Sublinear Time and Space Algorithms 2022B – Lecture 5
Adversarially Robust Streaming, Flip-Number and Sketch Switching*

Shay Sapir

1 Adversarially Robust Streaming Algorithms

Consider tracking algorithms, which track a function throughout the stream updates.

Tracking algorithms: Let $x^{(t)}$ be the frequency vector after t updates. An algorithm is said to be (ε, δ)-strong f-tracking if w.p. $1-\delta$, it outputs an estimate $R_t \in (1 \pm \varepsilon) f(x^{(t)})$ for all $t \in [m]$.

Example: ℓ_2-norm.

Exer: Design an (ε, δ)-strong ℓ_2-tracking algorithm using $O(\epsilon^{-2} \log(n/\delta))$ words.

Hint: amplify, by using $O(\log \frac{n}{\delta})$ independent copies of AMS, and apply a union bound.

Until now, we only considered streams that are fixed in advance, i.e., do not depend on choices of the algorithm. Streams that depend on outputs of the algorithm are called *adaptive*, and correspond to interactions with the environment.

Adaptive streams: For $t = 1, ..., m$,

1. Adversary chooses the next stream update σ_t.

2. Streaming algorithm process σ_t and outputs an estimate R_t.

3. Adversary observes R_t.

Strong tracking algorithms in this model are called *Adversarially Robust*.

Note: The previous analysis of a strong tracking algorithm breaks, since the outputs of the copies are not independent (informally, they depend on the stream, which depend on the other copies).

[Hardt-Woodruff, 2013]: Every adversarially robust linear sketch for ℓ_2-norm must have sketching dimension $\Omega(n)$.

Intuition: the adversary can use the observations to learn the sketching matrix A (this is the

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

difficult, technical part, and was not discussed), and then query a non-zero vector x such that $Ax = 0$. The algorithm then must report the same output for x and $2x$.

2 Flip Number and Sketch Switching

Theorem 1 [Ben-Eliezer,Jayaram,Woodruff,Yogev,2020]: In insertion-only streams, there is an adversarially robust $(1+\epsilon)$-approximation algorithm for ℓ_2-norm using $\tilde{O}(\varepsilon^{-3})$ bits of space.

Flip number [BJWY20]: For a function f and stream σ, the flip number $\lambda_\varepsilon(f, \sigma)$ is the size k of the largest subsequence $1 \leq t_1 \leq ... \leq t_k \leq m$ such that $f(x^{(t_i)}) \notin (1 \pm \varepsilon) f(x^{(t_{i+1})})$ for all $i \in [k-1]$. The flip number of f is $\lambda_\varepsilon(f) = \max_\sigma \{\lambda_\varepsilon(f, \sigma)\}$.

Lemma 2: In insertion-only streams, $\lambda_\varepsilon(\|\cdot\|_2) = O(\varepsilon^{-1} \log n)$.

Proof: Was seen in class.

Algorithm Sketch Switching:

1. Init:
   (a) $\rho = 1, g = 0$.
   (b) initialize $\lambda = \lambda_{\varepsilon/8}(\|\cdot\|_2)$ independent copies of an $(\varepsilon/8, \delta/\lambda)$-strong ℓ_2-tracking algorithm, denoted $A_1, ..., A_\lambda$.
2. Update + Output: $\forall \sigma_t$,
   (a) insert σ_t to $A_1, ..., A_\lambda$.
   (b) $y \leftarrow$ current output of A_ρ.
   (c) if $g \notin (1 \pm \varepsilon/2)y$, then set $g = y$ and increment ρ.
   (d) output g_t.

Space: $\tilde{O}(\epsilon^{-2}\lambda) = \tilde{O}(\epsilon^{-3})$ bits.

Analysis: We can assume the adversary is deterministic (Yao's principle). Why? Consider randomized adversary s.t. the algorithm fails w.p. p (over the randomness of alg + adversary). Then by an averaging argument, there is at least one choice for the randomness of the adversary for which the streaming algorithm fails w.p. p. Fix that randomness.

The proof is by induction. Let t_ρ be the time-step when the "if condition" was fulfilled for ρ, hence from this time on, the algorithm "switches" to $A_{\rho+1}$. Let $y_\rho = A_\rho(t_\rho)$. Consider an 'imaginary' output sequence, where the output for all $t > t_\rho$ is y_ρ. For this output, the adversary responds with a stream that is independent of $A_{\rho+1}$. Hence, $A_{\rho+1}$ is correct w.p. $1 - \delta/\lambda$ until the next time that the "if condition" is fulfilled, and thus $g_t \in (1 \pm \epsilon)\|x^{(t)}\|_2$.

Exer: Prove that the "if condition" is fulfilled at most $\lambda_{\varepsilon/8}(\|\cdot\|_2)$ times.

Exer: Design an adversarially robust streaming algorithm for ℓ_1/ℓ_2 point queries.

3 Streams with deletions

λ can be as large as m if there are many insertions + deletions to the same coordinate.

More techniques:

1. Using differential privacy to protect the random coins of the algorithm, achieving space $\tilde{O}_\epsilon(\sqrt{\lambda})$ [Hassidim, Kaplan, Mansour, Matias and Stemmer, 2022].[1]
2. Using sparse recovery for a sparse-dense tradeoff, to achieve space $\tilde{O}_\epsilon(m^{2/5})$ [Ben-Eliezer, Eden, Onak 2022].

We discussed the following weaker version of BEO.

Theorem 3: There is an adversarially robust streaming algorithm for ℓ_2-norm using $\tilde{O}_\varepsilon(m^{2/3})$ bits of space.

Sparse Recovery:

Input: x is a k-sparse vector.

Goal: recover x using a linear sketch of dimension $\tilde{O}(k)$.

Exer: Show that CountMin/CountSketch with $\tilde{O}(k)$ buckets solve Sparse Recovery.

High level algorithm of Theorem 3:

- If x is T-sparse, then maintain it explicitly.
- If x is T-dense (not sparse), then use the Sketch Switching algorithm.
- Use Sparse Recovery algorithm to recover x when it becomes sparse (after being dense).

Exer: Show that for T-dense vectors, $\lambda_\epsilon(\|\cdot\|) = O_\varepsilon(m/\sqrt{T})$.

Space: the Sparse Recovery algorithm uses $\tilde{O}(T)$ bits, and the Sketch Switching uses $\tilde{O}_\varepsilon(m/\sqrt{T})$ bits of space. Pick $T = m^{2/3}$, resulting in the desired space bound.

One may need to track the number of non-zeros in x in order to decide if the regime is sparse or dense. This can be done using a Distinct Elements algorithm for streams with deletions, and by a slight change of the algorithm: make it such that if x is $2T$-dense then we use Sketch Switching, and if $\|x\|_0 \in [T, 2T]$, then either maintain it explicitly or use Sketch Switching.

[1] The notation $\tilde{O}_\varepsilon(\cdot)$ hides multiplicative factor $\text{poly}(\varepsilon, \log n)$.