Separating Adaptive Streaming from Oblivious Streaming using the Bounded Storage Model

Uri Stemmer

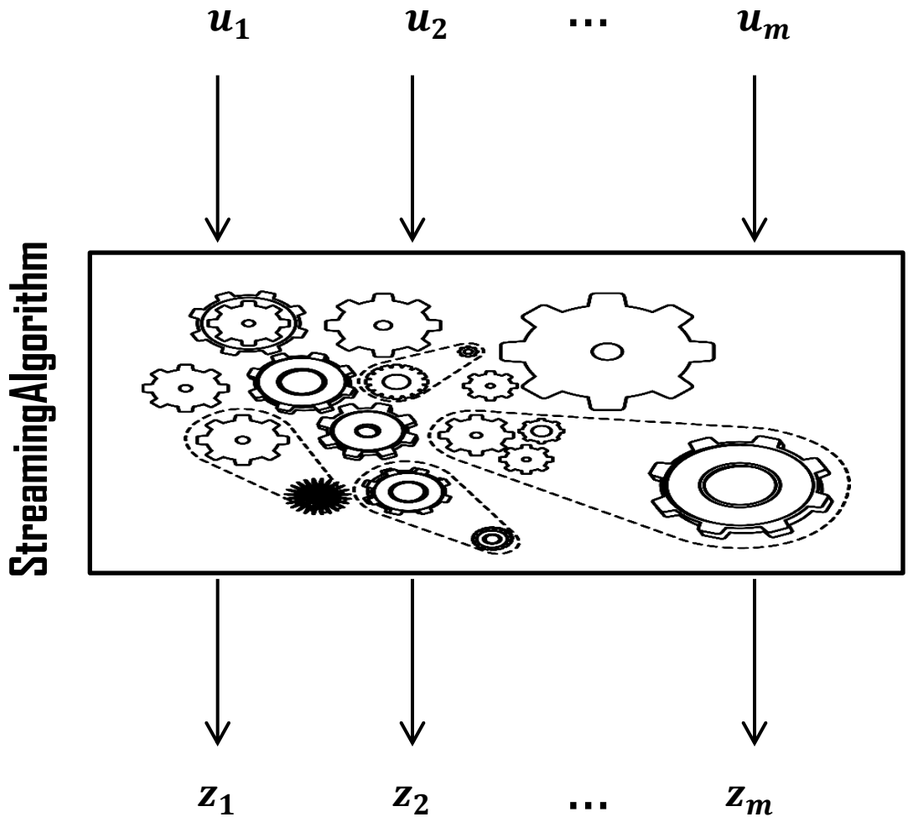
+Haim Kaplan +Yishay Mansour +Kobbi Nissim

Consider a streaming algorithm that continuously estimates the value of the desired function (The goal is to minimize space requirements)

Oblivious Streaming

[Alon, Matias, Szegedy '96]

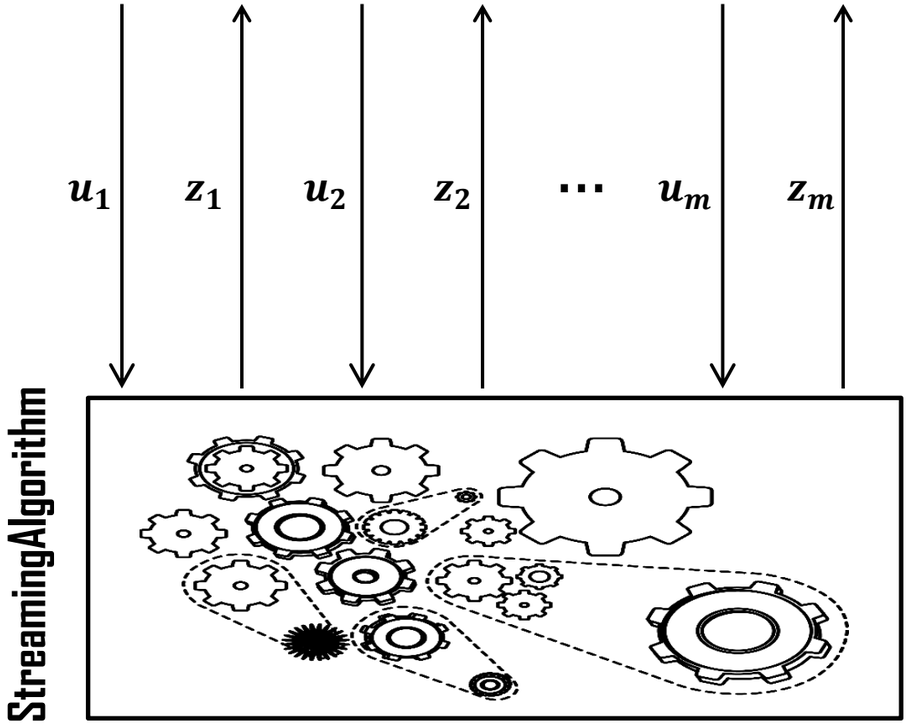
$(u_1, u_2, \ldots, u_m)$ = fixed stream (unknown to the algorithm)



Adaptive Streaming

[Hard, Woodruff '13], [Ben-Eliezer, Jayaram, Woodruff, Yogev '20]

Adversary chooses u_i based on previous answers

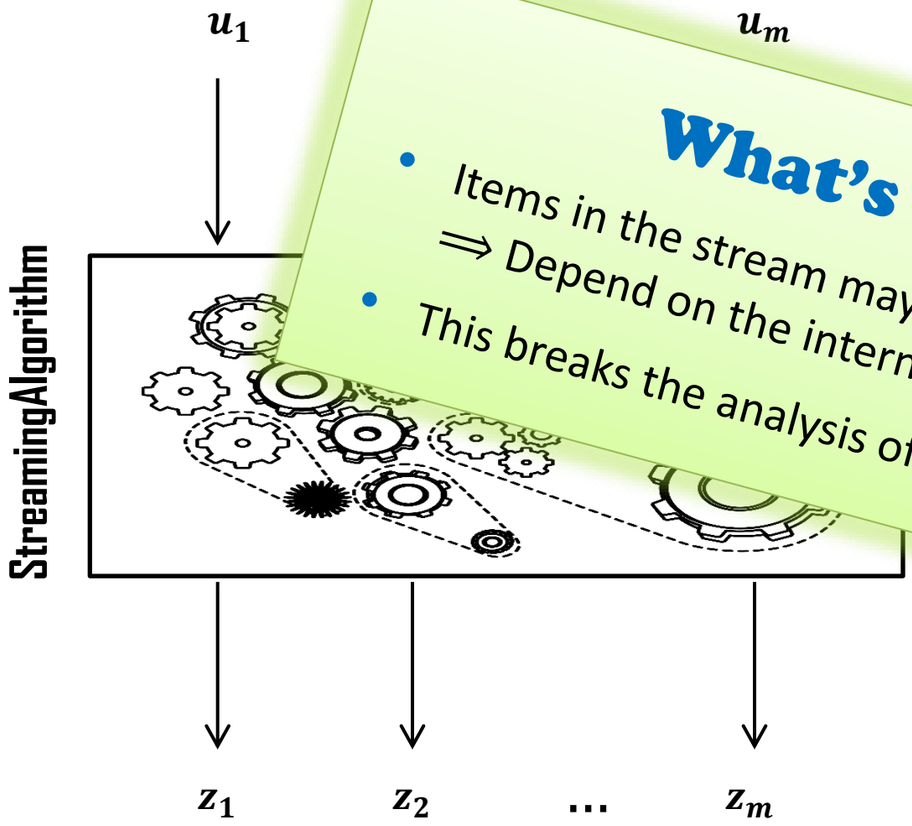


Consider a streaming algorithm that continuously estimates the value of the desired function
(The goal is to minimize space requirements)

Oblivious Streaming

[Alon, Matias, Szegedy '96]

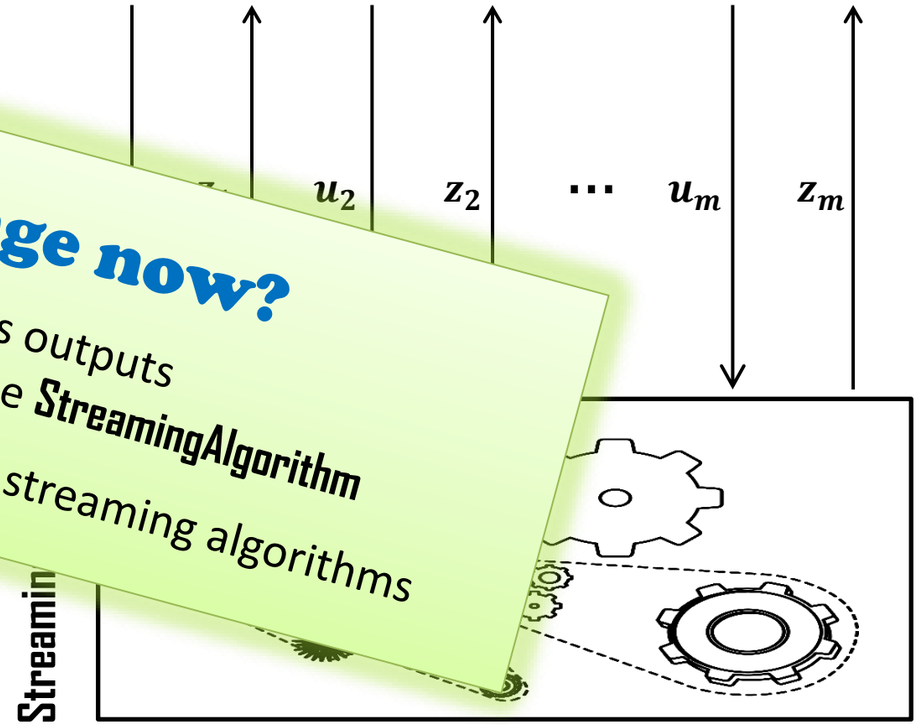
$(u_1, u_2, \dots, u_m)$ = fixed stream (unknown to the algorithm)



Adaptive Streaming

[Hard, Woodruff '13], [Ben-Eliezer, Jayaram, Woodruff, Yogev '20]

Adversary chooses u_i based on previous answers



What's the challenge now?

- Items in the stream may depend on previous outputs
  ⟹ Depend on the internal randomness of the **StreamingAlgorithm**
- This breaks the analysis of classical (oblivious) streaming algorithms

The Adversarial Streaming Model

HW13, BJWY20

- Fix a function g mapping a (prefix of the) stream to a real number, and an approximation parameter α
  - E.g., g might count the number of distinct elements in the stream
- Two-player game between a (randomized) **StreamingAlgorithm** and an **Adversary**
- In the ith round:
  1. The **Adversary** chooses an update u_i for the stream, which can depend on all previous stream updates and outputs of **StreamingAlgorithm**
  2. The **StreamingAlgorithm** processes the new update and outputs its current response z_i
- The goal of the **Adversary** is to make the **StreamingAlgorithm** output an incorrect response z_i at some point i

Do oblivious streaming algorithms work in the adversarial model?

- Deterministic streaming algorithms are adversarially robust
  - However, many streaming algorithms provably **must** be randomized [AMS '96]
- Many of the randomized streaming algorithms are *not* adversarially robust

Informal takeaway: The difficulty with adversarial streaming is that as time goes by the adversary might learn information about the internal randomness of the algorithm

Summary of Existing and New Results

Several works presented positive results:

Transformations from oblivious algorithms into adversarially robust algorithm, with small space overhead

Definition [BJWY'20]: The (α, m)-***flip number*** of a function g, denoted as λ, is the max number of times that the value of g can change by a factor of $(1+\alpha)$ during a stream of length m

[BJWY'20]:	Oblivious alg $\mathcal{A} \implies$ Adversarially robust alg using space $O\left(\lambda \cdot \mathbf{Space}(\mathcal{A})\right)$
[HKMMS'20]:	Oblivious alg $\mathcal{A} \implies$ Adversarially robust alg using space $\widetilde{O}\left(\sqrt{\lambda} \cdot \mathbf{Space}(\mathcal{A})\right)$
[WZ'21], informal:	Oblivious alg $\mathcal{A} \implies$ Adversarially robust alg using space $\widetilde{O}\left(\alpha \cdot \lambda \cdot \mathbf{Space}(\mathcal{A})\right)$
[ACSS'21], informal:	Oblivious alg $\mathcal{A} \implies$ Adversarially robust alg using space $\widetilde{O}\left(\sqrt{\alpha \cdot \lambda} \cdot \mathbf{Space}(\mathcal{A})\right)$

Negative results:

[HW'13]:	Impossibility results for linear streaming algorithms
New:	**Separating Adaptive Streaming from Oblivious Streaming:** There is a streaming problem that requires $\tilde{\Omega}\left(\sqrt{\lambda}\right)$ in the adversarial setting (even by non-linear algorithms) but can be solved in the oblivious setting using space $\tilde{O}\left(\log^2 \lambda\right)$

Adversarial Streaming, Differential Privacy, and Adaptive Data Analysis

Talk Outline

1. What is adversarial streaming?

2. Related works

3. Statement of our results

4. What is adaptive data analysis?

5. Negative results for adversarial streaming via adaptive data analysis

6. Summary

What is the Problem of Adaptive Data Analysis (ADA)?

Warmup: Simple Problem #1

- Let $\mathfrak{D}$ be an unknown distribution over a domain X
- Fix a function $h: X \to \{0, 1\}$ and a parameter α
- Input: Sample $S = (x_1, \ldots, x_n)$ containing n iid samples from $\mathfrak{D}$
- The Goal: Estimate $\mathbb{E}_{x \sim \mathfrak{D}}[h(x)]$ to within error $\pm\alpha$

The Questions:

- How should we compute our estimation?
- What should n be in order to ensure error at most α w.h.p. ?

Solution:

- Output the empirical average $\frac{1}{n}\sum_{i=1}^{n} h(x_i)$
- By the Hoeffding bound, suffices to take $n \gtrsim \frac{1}{\alpha^2} \cdot \ln\left(\frac{1}{\beta}\right)$

What is the Problem of Adaptive Data Analysis (ADA)?

Warmup: Simple Problem #1

- Let $\mathfrak{D}$ be an unknown distribution over a domain X
- Fix a function $h: X \to \{0, 1\}$ and a parameter α
- Input: Sample $S = (x_1, \ldots, x_n)$ containing n iid samples from $\mathfrak{D}$
- The Goal: Estimate $\mathbb{E}_{x \sim \mathfrak{D}}[h(x)]$ to within error $\pm\alpha$

Warmup: Simple Problem #2 (same as before but with k functions)

- Let $\mathfrak{D}$ be an unknown distribution over a domain X
- Fix k functions $h_1, \ldots, h_k: X \to \{0, 1\}$ and a parameter α
- Input: Sample $S = (x_1, \ldots, x_n)$ containing n iid samples from $\mathfrak{D}$
- The Goal: Estimate each $\mathbb{E}_{x \sim \mathfrak{D}}[h_i(x)]$ to within error $\pm\alpha$

Importantly: all k functions are fixed before sampling S

Same Solution:

- Output the empirical average of each of the k functions
- By the Hoeffding bound (and a union bound), suffices to take $n \gtrsim \frac{1}{\alpha^2} \cdot \ln\left(\frac{k}{\beta}\right)$

What is the Problem of Adaptive Data Analysis (ADA)?

Warmup: Simple Problem #1

- Let $\mathfrak{D}$ be an unknown distribution over a domain X
- Fix a function $h: X \to \{0, 1\}$ and a parameter α
- Input: Sample $S = (x_1, \dots, x_n)$ containing n iid samples from $\mathfrak{D}$
- The Goal: Estimate $\mathbb{E}_{x \sim \mathfrak{D}}[h(x)]$ to within error $\pm\alpha$

Warmup: Simple Problem #2 (same as before but with k functions)

- Let $\mathfrak{D}$ be an unknown distribution over a domain X
- Fix k functions $h_1, \dots, h_k: X \to \{0, 1\}$ and a parameter α
- Input: Sample $S = (x_1, \dots, x_n)$ containing n iid samples from $\mathfrak{D}$
- The Goal: Estimate each $\mathbb{E}_{x \sim \mathfrak{D}}[h_i(x)]$ to within error $\pm\alpha$

Importantly: all k functions are fixed before sampling S

The Basic ADA Problem (the functions are chosen adaptively)

- Let $\mathfrak{D}$ be an unknown distribution over a domain X
- Input: Sample $S = (x_1, \dots, x_n)$ containing n iid samples from $\mathfrak{D}$
- For $i = 1, 2, \dots, k$:
  - Get a function $h_i: X \to \{0, 1\}$ and output an answer a_i
- The Goal: Every answer a_i is α-accurate w.r.t. $\mathbb{E}_{x \sim \mathfrak{D}}[h_i(x)]$

The difficulty now is that the choice of h_i may depend on previous answers, and hence depend on S

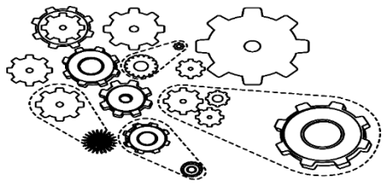
The Basic ADA Problem

Dwork, Feldman, Hardt, Pitassi, Reingold, Roth 2015

Unknown dist. $\mathfrak{D}$
over domain X

iid

$S = (x_1, \ldots, x_n)$



Mechanism $\mathcal{M}$

$h_1 : X \to \{0,1\}$

$a_1 \in \mathbb{R}$

$h_2 : X \to \{0,1\}$

$a_2 \in \mathbb{R}$

$\vdots$

$h_k : X \to \{0,1\}$

$a_k \in \mathbb{R}$

Data analyst

The Goal: With high probability, every answer a_i is α-accurate w.r.t. $\mathbb{E}_{x \sim \mathfrak{D}}[h_i(x)]$

The Question: What should $n = n(k, \alpha)$ be in order to ensure this?

Can we answer each query with the empirical average?

No! This can fail after just 1 answer!

Known Bounds for the ADA Problem

Upper Bounds

Dwork, Feldman, Hardt, Pitassi, Reingold, Roth 2015

Bassily, Nissim, Smith, Steinke, Stemmer, Ullman 2016

- ∃ efficient mechanism with

$$n = O\left(\sqrt{k}/\alpha^2\right)$$

- ∃ inefficient mechanism with

$$n = O\left(\frac{1}{\alpha^3} \cdot \sqrt{\log|X|} \cdot \log k\right)$$

Lower Bounds

Hardt, Ullman 2014

Steinke, Ullman 2015

- Assuming OWF, ∀ efficient mechanism:

$$n = \Omega\left(\sqrt{k}/\alpha\right)$$

- ∀ mechanism:

$$n = \Omega\left(\min\left\{\frac{\sqrt{\log|X|}}{\alpha}, \frac{\sqrt{k}}{\alpha}\right\}\right)$$

Adversarial Streaming, Differential Privacy, and Adaptive Data Analysis

Talk Outline

1. What is adversarial streaming?

2. Related works

3. Statement of our results

4. What is adaptive data analysis?

5. Negative results for adversarial streaming via adaptive data analysis

6. Summary

Negative Results for Adversarial Streaming via ADA

- We show a reduction from the ADA problem to the problem of adversarial streaming
- Specifically, we design a specific streaming problem such that solving it in the adversarial setting implies algorithms for the ADA problem with related parameters

The Streaming Adaptive Data Analysis (SADA) Problem – informal

- Every update in the stream $\boldsymbol{u_i}$ is either a data point $\boldsymbol{u_i = p_i \in X}$ or a function $\boldsymbol{u_i = h_i : X \to \{0,1\}}$
- **The Goal:** On every time step $\boldsymbol{i}$, after obtaining $\boldsymbol{u_i}$, we need to output an approximation to the average of the last function on the multiset containing all data points

Intuitively:

- Solving the SADA problem in the **oblivious** streaming model is easy (like basic problem #2) because we can maintain a **small** representative sample of the data points and use it to estimate **many** functions
- Solving the SADA problem in the **adversarial** streaming model is hard because if we try to estimate **many** functions using a sample then in the adaptive setting this sample must by **large**

Negative Results for Adversarial Streaming via ADA

- We show a reduction from the ADA problem to the problem of adversarial streaming
- Specifically, we design a specific streaming problem such that solving it in the adversarial setting implies algorithms for the ADA problem with related parameters

The Streaming Adaptive Data Analysis (SADA) Problem – informal

- Every update in the stream $\boldsymbol{u_i}$ is either a data point $\boldsymbol{u_i = p_i \in X}$ or a function $\boldsymbol{u_i = h_i : X \to \{0,1\}}$
- **The Goal:** On every time step $\boldsymbol{i}$, after obtaining $\boldsymbol{u_i}$, we need to output an approximation to the average of the last function on the multiset containing all data points

Challenge:

- The strong negative results for the ADA problem only hold for ***computationally efficient*** algorithms
- We aim for an information-theoretic separation, and hence cannot use these impossibility results as is
- Negative results for ADA rely on ***encryption schemes*** whose security hold under computational assumptions
- We modify the construction using cryptographic tools from the ***bounded storage model*** [Maurer 1990]

Summary

We put forward a connection between adversarial streaming and adaptive data analysis

Negative results via a reduction to the ADA problem:

There is a streaming problem that requires $\widetilde{\Omega}\left(\sqrt{\lambda}\right)$ to be solved in the adversarial setting (even by non-linear algorithms) but can be solved in the oblivious setting using space $\widetilde{O}\left(\log^2 \lambda\right)$

- **This is the first general separation between classical (oblivious) streaming and adversarial streaming**

Open Questions

1. Is λ the "correct" parameter to look at?
   * Recall: λ = number of times the value to be estimated can change by a factor of $(1+\alpha)$
2. Can we show a separation for "natural" streaming problems? What about F2 (with deletions)?
3. What happens for symmetric functions?