Exploring K-means Clustering Over Sliding Window in Continuous Data Stream

Purva Kulkarni kpurva1@umbc.edu
Rujuta Palande palande1@umbc.edu
Kishan Pawar kish5@umbc.edu
Siddharth Utgikar sidd3@umbc.edu

Department of Computer Science and Electrical Engineering,

University of Maryland, Baltimore County

ABSTRACT

We explore K-means clustering in the sliding window model for streaming data. In the streaming model, we are restricted to use space sub-linear to the size of input and this input typically must be processed in a single pass. We have designed an algorithm called Bucket Algorithm which uses logarithmic space and time for K-means clustering in the sliding window for streaming data.This algorithm maintains two buckets each of size equal to the size of window to store active data elements from the data stream and then data in these buckets is used for clustering to get K centers using K-means clustering.

Keywords

K-means, K-median, clustering, sliding window, data stream, bucket algorithm

1. INTRODUCTION

Many applications like network monitoring, telecommunications, financial monitoring, social media communications,sensor networks generate continuous data streams. For example data coming from temperature sensors over the network is received and processed in the form of data streams. Temperature sensors send data continuously but the data varies after certain time interval. Hence such kind of data needs clustering. For example if sensors are sending temperature readings every minute, the chance of finding significant change in the reading every minute or every few minutes is less. Instead if we get this data and cluster the data over a specific window, we can get significantly changing temperature readings for example clustering the data over a window of an hour, where we can see changes in temperature. Many algorithms have been proposed for clustering of data over sliding window on such data streams. Because of practical reasons, any algorithm for data streams must satisfy the following requirements :
1) Process the stream in one pass.
2) Use very small amount of working memory. Approach focuses on K-means clustering on the sliding window for continuous data streams. We are considering an infinite data stream where data items in the stream are received one at a time and K-means clustering is applied over a sliding window of size W (not over the whole stream) which covers W most recent data items received. These most recent W elements are called active data elements, the rest are called expired and they are not used in any further processing. Once a data element has been processed, it cannot be retrieved for further computation at a later time unless it is explicitly stored in memory. The amount of memory available is assumed to be limited, in particular, sub-linear in the size of sliding window. Therefore algorithms that require storing the entire set of active elements, are not acceptable in our model.

We employ the notion of buckets which are used to store active data elements in the window. At a time, we will be maintaining two buckets of size equal to window size-1 and the active data collected in buckets is used for clustering by using K-means algorithm. The K-center values thus obtained are used for further processing like query answering.

2. PREVIOUS WORK

Lot of work has been done in the area of insertion-only streaming algorithms and dynamic geometric streaming algorithms for the clustering problems like the k-median and the k-means problem. Some of the examples of work done is are follows:
a. The first of its kind, insertion-only streaming algorithm was proposed by Guha, Mishra, Motwani and OCallaghan for the k-median problem. An approximation of 2 $\wedge$ O(1/ ϵ) was given by them using O(n $\wedge$ ϵ) space, where $\epsilon < 1$.
b. Charikar, OCallaghan, and Panigrahy, came up with an approximation of O(1) insertion only s streaming algorithm which used O(k log2 n) space. In this algorithm, they maintained a set of O(k log n) candidate centers that would be eventually reduced to exactly k centers using an offline k-median algorithm after the observation or monitoring of the entire stream.
c. Har-Peled and Mazumdar gave an approximation of (1 + ϵ), for k-median and k-means in the insertion-only streaming model which used coresets in case of e elements of the stream which are points in d-dimensional Euclidean space. The coresets presented by Har-Peled and Mazumdar required O(kϵd log n) space, yielding a streaming solution using O(kϵd log2d+2 n) space via the famous merge-and-reduce approach.

3. METHODS

The algorithm accepts two inputs. The data itself and K, the number of clusters/buckets. The output is K clusters with input data partitioned among them. The aim of K-means clustering is that we want to group the items into K clusters such that all the items in same cluster are as similar to each other as possible. And items not in same cluster are as different as possible. We use the distance measures to calculate similarity and dissimilarity. One of the important concepts in k-means is that of centroid/mean-vector. Each cluster has a centroid. You can consider it as the point that is most representative of the cluster. Equivalently, centroid is point that is the center of a cluster.The inputs considered are in the simple binary form. This implies that there are K-constant number of groups. We have used n-dimensional Eulidean distance measure as follows:

$$d = \sqrt{(x_1 - y_1) \wedge 2 + ... + (x_n - y_n) \wedge 2}$$

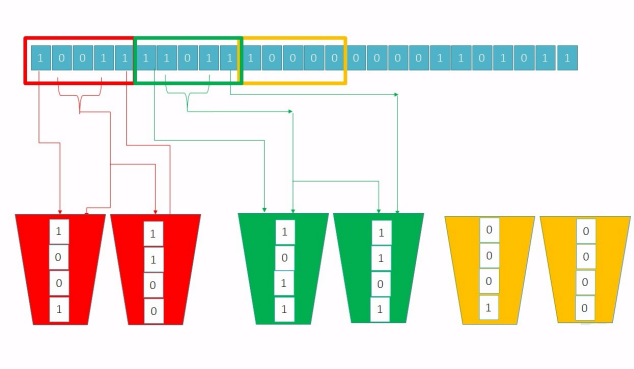

Fig. 1. Execution of Bucket Algorithm over Sliding Window

The algorithm devised by us is as follows :
Step 1. Start
Step 2. Randomly choose k items and make them as initial centroids.
Step 3. For each point, find the nearest centroid and assign the point to the cluster associated with the nearest centroid.
Step 4. Update the centroid of each cluster based on the items in that cluster. Typically, the new centroid will be the average of all points in the cluster.
Step 5. Repeat steps 2 and 3, till no point switches clusters.

4. IMPLEMENTATION

Our team followed incremental approach to implement and analyze the proposed work[1]. Initially we did requirement analysis of our work by scramming through the related work done and referred

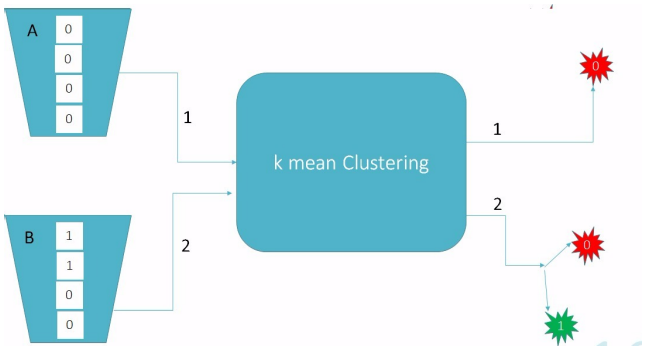

Fig. 2. K-means Clustering

the related research papers. While implementing the sliding window module, we refined it at each phase of the project. One of the major challenges was the implementation of the K-means algorithm on sliding window for the streaming data. The approach we used for the same was the generic software development life cycle. It consisted of various phases like planning, defining the scope, designing, testing and analyzing the results. The analyzing part was preceded by plotting graphs from the obtained results. We also verified our results for different window sizes W.

The system we used to implement our work has following configuration:

Processor: Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz

Cores: 4 Cores

Installed Memory (RAM): 8.00 GB DDR3

System Type: 64-bit Operating System, x64-based processor

Language: JAVA

Compiler: JAVA compiler of Eclipse IDE

IDE: Eclipse To calculate the actual running time of the algorithm we used the Java function of System.nanoTime().We calculated the difference between the system time before the beginning of execution of our algorithm and the system time recorded after the execution. For the space complexity analysis we used the Java Runtime to get the total memory and the free memory. The difference of which provided us with the space used by our algorithm.

5. RESULTS

We executed Bucket Algorithm for various window sizes over a continuous data stream and calculated space and time required for each execution. The results plotted in the report are based on the experimental data set of 50000 samples of data. We varied the window size from 25 to 1000 in the steps of 25. This gave us the projection of the performance of our algorithm. Fig. 1 and 2 show the plotted data for space usage and time required for the execution of various window sizes respectively.

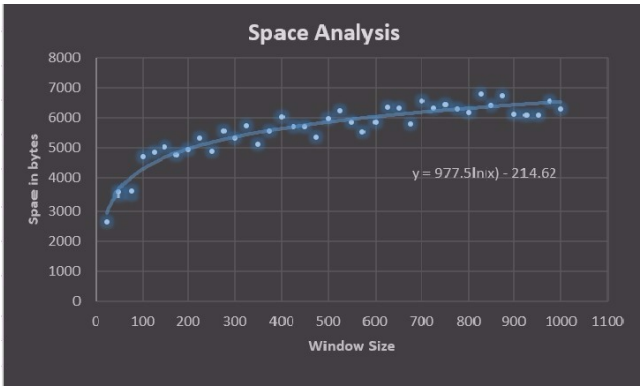

Fig. 3. Space Analysis

Window Size "W"	Space Required (Bytes)	Window Size "W"	Space Required (Bytes)
25	3296.128845	525	6413.719821
50	4005.911558	550	6461.356317
75	4421.107828	575	6506.874922
100	4715.69427	600	6550.455967
125	4944.193267	625	6592.257689
150	5130.890541	650	6632.4197
175	5288.740837	675	6671.065795
200	5425.476983	700	6708.306263
225	5546.086812	725	6744.239775
250	5653.97598	750	6778.954963
275	5751.573604	775	6812.531742
300	5840.673254	800	6845.042409
325	5922.636987	825	6876.552588
350	5998.52355	850	6907.122022
375	6069.172251	875	6936.80526
400	6135.259696	900	6965.652238
425	6197.339309	925	6993.708787
450	6255.869525	950	7021.017072
475	6311.234359	975	7047.61597
500	6363.758693	1000	7073.541406

Fig. 4. Table : Space Analysis

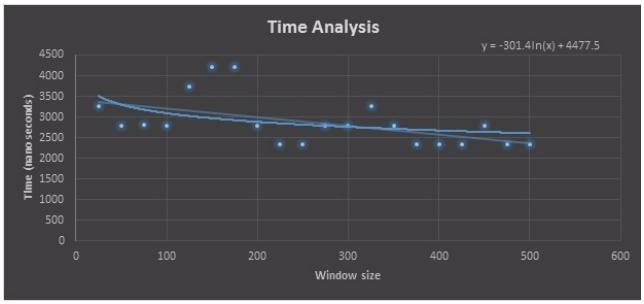

Fig. 5. Time Analysis

Window size "W"	Time required (ns)	Window size "W"	Time required (ns)
25	3266	525	2333
50	2799	550	2333
75	2800	575	2333
100	2799	600	3732
125	3732	625	12596
150	4199	650	4665
175	4199	675	3732
200	2799	700	3733
225	2333	725	3732
250	2333	750	4198
275	2799	775	4198
300	2799	800	4198
325	3265	825	4198
350	2799	850	4198
375	2333	875	4199
400	2333	900	4199
425	2333	925	5598
450	2799	950	3266
475	2333	975	3453
500	2333	1000	4676

Fig. 6. Table : Time Analysis

6. DISCUSSION

As seen from Space and Time Analysis graphs, Bucket algorithm takes logarithmic space and time for various window sizes. The algorithms previously proposed take polynomial or poly-logarithmic space. But the algorithms proposed in [1] and [7] take approach of K-median clustering in General Metric Space and Euclidean Space. Our proposed Bucket algorithm does value based K-means clustering instead of distance(in General Metric Space and Euclidean Space) based clustering done in [1] and [7].

7. CONCLUSION

We proposed Bucket Algorithm for K-means clustering over Sliding Window on continuous data streams.This algorithm takes logarithmic space and time. We implemented value based clustering approach for Bucket Algorithm and based our results on an integer data stream. Also we realized that the size of the window being used along with the nature of the data stream may result in data loss in some scenarios. Hence the choice of window size is very crucial.

8. FUTURE WORK

We implemented Bucket Algorithm with value based clustering, this makes our algorithm applicable only for data streams containing similar type of data items that is data streams containing only integer data items or character data items etc. We are planning to extend our work for K-means clustering proposed in [1] for General Metric Space.

9. ACKNOWLEDGMENT

We would like to thank Professor Alan Sherman for his persistent support and constant encouragement throughout the course of the project.We would like to thank authors of the paper[1] Vladimir Braverman, Harry Lang, Keith Levin and Morteza Monemizadeh for their prompt response and for sharing additional material explaining the algorithm devised by them.

10. REFERENCES

[1] Vladimir Braverman, Harry Lang, Keith Levin, Morteza Monemizadeh- Clustering Problems on Sliding Windows (SODA).
[2] D. Feldmam, M. Monemizadeh, C. Sohler and D. Woodruff. Coresets and sketches for high dimensional subspace problems. In proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 630-349, 2010.
[3] K. Chen. On coresets for k-median and k-means clustering in metric and Euclidean spaces and their applications. SIAM Journal on Computing, 39(3):923-947, 2009.
[4] D. Feldman, A. Fiat and M. Sharir. Coresets for weighted facilities and their applications. In proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS),pages 315-324, 2006.
[5] S. Har-Peled and S. Mazumdar. Coresets for

k-means and k-median clustering and their applications. In Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC), pages 291-300, 2004.
[6] Tim Roughgarden and Gregory Valiant. Approximate Heavy Hitters and the Count-Min Sketch
[7] Babcock, Brian and Datar, Mayur and Motwani, Rajeev and O'Callaghan, Liadan (2002) Sliding Window Computations over Data Streams. Technical Report. Stanford InfoLab.

11. APPENDICES

Appendix 1A: Source code for clustering in sliding windows using K-means algorithm.