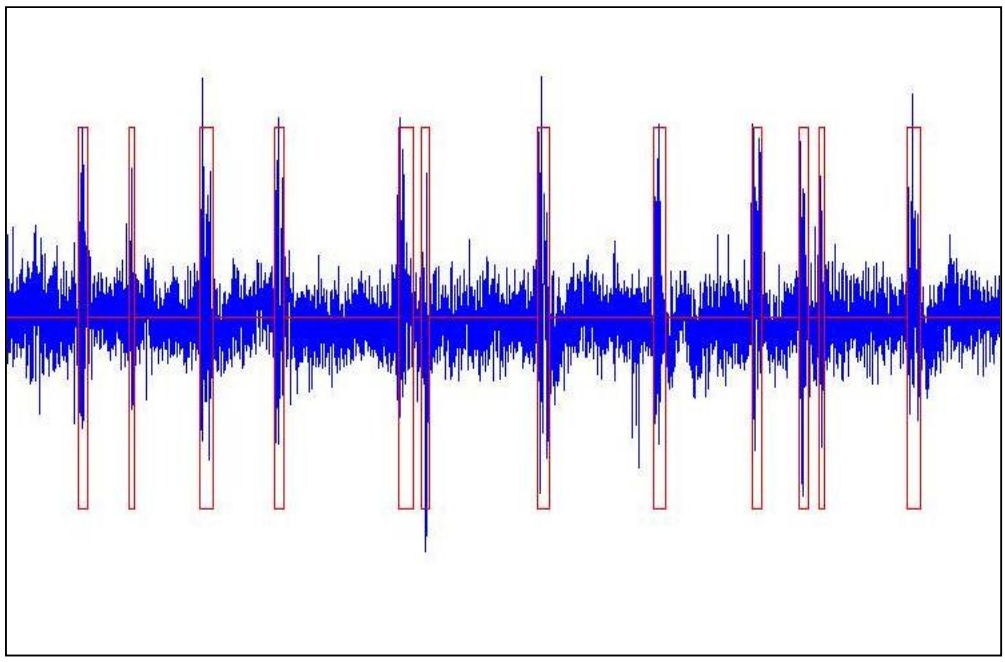
CHALMERS



Automation of Greedy Algorithm for Segmentation with Minimized Exception-Binary for EEG Burst Detection

Master of Science Thesis in Complex Adaptive Systems

SWAPNA SARIT SAHU

Department of Computer Science and Engineering
Division of Algorithms
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, 2009

Automation of Greedy Algorithm for Segmentation with Minimized Exception-Binary for EEG Burst Detection

SWAPNA SARIT SAHU

Gothenburg, Sweden, 2009

Abstract

A survey has been carried out to investigate, various homogeneous criteria and algorithms used for string segmentation. An efficient implementation of greedy algorithm for SME – Binary (segmentation with minimized exception) is carried out for EEG signal burst detection. The result of greedy algorithm is analyzed in the context of successful burst detection for various values of threshold and number of segments. An automation of this greedy algorithm is purposed for the burst detection in EEG signal. The results obtained form this automation is analyzed.

Greedy algorithm for SME-Binary is able to detect bursts. But it is not very successful for a single value of threshold. It is also not known for what value of k (number of segments in a given string), the algorithm will successfully recognize all bursts as k is an input parameter to this algorithm. Automation of this greedy algorithm is quite successful in recognizing the bursts in the EEG signal. The main advantage is, it is independent of value of k and threshold. This algorithm is also quite fast in detecting bursts. The disadvantage is, some bursts are recognized by number of segments rather than a single one. So to count the number of bursts a clustering algorithm is required to enhance this automation. This is purposed as a future work in this report.

Keywords: Homogenous String Segmentation, Survey, Greedy algorithm for SME-Binary, Automation

Acknowledgement

This thesis is written in order to document the result obtained from my master thesis work carried out during my Masters in Chalmers University of Technology, Gothenburg, Sweden. I have taken this work as my master thesis due to my inclination towards algorithm design and machine learning techniques.

This project has been initiated as a master thesis work by Peter Damaschke, associate professor at Chalmers University of technology who gave me the chance to work on this project under his supervision. I would like to express my sincere gratitude to Peter Damaschke for his motivation, enthusiasm, and guidance through out the process of my thesis work. His support and encouragement has immensely helped me towards the completion of this thesis. My examiner is prof. Kristian Lindgren at Complex systems at Chalmers University of Technology, Sweden. I would also like to thank him for the time he has put into guiding me.

Contents

1. Introduction

1.1. Background

In Computer Science, a string is a sequence of data where the data can be from any field or of any background. The string thus can be created by taking a sequence of images, texts, symbols or any time variant data. A string thus formed out of such data can be very huge or difficult to analyze manually. In order to extract necessary and meaningful information from a string, a number of scenarios have been defined. One of the well known problems for analyzing a string is called string segmentation where a string is divided into a desired number of segments. String segmentation is a well known problem in computer science. It has appeared in diverse problems such as in hand writing recognition, speech recognition, image recognition, classification, machine learning and many more. The common string segmentation problem is to partition a string, which is basically a sequence of symbols or numbers into segments by fulfilling a homogeneity criteria with optimality. A number of algorithms have been proposed to solve this problem. A special type of string that has been addressed here is a time series. A time series is a set of consecutive time dependent data points. Time series are obtained from diverse scientific areas including physics, chemistry, geology, economics and many more. Manual analysis of a time series is slow and inconsistent.

Having numerous applications; time series segmentation has been a research area for long. In the last decades numerous methods have been developed to execute automated analysis of time series. In this project, different homogeneous criteria for string segmentation have been surveyed and a segmentation algorithm has been applied for electroencephalograms (EEG) signal data. The results of the algorithm are analyzed and an attempt is made to improve the algorithm for the particular problem.

1.2. Aim

The main objective behind this master thesis is to:

- Survey different homogeneity criteria and methods developed while solving string segmentation problems
- Efficient implement a greedy algorithm for SME – Binary for string segmentation.
- Analysis of the greedy algorithm for SME – Binary for EEG signal bursts detection.
- Automation of the greedy algorithm for EEG signal burst detection.
- Analysis of the result of automation for greedy algorithm for EEG signal burst detection.

As the type of string for segmentation problem can be of numerous backgrounds; hence the main focus will be using EEG signal while implementing and automating the greedy algorithm.

1.3. Scope and Limitations

The survey part of thesis work mainly concentrates on making an extensive search of criteria and methods for string segmentation. Due to limited time this survey has been restricted to mainly time series data even though other types of string data have been considered. The survey tries to discover the different criteria that have been proposed for dividing a string into segments and to study the homogeneity criteria involved to develop the algorithm. Hence the survey does not include the implementation of these criteria or their applications. Thus this survey provides a list of string segmentation criteria and methods only.

An algorithm has been also developed using greedy algorithm as the basis for string segmentation. This algorithm has used an EEG (Electroencephalography) signal as input data. This algorithm is designed keeping in mind of the characteristic of the EEG signal .Of course it will be interesting to apply the same algorithm for various time series data and make a comparative study of it. However the lack of time has restricted the work to only one type of time series data, EEG signal.

The implementation of the greedy algorithm for SME-Binary is done by using linked list and priority queue. The greedy algorithm is chosen to automate since it easier to understand and manipulate the algorithm. Automation of the W-tree algorithm which is proposed in the same paper as the greedy algorithm [8] can also be done in similar way .But it's not done in this thesis due to the lack of time.

1.4. Contributions

In order to carry out this Master thesis work, following activities have been taken

- Study of a number of literatures relating to string segmentation. Careful survey the segmentation criteria and methods mentioned in the string segmentation.
- Implementation of greedy algorithm using an example EEG signal as input data to the algorithm.
- Analysis of the result produced by greedy algorithm for SME-Binary.
- Automation of the greedy algorithm for SME-Binary to detect bursts in EEG signal.
- Analysis of the result produced from automation of greedy algorithm for detection of EEG signal.

1.5. Thesis Structure

The thesis report mainly contains the survey of different homogeneous criteria for string segmentation followed by implementation of a greedy algorithm for said problem. An improvement of the greedy algorithm is proposed keeping in mind of the problem specific data. The thesis report in sort is as follows:

Chapter 1: [chapter 1] Gives a basic idea of the master thesis problem. It also provides the aims and limitations of the thesis work.

Chapter 2:[chapter 2] Gives preliminary knowledge on time series data, EEG signal, greedy algorithm for SME-Binary and string segmentation which helps to understand the thesis work further.

Chapter 3: [chapter 3] Includes the survey of the homogeneous criteria for string segmentation. The survey thus provides a list of string segmentation criteria.

Chapter 4: [chapter 4] Gives the idea on the problem and the greedy algorithm for SME-Binary for study of EEG signal.

Chapter 5: [chapter 5] An implementation of proposed greedy algorithm for segmentation has been discussed.

Chapter 6: [chapter 6] Results obtained from greedy algorithm implementation in discussed for EEG signal burst detection.

Chapter 7: [chapter 7] Automation of greedy algorithm's design and results of obtained form the automation for EEG signal burst detection.

Chapter 8: [chapter 8] Includes a discussion on the results and conclusion of the master thesis work.

Chapter 9: [chapter 9] future work of the thesis work has been stated.

2. Preliminaries

This chapter gives a basic idea on some of the facts that is necessary to make some one familiar to the problem. The following section thus discusses time series data, EEG (a time series data which are special interest to the problem), string segmentation problem, greedy algorithm for string segmentation and the advantage of using greedy algorithm for SME-Binary.

2.1. Time Series Data

Time series data is a sequence of data obtained at successive times mostly spaced over a time interval. A time series data can be represented as a set of observations x_t , each of which is obtained over time t. A discrete time series data is obtained with set of times where observations are made at discrete time intervals, where as a continuous time series is obtained when the observations are made over continuous time intervals. As each data point in a time series are related to a particular time, a time series T can be noted as

$$T = \{(x_1,t_1),(x_2,t_2),(x_3, t_3),\ldots,(x_n,t_n)\} \quad (1)$$

In a continuous time series the difference between two consecutive pair is constant. In other words there exists a 'd' such that for every i where $(0<i<n)$ $t_i=t_{i+1} + d$. The use of continuous time series is more frequent than desecrate time series. Most of the scientific and business data are represented as time series data. Some of the basic examples can be borrowed from finance to show the gain and losses of stock market, from biomedical to show the EEG (Electroencephalograph) signal or from weather forecast to show the climatic changes of a location. Several representations of Time Series Data exist, for example Fourier Transform, Wavelets, Piecewise Linear Representation and Symbolic mapping etc.

In order to obtain the underline meaning or to find a correlation among the data points of a time series, analysis becomes essential. For time series a number of data analysis are available that fit to different purpose. Some of the well known data analysis methods are Graphical analysis, Autocorrelation analysis and spectral analysis etc. A number of algorithms have been proposed for analyzing these data such as classification, clustering, detection of periodicity, segmentation, similarity search and many more. In this thesis work the main concentration is on segmentation problem of a time series data.

In this thesis work an EEG (Electroencephalograph) signal data has been used as a raw for automating the greedy algorithm. This greedy algorithm divides this time series data into segments using homogeneous criteria which is explained in chapter 4.

2.2. EEG (Electroencephalograph)

EEG is the recording of brain's electrical activities during a short time span. The recording is done mostly for 20-40 minutes by placing multiple electrodes on the scalp. EEG is mainly used in the diagnosis of epilepsy (a disease of neurological disorder), coma, encephalopathy (disorder in brain functioning) and brain death. The EEG signal data varies with age. EEG signal of a new born varies greatly in comparison to an adult. As EEG signal is recorded over a time interval each data point depends on the time at which it was recorded. Thus EEG is a time series data.

EEG signal data are generally evaluated manually by neurophysiologists. In order to automate this evaluation process, the main focus is to find length of bursts, suppression intervals and percentage of suppression [2] [4] in the EEG signal. The burst suspension pattern detection is one of the most convincing indicators for recognizing any disorder in electroencephalogram for detecting brain damage. Burst and suppression intervals are considered to be the high and low amplitude segments respectively in the EEG data. Thus segmenting the EEG signal is necessary for automating the evaluation process. One of the attempt to automate the process, some EEG specialists have applied criteria borrowed from generic machine learning which is based on discretized spectral features of EEG signals [25].

By taking EEG as input, the criteria for segmentation considered is known as Segmentation with Minimized Exceptions (SME) .This criteria can be stated as with a string *S* of *n* symbols from an alphabet of size *b*, and some k segments where $k<n$, compute a segmentation of *S* with at most *k* segments that minimizes the total number of exceptions [8]. The sting segmentation is homogeneous if it has minimum exceptions [6] in it. My goal is to detect the bursts in a given EEG signal.

2.3. String Segmentation

Data in the form of string is difficult to monitor and predict. String segmentation makes the data more meaningful and easy to understand. In case of time series data segmentation arranges with some characteristics like linearity, flatness or monotonocity etc.

The segmentation of a time series divides it into a set S of a number of homogeneous segments, where $S= \{S_1, S_2 \ldots S_k\}$ and k = no of segments. Each segment S_i where $(1 \leq i \leq n)$ is defined as consecutive time points that is all the points from point x_a and x_b of time series T as defined in equation 1. Segmentation is not same as partition as all time value pairs in T does not belong to S. The variation to this statement can occur if the segments overlap. Different applications have different time series data as input and the segmentation criteria varies accordingly. This suggests the development of a number of segmentation criteria of time series data. The aim of segmentation may vary from application to application. The main objective of segmentation can be to point out a stable period of time, to locate change point, to get compact representation of the original time series.

2.4. Why Greedy?

There are many kinds of string segmentation algorithms are proposed. But the most common approach is supervised learning such as neural network, hidden markov chain [5][14][28] and dynamic programming [3][11][20]. The problem with the supervised learning is it needs quite significant amount of time and learning data to work properly. It also suffers from over-fitting. In the paper [8] it was also proved that the greedy algorithm is faster than many dynamic programming approaches suggested in [11][20]. Other motivation is to study how good a simple algorithm like greedy [8] can be producing segmentation for EEG signal. Since greedy algorithm for SME- Binary is quite easy to grasp the automation of this algorithm to recognize the bursts in EEG data is also quite simple is nature.

3. Survey

In recent years, there has been a number of interesting works in solving the homogenous string segmentation problem for various scientific data. Different criteria and algorithms have been developed to solve such problems. This chapter mainly shows the result of a survey conducted in order to find out a list of such criteria and algorithms and also to establish relation among them for better readability.

3.1. String Segmentation Criteria

A number of segmentation criteria, proposed so far to suit to particular applications, are mostly some adaptation to some basic criteria. A general categorization of these criteria can be on the basis of the error value minimization. This error value is defined either as a local maximum error or as a global maximum error. If every segment is restricted to a particular error bound while segmenting, then it falls into local maximum error segmentation criteria. On the other hand if the total error of all the segments as a whole is restricted to a global error value then it is global maximum error segmentation criteria.

Based on this categorization, a number of segmentation criteria are given in the article [6].

3.1.1. Cost Minimization Criteria

In order to generate homogeneous segments from a given time series, a cost function for individual segments is defined. This cost function can be defined in numerous ways. In a general sense the cost for individual segment is defined based on the distance between a simple function and the actual time series values. A simple function mentioned here can be a linear or a polynomial function of limited degree. Some of the cost minimization functions are described below.

3.1.1.1 Sum of Variance

The sum of the variance of the variable is used as a cost function in many algorithms. This cost denoted as $cost(S_i(a_i,b_i))$ is the cost of segment $S_i(a_i,b_i)$, is given in equation (1)

$$cost(S_i(a_i,b_i)) = \frac{1}{bi - ai + 1} \sum_{k=a_i}^{b_i} || x_k - v_i ||^2$$

(1)[22]

Where i represents index of each segment, a_i, b_i are the two end points of segment S_i, x_k is any point in Time series T and v_i is the mean of the data vectors in segment Si. A segmentation algorithm aiming to minimize this cost needs to determine the end points of a segment while monitoring model parameters simultaneously.

3.1.1.2 Residual Error

Residual error is another simple cost function considered for segmentation of a time series data. This error for any segment S_i is as given in equation (2)

$$cost(S_i(a_i,b_i)) = \frac{1}{b_i - a_i + 1} \sum_{k=a_i}^{b_i} || x_k - v_i ||$$

(2)[29]

Where i represents index of each segment, a_i, b_i are the two end points of segment S_i, x_k is any point in Time series T and v_i is the mean of the segment S_i. For segmentation this cost is also minimized using dynamic programming. Thus the cost function so defined is the sum of all residual errors of every point in the segment

3.1.1.3 Covariance based Similarity Measurement

Some of the measured variables are found to be correlated. This correlation may vary with time as result of a process transition. In this case the segmentation can't depend on only a single measured value and as a result multivariable statistic is considered. If k^{th} data point of the segment S_i is considered then using the covariance matrices P_k ($a_i \leq k \leq b_i$) the cost function is given as

$$cost(S_i(a_i,b_i)) = \frac{1}{b_i - a_i + 1} \sum_{k=a_i}^{b_i} s_{cov}(P_k, P_{Si})$$

(3)[1]

Where P_{Si} = covariance matrix of S_i which is the average value of all P_k matrices.
s_{cov} = measure of the similarity between two covariance matrices P_k and P_{Si} derived by Krzanowski in 1979 [21]

If first p eigenvectors of two matrices P_i and P_j is considered then s_{cov} is the sum of squares of cosines of angles (Θ). Thus s_{cov} is given as

$$s_{cov}(P_i, P_j) = \frac{1}{p} \sum_{i=1}^{p} \sum_{j=1}^{p} \cos^2 \Theta_{i,j}$$

(4)[1]

Although a number of cost criteria are proposed for computing cost of a segment, the minimization of the cost is mostly done with dynamic programming for an optimized result.

3.1.2. Entropy Minimization Criteria

Entropy minimization can also be considered as homogeneous criteria for string segmentation. In this criterion a selection of attributes is made locally while minimizing the entropy of the classes in a data set.

Let us consider a string S of N data points which is divided into two segments for a better understanding of the criteria. At the beginning all the data points are sorted with their increased value of the attribute. Then the midpoint between each successive pair of data point of the sorted sequence is evaluated as a potential cut point. The best cut point T_A is selected for every continuous-valued attribute *A,* by evaluating all candidates cut point in the range of values. (N – 1) evaluations are made for all continuous valued attribute. For each evaluation of a candidate cut point T*,* the data are partitioned into two sets. The class entropy of the resulting partition is computed. The set S is partitioned into subsets S_1 and S_2. Let there be k classes C_1, C_2,…, C_k and $P(C_i, S)$ be the part of S with class C_i . The class entropy of a subset *S* is given as

$$Ent(S) = -\sum_{i=1}^{k} P(C_i, S) \log(P(C_i, S)) \quad (5)[17]$$

After the formation of two sets S_1 and S_2 out of S, to compute the resulting class entropy weighted average of S_1 and S_2 are taken. Let's assume that for a cut value T, S_1 is the subset of S representing all data points with value less than T whereas S_2 represents all rest of the points in S. The class information entropy of the partition is given by

$$E(A, T; S) = \frac{|S_1|}{N} Ent(S_1) + \frac{|S_2|}{N} Ent(S_2) \quad (6)[17]$$

The best cut point T_A is the one for which the $E(A, T_A; S)$ is minimum. Thus for single attribute A, the discretization is made into two parts. Similar method is employed to discretize all the continuous attributes and a selection of only one is made out of all. Thus entropy minimization is considered for selection of attribute for discretization of the string [18].

3.2. String Segmentation Methods and Representations

The above cost functions are minimized in various ways. The most common approach is dynamic programming. Though dynamic programming approaches results optimal solution, they are slow due to higher time complexity. So there are many greedy approaches described below which though give sub-optimal solutions but they are fast algorithms.

3.2.1. Piecewise Linear Representation

For efficient and effective solution data representation has always been an important factor. Piecewise Linear Approximation (PLA) is one of most common data representation which has been part of clustering, classification, indexing and mining of time series data. If time series data with piecewise linear representation is considered instead of considering all other types of representation (such as Fourier Transforms, Wavelets, Symbolic Mappings) then a number of algorithms can been found out those have been proposed for solving segmentation problem [6]. These time series segmentation algorithms can be divided into three major categories based on the basic criteria employed for segmentation. These categories are given as follows. PLA representation can be done in two ways; linear interpolation where the segments are connected and Linear Regression where the segments are disconnected .There are three ways of obtaining PLA which are given below. The standard notation is used $\theta(F(n))$ is both upper and lower bound. $O(F(n))$ is upper bound and $\Omega(F(n))$ is lower bound.

3.2.1.1. Sliding Window technique

In sliding window an error bound is defined. A segment grows until it has not crossed this error bound. On exceeding this error bound the current segment stops and a new segment is initiated.

The algorithms belonging to this criterion mainly declare an error bound. At the beginning a left point of segment, considered as the anchoring point is mapped to the first point of the data series and the next data points are approximated with increasing segment. A segment keeps on growing until this bound is exceeded. Let say at any point of time t if error of the segment exceeds the declared error bound then segment from anchor till t-1 is transferred as a new segment. The new anchoring point moves to the point t and the process continues until the whole time series data is divided into piecewise linear approximation [6].

The maximum time to compute a segment is $\sum \theta(t) = \theta(L^2)$ where $L = n/k$ (k = number of desired segments) thus L is the average length of each segment. The local time to compute all the segments is $\theta(L^2 k)$ or $\theta(Ln)$ as the number segments can be maximum k and minimum k/c (where $c = T(2)$).

Sliding Window is an online algorithm. A number of variations of these basic algorithms have been proposed. One of such algorithm is suggested to have speeded up on ECG signal by taking t to 'leaps of k' instead of taking 1 is given in the article [23]. Another variation of the basic algorithm is made on the error measurement where the initial value of t is set to the mean length of the previous segment. If measured error is less than the initially declared error then t is continued to increase similar to basic algorithm otherwise it is decremented until the error less than declared error [30].

3.2.1.2. Top down technique

There is always a stopping criteria declared before starting the process. The partitioning of the time series continues until this criterion is not exceeded. In this algorithm all possible partitioning of the time series and the best location is considered to split. The approximation error of the two resulted segments is compared with the threshold error value. If the error is less than the threshold then segments are splitted again. This process continues recursively until all the resulted segments have error below threshold.

The best split point in Top Down approach is at the mid point of data. Each split takes $\theta(n)$ time and $\theta(n^2)$. The next iteration are found out for each half segments and the process is continued. Thus the recurrence formula can be derived as $T(n) = 2T(n/2) + \theta(n^2)$. Thus the lower bound of execution time of the algorithm is $\Omega(n^2)$. In worst case the split points are found at the beginning and thus the recurrence is given by $T(n) = T(n-2) + \theta(n^2)$. If we stop at k iteration the time is given by $O(n^2k)$ where k is the number segments intended.

The available variation to this algorithms are Douglas-Peucker algorithm[10] used in image processing, Iterative End-Points Fits (explained in the text book by Duda and Harts[9]) used in machine learning and data mining. A modified version of Top Down algorithm first finds out the extreme values by scanning the whole data marking peaks and valleys. These extreme values acts as initial splitters and Top down algorithm is used to these segments (Park et al. [13]).

3.2.1.3. Bottom Up technique

In this algorithm, there is a stopping criteria pre-declared like the Top Down approach. The segments are merged instead of dividing, starting from a possible approximation value. The algorithm starts by dividing the n length time series data into n/2 segments. Then it calculates the cost of merging of each adjacent pairs. Iteratively the lowest merging cost pairs are merged until crossing the stopping criteria.

The first step to calculate the segment of pair of points and their cost of merging which takes O(n) time. The time to look for the best cost pair is $\theta(1)$ time and O(logn) to add and then delete the selected ones from heap. The total complexity is computed to be O(Ln) where L = n/k (k = number of desired segments) thus L is the average length of each segment.

The known application of this category of algorithms is found in the field of computer graphics known as decimal methods [15]. Another variation of this algorithm is given by Hunter and McIntosh to the field of computer science, in order to get a high level representation in medical pattern matching systems [16].

Another approach that has been proposed to have the Bottom-Up properties and possess the online quality of Sliding Windows is called SWAB (Sliding Window and Bottom-up).

3.2.1.4. SWAB (Sliding Window And Bottom-Up) technique

This is online algorithm which is nothing but a hybrid of Sliding window and bottom up technique. In SWAB[6] a buffer is created at the beginning with size that can store at least 5-6 segments. To the data in the buffer the Bottom-Up algorithm is applied which detects the left most point. If a segment is reported then it is removed from the buffer and then new values are read into it where the amount of data points depend on incoming data size. This is known as Best_Line function, a basic Sliding Windows technique. This process is repeated as long as the data is available. In a single line in SWAB algorithm, Best_Line function finds data corresponding to a segment and puts into buffer where Bottom-Up algorithm refines the segment.

3.2.2. PAA (Piecewise Aggregate Approximation) Representation

Let a time series is given as $S = S_1, S_2, \ldots S_K$ where S_i is any segment, $0 \le i \le K$ and S consists of n units. Let K be a factor of n. S is represented in K space by a vector $S^{av} = s^{av}_1, s^{av}_2 \ldots\ldots s^{av}_K$. the i^{th} element is given by the following equation (7)

$$s^{av}_i = \frac{K}{n} \sum_{j = \frac{n}{K}(i-1)+1}^{\frac{n}{K} i} S_j \qquad (7)\ [7]$$

It is desired to approximate the n units to K spaces where K << n. The data is defined into K equi-sized frames. A mean value of data coming under such a frame is calculated. A vector of these values becomes the data reduced representation.

Two special cases are considered. First one when K = n, the approximated result is same as the original one. Second one, When K =1, the approximated result is the mean of original sequence. The approximation is produced with a piecewise constant approximation of original sequence thus the name Piecewise Aggregate Approximation (PAA)[7].

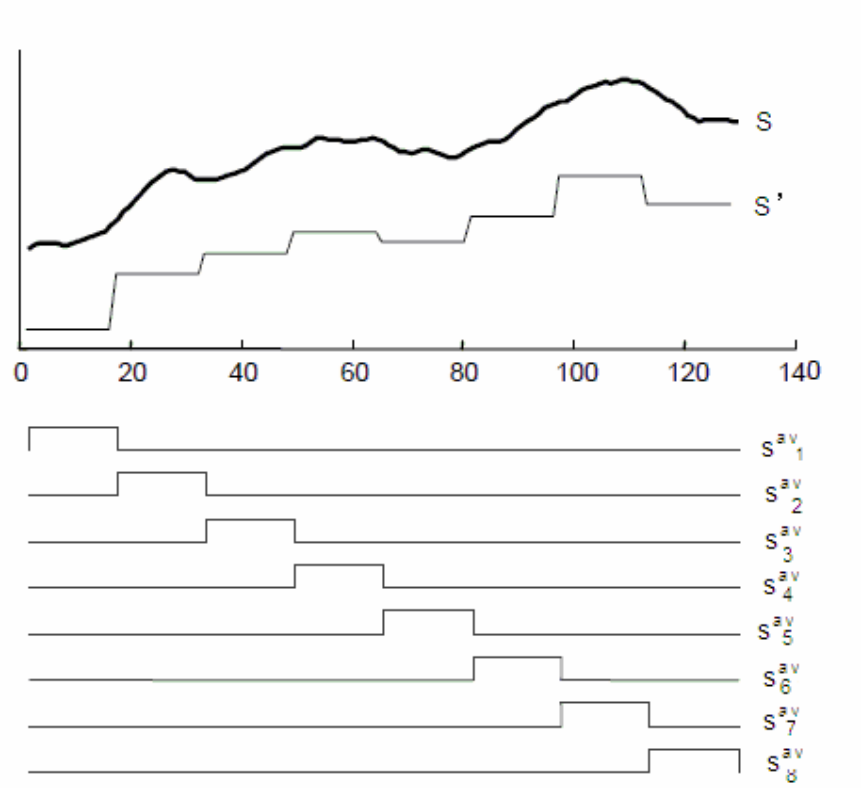


Figure 1: PAA transformed **S' is** generated by approximating S with linear combination of "box" basic functions

Above *Figure 1* gives a visualization of approximation of a sequence with a linear combination of "box" basis functions [7].

3.2.3. APCA (Adaptive Piecewise Constant Approximation) Representation

The problem with PAA that is creates segments of same length. But Using APCA [19] representation one can create segments of arbitrary length. So there are two values in APCA .one is mean value like PAA and the other is length. Given a time series $S= \{S_1 , S_2 , \ldots, S_n\}$.The APCA representation is given by

$S = \{<sv_1 ,sr_1>, \ldots <sv_n ,sr_n>\}$

Where sv_i = mean ($S_{sr\,(i\,-\,1)\,+\,1}$,.....,$S_{sr\,(i)}$)
sr_i = right endpoint of i^{th} segment
So the length of i^{th} segment is given by $(sr_i – sr_{i-1})$

Now, like the previous methods polynomial representation of the time series can be done using optimal dynamic programming algorithms or suboptimal greedy algorithms.

3.2.4. Segmentation based on Energy

In this a sliding window of constant length is considered.(Eg 10 millisecond) . This sliding window is moved over the time series to produce the segments. The energy points at time t_i is given by x_i where segmentation boundary is chosen with some threshold value. The energy based segmentation mainly takes place at the transition point of high and low of energy points. t_i is a segment boundary if $x_i - 1 < T_{threshold}$ and $x_i > T_{threshold}$ or if $x_i - 1 > T_{threshold}$ and $x_i < T_{threshold}$, where $T_{threshold}$ is some threshold. It is assumed that there is data when energy is high in amplitude and zero when it is lower than the $T_{threshold}$. Energy based segmentation is very fast and feasible in real time. This technique has its application in speech and motion time series segmentation.

For speech segmentation, this method produces good results with recordings having little or no background noise. However noise has been a part of speech signals in many practical speech applications. In order to handle noise a number of techniques have been proposed. Based on noise estimation, two common methods are used: 1) Adjust the threshold dynamically based on estimated amount of background noise and 2) preprocesses the audio signal to remove the estimated noise. Another problem faced in this technique other than noise handling is in detection of segment points [27].

3.2.5. Segmentation based on Zero Crossing

Zero crossing is similar to that of energy segmentation. This segmentation technique overcomes the difficulty of handling low energy points as found in energy method while segmenting speech time series. As zero crossing is based on deriving the second derivative of a time series, It has improved functionality in detecting the beginning and the ending. When the second derivative of x_i value transition takes place from positive to negative or from negative to positive, then the segment boundaries are given by either where $x_i - 1 < 0$ and $x_i > 0$ or $x_i - 1 > 0$ and $x_i < 0$. The performance of zero crossing is found to be less productive than energy method in presence of noise if even after noise is reduced using filters. There for zero crossing method is used along with energy method to over come the above problem [12].

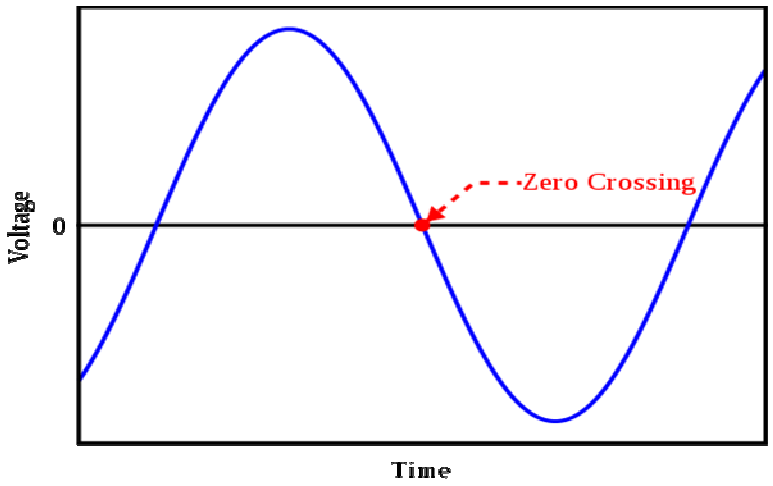


Figure 2: Zero crossing in a wave form showing voltage verses Time

Zero crossing is also applicable to motion time series as it can be adapted to a multidimensional time series. This is done by extracting the features for reducing the number of dimensions which is achieved by selecting one dimension at a time, often the position of end effectors,

and applying the algorithm similar to speech. When the end effector comes to be static or begins to move quicker, there zero crossing finds segment points.

4. Greedy Algorithm for SME-Binary

This is the algorithm that has been designed and analyzed by my supervisor in [8]. In this paper the time complexity of the algorithm is provided along with the necessary proofs. Though, this algorithm in general can be implemented for the real number strings, the only concern here is with the binary ones.

4.1. Problem Statement

The Problem statement in the paper [8] is given as follows

Given a string S of n symbols from an alphabet of size b, and some k < n, compute a segmentation of S with at most k segments that minimizes the total number of exceptions. By SME-Binary we denote SME over alphabet {0, 1}.where SME is segmentation with minimized exception coined in the same paper.

Where, S is a string of n symbols.

k = no of required segmentation.

4.2. Terms and Definition

The necessary terms to understand the algorithm are given as follows. A *RUN* in string is a maximal contiguous substring with only one sort of symbol. Let it be represented by r. E.g. if the string is 000111000 then we can clearly see there are 3 runs. An *EXCEPTION* in a segment is any occurrence of a symbol distinct from the designated one. E.g. If the string is 100001111001110000 and we divided the string into 3 segments 10000|111100111|0000 with designated symbol 0 1 0 then there are 3 exceptions. The first ONE is a exception since the designated symbol is ZERO and the 10^{th} and 11^{th} 0's are 2 more exception since the designated symbol is ONE. k = no of final segments after the minimization of the exceptions. Therefore (S, k) represents the string S with k segments. First and the last segments of the string S are known as the *outer runs* where all the others are known as *inner runs*.

4.3. Algorithm

Any set X of pairwise non-adjacent runs describes a segment with X as the set of exception runs. Thus we can characterize a segment simply by the set X of exception runs. Now if a new exception is added to the set X which is not adjacent to any of the previous members of the set then the string is reduced by two runs since adjacent runs (on both sides of the new exception) of the new exception will now form a single run. If the new exception is a inner run, then the string is reduced by two otherwise it is reduced by one. In the pre-processing step it can be decided which outer runs to keep, and which outer runs to remove. For (S, k) there are two choices in the beginning. If r – k is even then either both the outer runs are included or both the outer runs are removed. On the other hand if r –k is odd then either first outer run is removed or the last outer run. If both the outer runs are counted as exception then the string is reduced by two segments and if only the first one or the last run is counted as exception then we the string is reduced by one in the pre-processing step.

Now, a parameter m is defined, which shows the no of times the greedy algorithms needs to run to produce the desired amount of segments k.

m = (r – k)/2 if r - k is even, and the outer runs are not in X,

m = (r - k - 2)/2 if r - k is even, and the outer runs are in X,

m = (r - k - 1)/2 if r - k is odd.

Now this can be visualized as a graph problem where length of each run is presented by node of weight and these nodes are joined by an edge. Of course this graph is only a linear path. Let L*(X)* is denoted as the total weight of the nodes and *l(v)* is denoted as weight of some node v. A huge constant, INFINITY() is considered for the outer Runs. This is done in order to avoid the outer runs to be selected during below iterative process.

1. This is done m times. Find the minimum *l(v)* of inner vertex v. Merge this inner vertex v with its neighbors u, w (adjacent of the vertex v) into a new vertex z with weight *l(z) := l(u) + l(w) – l(v)*
2. After termination of the (1) we obtain the path of the vertices labeled with odd-length sub paths of the original path. Output as members of X the original vertices which are have even position in the sub paths.

The mechanism of the algorithm is better understood with an example demonstrated as below.

E.g.: Let's take a sting S: = 00000000001111100011011100111111000000. The path of vertices's is obtained as given below

8-5-3-2-1-3-2-5-6

So the total number of runs is nine. Let's say, it is desired to produce three segments. Then m = (9 – 3) / 2 = 3 .Since r –k is even, in the first step of the algorithm (pre – processing step) the outer vertex is set to be infinity (∞). So the result is as given below

∞-5-3-2-1-3-2-5-∞

For m = 1:

Next the vertex with minimum weight is selected which is vertex with 1. This is merged with its neighboring vertex and is replaced with 4 = 2+3-1. With total no of exceptions = 1. Thus the resulted string is

∞-5-3-4-2-5-∞

For m = 2:

Now 2 is the minimum weight vertex. This 2 merges with its neighboring 4 and 5 to produce vertex of weight 4+5-2 = 7 .With total no of exceptions = 1+2 = 3 and gives :

∞ -5-3-7-∞

For m = 3:

Now 3 is the vertex with minimum weight. This 3 merges with its neighbor 5 and 7 to produce vertex of weight 5+7 – 3 = 9. With total no of exceptions = 1+2 + 3 = 6 and the result is

∞ – 9 – ∞

The above is the final output of the greedy algorithm. Now if we see which of the original nodes exist in the solution then we can see only the first and the last outer runs exists in the original solution and there are total 6 exceptions.

Original String is given by:	000000000011111000110111001111000000
Segmented string with minimum exception:	0000000000|11111111111111111111|000000

5. Greedy Algorithm Implementation

In this chapter a description of implementation method of the greedy algorithm is given. C++ is used as a programming language for this implementation. MATLAB is used for visualization and analysis of the results. Two different implementation methods have been employed. The second implementation is more efficient than that of the first one.

5.1. First Implementation of Greedy Algorithm

To implement the greedy algorithm, a datastructure is required for easy insertion and deletion of nodes. There are actually two requirements, first one should return vertex with minimum weight and the second one should allow easy insertion and deletion of the vertices .For the first requirement, priority queue is selected where as for the second one double link list is considered. The reason for not choosing STL priority queue and List is revealed along with the step by step description of implementation method given below.

Vertex is represented by a node class. The members of this class are Original weight or value, Current weight or value, Original Type (either 0 or 1), Current Type (0 or 1), two Boolean flags: Infinity and Removed and Pointers members: Next, Left, Right, smaller and larger. Next pointer points to the next element in the double link list which is never altered during any operation .The Left pointer points to the un-removed left node of the current node and the Right pointer points to the un-removed right node of the current node. The smaller and the larger pointers are used for the priority queue. *Figure 3* is the representation of nodes for a clear understanding.

Figure 3: The dark one is the current node which points to un-removed node to its left and its right. It also points to the next immediate node.

The double link list is represented by a class called nodelist. There are methods in the class to create the list, remove a node i.e. to flag it removed and set the left and right pointers accordingly. To flag a node to infinite and set the pointer accordingly and to check if both sides of a node is removed or not .The priority queue has few more methods than the generic priority queue implementation. The priority queue implemented using a heap data structure which guaranties O(r log r) complexity, where r is the no of nodes or runs. The methods in priority queue class are

Pop: to pop the smallest element in the queue,

Push: pushes a element to the queue,

Remove: to remove a certain element from the queue.

Clear: to empty the priority queue etc.

There are some other methods which is required for the programs to function such as to read the file, to write to the file, make the original data from vector to a binary string, to produce the calculated

length of run, to add a node to the queue or to double link list etc. To explain how the implementation works, I have taken an example below. Let's say the given vertex path is 8-5-4-2-1-3-5-5-6

d = flagged as removed, I = flagged as Infinity, s = remains same, a = alters its type.

The current type of the path is given by alternating 0 and 1.

```
8 5 4 2 1 3 5 5 6
0 1 0 1 0 1 0 1 0
```

In the pre-processing step the outer nodes are flagged as infinity.

```
8 5 4 2 1 3 5 5 6
0 1 0 1 0 1 0 1 0
I                 I
```

For m = 1, the node with minimum weight is returned by the priority queue. It alters its type and its value is replaced by 2 + 3 – 1 = 4. The nodes to the left and to the right of it are flagged as removed. Now the left pointer points to left of the left node and the right pointer points to the right of its right. The nodes 2, 1, 3 are removed from this queue and the new value 4 is pushed to the priority queue.

```
8 5 4 d 4 d 5 5 6
0 1 0 1 1 1 0 1 0
I     s a s     I
```

Now if the middle 4 is returned form the priority queue then its new value is 4 + 5 - 4 = 5. Its current type again changes to 0 which is equal to the un-removed node to its right and left. But since the whole thing in between should be of same type, we alter the type of all the deleted nodes between left node and right node of the current node (using the next pointer which never changes). Similarly the 4, 4, 5 node is now removed from the priority queue and new value 5 is pushed to the queue.

```
8 5 d d 5 d d 5 6
0 1 0 0 0 0 0 1 0
I   s a a a s   I
```

Now let's say 5, next to 8 is retuned as the minimum weight by the priority queue. Its left node is considered as infinity. The type of this node changes to type of the infinity and all the nodes till the right of the current node becomes infinity. The current node and the node right to it, are deleted from the priority queue.

```
8 d d d d d d 5 6
0 0 0 0 0 0 0 1 0
 I  I I  I I       I
```

In the following section a stepwise explanation is given for explaining the above implementation.

5.2. Steps for Implementation

Pop the node with the lowest weight from the priority queue.

Alter nodes Current Type.

If left/right node is infinity {

Then flag node and node's right/left as infinity

Remove node and nodes right from the priority queue.

If deleted nodes exits between infinity and its right/left {

Then flag them as infinity.

Change their Current Type to Infinity's Current Type

}

}

Else {

Remove node's left and right from the double link list.

Remove the node, its left node and right node from the priority queue.

Push the new value of node to the priority queue.

If deleted node exits between node its left and its right {

Change their Current Type to node's Current Type

}

}

The advantage of this implementation lies in its easy trace back mechanism for segmentation. All needed to be done at the end is to print its original value and its current type to produce the output. But this implementation is not quite efficient. The reason is, if a number of deleted nodes appear between the current node and its left and right then one has to traverse all of them to change their type, if such a situation raises quite often then this extra computation can be really expensive. To overcome this problem, a more efficient implementation has been developed in this thesis work and is described below.

5.3. Second Implementation of Greedy Algorithm

In this implementation, priority queue and double link list remain same as above. There is one member to node class, a pointer is added which points to the child of the node. The remove flag is no longer required. This implementation can also be understood by taking an example. Let's say that the string is given by 8-5-4-2-1-3-5-5-6. In the pre processing step we will add infinity to the outer runs.

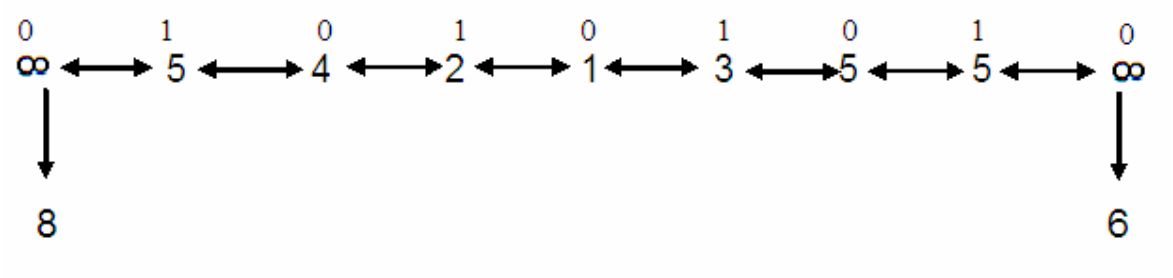


Figure 4: Both the outer Runs are made infinity and the original value is made the child of this node.

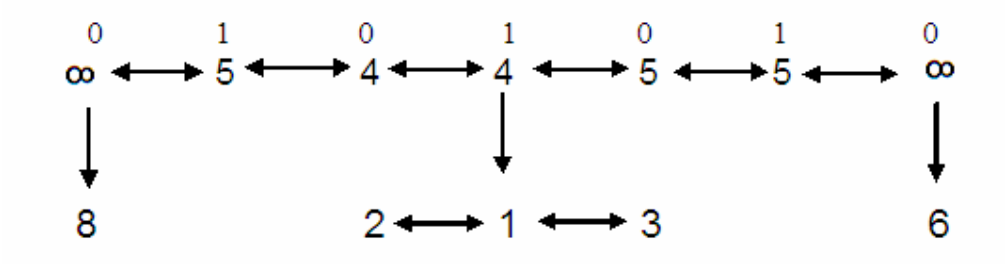


Figure 5: For m = 1. Node with weight 1 is replaced by the new node with weight 4. Node 1 becomes the child of 4 and node 2 and 3 become its siblings. 2, 1 and 3 are removed from the priority queue and 4 is pushed to the queue. The current type of the new node is opposite to its previous node.

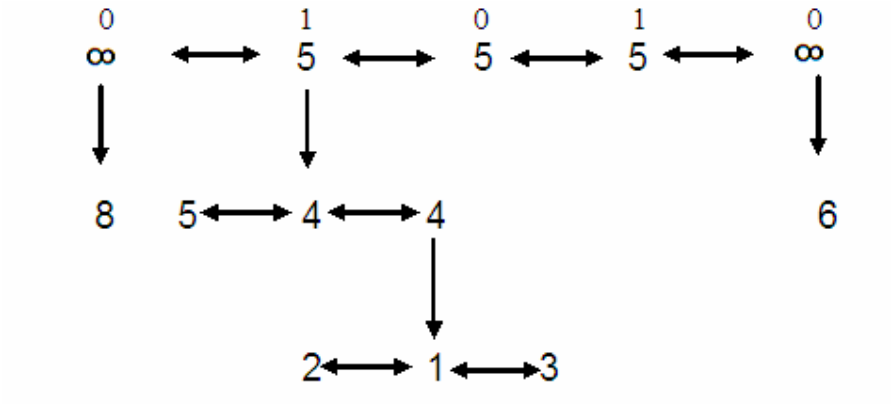


Figure 6: For m = 2, a similar structure follows.

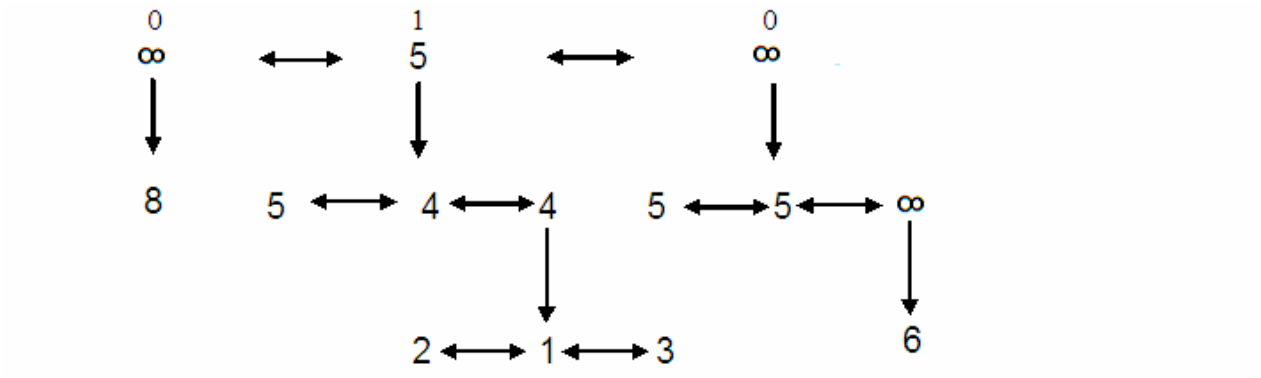


Figure 7: For m = 3. The final result obtained from the given structure.

The advantage of this implementation is that, there is no need to check for the existence of any deleted node between left and right node of the selected node. The above implementation saves lot of computation. The trace back mechanism is quite simple in this case. All we have to do is to go for a node check to find out if there exists a child to a node or not. If there is a child node, then we discard its value and take the values of its children and so on for producing the segmented string.

5.4. Relevant Data for Greedy Implementation

In this thesis, the greedy algorithm is used to detect the number of bursts in EEG data. But EEG data are real values. To convert the EEG data as an input to greedy algorithm, we choose a specific threshold ($T_{threshold}$) and then we take the absolute values of the data points (x_i) of the EEG data. If the abs (x_i) > $T_{threshold}$, then it is taken as 1 and if abs (x_i) < $T_{threshold}$ then it is taken as 0. So for a particular threshold our EEG data is converted into a binary string. It can also be noted that, as the absolute value is taken into account data points above the threshold on the both sides of the axis has been taken into account. In this simplification process we lose the amplitude information.

6. Results Obtained From Greedy Algorithm

In the below section EEG data is used as an input to the greedy algorithm for a particular threshold. Since the greedy algorithm needs k i.e. number of segments as an input, it is not ideal for detecting the bursts. But by using greedy algorithm for a given threshold, it can be seen which bursts are selected. This result can be compared with [Figure 8] (Which is classified by an expert) to see how good greedy algorithm is for a given k and threshold.

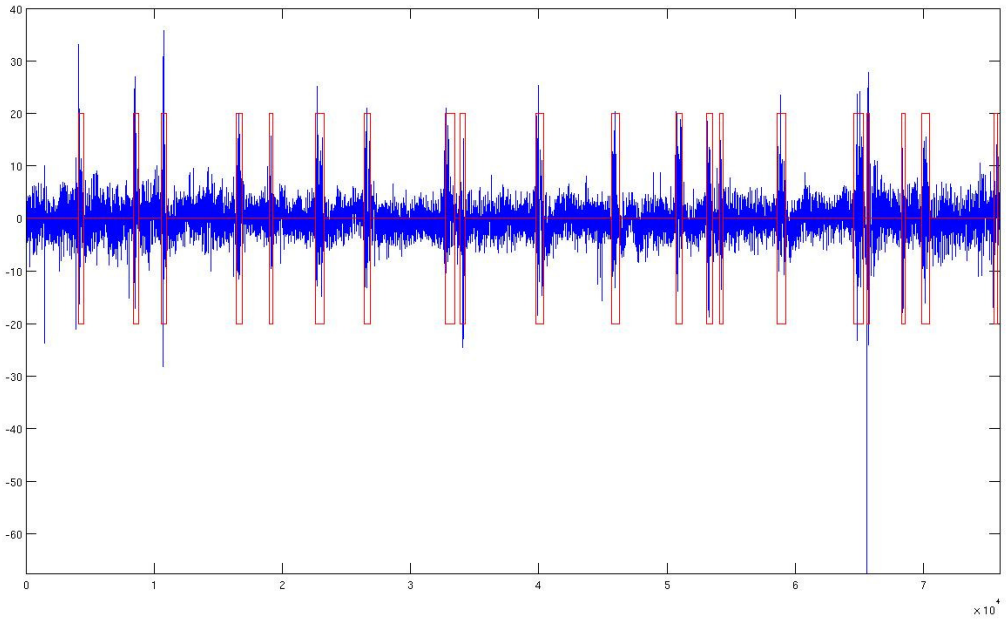


Figure 8: This figure represents a typical EEG data which is segmented for burst detection by an expert.

Since most of the bursts are above the threshold eight. Let's take that as threshold. From the Figure 8 the number of bursts can be easily counted as 21. So there are 43 segments exists for this EEG data. The result of the greedy algorithm is shown in [Figure 9] for threshold= 8 and k = 43. It can be noted that number of runs for threshold eight is 715. So the greedy algorithm produces 43 segments out of 715 segments with minimum exception. Though intuitively eight is a good threshold value (since most of the burst are above this value) and 43 is ideal no of segments (since there are 21 bursts in the given data) ,the greedy algorithm doesn't produce desired result.

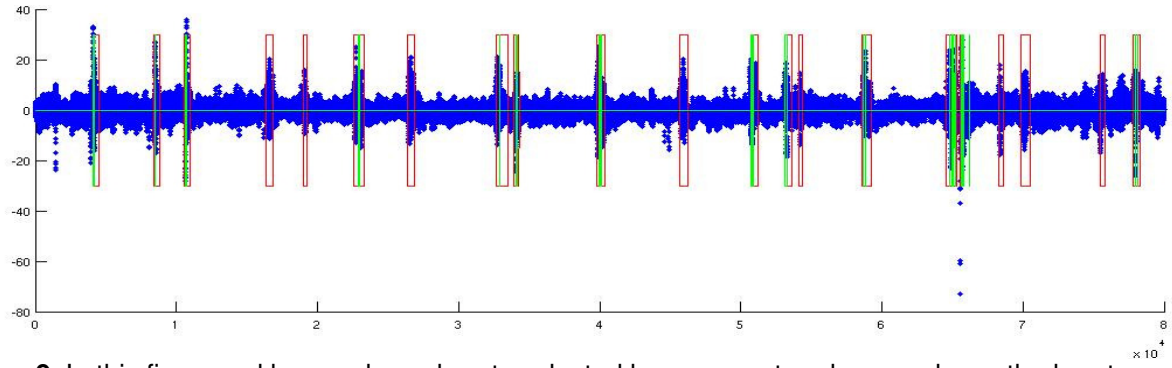


Figure 9: In this figure red boxes shows bursts selected by an expert and green shows the bursts selected by the greedy algorithm

It can be observed from Figure 9 that many bursts are missed by greedy algorithm [4th, 5th, 7th, 11th, 14th, 18th, 19th, and 20th] and some false bursts are detected for k = 43 and threshold = 8.

Taking a higher threshold with lower number of segments can avoid false bursts [Figure 10] but will miss a number of real bursts and taking lower a threshold with higher number of segments can identify all the bursts. [Figure 11] but will produce higher number of false bursts.

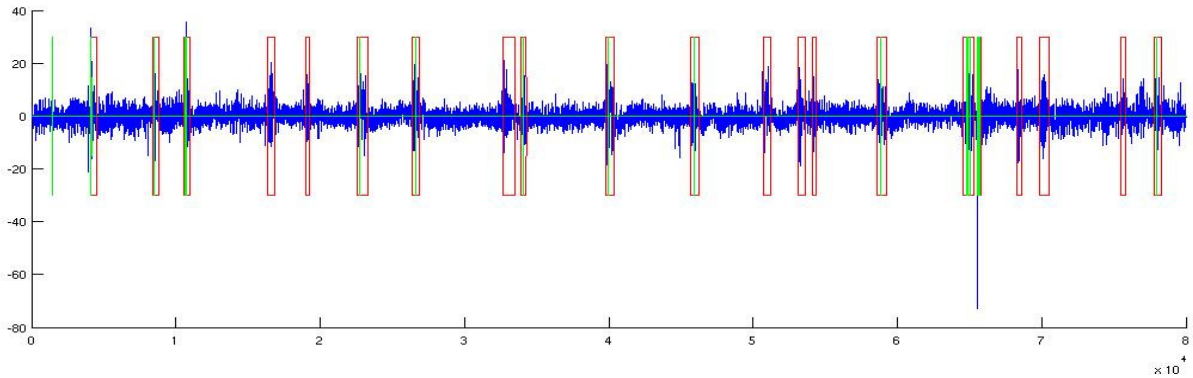


Figure 10: k = 43 and threshold = 20. Many of the real bursts are missed by the greedy algorithm. There are 53 runs for threshold = 20**.**

In this [Figure 10] threshold is high. So the number of runs is very low. Taking value of k = 43 missed a lot of real bursts.

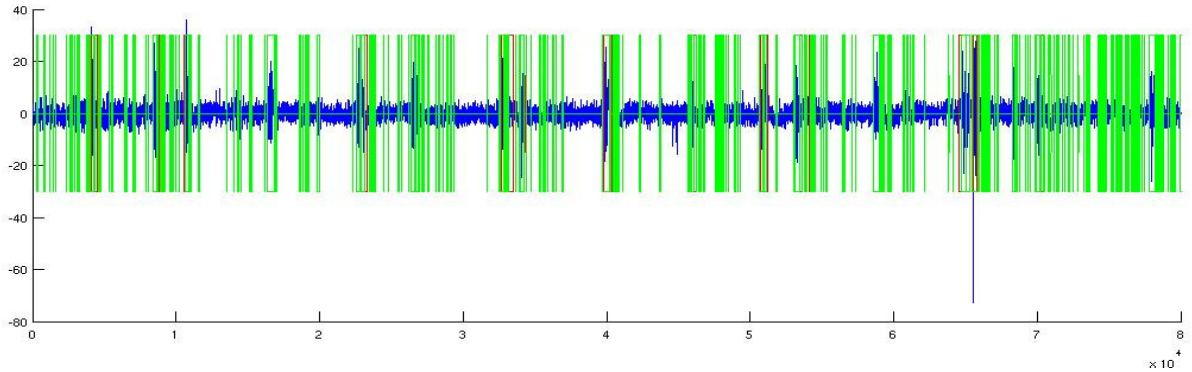


Figure 11: k = 500 and threshold = 3. Many false bursts are selected due to the higher value of k. For k = 3 there are 6601 runs.

In this [Figure 11] a lower threshold is taken. So the number of runs is very high (6601). If the value of k is taken high i.e. 500 in the above case, a lot of false bursts are detected [Figure 11].

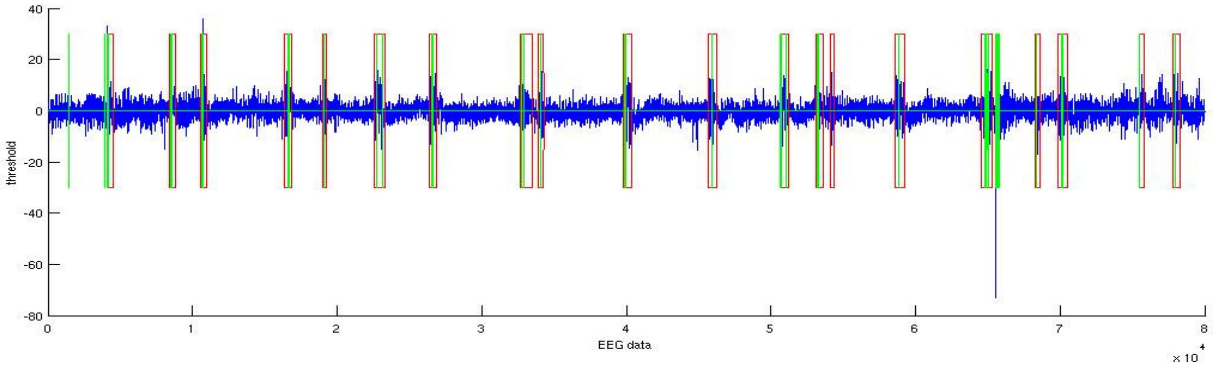


Figure 12: Threshold for the above figure is 15 and k = 100. It can be noted that number of runs for threshold 15 is 151. So 100 segments are produced by the greedy algorithm out of 151 segments with minimum exceptions. Red boxes shows bursts selected by an expert and green shows the bursts selected by the greedy algorithm

This pair of k and threshold produces quite desirable result. But it is a good question to ask why we need 100 segments for threshold 15 to detect all the bursts rather than 43 which is ideal number of segments for given data. An ideal case is one in which a low-high-low segment represents a burst (010 represents a burst). Since 100 segments give better a detection then 43 ideal case, is not been achieved. The reason for such discrepancy can be understood by zooming into one burst [Figure 13].

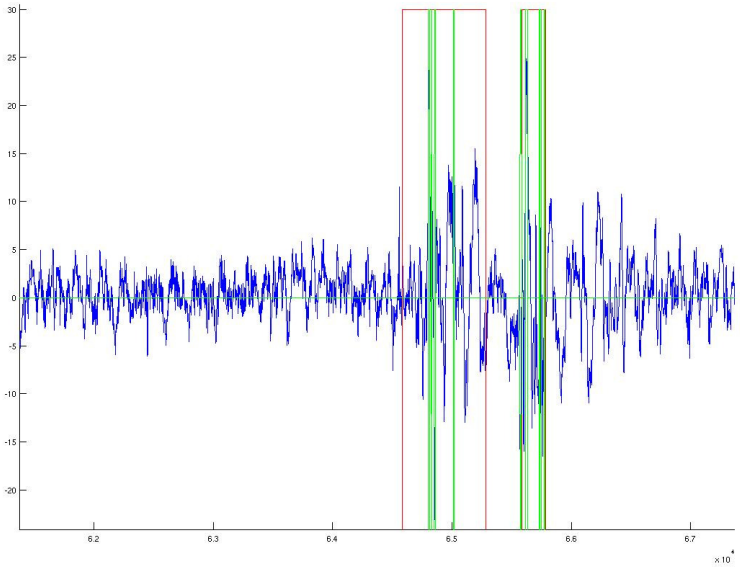


Figure 13: Two bursts at higher zoom level. It can be noted that more than one low-high-low pair is produced by the algorithm in certain bursts.

It can be observed that a single burst is detected by more than one low-high-low pair of segments. The reason is, it is the best way greedy algorithm minimizes number of exceptions for a given string and threshold. So there are more segments required to detect all the bursts then that of ideal case. The result in Figure 12 is a good result because it has only one false burst and one missed burst when run by the greedy algorithm. So to find such a result is an exhaustive search for various values of Threshold and k.

7. Automation of Greedy Algorithm

From the previous chapter, it is clear that greedy algorithm for SME-Binary is able to detect bursts. Since k is an input parameter to this algorithm it can't be used for practical purpose as it is unknown. Given an EEG signal, value of k depends on the number of bursts exists in the data. Since interval of bursts for various EEG signal is different. It can't be said how many bursts on an average exists in the given data.

7.1. Design of Automation of Greedy Algorithm

Design of this automation is based on how greedy algorithm for SME-Binary behaves and the distribution of the points in the EEG signal. Let's say that width of a burst is d and width of an anomaly or a spike is t for a given threshold ($T_{theshold}$) such that t << d . So t will be eaten up by the greedy algorithm rather than d. [Figure 14]

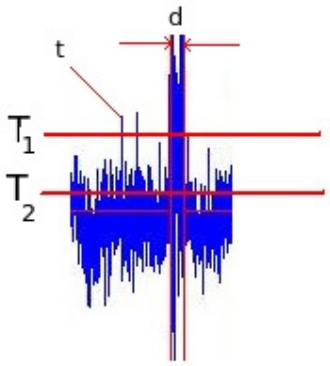


Figure 14: For threshold T_1, First two bursts are will result as a run of lower value i.e. 0's.But the third one will result as a run of higher value i.e. 1's. This may not be the case for Threshold T_2.

Since the amplitude information is lost taking a single threshold, it might be a good idea to combine the classification of bursts at various thresholds and take a majority vote. Since from the previous chapter it is known that higher threshold with small value of k misses the real bursts and lower threshold with higher value of k produce false bursts. The threshold can varied from higher to lower value.

7.2. Pseudo code Automation of Greedy Algorithm

Let S be the EEG signal string, k = number of segments and r = number of runs for Threshold T.V = segmented binary string which is the output of the Greedy Algorithm

```
Variable T
Variable HigherThreshold
Variable LowerThreshold
Variable r
Variable k
Vector   V
Vector   Vsum
Subroutine GreedyAlgo(S, T ,k)

1.   T = HigherThreshold ;
2.   calculate r;
3.  For T = HigherThershold - 1 → LowerThreshold
4.            k = r;
5.            call GreedyAlgo (S, T, k);
6.             return r ;
7.            V= output vector.
8.            T = T -1
9.  end
10.  Vsum = Sum (V)
11. If   Vsum < (logical) (( ( HigherThershold – 1) –  LowerThreshold)/2))
          Then 0
    Else 1
```

A higher value of threshold is taken. The number of runs for this threshold is calculated. Threshold value is lowered and greedy algorithm is run for this value of threshold and input parameter k is taken as number of runs of previous value of threshold. The output binary vector for each value of threshold is saved .Now a majority vote is taken to classify bursts in the EEG signal .There is a bias involved in this design. The bias is, in each new threshold (from higher to lower) as the number of run increases, it is assumed that, these increase in number are exceptions. That is the reason; the previous number of runs is used as the value of k.

7.3. Results obtained from the Automation

The biggest advantage of this automation is, there is no requirement of value of K as an input. Since it is shown in [8] that greedy algorithm for SME-Binary takes O(r + (r-k)log(r-k)) time .This automation runs greedy algorithm for C times where C is a constant . Then complexity of this algorithm is O(C(r + (r-k) log(r-k))).

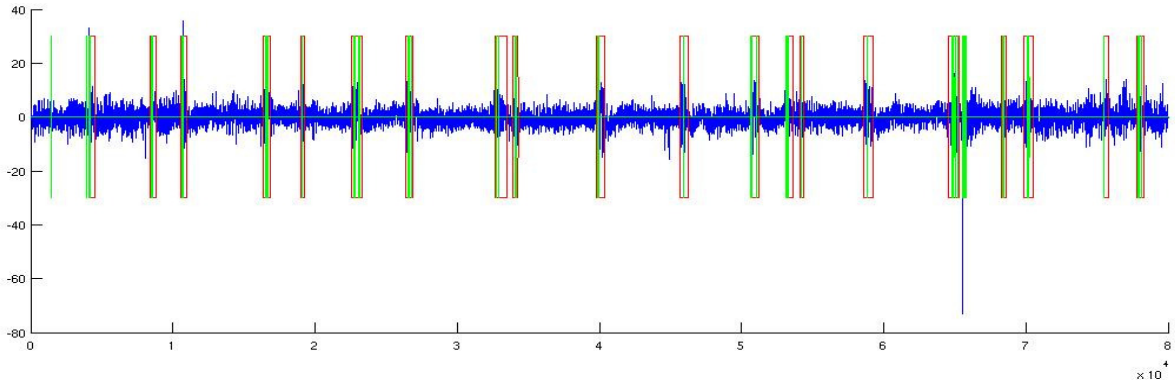


Figure 15 : Higher threshold = 25, lower threshold is 1. More than 50% of the vote is taken. Green boxes shows classification of bursts of the algorithm where as red boxes shows classification by an expert.

In the Figure 15 algorithm is run for the entire amplitude. Since there is hardly any points above the threshold value of 25. The higher threshold is taken as 25. The lower value of threshold is taken as 1.Each pair of threshold and k classifies something as the bursts. A majority vote is taken for these values of threshold and k. It can be observed that 50% of these pairs correctly classify the real bursts. We just get one false burst at the beginning of the string. Closer examination of this burst reveals that it has the same structure that of other bursts. It might be an interesting to observe what happens if this window (higher value of threshold to lower value of threshold) is moved upwards or downwards. The results are showed and analyzed below.

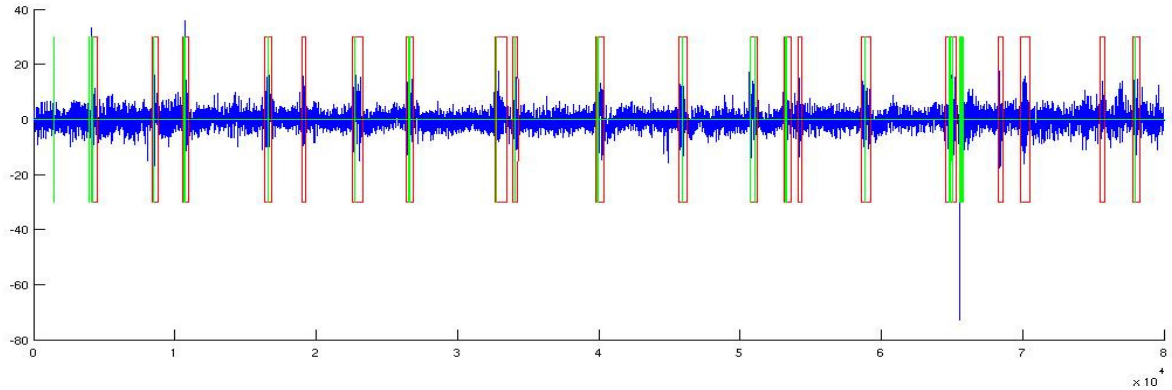


Figure 16: Higher threshold = 25, lower threshold is 10. 50% of the vote is taken. Green boxes shows classification of bursts of the algorithm where as red boxes shows classification by an expert

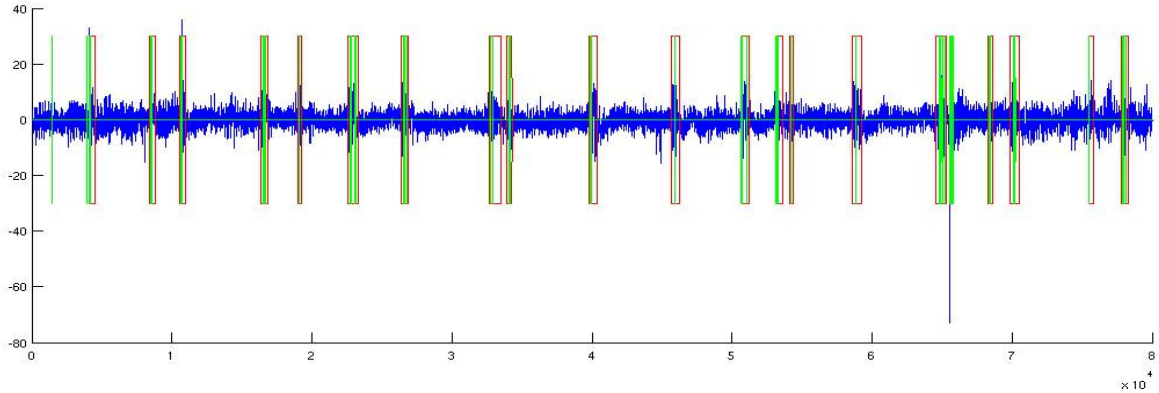


Figure 17: Higher threshold = 25, lower threshold is 10. 30% of the vote is taken. Green boxes shows classification of bursts of the algorithm where as red boxes shows classification by an expert

In the **Figure 16** and **Figure 17** higher threshold is taken as 25 and lower threshold is taken as 10. In can be seen that majority vote (50 %) isn't able to recognize all the bursts where as 30% of vote is able to identify all the bursts. As the basis of this automation is greedy algorithm and it is known in the previous chapter for higher value of threshold and lower value of k, lot of bursts are missed. So this result is expected. So 50% vote can able to detect few bursts only.

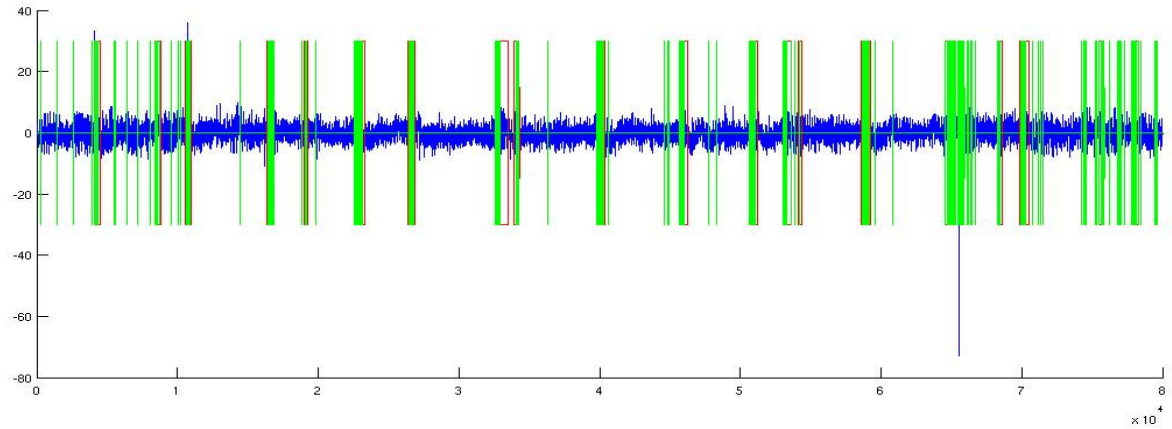


Figure 18 : Higher threshold = 15, lower threshold is 1. 50% of the vote is taken. Green boxes shows classification of bursts of the algorithm where as red boxes shows classification by an expert

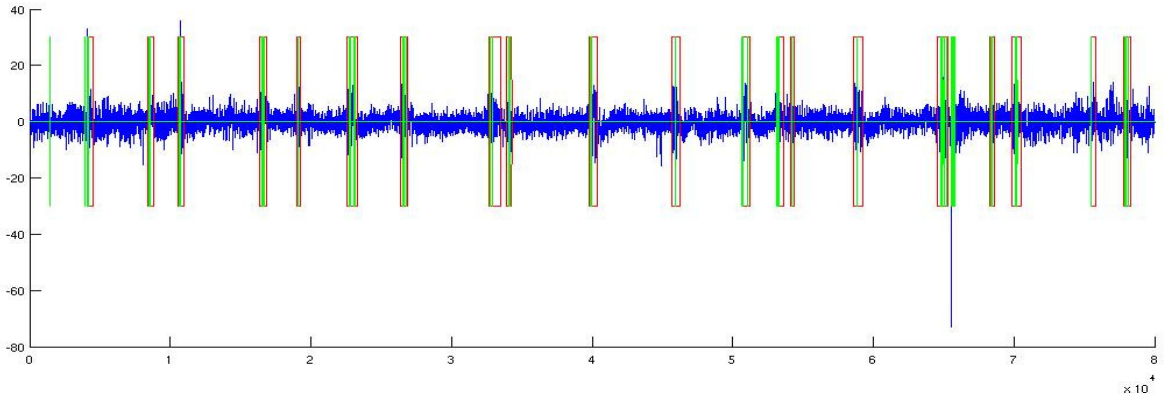


Figure 19: Higher threshold = 15, lower threshold is 1. 90% of the vote is taken. Green boxes shows classification of bursts of the algorithm where as red boxes shows classification by an expert

In the **Figure 18** and **Figure 19**, higher threshold = 15, lower threshold is 1.In can be seen that 50% of vote picks up a lot of false bursts and where as 90% of vote correctly classify the bursts. This is due to the reason that lower threshold with higher value of k results fault bursts .So we need 90% of the vote for identifying the real bursts. This two results also gives an institution that there may exists a range where majority vote (50 %) will result in correct classification. This analysis is done below.

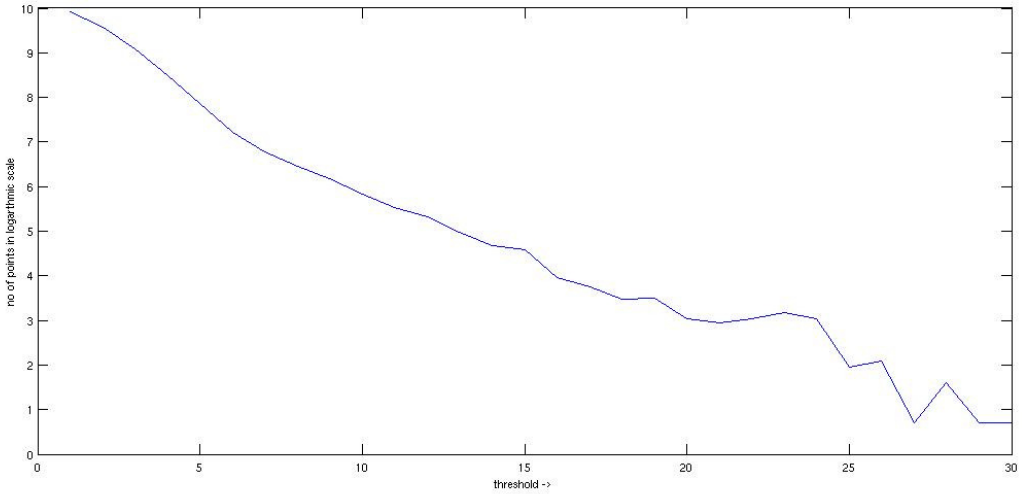


Figure 20: This graph represents number of data points between the thresholds. X-axis represents increase in threshold and Y-axis represents logarithmic increase in data points.

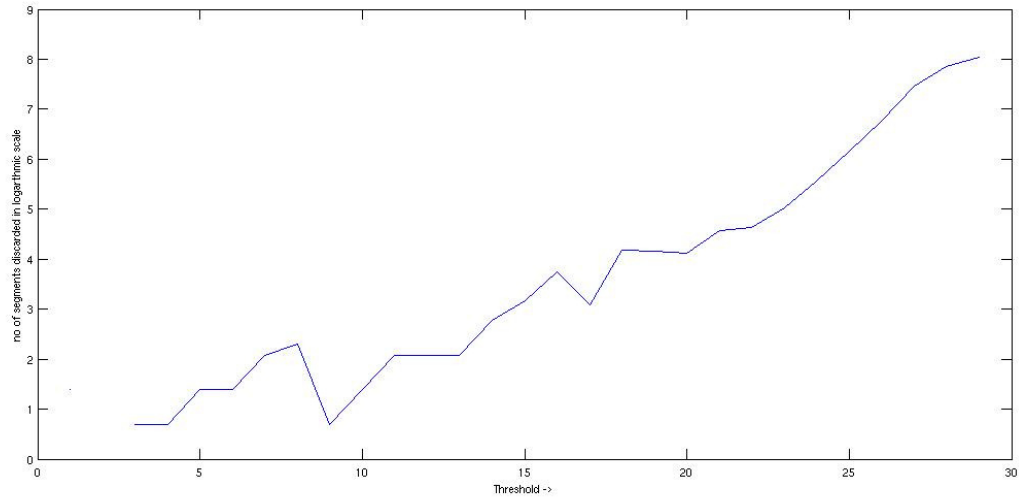


Figure 21: This graph shows how many segments are discarded in each run. X-axis represents increase in threshold and Y-axis represents decrease in the value of k in logarithmic scale.

In the Figure 20, it can be noted that number of data points decreases logarithmically as the value of threshold increases. Careful observation in this graph shows that decrease of data points is linear in logarithmic scale form threshold 8 till 22. The mean distribution of points between these threshold values uniformly decreases in logarithmic scale. Similarly in automation of greedy algorithm for SME-Binary value of k decreases linearly in logarithmic scale from the threshold 9 till 23 [**Figure 21**]. So if we take the majority vote (> 50 %) in this common region then it should classify the bursts correctly. Since both the distribution of points and values of k are linear in this region, it also should be independent of step size (decrease in threshold value.)

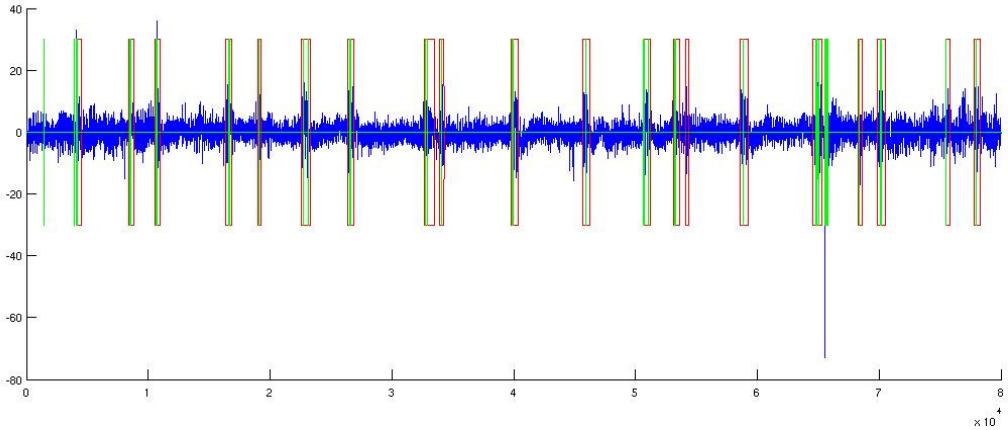


Figure 22 : Higher threshold = 22, lower threshold is 9. 50% of the majority is taken with step size 1. Green boxes shows classification of bursts of the algorithm where as red boxes shows classification by an expert

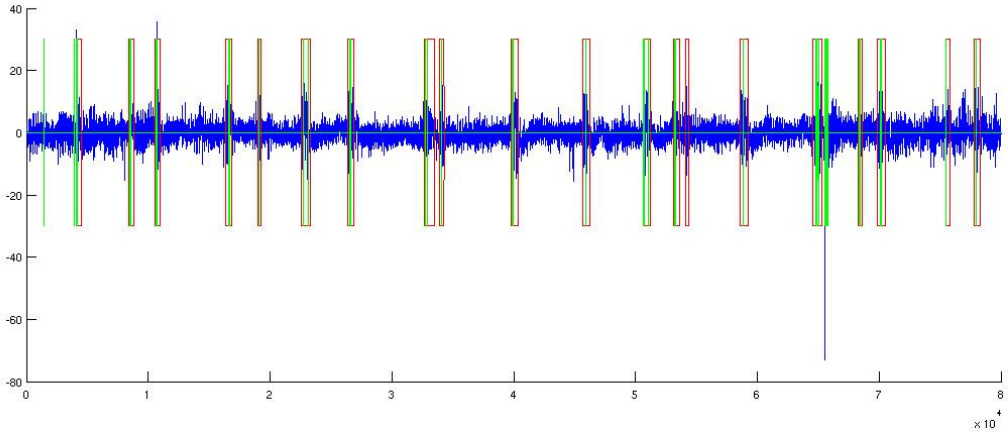


Figure 23: Higher threshold = 22 , lower threshold is 9. 50% of the majority is taken with step size 0.5. Green boxes shows classification of bursts of the algorithm where as red boxes shows classification by an expert.

In the Figure 22 and Figure 23, automation of greedy algorithm for SME-Binary runs between the thresholds 22 till 9. There are two different step size chosen those are 1 and 0.5. In both the cases, majority vote produces correct detection of bursts. Though in case of 25 till 1, majority vote makes correct classification, test results shows that it is not quite independent of step size since assumption of this automation and distribution of points in EEG signal are not linear in logarithmic scale in this region.

8. Conclusions

Greedy algorithm for SME-Binary though a quite simple algorithm, it detects bursts in EEG signal good for certain value of k and threshold. The proposed method of automation of greedy algorithm for SME-Binary is quite well detecting bursts. It is also independent of k (number of segments). So any length of EEG signal can be taken to visualize the bursts in it. One can employ simple statistical technique in pre-processing step to find out linear region of growth of runs and number of data points between the highest and the lowest value threshold. Then this method can be applied in this linear region to identify the bursts. Since in this region majority vote will correctly classify the bursts and it is independent of step length, one shouldn't worry too much in adjusting these parameters. The main disadvantage of this method is a single burst is identified by a number of segments rather than a single one. So output segments just can't be counted to tell the number of bursts in the EEG signal data. This limitation can be solved by using a sliding window of fix length which will cluster the near by segments into one but has a buffer size less than that of minimum distance between two real bursts. This feature is proposed as a future work.

9. Future Works

The main focus of this survey is time series segmentation since the EEG signal data is a time series. But other segmentation techniques such as word segmentation, color segmentation can be surveyed to find out various homogenous criteria and methods.

In the survey cost minimization methods has been given importance since they were studied carefully. Entropy minimizations algorithms take different approaches which aren't surveyed in detail.

Dimensional reduction techniques such as PLA, PAA, and APCA are mentioned in the report .But there are other such dimensional reduction techniques which are somewhat similar to that of PAA. These dimensional reduction techniques are DFT (Discrete Fourier Transformation), DWT (Discrete Wavelet Transformation), and SVD (Single value decomposition). DFT approximates the time series with combination of sine and cosine curves and DWT approximates with Haar wavelets rather then boxes like the PAA method. A survey can be made to see how good each of these methods approximates a given time series.

Automation of the greedy algorithm for SME – Binary produces number of segments to identify a single burst rather than a single one. A clustering algorithm can be designed to overcome this problem. So suppression intervals between bursts and total number of bursts can be easily calculated.

This greedy algorithm can be easily constructed as a learning problem where one can learn for what value of k and threshold this algorithm correctly identifies the bursts. To perform this analysis more data must be available.

The automation of greedy algorithm for SME-Binary can also be used for other time series where there is such high and low values (jig –jazz patterns).

10. References

[1] Abonyi, J., Arva, P., Feil, B., Nemeth, S., Monitoring process transition by Kalman Filtering and time-series Segmentation. *University of Veszprem, Department of Process Engineering, P.O. Box 158, H-8201, Veszprem Hungary.*

[2] Bagenholm, R., Flisberg, A., Kjellmer M., I., Lindecrantz, K., Lofgren, N. and Thordstein, "Infraslow EEG activity in burst periods from post asphyctic full term neonates," *Clinical Neurophysiology,* vol. 116, pp. 1501-1506, 2005.

[3] Bellman, R., and Roth, R., Curve fiting by segmented straight lines,J.Am. Stat Assoc., 64:1079-1084,1969.

[4] Bourgeois,B, Menache, B. F. D., C.C and Volpe, J. J., "Prognostic value of neonatal discontinuous EEG," Pediatric Neurology, vol. 27, pp. 93-101, 2002.

[5] Brand, M. E., and Kettnaker, V., Discovery and Segmentation of Activities in Video. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 844851, 2000

[6] Chu, S., Hart D., Keogh, E., Pazzani, M., Segmenting Time Series: A Survey and Novel Approach, *Department of Information and Computer Science University of California, Irvine, California 92697 USA.*

[7] Chokrobarti, K., Keogh, E., Mehotra, S., Pazzani, M., Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases, Department of Information and Computer Science, University of California.

[8] Damaschke, P., Homogeneous String Segmentation using Trees and Weighted Independent Sets, Algorithmica, an article presented at the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science WG 2007, pp 214-225.

[9] Duda, R. O. and Hart, P. E. 1973. Pattern Classification and Scene Analysis. Wiley, New York.

[10] Douglas, D. H. & Peucker, T. K.(1973). Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature. *Canadian Cartographer*, Vol. 10, No. 2, December. pp. 112-122.

[11] Fragkou, P., Kehagias, A., Petridis, V., A Dynamic Programing algorithm for linear text segmentation, journal of Intelligent Information System 23(2004), 179-197

[12] Fod, A., Mataric, M., and Jenkins, O., Automated Derivation of Primitives for Movement Classification. *Proceedings of First IEEE-RAS International Conference on Humanoid Robots, 2000*

[13] Gong, W., Han J., Yin Y., Mining Segment-Wise Periodic Patterns in Time-Related Databases. Intelligent Database Systems Research Laboratory, School of Computing Science Simon Fraser University, Burnaby, BC, Canada V5A 1S6

[14] Guo, Y., Xue, G., and Tsuji, S., Understanding Human Motion Patterns. Proceedings of the 12th International Conference on Pattern Recognition, pp. 325329, 1994

[15] Heckbert, P. S. & Garland, M. (1997). Survey of polygonal surface simplification algorithms, Multiresolution Surface Modeling Course. Proceedings of the 24th International Conference on Computer Graphics and Interactive Techniques.

[16] Hunter, J. & McIntosh, N. (1999). Knowledge-based event detection in complex time series data. Artificial Intelligence in Medicine. pp. 271-280. Springer.

[17] Irani K.B, Fayyad U, On the handling of continuous-valued attributes in decision tree generation, *Machine Learning 8(1992)*, 87-102

[18] Irani K.B, Fayyad U, Multi- interval discretization of continuous-valued attributes for classification learning; *In Proceedings of the 13th International Joint Conference on Artificial Intelligence*, IJCAI 1983, pp. 1022-1027

[19] Keogh E, Chakrabarti K , Pazzani M, Mehrotra S, Dimension reduction of fast similarity search in large time series databases

[20] Kleinberg, J., Tardos, E., Algorithm Design, Pearson/Addison-Wesley 2006

[21] Krzanowski W.J, Between groups comparison of Principle component , 1979

[22] Kujala, J., Elomaa, T., Improved algorithms for univariate discretization of continuous features, In: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases PKDD

[23] Koski, A., Juhola, M. & Meriste, M. (1995). Syntactic Recognition of ECG Signals By Attributed Finite Automata. *Pattern Recognition*, 28 (12), pp. 1927-1940.

[24] Lamel, L. F., Rabiner, L. R., Rosenberg, A. E., and Wilpon, J. G., An Improved Endpoint Detector for Isolated Word Recognition. IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-29, pp. 777–785, August 1981.

[25] Löfhede, J., Löfgren, N., Thordstein, M., Flisberg, A., Kjellmer, I., Lindecrantz, K.: Comparison of three methods for classifying burst and suppression in the EEG of post asphyctic newborns. In:Proceedings of the 29th IEEE Engineering in Medicine and Biology Society Annual International Conference EMBC 2007 (2007)

[26] Park, S. & Lee, D., & Chu, W. W. (1999). Fast Retrieval of Similar Subsequences in Long Sequence databases*", Proceedings of the 3rd IEEE Knowledge and Data Engineering Exchange Workshop*.

[27] Sekhar, C.C., Yegnarayan, B., Neural network models for spotting stop consonant-vowel (SCV) segments in continuous speech. *IEEE international.*

[28] Starner, T., and Pentland, A., Visual Recognition of American Sign Language Using Hidden Markov Models. International Workshop on Automatic Face and Gesture Recognition, 1995

[29] Terzi, E, Tsaparas, P., Efficient algorithms for sequence segmentation, In: Proceedings of the 6th SIAM Conference on Data Mining SDM 2006

[30] Vullings, H.J.L.M., Verhaegen, M.H.G. & Verbruggen H.B. (1997). ECG Segmentation Using Time-Warping. *Proceedings of the 2nd International Symposium on Intelligent Data Analysis*.

[31] Zhao, L., Synthesis and Acuqisition of Laban Movement Analysis Qualitative Movement Parameters for Communicative Gestures. Technical Report, MS-CIS-01-24, *Ph.D. Dissertation, Computer and Information Department, University of Pennsylvania, 2001.*