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

Analyse and Describe 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 Analysing and Describing of Boosting Algorithm which 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.

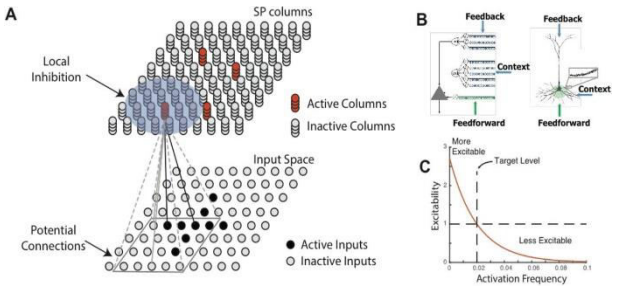

Figure 1 The HTM spatial pooler converts inputs (bottom) to SDRs (top). [1]

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 spatial pooler. The rest of the paper is structured as follows; Section 2 covers HTM Overview. Section 3 Literature Survey, Section 4 explains each of the methods used in HTMconfig. Section 5 contains the cases which were implemented, section 6 and 7 has the code and the unit tests of each method. Experimental results and conclusion are presented in section 8 and 9 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.

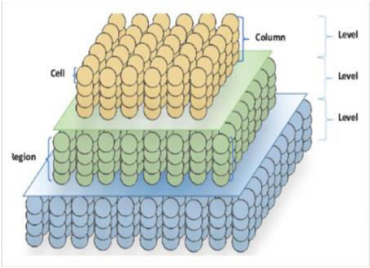

Figure 2: A representation of three layers of HTM arranged hierarchically. Each region contains columns of cells arranged vertically.

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 in charge 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]

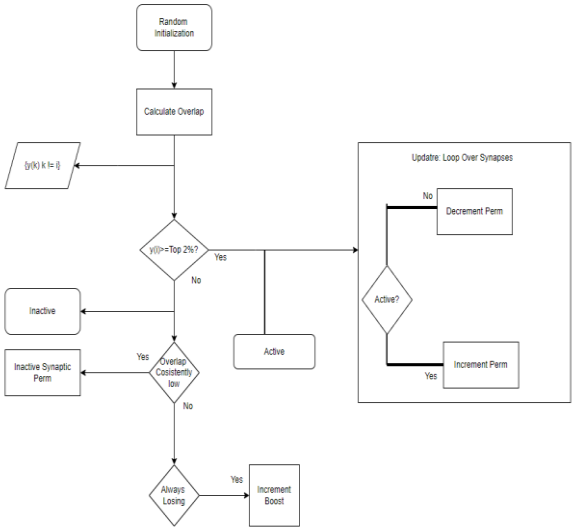

Figure 3: Spatial pooler flowchart [3]

D. Boosting:

Boosting is a technique used to maintain a higher level of homeostasis in column activation. Homeostasis is a property of a system to maintain some variable nearly constant across its constituent elements. In the context of the HTM, it is a measure of a region's ability to maintain similar activation duty cycles for all of its columns. An HTM with a high level of homeostasis would inhibit overactive columns, and boost less active columns to maintain a similar duty cycle across all columns. An HTM with poor homeostasis would have some columns which are exceptionally active relative to others. When a column is active across broadly differing input patterns, its activation becomes meaningless for classification, and reduces the likelihood other columns can express themselves and identify meaningful features in the input space. Boosting seeks to solve this problem by inhibiting overactive columns, boosting the less active columns, or a combination of both. [4]

The boosting algorithm computes a boosting factor for each column based on the duty cycle of that column, as well as

the duty cycles of the columns within its inhibition radius. The overlap score of each column is then multiplied by this boosting factor to obtain its boosted overlap score. If the boosted overlap score of a column is higher than that of the other columns in its inhibition radius, that column is chosen as a winner for that iteration.

For more detailed information about Boosting please see ("Github-BoostingAlgorithm," 2022 [5]).

E. Learning:

Learning happens only in those columns of the Spatial Pooler which are active. All the connections that are falling in the input space, the permeance value will increase for them i,e the synaptic connection between them will strengthen while any connections that fall outside of the input space for those the permeance value will be decremented . Learning Spatial Pooler learns better in comparison to the Random Spatial Pooler. This step has been basically derived from the concept 'Hebbian Learning: Neurons that fire together wire together '. For instance, in the diagram given below for any given column in a Spatial Pooler those cell that are connected to the active cells in the input space (i.e. which are in green) their permeance will be incremented else those connected outside the input space their permeance will be decremented(the one's in grey). No inactive column will learn anything. [2]

F. Temporal Pooling:

The goal of temporal pooling is to identify and learn patterns in sequences of inputs. With the temporal pooler, it extends not only the ability of the HTM beyond just identifying features and classes of inputs, but also to predict future inputs from past patterns, based on the current temporal context of the current input. [4]

III. Literature Survey

Improved HTM Spatial Pooler with Homeostatic Plasticity control

Hierarchical Temporal Memory (HTM) -
Spatial Pooler (SP) is a Learning Algorithm for learning of spatialpatterns inspired by the neo-cortex. It is designed to learn the pattern in a few iteration steps and to generate the Sparse Distributed Representation (SDR) of the input. It encodes spatially similar inputs into the same or similar SDRs memorized as a population of active neurons organized in groups called micro-columns. Findings in the research show that produced SDRs can be forgotten during the training progress, which causes the SP to learn the same pattern again and converts into the new SDR. This work shows that instable learning behaviour of the SP is caused by the internal boosting algorithm inspired by the homeostatic plasticity mechanism. Previous findings in neurosciences show that this mechanism is only active during the development of new-born mammals and later deactivated or shifted from cortical layer L4, where the SP is supposed to be active. The same mechanism was used in this work. The SP algorithm was extended with the new homeostatic plasticity component that controls the boosting and deactivates it after entering the stable state. Results show that learned SDRs remain stable during the lifetime of the Spatial Pooler [6]

For detailed explanation on this topic please see("Github-BoostingAlgorithm," 2022 [5]).

IV. 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.

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

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.

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

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.

V. 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 shown in Table1.

For more detailed information about the meaning of all parameters please see ("Github-BoostingAlgorithm," 2022 [5])

double minOctOverlapCycles	1.0
double maxBoost	10
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 Parameters of case1

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 parameters of case2

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
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 parameters of case3

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 Weak Columns parameter as true, table 4.

double minOctOverlapCycles	1.0
double maxBoost	5
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
IsBumpUpWeakColumnsDisabled	true

Table: 4 parameters of case4

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
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 parameters of case5

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
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
SynPermBelowStimulusInc	0.1
SynPermTrimThreshold	0.08

Table: 6 parameters of case6

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
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 parameters of case7

VI. CODE

We Updated Spatial Pooler code to generate the CSV files. These csv files are required for the analysis of Boosting algorithm. The code generated to create CSV files are implemented under the following methods in spatial pooler

1. BumpUpWeakColumns
2. UpdateBoostFactors
3. UpdateDutyCycles
4. UpdateMinDutyCycles

you can refer [7]

we have also updated the Spatial Pattern Learning to generate results.csv, which contains detailed description of the experiment executed, you can find the code changes in the following repo ("Github-Spacial-pooler," 2022 [8]).

```
String CurrentDirectory =
System.IO.Directory.GetCurrentDirectory();
string FolderName = "UpdateDutyCycles";
string CurrentDirectoryPath = Path.Join(CurrentDirectory,
FolderName);
if (!Directory.Exists(CurrentDirectoryPath))
{
Directory.CreateDirectory(CurrentDirectoryPath);
}
string filepath = "";
if (c.HtmConfig.cyclesVal == 0)
{
string TempPath = "UpdateDutyCycles_analysis_0" + ".csv";
filepath = Path.Join(CurrentDirectoryPath, TempPath);
}
else
{
string TempPath = "UpdateDutyCycles_analysis_" +
c.HtmConfig.cyclesVal.ToString() + ".csv";
filepath = Path.Join(CurrentDirectoryPath, TempPath);
}
if (!File.Exists(filepath))
{
using (StreamWriter writer = new StreamWriter(new
FileStream(filepath, FileMode.Create, FileAccess.Write)))
{
writer.WriteLine("Input,OverlapDutyCycles,ActiveDutyCycles");
var OandACycles =
c.HtmConfig.OverlapDutyCycles.Zip(c.HtmConfig.ActiveDutyCy
cles, (n, w) => new { OverlapDutyCycles = n, ActiveDutyCycles =
w });
foreach (var nw in OandACycles)
{
writer.WriteLine($"{c.HtmConfig.count},{nw.OverlapDutyCycles
},{nw.ActiveDutyCycles}");
}
}
}
else
{
using (StreamWriter writer = new StreamWriter(new
FileStream(filepath, FileMode.Append, FileAccess.Write)))
{
var OandACycles =
c.HtmConfig.OverlapDutyCycles.Zip(c.HtmConfig.ActiveDutyCy
cles, (n, w) => new { OverlapDutyCycles = n, ActiveDutyCycles =
w });
foreach (var nw in OandACycles)
{
writer.WriteLine($"{c.HtmConfig.count},{nw.OverlapDutyCycles
},{nw.ActiveDutyCycles}");}}}}
```

Listing 1: Code to generate a CSV file.

VII. UNIT TESTS

The unit tests for 4 methods which we worked on can be found under the solution "NeoCortexapi" in "UnitTestsProject" [9].

A. Bump Up Weak Columns

This method increases the permanence values of synapses of columns whose overlap level is too low. Such columns are identified by having an overlap duty cycle (activation frequency) that drops too much below those of their peers. The permanence values for such columns are increased. [10]

Description:

This Unit-Test ensures that the Testing permanence values are updated correctly in the BumpUpWeakColumns method with SynPermBelowStimulusInc = 0.1 and SynPermTrim Threshold = 0.05.

The permanence values of weak columns must be updated, and weak columns are identified with the If OverlapDutyCycles > MinOverlapDutyCycles the column is considered as a weak column and hence the permanence values of synapses of columns increase.

First, by using the parameters setInputDimensions and setColumnDimensions we set the test synapses of columns which is the array of permanence values for a column. Then the Expected permanence values are calculated manually using SynPermTrimThreshold and SynPerBelowStimulus Inc when OverlapDutyCycles < MinOverlapDutyCycles. Later, the expected values are verified with the calculated values generated by the unit-test. [10]

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 doesnt get updated when the OverlapDutyCycles are greater than MinOverlapDutyCycles (as it is not considered as a weak column as per the condition).

B. Update Boost Factor

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 [10].

Description:

This unit-test helps in verifying the absolute values of manually calculated Boost Factors from the below mentioned formulae and Boost Factor values from UpdateBoostFactors method.

Formula : boost =(1-maxBoost)/minDuty*activeDutyCycle + maxBoost

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.

This Unit-test helps in Verifying if manually calculated duty cycle values and dutycycle values from CalcEventFrequency method are equal or not with the help of below mentioned formulae.

Formulae: dutyCycle = (period - 1)*dutyCycle + newValue / period

*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

It updates the minimum duty cycles for SP that 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.

Description :

Whenever the UpdateMinDutyCycles method is called in the SP its either Local or Global Inhibition. Through inhibition radius we get the size of the local neighborhood, Then the Maximum value in the neighborhood of maxActiveDuty and maxOverlapDuty is multiplied with MinPctActiveDutyCycles and MinPctOverlapDutyCycles Respectively to be a percent of the maximum in the region. [10]

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.

VIII. RESULTS

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.

Case 1: In this case the boost value is kept 1.0 and duty cycle period value 50000. By changing max boost and duty cycle period the stability is attaining at cycle=0153, i=82, cols=41, s=100 and we observed that initially the SDR is not representing complete values and the program is slow. When the boosting parameter is activated the SDR is representing complete values and the total time to complete the program is 46:00 minutes. The below graph shows the output graphs of Permeance values in Figure 5 and active-Duty Cycles vs boost Factors in Figure 6.

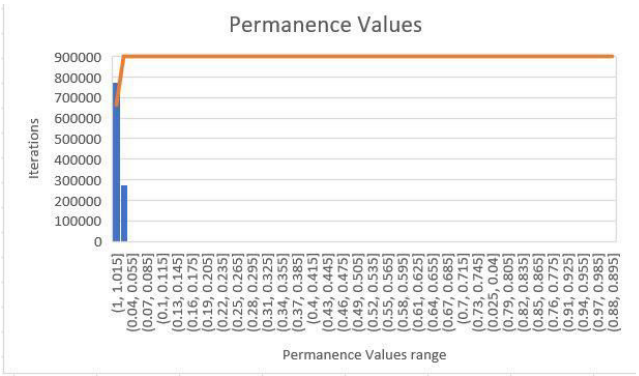

Figure 5 Permeance Values

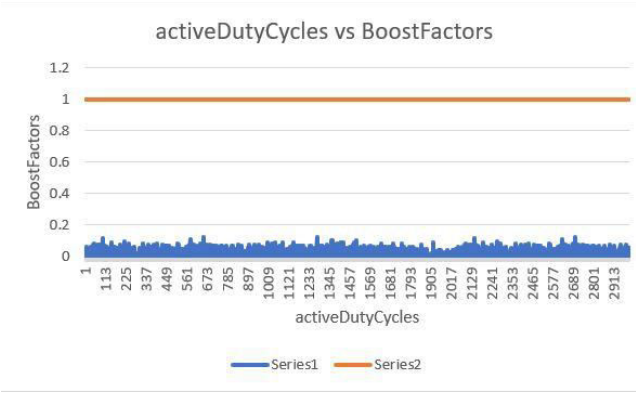

Figure 6 active-Duty Cycles vs boost Factors

Case 2: In this case we increased the boost value to 5.0 and duty cycle period value 100000. By changing max boost and duty cycle period the stability is attaining at cycle=0232, i=0, cols=40, s=100 and we again observed that initially the SDR is not representing complete values and the program is slow. When the boosting parameter is activated the SDR is representing complete values and the total time to complete the program is same when compared with case1. In this case we increased both values and remaining parameters are same, so the stability also increased to cycle=0232 not 100% but approximately 90 to 95%. The below graph shows the output graphs of Permeance values in Figure 7 and active-Duty Cycles vs boost Factors in Figure 8.

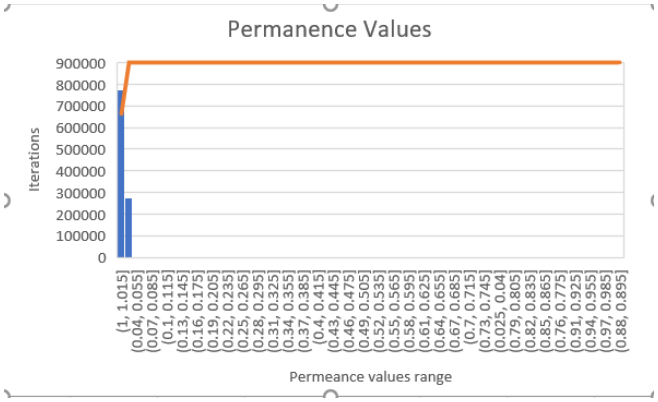

Figure 7 Permeance Values

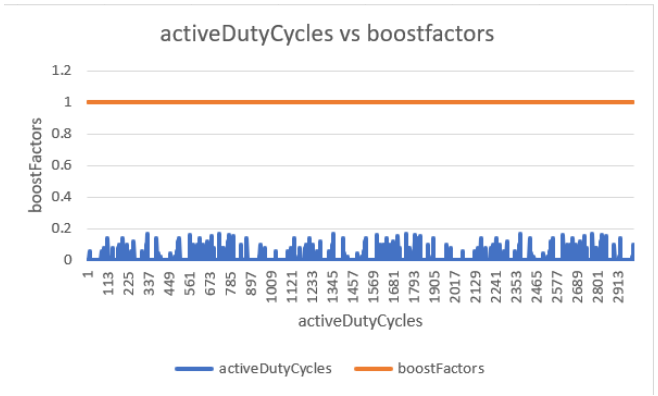

Figure 8 active-Duty Cycles vs boost Factors

Case 3: In this case we increased the boost value to 10.0 and duty cycle period value 150000. By changing max boost and duty cycle period the stability is attaining at same cycle=0232, i=0, cols=40, s=100 and we again observed that initially the SDR is not representing complete values and this time the program is somewhat fast compared to case1 and case2. When the boosting parameter is activated the SDR is representing complete values and the total time to complete the program is reduced when compared with case1 and case2 (42 minutes). In this case we increased both values and remaining parameters are same, so the stability also increased to cycle=0232 not 100% but approximately 90 to 95%. We observed that with increasing the max boost and duty cycle period the stability is also increases. The below graph shows the output graphs of Permeance values in Figure 9 and active-Duty Cycles vs boost Factors in Figure 10.

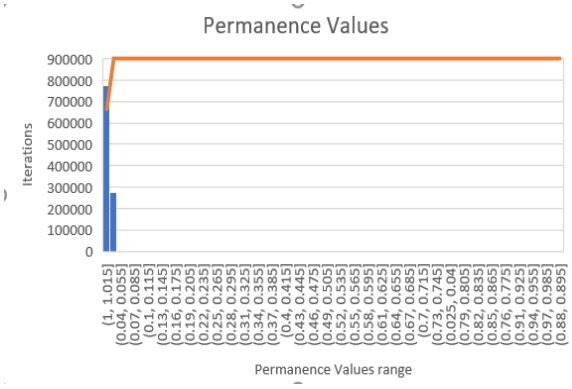

Figure 9 Permeance Values

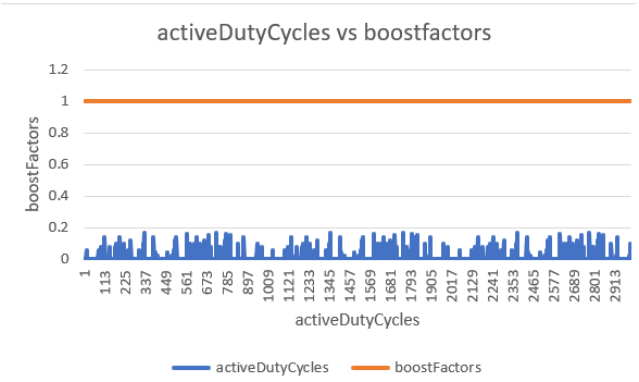

Figure 10 active-Duty Cycles vs boost Factors

Case 4: In this case we decreased the boost value to 5.0 and duty cycle period value 100000 and changed one more parameter which plays a vital role for making it stable is "BUMP UP WEAK COLUMNS" to true. By changing max boost, duty cycle period and Bump Up Weak Colum to true the stability is attaining at cycle=096, i=0, cols=41, s=100 and this time we observed that it disables all the weak columns (Columns in which average or all the cells are not active). This test takes less than 30 minutes to execute all input series from 0 to 10. Initially the SDR is not representing complete values and this time the program is very fast compared to case1, case2 and case3. When the boosting parameter is activated the SDR is representing complete values and the total time to complete the program is reduced when compared with case1, case2 and case3. The below graph shows the output graphs of Permeance values in Figure 11 and active-Duty Cycles vs boost Factors in Figure 12.

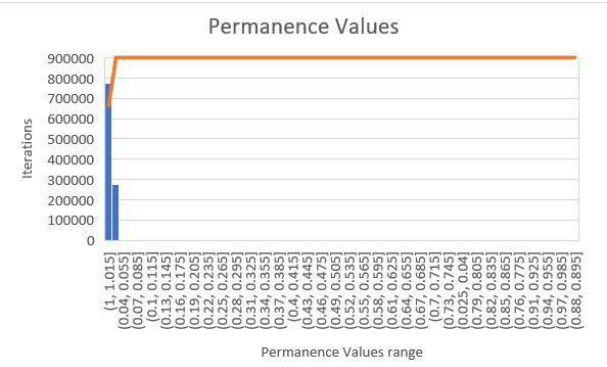

Figure 11 Permeance Values

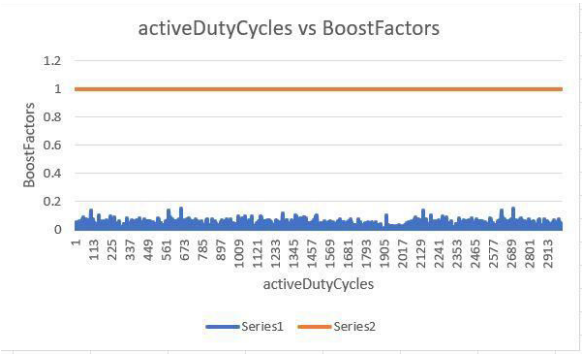

Figure 12 active-Duty Cycles vs boost Factors

Case 5: In this case we kept the values same as case4 and changed the parameters input bits, num columns and double max to 100, 1024 and 50. By changing these values the system is attaining stable state at cycle=0331, i=41, cols=20, s=100. We observed that when we reduce these values the stability increases, and it reduces the execution time. The total time to execute this case is 2 minutes because of reducing num of columns and double max. The below graph shows the output graphs of Permeance values in Figure 13 and active-Duty Cycles vs boost Factors in Figure 14.

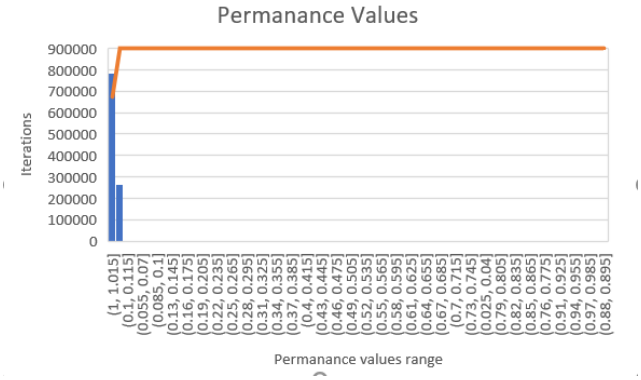

Figure 13 Permeance Values

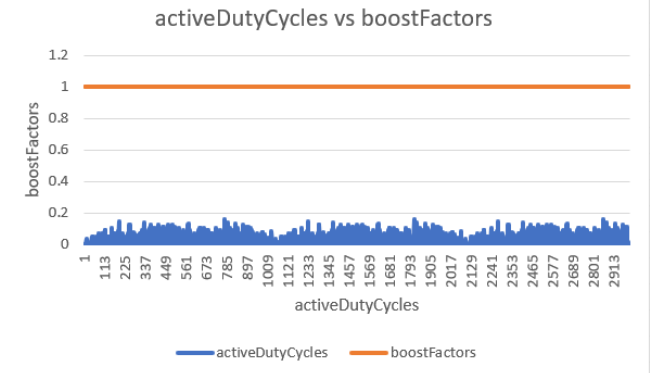

Figure 14 active-Duty Cycles vs boost Factors

Case 6: In this case we changed two more parameters which comes under Bump Up Weak Columns method are Syn Perm Below Stimulus Inc and Sync perm Trim Threshold along with max boost, duty cycle period, input bits, num of columns and double max. By changing these values, we

observed that the system is attaining stability at cycle=0100, i=40, cols=20, s=100. We observed that the Syn Perm Below Stimulus Inc will increment the value to the current permeance value of the synapse and Sync perm Trim Threshold value is used by SP and when some permanence is under this value, it is set on zero. The total time to complete the program is 14 minutes. The below graph shows the output graphs of Permeance values in Figure 15 and active-Duty Cycles vs boost Factors in Figure 16.

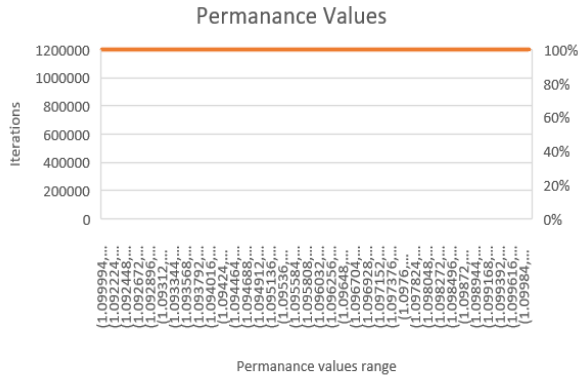

Figure 15 Permeance Values

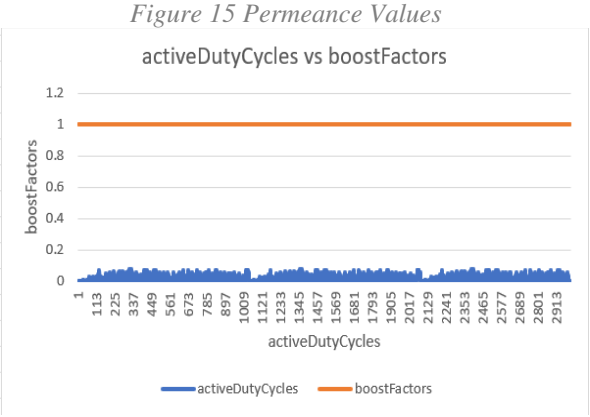

Figure 16 active-Duty Cycles vs boost Factors

Case 7: In this case we changed two more parameters which comes under Bump Up Weak Columns method are Syn Perm Below Stimulus Inc and Sync perm Trim Threshold by keeping all other values constant. By changing these values, we observed that the system is attaining stability at cycle=095, i=19, cols=41, s=100. We observed that the Syn Perm Below Stimulus Inc will increment the value to the current permeance value of the synapse and Sync perm Trim Threshold value is used by SP and when some permanence is under this value, it is set on zero. This time the total time to complete the program is 46 minutes because of input bits, num of columns and double max. The below graph shows the output graphs of Permeance values in Figure 17 and active-Duty Cycles vs boost Factors in Figure 18.

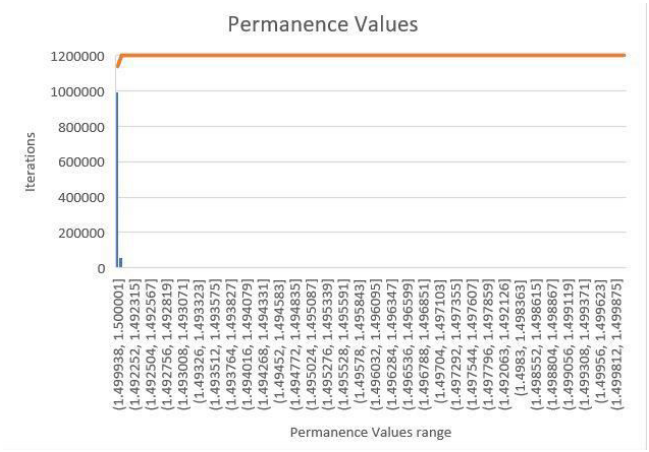

Figure 17 Permeance Values

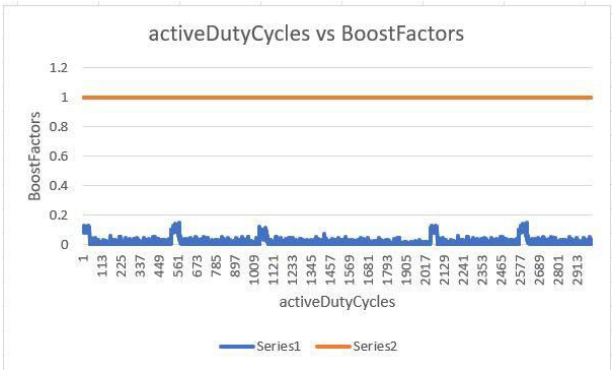

Figure 18 active-Duty Cycles vs boost Factors

These graphical representations are the representation of data which is fetched from Spatial Pooler experiment. We ran this program many times by setting up the different values of maxBoost, Duty Cycles, Syn Perm Below Stimulus Inc, Syn Perm Trim Threshold, num of columns, input bits and double max.

By setting up the different values of maxBoost from 1 to 10 and Duty Cycles from 10000 to 150000 we concluded that there is an 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 approximately 90 to 95%. 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 1024, it reduces the time of execution however, the stability still at 100%.

IX. CONCLUSION

Boosting is a technique used to maintain a higher level of homeostasis in column activation. Homeostasis is a property of a system to maintain some variable nearly constant across its constituent elements. In the context of the HTM, it is a measure of a region's ability to maintain similar activation duty cycles for all of its columns. An HTM with a high level

of homeostasis would inhibit overactive columns, and boost less active columns to maintain a similar duty cycle across all columns. An HTM with poor homeostasis would have some columns which are exceptionally active relative to others. When a column is active across broadly differing input patterns, its activation becomes meaningless for classification, and reduces the likelihood other columns can express themselves and identify meaningful features in the input space. Boosting seeks to solve this problem by inhibiting overactive columns, boosting the less active columns, or a combination of both. [4]

Based on the above experiments, it can be said that various parameters of Spatial Pooler and Boosting have effects on stability. We have observed how the parameters like max Boost, Duty Cycle period, input bits (N), num of columns, double max in the Spatial Pooler effect the stability by changing the values of these parameters and comparing them with the minimum number of cycles required to get the stability. We have tested these parameters and documented the results above.

Through these experiments we understood various methods contributing to boosting in the HTM config and wrote unit-tests on them. Also, understood how boosting will help the output SDR to utilize more of input spaces and provide more information.

X. REFERENCES

[1] 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/.

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

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

[4] P. J. Mitchell, A NOVEL FPGA IMPLEMENTATION OF HIERARCHICAL TEMPORAL MEMORY SPATIAL POOLER, 2018.

[5] Team_Devil_coders, "Github(BoostingAlgorithm)," 2022. [Online]. Available: https://github.com/sandhyabagadi/neocortexapi/blob/Devil_Coders/source/MySEProject/Documentation/BoostingAlgorithm.md.

[6] 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.

[7] Team_DevilCoders, "Github," [Online]. Available: https://github.com/sandhyabagadi/neocortexapi/blob/43358e4ee36cb1d8cf44bb39384ca7b2e62ad24d/source/MySEProject/SpatialPooler.cs#L566.

[8] Team_DevilCoders, "Github(Spacial-pooler)," 2022. [Online]. Available: https://github.com/sandhyabagadi/neocortexapi/blob/Devil_Coders/source/MySEProject/SpatialPatternLearning.cs.

[9] TeamDevilCoders, "Github," 2022. [Online]. Available: https://github.com/sandhyabagadi/neocortexapi/tree/Devil_Coders/source/MySEProject/Unit-Tests.

[10] Team_DevilCoders, "Github," 2022. [Online]. Available: https://github.com/sandhyabagadi/neocortexapi/blob/Devil_Coders/source/MySEProject/Documentation/ReadMe.md.

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

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