Project Report: Streaming Algorithms for Clustering

Jatin Garg

April 26, 2015

An ordered sequence $x_1, x_2, ..., x_n$ is termed as a data stream if it can be accessed in that order only. These datsets are generally very large that can not be fit in main memory and are accessed from secondary storage devices. So, it can be accessed only small number of times due to larger time required to access them.

Since Main memory is very low compared to data, the algorithm trying to cluster them should somehow store a summary of previous data and should have enough space left to process remaining data. Also, to judge the performance of any stream algorithm, we must include number of linear scans required by it.

Now all the algorithms discussed below are in context of K-median where objective function is to minimize the sum of distance from nearest center for each point as opposed to square of distance in K-means.

First I will discuss a clustering framework for streaming algorithms where already existing algorithms for static data with provable guarantees can be used to obtain provable guarantees in streaming setting. Then I will discuss Indyk's algorithm(1999) which is a random-sampling based algorithm.

1 A Provable Stream Clustering Framework

Firstly I discuss a simple divide and conquer algorithm for K-median without dwelving into streaming setting. Then a modified version of this algorithm which works on streams is discussed.

1.1 Small-Space Algorithm

1. Divide Dataset S into l disjoint pieces $X_1, ..., X_l$
2. for each i find O(k) centers in X_i. Assign each point in X_i to its cosest center.
3. Let X' be the O(lk) centers obtained in (2) where each center c is weighted by number of points assigned to it.
4. Cluster X' to find k centers.

Following are the set of lemmas that help us to give approximation guarantee for above algorithm:

Lemma 1 *Given an instance (S,K) of K-median cost(S,S) $\leq$ cost(S,Q) for any Q.*

Lemma 2 $\Sigma cost(X_i, X_i) \leq 2cost(S, S)$.

Lemma 3 *Let $C = \Sigma cost(X_i, X_i)$ and C* = cost(S,S), then cost(X',X') $\leq$ 2(C+C*)*

Now assume that we run (a,b)- approximation algorithm on all X_i in second step and in fourth step we run a c-approximation algorithm to obtain final set of k centers. Then,

Lemma 4 *The approximation factor of small-space algorithm is 2c(1+2b)+2b .*

Note that proofs of above 4 lemmas have already been discussed in class.

Now authors suggest that instead of doing reclustering by dividing the data only once, we can do it for some constant number of times. They call this recusrsive algorihm smaller-space.

Lemma 5 *For constant i, Algorithm Smaller-space(S,i) gives a constant factor approximation to k-median problem.*

The proof of above lemma can be shown by writng recurrence of cost at each step and solving it. Final cost comes out to be $c(2(2b + 1))^i$ which is constant given that i is constant.

For above algorithms, we know that lk $\leq$ M and n/l $\leq$ M due to memory constraints. It is not possible always for some l to exists. So, we can not say that above framework would always be valid. So, we will make an assumption that M is not too small, specifically, it is of some O(n^ϵ) and then a modified version of above algorithm gives guaranteed approximation for the problem and is valid within the specified assumption.

1.2 Data Stream Algorithm

Let m = $\sqrt{M}$

1. Input first m points,use bicriterion algorithm to reduce it to O(k) centers(say 2k). Running bicriterion algorithm requires O(f(m)) space which is O(m^2) for a primal-dual algorithm.
2. Repeat step-1 for m/2k iterations. Now we have m first level medians. So, just cluser these m points first and then continue. In general, whenever m medians for level-i are available cluster them into 2k level-i+1 ceners.
3. At the end use c-approximation algorithm to give final set of k centers

In above algorithm, the maimum number of levels required are O(log(n/m)/log(m/k)). We can safely assume that k$\ll$m. Let m=n^ϵ . Then number of levels are O(1/ϵ). So, maximum memory required to store all the points at all levels are m*number of levels = log(n) which according to our assumption can be accommodated in memory. Hence this algorithm is valid in all settings.

Now there exists a local search based linear space bicriterion algorithm which runs in quadratic time[2]. So, author suggests to use it. Then we have the following theorem.

Theorem 6 *We can solve the kMedian problem on a data stream with time O($n^{1+\epsilon}$) and space O(n^ϵ) up to a factor $2^{O(1/\epsilon)}$.*

The proof of above theorem is straightforward using Lemma 5. Note that in theorem M=n^ϵ.

1.3 Framework in retrospect

The framework that we have discussed is composed of sequence of algorithms. We just have to choose an algorithm for compressing the points into O(k) centers and then finally use a contant factor approximation. Thus, we have proved theoretically that any algorithm for k Median problem that work well on static chunk of data can be used in this recursive manner to work on data stream still giving good bounds for the solution. Also, note that though we have discussed the framework with respect to algorithms with proven guarantees, we can also use it for any algorithm that practically works well on staic data.

2 Indyk's Algorithm

This is one of oldest algorithm which works on data streams. The main drawback of this algorithm is that it is a 2 pass algorithm, i.e., it requires 2 linear scans of the data which leads to high running time.

The main intution behind the algorithm is that a large enough random sample from whole of data is a good representation of all data and solving K-median problem on this sample should work well for whole of data unless some cluster is very far from rest of clusters and has very few points in the data. Hence we do second clustering to cover such outliers.

Algorithm (Data S, confidence δ) :

1. Let S' $\subset$ S be a random subset of size s
2. run an (a, b) approximation algo on S' to get O(ak) medians T'
3. Let S" $\subset$ S be the 8kn log(k/δ) points farthest from T'
4. run an (a, b) approximation algo on S" to get O(ak) medians T"
5. Output T'$\cup$ T"

Let s=$\sqrt{8knlog(k/\delta)}$. Then we require O(8kn log(k/δ)) memory which is reasonable to expect.

Theorem 7 *With a probability $\geq 1-2\delta$, above alorithm is (2a, (1+2b) (1+2/δ)) approximation.*

We will prove above theorem through a seies of lemma. Let T* = $\{t_1*, ..., t_k*\}$ be the optimal medians and let $S_1, ..., S_k$ be the corresponding optimal clusters. Let L be the large clusters such that :
L = {i:$|S_i| \geq (8n/s)log(k/\delta)$ }.
The sample S' contains point $S'_i = S_i \cap S' from cluster S_i$.

Lemma 8 *with probability* $\geq 1-\delta$*, for all* $i \in L : |S'_i| \geq (s/2n) * |S_i|$.

Proof: Using Chernoff bounds we can say for all $i \in L$: that Pr$\{|S'_i| \leq (s/2n) * |S_i|\} \leq \delta/k$. Now use multiplicative chernoff bound to get the proof.

Lemma 9 *with probability* $\geq 1-\delta$*, cost(S',T*)* $\leq (1/\delta)$** (s/n) cost(S,T*).*

Proof: The expected value of cost(S', T*) is exactly (s/n)cost(S, T*). The lemma follows by Markovs inequality.

Now Assume that high probability events of Lemma 8 and 9 hold.

Lemma 10 *cost(S', T') $\leq$ 2b* (1/δ) * (s/n) * cost(S, T*).*

Proof: Using Lemma 1 and b approximation, cost(S', T') $\leq$ 2b* cost(S', T*). Now use Lemma 9.

Lemma 11 *cost($\cup_{i\in L}$ S_i,T') $\leq$ cost(S,T*) * (1+2*(1+2b)*(1/δ))* .

Proof. Let d(x,y) be the distance between x and y. and if y is a set then it is distance of x from closest point in that set. Using traingle inequality, we can write that d(t_i*,T') $\leq$ $d(t_i*, y)$ + d(y,T') for every point that belongs to S_i'. Averaging it over all points of S_i'

$$d(t_i*, T') \leq (1/|S_i'|)\Sigma d(t_i*, y) + d(y, T')$$

Now bounding the cost of large clusters wih respect to T'

$$cost(\cup_{i\in L} S_i, T') \leq \sum_{i\in L}\sum_{x\in S_i}(d(x, t_i*) + d(t_i*, T'))$$

$$cost(\cup_{i\in L} S_i, T') \leq cost(S, T*) + \sum_{i\in L}|S_i|d(t_i*, T')$$

Now using above averaged traingle inequality and using Lemma 8 we get,

$$cost(\cup_{i\in L} S_i, T') \leq cost(S, T*) + (2n/s)\sum_{y\in S'} d(t_i*, y) + d(y, T')$$

$$cost(\cup_{i\in L} S_i, T') \leq cost(S, T*) + (2n/s)(cost(S', T*) + cost(S', T')))$$

Now use Lemma 9 and 10 to get the result. ∎

From the algorithm it is clear that T" takes care of all the points in S". So, we want to find a bound over S\S" with respect to T'.

Lemma 12 *cost(S\S", T') $\leq$ cost(S,T*)* (1+2*(1+2b)*(1/δ))* .

Proof: As there can be atmost k small clusters, maximum number of points in small clusters can be (8kn/s) log(k/δ). This implies that number of points in large clusters is greater than or equla to n - (8kn/s) log(k/δ).

Now S\S" contains exactly n - (8kn/s) log(k/δ) points that are closest to T'. Hence we can say that cost(S\S", T') $\leq$ cost($\cup_{i\in L}$ S_i,T'). Thus proof follows from Lemma 11.

Lemma 13 *cost(S", T") 2b * cost(S, T*).*

The proof of above lemma is straightforward.

Now summing Lemma 12 and Lemma 13 proves our theorem.

Though above algorithm is not as good as when run on a data stream but on a static chunk of data, this algorithm is very useful and is often used within other data stream algorithms as it runs very fast.

3 References

[1] S. Guha, A. Meyerson, Nina Mishra, Rajeev Motwani, and Liadan OCallaghan. Clustering Data Streams: Theory and Practice.
[2] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In Proc. FOCS, pages 378388, 1999.