Adversarially Robust Streaming Algorithms

Fan Jue (A0221578B)[†], Tian Xiao (A0220592L)[†]

Abstract

As the world is getting data-intensive while memory space remains limited, streaming algorithms become increasingly important since they process data in a stream-like, space-efficient manner. More than often, people not only require the final output of a stream, but also wish to track its intermediate outputs to understand the real-time trends (e.g., in stock markets) and make decisions. In such scenarios, future streaming data can be affected by past decisions. This drives the need for streaming algorithms to be adversarially robust, which guarantees a good approximation with high probability even if the stream is manipulated by an adversary. In this write-up, we give an introduction to adversarially robust streaming algorithms and comprehensively explain two such algorithms, SKETCHSWITCHING and ROBUSTSKETCH, and analyze their theoretical guarantees. We also discuss some potential future directions in this field.

1 Introduction

In this era of information, a huge number of data are transmitted, received, and analyzed frequently. Stock prices change and traders make decisions within seconds; banks handle massive numbers of transactions while identifying fraudulent acts. A naïve solution is to store all past data for future analysis and decision making. However, the complexity of real-world problems usually results in data that are exceptionally costly to store. Therefore, streaming algorithms emerge to provide an approximated answer based on a summary or "sketch" of the data stream using limited space. Streaming algorithms are prevalently applied in the real world because data are always received and processed in a streaming manner. For example, website managers can use the *heavy hitter* algorithm [1] to identify IP addresses that frequently access their websites as potentially malicious.

Classic streaming algorithms target streams with "fixed" data (i.e., unknown fixed randomness). Particularly, the order of tokens in the input streams is independent of the algorithm's output. However, the future depends on the past in many real-life scenarios. For example, traders' decisions affect future stock prices and Instagram users' viewing histories affect their future feeds. Under these circumstances, the desirable guarantee given by streaming algorithms may not hold any more. This gives rise to adversarially robust streaming algorithms that strive to provide approximation with high probability guarantee even when the future inputs depend on previous outputs.

In this paper, we investigate how to construct adversarially robust streaming algorithms. We first introduce some essential preliminary backgrounds in Sec. 2. Next, we introduce in Sec. 3 the SKETCHSWITCHING algorithm [2], one of the most fundamental adversarially robust streaming algorithms. In Sec. 4, we elucidate the ROBUSTSKETCH algorithm [3] which improves upon SKETCHSWITCHING and uses *differential privacy* [4] to achieve more efficient bounds. We also provide examples on how the two algorithms work on

[†]Equal contribution.

F_2-approximation, a classical streaming problem. Then we end with a discussion on the two algorithms and some future directions in Sec. 5.

2 Preliminaries

2.1 Streaming Algorithms

Let $[n]$ denote the set $\{1, 2, \cdots, n\}$. For each $i \in [n]$, let f_i denote the *frequency* (i.e., number of occurrences) of element i. A general *turnstile* stream of length T can be represented by a sequence of tokens (i_t, c_t) where $i_t \in [n]$ and $c_t \in \mathbb{Z}$ such that each token updates the frequency of element i_t, f_{i_t}, to $f_{i_t} + c_t$. Such a stream is *insertion-only* if $c_t \geq 0$ for all $t \in [T]$. Let $g : \mathbb{R}^n \to \mathbb{R}$ be the function that maps the frequency vector $\mathbf{f} = \begin{bmatrix} f_1 & f_2 & \cdots & f_n \end{bmatrix}$ to our ideal output. We call g our *objective function*. For example, if we are interested in the *second frequency moment* (F_2) of the stream, then $g(\mathbf{f}) = \sum_{i \in [n]} |f_i|^2$. Since it is space expensive to store $\mathbf{f}$ in full, we may use streaming algorithms to obtain an approximation to $g(\mathbf{f})$ instead. Commonly, we aim to devise an (ϵ, δ)-*approximation* algorithm such that its output is no more than $\epsilon \cdot g(\mathbf{f})$ away from the true quantity $g(\mathbf{f})$ with probability at least $1 - \delta$ for some $\epsilon > 0$ and $\delta \in (0, 1)$.

Common streaming tasks include counting distinct elements, approximating frequency moments, spotting heavy hitters, etc. We refer the readers to [1] for a detailed explanation of corresponding streaming algorithms to these tasks.

2.2 Tracking Algorithms

Conventional streaming algorithms satisfy approximation guarantees at one single query after all updates are processed. In contrast, *tracking algorithms* are another group of streaming algorithms that aim to output accurate (temporary) approximations after every update. The performance of a tracking algorithm is ensured through the notion of *strong tracking guarantee* [5], which means that at every timestamp t (where the frequency vector is $\mathbf{f}^{(t)}$), the algorithm outputs an approximation that is close to the true (temporary) quantity given by the objective function g, $g\left(\mathbf{f}^{(t)}\right)$. Formally,

Definition 1 (Strong Tracking Guarantee)**.** For timestamp $t \in [T]$, let the frequency vector after the t-th update be $\mathbf{f}^{(t)}$. A randomized algorithm $\mathcal{A}$ is said to provide (ϵ, δ)-*strong g-tracking* if, at every timestamp $t \in [T]$, $\mathcal{A}$ outputs an estimate $R^{(t)}$ such that

$$\left|R^{(t)} - g\left(\mathbf{f}^{(t)}\right)\right| \leq \epsilon \cdot \left|g\left(\mathbf{f}^{(t)}\right)\right|,$$

with probability at least $1 - \delta$, for some $\epsilon > 0$ and $\delta \in (0, 1)$.

Example. Indyk's 2*-stable sketch* [6] is a strong tracking algorithm for F_2-approximation[1] for insertion-only streams, which aims to estimate $g\left(\mathbf{f}^{(t)}\right) = \sum_{i \in [n]} \left|f_i^{(t)}\right|^2$ for each $t \in [T]$. It uses a matrix $\Pi \in \mathbb{R}^{d \times n}$ whose entries are sampled from a 2-stable distribution [7] (i.e., for any independent random variables X, Y, Z following the distribution and any $a, b \in \mathbb{R}$, $aX + bY$ and $(|a|^p + |b|^p)^{1/p} Z$ follow identical distributions) such as Gaussian distribution. As the stream tokens arrive, we maintain a vector $\Pi\mathbf{f}^{(t)}$ such that $\Pi\mathbf{f}^{(t)} = \Pi\mathbf{f}^{(t-1)} + c_t \Pi\mathbf{e}_{i_t}$,[2] where $\mathbf{e}_{i_t} \in \mathbb{R}^n$ denotes the vector whose i_t-th element is 1 and other elements are 0. At the end of each timestamp t, the algorithm outputs the median of $\Pi\mathbf{f}^{(t)}$ as the F_2-approximation. [8] shows that Indyk's 2-stable sketch achieves (ϵ, δ)-strong g-tracking guarantee using $\tilde{\mathcal{O}}\left(\epsilon^{-2} \log T \log(1/\delta)\right)$

[1] In general, Indyk's p-stable sketch can be used for F_p-approximation.

[2] Thus, it is also a linear sketch.

space. Later on, we will give examples on how to convert this sketch into adversarially robust streaming algorithms.

2.2.1 Adversarial Robustness of Tracking Algorithms

At each update $t \in [T]$, the tracking algorithm publishes its temporary approximation of $g\left(\mathbf{f}^{(t)}\right)$ to the adversary. Upon observing the latest output, the adversary adaptively chooses the next token to be passed to the algorithm. The goal of the adversary is to eventually trigger an **incorrect** output from the streaming algorithm. The interactions between a (randomized) streaming algorithm $\mathcal{A}$ and an adversary `Adversary` are modeled as a two-player game. The two players take turns to make decisions upon observing the other player's previous behavior. Formally,

1. `Adversary` chooses a token (i_t, c_t), which may depend on the previous tokens and outputs of $\mathcal{A}$, and passes the token to $\mathcal{A}$.
2. $\mathcal{A}$ processes (i_t, c_t) and outputs $R^{(t)}$ as the current approximation.
3. `Adversary` observes $R^{(t)}$ and proceeds to the next round.

A tracking algorithm is said to be *adversarially robust* if it still satisfies strong tracking guarantee (Def. 1) even if the stream is updated by an adaptive adversary.

Remark. In general, linear sketches such as Indyk's 2-stable sketch are not adversarially robust [9].

2.3 Differential Privacy

In this section, we will introduce a separate branch of studies, *differential privacy* (DP) [4], and explain why it can be associated with adversarially robustness of streaming algorithms. Consider an (randomized) algorithm $\mathcal{A} : \mathcal{D} \to \Theta$ that takes in a dataset $D \in \mathcal{D}$ and returns an output $\theta \in \Theta$. DP is actually a measurement of how private $\mathcal{A}$ is (i.e., how likely an attacker can reconstruct the input dataset D). It works by comparing the outputs when $\mathcal{A}$ takes in a pair of *neighboring datasets* D and D' that differ by only one element. Intuitively, if the two outputs are close to each other, the attacker is unlikely to tell whether the element by which the two datasets differ is used in the input; if the outputs for all such neighboring datasets are close, the attacker is unlikely to tell whether any element is used in the input and thus the algorithm is private. Formally,

Definition 2 (Differential Privacy [4])**.** An algorithm $\mathcal{A} : \mathcal{D} \to \Theta$ is said to be preserve (α, β)-*differential privacy* ((α, β)-DP) if for any pair of neighboring datasets $D, D' \in \mathcal{D}$ and any subset of outputs $O \subseteq \Theta$,

$$\Pr[\mathcal{A}(D) \in O] \leq e^{\alpha} \cdot \Pr[\mathcal{A}(D') \in O] + \beta.$$

Here $\alpha > 0$ represents the privacy level of the algorithm, where a smaller α indicates higher privacy; $\beta \in [0, 1)$ is an absolute tolerance term which represents a very small probability of privacy breach. In particular, $(\alpha, 0)$-DP is abbreviated as α-DP.

To achieve DP, the most common technique is *noise addition*, where we add some certain type of noise that is properly scaled to the algorithm output, such that the attacker is unlikely to tell whether a particular output he sees is due to the algorithm output or the added noise. To determine the appropriate scale of the added noise, we define the *sensitivity* of algorithm $\mathcal{A}$ as follows:

Definition 3 (Sensitivity)**.** The *sensitivity* of an algorithm $\mathcal{A} : \mathcal{D} \to \Theta$ is defined as the maximum change of its output when it takes in a pair of neighboring datasets, that is,

$$\Delta(\mathcal{A}) \coloneqq \max_{\text{neighboring} D, D' \in \mathcal{D}} |\mathcal{A}(D) - \mathcal{A}(D')| .$$

For any algorithm $\mathcal{A}$ (either deterministic or randomized), we can then achieve ϵ-DP by adding Laplacian noise proportional to $\mathcal{A}$'s sensitivity:

Theorem 1. For any algorithm $\mathcal{A} : \mathcal{D} \to \Theta$, define a new randomized algorithm $\widetilde{\mathcal{A}}$ such that for any $D \in \mathcal{D}$,

$$\widetilde{\mathcal{A}}(D) \coloneqq \mathcal{A}(D) + \mathrm{Lap}\left(\frac{\Delta(\mathcal{A})}{\alpha}\right),$$

where $\mathrm{Lap}(\cdot)$ denotes a randomly sampled Laplacian noise with mean 0 and scale $\cdot$. The new algorithm $\widetilde{\mathcal{A}}$ satisfies α-DP.

Differentially private algorithms can be adaptively composed together, that is, a latter DP algorithm $\widetilde{\mathcal{A}}_2$ may use not only dataset D but also the (private) output of an earlier algorithm $\widetilde{\mathcal{A}}_1(D)$. The resulting algorithm is still differentially private:

Theorem 2. When k algorithms satisfying α-DP are adaptively composed together, the resulting algorithm satisfies (α', β)-DP, where $\alpha' = \sqrt{2k \log(1/\beta)} \cdot \alpha + 2k\alpha^2$.

DP implies adversarial robustness. For tracking algorithms, the adversary can adaptively choose the next token based on its inference of the internal state of the algorithm. Conversely, if the internal state is protected with DP, the adversary will be uncertain about the exact internal state and thus which token to choose. This would make the algorithm adversarially robust.

3 Adversarial Robustness via Sketch Switching

We focus on converting non-robust strong tracking algorithms to adversarially robust algorithms. Specifically, we use $L(\epsilon, \delta)$ to denote the space required by an (ϵ, δ)-strong g-tracking algorithm or sketch, which we use as our base algorithm.

In the adversarial setting described in Sec. 2.2.1, the `Adversary` observes intermediate outputs and strategically decides the next token in the input stream. To avoid being targeted by the `Adversary`, an intuitive solution is to change the streaming algorithm every time after an output is made. However, this solution requires $\mathcal{O}(T \cdot L(\epsilon, \delta))$ space, which can be very expensive when the stream length T is large. Alternatively, at timestamp t, if the most recent output $\hat{R}_\rho$ is within an acceptable distance from the estimate $R_\rho^{(t)}$ given by the current algorithm $\mathcal{A}_\rho$, then we continue to output $\hat{R}_\rho$. Otherwise, we output $R_\rho^{(t)}$ and switch to the next algorithm $\mathcal{A}_{\rho+1}$. This is the intuition behind the SKETCHSWITCHING algorithm shown in Alg. 1.

The total space complexity of this algorithm is $\mathcal{O}(\lambda \cdot L(\epsilon/10, \delta/\lambda))$, which is linearly dependent on the parameter λ. Intuitively, we would want λ to be as small as possible. The *flip number* defined below is used to provide a tight lower bound for λ which ensures a sufficient number of sketches.

Definition 4 (Flip Number)**.** Let $\epsilon \geq 0$ and $\bar{y} = (y_1, y_2, \ldots, y_T)$ be a sequence of real numbers. The *ϵ-flip number* $\phi_\epsilon(\bar{y})$ of $\bar{y}$ is the maximum number $k \in \mathbb{N}$ where there exists $i_0 = 0 < i_1 < \cdots < i_k \leq T$ such that $|y_{i_j} - y_{i_{j+1}}| > \epsilon \cdot y_{i_{j+1}}$ for all $j \in \{0, 1, 2, \ldots, k-1\}$.

Algorithm 1 The SKETCHSWITCHING algorithm.

Initialize λ $(\epsilon/10, \delta/\lambda)$-strong g-tracking sketches $\mathcal{A}_1, \cdots, \mathcal{A}_\lambda$ with independent randomness
$\rho \leftarrow 1$; $\hat{R}_1 \leftarrow \mathcal{A}_1(\mathbf{0})$
for token (i_t, c_t) in stream **do**
    Feed (i_t, c_t) to all sketches $\mathcal{A}_1, \ldots, \mathcal{A}_\lambda$
    $R_\rho^{(t)} \leftarrow$ approximation given by $\mathcal{A}_\rho$
    if $\hat{R}_\rho \in (1 \pm \epsilon/2) R_\rho^{(t)}$ **then**
        output $\hat{R}_\rho$
    else
        output $R_\rho^{(t)}$
        $\hat{R}_{\rho+1} \leftarrow R_\rho^{(t)}$
        $\rho \leftarrow \rho + 1$
    end if
end for

Recall that $g : \mathbb{R}^n \to \mathbb{R}$ is the objective function that maps each frequency vector $\mathbf{f}^{(t)} = \begin{bmatrix} f_1^{(t)} & f_2^{(t)} & \cdots & f_n^{(t)} \end{bmatrix}$ to the ideal output $g\left(\mathbf{f}^{(t)}\right)$. Let $\bar{y} = (y_1, y_2, \ldots, y_T)$ be a sequence where $y_t = g\left(\mathbf{f}^{(t)}\right)$ for each $t \in [T]$. We denote the ϵ-flip number of such a sequence $\bar{y}$ as $\phi_{\epsilon,g}$ from here onward.

Theorem 3. The SKETCHSWITCHING algorithm satisfies (ϵ, δ)-strong g-tracking guarantee even if the stream is chosen by an `Adversary`, if we set $\lambda = \phi_{\epsilon/10,g}$.

Proof. From Def. 4, it can be easily seen that setting $\lambda = \phi_{\epsilon/10,g}$ will allow the entire stream to be processed before the algorithm halts. According to Alg. 1, all sketches are $(\epsilon/10, \delta/\lambda)$-strong g sketches. Thus, at any timestamp t, denoting the value of ρ as $\rho(t)$, $A_{\rho(t)}$ computes some approximation $R_{\rho(t)}^{(t)}$ such that $\left|R_{\rho(t)}^{(t)} - g\left(\mathbf{f}^{(t)}\right)\right| \leq (\epsilon/10) \cdot g\left(\mathbf{f}^{(t)}\right)$ with probability at least $1 - \delta/\lambda$. The actual output at timestamp t, $\hat{R}_{\rho(t)}$, satisfies $\left|\hat{R}_{\rho(t)} - R_{\rho(t)}^{(t)}\right| \leq (\epsilon/2) \cdot R_{\rho(t)}^{(t)}$ no matter if $\hat{R}_{\rho(t)}$ is updated or not. Note that $(1+\epsilon/2)(1+\epsilon/10) < 1 + \epsilon$ and $(1 - \epsilon/2)(1 - \epsilon/10) > 1 - \epsilon$ for all $\epsilon \in (0, 1)$. Hence $\left|\hat{R}_{\rho(t)} - g\left(\mathbf{f}^{(t)}\right)\right| \leq \epsilon \cdot g\left(\mathbf{f}^{(t)}\right)$ with probability at least $1 - \delta/\lambda$. Since there are λ sketches in total, we can apply union bound to show that $\left|\hat{R}_{\rho(t)} - g\left(\mathbf{f}^{(t)}\right)\right| \leq \epsilon \cdot g\left(\mathbf{f}^{(t)}\right)$ for all timestamp t with probability at least $1 - \delta$. Therefore, Alg. 1 yields an approximation with (ϵ, δ)-strong g-tracking guarantee. □

3.1 Example: F_2-Approximation for Insertion-Only Streams

Now we will give an example of applying the SKETCHSWITCHING algorithm to F_2-approximation for insertion-only streams. We use Indyk's 2-stable sketch as our non-robust $(\epsilon/10, \delta/\lambda)$-strong g-tracking sketch, which means $L(\epsilon/10, \delta) = \tilde{\mathcal{O}}\left(\epsilon^{-2} \log T \log(1/\delta)\right)$. Then the SKETCHSWITCHING algorithm can, with probability at least $1 - \delta$, output at each timestamp t an estimate $\hat{R}_{\rho(t)}$ such that $|\hat{R}_{\rho(t)} - F_2(\mathbf{f}^{(t)})| \leq \epsilon \cdot F_2(\mathbf{f}^{(t)})$. We use the following theorem (proven in App. B.1) to further bound the flip number $\phi_{\epsilon/10,g}$:

Theorem 4. In the insertion-only streaming model, the flip number ϕ_{ϵ,F_2} is $\mathcal{O}(\epsilon^{-1} \log T)$.

Therefore, choosing $\delta = 1/T$, the total space is $\tilde{\mathcal{O}}\left(\epsilon^{-3} \log^2 T\right)$, which is sublinear in T.

4 Adversarial Robustness via Differential Privacy

Although the SKETCHSWITCHING algorithm is adversarially robust, it does not seem to be space efficient because at any timestamp $t \in [T]$, only one of the λ sketches is in use while the rest are wasted. On the other hand, if we remove the requirement for adversarial robustness, the λ sketches could be used to compute a median that better approximates g (i.e., the *median trick* [1]). Therefore, if we use DP to protect the median from the `Adversary`, then we can achieve adversarial robustness while making use of all the sketches together. Specifically, to achieve α-DP, we may add Laplacian noise $\mathrm{Lap}(1/\alpha)$ to the **rank** of each individual approximation according to Thm. 1. Denote this method of computing the private median as PRIVATEMEDIAN and we have the following theorem (proven in App. B.2):

Theorem 5. Suppose we have λ real numbers $R_1, R_2, \cdots, R_\lambda$ and we use the PRIVATEMEDIAN algorithm satisfying α-DP to obtain a private median $\hat{R}$. Then with probability $1-\delta$, there are at least $\lambda/2-\tau$ numbers that are greater than or equal to $\hat{R}$ and at least $\lambda/2-\tau$ numbers that are smaller than or equal to $\hat{R}$, where $\tau = \mathcal{O}(\alpha^{-1}\log(\lambda/\delta))$.

Although a single run of PRIVATEMEDIAN is highly private, from Thm. 2 we know that if we repeat PRIVATEMEDIAN for T times, the composed privacy level is $\sqrt{2T\log(1/\beta)}\alpha + 2T\alpha^2$, which can be large (and thus no longer private or adversarially robust) if T is large (which is reasonable for streaming tasks). Hence, we need to run PRIVATEMEDIAN as few as possible. Inspired by the SKETCHSWITCHING algorithm, we know that it suffices to output the private median for $\phi_{\epsilon/10,g}$ (i.e., flip number) times, given that we only output a new median when the previous output is not in the range $(1\pm\epsilon/2)$ of the new median. Let $\hat{R}_\rho$ determine the ρ-th output private median. At some timestamp t where $\hat{R}_\rho$ is the currently released output, we need to determine whether $\hat{R}_\rho$ is within $(1\pm\epsilon/2)$ of the median of $R_1^{(t)}, R_2^{(t)}, \cdots, R_\lambda^{(t)}$. The `Adversary` will know this information too by checking whether our algorithm still outputs $\hat{R}$ after t. Therefore, we still need to apply DP to protect this information. Despite so, to achieve α-DP, we do not need to add noise to the rank of every individual approximation as in PRIVATEMEDIAN. Instead, it suffices to add noise to the number of individual approximations whom $\hat{R}_\rho$ is not within $(1\pm\epsilon/2)$ of, because we only need to check whether this number is smaller than $\lambda/2$. This, together with the PRIVATEMEDIAN algorithm, forms the ROBUSTSKETCH algorithm as given in Alg. 2. By Thm. 2, we can show that the ROBUSTSKETCH algorithm satisfies $(1/100,\delta)$-DP (see App. B.3).

Theorem 6. The ROBUSTSKETCH algorithm satisfies (ϵ,δ)-strong g-tracking guarantee even if the stream is chosen by an `Adversary`, if we set

$$\lambda = \Omega\left(\sqrt{\phi_{\frac{\epsilon}{10},g}\log\tfrac{1}{\delta}}\cdot\log\tfrac{T}{\epsilon\delta}\right).$$

Proof. For $t\in[T]$, let $\mathbf{R}^{(t)}$ be the random variable representing the approximation to $g\left(\mathbf{f}^{(t)}\right)$ given by a random $(\epsilon/10,1/10)$-strong g-tracking sketch. Let $\mathbf{I}^{(t)} := \mathbb{1}\left[\mathbf{R}^{(t)}\in(1\pm\epsilon/10)g\left(\mathbf{f}^{(t)}\right)\right]$ be the indicator variable that equals 1 when the approximation $\mathbf{R}^{(t)}$ is within $(1\pm\epsilon/10)$ of the exact value. For $j\in[\lambda]$, define $I_j^{(t)} := 1$ if $R_j^{(t)}\in(1\pm\epsilon/10)g\left(\mathbf{f}^{(t)}\right)$ and 0 otherwise (i.e., each $I_j^{(t)}$ is a realization of $\mathbf{I}_j^{(t)}$). By the generalization property (App. A.2) of DP, we have for any $t\in[T]$

$$\Pr\left[\left|\mathbb{E}\left[\mathbf{I}^{(t)}\right]-\tfrac{1}{\lambda}\textstyle\sum_{j=1}^{\lambda}I_j^{(t)}\right|\le\tfrac{1}{10}\right]\ge 1-100\delta = 1-\mathcal{O}(\delta). \tag{1}$$

By definition of strong tracking algorithm (Def. 1), the probability that $\mathbf{I}^{(t)}=1$ is at least $9/10$ for all $t\in[T]$. Hence $\mathbb{E}\left[\mathbf{I}^{(t)}\right]\ge 9/10$. Hence by Eq. (1) with probability at least $1-\mathcal{O}(\delta)$, $(1/\lambda)\sum_{j=1}^{\lambda}I_j^{(t)}\ge 8/10$,

Algorithm 2 The ROBUSTSKETCH algorithm via differential privacy.

1: Initialize λ $(\epsilon/10, 1/10)$-strong g-tracking sketches $\mathcal{A}_1, \cdots, \mathcal{A}_\lambda$ with independent randomness
2: $\alpha \leftarrow 1/\left(400\sqrt{2\phi_{\epsilon/10,g}\log(1/\delta)}\right)$; $\rho \leftarrow 1$; $\hat{R}_1 \leftarrow g(\mathbf{0})$
3: **repeat** at most $\phi_{\epsilon/10,g}$ **times**
4: $\quad\hat{\ell} \leftarrow k/2 + \mathrm{Lap}(2/\alpha)$
5: $\quad$**repeat**
6: $\quad\quad$Receive token (i_t, c_t)
7: $\quad\quad R_j^{(t)} \leftarrow$ approximation given by $\mathcal{A}_j$ for all $j \in [\lambda]$
8: $\quad\quad$**if** $\left|R_j^{(t)} : \hat{R}_\rho \notin (1 \pm \epsilon/2)R_j^{(t)}\right| + \mathrm{Lap}(4/\alpha) < \hat{\ell}$ **then**
9: $\quad\quad\quad$**output** $\hat{R}_\rho$
10: $\quad\quad$**else**
11: $\quad\quad\quad$**break**
12: $\quad\quad$**end if**
13: $\quad$**end**
14: $\quad\rho \leftarrow \rho + 1$
15: $\quad\hat{R}_\rho \leftarrow$ PRIVATEMEDIAN $\left(R_1^{(t)}, R_2^{(t)}, \cdots, R_\lambda^{(t)}\right)$
16: $\quad$**output** $\hat{R}_\rho$
17: **end**

which means that **(§) at least** $8/10$ **of** $I_j^{(t)}$**'s are equal to** 1. Note that we can achieve probability at least $1-\delta$ (instead of $\mathcal{O}(\delta)$) by tightening the δ used in the algorithm by a constant factor (e.g., 100 in this case).

If the output $\hat{R}_\rho$ is **updated**, by Thm. 5, with probability $1-\delta$ we have $\tau = \mathcal{O}(\alpha^{-1}\log(\lambda/\delta)) \le \lambda/10$ given that $\alpha = 1/\left(400\sqrt{2\phi_{\epsilon/10,g}\log(1/\delta)}\right)$ and $\lambda = \Omega\left(\sqrt{\phi_{\epsilon/10,g}\log(1/\delta))}\cdot\log(T/\epsilon\delta)\right)$, that is, at least $4/10$ of the λ approximations are greater than or equal to $\hat{R}_\rho$ and at least $4/10$ are smaller than or equal to $\hat{R}_\rho$. By **(§)**, $\hat{R}_\rho \in (1\pm\epsilon/10)g\left(\mathbf{f}^{(t)}\right)$ too.

If the output $\hat{R}_\rho$ is **not updated**, then $\left|R_j^{(t)} : \hat{R}_\rho \notin (1\pm\epsilon/2)R_j^{(t)}\right| + \mathrm{Lap}(4/\alpha) < \lambda/2 - \mathrm{Lap}(2/\alpha)$. We bound the two Laplacian noises using the following claim:

Claim. With probability at least $1-\delta$, the absolute values of all the Laplacian noises sampled in Line 6 and Line 10 of Alg. 2 are smaller than $(4/\alpha)\log(2T/\delta)$.

Proof. We know that if $X \sim \mathrm{Lap}(4/\alpha)$ then $|X| \sim \mathrm{Exp}(\alpha/4)$. $\Pr[|X| < (4/\alpha)\log(2T/\delta)] = 1 - \exp(-(\alpha/4)(4/\alpha)\log(2T/\delta)) = 1 - \delta/(2T)$. Note that the number of Laplacian noises sampled in Line 6 and Line 10 is strictly smaller than $2T$. Also, the probability that a $\mathrm{Lap}(2/\alpha)$ noise has magnitude smaller than $(4/\alpha)\log(2T/\delta)$ is strictly larger than the probability that a $\mathrm{Lap}(4/\alpha)$ noise does so. Thus, the probability that the absolute values of all the Laplacian noises sampled in Line 6 and Line 10 are smaller than $(4/\alpha)\log(2T/\delta)$ is larger than $(1-\delta/(2T))^{2T}$, which is at least $1-\delta$. □

Thus, $\left|R_j^{(t)} : \hat{R}_\rho \in (1\pm\epsilon/2)R_j^{(t)}\right| \ge \lambda/2 - 2(4/\alpha)\log(2T/\delta) \ge 4\lambda/10$ given that $\alpha = 1/\left(400\sqrt{2\phi_{\epsilon/10,g}\log(1/\delta)}\right)$ and $\lambda = \Omega\left(\sqrt{\phi_{\epsilon/10,g}\log(1/\delta))}\cdot\log(T/\epsilon\delta)\right)$. Combined with our earlier

statement (§), there is at least one approximation $R_j^{(t)}$ such that $R_j^{(t)} \in (1 \pm \epsilon/10) g\left(\mathbf{f}^{(t)}\right)$ and $\hat{R}_\rho \in (1 \pm \epsilon/2) R_j^{(t)}$. Now we can use a similar argument to the proof of Thm. 3: since $(1+\epsilon/2)(1+\epsilon/10) < 1+\epsilon$ and $(1-\epsilon/2)(1-\epsilon/10) > 1-\epsilon$, the output $\hat{R}_\rho$ is within $(1 \pm \epsilon)$ of the exact value $g\left(\mathbf{f}^{(t)}\right)$.

Therefore, the RobustSketch algorithm is indeed a strong tracking algorithm even if the stream is chosen by an Adversary. □

Corollary 1. The space required by the RobustSketch algorithm is

$$\mathcal{O}\left(L\left(\tfrac{\epsilon}{10}, \tfrac{1}{10}\right) \cdot \sqrt{\phi_{\frac{\epsilon}{10}, g} \log \tfrac{1}{\delta}} \cdot \log \tfrac{T}{\epsilon\delta}\right).$$

Remark. One may notice that the RobustSketch algorithm may halt before all tokens (i_t, c_t) have arrived (i.e., when the $\phi_{\epsilon/10,g}$ iterations are used up). However, one can show that if we set λ according to Thm. 6, then with high probability such *early halting* will not happen (App. B.4).

4.1 Example: F_2-Approximation for Insertion-Only Streams

Now we will give an example of applying the RobustSketch algorithm to F_2-approximation for insertion-only streams. Simply, we can still use Indyk's 2-stable sketch as our non-robust $(\epsilon/10, 1/10)$-strong g-tracking sketch, which means $L(\epsilon/10, 1/10) = \tilde{\mathcal{O}}(\epsilon^{-2} \log T)$. Moreover, by Thm. 4, $\phi_{\epsilon/10,g}$ is at most $\mathcal{O}(\epsilon^{-1} \log T)$. Setting $\delta = 1/T$ and using Cor. 1, we can get that a RobustSketch algorithm that guarantees $(1 \pm \epsilon)$-approximation for all $t \in [T]$ with probability at least $1 - 1/T$ requires $\tilde{\mathcal{O}}(\epsilon^{-2.5} \log^4 T)$ space. As compared to the SketchSwitching algorithm in Sec. 3.1, we reduce the dependency on ϵ by a factor of $\sqrt{\epsilon}$ at the cost of an additional $\log^2 T$ factor. Therefore, the RobustSketch algorithm is more space efficient when a strong approximation guarantee (small ϵ) is required.

5 Discussions and Future Direction

In this writeup, we introduce the study of adversarially robust streaming algorithms and explain two important algorithms, SketchSwitching and RobustSketch, that achieve (ϵ, δ)-strong g-tracking guarantee even if the stream is chosen by an `Adversary`. Moreover, the spaces required by the two algorithms can still be sublinear (given that the base non-robust sketches use at most $\sqrt{T}$ space). RobustSketch slightly improves on the space complexity of SketchSwitching by a factor of $\sqrt{\phi_{\epsilon/10,g}}$ through utilizing more information from the base non-robust sketches, and it can be used in general turnstile streams whereas SketchSwitching focuses on insertion-only streams or turnstile streams with bounded number of deletions. However, it might suffer from early halting (i.e., before stream ends) with a small probability whereas SketchSwitching will not, which is a possible area for improvement.

Some possible future directions remain. Although both flip-number-based algorithms work successfully for a range of problems, all these problems are one-dimensional (e.g., F_2-approximation, heavy hitters, etc.), it is still unclear whether and how the concept of flip numbers can be extended to the approximation of higher-dimensional quantities. Secondly, most works in this field (including SketchSwitching and RobustSketch) focus on how to adapt from non-robust algorithms and incur additional space costs during the conversion. This leads to a gap between the optimal bounds of non-robust and robust algorithms. It is meaningful to study whether it is possible or how to reduce such a gap. For example, one possible direction is that the additional costs may not be needed if there exist some algorithms that are **inherently** adversarially robust (works such as [10] have shown that some specific algorithms can be robust by nature).

References

[1] Amit Chakrabarti. Data stream algorithms. *Computer Science*, 49:149, 2015.

[2] Omri Ben-Eliezer, Rajesh Jayaram, David P Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. *ACM Journal of the ACM (JACM)*, 69(2):1–33, 2022.

[3] Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially robust streaming algorithms via differential privacy. *Journal of the ACM*, 69(6):1–14, 2022.

[4] Cynthia Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12, 2006.

[5] Zengfeng Huang, Wai Ming Tai, and Ke Yi. Tracking the frequency moments at all times, 2014. URL `https://arxiv.org/abs/1412.1763`.

[6] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

[7] Vladimir M Zolotarev. *One-dimensional stable distributions*, volume 65. American Mathematical Soc., 1986.

[8] Jaroslaw Blasiok, Jian Ding, and Jelani Nelson. Continuous monitoring of ℓ_p norms in data streams. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, 81:32.

[9] Moritz Hardt and David P Woodruff. How robust are linear sketches to adaptive inputs? In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 121–130, 2013.

[10] Vladimir Braverman, Avinatan Hassidim, Yossi Matias, Mariano Schain, Sandeep Silwal, and Samson Zhou. Adversarial robustness of streaming algorithms through importance sampling. *Advances in Neural Information Processing Systems*, 34:3544–3557, 2021.

[11] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[12] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 117–126, 2015.

A Additional Preliminaries

A.1 Exponential Mechanism of Differential Privacy

The exponential mechanism is yet another mechanism for DP. Instead of releasing a noisy output, it would like to select from a dataset a **precise** datum privately. Formally, we define a *scoring function* $v : \mathcal{D} \times \mathcal{R} \to \mathbb{R}$ such that given a dataset D and an element $x \in D$, $v(D, x)$ returns how good the element x is. For example, if we want to know the maximum number in a dataset D, then $v(D, \max D) > v(D, x)$ for all $x \in D$ and $x \neq \max D$. The exponential mechanism states that

Theorem 7. An algorithm $\mathcal{A} : \mathcal{D} \to \mathcal{R}$ is α-differentially private if given a dataset D, $\mathcal{A}$ outputs $r \in \mathcal{R}$ with probability proportional to $\exp((\alpha v(D, r))/(2\Delta(v)))$, where $\Delta(v)$ is the sensitivity of v as defined in Def. 3.

In other words, the log probability of selecting a particular element from a dataset is proportional to its score.

Connection to the noise addition mechanism. We can construct an algorithm that satisfies Thm. 7 using Laplacian mechanism. Specifically, assume we want to know the maximum number in a dataset D. We can define the scoring function $v := \mathrm{rank}(D)$, where $\mathrm{rank}(\cdot)$ returns the rank of $\cdot$ in ascending order. The REPORTNOISYMAX algorithm works as follows:

1. Add Laplacian noise $\mathrm{Lap}(1/\alpha)$ to the score of each element in D.
2. Output the element with the highest noisy score.

[11] shows that this algorithm indeed satisfies Thm. 7 although it outputs a precise value and hence it is an α-differentially private algorithm.

A.2 Generalization Property of Differential Privacy

The generalization property [12] of DP is rather advanced for the sake of this writeup, so we delegate it to the appendix. Readers may also regard this as an established factor.

Suppose $x \in X$ follows the probability distribution p_X. The generalization property states that if we draw a random samples $x_1, x_2, \cdots, x_a$ from X and use some differentially private algorithm to obtain u predicates $h_1, h_2, \cdots, h_u$ that answer some particular questions, then the empirical means of all these privately computed predicates are all close to their expected means. Formally,

Theorem 8. For $\alpha \in (0, 1/3), \beta \in (0, \alpha/4), u \in \mathbb{N}$ and $a \geq 1/\alpha^2 \log(2\alpha u/\beta)$. Let $\mathcal{A} : X^a \to \left(2^X\right)^u$ be an (α, β)-differentially private algorithm that outputs u predicts $h_1, \ldots, h_u : X \to \{0, 1\}$. Let p_X be the distribution over X and S be the a i.i.d. samples from $\mathcal{D}$, and let $(h_1, \ldots, h_u)$ be the output of $\mathcal{A}(S)$. Then the following holds:

$$\Pr_{\substack{S \sim p_X^a \\ h_1, \cdots, h_u \leftarrow \mathcal{A}(S)}} \left[\max_{1 \leq j \leq u} \left| \frac{1}{a} \sum_{x \in S} h_j(x) - \mathbb{E}_{X \sim p_X} \left[h_j(X) \right] \right| \geq 10\alpha \right] < \frac{\delta}{\epsilon}. \tag{2}$$

B Proofs

B.1 Proof for Theorem 4

F_2 is a monotone function, and in insertion-only streaming model, $\mathbf{f}^{(t_1)} < \mathbf{f}^{(t_2)}$ for all $t_1 < t_2$. Let M be a scalar upper bound on all entries of the frequency vector. Let $C = n^{2c}$, where c is some constant satisfying $M^2 n \leq n^{2c}$. Note that $F_2(\mathbf{f}^{(1)}) \geq 1 \geq C^{-1}$ and $F_2(\mathbf{f}^{(T)}) \leq M^2 n \leq C$.

The flip number ϕ_{ϵ,F_2} is upper bounded by T and equals T when $F_2(\mathbf{f}^{(t)}) < (1-\epsilon) \cdot F_2(\mathbf{f}^{(t+1)})$ for all $t \in [T-1]$. This means that at each timestamp t, the objective value increases by a factor greater than $1/(1-\epsilon)$, and thus $(1/(1-\epsilon))^{\phi_{\epsilon,F_2}-1} C^{-1} \leq (1/(1-\epsilon))^{\phi_{\epsilon,F_2}-1} F_2(\mathbf{f}^{(1)}) < F_2(\mathbf{f}^{(\phi_{\epsilon,F_2}-1)}) \leq C$. Using the fact $1 - x \in [e^{-2x}, e^{-x}]$ for $x \in (0, 3/4)$, after taking logarithms on both sides, we get $\phi_{\epsilon,F_2} = \mathcal{O}(\epsilon^{-1} \log C) = \mathcal{O}(\epsilon^{-1} \log n)$.

B.2 Proof for Theorem 5

Consider the exponential mechanism of DP given in App. A.1. We can define the scoring function v such that if μ is the median of dataset D, then $v(D, \mu) > v(D, x)$ for any other $x \in D$ and $x \neq \mu$. Also, $v(D, x) > v(D, y)$ if the rank of x is closer to μ than y. As such, by running the REPORTNOISYMAX algorithm and add Laplacian noise to the score, we obtain an algorithm PRIVATEMEDIAN that satisfies Thm. 5.

B.3 Differential Privacy Guarantee of Robust Sketch

The ROBUSTSKETCH algorithm is basically an adaptive composition of at most $\phi_{\epsilon/10,g}$ PRIVATEMEDIAN algorithms and at most $\phi_{\epsilon/10,g}$ algorithms to determine whether $\hat{R}_\rho$ is within $(1 \pm \epsilon/2)$ of the median of approximations. Therefore, by Thm. 2, the ROBUSTSKETCH algorithm as a whole is also differentially private. Specifically, for any $\delta \in (0,1)$, the privacy level of ROBUSTSKETCH is at most $\sqrt{4\phi_{\epsilon/10,g} \log(1/\delta)} \cdot \epsilon + 4\phi_{\epsilon/10,g}\epsilon^2 = \sqrt{4\phi_{\epsilon/10,g} \log(1/\delta)} \cdot \left(1/\left(400\sqrt{2\phi_{\epsilon/10,g} \log(1/\delta)}\right)\right) + 4\phi_{\epsilon/10,g} \left(1/\left(400\sqrt{2\phi_{\epsilon/10,g} \log(1/\delta)}\right)\right)^2 \leq 1/100$.

B.4 Early Halting of Robust Sketch

When the claim in our proof for Thm. 6 holds, whenever the output $\hat{R}_\rho$ is updated, the exact value $g\left(\mathbf{f}^{(t)}\right)$ will have changed more than $(1+\epsilon/10)$ or less than $(1-\epsilon/10)$. Using a similar argument to App. B.1, we know that it will not be updated more than $\phi_{\epsilon/10,g}$. Therefore, in this case the algorithm will not halt before the stream ends. The probability of this case is no less than $1-\delta$. On the other hand, if some of the sampled Laplacian noises are surprisingly large, then early halting might happen.