New Streaming Algorithm for Clustering

Tanuj Gupta[1], **Tinish Gupta**[2]

[1]*Graduate Student, Department of Computer Engineering, BITS Pilani Hyderabad, Telangana, India*
[2]*Graduate Student, Department of Civil Engineering, IIT Delhi, Delhi, Delhi, India*

---***---

Abstract- *The data stream model has recently attracted attention for its applicability to numerous types of data, including telephone records, web documents and clickstreams. For analysis of such data, the ability to process the data in a single pass, or a small number of passes, while using little memory, is crucial. This can be done by using streaming algorithms. In the paradigm of clustering algorithms, no such algorithm has been devised that is both a streaming algorithm and that guarantees time complexity and quality within a fixed factor of the value of the optimal solution. In this paper, we studied various aspects of clustering techniques being used today. We have devised a clustering technique that is a streaming algorithm that uses Linear space to cluster data without compromising the quality and time complexity. Finally based on empirical evidence, we have tested the quality of our streaming algorithm as good as K-means++, that is O(log(k))-approximation to the optimal solution.*

Key Words: Streaming Algorithm, Clustering, K-means++, sublinear-space, Big Data, …

1. Introduction

1.1 Advent of Big Data

With the advent of new technologies, devices, and communication means like social networking sites, data being produced by humanity has been increasing exponentially. Statistically speaking, the pivot point for surge in data was around 2012 when companies began collecting more than three million pieces of data every day. Since then, this volume doubles about every 40 months. This enormous data has marked the arrival of a new paradigm in technology, called 'Big Data'.

1.2 Challenges with Big Data

By the definition of BID DATA itself, it's clear that we are dealing with humongous amounts of data that the 21st century is producing on a daily basis. But with great data come great challenges. With increasing amounts of data being produced, businesses demand real time analysis of data to find useful patterns in data in order to be able to make useful predictions. In order to extract useful insights from such copious amounts of data in real time, the only viable solution known is linear scan of data. This is because such huge amounts of data cannot be stored in main memory to run computations on. Data being produced far exceeds memory (RAM) available for algorithms to use, hence making it impossible for algorithms to remember the already scanned data.

1.3 Streaming Data

In order to cater to the challenges that Big Data brings and provide real-time analytics over it, the only viable solution is to be able to make computation in a few linear scans of data. In other words, use streaming data instead of static data. Now what is streaming data?? A data stream is an ordered sequence of data points x1,x2,x3 …,xn that must be accessed in order and that can be read only once or a small number of times. Each reading of the sequence is called a linear scan or a pass. With the proposal of streaming data comes the need to devise algorithms that could work on streaming data. Such algorithms are called streaming algorithms. In a more abstract sense, if we can make algorithms run in small spaces, it would bring algorithms one step closer to streaming algorithms.

1.4 Overview

In the next sections we will describe what is clustering. We will also study various clustering algorithms being used and describe how none of them can work for streaming data. We will also describe our approach to optimize already existing clustering techniques to work for streaming data. At the end we will compare and discuss the quality of our approach with well-known clustering technique K-Means++ and take into consideration possible future work.

2. Clustering

Clustering is a technique in machine learning that divides data into different clusters or groups based on their similarity. Data points in the same group are more similar than data in other groups. Clustering comes under the paradigm of unsupervised learning techniques, where classifies data based on hidden patterns and similarities, without pre-existing classification of data.

2.1 Metric of good quality Clustering

For most natural clustering objective functions, the optimization problems turn out to be NP hard. Therefore, most theoretical work when it comes to designing clustering algorithms is to propose approximation algorithms: algorithms that may not provide optimal solutions, but guarantee a solution whose objective function value is within a fixed factor of the value of the optimal solution. Best clustering is one where intercluster distance (distance between different clusters) is maximum and intracluster distance(average distance between data points in the same cluster) is minimum.

2.2 Clustering Techniques Known Widely

The cost function for all techniques discussed below is to minimize intercluster distance and maximize intracluster distance. This can be achieved by minimizing the loss function.

2.2.1 K-Means Clustering:

K-Means algorithm is a clustering technique which divides data points into K Different Clusters. Each cluster has approximately equal number of points. K-Means clustering uses the concept of centroid to make this division. Each cluster is represented by a centroid. The idea of the K-Means algorithm is to find k centroid points (C_1, C_1, . . . C_k) such that it minimizes the sum over each cluster of the sum of the square of the distance between the point and its centroid. In other words, it aims to reduce intra cluster distance and maximize inter cluster distance.

Challenges: K-Means clustering uses randomization to find initial centroids. The initial k-centroids are picked randomly from the data points. This randomization of picking k-centroids points results in the problem of initialization sensitivity. This problem tends to affect the final formed clusters. The final formed clusters depend on how initial centroids were picked. In other words, the K-Means algorithm guarantees no approximation to optimal solutions. In fact, if the data contains outliers, the density spread of data points across the data space is different and the data points follow non-convex shapes, K-Means algorithm performs the worst.

One of the best solutions for the challenge above is K-Means++ technique.

2.2.2 K-Means++ Clustering:

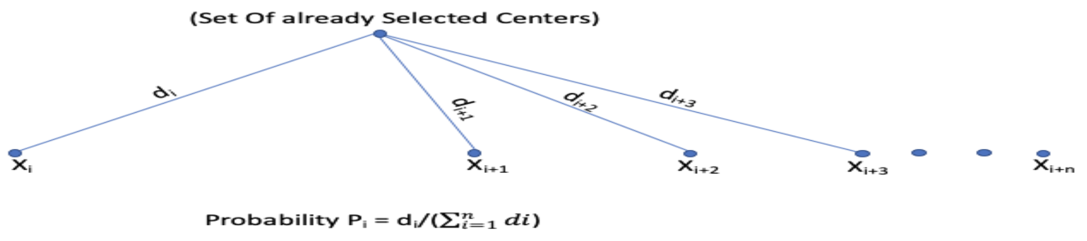
K-Means++ is the optimization over K-Means algorithms which picks initial K centroids smartly thus guaranteeing O(log(k))-approximation to the optimal solution. Here, the first step is to Pick the first centroid point (C_1) randomly. At any stage with m centroids selected, Compute distance of all points in the dataset from the selected centroids. Make the point x as the new centroid that is having maximum probability proportional to distance from selected centroids. Repeat the steps until k-centroids are selected from the data set.

Challenges: A problem with the K-Means and K-Means++ clustering is that the final centroids are not interpretable. In other words, centroids are not the actual point but the mean of points present in that cluster.

One of the best solutions for the challenge above is K-Medoids technique.

2.2.3 K-Medoids Clustering:

K-Medoids clustering is optimization over K-Means++ where final centroids selected are actual data-points. This results in making the centroids interpretable. The algorithm of K-Medoids clustering is called Partitioning Around Medoids (PAM) which is almost the same as that of K-Means++ algorithm with a slight change in the update step. In the case of K-Means we were computing the mean of all points present in the cluster. But for the PAM algorithm, updation of the centroid is different. If there are m-point in a cluster, swap the previous centroid with all other (m-1) points from the cluster and finalize the point as a new centroid that has a minimum loss.


(Set Of already Selected Centers)
d_i
d_{i+1}
d_{i+2}
d_{i+3}
X_i
X_{i+1}
X_{i+2}
X_{i+3}
X_{i+n}
Probability $P_i = d_i/(\sum_{i=1}^{n} di)$

How to pick the best value of K: The best value of K can be computed using the *Elbow method.* Here, the right value for 'K' is determined by drawing a plot between loss function vs k. Point with drastic change in plot reflects best value for k.

Combining summary of algorithms discussed:

	K-means	K-means++	K-medoids
Quality	No bound	O(log(k))-approx.	O(log(k))-approx.
Time	O(n)	N*k*d	O(k(n-k)^2)
Space	Linear	Linear	Linear

3. Streaming algorithm for clustering

Streaming algorithms as mentioned above, should be able to run using limited space. More precisely, they should be able to do computation using one or few linear scans of data. This is possible only if the algorithm uses sup-linear space of input data. In all of the above algorithms discussed, space isn't sub-linear. So, our goal in this paper will be to devise a streaming algorithm in the domain of clustering. Although few such algorithms have already been proposed, there is always a compromise in quality of clustering. K-means++ is a well-known clustering algorithm with one of the best qualities of objective function. We have provided a clustering algorithm that optimizes K-means++ objective by using only three-passes that provide approximately the same quality as that of k-means++. So keeping the same properties as K-means++, we have tried to optimize it to be able to run on streaming data. After proposal, quality of two algorithms was found to be approximately the same based on empirical evidence whose results have been mentioned in Paper. In short, We provide a clustering algorithm in the three-pass streaming setting that provides approximately the same quality as that of k-means++(that uses Linear space).

3.1 K-Means++ and its challenges

There are various clustering algorithms that use different techniques and thus have different quality and use different amounts of space and time. Our algorithm provides sublinear space by using a three-pass streaming setting that provides approximately the same quality as that of k-means++(On the basis of empirical evidence). We use the idea of pre-selecting centres as in K-Means++ but in just 3 linear scans of data. Problem with K-Means++ was that, at every stage of selecting a new centre, the distance of "every point" is calculated from the selected centres. It was basically assigning a probability to every data point for getting selected as a new centre. Probability was directly proportional to distance of point from all centres selected divided by sum of distance of all points from all centres selected. In K-Means++, Probability function to select new point as centre is directly proportional to: -

$\mathbf{d_i = \min_{(j : 1 \to m)} ||x_i - C_j||^2}$

d_i: distance of x_i from nearest centres

m: Total No. of centres selected

This necessitates the requirement of presence of all points in memory at the same time. How can this be modified to work for streaming data??

3.2 Solution Proposed

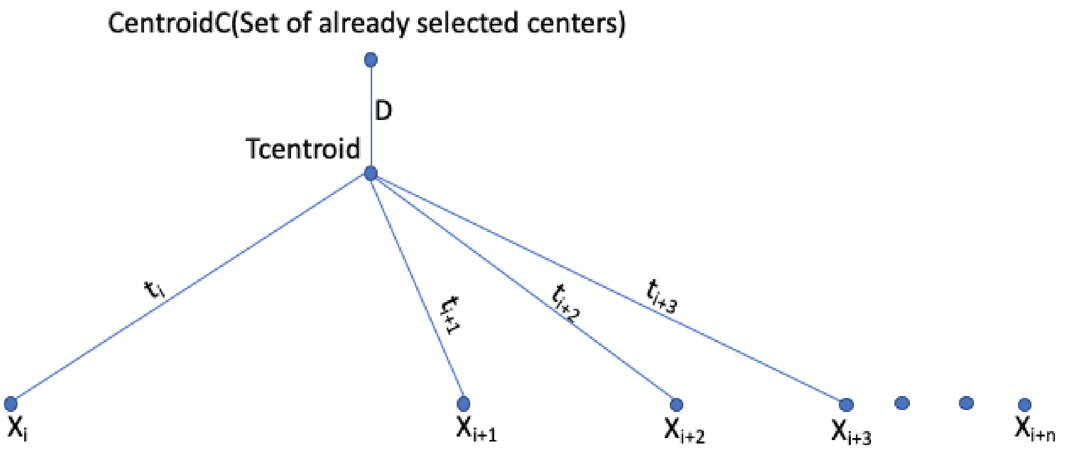
This can be solved if we understand K-Means++ in more abstract form. Problem above is that, to calculate probability at any point, we need to calculate the distance of all points from centres selected. If we could propose a way to find the distance of any point from centres, without having to store the point information, we could end up devising a streaming algorithm. Using the very same idea, We in our algorithm use the idea of Tcentroid: Centroid of all data points that can be calculated in 1 linear scan of data. Now in order to get the distance of any point x_i from centre, we proposed to compute the distance of every point from Tcentroid (t_i), which can be done in 1 linear scan of data as well. Also consider CentroidC: Centroid of all centres selected at any given time. Now the distance of any point(x_i) from centres will be [t_i + distance of t_i from CentroidC]. As we can see here, we do not need information of point itself, i.e., (x_i) to calculate distance of point from centroids.

Now probability of new data point(xi) being read will be directly proportional to the distance of xi from CentroidC. But this marks the challenge of selecting outliers as centroids. In order to solve this problem in a streaming algorithm way, we introduced the idea of Tcentroid: Tcentroid is the centroid of all data points that can be calculated in 1 linear scan of data. In essence, Tcentroid represents data points as a whole. So, adding distance of point xi from Tcentroid in its probability to get selected will help in picking points closer to the general population of data points thus avoiding outliers. In our algorithm, Probability function to select new point as centroid is proportional to: -

$\mathbf{d_i = t_i + D}$

t_i : distance of Tcentroid from point x_i

D: Distance of Tcentroid from CentroidC(Set of already selected centers)



Probability $P_i = (t_i+D)/(\sum_{i=1}^{n}(ti + D))$

3.3 Our Algorithm

```
Pass1:
    ❖ Centroid of all the m points is calculated, say Tcentroid

Pass2:
    ❖ Distances of all the m data points from the Tcentroid calculated and stored in an array say DAT

Pass3
    ❖ Define C //Array of centres
    ❖ Define CentroidC //Centroid of centres set C
    ❖ While (Till all the points are covered)
 For each point X1 in Data points:
 Select point X1 as centre with a probability of: $(d1 + d2)/ (\Sigma [d1 + d2])$

 Where, d1= distance of point X1 from Tcentroid
 d2= distance of point Tcentroid from CentroidC

 If point X1 gets selected as a new centre then update array $C \rightarrow C + \{X1\}$
 Update $CentroidC \rightarrow (n*CentroidC + X1)/ (n+1)$
 //n= no. of centres selected till prev.step
    ❖ EndWhile
Return C
```

Clearly the algorithm makes only three passes through the dataset and uses sublinear space of the order $O(m)$. (where m is the number of data points). And thus, the algorithm is a 3-pass streaming algorithm

It is an approximation algorithm to K Means++ as every point is assigned a probability to get selected as a centre. And larger the distance from the centroid of centres, larger will be the probability of the point to get selected.

3.4 Drawbacks

The algorithm proposed does not guarantee to select a given(k) number of centroids because probability is used to select any data point as a new centroid. To overcome the problem, we can add a multiplication factor to the probability function. More the multiplication factor, more centroids selected. For example, if our algorithm is selecting c centroids from n data points opposed to k no. of centroids we wanted the algorithm to select, then add multiplication factor of (k/c).

4. Experiments

In this section we will describe the results of testing we performed to compare our algorithm against K-Means++ and their outcome. As we shall see, the results are very encouraging, showing that the streaming clustering algorithm proposed in this paper is appropriate for clustering guaranteeing approximately the same quality of clustering as K-Means++. In order to make substantiated results, we have compared 2 algorithms for various values of k and the results have been shown below.

Data set used for running and comparing our algorithm with k-means++ is taken from the Stanford Machine Learning course by Andrew Ng. (*https://www.coursera.org/learn/machine-learning*)

4.1 Comparison with K-means++

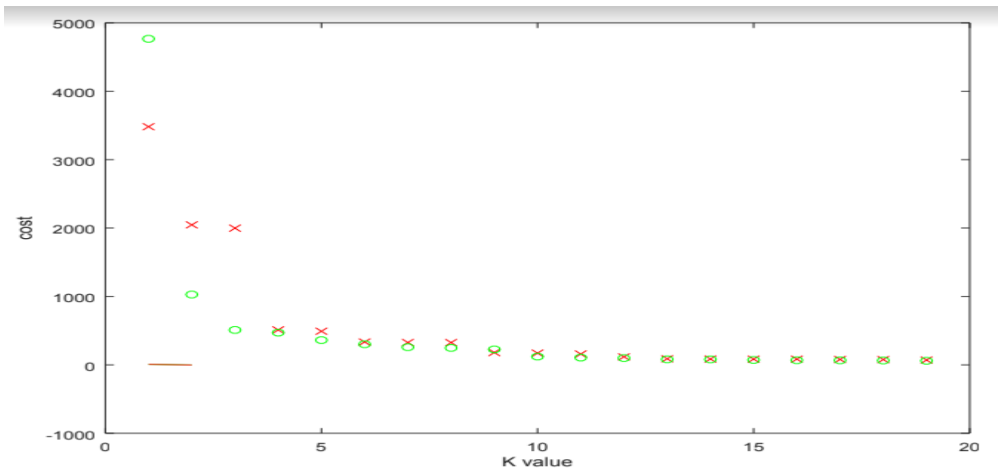
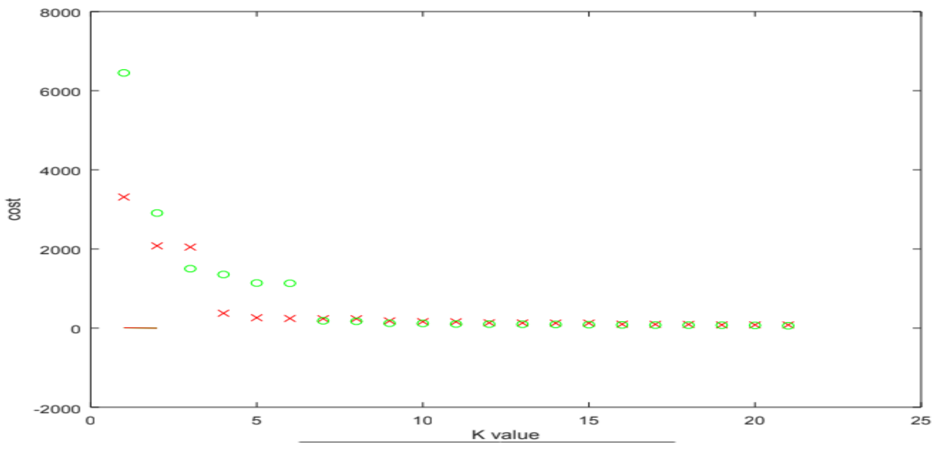
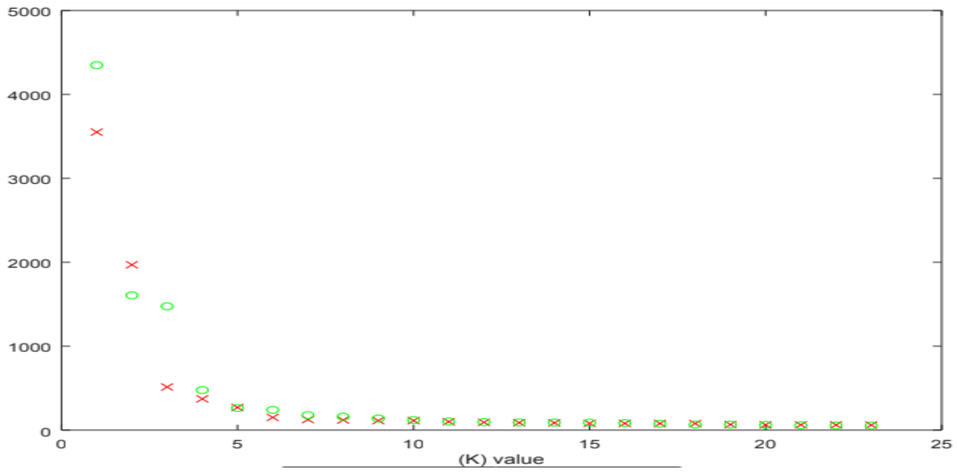
We tested the quality of our algorithm by running our algorithm and K-means++ on a data set mentioned above and plotted the (cost vs K) and (log(cost) vs log(k)) plots (where cost is the average of sum of distances of the points from the nearest assigned centre and K represents number of centres).

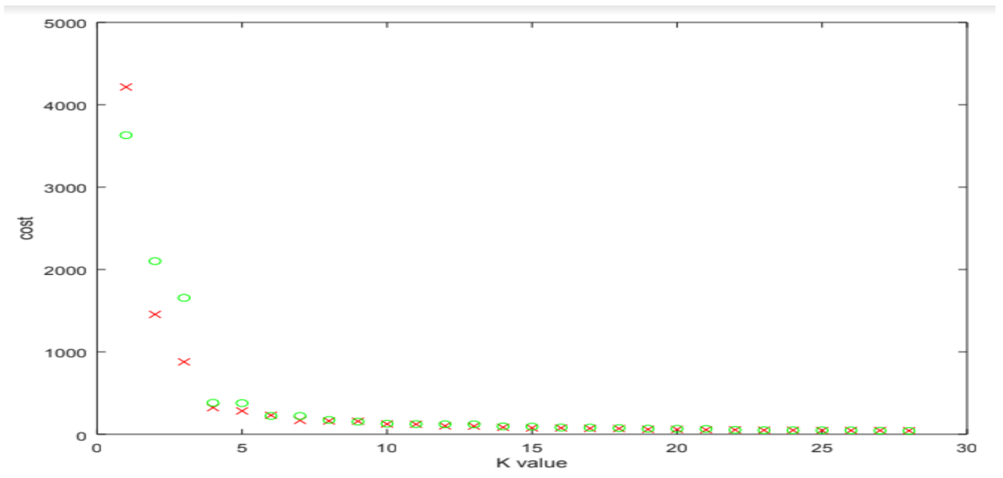
4.2 Plots

RED plot - KMeans++

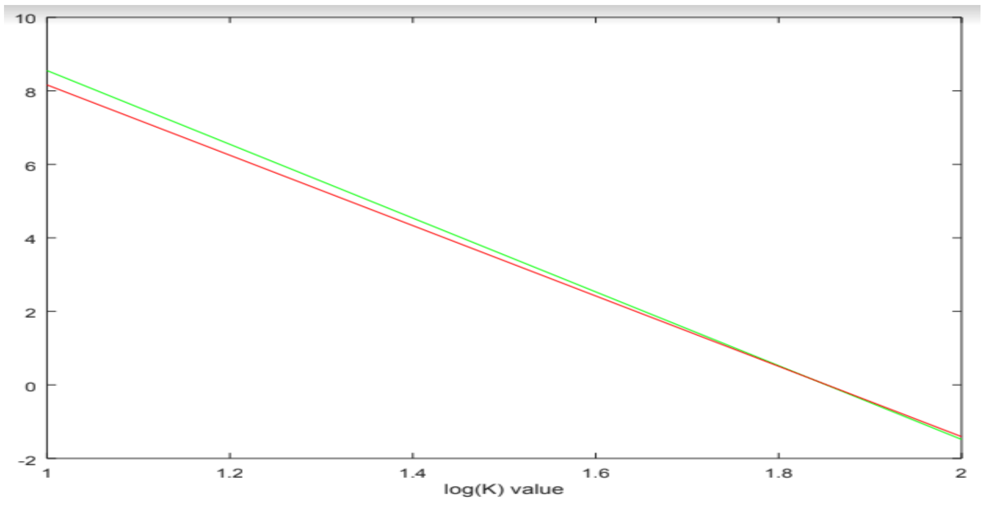
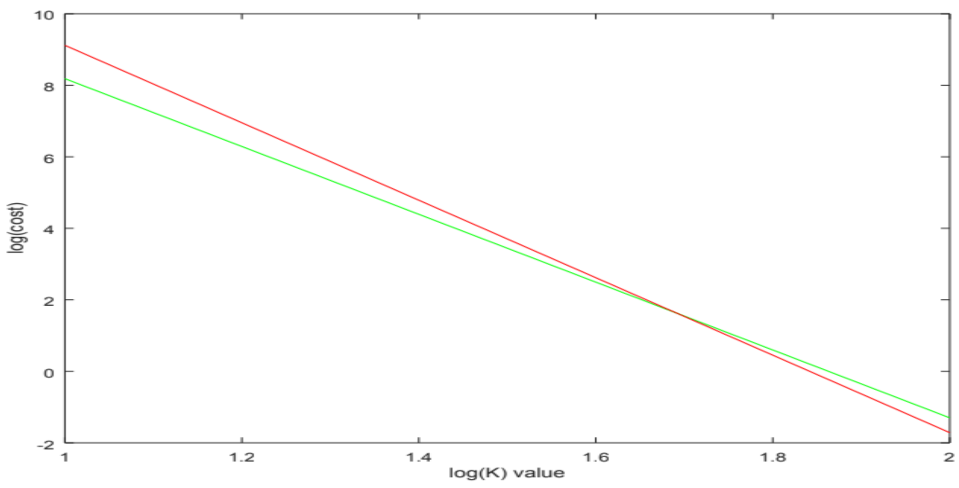
GREEN plot - Our Implemented algorithm

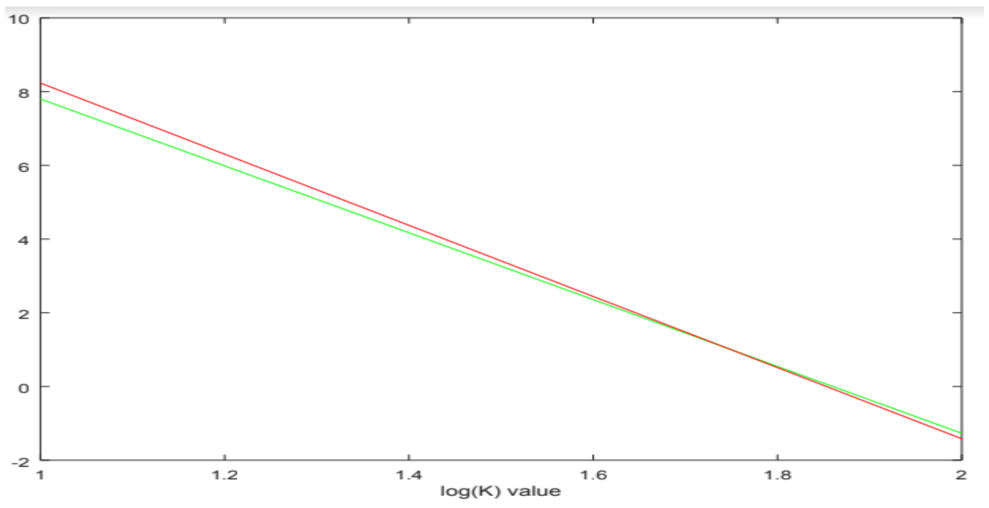
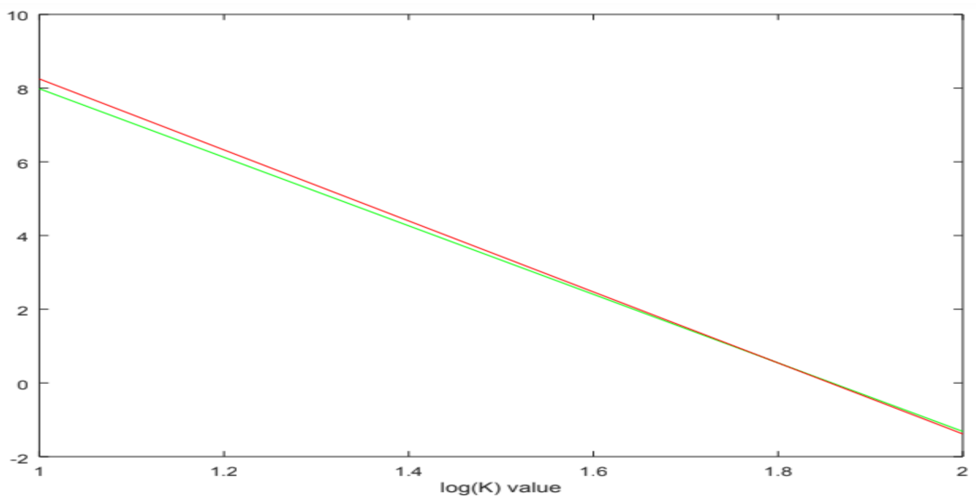
4.2.1 Cost vs K for different values of centres selected

	K = 19
	K = 21
	K = 23

	K = 28

4.2.2 Log(cost) vs Log(K)

	K = 19
	K = 21

	K = 23
	K = 28

It can be seen from the above plots that nature is quite similar to that of K Means++.

5. Results and conclusions

Motive of this paper was to devise a streaming algorithm in the domain of clustering due to increasing demand of such algorithms in the Big Data era that prevails today. As we can see from the experimental results, by using appropriate objective functions that can be updated in real time with every new data point read can lead to an intelligent clustering technique that works on streaming data with almost the same quality as K-Means++. In this paper we successfully provided a clustering algorithm that optimises K-means++ objective by using only three-passes that provide approximately the same quality as that of k-means++.

We can very well conclude that optimizing existing clustering techniques to work as streaming algorithms is very well possible. However, only empirical evidence was studied while making such a conclusion. In order to make firm conclusion results need to be theoretically proven.

5.1 Future work

Future work should be to mathematically work out proof for the quality of the algorithm proposed to be as good as K-Means++ algorithm as concluded from empirical evidence. Also, the algorithm proposed does not guarantee to select k centroids in one pass as probability is used to select any data point as a new centroid. So future work could be to optimize probability functions to guarantee selection of best points as centroids.

REFERENCES

[1] Nir Ailon, Ragesh Jaiswal, Claire Monteleoni: Streaming k-means approximation

[2] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, Liadan O'Callaghan (January 14, 2003): Clustering Data Streams- Theory and Practice

[3] M. Ankerst, M. Breunig, H. Kriegel, and J. Sander. Optics (In Proc. SIGMOD, 1999): Ordering points to identify the clustering structure.

[4] David Arthur and Sergei Vassilvitskii (SODA, 2007): k-means++: the advantages of careful seeding.

[5] Mahendra Tiwari, Randhir Singh (November 2012): Comparative Investigation of K-Means and K-Medoid Algorithm on Iris Data

BIOGRAPHIES

Tanuj Gupta, currently working as full stack developer at Airtel X Labs is a Driven SDE with experience in setting up big data and analytics systems, their maintenance and working. Graduated with Bachelor of Engineering (B.E.) in Computer Engineering from Bits Pilani, has keen interest in data science and designing microservice workflows for backend technologies. Currently interested in a new role of designing and developing impactful services and products.

Tinish Gupta, currently working as Commodity Trader at Futures First Info. Services, is an Experienced Commodity Trader with a demonstrated history of working in the capital markets industry. Skilled in Data analytics, Python, Java, Microsoft Excel, leverages technology to solve problems in finance. Strong finance professional with a Bachelor of Technology (B.Tech.) in Civil Engineering from Indian Institute of Technology, Delhi.