Streaming Algorithm for k-median Dynamic Geometric Problem

Jai Mohan Shrivastava

A Thesis Submitted to
Indian Institute of Technology Hyderabad
In Partial Fulfillment of the Requirements for
The Degree of Master of Technology

Department of Computer Science and Engineering

June 2012

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

Jaimohan

(Signature)

(Jai Mohan Shrivastava)

CS10M02

(Roll No.)

Approval Sheet

This thesis entitled Streaming Algorithm for k-median Dynamic Geometric Problem by Jaimohan Shrivatava is approved for the degree of Master of Technology/ Doctor of Philosophy from IIT Hyderabad.

ch sobhan babu

ch sobhan babu

-Name and affiliation-
Examiner

-Name and affiliation-
Examiner

-Name and affiliation-
Adviser

[illegible]

-Name and affiliation-
Chairman

Acknowledgements

I express my sincere gratitude toward my adviser Dr. Naveen Sivadasan for his constant help, encouragement and inspiration throughout the thesis work. Without his invaluable guidance, this work would never have been a successful one. I would also like to thank Prof. Dr. Ravindra N. Guravannavar and Mr. Sandeep R.B(Phd. Scholar) for their valuable time and suggestions.

Abstract

Many applications such as financial transactions data, customer click stream continuously generates huge data sets at very rapid rate. This is generally termed as data stream. These data sets are too huge to store on limited secondary storage. Therefore, it directs us to adopt a technique to maintain the sketch or summary of the data stream. This will allow us to answer any query, that we wish to perform on this data sets, approximately. Many applications on data streams such as counting of distinct items, estimating frequency moments [7, 8], the counting of frequent items [9, 3], clustering and the computation of histograms are of great interest among researchers. We consider k-median clustering problem in the geometric data stream. The task of the clustering is to partition a set of objects into disjoint group wherein the data objects in the same group are similar and objects in the different groups are dissimilar.

We consider the k-median clustering algorithm in the context of data stream as proposed in [4]. We propose modified algorithm for 2-dimension that achieve improved time and space bounds for k-median problem in [4]. We run the experiment by implementing the modified algorithm to see execution time.

Contents

Chapter 1

Introduction

1.1 Overview: Data stream and Streaming algorithm

A data stream is an ordered sequence of points $x_1, x_2, ..., x_n$ that must be accessed in order and can be read only once or a small number of times [4]. The data stream model is highly motivated by applications which involve massive datasets e.g. datasets of computer network traffic, ATM transaction, web searches, financial transaction, sensor network. In the data stream model, the whole input dataset is not readily available to the algorithm, but rather arrives as one or more continuous data streams at regular interval. Storing the entire data is not possible due to the limitation of space(main memory). Other applications such as mining of stock price, transactional log analysis, etc. having huge datasets to be operated on are typically stored in secondary storage devices where random access is very expensive and the only cost-effective access method is linear scan(although more than one scan is allowed), these applications can also be modeled as data stream. Owing to these constraints of space and time, many researchers have focused on designing data structure and algorithm in data stream model. Since the amount of memory space available to the data stream algorithm, perhaps in polylogarithmic of input size, is far too small to store the whole input. Therefore, algorithm must maintain sketches or summaries of data.

Streaming algorithm merely maintains the sketch of the input data, therefore it would be impossible to answer a query precisely and so some approximation is provably needed with usable accuracy guarantees. Typically, accuracy guarantees will be made in terms of a pair of user specified parameters, ϵ and δ such that the error in answering a query is within a factor of ϵ with probability δ. Many streaming algorithms and data structures have been proposed for computation of simple numerical statistics of the input, like finding clusters, number of distinct elements, frequency moments, heavy hitters, etc.

1.2 Clustering: k-median in Data Stream Model

Clustering algorithm arranges a data set into several disjoint groups such that objects in the same group are similar to each other and are dissimilar to other groups according to some similarity measures[10]. Clustering has very rich application in various field such as Social Network Analysis wherein clustering may be used to recognize communities within large groups of people, Market Research, Image Segmentation wherein Clustering can be used to divide a digital image into distinct regions for border detection or object recognition.

In this thesis, we are considering k-median clustering problem for dynamic geometric problems, wherein the goal is to find an efficient algorithm for estimating the median cost repeatedly (or at particular interval) after each incremental modification of the input data i.e. under the addition and deletion of input geometric objects. For more clustering problems see [12, 13]. From the data stream perspective, the stream consist of either Add(p) or Remove(p) operations(adding p object to the data structure or deleting p object from the data structure). The set of geometric objects P lie in the discrete 2-dimension space $\{1...\Delta\}^2$. Only k-median and the cost of the solution are reported rather than reporting the solution itself. Otherwise, the size of the solution would be linear of input size which makes it impossible to design algorithms with polylogarithmic space. K-median clustering objective is to identifying k-centers so that the sum of the distances from each object to its nearest neighbor is minimized. Let Q and P be the set of medians and set of points respectively and C(Q,P) be the objective cost then,

$$C(Q,P) = \sum_{p \in P} \underset{q \in Q}{min} ||p - q||$$

We will assume that the input points P are all distinct and distributed over the 2-dimensional discrete space $\{1...\Delta\}^2$. All the theoretical results for the time and space complexity have been proved under the assumption that dimension d=2 and $\log n = O(\log \Delta)$, where n is the maximum number of elements in the set P. All of the algorithms are randomised and use the space polynomial in $\log \Delta$.

1.3 Discrete Geometric Space

As aforementioned, the input points live in a discrete space $\{1...\Delta\}^2$. The reason behind taking the discrete space is to ease the computational complexity analysis, for in the real space R^2 with the infinite precision, the notion of the storage is not well defined. However, in practice its common assumption that the input elements in geometric problems take real values. So, we mention that the algorithm can be easily adapted to work for real space with bounded precision wherein the minimum inter-point distance is at least 1 and the diameter is bounded by Δ [4].

1.4 Outline: Finding k-median Cost in Data Stream Model

In a brief, the general idea for estimating the k-median Cost approximately in a data stream model is implemented as follows. Impose $\log \Delta$ randomly shifted, nested square grid over the point space. The grid cells have side length 2^i for $i = 0 ... \log(\Delta) - 1$. For each grid, compute certain statistic of the distribution of points in the grid. Let $n_P(c)$ be the number of points in set P falling into cell c. At level i, given a set of medians Q of size k, consider a ball $B(q, r_i)$ of radius r_i around points $q \in Q$. Let $B(Q, r_i)$ be the union of all balls. Let $XC_i = \sum_{c \cap B(Q, r_i) = \emptyset} n_P(c)$ be the number of points lying outside $B(Q, r_i)$ i.e. Exclusive Count . Then the k-median cost is calculated as:

$$Cost = \sum_i XC_i * (r_i - r_{i-1})$$

Then we find an approximately optimal set $Q \subset \{1...\Delta\}\, 2d$ via exhaustive search or local search.

1.5 Thesis Outline

The rest of the thesis is outlined as follows: In chapter 2, we briefly discuss the tools like point query using Count-Min sketch, range query using dyadic range and so on. These tools act as building blocks to the algorithms. In chapter 3, we give overview of the streaming algorithm for k-median clustering problem proposed by Piotr Indyk in [4]. In chapter 4, we propose modified algorithm to the k-median clustering problem [4], that achieves better space and time bounds. In chapter 5, we perform the experiment for both the algorithms for 2-dimension and compare the results. We conclude the thesis in chapter 6.

Chapter 2

Preliminaries

2.1 Count-Min Sketch

2.1.1 Overview

Consider a vector $A\,(1...n)$, which is presented in an implicit, incremental fashion. Initially, vector A is set to zero; A(i)=0 for $1 \leq i \leq n$. The vector A is, then, continuously presented with updates as a stream of pairs (i, c), meaning that the value of vector at index i is added by value c i.e. A(i)=A(i)+c. In some cases, c is be strictly positive and values of A(i) always increase. It is known as cash register model. In other cases, c is allowed to be negative also and some value A(i) might be negative. This is called turnstile model. But if no A(i), at any time, is less than zero i.e. $A(i) > 0$ for $1 \leq i \leq n$, even though c is allowed to be negative, then it is called strict turnstile model. The term Point Query, denoted Q(i), is defined as the value of A(i) at the particular instant.

Since the space available is merely polynomial in $\log(n)$, the algorithm will only be able to store the sketch or summary of the vector A and hence no function or query, to be computed on A, can be done precisely. So some approximation is provably needed in answering the query with usable accuracy guarantees. Typically, accuracy guarantees will be made in terms of a pair of user specified parameters, ϵ and δ such that the error in answering a query is within a factor of ϵ with probability δ. The space and update time will consequently depend on ϵ and δ.

Count-Min sketch, [3], is a probabilistic sub-linear space streaming algorithm that has been extensively used for summarizing data streams. The algorithm was invented in 2003 by G Cormode and S. M Muthukrishnan. This sketch allows fundamental queries in data stream summarization such as point, range, and inner product queries to be approximately answered very quickly. Generally, the accuracy estimates for the queries answered using CM sketch depend on the L1 norm of the vector. Here we investigate the time, space and Update time complexity for performing the point query using CM sketch. In order to provide tighter complexity bounds, Markov inequality is used.

2.1.2 Point Query

A Count-Min sketch with parameters (ϵ, δ) is represented by a two-dimensional array with width w and depth d called here *count*. Given parameters (ϵ, δ), set $w = \lceil \frac{e}{\epsilon} \rceil$ and $d = \lceil \log(\frac{1}{\delta}) \rceil$. Each entry of the array is initially zero. Additionally, d hash functions are chosen uniformly at random from a pairwise-independent hash family, viz, h_j for $j \in \{1, ..., d\}$. The CM sketch data structure supports following operations.

• **Update (i,c):** $\forall j$, $count[j][h_j(i)] \leftarrow count[j][h_j(i)] + c$

• **Point Query Q(i):** return $\hat{A}_i = min_j \; count[j][h_j(i)]$

The total space used by Count-Min sketches is $wd + 2d$ words, where wd words for storing the array and 2d words for d hash function each of which requires 2 words. we will only analysis the point query for the non-negative case(strict turnstile case) i.e. all $A(i) > 0$ at any time.

Theorem 2.1.1 ([3]). *This estimate has the guarantee that $\hat{A}(i) \leq A(i) + \epsilon ||A||_1$ with probability $1 - \delta$.*

Proof. Fix any j $\in \{1, ..., d\}$, the value stored at $count[j][h_j(i)]$,

$$\begin{aligned} count[j][h_j(i)] &= A(i) + \Delta \\ E(\Delta) &= \sum_{t=\{1,...,n\} \text{ and } t \neq i} Pr[h_j(t) = h_j(i)]A(t) \\ &\leq \frac{1}{w} \sum_{t=\{1,...,n\}} A(t) = \frac{1}{w}||A||_1 \\ &\leq \frac{\epsilon}{e}||A||_1 \end{aligned}$$

By Markov inequality,

$$Pr[\Delta \geq \epsilon * ||A||_1] = Pr[\Delta \geq e * E(\Delta)] \leq \frac{1}{e}$$

For all $j \in \{1, ..., d\}$,

$$Pr[\forall_j \Delta \geq \epsilon * ||A||_1] = Pr[\forall_j \Delta \geq e * E(\Delta)] \leq (\frac{1}{e})^d = \delta$$

Therefore, A(i) $\leq \hat{A}(i) \leq A(i) + \epsilon * ||A||_1$ with probability atleast $1 - \delta$. □

The time to compute the estimate is $O(\log(\frac{1}{\delta}))$. The space used is $O(\frac{e}{\epsilon} * \log(\frac{1}{\delta}))$. Update time is $O(\log(\frac{1}{\delta}))$.

2.1.3 Range Query

Given two end points l and r, both positive and less than or equal to n, the range query denoted as Q(l,r) is defined as sum of all A(t) for $l \leq t \leq r$, which can be obtained by simply computing point queries for each item in the range, and summing the estimates. However, the error guarantee could be as large as the factor of $||A||$ and n i.e. $\hat{Q}(l, r) \leq Q(l, r) + \epsilon n ||A||_1$, which is very huge because n itself is very large. To avoid this, the new technique is adopted wherein the vector A is stored using $\log(n)$ CM sketches with dyadic ranges of length 2^y each for $y = 0, ..., \log(n) - 1$. Each point in the range [1...n] is a member of $\log(n)$ dyadic ranges, one for each $y = 0... \log(n) - 1$. For each input pair (i,c), update to the data structure is similar to the updation of CM sketch, except here the updation is done at every level of CM sketch for $y = 0 \ldots \log(n) - 1$ after finding the new index i_y, given as $i_y = \frac{i}{2^y}$. Then, given a range query Q(l,r), compute the at most $2\log(n)$ dyadic ranges which canonically cover the range, and pose that many point queries to the sketches, returning the sum of the queries as the estimate. For example, consider n = 256, the range [48, 107] is canonically covered by the non-overlapping dyadic ranges [48...48], [49...64], [65...96], [97...104], [105...106], [107...107].

Let $A[l,r] = \sum_{k=l}^{r} A(k)$ be the answer to the query Q(l,r) and let $\hat{A}[l,r]$ be the approximate value of $A[l,r]$.

Theorem 2.1.2 ([3]). *$A[l,r] \leq \hat{A}[l,r]$ and $\hat{A}[l,r] \leq A[r,l] + \epsilon||A||_1$ with probability at least $1-\delta$ for $w = \lceil \frac{2e\log(n)}{\epsilon} \rceil$ and $d = \lceil \log(\frac{1}{\delta}) \rceil$.*

Proof. Consider a dyadic level y with hash function h_j mapping $\{1, ..., \alpha\} \rightarrow \{1, ..., w\}$, $\alpha = \frac{n}{2^y}$. Consider i^{th} subrange, fix any hash function j. The value of A(i) stored at $count[j][h_j(i)]$ is,

$$\begin{aligned}
count[i][h_j(i)] &= A(i) + \Delta \\
E[\Delta] &= \sum_{t=1\ldots\alpha \text{ and } t\neq i} Pr[h_j(i) = h_j(k)]A(t) \\
&\leq \frac{1}{w} \sum_{t=\{1,\ldots,\alpha\}} A(t) \\
&\leq \frac{1}{w}||A||_1
\end{aligned}$$

For total $2\log(n)$ point query,

$$\begin{aligned}
E[\Delta_{total}] &\leq \frac{2\log(n)}{w}||A||_1 \\
E[\Delta_{total}] &\leq \frac{\epsilon}{e}||A||_1
\end{aligned}$$

hence, by Markov inequality,

$$Pr[\Delta_{total} \geq \epsilon||A||_1 = eE[\Delta_{total}]] \leq \frac{1}{e}$$

For all $j \in 1...d$,

$$Pr[\forall_{j\in\{1,\ldots,d\}} \Delta_{total} \geq \epsilon||A||_1 = eE[\Delta_{total}]] \leq (\frac{1}{e})^d \approx \delta$$

Since Δ_{total} is non-zero, so clearly $A[l,r] \leq \hat{A}[l,r]$ and $\hat{A}[l,r] \leq A[l,r] + \epsilon||A||_1$ with probability at least $1-\delta$.

The space used is $O(\frac{\log^2(n)}{\epsilon} \log \frac{1}{\delta})$. The time to estimate the query is $O(\log n \log \frac{1}{\delta})$. □

Chapter 3

Streaming Algorithm for k-median Dynamic Geometric Problem

3.1 Overview

In this chapter, we give the overview of k-median clustering algorithm proposed by Piotr Indyk in [4] for 2-dimension. K-median clustering objective is to identifying k-centers Q so that the sum of the distances from each point to its nearest neighbor is minimized. Let $Q \subset \{1, ..., \Delta\}^2$.

$$C(Q, P) = \sum_{p \in P} \min_{q \in Q} ||p - q||$$

The main focus is on estimating the cost of the solution, rather than reporting the solution itself, because the solution size could be as large as input size which makes it impossible to design algorithm with poly-logarithmic space. To be precise, the algorithm only provides the k-median cost to the given k Centres. Subsequently, this algorithm can be used as a subroutine to the various approaches such as exhaustive search, local search, greedy algorithm, etc. to find the best k Centres. Note that the k centres could belong, not just from the set P, also from the space $\{1, ..., \Delta\}^2$.

3.2 Input Format

The algorithm sees the input in the following format:

$$input(id, old\text{-}coordinate, new\text{-}coordinate)$$

3.3 Exclusive Count

It is low-space data structure that maintains the vector $x[1...M]$ using data structure akin to *Min-Count sketches* [3] using a pair-wise independent hash function $h : \{1, ..., M\} \rightarrow \{1, ..., w\}$, for $w = 2t(1 + 1/\epsilon)$ and an array $count[1...w]$, where t is the maximum number of point query i.e. $|Q| \leq t$. It supports the following operations:

- **Update(i,c):** count[i] += c
- ***XCount*(Q):** return $R = \sum_{i \notin h(Q)} count[i]$

LEMMA 1 ([4]). Any *XCount* query reports a correct (approximate) value R with probability at least 1/2.

Proof. Let $XC = \sum_{i \notin Q} x[i]$ be the original value for Exclusive Count and $\Delta = \sum\limits_{j \notin Q \text{ and } h(j) \in h(Q)} x[j]$. Clearly $XC = R + \Delta$.

$$\begin{aligned}
E(\Delta) &= \sum_{j \notin Q \text{ and } h(j) \in h(Q)} Pr[h(j) \in h(Q)] * x[j] \\
E(\Delta) &\leq t * \frac{1}{w} XC \\
&= \frac{\epsilon}{2(1+\epsilon)} XC \\
&\leq \frac{\epsilon}{2} XC
\end{aligned}$$

Using Markov inequality,

$$Pr[\Delta \geq \epsilon XC] = Pr[\Delta \geq 2E[\Delta]] < \frac{1}{2}$$

Therefore, $R \geq (1-\epsilon)XC$ with probability at least $\frac{1}{2}$.

□

3.4 Median Cost Evaluation

This tool called *MediEval* provides the function to calculate k-median cost by using the underlying data structure *XCount*(Q), under the addition and deletion of points. It also supports an operation $EVAL(Q)$, that returns cost C such that $C = (1 \pm O(\epsilon))C(Q,P)$. The data structure is parametrized by l and $|Q| \leq l$. Let G_i be the grid of size length $\frac{\epsilon}{\sqrt{2}}(1+\epsilon)^i$ imposed on space $\{1, ..., \Delta\}^2$. Let n_i is the vector which is stored by the *XCount* data structure to solve Exclusive Count problem and $n_i(c)$ to be number of points in P that fall into cell c of grid G_i. Remember, the range of the hash function in *XCount*(Q) is a function of t, set $t = \Theta(\frac{l}{\epsilon^2})$. Let $B(q, r_i)$ be a set of points from the space $\{1, ..., \Delta\}^2$ within the radius r_i from q. Let $B(Q, r_i) = \bigcup_{q \in Q} B(q, r_i)$. Set $r_i = (1+\epsilon)^i$. Let $i_0 = O(\log(1/\epsilon)/\epsilon)$ such that $r_{-i_0} < \epsilon$. Let $G_i(B(Q, r_i))$ be the set of grid cells that contain points in $B(Q, r_i)$, observe that for any i,

$$B(Q, r_i) \subset G_i(B(Q, r_i)) \subset B(Q, r_{i+1})$$

Note that $|G_i(B(Q, r_i))| = O(\frac{l}{\epsilon^2}) \leq t$. To achieve this, divide the area of ball $B(q, r_i)$ $\forall q \in Q$ with area of one cell. Let $\hat{R}_i(Q)$ be the value of *XCount* for $G_i(B(Q, r_i))$ and let $R_i(Q)$ be the approximation of $\hat{R}_i(Q)$ and that $R_i(Q) \geq (1-\epsilon)\hat{R}_i(Q)$, and

$$|P - B(Q, r_i(1+\epsilon))| \leq \hat{R}_i(Q) \leq |P - B(Q, r_i)|$$

.

Define $\hat{C}(Q,P) = \sum_{-i_0 \leq i \leq \log(2\Delta)} (r_i - r_{i-1})\hat{R}_i(Q)$

LEMMA 2 ([4]). The quantity $\hat{C}(Q,P)$ provides a good approximation of C(Q,P) [4]. That is $C(Q,P) = (1 \pm O(\epsilon))\hat{C}(Q,P)$

Proof. In paper [4], only $\leq$ inequality is given. Here, we give the proof for both $\leq$ and $\geq$ inequality.

For $\leq$ inequality:

$$\begin{aligned}
C(Q,P) &= \int_0^\infty |P - B(Q,r)|dr \\
&\leq \sum_i |P - B(Q, r_{i+1})|(r_{i+2} - r_{i+1}) \\
&\leq \frac{1}{1-\epsilon} \sum_{i \geq -i_0} |P - B(Q, r_{i+1})|(r_{i+2} - r_{i+1}) \\
&\leq \frac{(1+\epsilon)^2}{1-\epsilon} \sum_{i \geq -i_0} |P - B(Q, r_{i+1})|(r_i - r_{i-1}) \\
&\leq \frac{(1+\epsilon)^2}{1-\epsilon} \sum_{i \geq -i_0} \hat{R}_i(Q)(r_i - r_{i-1}) \\
&\leq \frac{(1+\epsilon)^2}{1-\epsilon} \sum_{-i_0 \leq i \leq \log(2\Delta)} \hat{R}_i(Q)(r_i - r_{i-1})
\end{aligned}$$

For $\geq$ inequality:

$$\begin{aligned}
C(Q,P) &= \int_0^\infty |P - B(Q,r)|dr \\
&\geq \sum_i |P - B(Q, r_i)|(r_{i+1} - r_i) \\
&\geq \frac{1}{1-\epsilon} \sum_{i \geq -i0} |P - B(Q, r_i)|(r_{i+1} - r_i) \\
&\geq \frac{1+\epsilon}{1-\epsilon} \sum_{i \geq -i0} |P - B(Q, r_i)|(r_i - r_{i-1}) \\
&\geq \frac{1+\epsilon}{1-\epsilon} \sum_{i \geq -i0} \hat{R}_i(Q)(r_i - r_{i-1})
\end{aligned}$$

□

The total space required to estimate median cost is $O(\frac{k}{\epsilon^3}(\log\Delta + \frac{\log(1/\epsilon)}{\epsilon}))$. The query time, given $Q \subset \{1, ..., \Delta\}^2$, is $O(\frac{k}{\epsilon^2}(\log\Delta + \frac{\log(1/\epsilon)}{\epsilon}))$. Update time is $O(\log\Delta + \frac{\log(1/\epsilon)}{\epsilon})$.

3.5 Algorithm : Exhaustive Search and Local Search for 2-Dimension

The data structure *MediEval* keeps the information about the distribution of the points $P \in \{1, ..., \Delta\}^2$. The *MediEval* data structure also provides the function to approximately estimate the median Cost C(Q,P) to any $Q \subset \{1, ..., \Delta\}^2$, probability atleast $\frac{1}{2}$. This data structure can be used in several approaches, such as Exhaustive Search or Local Search, to find the best k-medians. The total space used by *MediEval* is $O(\frac{k}{\epsilon^3}(\log \Delta + \frac{\log(1/\epsilon)}{\epsilon}))$. The time to estimate the k-median Cost, given $Q \subset \{1, ..., \Delta\}^2$, is $O(\frac{k}{\epsilon^2}(\log \Delta + \frac{\log(1/\epsilon)}{\epsilon}))$.

The Exhaustive Search algorithm solves the K-median problem by enumerating all sets $Q \subset \{1, ..., \Delta\}^2$ and choosing the best. Although the time taken is in the factor of $\Delta^{O(k)}$ which is prohibitively expensive, but it shows that, it is possible to get $(1 + \epsilon)$-approximate solution to the k-median problem. The time needed to report the solution, i.e. optimal k centre's, is $O(\frac{\Delta^k k}{\epsilon^2}(\log \Delta + \frac{\log(1/\epsilon)}{\epsilon}))$.

The Local Search algorithm initially chooses an arbitrary set Q of size k. Then, at any step, it enumerates all sets $Q\prime = Q - \{q\}\{p\}$ for $q \in Q, p \notin Q$. If $C(Q\prime, P) < (1 - \alpha)C(Q, P)$, then Q becomes $Q\prime$. The algorithm ends when there is no $Q\prime$ satisfying the above condition. The time needed to report the solution, i.e. optimal k centre's, is $O(\frac{\Delta^2 k}{\epsilon^2}(\log \Delta + \frac{\log(1/\epsilon)}{\epsilon}))$. Arya [11] et al show that for the case $\alpha = 0$, the algorithm provides a 5-approximation.

Chapter 4

Improved Space and Time Bounds for K-median Problem

4.1 Overview

In this chapter, we propose a modified algorithm to the *Median Cost Evaluation* function of k-median algorithm in [4] for 2-dimension. Using the modified *Median Cost Evaluation* algorithm, we achieve asymptotically better time and space bounds for k-median clustering problem.

4.2 Simultaneous Range Query using CM sketch and Dyadic Ranges

Consider a matrix A of size $\Delta \times \Delta$. Assume that Δ is power of 2. Each entry $A(i, j)$ is a non-negative value i.e. strict turnstile model. Let $r_1, r_2, ..., r_m$ be the m non overlapping ranges along y-axis defined by the end points $\{(u_i, v_i), (u_i, v_i')\}$ over the matrix A (see figure 4.1). Also given the width of any r_i is at most L, that is $|v_i - v_i'| \leq L$. Let R_i be the range sum for r_i given as,

$$R_i = \sum_{j=v_i}^{v_i'} A(u_i, j)$$

It is required to support the following operations:
• Update(u,v): Update operation could be either insertion or deletion.
Insert(u,v): perform A(u,v) = A(u,v)+1
Delete(u,v): perform A(u,v) = A(u,v)-1
• Simultaneous Range Query given $r_1, r_2, ..., r_m$: return $R = \sum_{i=1}^{m} R_i$

This can be implemented using $O(n)$ space which is large, where $n = ||A||_1$. So the task is to compute the value of R using low-space, more precisely in polylogarithmic space. Here we give a low-space data structure(we name it as SRQ data structure) that computes the value R approximately

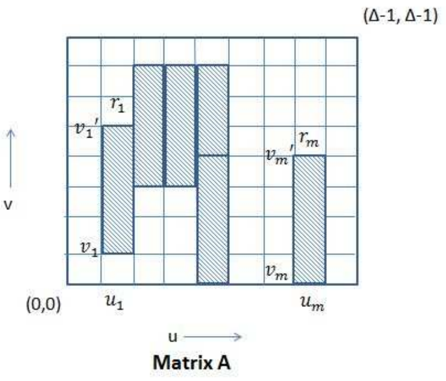

Figure 4.1: *Non-overlapping ranges $r_1, r_2, ...r_m$ defined over Matrix A*

by maintaining the sketch of matrix A. This is accomplished using CM sketch [3] and dyadic range technique. Let $\hat{R}$ be the approximation of R. It is implemented as follows. Consider a vector C defined as follows.

$$C =< A(0,0), A(0,1)...A(0,\Delta-1), A(1,0), A(1,1)...A(1,\Delta-1), A(\Delta-1,0),$$
$$A(\Delta-1,1)...A(\Delta-1,\Delta-1) >$$

Now define a dyadic range of up to $\lceil \log L \rceil + 1$ level over the vector C of the form $[j2^k...(j+1)2^k - 1]$ for level $0 \leq k \leq \lceil \log L \rceil$ and index $0 \leq j \leq \lceil \frac{\Delta^2}{2^k} \rceil - 1$. Let C_k be the vector for $0 \leq k \leq \lceil \log L \rceil$, such that $C_k(j) = \sum_{l=j2^k}^{(j+1)2^k-1} C(l)$ for $0 \leq j \leq \lceil \frac{\Delta^2}{2^k} \rceil - 1$. Each such C_k is then stored using CM sketch. It consists of an array $count[1...\text{w}]$ and d hash function, each hash function is chosen from pair-wise independent hash family, $h_j\{1, ..., |C_k|\} \rightarrow \{1, ..., w\}$ for $1 \leq j \leq d$. Note that $|C_k| \leq \Delta \times \Delta$ for any k. The operations, insertion and deletion is performed at every dyadic level, are implemented as follows.

- **Insert** (u,v): map (u,v) $\rightarrow$ i and perform $count_j[h_j(i)] = count_j[h_j(i)] + 1$
- **Delete** (u,v): map (u,v) $\rightarrow$ i and perform $count_j[h_j(i)] = count_j[h_j(i)] - 1$
- **Simultaneously Range Query** given $r_1, r_2, ..., r_m$: Consider any level l and CM sketch with hash function h_k. By [3], each r_i corresponds to at most 2 point queries in l. Let X denote the set of all buckets of h_k to where $r_1, r_2, ..., r_m$ mapped. Let $C[l] = \underset{k=1,...,d}{min} (\sum_{i \in X} count_k[i])$. Return $\hat{R} = \sum_{l=0,...,\lceil \log L \rceil} C[l]$.

LEMMA 1. $R \leq \hat{R} \leq R + \epsilon(||C||_1 - R)$ with probability at least $1 - \delta$ for $w = \frac{em \log L}{\epsilon}$.

Proof. Consider $\hat{R} = \sum_{i=1}^{m} \hat{R}_i$ where $\hat{R}_i$ is the range sum for r_i.

$$\hat{R} = \sum_{i=1}^{m} R_i + \sum_{i=1}^{m} \Delta_i$$

Let $\Delta = \sum_{i=1}^{m} \Delta_i$

$$\begin{aligned} E[\Delta] &= \sum_{i=1}^{m} E[\Delta_i] \\ &\leq \sum_{i=1}^{m} \frac{2 \log L}{w} \\ &\leq \frac{2m \log L}{w} (||C||_1 - R_i) \\ &\leq \frac{\epsilon}{e} (||C||_1 - R_i) \end{aligned}$$

By Markov inequality, we have

$$Pr[\Delta > \epsilon(||C||_1 - R_i)] \leq \frac{1}{e}$$

For $\hat{R}$ is taken as minimum of d estimates, we have

$$Pr[\Delta > \epsilon(||C||_1 - R_i)] \leq (\frac{1}{e})^d \approx \delta$$

$\square$

$R \leq \hat{R} \leq R + \epsilon(||C||_1 - R)$ with probability at least $1 - \delta$ using $O(\frac{m \log^2 L}{\epsilon} \log(\frac{1}{\delta}))$ space. The estimate time is $O(m \log L \log(\frac{1}{\delta}))$. Update time is $O(\log L \log(\frac{1}{\delta}))$.

4.3 Exclusive Count

This is a tool which uses the SRQ data structure and provides a function *XCount(Q)* to estimate the exclusive count for the following problem.

Consider a matrix of size $\Delta \times \Delta$. Assume Δ is power of 2. Let $Q_i = \{(x, y) | u_i \leq x \leq u'_i$ and $v_i \leq y \leq v'_i\}$ be the i^{th} square region defined by a diagonally end points $\{(u_i, v_i), (u'_i, v'_i)\}$ for $1 \leq i \leq k$. There might be over lapping regions as shown in fig 4.2. Let $Q = \underset{i=1\ldots k}{\cup} Q_i$. Let R be the range sum for Q and $\hat{R}$ be the approximation of R. The exclusive count *XCount(Q)* is given as,

$$XC = \sum_{(i,j) \notin Q} A(i, j)$$

We use SRQ data structure to maintain a sketch of matrix A which provides the functionality to answer the exclusive count XC approximately. Let $\hat{XC}$ be the approximate value of XC given as, $\hat{XC} = ||A||_1 - \hat{R}$ where $\hat{R}$ is range sum returned by SRQ data structure for Q. We claim that the Q can be reduced to at most kL non-overlapping ranges $r_1, r_2 \ldots r_m$ where $m \leq kL$ and $|r_i| \leq L$. It is straight forward to verify that $m \leq kL$. Note that in order to compute kL non-overlapping ranges, we need additional $O(kL)$ space and $O(k^2 L)$ time.

Theorem 4.3.1. *$(1 - \epsilon)XC \leq \hat{XC} \leq XC$ with probability at least $1 - \delta$ for $w = \frac{e2kL \log L}{\epsilon}$.*

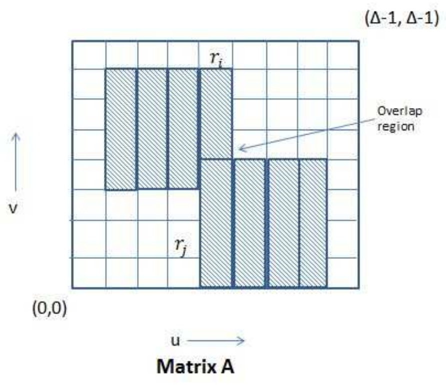

Figure 4.2: *Over-lapped region r_i and r_j with ranges (v_i, v_i') and (v_j, v_j') respectively*

Proof. Consider $\hat{R} = \sum_{i=1}^{m} \hat{R}_i$ where $\hat{R}_i$ is the range sum for r_i.

$$\hat{R} = \sum_{i=1}^{m} R_i + \sum_{i=1}^{m} \Delta_i$$

Let $\Delta = \sum_{i=1}^{m} \Delta_i$

$$\begin{aligned} E[\Delta] &= \sum_{i=1}^{m} E[\Delta_i] \\ &\leq \sum_{i=1}^{m} \frac{2 \log L}{w}(||A||_1 - R_i) \\ &\leq \frac{2kL \log L}{w} XC \\ &\leq \frac{\epsilon}{e} XC \end{aligned}$$

By Markov inequality, we have

$$Pr[\Delta > \epsilon XC] \leq \frac{1}{e}$$

For $\hat{R}$ is taken as minimum of d estimates, we have

$$Pr[\Delta > \epsilon XC] \leq (\frac{1}{e})^d \approx \delta$$

□

Hence, $(1-\epsilon)XC \leq \hat{XC} \leq XC$ with probability at least $1-\delta$ using $O(\frac{kL \log^2 L}{\epsilon} \log(\frac{1}{\delta}))$ space. The time to estimate XC is $O(kL \log L \log(\frac{1}{\delta}))$. Update time is $O(\log L \log(\frac{1}{\delta}))$.

4.4 Median Cost Evaluation

The main tool is *Median Cost Evaluation* data structure. The data structure maintains a set $P \subset \{1, ..., \Delta\}^2$, under addition and deletion of points. This tool provides the function to estimate the median cost C for any set $Q \subset \{1, ..., \Delta\}$ of size k, such that $C = (1 \pm O(\epsilon))$C(Q,P).

The data structure is implemented as follows. Define G_i to be a square grid imposed on space $\{1, ..., \Delta\}^2$, with side length $\frac{\epsilon}{\sqrt{2}}(1+\epsilon)^i$. Clearly G_i can be considered as Matrix A of size $N \times N$ where $N = \lceil \frac{\sqrt{2}\Delta}{\epsilon(1+\epsilon)^i} \rceil$ and let $n_i(c)$ be the number of points from P falls into cell $c \in G_i$. We now show how to estimate median cost for any Q. Let $r_i = (1+\epsilon)$ and $B(q, r_i)$ be the set of points in $\{1, ..., \Delta\}^2$ with distance less than r_i from q. Let $B(Q, r_i) = \cup_{q \in Q} B(q, r_i)$. Let $G_i(B(q, r_i))$ be the set of grid cells that are superimposed by $B(q, r_i)$. Observe that $|G_i(B(q, r_i))| = O(\frac{1}{\epsilon^2})$. To achieve this, divide the area of a ball with radius r_i to the area of one cell. Set $L = O(\frac{1}{\epsilon})$ since the number of cell in a row is $O(\frac{1}{\epsilon})$. Let $G_i(B(Q, r_i)) = \cup_{q \in Q} G_i(B(q, r_i))$. Note that $G_i(B(Q, r_i))$ corresponds to the query region. The sketch of grid G_i is maintained by our SRQ data structure. Let $\hat{R}_i(Q)$ be the estimated exclusive count returned by XCount for $G_i(B(q, r_i))$. The median cost for any Q is given as,

$$\hat{C}(Q, P) = \sum_{-i_0 \le i \le \log \Delta} (r_i - r_{i-1}) \hat{R}_i(Q)$$

Therefore, by [4] $\hat{C}(Q, P) = (1 \pm O(\epsilon)) C(Q, P)$ with probability at least $1-\delta$ using $O(\frac{k \log^2(1/\epsilon)}{\epsilon^2} \log(1/\delta)(\log \Delta + \frac{\log(1/\epsilon)}{\epsilon}))$ space. The estimate time is $O(\frac{k}{\epsilon} \log(1/\epsilon) \log(1/\delta)(\log \Delta + \frac{\log(1/\epsilon}{\epsilon}))$. Update time is $O(\log(1/\epsilon) \log(1/\delta)(\log \Delta + \frac{\log(1/\epsilon}{\epsilon}))$.

Chapter 5

Experimental Results

In this section we present some of the implementation details of the algorithms described in the previous two chapters. For comparison, we focus on d=2 i.e. the set of points P live in discrete plane $\{1, ..., \Delta\}^2$.

All the programs were compiled using g++ version 5.4.2. and all the computations were performed on a DELL PRECISION T1600 machine with intel(R) xeon(R) CPU @ 3.40 GHz and 8 GB main memory, using Linux 2.6.38-8-generic kernel.

We name the algorithm as A1 and A2 described in Chapter 3 and 4 respectively. For comparison, our implemented algorithms only estimate median cost for a given k centres. Subsequently, these algorithm can be used as a subroutine by the various approaches explained in section 3.5, to obtain the best k-centres. Here, we compare the quality of the algorithms A1 and A2 in terms of estimated median cost and the CPU execution time taken. We also compare the quality of the algorithms A1 and A2 with the accurate cost, calculated as the sum of the distance for all points $p \in P$ to the nearest $q \in Q$, given as:

$$\text{Accurate Cost} = \sum_{p \in P} \min_{q \in Q} \parallel p - q \parallel$$

Figure 5.1 and 5.2 summarizes cost of the clustering and the average running time for various k. Since all the algorithms are randomized, we run both the algorithms several time (mostly 5 times) and then taking the average value. We set the parameter for both the algorithms as specified by the author [4]. For comparison, we fix the value of $\Delta = 1000$, $|P| = 500$ and $\epsilon = 0.1$.

In this experiment, we observe that the quality of the median costs were quite promising and lies within the constant factor($\pm 5\%$) from each other. In terms of running time, it turns out that our algorithm A1 outperforms the algorithm A2 and hence provides a good alternative to the algorithm A2. Figure 5.2 summarizes the CPU execution time taken by both the algorithms in milliseconds.

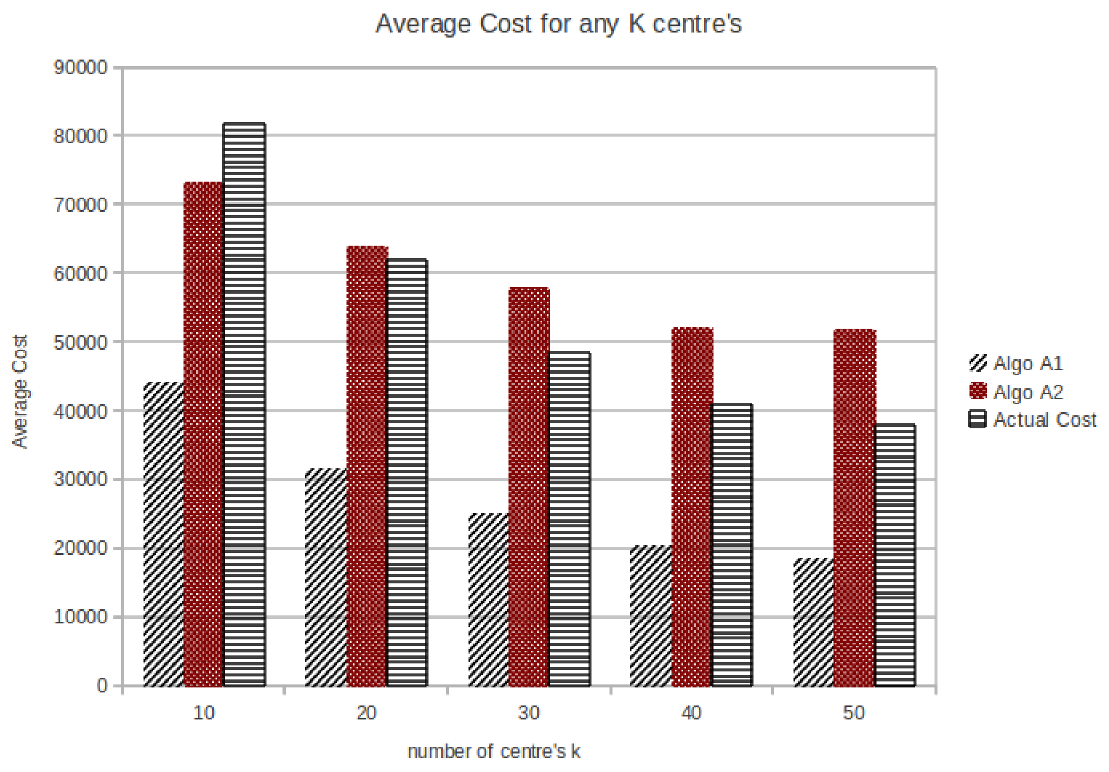

Figure 5.1: *Median Cost results.*

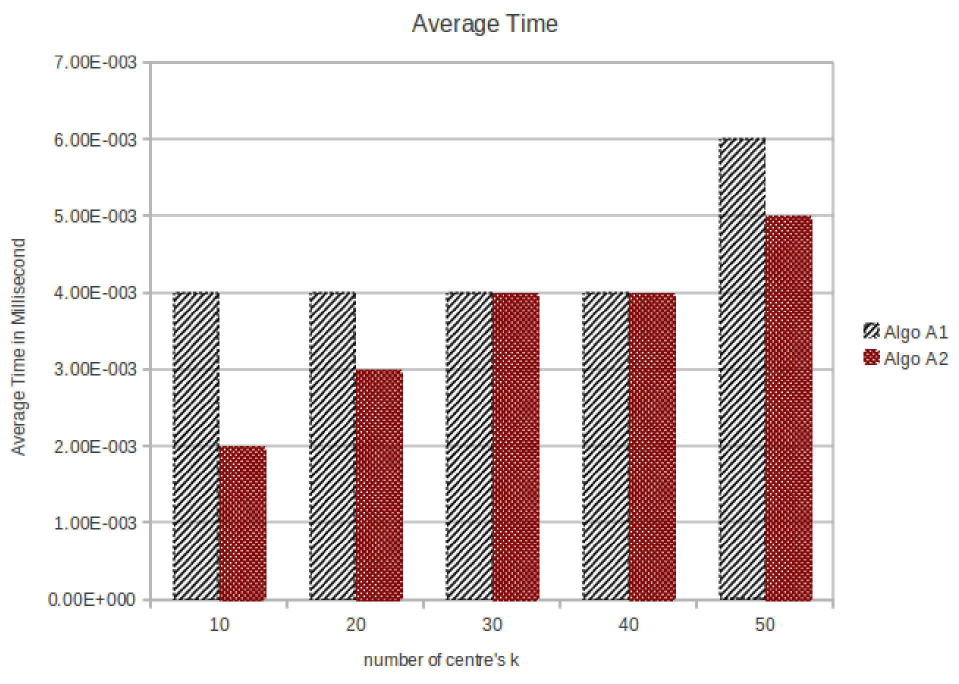

Figure 5.2: *Median Cost results.*

Chapter 6

Conclusion And Future Work

In this thesis we developed new methods to solve $O(1 \pm \epsilon)$-approximation k-median clustering problem for geometric data stream. We have shown results for 2-dimensions only. But this approach can easily be extended for higher dimensions.

Our new technique partitions the space very effectively and reduces it to high dimensional vectors which in turn is stored by CM sketches using dyadic range scheme. Our algorithm takes $O(\frac{k}{\epsilon}\log^4(\frac{1}{\epsilon})(\log\Delta + \frac{\log(1/\epsilon)}{\epsilon}))$ space and $O(k\log^2(\frac{1}{\epsilon})(\log\Delta + \frac{\log(1/\epsilon)}{\epsilon}))$ time to estimate median cost for a given k centers. Our algorithm improves the previous best known space and time bounds in [4].

The main disadvantage of both the algorithms is the dependency of the space and time complexity on Δ, which affects the performance significantly for large Δ.

As we mention that time and space bounds for our algorithm has been provided for 2-dimension. So our future work would be to provide the bounds for higher dimension i.e. for arbitrary d. we also mentioned that time and space bounds depend on Δ. Hence, it has become an interesting area of research to provide a method such that the time and space bounds be independent of Δ.

References

[1] Sumit Ganguly and Graham Cormode. “On Estimating Frequency Moments of Data Streams”. vol. 57, (2007).

[2] Graham Cormode and Marios H.. “Finding frequent items in data streams”. *Proceedings of the VLDB Endowment*. vol. 1, Pages 1530-1541 (2008) .

[3] Graham Cormode and S. Muthukrishnan. “An Improved Data Stream summary: the Count-Min Sketch and its Applications”. *Proceedings of the 6th Latin American Theoretical Informatics (LATIN)*, April 2004.

[4] Piotr Indyk. “Algorithms for Dynamic Geometric Problems over Data Streams”. *ACM*, (2, August 2008.)

[5] Graham Cormode and Marios H. “Finding Frequent Items in Data Streams”. *VLDB*, (2004).

[6] Guha, Meyerson, A. Mishra, N. Motwani, O.C. “Clustering Data Streams: Theroy and Practice”. *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, Pages 515-528, (2008).

[7] N. Alon, Y. Matias, and M. Szegedy. “The Space Complexity of Approximating the Frequency Moments”. *J. Comput. Syst. Sci.*, vol. 58(1), Pages 137147, (1999).

[8] P. Indyk and D. Woodruff. “Optimal Approximations of the Frequency Moments of Data Streams”. *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, (2005).

[9] M. Charikar, K. Chen, and M. Farach-Colton. “Finding Frequent Items in Data Streams”. *Proceedings of the 29th Annual International Colloquium on Automata, Languages and Programming (ICALP)*, Pages 693703, (2002).

[10] A. K. Jain, M. N. Murty, and P. J. Flynn. “Data clustering: a review”. *ACM Computing Surveys (CSUR)*, (1999).

[11] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala and V. Pandit. “Local search heuristic for k-median and facility location problems”. *Proceeding STOC '01 Proceedings of the thirty-third annual ACM symposium on Theory of computing*, (2001).

[12] M. R. Ackermann, C. Lammersen, M. Martens, C Raupach, C. Sohler and K. Swierkot. “StreamKM++: A Clustering Algorithm for Data Stream”. *SIAM*, (2010).

[13] YI-HONG LU and YAN HUANG. “Mining Data Streams Using Clustering”. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, (2005).