An overview of Data Stream Clustering using Partitional Approach

Project Report Submitted in Partial Fulfilment of the

Requirements for the degree of

Master of Computer Application

Of

Jadavpur University

May, 2018

By

SUMANTA ADHIKARY

Master of Computer Application – III

Examination Roll Number: MCA186022

Registration Number: 133685 of 2015 – 2016

Under the guidance of

Dr. Susmita Ghosh

Department of Computer Science and Engineering

Faculty of Engineering and Technology

Jadavpur University

Kolkata – 700032, India

2018

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

TO WHOM IT MAY CONCERN

I hereby forward the project report entitled ***"An overview of Data Stream Clustering using Partitional Approach"*** prepared by **Sumanta Adhikary** under my supervision to be accepted in partial fulfilment for the degree of **Master of Computer Application** in the Faculty and Technology of Jadavpur University, Kolkata.

_____

(Dr. Susmita Ghosh)
Project Supervisor
Dept. of Computer Science and Engineering
Jadavpur University
Kolkata – 700032

Countersigned:

_____

Prof. Ujjwal Maulik
Head, Dept. of Computer Science and Engineering
Jadavpur University
Kolkata – 700032

_____

Prof. Chiranjib Bhattacharjee
Dean, Faculty of Engineering and Technology
Jadavpur University
Kolkata – 70032

Department of Computer Science and Engineering
Faculty of Engineering and Technology
Jadavpur University

CERTIFICATE OF APPROVAL *

The foregoing project report is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to the degree for which it has been submitted. It is understood that, by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project report only for the purpose for which it has been submitted.

_____

Final Examination for
evaluation of the project

_____

(Signatures of Examiners)

* Only in case the project report is approved

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this project report contains literature survey and original research work by me, the undersigned candidate, as part of my Master of Computer Application studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

NAME	**:**	Sumanta Adhikary
Examination Roll Number	**:**	MCA186022
Registration Number	**:**	133685 of 2015 - 2016
Project Title	**:**	An overview of Data Stream Clustering using Partitional Approach
Signature with Date	**:**	

ACKNOWELDGEMENT

The satisfaction and euphoria that accompanies the successful completion of this task would be incomplete without the mention of the people who made it possible. Their constant guidance and encouragement crowned my effort with success.

It is a great pleasure to express my sincerest thanks to my project supervisor **Dr. Susmita Ghosh**, Department of Computer Science and Engineering, Faculty of Engineering and Technology, Jadavpur University, for her encouragement, valuable suggestion, and constant support during the course of this project.

I would like to thank all the professors of the Department of Computer Science and Engineering, Jadavpur University, Kolkata for the guidance they provided me throughout the duration of the Master of Computer Application course.

A special note of thanks goes to Prof. Ujjwal Maulik, Head, Department of Computer Science and Engineering, Jadavpur University.

I am also thankful to Prof. Chiranjib Bhattacharjee, Dean, Faculty of Engineering and Technology, for providing an excellent environment for completion of this project.

Sumanta Adhikary
Master of Computer Application – III
Examination Roll No. – MCA186022
Registration No: 133685 of 2015 - 2016

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Clustering is the process of keeping similar data together in same group and keeping different data in different group. Hence, give an idea about the data. Clustering is largely used in the fields of data mining, data analysis and pattern recognition. Sometimes the huge data comes in a non stationary form, called data stream. Data stream is a source of unbounded data that comes in an online fashion. Data stream clustering is basically an unsupervised learning. No prior information about the upcoming data is available. We can not use any type of iteration here, because, when the online data stream flows, we do not get any chance for the modification of the previously determined clusters. Therefore, we need streaming algorithms for this type of data set.

1.1 Clustering Approaches

In this paper we discuss about some partition based algorithms for Data Stream Clustering. Most widely used partitioning approach algorithm is K-Means clustering algorithm. It is actually Lloyd's algorithm [1,14]. For general data set clustering, this algorithm goes through a no of iterations to achieve the current optimization in each step and finally reach to the solution. We use this algorithm for our data stream clustering, keeping the procedure same and just leaving the iterative steps. In recent days an algorithm called Single Pass K-Means is developed [15]. In this algorithm we use the single data, single time pass method. Each data vector comes to the cpu memory alone. Another algorithm, prior to the K-means and Single pass K-Means ,is developed in 2007, by

Arthur and Vassilvitskii, called K-Means ++. In K-Means ++ we use some probabilistic measure[16], while selecting the initial centroids of the clusters. It leads us to achieve good clusters as well as completes the task in less time.

1.2 Introduction to stream clustering

Now a days we have to handle huge data sets all over the world. Worldwide organizations like various multi-national companies, mobile operators, online shopping companies have to keep record of their huge customer base and tracking the record they have to take their organisations. This huge data analysiswas handled through BIG Data clustering[17] and analysis method. But, through Big Data analysis, we had to store the large data set first and then we could analyze that. In modern day life to optimize the memory taken we have to refine the data before storing and have to exclude the not required ones. Coming in this phase, we introduce data stream clustering, where we don't have to store all the data received and we will be able to take decisions and analyze the data while coming online process. Here in this project paper, we first will develop some of the popular partition approach algorithms for data stream clustering. Then we will check the acceptibility of them upon some streaming or, online data like, Spambase, Intrusion, Covertype [3]. After that we will run analyze the results with respect to various martices.

Chapter 2

Clustering Approaches

Clustering of data is the main analytical method of Data mining. The simplest definition of clustering is the grouping of similar data together. The data is generally represented as a vector of measurement or a point in multidimensional space. Clustering is of basically two types, supervised and unsupervised. In supervised[18], we have some pre-classified patterns and we have to do the task depending the information available from those patterns. In case of unsupervised[18], we do not have any predefined information about the classification and we have to check the nature of the data and have to do the clustering after collecting the information about the nature of the dataset. A good clustering will provide high intra-class similarity and low inter class similarity.

The major clustering approaches [5] are described in the next section.

2.1 Major Clustering Approaches

2.1.1 Partitioning approach

In this approach, we have to construct various partitions and then evaluate them by some criterion. In this way we can get the similar points together, which can create a cluster.

e.g., minimizing the sum of square errors.

Typical methods: k-means, k-medoids, CLARANS.

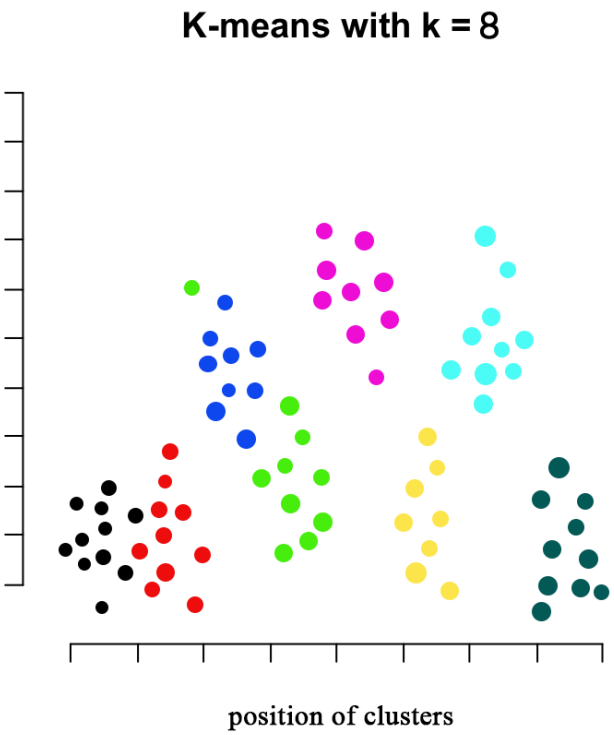


Figure 2.1: Partition Based clustering

2.1.2 Hierarchical approach

Here we have to create a hierarchical decomposition of the set of data (or objects) using some criterion. Then the similar clusters will be summed up hierarchialy step by step to create greater clusters. Hierarchical methods can be classified as agglomerative and divisive.

Agglomerative is a bottom up method which, at starting considers each object as a single cluster and in consecutive iteration it joins different similar clusters to a single cluster. On the other hand divisive method is a top down method which starts, considering all elements to a single cluster and then in successive iteration it separates that cluster into smaller ones.

Typical methods: Diana, Agnes, BIRCH, CAMELEON.

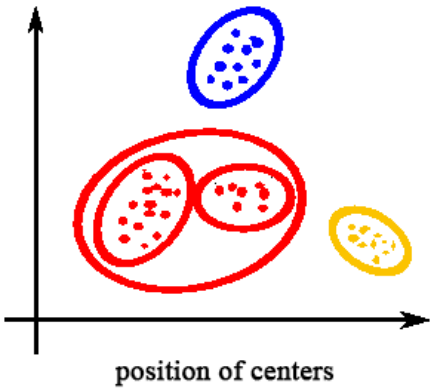


Figure 2.2: Hierarchical clustering

2.1.3 Density-based approach

This method is based on connectivity and density functions, which help us to create required clusters. The general idea of density based clustering is to grow the cluster as long as the cluster exceeds some density threshold. This type of clusters can be of arbitrary shapes.

Typical methods: DBSACN, OPTICS, DenClue.

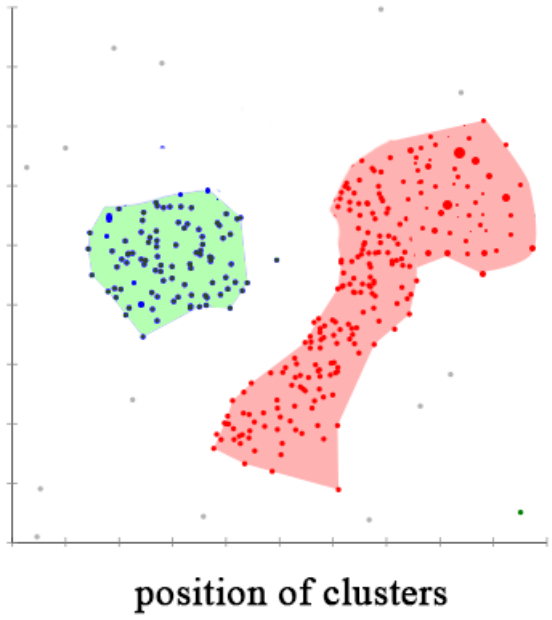


Figure 2.3: Density-based clustering

2.1.4 Grid-based approach

This approach is based on a multiple-level granularity structure. This structure helps us to get the proper shape of the cluster.

Typical methods: STING, WaveCluster, CLIQUE.

2.1.5 Model-based

A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other. This approach is faster than other methods of clustering. It does not depend upon the number of objects. It only depends on the number of cells in each dimension.

Typical methods: EM, SOM, COBWEB.

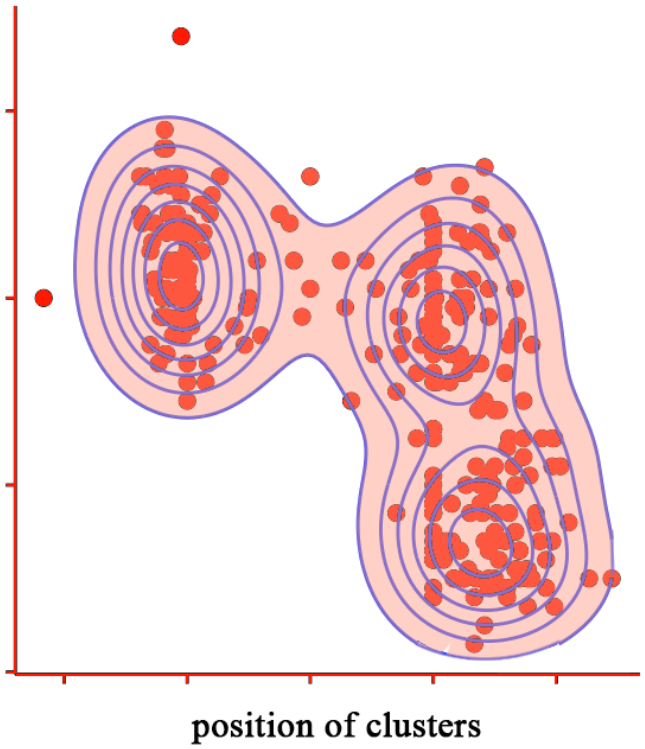


Figure 2.4: Model-based clustering

2.1.6 Frequent pattern-based

This approach is based on the analysis of frequent patterns. Some previously decorated patterns are used to form clusters of different shapes.

Typical methods: p-Cluster.

2.1.7 User-guided or constraint-based

This type of clustering is done by considering user-specified or application-specific constraints.

Typical methods: COD (obstacles), constrained clustering.

2.1.8 Link-based clustering

The objects are often linked together in various ways Massive links can be used to cluster objects.

Typical Methods : SimRank, LinkClus.

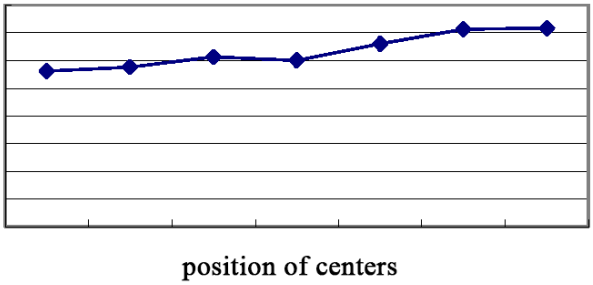


Figure 2.5: Link-based clustering

2.2 Real Life Applications

Clustering of data have many real life applications like :

- Biology: taxonomy of living things: kingdom, phylum, class, order, family, genus and species Information retrieval: document clustering
- Land use: Identification of areas of similar land use in an earth observation database
- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs.
- City-planning: Identifying groups of houses according to their house type, value, and geographical location.
- Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults.
- Climate: understanding earth climate, find patterns of atmospheric and ocean.
- Economic Science: market resarch.

2.3 Comparison among the Clustering approaches

Some of this algorithms are accurate for a single task, some other are accurate for other tasks. Each approach has its own drawbacks and usefulness. Some of them use the whole data stream available at the moment as a cluster and then take steps to create different clusters. Some other use to take arbitrary points as cluster centers, then comparing with other points, it creates the clusters. Therefore not a single approach is best for doing a whole work. Depending on low computational cost, memory use and time lapse we have to choose our required clustering approach. Let us have a comparison among the most popular clustering approaches[9].

Approaches	Advantages	Disadvantages
Partitioning	Easy to implement	The numbers of clustering should be predefined
Hierarchical	Helps to handle any type of distances	Generally this type of algorithms are of high complexity.
Grid-Based	It can handle noises very well	The size of grid in the space must be pre defined.
Density-based	It can handle any shaped clusters	Does not handle multi dimensional data properly and the number of parameters used it generally high.
Model-Based	The number of clusters is determined automatically based on standard statistics	It always depends on the previously hypothesized model.

Table 2.1: Stream Processing Vs. Traditional Processing

Chapter 3

Streaming Data and State-of-the-art techniques on Data Stream Clustering

3.1 Why Stream Clustering

In modern day life to optimize the memory taken we have to refine the data before storing and have to exclude the not required ones. Coming in this phase, we introduce data stream clustering, where we don't have to store all the data received and we will be able to take decisions and analyze the data while coming online process.

3.2 Basics of Stream Clustering

Basic definition of Data Stream is uninterrupted flow of a long sequence of data. Streaming data mainly focuses on speed.The data stream is continuously analyzed before it is stored on disk or any decision is taken depending on the data nature. Data Stream comes in packet form. We can consider two main parameters of it

- Sequence of tuples in a single packet.
- No of packets entering in a time interval.

So, it defines data stream as a very large sequence of data objects. The sequence is unbounded indeed and can be generated at any speed. In the data stream, data objects are described by a multidimensional attribute vector within a continuous, or

mixed attribute space[7]. So, finally we can say some typical characteristics of Data Stream are,

- Continuous Arrival
- Potentially Unbounded
- Disordered arrival

3.3 Difference of Stream Data Clustering and Traditional Clustering

This type of streaming data clustering requires some approaches that can be applied having partitioning observations continuously with keeping restrictions of memory and time in mind. In data stream clustering methods, there are many two-phase scheme which consists of an online component and an offline component. The online component processes data stream points and produces the statistics, and the offline component uses the summary data to generate the clusters. An alternative class is capable of generating the final clusters without the need of an offline phase. In this paper we will discuss about some algorithms that are used to cluster this unbounded streaming data and will also conclude about their efficiency. The basic differences between stream processing and traditional processing are given in the folowing table [7].

Stream Processing	Traditional Processing
Online Processing, Realtime	Offline Processing
Data comes rapidly inspite of having low resource	Normal data generation regarding to the available resource
It generally gives approximate results	It generally gives accurate result
Generally Linear and sublinier approaches are used	The approaches used here are of higher complexity, if necessary
RAW data is not stored here	Raw data is stored here

Table 3.1: Stream Processing Vs. Traditional Processing

3.4 Problems for Stream clustering

The process of mining data streams by creating data clusters remains a challenge due to various factors: (i) single-scan clustering: data clustering has to be done quickly just once, in a single pass due to the data stream arriving continuously; (ii) limited time: data clusters have to be created in real time within a limited time frame; (iii) limited memory: the clustering algorithm is equipped with only limited memory but it has to process a continuous, incoming, infinite data stream; (iv) unknown number and shape of clusters: these aspects of the data stream remain unknown prior to processing; (v) evolving data: the algorithm has to be designed in such a way as to be prepared to handle the ever changing aspects of the data stream; and (vi) noisy data: noise in data affects clustering results so the clustering algorithm has to withstand the noise that exists in the data stream [9].

To handle these problems, some window models were developed for ensuring proper clustering of Data Streams.

3.5 Introduction to window models

In most data stream scenarios, the stream reflects the emerging of new trends or changes on the data distribution from the more recent information. This information can be used to explain the evolution of the process under observation. Systems that give equal importance to outdated and recent data do not capture the evolving characteristics of stream data [19]. The so-called moving window techniques have been proposed to partially address this problem[20-22]. Data stream clustering algorithms obtain a data partition via an offline clustering step (offline component). The offline component is used together with a wide variety of inputs (e.g., time horizon, and number of clusters) to provide a quick understanding of the broad clusters in the data stream. Since this component requires the summary statistics as input, it turns out to be very efficient in practice [23]. There are three commonly-studied models in data streams [10]: i) Sliding windows; ii)Damped windows , iii) Landmark window.

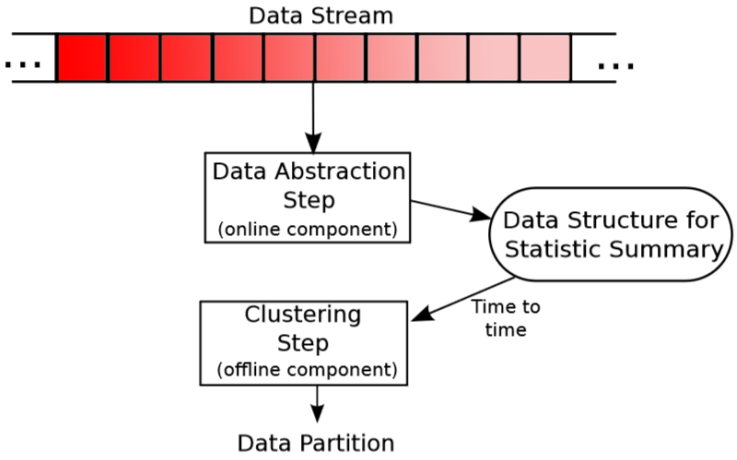


Figure 3.1: Data stream clustering framework

3.5.1 Sliding Window Model:

In the sliding window model, we use FIFO model. The most recent information from the stream is stored in a queue whose size can be variable or fixed. This queue is taken as the object for a certain time period. The sliding of the stream is based on the principles of queue processing, where the first object added to the queue will be the first one to be removed.

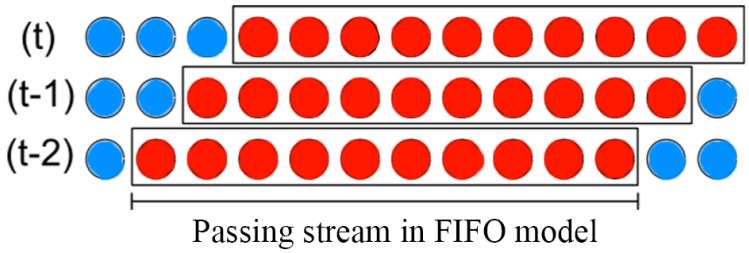


Figure 3.2: Sliding window model

Several data stream clustering algorithms find clusters based on the sliding window model[21,24,25]. In summary, these algorithms only update the statistic summaries of the objects inserted into the window. The size of the window is set according to the available computational resources.

3.5.2 Damped Window Model:

Damped window model is also known as as time-fading model. This model considers the most recent information by associating weights to objects from the data stream [26]. As new the object of the stream is, it is as heavier than the older ones. The weight of the information fades with time.

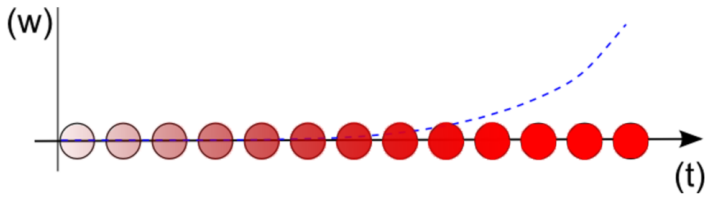


Figure 3.3: Damped window model

3.5.3 Landmark Window Model:

In this model the window requires handling disjoint portions of the streams (chunks), which are separated by landmarks (relevant objects). Landmarks are selected in terms of time, (e.g., on daily or weekly basis) or in terms of the number of elements observed since the previous landmark [27]. All objects that arrived after the landmark are kept or summarized into a window of recent data. When a new landmark is reached, all objects kept into the window are removed and the new objects from the current landmark are kept in the window until a new landmark is reached.

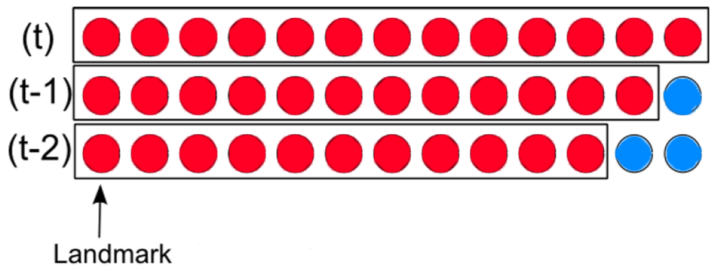
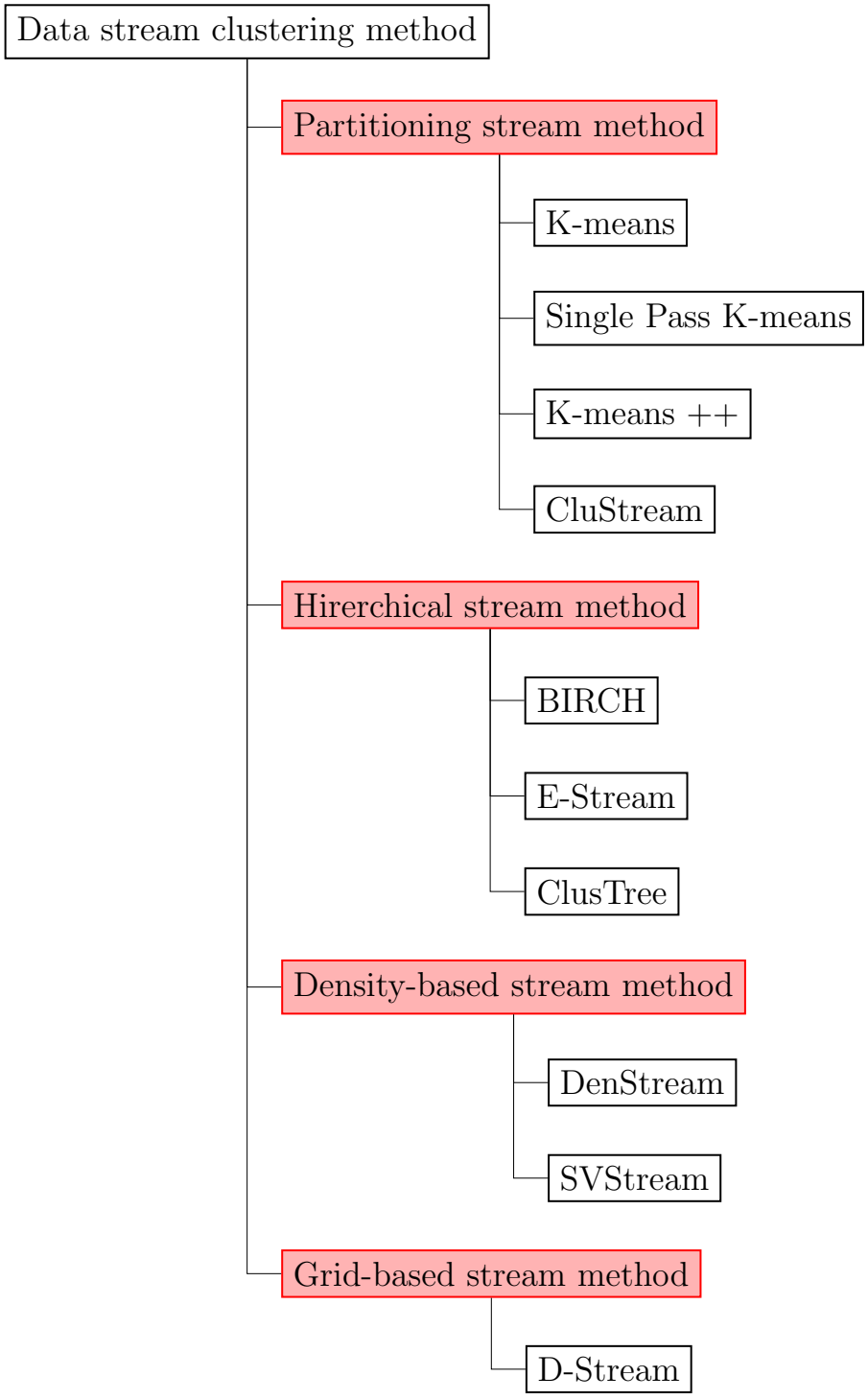


Figure 3.4: Landmark window model

After development of these models, algorithm of various approach were used upon the models. The most popular and widely used model is Sliding Window model. The other two models are used in critical conditions. In the next section we will see the State-of-the-art Techniques of data stream clustering.

3.6 State-of-the-art Techniques

Let us have an overview about the works on the data stream clustering til date and gives knowledge about some relevant algorithms used to deal with the problem. The given figure gives us idea about those algorithms regarding to their underlying clustering approach.[8] We will see this in the following tree.


Data stream clustering method
Partitioning stream method
K-means
Single Pass K-means
K-means ++
CluStream
Hirerchical stream method
BIRCH
E-Stream
ClusTree
Density-based stream method
DenStream
SVStream
Grid-based stream method
D-Stream

Among these algorithms, in this project we have mainly studied Partitional approach algorithms. In the next section it is described how we go through the whole task and the analysis of the received result.

Chapter 4

Results and Analysis

4.1 Preliminaries

We begin by defining a stream more formally. A data stream is a set N of points $x_1, ...x_i, ...x_n$ that can only be read in increasing order of the index i. For example, these points might be vectors in ¡d. A data stream algorithm is not allowed randomness but can retain a small amount of information about the data it has seen so far. Its performance is measured by the number of linear scans it takes over the data stream, the amount of information it retains, and the usual measures: in the case of a clustering algorithm, for example, these could be Memory used, Cost and running time.[11]

We will define the k-Median problem; Suppose we are given a set N of n objects in a metric space M with distance function d. Then the kMedian problem is the problem of choosing k point $c_1....c_k$ so as to minimize

$$\sum_{i=1}^{k} \sum_{x \in N_i} d(x_i, c_i)$$

4.2 Algorithms Studied

In this paper we have basically studied the data streaming algorithms. based on partitional approach. The most popular partitional approach algorithm is K-Means. It is so much popular because it is very easy to implement and if the preliminary assumption is done properly, it can be done very fast too. This method fully depends upon the mean of the points inside a cluster.

4.2.1 K-Means clustering algorithm

In the clustering problem, we are given a training set x(1),...,x(m), and want to group the data into a few cohesive "clusters." Here, we are given feature vectors for each data point $x(i) \in Rn$ as usual; but no labels y(i) (making this an unsupervised learning problem). Our goal is to predict k centers and a label c(i) for each data point. The k-means clustering algorithm is as follows[12]:

1. Initialize cluster centers $\mu_1, \mu_2, \mu_3 ... \mu_k \in R^n$ randomly.
2. Repeat until convergence
For every $\imath$, set $c^{(i)} = arg\min_j \|x^{(i)} - \mu_j\|^2$

For each j, set $\mu_j = \frac{\sum_{i=1}^{m} 1\{c^i=j\}x^i}{\sum_{i=1}^{m} 1\{c^i=j\}}$

Table 4.1: Pseudocode of K-Means algorithm

4.2.2 Single Pass K-Means Algorithm

A single Pass streaming algorithm is one such algorithm, where the data is handled by the algorithm just one time. Now if after reading the data one time we can apply K-Means upon that. Lets discuss in details :

first we read the data points one by one. At starting we take each data point as a cluster and the point itself is its center. We also define some criterion, depending on which we will decide whether to keep two points in same cluster or not. Now, when a data enters to the running environment, firstly it is compared to the previously created cluster centers. If any the cluster centers satisfy the criterion with the incoming data point, then, we keep that point in that cluster. If not any cluster center satisfies the criterion, with respect to the incoming point, then we create another new cluster, having the new point as its center. This process continues until the pre-described value of k is reached.

4.2.3 K-Means ++ Algorithm

The k-means++ algorithm is an expected O(logk)-approximation algorithm. There is a difference in choosing the preliminary centers with K-Means. In K-Means we use to choose the preliminary centers randomly. Then, by consecutive iterations, we get the final cluster. But, here we apply a probabilistic technique while selecting the initial points. This helps us to reach to the final clustering and helps to get it in lesser time than simple K-Means algorithm.

1. Choose an initial center c_1 uniformly at random from X.
2. Repeat $(k-1)$ times.
3. Choose the next center c_i,selecting $c_i = x' \in X$
with probability $= \frac{D(x')^2}{\sum_{x \in X} D(x)^2}$
($here D()$ *denotes the distances w.r.t. the subset of points chosen in the previous rounds*)

Table 4.2: Pseudocode of K-Means++ algorithm

4.3 Parameters

For the algorithm we set all parameters of the experimental environment as required for them to run. The more less number of clusters it has, the finer is the clustering. Therefore, from time to time, Partitional Algorithms did not produce the correct number of centers, especially when the number of clusters k was high. For this reason, the memory settings had to be manually adjusted for each individual dataset. For algorithm K-Means++,we determined experimentally an appropriate memory block size m as a function of k. For obvious reasons we need to choose m k. To estimate an m that is sufficient to obtain good approximation results, we ran several experiments for different values of k and m on the datasets Spambase and Covertype. Due to the randomized nature of K-Means++, we conducted ten runs for each xed k and for each fixed size m.

On the one hand m should be chosen not too small (e.g. a very small multiple of k), because, for these values of m, the quality of a clustering can be easily improved, without sacrificing too much running time. On the other hand m should not be chosen too large (e.g. a large multiple of k), because the increase in quality is only very small compared to clusterings for smaller custers, but the running time is significantly higher. Therefore, we assume that our choice of m = 200k provides a good trade-off for arbitrary datasets. However, smaller sizes such as m = 20k or m = 50k might still be sufficient to obtain very good clustering results on datasets with k well separated clusters. [13]

4.4 Metrics

We have taken basic three matrices like Runtime, Memory Used and Program cost. Each case our primary ambition was to optimize the matrices. like, we tried to minimize Runtime, Memory Used and Program cost by using the parameters properly. Sometimes the small change of any parameter results very large change in the matrices. On the other hand sometimes change in parameter values gives some unexpected change in metrics too.

4.5 Datasets

Since synthetic datasets (like Gaussian distributed points near some uniformly distributed centers in Rd) are typically easy to cluster, we use real world data sets to obtain practically relevant results. Our main source for data was the UCI Machine Learning Repository [3] (datasets Covertype2, Census 1990, Intrusion, and Spambase as well as data set Tower from. The size and dimensionality of the datasets is summarized in the table below.

Datasets	Data Points	Dimension	Type
Spambase	4601	57	float
Intrusion	311079	34	int, float
Covertype	581012	54	int, int

Table 4.3: used Streaming Datasets

4.6 Results

To get an overview of results of K-Means++, Single pass k-means and k-means, we conducted several experiments for different values of k on the three larger, online streaming datasets, i.e. the datasets Covertype, Spambase, Intrusion. In each of these experiments, we set m = 200k. For the randomized algorithms K-Means++ and Single Pass k-Means, ten experiments were conducted for each fixed k. For BIRCH, a single run was used, since it is a deterministic algorithm. The average running times and cost of the clustering is summarized in following figures. In our experiments, averagely algorithm K-Means++ had the best running time of all algorithms. However, this comes at the expense of a high k-means clustering cost. Furthermore, as already mentioned, one drawback of this algorithm is the need of adjusting parameters manually to obtain a clustering with the desired number of centers. By comparing K-Means and Single Pass K-Means, we observed that the quality of the clustering was on a par. More precisely, the absolute value of the cost of both algorithms lies within a 5

The cost of clustering computed by algorithm Single Pass K-means tends to be more stable than the costs computed by other two. The ratio between the running time needed by K-Means++ and the running time needed by Single Pass K-Means algorithm is decreasing with increasing number of clusters. For m = 500,K-Means++ computed the clustering for k = 100 in about 9%of the running time of Single Pass k-Means and for k = 200 in about 2% of the running time of Single Pass k-Means. For m = 1000, K-Means++ computed the clustering for k = 100 in about 38%of the running time of Single Pass k-Means, whereas for k = 200 it needed about3%of the running time of Single Pass k-Means. Overall, we conclude that, if the first priority is the time of the process, then in particular, if the number of cluster centers is large, single pass K-Means is doing a decent job.

Lets see the graphical representation of our result.

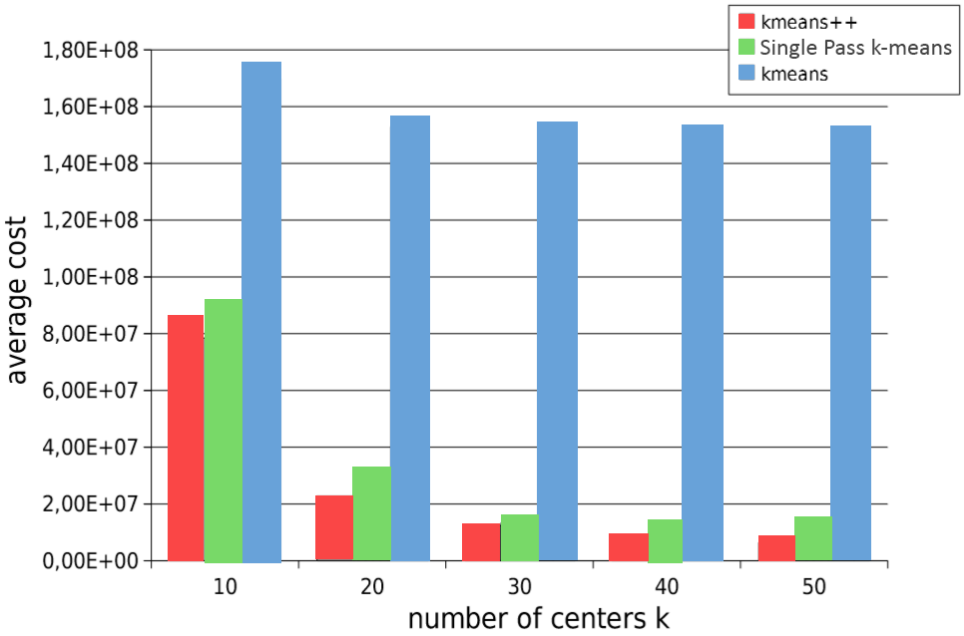


Figure 4.1: Spambase dataset costs

The graph shows the average cost of the process upon spambase dataset. The result for each value of k is taken for ten times and then the average is shown in the graph. It clearly states that among the three algorithms, K-Means++ is best, with respect to cost.

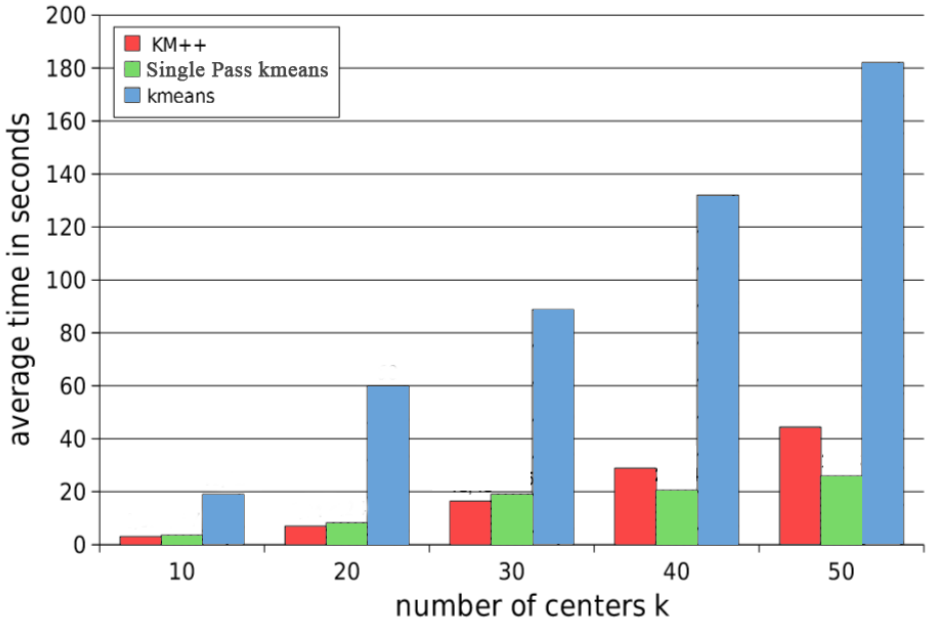


Figure 4.2: Spambase dataset runtime

The graph shows the average cost of the process upon spambase dataset. The result for each value of k is taken for ten times and then the average is shown in the graph. We can see that sometimes, for larger values of k, single-pass k means give better result than that of K-Maens++.

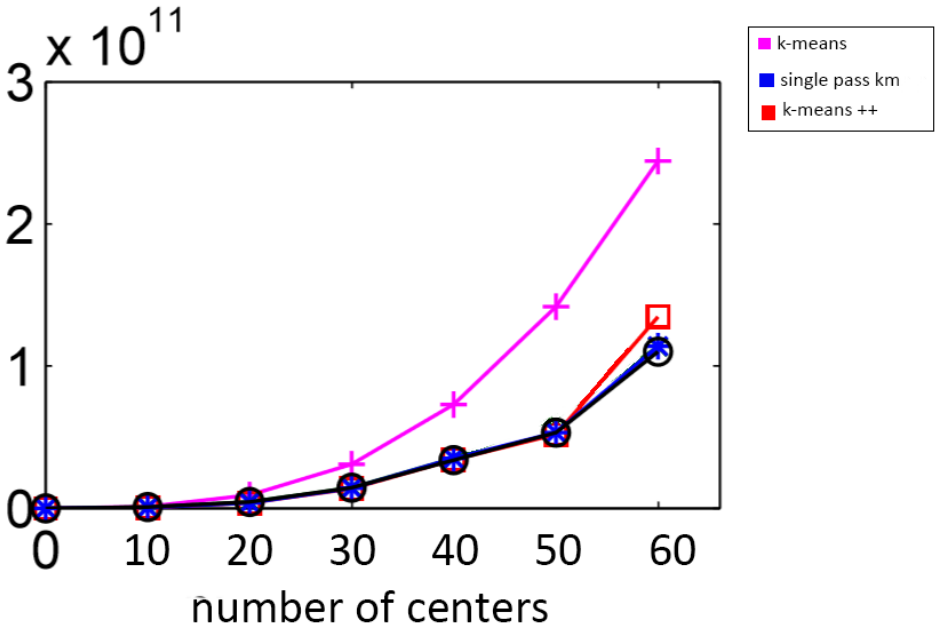


Figure 4.3: Spambase dataset used memory

The graphical representation shows that excluding the higher number of centers, Single-Pass K-Means and K-Means++ uses almost equivalent memory.

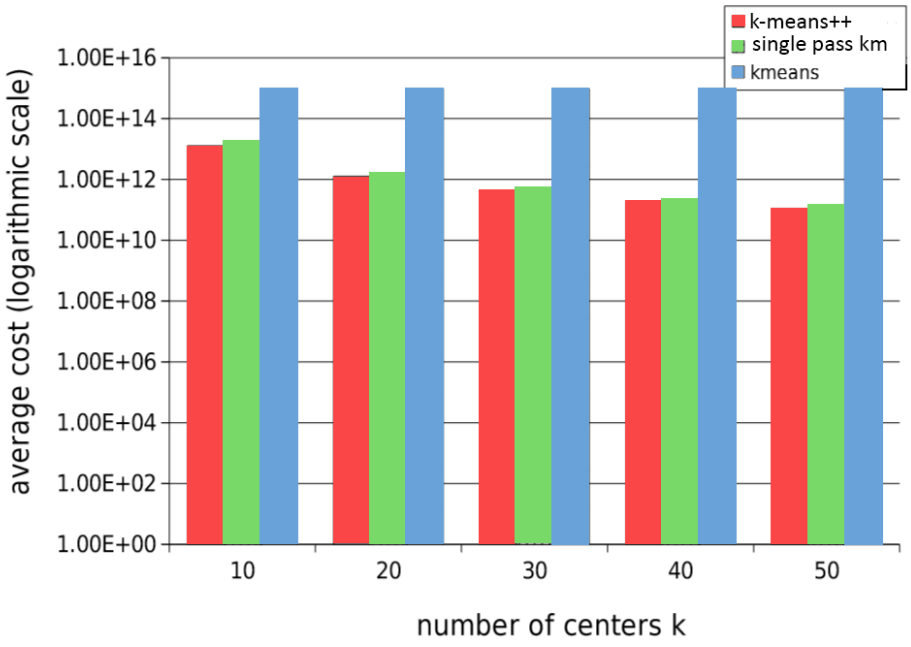


Figure 4.4: Intrusion dataset cost

Intrusion data set cost graph shows that for any number of centers, k means algorithm uses almost same cost. Here number of center does not hamper the result. On the other hand, k-means++ and single pass k-means takes almost equivalent cost, each time.

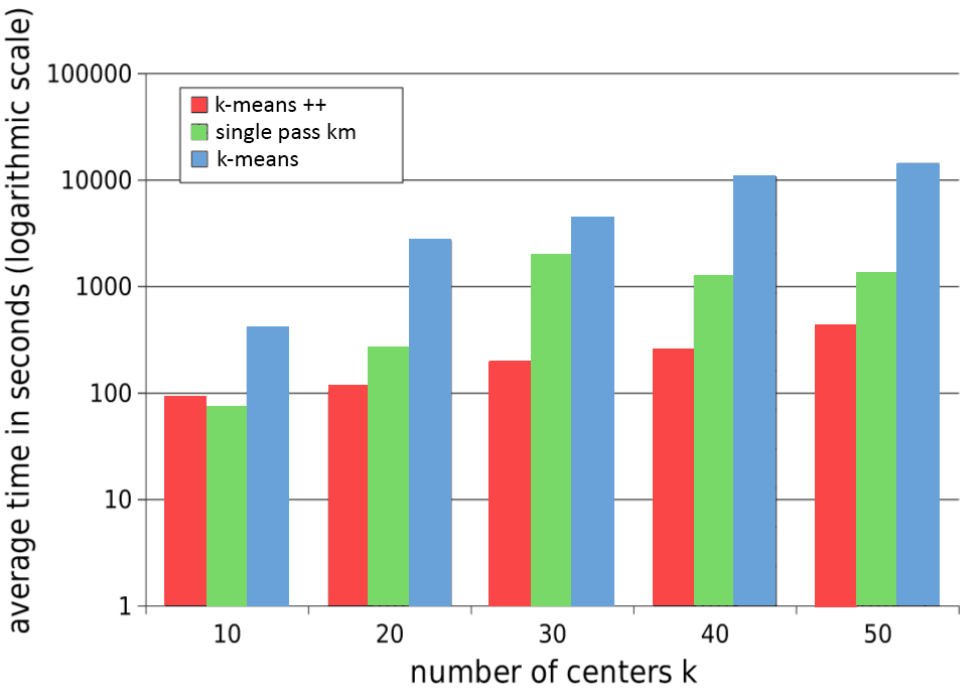


Figure 4.5: Intrusion dataset runtime

From the runtime graph, we can see, for very low number of cluster centers, k-means ++ takes more time than single pass k-means. But, averagely, for all the cases, k-means ++ always gives the optimized runtime.

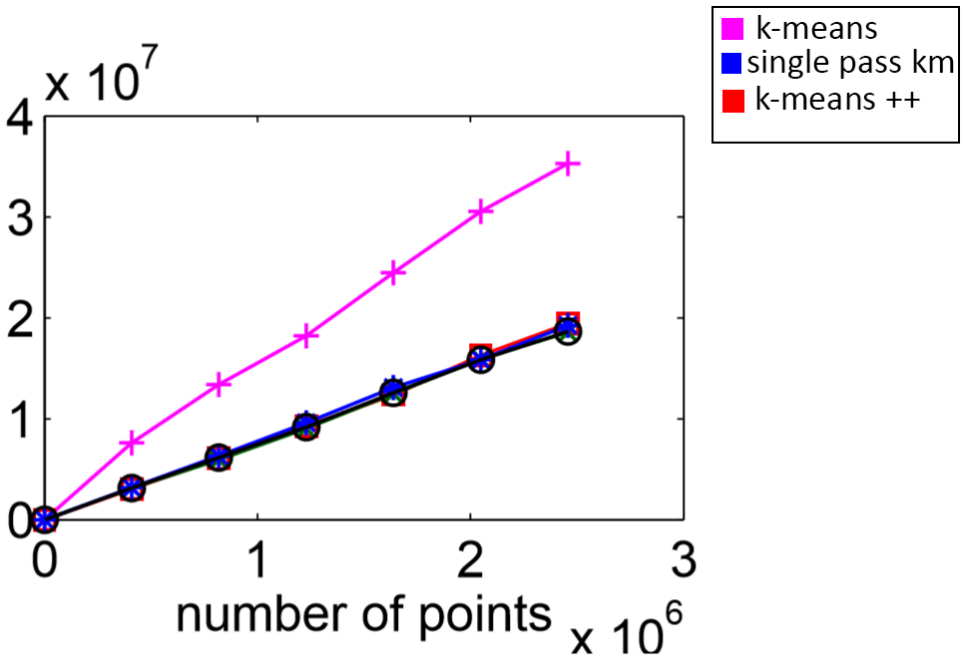


Figure 4.6: Intrusion dataset used memory

Again for this dataset also, we can see, single pass k-means and k-means ++ averagely uses equivalent memory for each case. On the other hand k-means takes more memory than other to algorithms, always.

4.7 Analysis

Here, we obtain a low dependency on the dimension d, so that our partitional approach is suitable for high-dimensional data. Of course, careful consideration has to be given to the choice of the cluster center number k. The basic K-Means algorithm is always the maximum time consuming and cost consuming algorithm. On the other hand K-Means++ is the most less time and cost consuming algorithm.

Generally our experiments have matched with the previously done historical datas. As per the theoretical discusion, we can conclude that the time period of K-Means++ is always lower than simple K-means and single pass k-means. But, we can see from our experimental results, if the no of clusters increase a lot, for a few dataset, it takes more time than Single-pass K-means. This may be,due to the fact that for K-Means++, every time a cluster center is chosen, a lot of comparisons and probabilistic calculation has to be done. Now, if the number of cluster centers increase, it means , choosing of each center initially increases time period of the process. So, we can conclude that K-means++ is better for lower number of cluster processes.

Chapter 5

Conclusion

Evaluation of clustering results on streaming data sets received a lot of research attention for in recent years. Many recent publications deal with the important task of clustering on evolving data streams. In this project we studied basically Partitional Approach streaming data algorithms for clustering on evolving data streams. It is the first measure that takes the important properties of the streaming context into account. It is based on the cluster center numbers and choosing the preliminary centers correctly. We have studied the results of some algorithm upon UCI data sets minutely and also have marked some special cases, where the traditional assumption fails. Finally we can conclude that K-Means ++ is best among these three algorithms, we studied. Sometimes, for very higher number of centers, Single Pass K-Means surpasses the time-consuming and memory used efficiency of K-Means++.

Bibliography

[1] S. Lloyd. *Least Squares Quantization in PCM*. IEEE Transactions on Information Theory, 28: 129-137, 1982.

[2] A. Aggarwal, A. Deshpande and R. Kannan. *Adaptive Sampling for k-means Clustering*. Page 18-21, 2009.

[3] A. Asuncion and D.J. Newman. *UCI Machine Learning Repository*. `http://www.ics.uci.edu/mlearn/MLRepository.html`. University of California, Irvine, School of Informationand-ComputerSciences,2007.

[4] N. Ailon, R. Jaiswal, C. Monteleoni *Streaming k-meansapproximation*. page 7-10, 2011.

[5] J. Han, M. Kamber, J. Pei. *Data Mining Concepts and Techniques*. [*ISBN 978-0-12-381479-1*]. Chapter 10 Cluster Analysis: Basic Concepts and Methods, Page 443-460.

[6] D. Namiot *International Journal of Open Information Technologies*. ISSN: 2307-8162 vol. 3, no. 8, 2015.

[7] J. A. Silva, E. Faria, R. Barros, R. C. Hruschka, A. C. de Carvalho, J. Gama. *Data stream clustering: A survey*. ACM Computing Surveys (CSUR), 46(1), 2013.

[8] Md. Ghesmoune, L. Mustapha and H. Azzag *State-of-the-art on clustering data streams*. Published: December 2016.

[9] M. Mousavi1, A. A. Bakar1 and Md. Vakilian1 *Data Stream Clustering Algorithms: A Review* . ISSN 2074-8523, 2015.

[10] J. A. Silva, E. Faria, R. Barros, R. C. Hruschka *Data Stream Clustering: Another approach*. ACM Computing Surveys (CSUR), 2008.

[11] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, R. Motwani *Streaming-Data Algorithms For High-Quality Clustering*. Page 3,4 , 2012.

[12] C. Piech, Ng. Andrew. `http://stanford.edu/ cpiech/cs221/handouts/kmeans.html` .

[13] M. R. Ackermann, M. Martens, C. Raupach, and K. Swierkot *StreamKM++: A Clustering Algorithm for Data Streams*. ACM J. Exp. Algor. V, N, Article , page 19-20, 2008.

[14] J. B. MacQueen *Some Methods for classification and Analysis of Multivariate Observations*. Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297, 1967.

[15] F. Farnstrom, J. Lewiss, and C. Elkan. Scalability for clustering algorithms revisited. SIGKDD Explor. Newsl., 2:51–57, 2000.

[16] D. Arthur and S. Vassilvitskii *Worst-case and smoothed analyses of the icp algorithm, with anapplicationtothekmeansmethod*. FOCS, 2006.

[17] O. Kurasova, V. Marcinkevicius, V. Medvedev et. al *Strategies for Big Data Clustering*. Tools with Artificial Intelligence (ICTAI),IEEE, 2014.

[18] R. Sathya, A. Abraham *Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification* . International Journal of Advanced Research in Artificial Intelligence,Vol. 2, No. 2, 2013.

[19] Y. Chen, and L. Tu *Density-based clustering for real-time stream data.*. ACM SIGKDD international conference on Knowledge discovery and data mining. ACM Press, 133–142, 2007.

[20] D. Barbara *Requirements for clustering data streams*. SIGKDD Explorations,Special Issue on Online, Interactive, and Anytime Data Mining 3, 23–27. 2002

[21] B. Babcock, M. Datar, R. Motwani and L. O'Callaghan. *Maintaining variance and k-medians over data stream windows*. Proceedings of the twenty-second ACM SIGMODSIGACT-SIGART symposium on Principles of database systems. ACM, 234–243. 2003.

[22] J. Gama *Knowledge Discovery from Data Streams*. Chapman Hall/CRC, 2010.

[23] C. Aggarwal *A Framework for Diagnosing Changes in Evolving Data Streams*. ACM SIGMOD Conference. 575–586. 2003.

[24] J. Ren, and R. Ma *Density-based data streams clustering over sliding windows*. Sixth International Conference on Fuzzy Systems and Knowledge Discovery. Vol. 5. 248–252. 2009.

[25] A. Zhou, F. Cao, W. Qian and C. Jin *Tracking clusters in evolving data streams over sliding windows*. Knowledge and Information Systems 15, 2, 181–214. 2008.

[26] N. Jiang, and L. Gruenwald *Research issues in data stream association rule mining*. SIGMOD Record 35, 1, 14–19. 2006.

[27] A. Metwally, D. Agrawal and A. El Abbadi *Duplicate detection in click streams*. Proceedings of the 14th international conference on World Wide Web. ACM, 12–21. 2005.