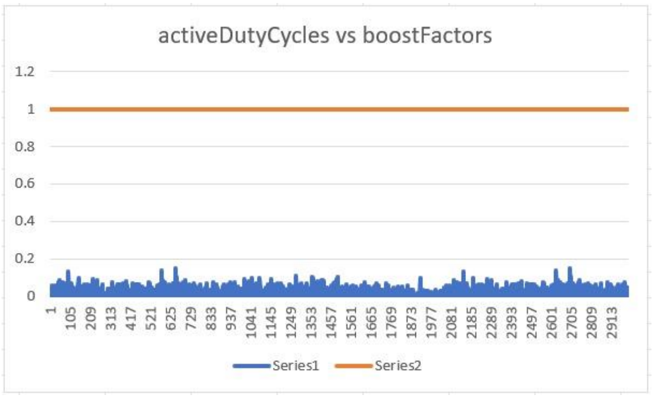


Figure 12

Case 5: By changing these values the stability is attaining at cycle= 331, i=41, cols=20, s=100 and the total time taken to complete the program is 2 minutes, Figure 13 and 14.

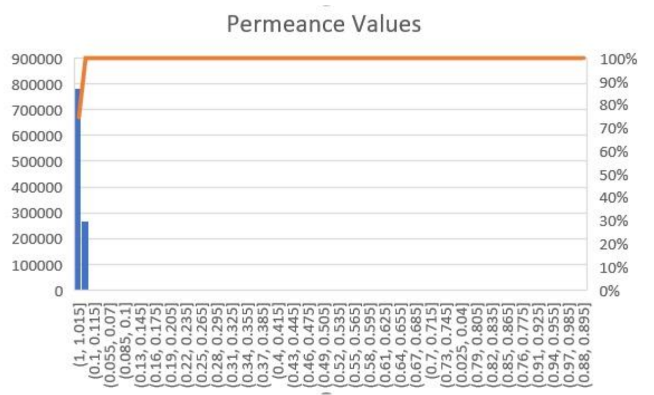


Figure 13

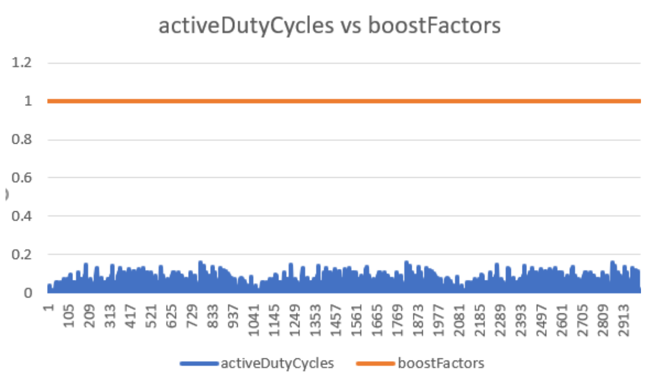


Figure 14

Case 6: By changing Syn Perm Below Stimulus Inc, Syn Perm Trim Threshold, inputBits, numColumns, double max, the stability is attaining at cycle=100 and the total time taken to complete the program is 14 minutes, Figure 15 and 16.

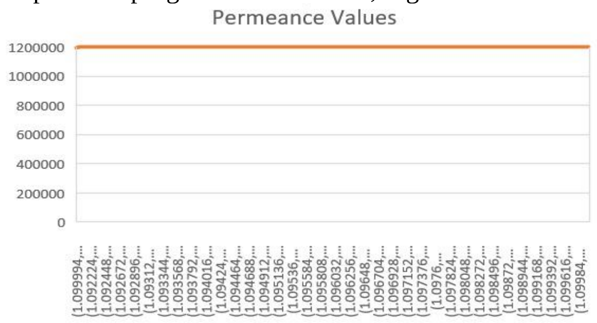


Figure 15

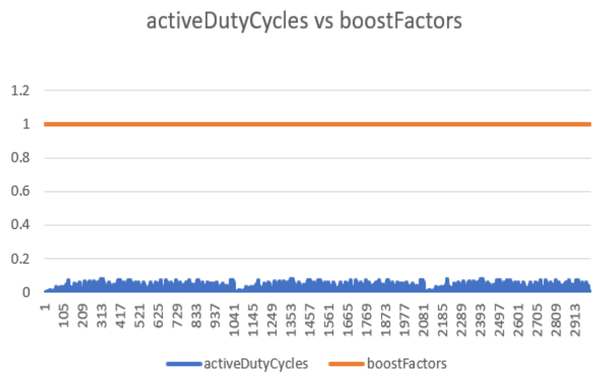


Figure 16

Case 7: By changing Syn Perm Below Stimulus Inc, Syn Perm Trim Threshold, the stability is attaining at cycle=95, i=19, cols=41, s=100 and the total time taken to complete the program is 46:00 minutes, Figure 17 and 18.

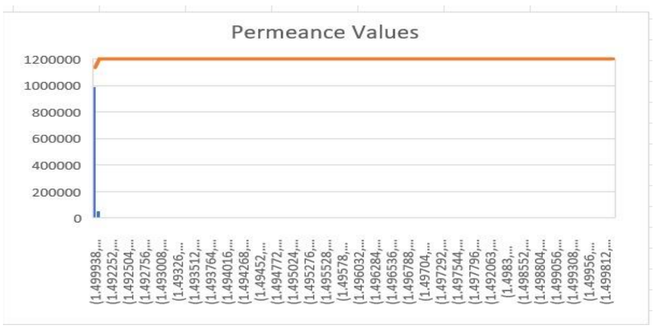


Figure 17

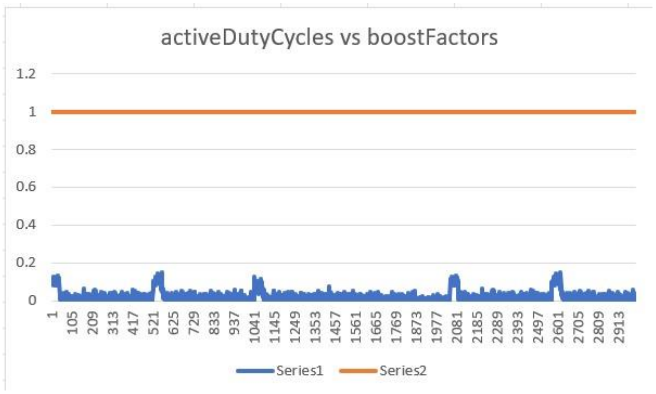


Figure 18

Boost from 10 to 1 and Duty Cycles from 10000 to 40000000 We concluded that there is inverse relation of Max boost and Duty Cycle. The higher value of boost the lower the stability is but in fact if we reduce boost but increasing Duty cycle while the remaining parameters are same, so the stability also increase not 100% but approx. 90 to 95%.We added three graphs at boost value 10, 5, and 1.We also ran program at duty cycles 8000000 while boost set to 5 so stability reached to 100% not always. Moreover, there is one more parameter which play a vital role for making it stable is "BUMPUP". It disables all the weak columns (Columns in which average or all the cells are not active). As this unit test takes more than 102 minutes to execute for all input series from 0 to 10 but, if we reduced no of columns from 2048 to 1000 or even 500 but keeping no of cell per column same so, it reduces the time of execution however, the stability still at 100%.

VIII. CONCLUSION

Spatial Pooler will only have a few active columns that represent distinct inputs, or their active duty-cycles will be near to one. During the whole learning process, other inactive columns will never be active. This means that the output SDR can only explain a limited amount of information about the input set. More columns will be able to participate in expressing the input space thanks to the boosting technique. The boosting approach of Spatial Pooler allows all columns to be used consistently across all patterns.

Boosting helps to change the overlap score before the inhibition occurs giving fewer active columns a better chance to express themselves and diminishing columns that seem overactive.

It may be concluded that the boosting process has an impact on the SP stability. The SP can reach a stable state, but SDRs with a drastically different number of active mini columns will result. If boosting is turned on, the SP will activate mini columns equitably, but the learning will be unstable.

IX. References

[1] J. &. A. S. (. awkins, "Why neurons have thousands of synapses, a theory of sequence memory in neocortex. Frontiers in neural circuits, 10.," [Online].

[2] R. Singh, "Medium.com," 31 May 2018 . [Online]. Available: https://medium.com/@rockingrichie1994/understanding-hierarchal-temporal-memory-f6a1be38e07e.

[3] S. A. a. J. H. Yuwei Cui, "Numenta," 29 November 2017. [Online]. Available: https://numenta.com/neuroscience-research/research-publications/papers/htm-spatial-pooler-neocortical-algorithm-for-online-sparse-distributed-coding/.

[4] D. Dobric, ""Github," [Online]," [Online]. Available: https://github.com/ddobric/neocortexapi-classification..

[5] A. P. 2. B. G. 1. T. W. 1. Damir Dobric 1, "Improved HTM Spatial Pooler with Homeostatic Plasticity control," [Online]. Available: https://www.academia.edu/45090289/Improved_HTM_Spatial_Pooler_with_Homeostatic_Plasticity_control.

[6] M. A. &. R. Khanna, "Deep Learning," [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4302-5990-9_9.