Självständigt arbete på avancerad nivå

Independent degree project – second cycle

Datateknik
Computer Engineering

A sliding window BIRCH algorithm with performance evaluations

Chuhe Li

Mittuniversitetet
MID SWEDEN UNIVERSITY

MID SWEDEN UNIVERSITY
Department of Information and Communication Systems

Examiner: TingTing Zhang, tingting.zhang@miun.se
Supervisor: Mehrzad Lavassani, mehrzad.lavassani@miun.se
Author: Chuhe Li, chuhe.li@outlook.com

Degree programme: International Master Programme in Computer Engineering, 120 credits
Main field of study: Computer engineering
Semester, year: Autumn, 2016

Abstract

An increasing number of applications covered various fields generate transactional data or other time-stamped data which all belongs to time series data. Time series data mining is a popular topic in the data mining field, it introduces some challenges to improve accuracy and efficiency of algorithms for time series data. Time series data are dynamical, large-scale and high-complexity, which makes it difficult to discover patterns among time series data with common methods suitable for static data.

One of hierarchical-based clustering methods called BIRCH was proposed and employed for addressing the problems of large datasets. It minimizes the costs of I/O and time. A CF tree is generated during its working process and clusters are generated after four phases of the whole BIRCH procedure. A drawback of BIRCH is that it is not very scalable. This thesis is devoted to improve accuracy and efficiency of BIRCH algorithm. A sliding window BIRCH algorithm is implemented on the basis of BIRCH algorithm.

At the end of thesis, the accuracy and efficiency of sliding window BIRCH are evaluated. A performance comparison among SW BIRCH, BIRCH and K-means are also presented with Silhouette Coefficient index and Calinski-Harabaz Index. The preliminary results indicate that the SW BIRCH may achieve a better performance than BIRCH in some cases.

Acknowledgements

I give my sincere thanks to Prof. Tingting Zhang for giving me the opportunity of working on this topic and supporting me with encouragement, patience and knowledge during my master study.

I also want to express my deep gratitude to my supervisor Mehrzad Lavassani. Her ideas and suggestions always helped me overcome the challenges I encountered during my thesis.

Last but not the least, I would like to thank my family, friends, and those who care about me. Without their support and understanding, I was not able to complete this thesis.

Table of contents

List of Figures

List of Tables

Acronyms

DM	Data mining
KDD	Knowledge discovery from data
TS	Time series
PLR	Piecewise linear representation
SW	Sliding window
TD	Top-down
BU	Bottom-up
BIRCH	Balanced iterative reducing and clustering using hierarchies
CF	Clustering feature
LS	Linear sum
SS	Square sum

1 Introduction

In recent decades, an increasing number of applications covered different fields (e.g. e-commerce, stock market, agricultural yield, sensor) generate web data, transactional data or other data that belongs to time-stamped data. Basically, time-stamped data are made up of any data that contains a time stamp. Specifically, time-stamped data is a type of time series data that associates values with points in time.

Time series data are large-scale, large-size, dynamical, high-dimensional and high-complexity, these characteristics make it difficult to deal with by using common data mining methods suitable for static data. Data are static if these data are not changeable with time or continually streaming, and static data are all fixed in size. Unlike static data, the values of time series data change with time. [1]

Data mining, also referred to as knowledge discovery from data (KDD), is to extract unknown, implicit and potentially useful information from datasets. [2, 3] Or in other words, the main task of data mining is discovering the hidden patterns or data correlations in large datasets with different data mining methods. Researchers will analyse those patterns from different perspectives, for example the found patterns can be used to better understand data, improve past behaviours and predict future data accurately. [4] A data mining system may have large quantities of data, noise or missing data and complex data structure, so multiple data mining tools including classification, clustering, association and regression are needed in a data mining system. Each mining tool contains various algorithms. The procedure of data mining occurs in most of application fields like social science, engineering, business, finance, economy and biology. Data mining is highly required to help people do the event prediction and anomaly detection. Based on different characteristics of time series and requirements of data mining, time series data mining gained lots of focus from many researchers for the last decades.

However, due to the special features of time series data, time series data mining is remaining an opening and challenging issue in data mining area, this also leads to problems with very few algorithms that can analyse time series. Even though there are many kinds of data mining tools, when there are no labelled in dataset, clustering procedure is necessary which acts on identifying structure in unlabelled datasets. Clustering time series data can show effectiveness on providing useful and helpful information in various application fields. [1]

1.1 Motivation and problem statement

So far many researchers have proposed clustering algorithms for handling time series data, from hierarchical clustering (e.g. Chameleon, BIRCH [5, 6]), partitioning-based clustering (e.g. K-means, k-Medoids [7, 8]), to density-based clustering (e.g. DBSCAN [9]), grid-based clustering (e.g. STRING [10]).

A hierarchical clustering algorithm can work by grouping time series data objects into a tree with clusters. BIRCH is one of hierarchical clustering algorithms that is suitable for time series data mining since it can handle large datasets. It can minimize the memory and time required for I/O. BIRCH can dynamically cluster incoming univariate or multivariate time series data points. [6]

Comparatively, one drawback of BIRCH is that it is not scalable. The data objects in BIRCH are inserted line by line and then each object would be stored as vectors. Sliding window, which is one of PLR approaches, can extract sub-sequences from time series sequence. This could be applied into BIRCH algorithm in order to clustering time series sub-sequences. Therefore, one research question in this project is that if the running results performance of sliding window BIRCH algorithm could be better than the existing BIRCH algorithm after adding sliding window on the basis of BIRCH algorithm.

Additionally, the value of inside diameter of a leaf node as threshold is hard to make sure and establish before running BIRCH algorithm. Many papers (e.g. [11]) try to solve the problem of BIRCH algorithm in making threshold value. Different clustering results would be received with different values in a range of threshold values.

1.2 Overall aim

Based on the current problem statement, the general aim of this project is to implement a sliding window BIRCH algorithm on the basis of BIRCH algorithm and make a performance evaluation comparison among three different clustering algorithms. Comparing with BIRCH algorithm, the sliding window BIRCH algorithm would have higher efficiency and accuracy, and become more scalable. If the evaluation results show that efficiency is higher while accuracy is lower, then making a balance between efficiency and accuracy.

Additionally, sliding window BIRCH should process time series datasets and satisfy the requirements of time series data mining. To summarize above, this project aims to carry out a hierarchical clustering algorithm which is sliding window BIRCH with better efficiency and accuracy than BIRCH and K-means that in order to handling time series data.

1.3 Research questions

Considering the opening challenges of clustering time series data mining and the shortcomings of BIRCH algorithm. A list of research questions is proposed as follows:

If the performance evaluation be improved after applying a PLR method into clustering time series procedure?

How to improve the efficiency and accuracy of BIRCH algorithm?

1.4 Methodology

This study would be carried out through several steps. It starts with literature study in order to be familiar with the basic knowledge and related work with clustering time series data mining as well as other extended BIRCH algorithms. Then the open issues in current time series time data mining would be identified. Some research questions can be proposed. The main aim of this study can be established. Thus, as discussed before, this project mainly aims at implementing a sliding window BIRCH algorithm on the basis of BIRCH algorithm for handling time series datasets and evaluating the results performance of this new algorithm by comparing with other clustering algorithms from different aspects. Then, three concrete studies would be designed to achieve the aim.

Firstly, data preprocessing or data preparation including data understanding, data cleaning and data normalizing. Data cleaning is to remove missing values or incomplete data. Data normalizing can let data be ranged from 0 to 1 which is helpful for further work.

Secondly, accomplishing the implementation of sliding window BIRCH algorithm. The sliding window BIRCH algorithm would be implemented from phase 1 to phase 4 which is same as in BIRCH algorithm but adding a sliding window which is used for extracting sub-sequences as time series data objects from time series sequence.

Thirdly, algorithm evaluation that evaluating sliding window BIRCH algorithm by comparing it with BIRCH and K-means. The comparison attributes would contain execution time and cluster quality.

As a last step, a conclusion including advantages and disadvantages of the sliding window BIRCH algorithm would be pointed out. The future work for any improvement would be presented as well.

1.5 Scope

The study has its focus on time series data mining, finding an effective solution of time series clustering then improving BIRCH algorithm's performance by adding a sliding window. Comparisons among different clustering algorithms as performance evaluations of clustering algorithm would also be covered in order to show the validity indices of sliding window BIRCH.

1.6 Outline

The structure of this thesis report is organized as follows: chapter 1 introduces the background and research motivation and give a brief introduction of overall aim, scope and concrete goals. Chapter 2 describes the concept of time series data mining, common data mining tools, clustering time series, BIRCH algorithm, and related work to improved BIRCH algorithms. Chapter 3 introduces methodology used in this study, including the research steps, challenges and how to deal with those challenges. Data preprocessing and implementation of sliding window BIRCH algorithm is carried out in chapter 4. Then the results and performance evaluations of sliding window BIRCH are in the chapter 5. Finally, chapter 6 claims the conclusions of this work, ethics consideration and future work are mentioned on the basis of this study.

2 Literature study

Data mining has been a popular research for last several decades. The main task of data mining is discovering hidden patterns in large datasets. As attributes have different types, it is important to choose an appropriate mining method that can be used in data mining process. Many data types exist in current rich datasets, such as time series, graph, multi-media, test, image. Time series data will be highlighted in this work.

Huge amounts of time series data are produced by applications in different fields over the last decade. In respect of data features, time series data has unique features like dynamically, continually and high-complexity.

Mining all meaningful time series data is the purpose of time series data mining. Considering that time series has distinctive features, time series data mining is remaining an open issue for researchers in data mining research area. [12]

2.1 Time series data mining

Time series is one of the data types in datasets which represents an ordered data sequence obtained from steaming data over time. Given a time series, the trend of data based on changeable value of each time instant can be observed. A time series can thus be defined as a set of contiguous time instants. This series can not only be univariate, but also be multivariate like figure 1. Figure 1 represents multiple variables (a, b, c and d) time series within the same time range. [12, 13, 14]

In the age of data, huge volumes of transactional data, web data and other time-stamped data are continuously generated as time series data. As time series data exists in current different fields, such as business economics (e.g. stock data), social science and engineering (e.g. sensor data), an increasing number of applications could be related to dynamic time series data.

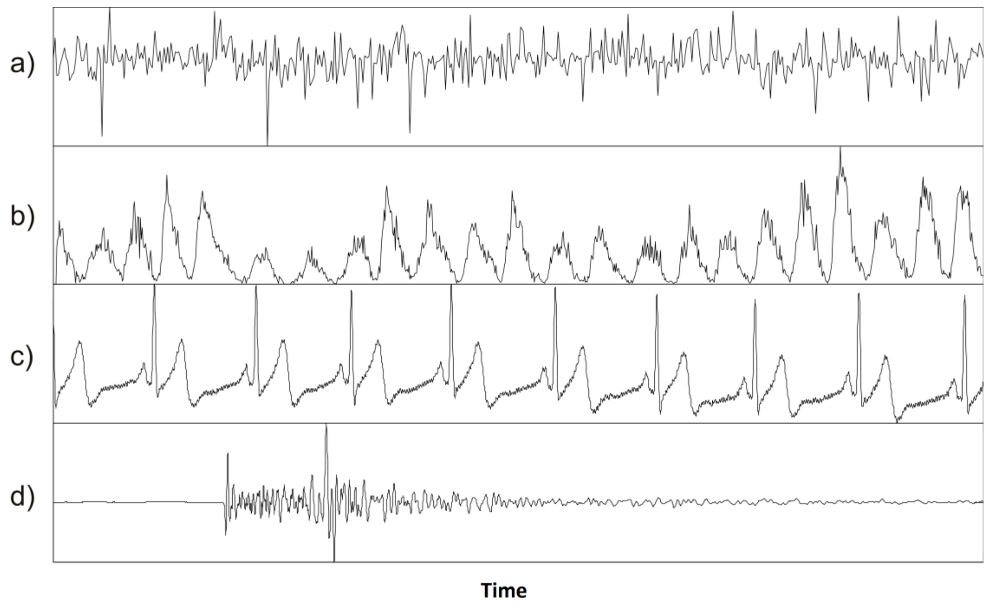


Figure 1 Example of time series data [15]

Dynamical, large-size, large-scale, high-dimensional and high-complexity are features that belong to time series data. Based on the requirements of mining time series data, time series data mining is one of the most challenging research directions in the data mining area based on above features. There are lots of time series data: data representation, similarity measure, time series clustering, time series classifying, anomaly detection and prediction. Nevertheless, the fundamental problem of time series data mining is how to present the time series data. [16, 17, 13]

Data representation plays a key role as solution to ensure the efficiency and effective of a data mining method, simultaneously it is hard to find the boundaries. There are several high-level data representation approaches that have been proposed, for example Fourier transforms, Wavelets and Piecewise linear representation (PLR). Among above approaches, PLR (e.g. top-down, bottom-up, sliding window) is the most used representations. [18]

Similarity measure which is also called distance measure is required during classifying and clustering. Once the similarity is defined, it is easy to calculate the distance between two objects. For time series data mining, Euclidean distance, Manhattan distance, Dynamic time warping (DTW) and Minimum description length (MDL) are all frequently used distance measures. [17, 14]

Here the Euclidean distance is highlighted because Euclidean distance is a distance measure method and it is used in BIRCH algorithm. It can calculate the dissimilarity of two entries, if a = (a1, a2, a3, …, an) and b = (b1, b2, b3, …, bn), then the formula is

$$\text{Euclidean\_Dist} = [(a1-b1)^2 + (a2-b2)^2 + (a3-b3)^2 + \ldots + (an-bn)^2]^{1/2}$$

Manhattan distance, also referred to as city block distance, is a form of geometry in which the usual distance function of metric is replaced by a new metric in which the distance between two points is the sum of the absolute differences of their coordinates. The formula is

$$\text{Manhattan\_Dist} = \|\ a-b\ \| = \textstyle\sum_{i}^{n} |\ a_i-b_i\ |$$

where a = (a1, a2, a3, …, an) and b = (b1, b2, b3, …, bn), a and b are vectors.

Dynamic time warping (DTW) can calculate an optimal match between two given time series data with certain restrictions. The time series are warped non-linearly in the time dimension.

Minimum description length (MDL) is a formalization of Occam's razor in which the best hypothesis for a given set of time series data is the one that can lead to the best compression of the data.

Time series clustering is that given a dataset of n time series data $N = \{N_1, N_2, N_3, \ldots, N_n\}$, the process of unsupervised learning partitioning method is group N original datasets into k groups $M = \{M_1, M_2, M_3, \ldots, M_k\}$ based on the similarity measure. The k is therefore much smaller than n. Above groups M are clusters that generated by above clustering process. [19] Most of the clustering algorithms aim at processing static data, but in respect of time series data or streaming data, time series clustering is required to solve the mining problems.

Time series classifying is that given a dataset of unlabelled time series data, the process of semi-supervised learning method is to map each time series data to one of the pre-defined classes. [20]

Anomaly detection which is also called outlier detection refers to the problem of finding patterns, items or events that do not confirm to expected behaviour. As mentioned before, anomalies are patterns which are not confirmed to the well-defined notion of normal behaviour in datasets. [21]

Prediction allows to predict outcomes by using supervised, semi-supervised or unsupervised learning method. Thus, the trend and behaviour patterns can be predicted.

2.2 Piecewise linear representation

Sliding window

Given a set of time series data T which has length m, a subsequence length which is defined with w and a matrix of S, which is composited by all possible sub-sequences S = {S_1, S_2, S_3, …, S_n}, is then built by sliding a window with length w across time series T and generating a sub-sequence S_i. [22] Figure 2 shows an example of sliding window over time series data.

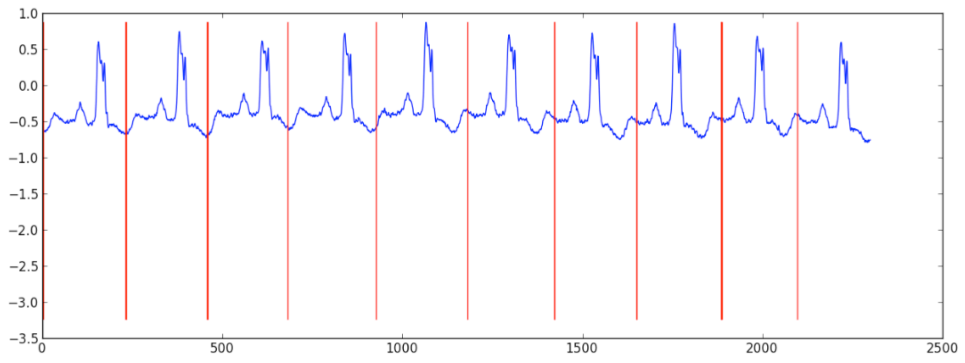


Figure 2 Example of SW over TS data

Sliding window is one of the piecewise linear representation (PLR) methods. Most time series segmentation algorithms can be categorized as follows: top-down, bottom-up and sliding window. [18] Since time series has infinite length (extremely long) in most cases, segmentation method is required to segment an input time series and then get several sub-sequences that each with finite length. Based on that, the hidden patterns and underlying properties of original time series data could be revealed.

```
Algorithm Seg_TS = Sliding_Window(T , max_error)
anchor = 1;
while not finished segmenting time series
i = 2;
 while  calculate_error(T[anchor: anchor + i ]) < max_error
   i = i + 1;
 end;
 Seg_TS = concat(Seg_TS, create_segment(T[anchor: anchor + (i-1)]);
 anchor = anchor + i;
end;
```

Figure 3 The generic sliding window algorithm [18]

Top-down

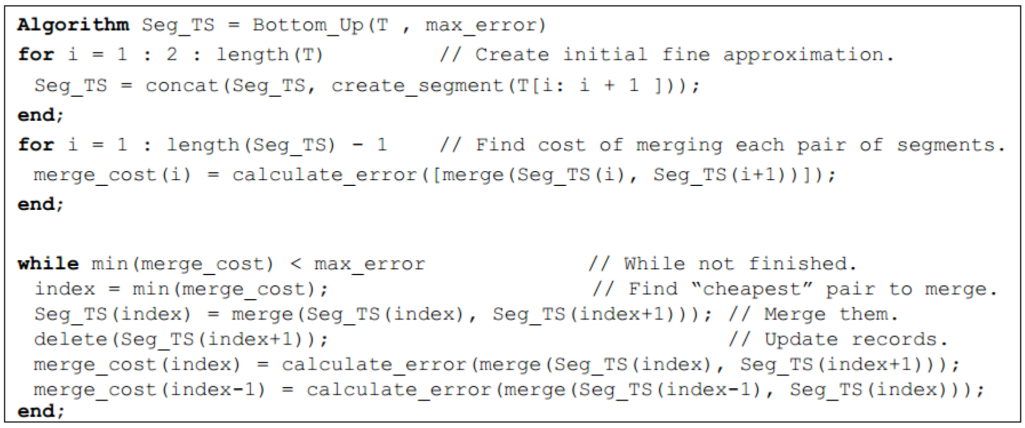
Algorithms using top-down approach consider every possible partitioning of the dataset and split it when breaking down. [18] It starts with the root node and then breaks down into child nodes or smaller segments. This top-down approach runs from complex to simple.

```
Algorithm Seg_TS = Top_Down(T , max_error)
best_so_far = inf;
for i = 2 to length(T) - 2  // Find best place to split the time series.
improvement_in_approximation = improvement_splitting_here(T,i);
  if improvement_in_approximation < best_so_far
    breakpoint = i;
    best_so_far = improvement_in_approximation;
  end;
end;
                            // Recursively split the left segment if necessary.
if calculate_error(T[1:breakpoint]) > max_error
  Seg_TS = Top_Down(T[1: breakpoint]);
end;
                            // Recursively split the right segment if necessary.
if calculate_error( T[breakpoint + 1:length(T)] ) > max_error
  Seg_TS = Top_Down(T[breakpoint + 1: length(T)]);
end;
```

Figure 4 The generic top-down algorithm [18]

Bottom-up

In contrast, bottom-up algorithm merges two segments or two sub-systems together to give rise to more complex systems. [18] It starts from specific system.



```
Algorithm Seg_TS = Bottom_Up(T , max_error)
for i = 1 : 2 : length(T)         // Create initial fine approximation.
  Seg_TS = concat(Seg_TS, create_segment(T[i: i + 1 ]));
end;
for i = 1 : length(Seg_TS) - 1    // Find cost of merging each pair of segments.
  merge_cost(i) = calculate_error([merge(Seg_TS(i), Seg_TS(i+1))]);
end;

while min(merge_cost) < max_error              // While not finished.
  index = min(merge_cost);                     // Find "cheapest" pair to merge.
  Seg_TS(index) = merge(Seg_TS(index), Seg_TS(index+1))); // Merge them.
  delete(Seg_TS(index+1));                               // Update records.
  merge_cost(index) = calculate_error(merge(Seg_TS(index), Seg_TS(index+1)));
  merge_cost(index-1) = calculate_error(merge(Seg_TS(index-1), Seg_TS(index)));
end;
```

Figure 5 The generic bottom-up algorithm [18]

2.3 Clustering techniques

Clustering is one of data mining techniques that can group different data objects into several clusters based on similarity. The data objects with high similarity are grouped into one cluster and data objects with high dissimilarity are in the different clusters. The knowledge of groups' definitions is not needed in advance. [23] Clustering is an unsupervised learning of a hidden data concept and a common data mining technique for statistical data analysis used in many fields including data mining, machine learning, pattern recognition and image analysis.

Most of the clustering algorithms can be six categories (figure 6): partitioning-based clustering, hierarchical-based clustering, grid-based clustering, model-based clustering, density-based clustering and multi-step clustering. [23]

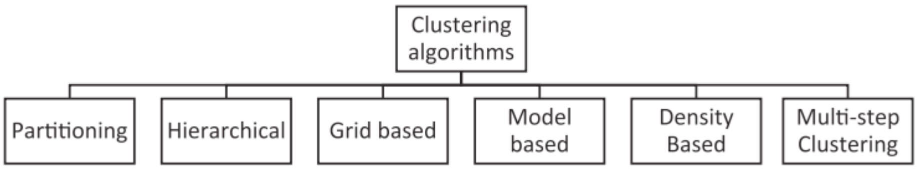


Figure 6 Clustering algorithms categories [23]

Partitioning-based clustering method first creates an initial set of k partitions and then uses iterative relocation approach to make k groups from n unlabelled objects by moving objects from one group to another. The most commonly used algorithms of partitioning-based clustering are K-means, K-medoids and Fuzzy C-means. [23]

Hierarchical-based clustering method makes a hierarchical structure of clusters and sub-clusters using agglomerative or divisive algorithms. Agglomerative algorithm considers each item of data as a cluster and then gradually merges the clusters which is a bottom-up approach. While divisive algorithm starts with all objects as a single cluster and then splits the cluster to reach the clusters with one object. Such a divisive algorithm is also called top-down approach. [23] There are many hierarchical-based clustering methods, such as BIRCH (will be talked about in the next section) which is a classical hierarchical-based clustering algorithm that enhances the merge approach and refines the constructed clusters.

Grid-based clustering method quantizes the space into numbers of cells that form a grid. Then it performs clustering on the grid's cells. [23]

Model-based clustering method assumes a model for each cluster and then finds the best fit model of each cluster. This method usually recovers the model by clustering a set of data. [23]

Density-based clustering method is that clusters are subspaces of dense objects separated by subspaces in which objects have low density. [23]

Multi-step clustering method is produced to improve the quality of representation approaches, enhance the clustering algorithms and present new models. The multi-step clustering method operates in several steps. [23]

For time series clustering techniques, they can be divided into several groups with different clustering approaches depending on whether they work with raw data, features extracted from the raw data or with models built from the raw data. [24] (see figure 7)

Raw data based approaches is to directly deal with the raw data (time series data) but with more suitable distance measures that are different with the distance measure for handling static data. Methods that deal with raw time series data, in the frequency or time domain, belong to raw data based approaches. [24]

Feature based approaches is to transform the time series data subsequences extracted by feature extraction into static data, and then deal with the static data with classic clustering algorithm for static data. Usually because raw data are noisy which is difficult to process. Different with raw data based approaches, it works in high-dimensional space, especially for data collected at first sampling rates. The number of feature dimensions received by feature extraction can be reduced by other feature selection approaches. [24]

Model based approaches is that considering each time series data is received by some kinds of model and every time series is similar. This type of approach is domain-dependent.

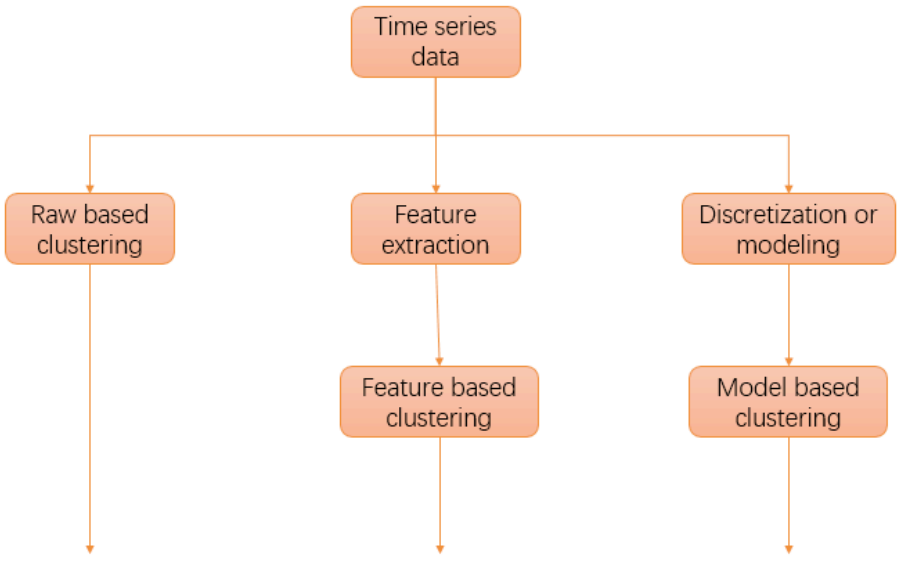


Figure 7 Clustering time series categories [24]

2.4 BIRCH algorithm

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [25] is a classical hierarchical-based clustering algorithm. During its working process, it continually takes a set of multi-dimensional data points as inserted data objects and generates a clustering feature (CF) tree. This CF tree is one of the major components of BIRCH algorithm which indicates the position of each inserted data object and vein structure as shown in figure 8. CF is stored as a vector that contains three values: (N, LS, SS), of which N means the amount of data objects it encloses, LS means that linear sum of data objects and SS means the square sum of data objects. It has been proven by Zhang et.al that this CF vector is easily used to calculate Euclidean distance or other distances between different entries in BIRCH algorithm. [25]

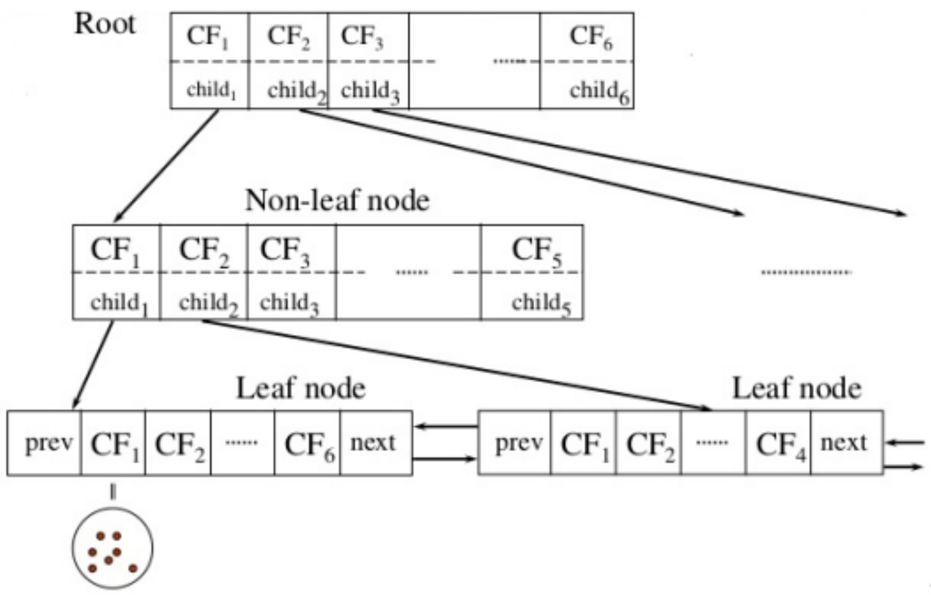


Figure 8 BIRCH CF tree structure [25]

Some parameters play key roles in BIRCH, like the inside diameter of a leaf node (T) or how many nodes a non-leaf node can have (B). By using parameters, the structure of tree can be controlled.

In general, BIRCH algorithm has four phases to do the whole clustering.

Phase one is scanning the original data set, inserting each data object and establishing a CF tree which is limited by available memory and time constraints. When it is inserting a data object, firstly, it recurs down from the root node and find the appropriate leaf node; secondly, if the closest leaf node could absorb the data object, a new CF entry or leaf node will be produced and if there is no room for this new entry to be established, the parent node will be split; thirdly, this operation will be updated on the path back to root node. [25]

Phase two can rebuild the CF tree several times which will generate a smaller tree later. To be concrete, the leaf entries in the initial CF tree will be scanned and a smaller CF tree will be rebuilt by clustering crowded subclusters into a larger one and removing outliers. After this rebuilding, a larger threshold T would be generated. This phase is optional. [25]

Phase three, global or semi-global clustering algorithm would be used to cluster all the leaf entries. Also, the existing clustering algorithms for a set of data points can be readily adapted to work with a set of sub-clusters, each described by its CF vector. After phase 3, we can get a set of clusters that capture the patterns in the data. [25]

Phase four can do the cluster refining and produce better clusters by using the centroids of the clusters produced by phase 3 as seeds and redistributing the data points to its closest seed to obtain a set of new clusters. This phase is also optional. [25]

After above four phases, a CF tree can be printed and the sub-clusters can be clearly received as child nodes in the CF tree.

2.5 Evaluations of clustering algorithms

As mentioned in the 2.3, there are many clustering methods including K-means, hierarchical clustering (HC), PAM. Yet there is no comprehensive evaluation study to evaluate the accuracy and effectiveness of above methods. [26]

Evaluating clustering algorithms is an important step to determine an optimal clustering solution or cluster structure for the given dataset. This step is always operated after getting clustering results. It has an important effort that selecting the best one from all candidate clustering algorithms or making a performance comparison among different clustering algorithms. The choice of suitable evaluation measures for evaluating clustering algorithms depends on the clustering objects and clustering tasks. However, there are no absolute schemes with which to evaluate clustering algorithms. [27, 28]

External indices (e.g. adjusted Rand index, Fowlkes-Mallows (FM) index) and internal indices (e.g. Weighted inter-intra index, Silhouette Coefficient, Calinski-Harabax index) are two validity indices as evaluation solutions. In respect of external indices, the clustering results are evaluated according to a known cluster structure of cluster labels in a dataset. In respect of internal indices, the inherent quantities and features in the dataset are used to evaluate the clustering results. [27, 29] In respect of relative indices, the best clustering scheme of a set of defined schemes is chosen based on a pre-specified criterion. [29] In most cases, unsupervised learning works with unlabeled data while supervised learning works with labelled data. The internal indices would thus be needed if there are no labelled data.

Evaluation metrics for clustering algorithms mainly aim at three categories: algorithm performance, cluster quality/the number of clusters, running time and memory usage. [14, 30]

Algorithm performance is usually evaluated by measurements as following aspects. The first one is test environment and datasets. Different data mining tools have different test environment which may affect the mining performance. Also, the use of various datasets may affect the mining performance. The features of data depend on the datasets. The second one is scalability test. The algorithm performance can be evaluated by testing the scalability and checking the relationship between scalability and other criteria (e.g. running time). [14]

Cluster quality/the number of clusters can be evaluated by validity indices. As introduced before, there are two categories of validity indices and each includes a list of specific methods. The choice of evaluation metrics depends on the clustering algorithm. [14, 30]

Running time and memory usage can also be used to get the time and memory consumed. Different algorithms can thus be compared with respect the running time and memory usage. Furthermore, the relationship between running time and other thresholds in the algorithm is easily presented. [14]

Above three aspects are most common evaluation metrics for evaluating an algorithm's performance.

2.6 Improved BIRCH algorithms overview

Although BIRCH algorithm is a hierarchical-based clustering algorithm suitable for clustering large datasets and it can minimize the memory and time, it has a few drawbacks. Some improved BIRCH algorithms aiming at dealing with those shortcomings that are proposed by other researchers would be talked about in the following.

A parallel version of BIRCH algorithm was proposed by Garg et al. in [31]. They proposed a scalable parallel clustering algorithm called PBIRCH for incremental data with the objective of enhancing the scalability without compromising on the quality of clustering. The incoming data is distributed in a cyclic manner to balance the load among processors. They implemented this optimized BIRCH algorithm on a message passing share-nothing model. In the conclusion part, the experiments showed that for very large data sets, the algorithm scales nearly linearly with the incoming number of clusters. Also, they compared the clusters obtained by PBIRCH to those obtained using BIRCH.

Prasad et al. in paper [32] introduced a privacy preserving BIRCH algorithm for clustering over vertically partitioned databased. One drawback of BIRCH algorithm the authors mentioned in paper is that it is basically designed as an algorithm for a single database. Thus, they proposed the first novel method for running BIRCH over vertically partitioned data sets, distributed in two different databases in a privacy preserving manner. In the conclusion part, they claimed that in entire new BIRCH algorithm clustering the database is scanned at most two times which has less I/O.

Du et al. proposed an improved BIRCH clustering algorithm and application in paper [33]. This improved BIRCH algorithm is used to resolving the problem of threshold value. Before that, they have discussed a algorithm of a kind of cluster optimizing BIRCH cluster's algorithm. The D-BIRCH can change threshold value in real time, operate the data and carry on better analysis of cluster to the big unit of electricity generation.

There is another proposed research called improved BIRCH clustering algorithm [34]. They also wanted to set the initial threshold value and found out how to increase the threshold value in the second stage of BIRCH. Thus, they changed CF structure and used a heuristic method. At last, the research showed that this improved algorithm has a better performance than original BIRCH algorithm.

Kovács et al. In paper [35] said that pre-clustering is a good method for data reduction facing with large datasets. An extended BIRCH per-clustering algorithm with optimized key parameters (e.g. threshold, branching factor) has been presented. It proved that it can improve the overall efficiency of the clustering system.

Even if an extended algorithm is implemented, it could not be ensured whether it is improved or not without performance evaluations. As talked, there are two kinds of validity indices including internal indices and external indices. Each index contains many specific approaches. Some related papers mainly talking about the performance evaluations of clustering algorithms are reviewed in this project.

In paper [28], a comparison among several cluster validity indices has been carried out based on five different real-life datasets. The validity indices are used to find the appropriate number of clusters. The authors presented validity index *I*, Dunn's index and the Xie Beni index.

In paper [30], which is a comparison of different procedures for determining the number of clusters (accuracy) in a dataset, they made some experiments to compare the performance of different validity indices. And totally thirty validity indices have been mentioned in this work.

3 Proposed methodology

Refer to the problem statement in the introduction, this project is about implementing the sliding window BIRCH algorithm on the basis of BIRCH algorithm. Then this work would be carried out through following two steps.

The first step is preparing data including data understanding and data normalization, implementing window BIRCH algorithm and running this algorithm by inserting matrix as time series data object that are extracted by sliding window. The clustering results can be received for further work.

The second step is that by using multiple validity indices as evaluation metrics to evaluate the results that received in the last step, then comparing the performance of window BIRCH algorithm with BIRCH algorithm and K-means algorithm.

Section 3.1 and 3.2 would be taken as two parts of the research to implement this project.

3.1 Model implementation

At the beginning of the data mining process, solving methods suitable for a list of tasks with which may meet during the research. In order to fulfil the requirements of different methods needed, reading papers and making reviews of literature study with related work is necessary.

To better mine data, data preprocessing including data understanding, data cleaning and data normalizing is required. In general, data cleaning may contain detecting missing values, correcting missing values, eliminate duplicated data and removing irregularities. Data normalization is used to reduce the redundancy of original data. The key point of time series data mining is to find hidden patterns or do the anomaly detection in large dataset. So data are dispensable. Moreover, high accuracy of algorithm results depends not only on the algorithm, but also on the parameters setting. In this project, after data cleaning and data normalizing, the threshold value as an input parameter of BIRCH clustering algorithms would be easily set from 0 to 1.

In order to fulfil the requirement of time series data mining, a sliding window model is designed and implemented to be added into BIRCH algorithm. It is used to extract matrix from multivariate time series and then this matrix would be inserted into CF tree as time series data object. In other words, several time series sub-sequences are generated to do BIRCH clustering. More concrete details of how to implement this sliding window model would be illustrated in the next chapter.

Similarity measure is an important part for clustering among data mining methods. In the BIRCH clustering algorithm, Euclidean measure is employed. To measure the similarity between different time series data objects and based on the sliding window model, the parameter used in Euclidean measure is modified from vector to matrix.

3.2 Evaluation metrics

Based on the results of previous step, the BIRCH algorithm with sliding window is designed and implemented. The evaluation methods would be carried out in the following.

Evaluation of clustering result is referred to as clustering validation. After getting results of algorithm, we need to evaluate the performance. To fulfil the requirement of clustering evaluation, running time and accuracy are chosen as two evaluation metrics in this project. The efficiency of algorithm can be checked by running time. The accuracy of algorithm can be checked by validity indices.

When it comes to evaluate accuracy, there are two categories: external indices and internal indices. In the BIRCH algorithm with sliding window, it is difficult to know the ground truth classes. According to the features of datasets used in this work, there are no labeled data. Thus, to satisfy that requirements, internal indices are employed in the evaluation part.

In order to do the clustering evaluation, several internal indices are applied in this project. Silhouette Coefficient is applied as the first metrics, assuming that a is the mean distance between a sample and all other points in the same class, b is the mean distance between a sample and all other points in the next nearest cluster, then the Silhouette Coefficient is given by

$$s = (b-a) / \max (a, b).$$

If s value is higher, then the accuracy performance of clustering would be better.

The second metrics is Calinski-Harabaz index. Assuming that k is the number of clusters, N is the number of points in dataset, Bk is the between cluster group dispersion matrix and Wk is the within-cluster dispersion matrix, then the Calinski-Harabaz is defined by

$$s(k) = (Tr(Bk) / Tr(Wk) * (N-k)/(k-1).$$

A higher s value, a better clustering accuracy.

Generally, in order to evaluate this work, the last step of completing this work is to present performance comparisons among window BIRCH, BIRCH and K-means.

4 Implementation

The main task of this study is to establish a clustering time series system which can process univariate time series data and multivariate time series data, thus a sliding window BIRCH clustering algorithm is implemented and the performance of this algorithm is evaluated. Figure 9 illustrates a general structure of this project, from start, data preprocessing including data cleaning and data normalizing, to sliding window BIRCH implementation including sliding window and BIRCH clustering algorithm, comparisons and evaluations. More concrete details of designing and implementing of each part would be described below.

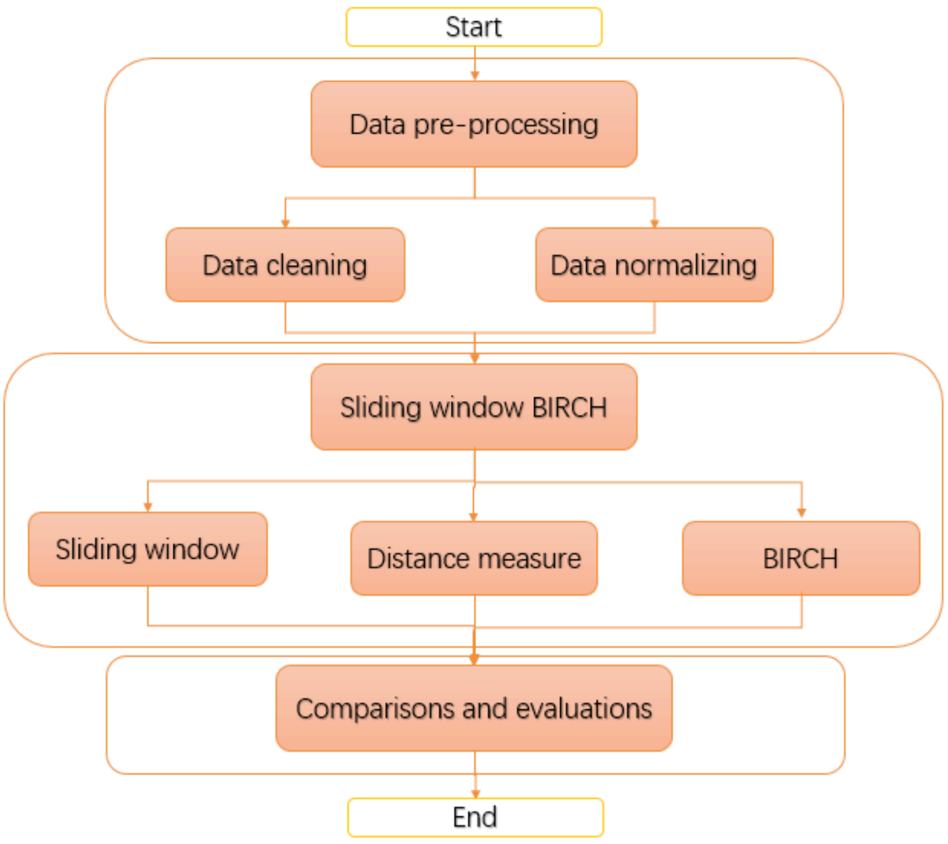
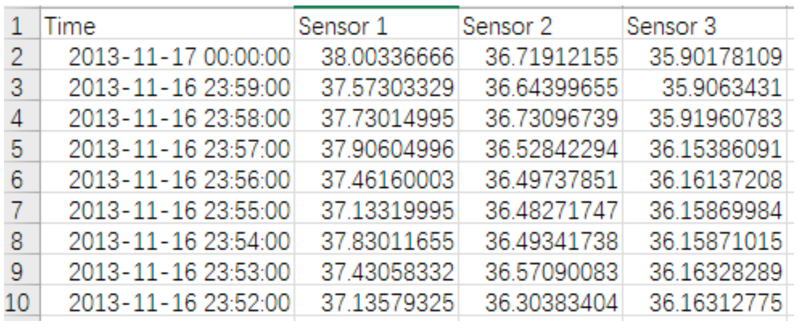


Figure 9 A general structure of this study

The implementation of this project is accomplished with Python v2.7.

4.1 Description of datasets

Due to the requirement of time series data mining, the datasets used for testing and evaluating in this study are all multivariate time series data, like figure 10 shows a part of KL20 dataset.



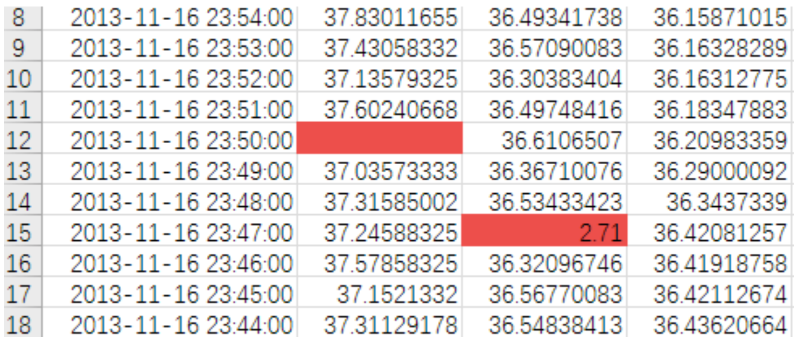
1	Time	Sensor 1	Sensor 2	Sensor 3
2	2013-11-17 00:00:00	38.00336666	36.71912155	35.90178109
3	2013-11-16 23:59:00	37.57303329	36.64399655	35.9063431
4	2013-11-16 23:58:00	37.73014995	36.73096739	35.91960783
5	2013-11-16 23:57:00	37.90604996	36.52842294	36.15386091
6	2013-11-16 23:56:00	37.46160003	36.49737851	36.16137208
7	2013-11-16 23:55:00	37.13319995	36.48271747	36.15869984
8	2013-11-16 23:54:00	37.83011655	36.49341738	36.15871015
9	2013-11-16 23:53:00	37.43058332	36.57090083	36.16328289
10	2013-11-16 23:52:00	37.13579325	36.30383404	36.16312775

Figure 10 Multivariate time series data

Time series data are changeable with time, in figure 10, the time interval between two entries is one minute. The sensors' data are dynamically changed every minute. The number of attributes could be two, three and above. Usually, time series data have large size and large scale. The time series datasets used in this study may have over thousand lines and over hundred attributes.

4.2 Data preprocessing

If there are much irrelevant or useless data in datasets, the working procedure of time series data mining would become more difficult. Irrelevant data may include missing values, improper values. A preparation for data to make data more clear or tidy is necessary. Thus, data preprocessing, which is an important step in the data mining process, can have a significant impact on performance of an unsupervised or supervised algorithm. [36]



8	2013-11-16 23:54:00	37.83011655	36.49341738	36.15871015
9	2013-11-16 23:53:00	37.43058332	36.57090083	36.16328289
10	2013-11-16 23:52:00	37.13579325	36.30383404	36.16312775
11	2013-11-16 23:51:00	37.60240668	36.49748416	36.18347883
12	2013-11-16 23:50:00		36.6106507	36.20983359
13	2013-11-16 23:49:00	37.03573333	36.36710076	36.29000092
14	2013-11-16 23:48:00	37.31585002	36.53433423	36.3437339
15	2013-11-16 23:47:00	37.24588325	2.71	36.42081257
16	2013-11-16 23:46:00	37.57858325	36.32096746	36.41918758
17	2013-11-16 23:45:00	37.1521332	36.56770083	36.42112674
18	2013-11-16 23:44:00	37.31129178	36.54838413	36.43620664

Figure 11 Example of irrelevant data in datasets

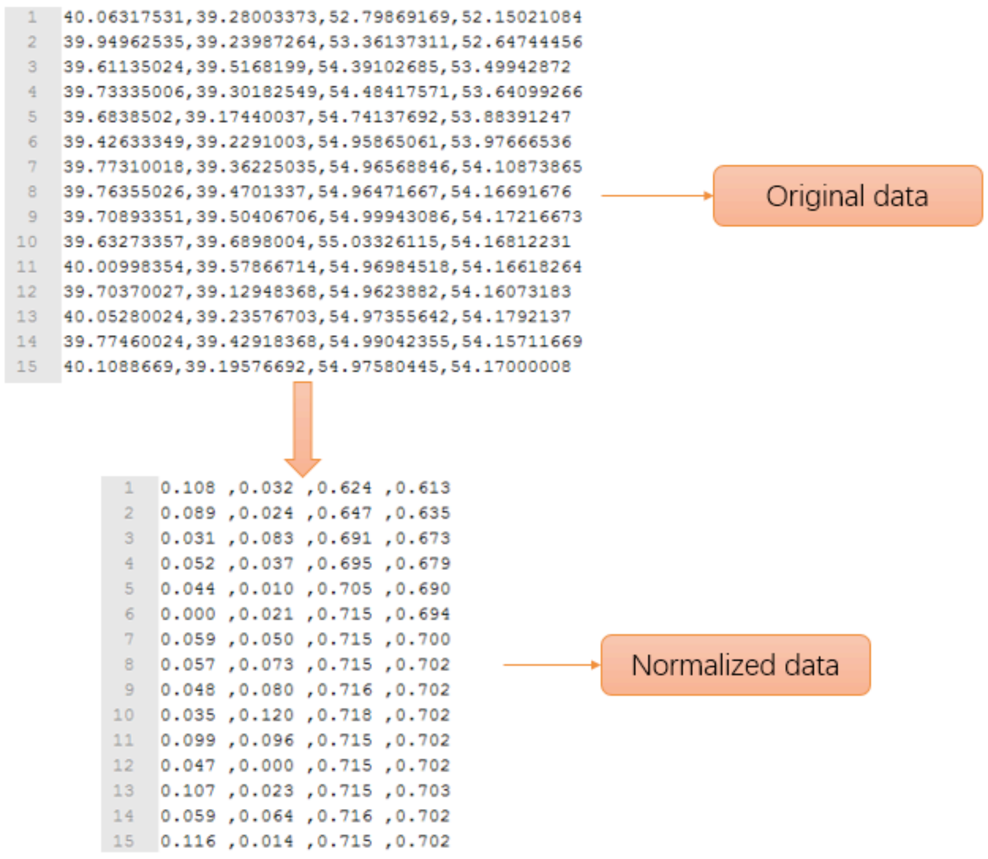
Data cleaning, data normalization, transformation, feature extraction and selection are usually included in data preprocessing. In this study, the data preprocessing step, including data cleaning and data normalization, is applied before achieving the core algorithm.

Firstly, data cleaning is used to remove missing values, incomplete data and improper data, for example, we modify the values which are missing and improper data in red cell (figure 11).

Secondly, there may be a large difference between the maximum value and minimum value in a real dataset, for example, 0.01 and 1000. The value range is needed to be scaled down. There are two most common methods are referred to as min-max normalization and z-score normalization. Due to the requirement that setting threshold value as input data in sliding window BIRCH algorithm, min-max normalization is applied in this project in order to transform all data in a dataset to a specific range that normalizing data on a scale of 0 to 1. Below is the min-max normalization formula

$$z_i = \frac{X_i - \min(X)}{\max(X) - \min(X)}$$

where $x_i = (x_1, ..., x_n)$ is the ith original data and $z_i = (z_1, ..., z_n)$ is the ith normalized data. Figure 12 shows an example of data after min-max normalizing. It can be seen from the figure that normalized data are all in range 0 to 1.



```
40.06317531,39.28003373,52.79869169,52.15021084
39.94962535,39.23987264,53.36137311,52.64744456
39.61135024,39.5168199,54.39102685,53.49942872
39.73335006,39.30182549,54.48417571,53.64099266
39.6838502,39.17440037,54.74137692,53.88391247
39.42633349,39.2291003,54.95865061,53.97666536
39.77310018,39.36225035,54.96568846,54.10873865
39.76355026,39.4701337,54.96471667,54.16691676
39.70893351,39.50406706,54.99943086,54.17216673
39.63273357,39.6898004,55.03326115,54.16812231
40.00998354,39.57866714,54.96984518,54.16618264
39.70370027,39.12948368,54.9623882,54.16073183
40.05280024,39.23576703,54.97355642,54.1792137
39.77460024,39.42918368,54.99042355,54.15711669
40.1088669,39.19576692,54.97580445,54.17000008
```

```
0.108 ,0.032 ,0.624 ,0.613
0.089 ,0.024 ,0.647 ,0.635
0.031 ,0.083 ,0.691 ,0.673
0.052 ,0.037 ,0.695 ,0.679
0.044 ,0.010 ,0.705 ,0.690
0.000 ,0.021 ,0.715 ,0.694
0.059 ,0.050 ,0.715 ,0.700
0.057 ,0.073 ,0.715 ,0.702
0.048 ,0.080 ,0.716 ,0.702
0.035 ,0.120 ,0.718 ,0.702
0.099 ,0.096 ,0.715 ,0.702
0.047 ,0.000 ,0.715 ,0.702
0.107 ,0.023 ,0.715 ,0.703
0.059 ,0.064 ,0.716 ,0.702
0.116 ,0.014 ,0.715 ,0.702
```

Figure 12 Example of data after normalizing

Now, this normalized data could be used.

4.3 Sliding window model

As mentioned in chapter 2, sliding window, which is a Piecewise Linear Representation (PLR) method, can be used to segment long time series data sequence. Time series sub-sequences extracted by sliding window from time series would be analysed by clustering or classification algorithms. It can be applied not only in univariate time series data, but also in multivariate time series data as figure 13 shows.

Figure 13 presents sliding window over KL20 dataset. One colourful line represents a sub-sequence of one attribute in a time series dataset. After an interval of time that is same as the width of sliding window, the x-axis (stands for time) would be changed to next time interval, this means that the sliding window would be moved to next time interval.

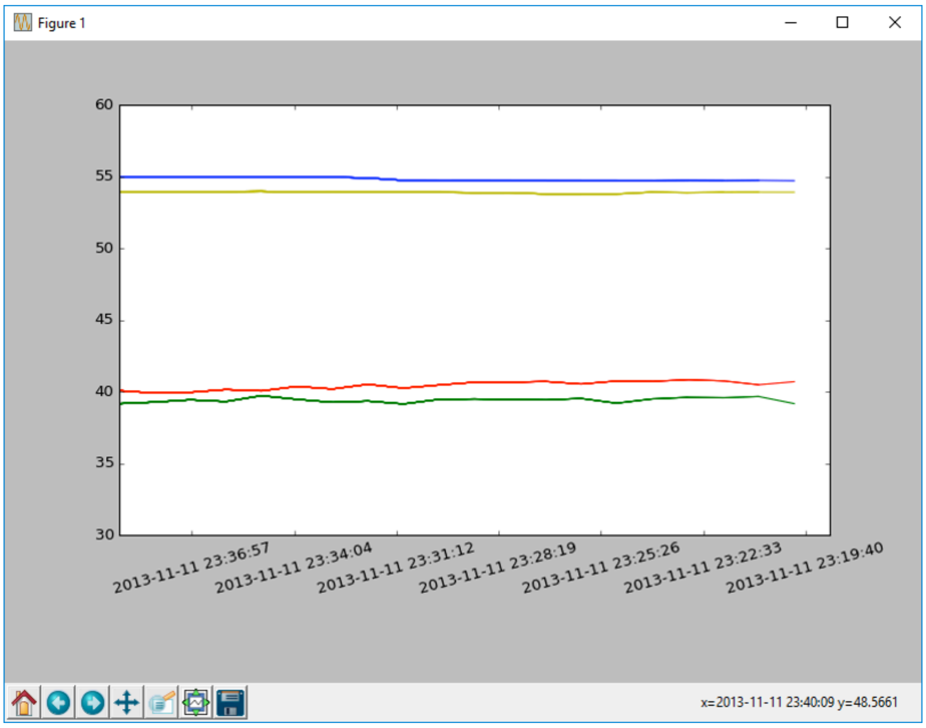


Figure 13 A sliding window over multivariate time series data

Below is a part of sliding window programming.

```
if counter%show_lim == 0:
    #show only the current window
    ax1=plt.axis([time[counter], time[counter+show_lim], y_min, y_max])
```

There are two parameters that play key roles, counter and show_lim. In every loop, counter is added by one. If counter is divided by show_lim and the remainder is 0, the x-axis value will be refreshed from time[counter] to time[counter+show_lim]. The time variable is declared as an array.

4.4 Sliding window BIRCH

The input data in BIRCH clustering algorithm is an entry in datasets. It can deal with univariate (one attribute) or multivariate (several attributes) data. If a sliding window is added in the clustering procedure, the input data object then could be either a vector or a matrix (figure 14) which depends on the width of sliding window. The sliding window BIRCH algorithm could have results which are more accurate and efficient than those in BIRCH algorithm.

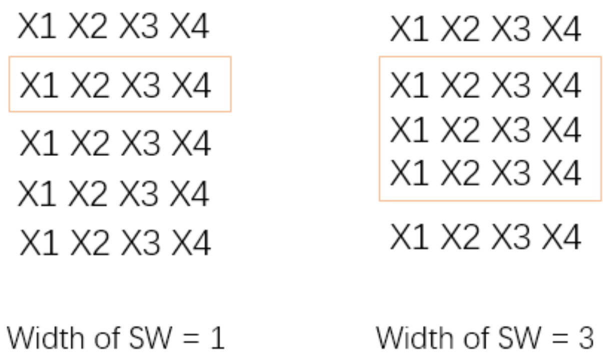


Figure 14 Different sizes of sliding window

4.4.1 Sliding window

A slidingWindowSize variable that can set the width of sliding window is added in the programming. The data type of inserted data object would thus be changed to matrix. The length of matrix would be based on slidingWindowSize variable, and the width of matrix would be based on how many attributes of dataset.

4.4.2 Distance measure

Similarity measure is an essential part in a clustering procedure. In BIRCH algorithm, Euclidean distance or Manhattan distance can both be used to measure the inter or intra distance between two clusters. Euclidean distance is calculated by

Euclidean distance = $[(x1-y1)^2 + (x2-y2)^2 + (x3-y3)^2 + \ldots + (xn-yn)^2]^{1/2}$

where x = (x1, x2, x3, …, xn) and y = (y1, y2, y3, …, yn) are member points. Below are the programming codes for calculating Euclidean distance.

```
# Euclidean distance computation
  def d0(self, e1, e2):
    dist=0
    #print "e1.sumX is %s" %e1.sumX
    #print "e2.sumX is %s" %e2.sumX
    for i in range(len(e1.sumX)):
      for j in range(len(e1.sumX[0])):
        diff=e1.sumX[i][j]/e1.n - e2.sumX[i][j]/e2.n
        dist += diff*diff
    if dist<0:
      print "d0<0!!!"
    return math.sqrt(dist)
```

Additionally, Manhattan distance measure can also be used instead of Euclidean distance in BIRCH algorithm, but Euclidean distance are set as default if the data are all quantitative.

4.4.3 BIRCH based on sliding window

As introduced in chapter 2, there are totally four phases. The first phase is scanning all data in dataset and build an initial CF tree using the limited memory. Additionally, the sub-clusters with crowded data points are set as find sub-clusters and the sparse data points are set as outliers. The task of the second phase is to build a smaller CF tree by scanning the leaf entries in the initial CF tree and then rebuilding a smaller tree. The outliers would be removed and crowded sub-clusters would be grouped into larger ones. The third phase is to cluster all leaf entries in CF tree by using a global or semi-global algorithm. And the last phase is to redistribute the data points produced by phase 3 to its closest seed to obtain a set of new clusters. Figure 15 shows a BIRCH procedure overview.

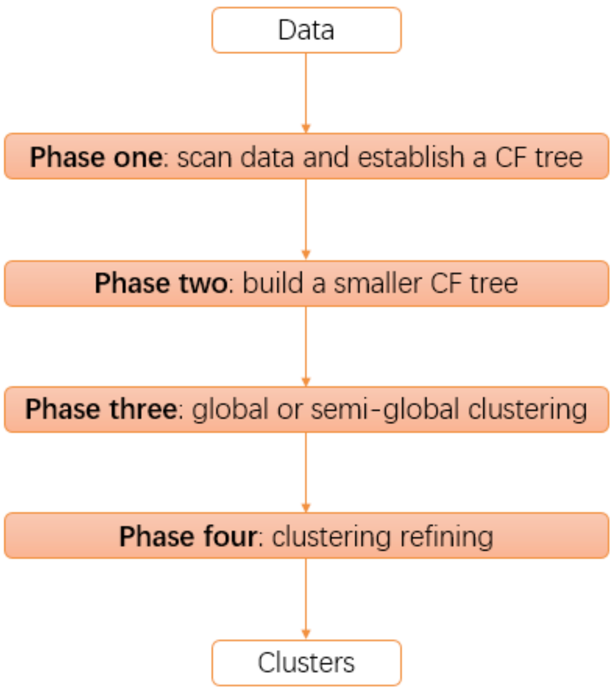

Data
Phase one: scan data and establish a CF tree
Phase two: build a smaller CF tree
Phase three: global or semi-global clustering
Phase four: clustering refining
Clusters

Figure 15 BIRCH procedure overview

Some parameters in this BIRCH algorithm are important as well. To be more specific, there is a maxNodeEntries variable that represents the maximum number of entries that a non-leaf node can have. The threshold variable represents the inside diameter of a leaf node. The setMemoryLimit variable is used to set the given memory.

Based on the requirement of this project, sliding window and updated Euclidean distance measure are all adapted in the BIRCH algorithm.

4.5 Summary

To summarize above, this chapter gives a complete implementation part from data preprocessing to core algorithm implementation. Based on the general structure of this time series clustering system as figure 9 presents, the whole implementation can be accomplished clearly. The first one is data preprocessing, it also shows the reason why min-max normalization method is applied. The second one is sliding window part which is added to make BIRCH algorithm more scalable and try to improve accuracy and efficiency. The third one is BIRCH implementation based on sliding window model. Each part is explained step by step.

5 Results and evaluation

This chapter is devoted to show the performance evaluation of sliding window BIRCH algorithm (SW BIRCH). It gives the results of various time series datasets running on SW BIRCH algorithm with respect to running time and number of clusters. Moreover, the accuracy and efficiency of SW BIRCH comparing with BIRCH and K-means would be presented.

The experiments in the following sections that have been taken for the evaluation part, KL20 dataset and several other datasets from UCR collection [37] are tested. Different dataset has different number of attributes and lines.

5.1 Results of sliding window BIRCH

In this section, a dataset called KL20 from company is used for testing the fundamental performance of SW BIRCH. Before testing, KL20 dataset has been normalized by min-max normalization. Below are table attributes in the experiments:

Algorithm: the name of algorithm.

SW size: the width of sliding window.

B: the maximum number of children of a non-leaf node.

Threshold: the diameter of a sub-cluster.

Datasets: the datasets used for testing.

Running time: the total execution time.

Number of clusters: how many clusters it generates.

The data in the following tables present the basic performance of SW BIRCH algorithm running on the KL20 dataset with 1440 lines and 4 attributes. B and threshold are set to different values.

Firstly, table 1 to table 4 show the results with different SW size based on different threshold (0.05, 0.1, 0.3, 0.6) respectively.

Algorithm	SW size	B	Threshold	Datasets	Running time	Number of clusters
SW BIRCH	1	100	0.05	KL20	2.843s	372
SW BIRCH	2	100	0.05	KL20	4.386s	506
SW BIRCH	4	100	0.05	KL20	5.753s	351
SW BIRCH	8	100	0.05	KL20	4.210s	180
SW BIRCH	10	100	0.05	KL20	4.671s	144

Table 1 Results for SW BIRCH (B=100, threshold=0.05) with different SW size on 1, 2, 4, 8, 10

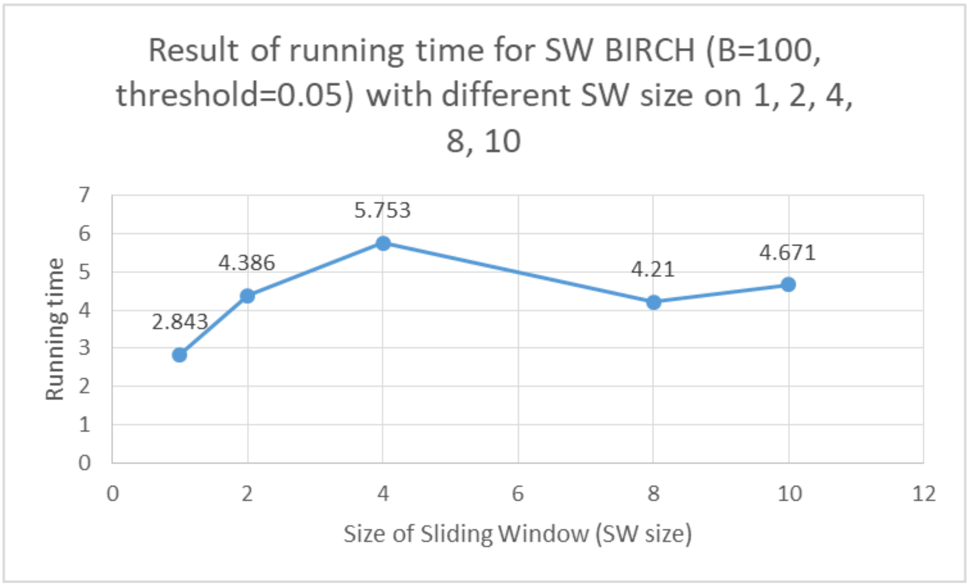


Figure 16 Results of running time for SW BIRCH (B=100, threshold=0.05) with different SW size on 1, 2, 4, 8, 10

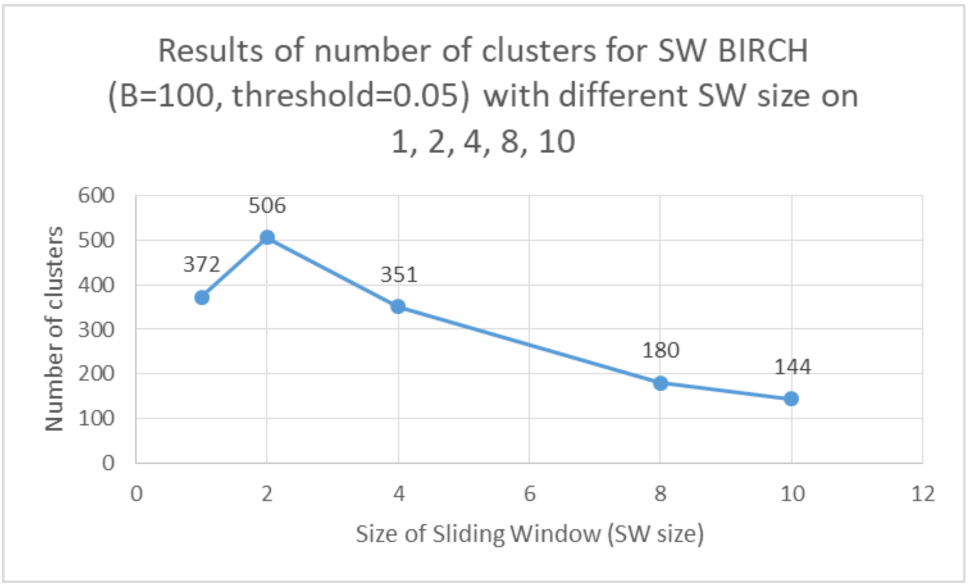


Figure 17 Results of number of clusters for SW BIRCH (B=100, threshold=0.05) with different SW size on 1, 2, 4, 8, 10

Table 1 shows the performance under the situation that threshold is set to 0.05. As SW size increases, the results are getting more and more overfitting according to a large number of clusters.

Algorithm	**SW size**	**B**	**Threshold**	**Datasets**	**Running time**	**Number of clusters**
SW BIRCH	1	100	0.1	KL20	1.579s	102
SW BIRCH	2	100	0.1	KL20	2.032s	213
SW BIRCH	4	100	0.1	KL20	2.985s	272
SW BIRCH	8	100	0.1	KL20	4.756s	177
SW BIRCH	10	100	0.1	KL20	3.892s	141

Table 2 Results for SW BIRCH (B=100, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

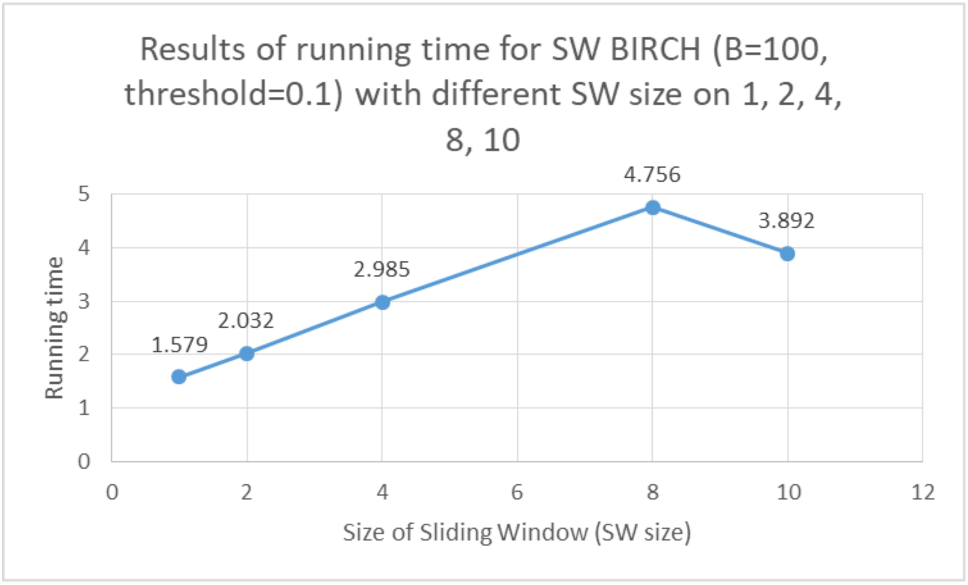


Figure 18 Result of running time for SW BIRCH (B=100, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

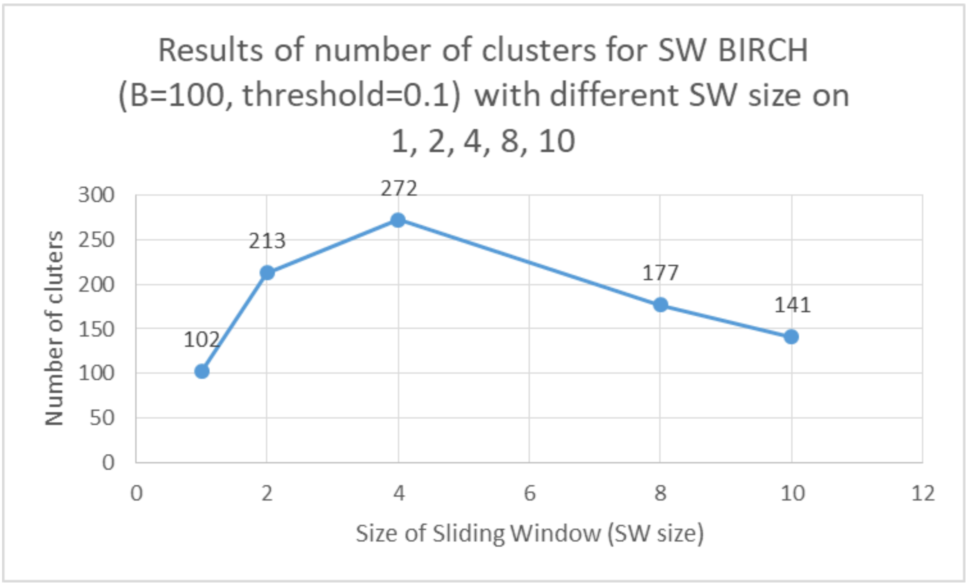


Figure 19 Result of number of clusters for SW BIRCH (B=100, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

Table 2 shows the performance under the situation that threshold is set to 0.1. If the SW size is relatively small, the results are better than in table 1. However, if the SW size is larger than 4, the results are still overfitting.

Algorithm	**SW size**	**B**	**Threshold**	**Datasets**	**Running time**	**Number of clusters**
SW BIRCH	1	100	0.3	KL20	0.420s	13
SW BIRCH	2	100	0.3	KL20	0.437s	21
SW BIRCH	4	100	0.3	KL20	0.616s	44
SW BIRCH	8	100	0.3	KL20	0.955s	72
SW BIRCH	10	100	0.3	KL20	1.122s	78

Table 3 Results for SW BIRCH (B=100, threshold=0.3) with different SW size on 1, 2, 4, 8, 10

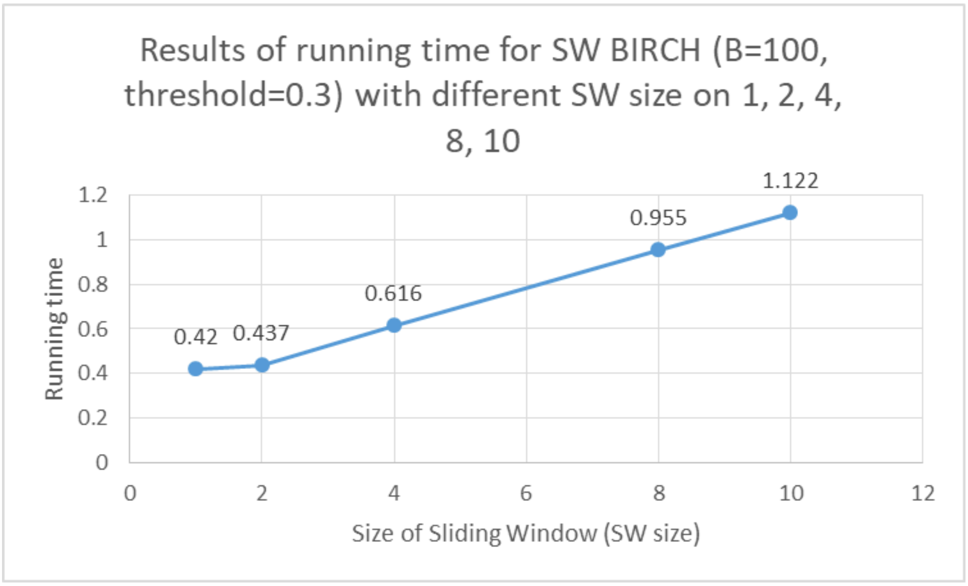


Figure 20 Results of running time for SW BIRCH (B=100, threshold=0.3) with different SW size on 1, 2, 4, 8, 10

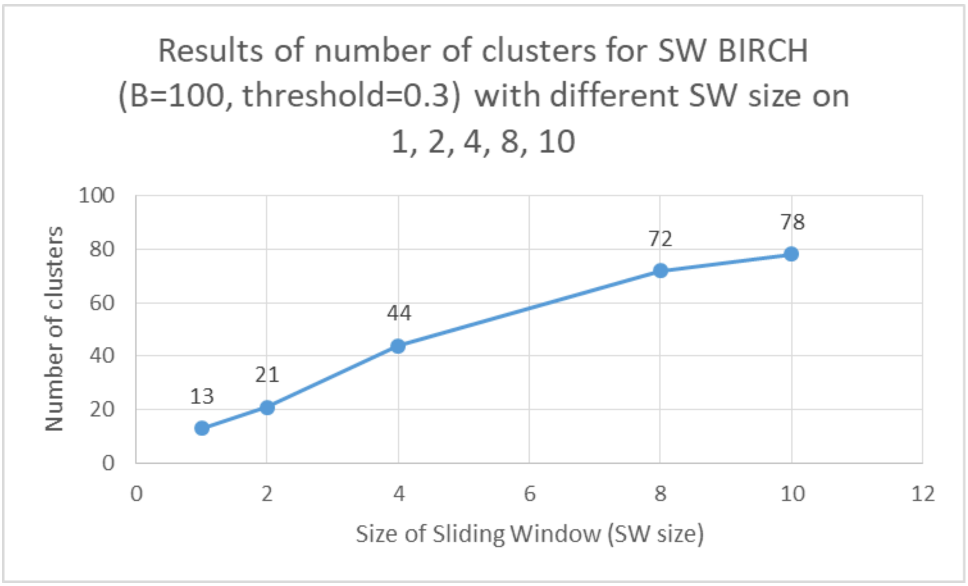


Figure 21 Results of number of clusters for SW BIRCH (B=100, threshold=0.3) with different SW size on 1, 2, 4, 8, 10

Algorithm	SW size	B	Threshold	Datasets	Running time	Number of clusters
SW BIRCH	1	100	0.6	KL20	0.343s	5
SW BIRCH	2	100	0.6	KL20	0.328s	8
SW BIRCH	4	100	0.6	KL20	0.281s	14
SW BIRCH	8	100	0.6	KL20	0.391s	24
SW BIRCH	10	100	0.6	KL20	0.510s	27

Table 4 Results for SW BIRCH (B=100, threshold=0.6) with different SW size on 1, 2, 4, 8, 10

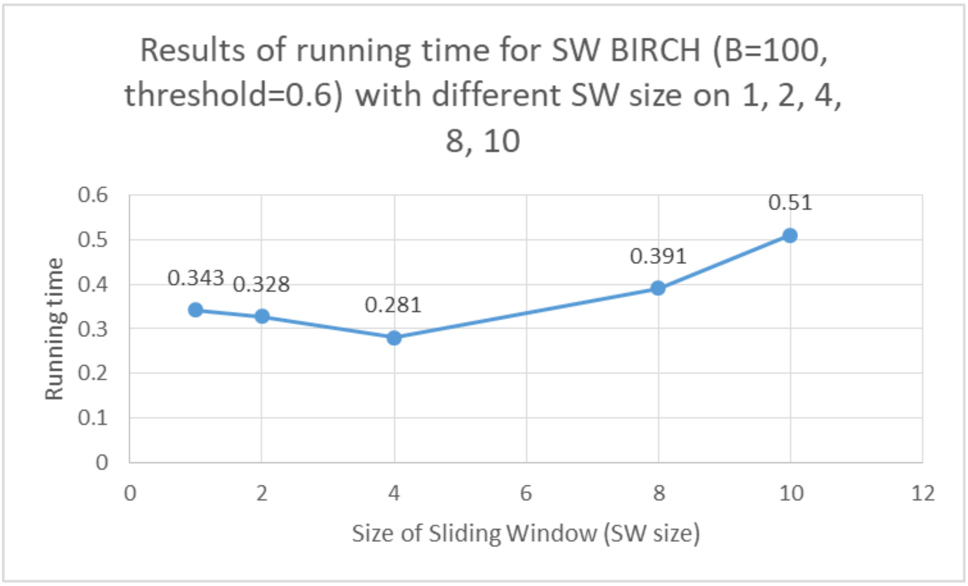


Figure 22 Results of running time for SW BIRCH (B=100, threshold=0.6) with different SW size on 1, 2, 4, 8, 10

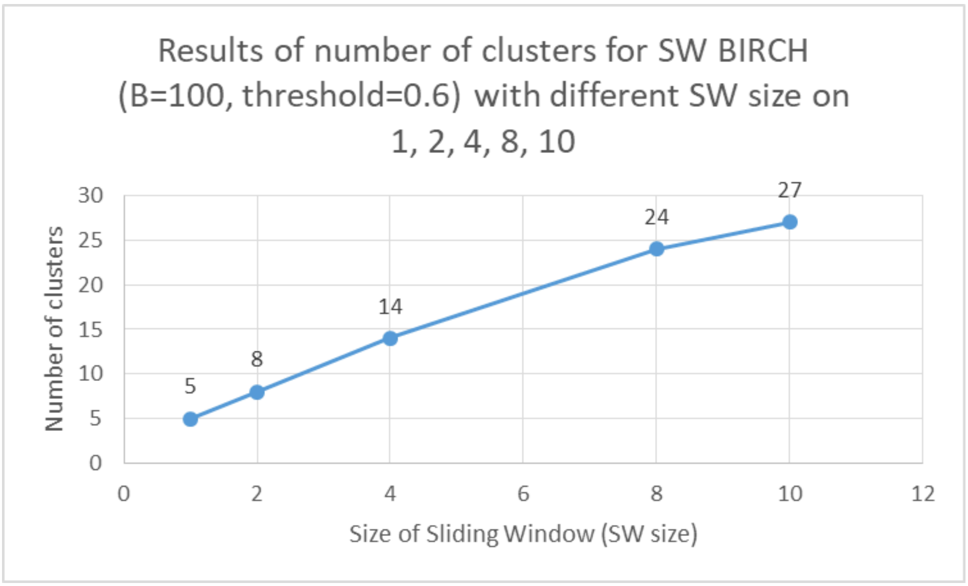


Figure 23 Results of number of clusters for SW BIRCH (B=100, threshold=0.6) with different SW size on 1, 2, 4, 8, 10

Above table 3 and table 4 show the performance with threshold is set to 0.3 and 0.6 respectively. As threshold value increases, the number of clusters decreases.

Secondly, table 5 to table 8 show results with different SW size based on different B values (50 to 300).

Algorithm	SW size	B	Threshold	Datasets	Running time	Number of clusters
SW BIRCH	1	50	0.1	KL20	1.160s	116
SW BIRCH	2	50	0.1	KL20	1.449s	219
SW BIRCH	4	50	0.1	KL20	3.029s	274
SW BIRCH	8	50	0.1	KL20	3.017s	177

SW BIRCH	10	50	0.1	KL20	2.868s	141

Table 5 Results for SW BIRCH (B=50, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

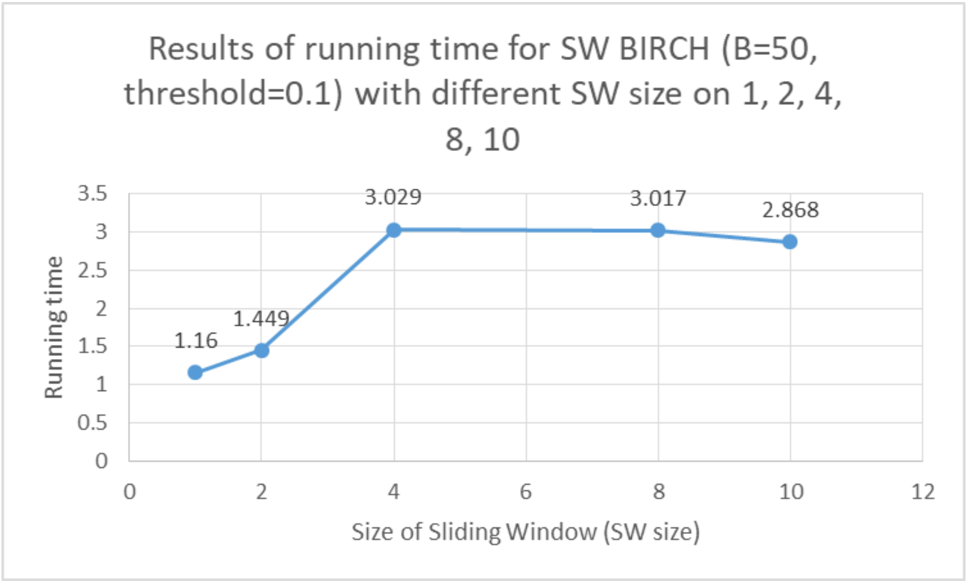


Figure 24 Results of running time for SW BIRCH (B=50, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

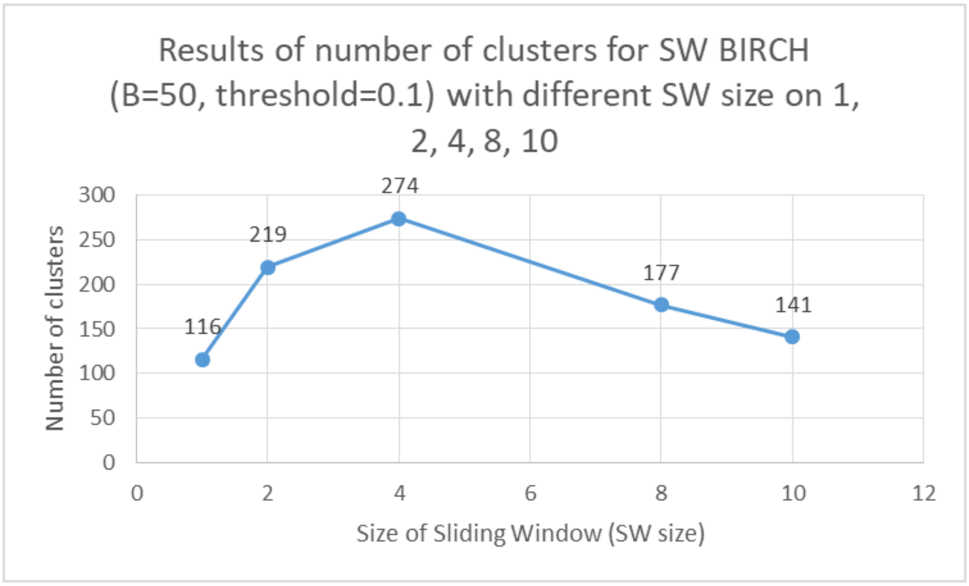


Figure 25 Results of number of clusters for SW BIRCH (B=50, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

Algorithm	SW size	B	Threshold	Datasets	Running time	Number of clusters
SW BIRCH	1	100	0.1	KL20	1.579s	102
SW BIRCH	2	100	0.1	KL20	2.032s	213
SW BIRCH	4	100	0.1	KL20	2.985s	272
SW BIRCH	8	100	0.1	KL20	4.756s	177
SW BIRCH	10	100	0.1	KL20	3.892s	141

Table 6 Results for SW BIRCH (B=100, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

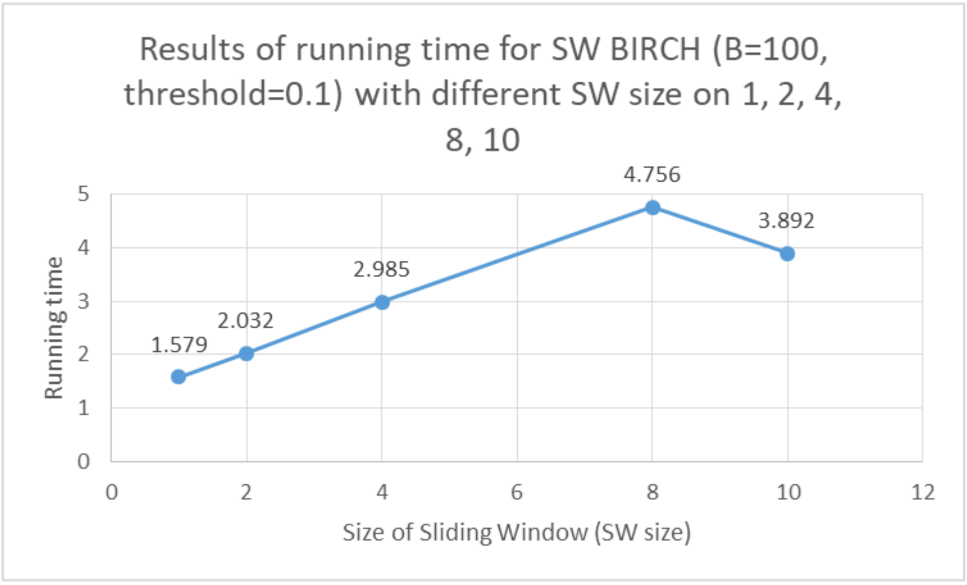


Figure 26 Results of running time for SW BIRCH (B=100, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

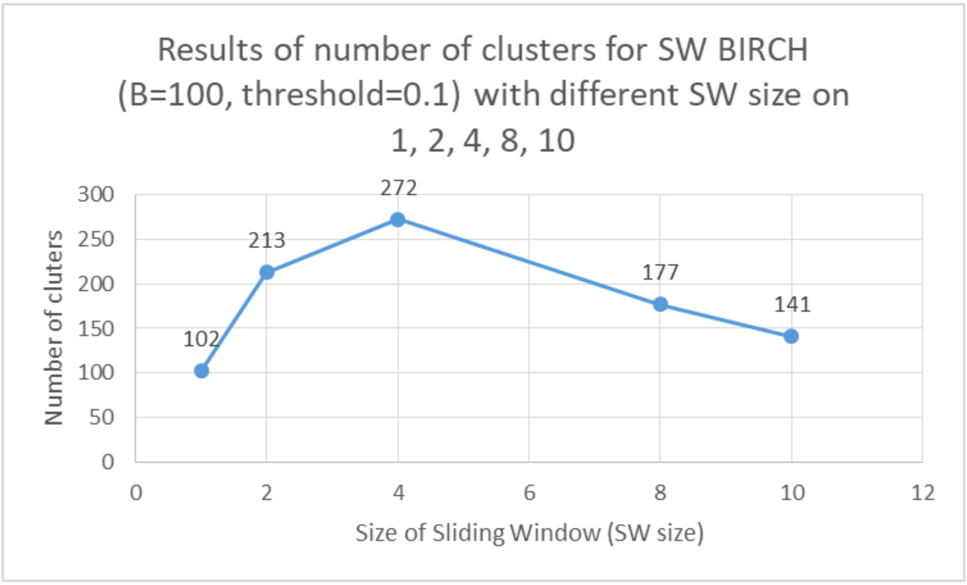


Figure 27 Results of number of clusters for SW BIRCH (B=100, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

Algorithm	SW size	B	Threshold	Datasets	Running time	Number of clusters
SW BIRCH	1	200	0.1	KL20	1.485s	101
SW BIRCH	2	200	0.1	KL20	2.992s	212
SW BIRCH	4	200	0.1	KL20	4.225s	272
SW BIRCH	8	200	0.1	KL20	4.240s	177
SW BIRCH	10	200	0.1	KL20	3.422s	141

Table 7 Results for SW BIRCH (B=200, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

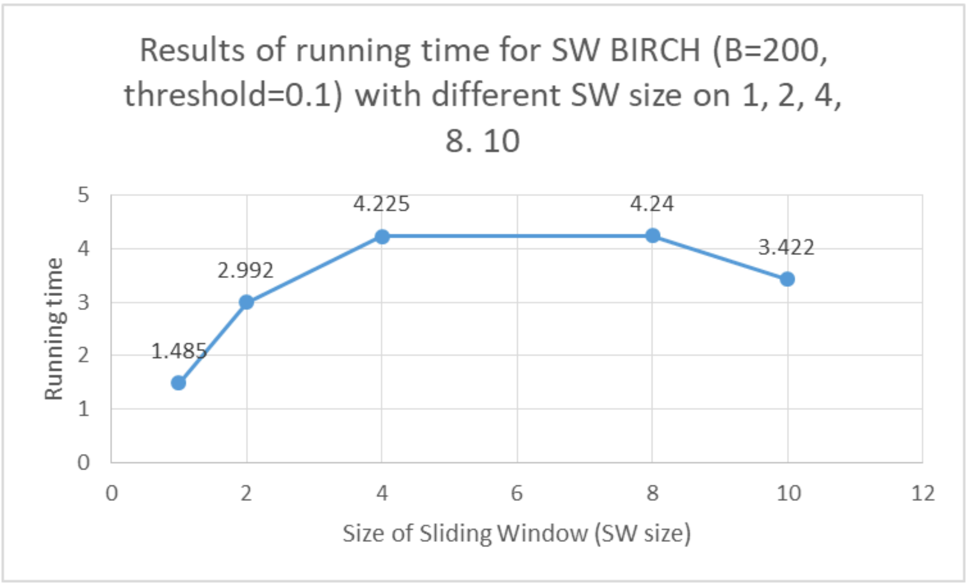


Figure 28 Results of running time for SW BIRCH (B=200, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

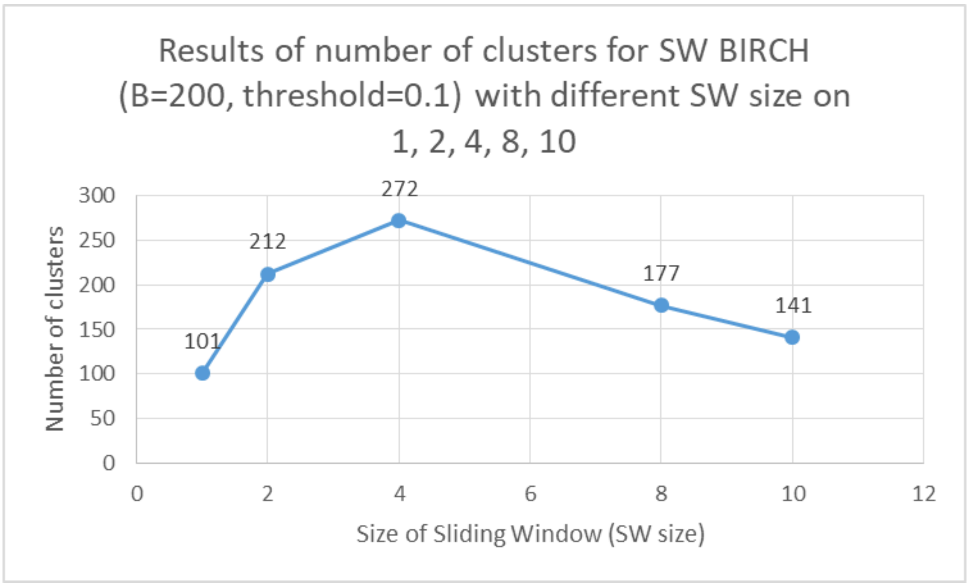


Figure 29 Results of number of clusters for SW BIRCH (B=200, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

Algorithm	**SW size**	**B**	**Threshold**	**Datasets**	**Running time**	**Number of clusters**
SW BIRCH	1	300	0.1	KL20	1.516s	101
SW BIRCH	2	300	0.1	KL20	2.985s	209
SW BIRCH	4	300	0.1	KL20	5.142s	272
SW BIRCH	8	300	0.1	KL20	4.024s	177
SW BIRCH	10	300	0.1	KL20	3.219s	141

Table 8 Results for SW BIRCH (B=300, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

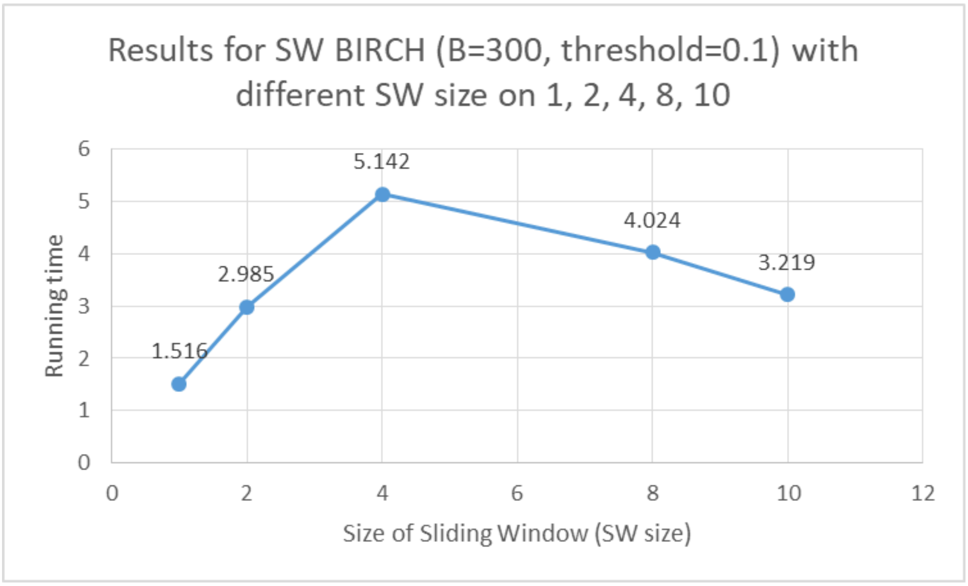


Figure 30 Results of running time for SW BIRCH (B=300, threshold=0.1) with different SW size on 1, 2, 4, 8, 1

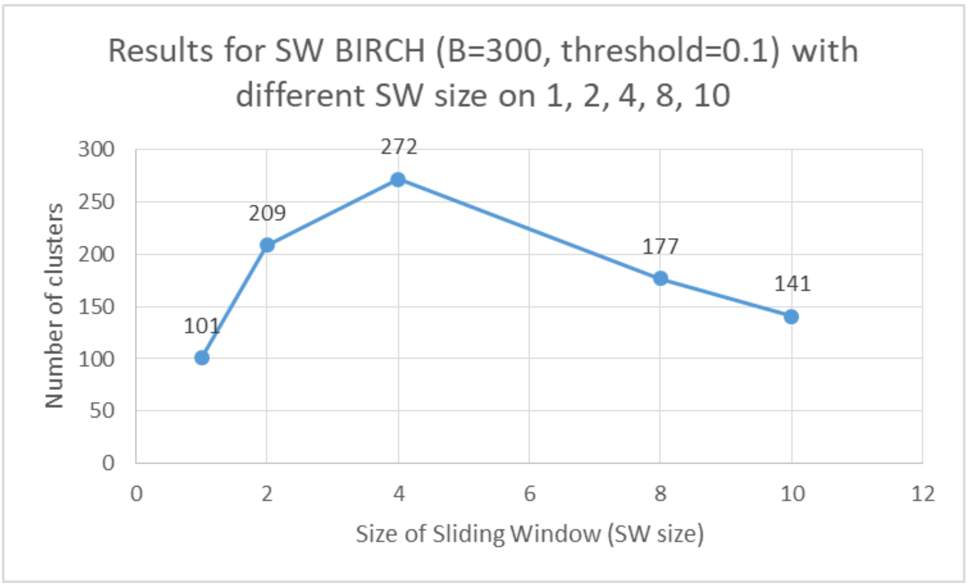


Figure 31 Results of number of clusters for SW BIRCH (B=300, threshold=0.1) with different SW size on 1, 2, 4, 8, 10

From above four tables, we can see that with same threshold value in different situations, when B increases, the number of clusters do not have any fluctuations. The reason is that above B values are all relatively large with respect to the number of lines and attributes of KL20 dataset. If B is small, more root splitting would be carried out.

The results in the following table present sliding window BIRCH algorithm running on six other datasets (50words, Coffee, Beef, Computers, ECG200, FordA) with different number of lines and different number of attributes. The datasets are all normalized before testing.

50words: 451 lines, 271 attributes.

Coffee: 29 lines, 287 attributes.

Beef: 31 lines, 471 attributes.

Computers: 251 lines, 721 attributes.

ECG200: 97 lines, 97 attributes.

FordA: 1321 lines, 501 attributes.

Algorithm	**SW size**	**B**	**Threshold**	**Datasets**	**Running time**	**Number of clusters**
SW BIRCH	2	50	0.05	50words	42s	225
SW BIRCH	2	50	0.3	50words	38.882s	225
SW BIRCH	2	200	0.05	50words	97.677s	225
SW BIRCH	2	200	0.3	50words	100.180s	225
SW BIRCH	2	5	0.05	Coffee	0.656s	14
SW BIRCH	2	5	0.3	Coffee	0.672s	14
SW BIRCH	2	15	0.05	Coffee	0.564s	14

SW BIRCH	2	15	0.3	Coffee	0.582s	14
SW BIRCH	2	5	0.05	Beef	0.978s	15
SW BIRCH	2	5	0.3	Beef	0.875s	15
SW BIRCH	2	15	0.05	Beef	0.889s	15
SW BIRCH	2	15	0.3	Beef	0.934s	15
SW BIRCH	2	30	0.05	Computers	36.822s	125
SW BIRCH	2	30	0.3	Computers	38.040s	125
SW BIRCH	2	100	0.05	Computers	78.215s	125
SW BIRCH	2	100	0.3	Computers	44.527s	125
SW BIRCH	2	10	0.05	ECG200	1.110s	50
SW BIRCH	2	10	0.3	ECG200	1.105s	50
SW BIRCH	2	50	0.05	ECG200	1.524s	50
SW BIRCH	2	50	0.3	ECG200	1.130s	50
SW BIRCH	2	100	0.05	FordA	462.386s	660
SW BIRCH	2	100	0.3	FordA	321.580s	660
SW BIRCH	2	500	0.05	FordA	1302.04s	660
SW BIRCH	2	500	0.3	FordA	1272.01s	660

Table 9 Results for SW BIRCH with other datasets (50words, Coffee, Beef, Computers, ECG200, FordA)

Table 9 shows the performance that SW BIRCH running on different datasets. For datasets, which have more lines, more running time would be needed on clustering data.

5.2 Algorithm evaluations and comparisons

In order to evaluate the performance of sliding window BIRCH algorithm (SW BIRCH), evaluation metrics is needed. Several internal indices are used to make comparisons among three clustering algorithms: SW BIRCH, BIRCH, K-means.

The first interval index is referred to as Silhouette Coefficient. The dataset is iris dataset with 150 lines and 4 attributes. B value and threshold value are set to 2 and 0.05 respectively.

Clustering algorithm	**Number of clusters**	**Dataset**	**Evaluation metrics**	**Metrics value**
BIRCH	2	Iris	Silhouette Coefficient	0.686
BIRCH	3	Iris	Silhouette Coefficient	0.553
BIRCH	4	Iris	Silhouette Coefficient	0.486
BIRCH	5	Iris	Silhouette Coefficient	0.483
BIRCH	6	Iris	Silhouette Coefficient	0.361

Table 10 Evaluation performance of BIRCH algorithm using Silhouette Coefficient internal index as evaluation metrics

Silhouette Coefficient is a measure to find the appropriate number of clusters in order to achieve higher accuracy. A silhouette coefficient close to 1 implies that the number of clusters is appropriate. We can see from table 10 that for BIRCH algorithm, the number of clusters equal to 2 has a relatively higher accuracy.

Clustering algorithm	**Dataset**	**Evaluation metrics**	**Metrics value**	**Execution time**
K-means	Iris	Silhouette Coefficient	0.371	0.260s
BIRCH	Iris	Silhouette Coefficient	0.361	1.689s
SW BIRCH (SW size=3)	Iris	Silhouette Coefficient	0.279	0.623s

Table 11 Evaluation performance of three clustering algorithms (K-means, BIRCH, SW BIRCH) using Silhouette Coefficient internal index as evaluation metrics (number of clusters is 6)

The second interval index is referred to as Calinski-Harabaz index. The dataset is iris dataset with 150 lines and 4 attributes.

Clustering algorithm	**Dataset**	**Evaluation metrics**	**Metrics value**

K-means	Iris	Calinski-Harabaz index	474.517
BIRCH	Iris	Calinski-Harabaz index	464.755
SW BIRCH **(SW size=3)**	Iris	Calinski-Harabaz index	278.199

Table 12 Evaluation performance of three clustering algorithms (K-means, BIRCH, SW BIRCH) using Calinski-Harabaz index as evaluation metrics (number of clusters is 6)

5.3 Discussion

The results of section 5.1 suggest that using different values of B and threshold can affect the performance of SW BIRCH while increasing the size of SW. As described, it will be overfitting with constant B attribute while decreasing the value of threshold to a small number when SW size is set to 1. Even if the number of clusters decreases when the size of SW increases, it is still overfitting. The reason is that the number of entries depends on the size of SW. Also, the number of lines and attributes of datasets has an impact on the execution time.

Table 11 presents an evaluation performance comparison among K-means, BIRCH and SW BIRCH using Silhouette Coefficient internal index with respect to metrics value and execution time. In above experiment, K, B and threshold are set to 6, 2, 0.05 respectively. K-means has the highest metrics value and shortest execution time. The metrics value of SW BIRCH is lower than BIRCH, but the execution time of SW BIRCH is shorter than BIRCH. The reason behind this performance is the size of SW. As SW size increases, the number of entries as inserted data objects will decrease. Thus the execution time will decrease. But it will generate more clusters.

In order to verify the performance generated in table 11, another internal index has been used to do the evaluation. Table 12 shows a performance comparison using Calinski-Harabaz index with respect to metrics value alone.

For metrics value, from the highest to the lowest are K-means, BIRCH, SW BIRCH respectively.

Current study shows that the SW BIRCH algorithm performance depends not only on the threshold and B value, but also on the SW size. It needs more investigation to show more results. The SW BIRCH can generate less number of clusters and require less execution time comparing with BIRCH.

6 Conclusion and future work

This project aims to implementing a sliding window BIRCH algorithm on the basis of BIRCH algorithm for handling time series datasets. It also aims to demonstrate that if the evaluation performance of sliding window BIRCH algorithm is better than those of the existing clustering algorithms. There are three steps to accomplish the whole implementation: first, data preprocessing including data cleaning and data normalizing; second, implementing BIRCH algorithm with sliding window; third, performance evaluation by making comparisons among three different clustering algorithms. In general, above steps are all carried out successfully and the objects of this project have been achieved.

By using one of PLR methods which is sliding window, the type of inserted data objects used in BIRCH algorithm could be matrix or vector which depends on the size of sliding window. With results and performance evaluation, we can see that the efficiency of sliding window BIRCH algorithm has improved when comparing with BIRCH algorithm. However, the accuracy is not very satisfied or expected. Several internal indices are applied to evaluate SW BIRCH algorithm. The evaluation values are all lower than BIRCH algorithm.

There some new questions arisen due to the evaluation results. It is not a novel problem that it is difficult to ensure the threshold value in BIRCH algorithm as well as in SW BIRCH algorithm.

6.1 Ethics considerations

All the data or results that project realized are real. There are no fabricated data in the project. All the quoted viewpoints are mentioned in the reference list. This implementation is a master thesis project, not a practical or commercial product. The project is achieved only for the study research.

6.2 Future work

Based on the performance evaluation of algorithm, I would like to give some views on future work. Basically, time series data mining is not an unknown concept, but it is a comprehensive view of technologies that having a large impact on data mining. Since clustering time series data mining is playing an essential role in time series data mining area, the most necessary thing for researchers is to improve accuracy or efficiency of clustering time series data technologies.

Regarding this project, future work could be carried out in following aspects. Employing more different kinds of datasets to test the efficiency of SW BIRCH algorithm. This study did not investigate any evaluation metrics to evaluate the algorithm, other internal indices could be applied in future work. Additionally, this new algorithm could be compared with other clustering algorithms except for BIRCH and K-means.

References

[1] Liao, T. Warren. "Clustering of time series data—a survey." Pattern recognition 38.11 (2005): 1857-1874.

[2] Han, Jiawei, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011.

[3] Witten, Ian H., and Eibe Frank. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2005.

[4] Perner, Petra, and Azriel Rosenfeld. Machine learning and data mining in pattern recognition. Springer-Verlag Berlin Heidelberg, 2011.

[5] Karypis, George, Eui-Hong Han, and Vipin Kumar. "Chameleon: Hierarchical clustering using dynamic modeling." Computer 32.8 (1999): 68-75.

[6] Zhang, Tian, Raghu Ramakrishnan, and Miron Livny. "BIRCH: an efficient data clustering method for very large databases." ACM Sigmod Record. Vol. 25. No. 2. ACM, 1996.

[7] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium Mathematical Statist. Probability, vol. 1, 1967, pp. 281–297

[8] L. Kaufman, P.J. Rousseeuw, E. Corporation, Finding groups in data: an introduction to cluster analysis, vol. 39, Wiley Online Library, Hoboken, New Jersey, 1990.

[9] M. Ester, H.P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, In Kdd 96 (34) (1996, August) 226–231.

[10] W. Wang, J. Yang, R. Muntz, STING: a statistical information grid approach to spatial data mining, in: Proceedings of the International Conference on Very Large Data Bases, 1997, pp. 186–195.

[11] Du, HaiZhou, and YongBin Li. "An improved BIRCH clustering algorithm and application in thermal power." Web Information Systems and Mining (WISM), 2010 International Conference on. Vol. 1. IEEE, 2010.

[12] Esling, Philippe, and Carlos Agon. "Time-series data mining." ACM Computing Surveys (CSUR) 45.1 (2012): 12.

[13] Fu, Tak-chung. "A review on time series data mining." Engineering Applications of Artificial Intelligence 24.1 (2011): 164-181.

[14] Zolhavarieh, Seyedjamal, Saeed Aghabozorgi, and Ying Wah Teh. "A review of subsequence time series clustering." The Scientific World Journal 2014 (2014).

[15] Cassisi, Carmelo, et al. "Similarity measures and dimensionality reduction techniques for time series data mining." Advances in data mining knowledge discovery and applications (2012).

[16] Yang, Qiang, and Xindong Wu. "10 challenging problems in data mining research." International Journal of Information Technology & Decision Making 5.04 (2006): 597-604.

[17] Lin, Jessica, et al. "Pattern recognition in time series." Advances in Machine Learning and Data Mining for Astronomy 1 (2012): 617-645.

[18] Keogh, Eamonn, et al. "Segmenting time series: A survey and novel approach." Data mining in time series databases 57 (2004): 1-22.

[19] Aghabozorgi, Saeed, Ali Seyed Shirkhorshidi, and Teh Ying Wah. "Time-series clustering–A decade review." Information Systems 53 (2015): 16-38.

[20] Wei, Li, and Eamonn Keogh. "Semi-supervised time series classification." Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006.

[21] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." ACM computing surveys (CSUR) 41.3 (2009): 15.

[22] Keogh, Eamonn, and Jessica Lin. "Clustering of time-series subsequences is meaningless: implications for previous and future research." Knowledge and information systems 8.2 (2005): 154-177.

[23] Aghabozorgi, Saeed, Ali Seyed Shirkhorshidi, and Teh Ying Wah. "Time-series clustering–A decade review." Information Systems 53 (2015): 16-38.

[24] Liao, T. Warren. "Clustering of time series data—a survey." Pattern recognition 38.11 (2005): 1857-1874.

[25] Zhang, Tian, Raghu Ramakrishnan, and Miron Livny. "BIRCH: an efficient data clustering method for very large databases." ACM Sigmod Record. Vol. 25. No. 2. ACM, 1996.

[26] Thalamuthu, Anbupalam, et al. "Evaluation and comparison of gene clustering methods in microarray analysis." Bioinformatics 22.19 (2006): 2405-2412.

[27] Wang, Kaijun, Baijie Wang, and Liuqing Peng. "CVAP: validation for cluster analyses." Data Science Journal 8 (2009): 88-93.

[28] Maulik, Ujjwal, and Sanghamitra Bandyopadhyay. "Performance evaluation of some clustering algorithms and validity indices." IEEE Transactions on Pattern Analysis and Machine Intelligence 24.12 (2002): 1650-1654.

[29] Halkidi, Maria, Yannis Batistakis, and Michalis Vazirgiannis. "On clustering validation techniques." Journal of intelligent information systems 17.2-3 (2001): 107-145.

[30] Milligan, Glenn W., and Martha C. Cooper. "An examination of procedures for determining the number of clusters in a data set." Psychometrika 50.2 (1985): 159-179.

[31] Garg, Ashwani, et al. "PBIRCH: a scalable parallel clustering algorithm for incremental data." 2006 10th International Database Engineering and Applications Symposium (IDEAS'06). IEEE, 2006.

[32] Prasad, P. Krishna, and C. Pandu Rangan. "Privacy preserving BIRCH algorithm for clustering over vertically partitioned databases." Workshop on Secure Data Management. Springer Berlin Heidelberg, 2006.

[33] Du, HaiZhou, and YongBin Li. "An improved BIRCH clustering algorithm and application in thermal power." Web Information Systems and Mining (WISM), 2010 International Conference on. Vol. 1. IEEE, 2010.

[34] JIANG, Shen-yi, and Xia Li. "Improved BIRCH clustering algorithm." Journal of Computer Applications 29.1 (2009): 293-296.

[35] Kovács, László, and László Bednarik. "Parameter optimization for BIRCH pre-clustering algorithm." Computational Intelligence and Informatics (CINTI), 2011 IEEE 12th International Symposium on. IEEE, 2011.

[36] Kotsiantis, S. B., D. Kanellopoulos, and P. E. Pintelas. "Data preprocessing for supervised leaning." International Journal of Computer Science 1.2 (2006): 111-117.

[37] The UCR Time Series Classification/Clustering Homepage.

http://www.cs.ucr.edu/~eamonn/time_series_data. Accessed: May 2014.

Appendix

Programming code for sliding window of this sliding window BIRCH study are presented here.

Program code for sliding window

```
import matplotlib

import matplotlib.pyplot as plt

import matplotlib.ticker as mticker

import matplotlib.dates as mdates

import numpy as np

from numpy import loadtxt

import time

#init

show_lim=20

counter=0

stats=[] #previous/current value

x=list()

y1=list()

y2=list()

y3=list()

y4=list()

#load data from file

date,td1,td2,td3,td4 = np.loadtxt('KL20.txt', unpack=True,
                    delimiter=',',

                    convert-
ers={0:mdates.strpdate2num('%Y%m%d%H%M%S')})

#prepare fiture

plt.ion()

fig=plt.figure(figsize=(10,7))

ax1=plt.axis([date[counter],date[counter+show_lim],30,60])

ax2 = plt.subplot2grid((30,30), (0,0), rowspan=30, colspan=30)

while True:

    if counter%show_lim == 0:

        #show only the current window

        ax1=plt.axis([date[counter], date[counter+show_lim], 30, 60])

    #get new value from data stream

    temp_x=date[counter]

    temp_y1=td1[counter]

    #temp_x=date[counter]

    temp_y2=td2[counter]

    temp_y3=td3[counter]

    temp_y4=td4[counter]

    max_y1=max(td1)

    min_y1=min(td1)

    max_y2=max(td2)

    min_y2=min(td2)

    #stats[0]=stats[1]
    #stats[1]=mean(y)

    #show and update plot

    #add to window

    x.append(temp_x)

    y1.append(temp_y1)

    y2.append(temp_y2)

    y3.append(temp_y3)

    y4.append(temp_y4)

    #ax2.plot(date,td1)

    plt.plot(x,y1,'r')

    plt.plot(x,y2,'g')

    plt.plot(x,y3,'b')

    plt.plot(x,y4,'y')

    ax2.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-
  %d %H:%M:%S'))

    for label in ax2.xaxis.get_ticklabels():

        label.set_rotation(15)

    plt.gca().get_yaxis().get_major_formatter().set_useOffset(False)

    plt.draw()

    plt.subplots_adjust(bottom=.23)

    plt.show()

    plt.pause(0.0001)

    counter=counter+1

    if counter>60:
        break

    plt.close()
```