# Analyze and document boosting algorithm

Information Technology Course

Module Software Engineering

by Damir Dobric / Andreas Pech

Your Name

youremail@stud.fra-uas.de

**Abstract— “Boosting” is a term which refers to a branch of algorithms which changes weak learners to strong learners. Boosting is a group strategy for improving the model forecasts of some random learning algorithm. Boosting is to prepare weak learner consecutively, each attempting to address its predecessor. This work shows the boosting algorithm which boosts the columns in spatial pooler (SP).** **The boosting algorithm tracks the column activity in spatial pooler(SP) and it makes it sure that all columns are uniformly used across all seen patterns** .**This paper shows several experiments like Adapt synapses, Update duty cycle, Bump up weak columns, Update boost factors, inhibition radius and min duty cycle of boosting algorithm. Each experiment is evaluated based on the theoretical and practical framework and then summarized the results in graphical diagrams.**

***Keywords— Boosting Algorithm, Synapses,***

***Boost Factor, Duty Cycle.***

1. **INTRODUCTION**

Hierarchical temporal memory (HTM) is a neuromorphic machine learning algorithm which resembles neocortex functions in the brain.

The HTM architecture consists of two poolers:

1. Spatial pooler (SP): It is a neutrally inspired learning algorithm which creates sparse representations when given a noisy data stream in an online fashion. It tells us how neurons learn feed forward connections and how it constructs an efficient representation of the input. It takes arbitrary binary patterns as input and converts them into sparse distributed representations (SDRs). It does this by using Hebbian Learning rules and homeostatic excitability control.

2. Temporal memory (TM): Temporal memory is in the charge of learning the process. It learns the patterns that are changing over time in these neurons that are on these STDs over time. It operates on motor commands as well as sensory inputs. HTM postulate that every excitatory neuron in the neocortex is learning transitions of patterns.

The Stability of the spatial pooler can be influenced by Boosting mechanism [5]. “An HTM region is organized in columns of cells. The SP operates at the column-level, where a column of a cells function as a single computational unit” [6]. When the boosting algorithm is enabled then more and more columns becomes active. The learning process is increased. Boosting is an ensemble meta-algorithm which is used for primarily reducing bias. Boosting is the family of machine learning algorithms which converts weak learners to stronger ones. The primary goal of this paper is to provide a thorough discussion of the boosting algorithm and demonstrate it by performing thorough unit tests of every method used in boosting algorithm.

1. **CODE**

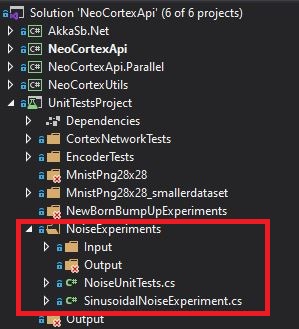
The unit tests used in the project can be found under the solution “NeoCortexapi”in MyProject folder.

Under ‘NeoCortexApi’ in MyProject folder we have implemented our unit tests. To verify this implemented code under different scenarios, several tests have been conducted in the folder:

se-cloud-2020-2021\MyProject\NeoCortexApi\

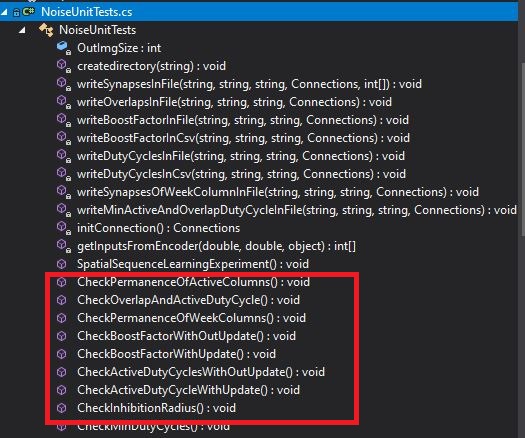
neocortexapi\NeoCortexApi\UnitTestsProject\NoiseExperiments.

To verify this implemented code under different scenarios, several tests have been implemented [1]



*Figure 1 This is the picture description.*

Under NoiseUnitTests, the unit tests performed are highlighted.



The results of the unit tests conducted in below folder:

se-cloud-2020-2021\MyProject\NeoCortexApi\ neocortexapi\NeoCortexApi\UnitTestsProject\bin\Debug\net5.0\boosting.

The results of the unit tests conducted in folder [1].

Initially we set these values for unit test:

|  |  |
| --- | --- |
| POTENTIAL\_RADIUS | 64\*64 |
| POTENTIAL\_PCT | 1 |
| GLOBAL\_INHIBITION | FALSE |
| INHIBITION\_RADIUS | 0.25 \* 64 \* 64 |
| NUM ACTIVE COLUMNS PER INH AREA | 0.1 \* 64 \* 64 |
| DUTY\_CYCLE\_PERIOD | 1000000 |
| MAX\_BOOST | 5 |
| Input Dimensions | 32, 32 |
| Column Dimensions | 64, 64 |

1. **METHODOLOGY AND RESULTS**

There are six methods in boosting algorithm, and I performed unit test for each method individually. After getting the result, I analysed it as per bellow:

**1.Adapt Synapses:**

I made a unit test for checking the permanence value of active column.

**Unit Test (**CheckPermanenceOfActiveColumns()**)**

The neurons in the cortex communicate with each other via chemical and electrical signals. The signals are the basis of learning in the cortex and the memory [2]. In a typical neuron consists of a single axon, cell body or soma, and many dendrites. The dendrites receive incoming signals from the other neurons, and the axon and its terminals transmit the received signal to the other neurons. Some axons are faster than others as they are coated with myelin. It is a fatty substance that insulates the axon and it increases the speed of communication[1]. The signals pass between the neurons at connections called synapses [1].

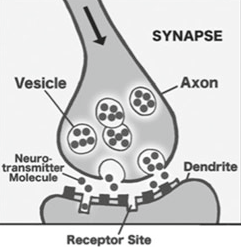


Fig. 1: Signal propagating down an axon to the dendrites of the next cells [1].

The concept of permanence is used in HTM to change the connectedness of synapses. Permanence is scalar value, and its values range from zero to one. This represents the degree of connectedness between the axon and the dendrite. The permanence value of zero signifies that this is not valid for potential synapse, and it cannot be a valid synapse. Typically, there is a threshold permanence value 0.2 which represents that a synapse is just connected but it is possible to get unconnected easily. If the permanence value is high for example 0.9 means that the synapse is connected, and it cannot be easily disconnected [3]. When the synapse’s permanence value is more than the threshold value, that synapse relates to weight one. If this value is below the threshold, that is unconnected with weight zero. In the classical neural networks, there is no individual weighting for synaptic connection. However, HTM has this feature to create and remove these connections. According to Hawkins, HTM achieves a higher data storage capacity by creating and extracting synaptic connections without changing the weights of permanent connections [1].

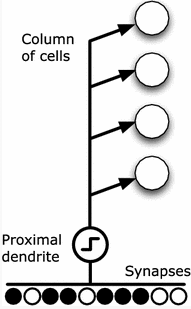


Fig. 2 [1]

The small black circle represents the synapses shared a proximal dendrite in the same column of that cell. The solid circle indicates a valid synapse connection with sufficient permanence value (more than the threshold), and an empty circle indicates a potential synapse connection with insufficient permanence value (less than the threshold value) [1].

From the input vector, each column determines its activation by summing the input bits in the positions with permanence which is larger than the threshold. The overlap score is constituted by the sum of bits.

In the code, the primary method in charge of learning. After the inhibition round, adapt the permanence values of the synapses according to the input vector, and the chosen columns. When the synapses connected to input bits are turned on, the permanence values will increase. The permanence values will decrease when the synapses connected to inputs bits that are turned off. From the unit test, we got two datasheets. One for before updated Synapses and another one are after the updated synapses. Updated synapses means when the input bits are turned on. We observed that after updating the synapses, if its value is zero the permanence value is increasing. For example, here we take first 20 values for input bit 62. For column 1, 11, 18 and 20, the permanence value is increased after updating synapses. A higher synapses value (e.g. 0.9 means that the synapse is strongly connected and cannot be easily disconnected)

|  |  |  |
| --- | --- | --- |
| Column No | After Update | Before Update |
| 1 | 0.06108 | 0.05108 |
| 2 | 1 | 0.99 |
| 3 | 1 | 0.99 |
| 4 | 0 | 0 |
| 5 | 1 | 0.99 |
| 6 | 1 | 0.99 |
| 7 | 0.09578 | 0.08578 |
| 8 | 1 | 0.99 |
| 9 | 1 | 0.99 |
| 10 | 1 | 0.99 |
| 11 | 0 | 0.1 |
| 12 | 1 | 1 |
| 13 | 1 | 1 |
| 14 | 1 | 1 |
| 15 | 1 | 1 |
| 16 | 1 | 1 |
| 17 | 1 | 1 |
| 18 | 0 | 0.1 |
| 19 | 1 | 1 |
| 20 | 0 | 0.1 |

**2.Active-Duty Cycle:**

I did two unit tests to analyze duty cycle method. The two unit tests are:

a)CheckActiveDutyCyclesWithOutUpdate()

b)CheckActiveDutyCycleWithUpdate()

Active-Duty Cycle is essentially just how many times the column has been active for certain period of time.

1. **CheckActiveDutyCyclesWithOutUpdate()**

This is the unit test which I conducted to check when the input is 0 then what will be our results.

Duty cycle, sometimes called "duty factor," is expressed as a percentage of ON time. A 60% duty cycle is a signal that is ON 60% of the time and OFF the other 40% and that is why duty cycle is represented as a value between 0-1.

1 means 100% time ON

0 means OFF

Before updating the duty cycles, all of the columns have zero values and they have never been active.

There are several inputs coming from encoder to spatial pooler. In our case, we took only 10 inputs.

(value ranges from 0 to 9).

Every Input is internally run 3 times by a loop inside the CheckActiveDutyCyclesWithUpdate() and so is called step 0, step 1, step 2. If it is run more than 3 times then more and more columns will get active but here in this experiment we have duty cycle of 0, So even if we run it more then we will have the same output. Here is the output representation of input 1 step 0 where we can clearly see that the columns have never been active.

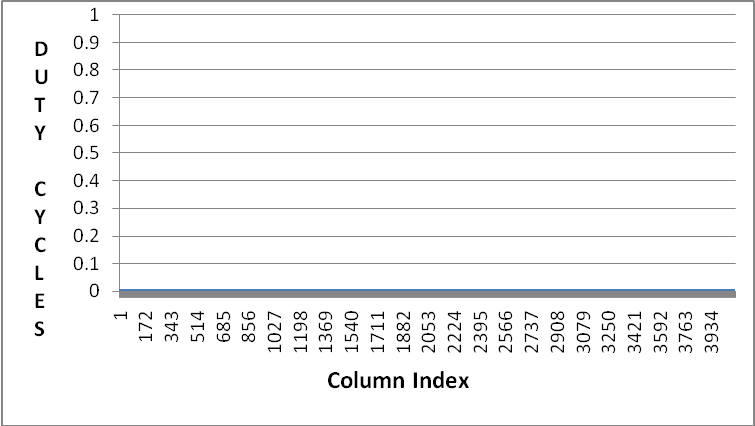


Fig3. Duty cycle before update

Similar results come for all the remaining inputs.

1. **CheckActiveDutyCycleWithUpdate()**

This unit test is conducted when the inputs are applied to spatial pooler from encoder.

In this case the number (1) have been active every time step because the active-duty cycle maximum is 1 as we can see in the figure below. That is the 100% of the time. All of these 1’s have never been turned off. They have been active for the entire time. All of those whose value is 0 have never been active and in many cases they may never be active. Here is the output representation of input 1 step 0 and input 1 step 2.

**Input 1 step 0:** Every input from encoder is internally run three times inside the unit test and is called step 0,1,2. We just took 1 input of many inputs to represent our results.

Columns whose value is 0.75 has been active for 75% of time. Columns whose value is 0.25 has been active for 25% of the time.

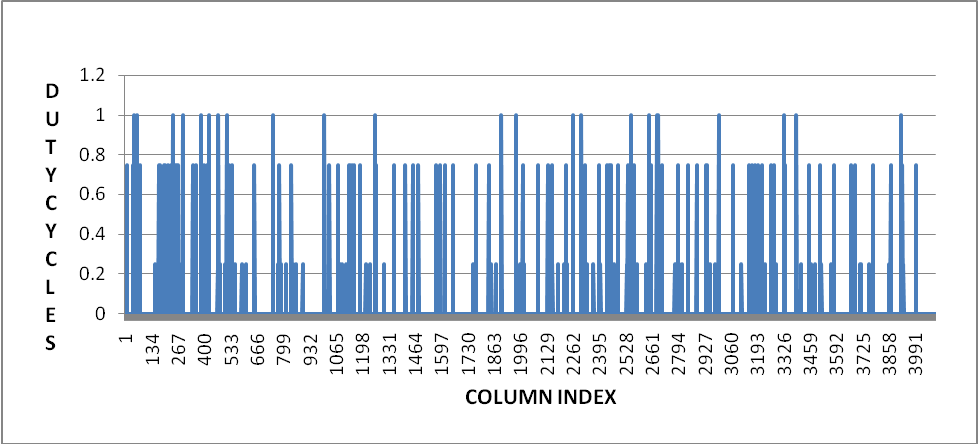


Fig 4. Input 1 step 0

**Input 1 Step 2:**

At input 1 and step 2 the value of columns are much improved than last time. Now more and more columns are getting active. The value of columns are increasing and so every column is getting active for a certain period of time as we can see it clearly on the chart below.

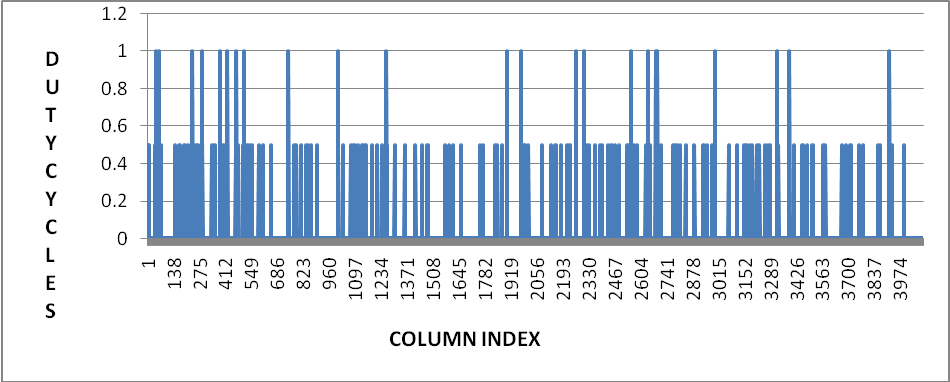


Fig 5. input 1 step 2

**3.Bump Up Weak Column:**

In the HTM determine the active columns of cells. Via the synapses on a proximal dendrite, each column is connected to a subset of the input bits. Subsets for different columns may overlap but they are not equal. For the activation of the columns different input patterns result in different levels. The columns with the strongest activation resist columns with weaker activation. A column’s inhibition area is adjustable, and it can be expanding from very small to the entire region. HTM learns by forming and unforming connections between cells. The permanence value of the active column’s synapses increases when it is connected to active bits and decrease otherwise. The higher permanence value represents the more overlap between the pattern represented by the columns and the input[1].

If any columns do not become active for a long period, it cannot learn anything. The columns whose activity level is too low, those columns permanence value can be increases via “BumpUpWeakColumns” methods. With the overlap duty cycle technique, it is possible to identify those columns, and it reinforces (increments) all the proximal synapses of columns that have lower average input compare to the others.

In the code for unit test, we did not get any value after doing BumpUpWeakColumn. We are predicting that there is no weak column for boosting.

**4.Boost Factor:**

I did two unit tests to analyze boost factor. Every Input in these unit tests are internally run 3 times by a loop inside this experiment and so is called step 0, step 1, step 2. The two Unit tests are:

a) CheckBoostFactorWithOutUpdate()

b) CheckBoostFactorWithUpdate()

The scenario for this testing is to figure out if MAX\_BOOST parameter have effects on the performance of spatial pooler.“MAX\_BOOST” is a maximum number of boost factor. This parameter is used to boost the overlap values of columns. The boost factors are associated with each column. Each column has a boost factor, and it is just a multiplier for the overlap score of that column. So, before we move to the inhibition phase, we can affect which column end up learning and which one end up expressing themselves and which one do not express themselves and that is what boosting does. This is useful because it helps inactive columns to have a chance to “express” themselves and become more competitive with other columns. Each column's overlap gets multiplied by a boost factor before it gets considered for inhibition. The actual boost factor for a column is number between 1.0 and “max boost.”

There are several inputs coming from encoder to spatial pooler. In our case, we took only 10 inputs (value ranges from 0 to 9). Here, we have analyzed the effect of boost factor.

1. **CheckBoostFactorWithOutUpdate()**

In this experiment, When boosting is off (When boost factor is at minimum level of 1 ) there may be small minority of columns that are even contributing to the model at all but the chance is almost none.

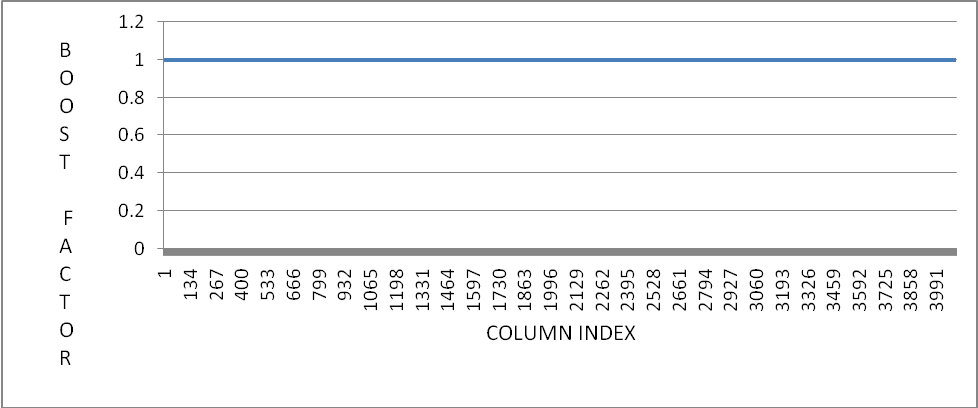


Fig 6. input 1, step 0

1. **CheckBoostFactorWithUpdate()**

In this experiment, The MAX\_BOOST is set 5 here. MAX\_BOOST is a feel that how aggressive our boosting algorithm is going to be to do the homeostatic normalization. When boosting is on, the spatial pooler is much more efficient because it uses more of its columns to represent the data/to model this data as we see in fig 7. So, when we have boosting on, we can spread that around and take advantage of all those columns that could potentially be contributing to the model. Boosting encourages the less active columns become more active. It is very important mechanism in spatial pooler. Otherwise, we cannot get that efficiency what we need. Here is the output representation of input 1 step 0 and input 1 step 2.

**Input 1 Step 0**

When loop is run for the first-time columns gets active and is potentially contributing to the model.

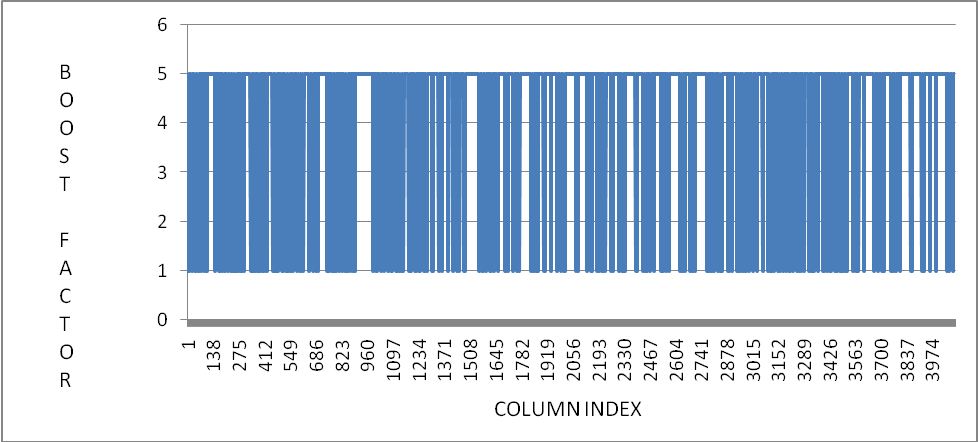
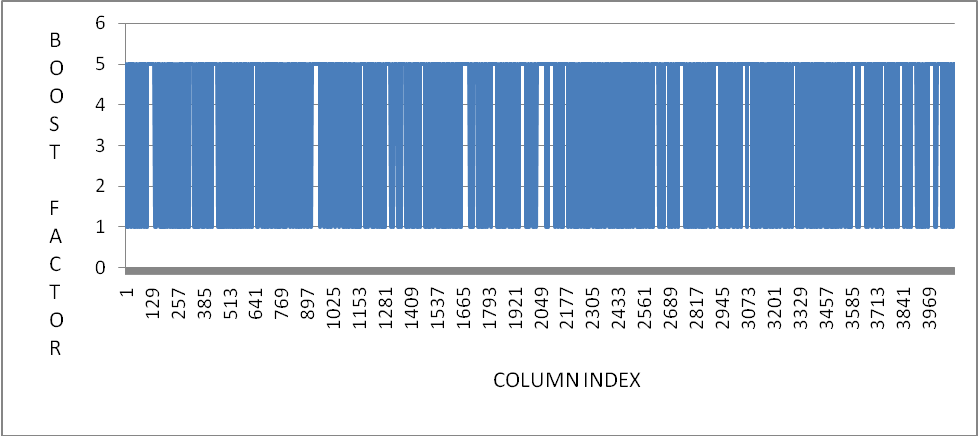


Fig.7 Input 1, step 0

**Input 1 Step 2**

When loop is run for the third time more and more columns get active and taking advantage of all those columns that could potentially contributing to the model.

Fig.8 Input 1, step 2



**5.Inhibition Radius:**

if (IsUpdateRound(this.connections))

{

UpdateInhibitionRadius(this.connections);

UpdateMinDutyCycles(this.connections);

}

}

Here, IsUpdateRound(this.connections) is a Boolean function. If it is true, then we will update the

1. Inhibition Radius
2. Min Duty Cycle

IsUpdateRound(this.connections) is a Boolean function. If it is true, then the learning process will happen. In order to do perform the ‘learning’, the permanence values of the synapses have to be updated and hence the ‘state’ of the model will be modified.

Inhibition is the nameof the mechanism which maintains the sparse activation of the columns [3]. It means, in SP, columns inhibit nearby columns from becoming active. So, inhibition radius is the size of a column’s local neighbourhood, within which the columns inhibit each other from becoming active.

In spatial pooling, if the top k mini columns are activated, then inhibition occurs. It can be called a competition between all the mini columns in a neighbourhood.

The inhibition radius determines the size of a column’s local neighbourhood. A cortical column has to surpass the overlap value of columns in its neighbourhood to become active. The inhibition radius is updated every learning round. It increases and decreases with the average number of connected synapses per column.

There are two types of inhibition. they are [4].

1. Local inhibition: If there are many neighbours and they are distributed throughout the layer, it is called local inhibition. It is more complex, and it assumes topology in the input data.

2. Global inhibition: when there is only one neighbourhood, then all the mini columns compete against each other. This is called global inhibition.

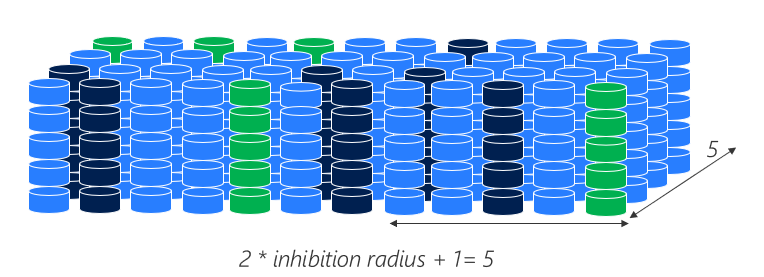


Fig 9: Global inhibition

**6.Minimum Duty Cycle:** Generally, duty cycle is the fraction of time in which the signal is active. In HTM spatial pooling, it means a time duration cycle that notices which column is active for which time after inhibition. This is also called active-duty cycle which call as a function. It updates the minimum duty cycle in a global way.

Before the inhibition, if a column’s overlap duty cycle is less than the acceptable value calculated dynamically as a function of minOverlapDutyCycle and the overlap Duty cycle in the neighbouring columns, then by the increment amount of all its permanence values are boosted.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Step | Min Active-Duty Cycle | Min Overlap Duty Cycle |
| 0 | 0 | 0.00002 | 0.00002 |
| 1 | 0.0000396 | 0.0000396 |
| 2 | 0.000038808 | 0.000058808 |
| 1 | 0 | 0.000038031 | 0.000077631 |
| 1 | 0.000037271 | 0.000096079 |
| 2 | 0.000036525 | 0.00011415 |
| 2 | 0 | 0.00003579 | 0.000131874 |
| 1 | 0.000035079 | 0.00014923 |
| 2 | 0.000034377 | 0.000166 |

I got the above values from the unit test. Here is the minimum active cycle and minimum Overlap duty cycle values are increasing so that cells are more active than before.

1. **CONCLUSION**

Based on the above experiments, it can be said that various parameters of Boosting algorithm have effect on learning. I have clearly observed that parameters like permanence, boost factor, Active duty cycle, minimum duty cycles, overlap duty cycles, inhibition radius affect the learning by changing the values of these parameters. I have tested these parameters by making unit tests for each method in boosting algorithm and the results were documented.

**REFERENCES**

[1]. Hole K.J. (2016) The HTM Learning Algorithm. In: Anti-fragile ICT Systems. Simula SpringerBriefs on Computing, vol 1. Springer, Cham. <https://doi.org/10.1007/978-3-319-30070-2_11>

[2]. Hole, K. J. (2016). *Anti-fragile ICT Systems.* doi:10.1007/978-3-319-30070-2

[3]. <https://numenta.com/resources/biological-and-machine-intelligence/spatial-pooling-algorithm/>

[4]. Mnatzaganian, J., Fokoué, E., & Kudithipudi, D. (2017). A Mathematical Formalization of Hierarchical Temporal Memory’s Spatial Pooler. Frontiers in Robotics and AI, 3. doi:10.3389/frobt.2016.00081

[5]. Pech, Andreas & Ghita, B.V. & Wennekers, Thomas. (2021). Improved HTM Spatial Pooler with Homeostatic Plasticity Control. 98-106. 10.5220/0010314200980106.

[6].https://numenta.com/resources/biological-and-machine-intelligence/spatial-pooling-algorithm/

# References

|  |  |
| --- | --- |
| [1] | D. Dobric, "This is title of the reference," *Some Journal ,* 2021. |
| [2] | D. Dobric, "Some Titel of Reference," 2021. [Online]. Available: se-cloud-2020-2021\MyProject\NeoCortexApi\ neocortexapi\NeoCortexApi\UnitTestsProject\bin\Debug\net5.0\boosting.. |