**CSCE 689: Special Topics in Modern Algorithms for Data Science** Fall 2023

Lecture 26 — November 1, 2023

*Prof. Samson Zhou* *Scribe: Dawei Xiang*

## 1 Recap of Streaming model

Given elements of an underlying data set $S$, which arrives sequentially, we are interested in the evaluation or approximation of a given function of the whole dataset. The goal is to only use space sublinear in the size $m$ of the input S to get the approximation.

## 2 Linear Sketch

Linear Sketch is an algorithm framework: Suppose stream $S$ induces a frequency vector $f$, we generate a random matrix $A$ and maintain $A \cdot f$. And we apply a post-processing function $g(A \cdot f)$ as the output for the estimation.

We can verify that

1. AMS, CountSketch, CountMin, sparse recovery, distinct elements, coreset construction for clustering fit into this framework.
2. Greedy algorithm for maximal matching, connectivity, bipartiteness, MisraGries **DO NOT** fit into this framework.

And

1. AMS, CountSketch, CountMin, sparse recovery, distinct elements, coreset construction for clustering work for insertion-deletion streams
2. Greedy algorithm for maximal matching, connectivity, bipartiteness, MisraGries **DO NOT** work.

Li et.al. proposed a theorem for linear sketch in [1] .

**Theorem 1.** *For sufficiently long data streams with arbitrarily large coordinates at intermediate stages of the stream, any one-pass insertion-deletion streaming algorithm can be implemented with a linear sketch*

## 3 Sliding Window model

In a similar setting as Linear Sketch, Sliding Window model cares about the most recent updates instead of the whole stream: given elements of an underlying data set S, which arrives sequentially, we want to output an evaluation or approximation of a given function on the most recent $m$ updates using space sublinear in the size $m$ of the input S.

Because the sliding window model emphasizes recent interactions, it is more appropriate for time-sensitive settings.

## 3.1 Problem of directly applying Linear Sketch on Sliding Window model

Because in the sliding window model we just know the newly added element, we can just see what is within the window but we don't know the deleted element on the left side. so we cannot directly apply the linear sketch method.

**Example 1.** For a given stream as $\{1,1,1,1,1,1,1,1,1,1,1,2,3,4,5,6,7,8,9,10,11\}$, let $u$ represents the window of first 11 elements and $v$ represents the window of last 11 elements.

We have $f(u) = [11,0,0,...]$ and $f(v) = [1,1,1,...]$. Thus $F_2(u) = n^2$ and $F_2(v) = n$. And if we apply the matrix $A$ to the whole stream, $A(u+v)$ should be $F_2(u+v) = n^2$. But in the sliding window model, when $u$ is expired, we only have $A(v) = n$ which causes a discrepancy.

## 3.2 Sliding Window and Sampling

Before, in sampling, we can sample each point with probability proportional to the "importance" of each point. Now in the Sliding Window model, we can also consider the "importance" of each point with respect to the following points in the stream.

This is inspired by the intuition that previous points do not matter in the importance of a point because the previous points can expire. Correspondingly we have another theorem:

**Theorem 2.** *There exists a sliding window model algorithm that samples roughly* $\frac{k^2 d}{\epsilon^2} polylog \frac{n\delta}{\epsilon}$ *points and outputs a coreset for* $(k,z)$*-clustering. [3]*

# 4 Sliding Window Algorithms

Suppose we are trying to approximate some given function and we already have a streaming algorithm for this function and this function is also "smooth".

Here "smooth" is defined as: If $f(B)$ is a good approximation to $f(A)$, then $f(B \cup C)$ will always be a good approximation to $f(A \cup C)$, which means that if B is a good approximation of A, then it will always be a good approximation no matter what comes after.

The smooth histogram framework [2] gives a sliding window algorithm for smooth functions, given a streaming algorithm for the function.

## 4.1 Smooth Histogram

In the smooth histogram framework [2], each time a new element arrives, we start a new instance of the streaming algorithm (along with existing instances). Each time there are three instances that report close values, we delete the middle one. And we use different checkpoints to "sandwich" the sliding window.

**Example 2.** For a given stream as $\{1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1\}$, we want to calculate the number of ones in the sliding window. Then we need to start a counter each time we meet a one. If three counters are close, then we delete the one in the middle. At last, we use the counters closest to the sliding window as the estimation.

With this framework, we can convert a streaming algorithm for a smooth function into a sliding window algorithm

## References

[1] Yi Li, Huy L Nguyen, and David P Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 174–183, 2014.

[2] Braverman, Vladimir and Ostrovsky, Rafail. Smooth histograms for sliding windows. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 283–293, 2007.

[3] Woodruff, David P. and Zhong, Peilin, and Zhou, Samson. Near-Optimal $k$-Clustering in the Sliding Window Model In *NeurIPS*, 2023.