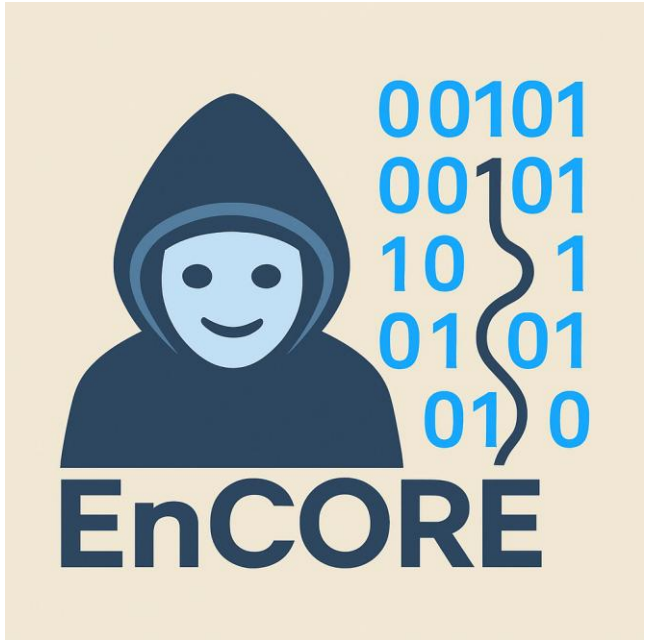
Adversarial Robustness: A Tutorial

Samson Zhou

UC San Diego



Why Adversarial Robustness?

- Adversarial input
  - Contested environments

Why Adversarial Robustness?

- Adversarial input
  - Adversarial ML

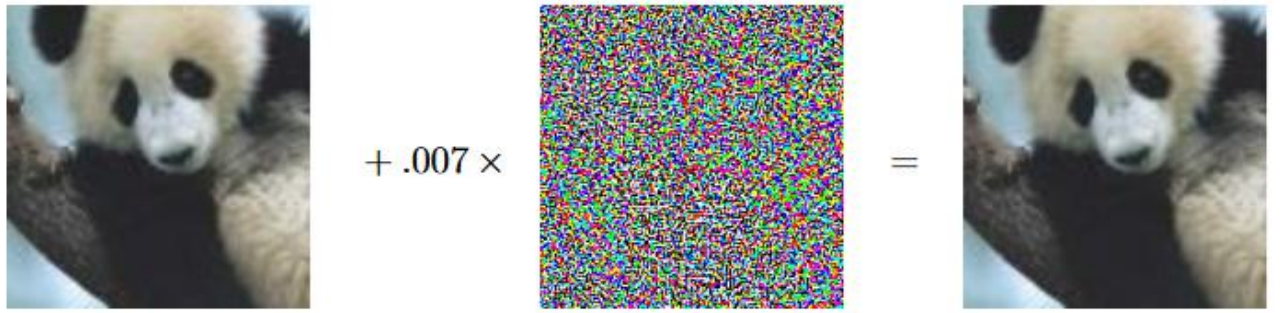
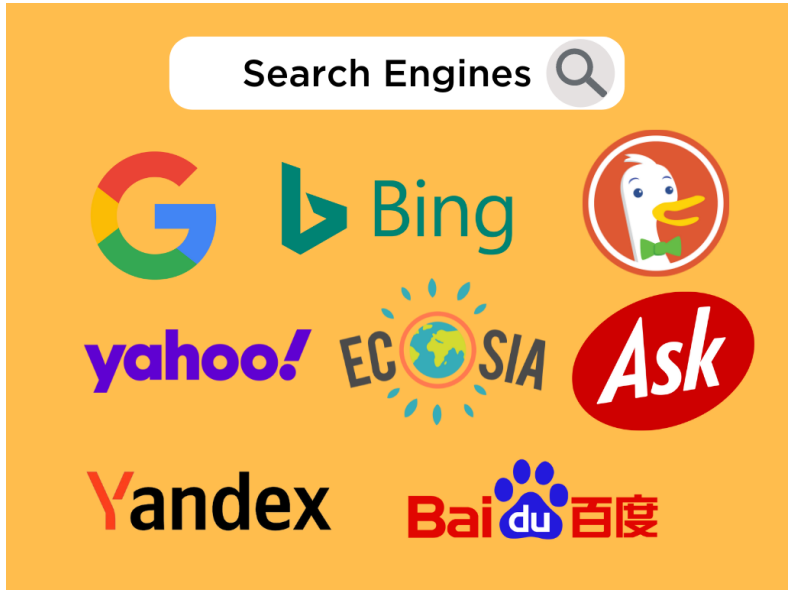


Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

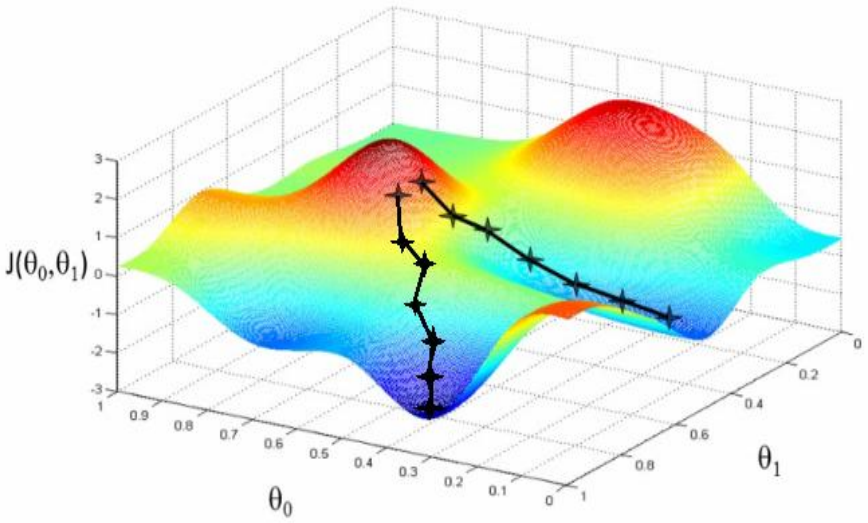
Why Adversarial Robustness?

- Adaptive input
  - Repeated interactions

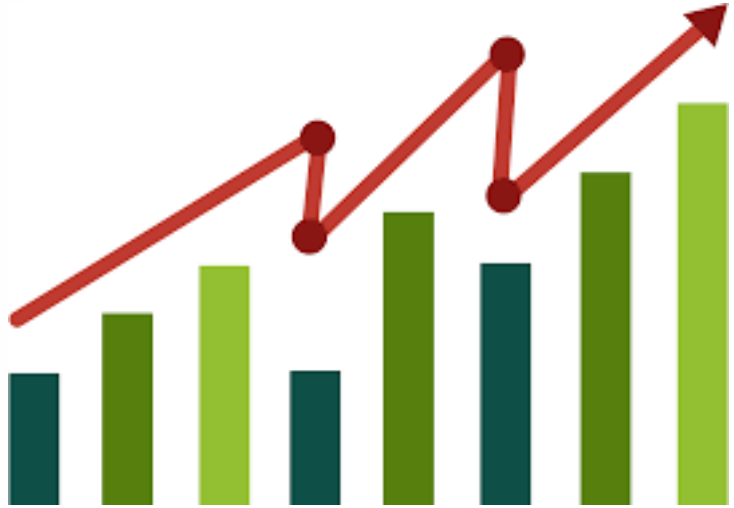
Database Queries



Stochastic Optimization



Financial Analytics



Sampling and Adversarial Input

- See a sequence of n updates, maintain a "representative" sample at all times

- Reservoir sampling: On the t-th item, replace the current sample with probability $\frac{1}{t}$

- Can always make the random sample among the largest (or smallest) element of a set [Ben-EliezerYogev20]

Sampling and Adversarial Input

- Have a “working range”, starting with the interval $[0,1]$ and $x_1 = 0.5$
- If x_1 is sampled, change interval to $[0,0.5]$
- Otherwise change to $[0.5, 1]$

- If item is sampled, change interval to “smaller”, else “larger”

- Sampled item among the $O(\log n)$ largest elements [Ben-EliezerYogev20]

Upshot

- Even fundamental primitives such as sampling are not robust!
- Need new techniques

- NEW Connections across many areas:
  - Sublinear algorithms
  - Differential privacy
  - Adaptive data analysis
  - Cryptography
  - Lattice theory

Workshop Themes

- What attacks are possible against classical data structure and algorithms? How efficient can we make them?

  Xin, Mihir, Sara, Edith

- What are the capabilities and limitations of robust data structures and algorithms in various models? How do different adversarial models change what's possible?

  Omri, Amit, Guy, David, Elena, Honghao

- New problems and models of adversarial robustness

  Uri, Binghui, Sandeep, Chen, Shenghao, Yuhan

Model #1: Streaming Model

- Input: Elements of an underlying data set S, which arrives sequentially
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

1 0 1 1 1 0 0 1

Frequency Vector

- Given a set S of m elements from $[n]$, let f_i be the frequency of element i. (How often it appears)

$$1\ 1\ 2\ 1\ 2\ 1\ 1\ 2\ 3 \rightarrow [5, 3, 1, 0] \coloneqq f$$

Frequency Moments

- Given a set S of m elements from $[n]$, let f_i be the frequency of element i. (How often it appears)
- Let F_p be the frequency moment of the vector:

$$F_p = f_1^p + f_2^p + \cdots + f_n^p$$

- Goal: Given a set S of m elements from $[n]$ and an accuracy parameter ε, output a $(1+\varepsilon)$-approximation to F_p
- Motivation: Entropy estimation, linear regression

F_2 Estimation

- AMS algorithm gives $(1+\varepsilon)$-approximation to F_2 using $O\left(\frac{1}{\varepsilon^2}\log n\right)$ bits of communication [AlonMatiasSzegedy99]

- Let $s \in \{-1,+1\}^n$ be a random sign vector
- Let $Z = \langle s, f \rangle = s_1 f_1 + s_2 f_2 + \cdots + s_n f_n$

AMS Algorithm

- $Z^2 = \sum_{i,j\in[n]} s_i s_j f_i f_j$
- $\sum_{i=j} s_i s_j f_i f_j = (f_1^2 + f_2^2 + \cdots + f_n^2) \coloneqq F_2$

- $\mathrm{E}[Z^2] = \sum_{i,j\in[n]} s_i s_j f_i f_j$
- $\mathrm{Var}[Z^2] \leq \mathrm{E}[Z^4] = \sum_{a,b,c,d\in[n]} s_a s_b s_c s_d f_a f_b f_c f_d \leq 6F_2^2$

- Chebyshev's inequality: the average of $O\left(\frac{1}{\varepsilon^2}\right)$ independent instances gets $(1+\varepsilon)$-approximation with probability at least $\frac{2}{3}$

Heavy-Hitters

- Given a set S of m elements from $[n]$, let f_i be the frequency of element i. (How often it appears)
- Let L_2 be the norm of the frequency vector:

$$L_2 = \sqrt{f_1^2 + f_2^2 + \cdots + f_n^2}$$

- Goal: Given a set S of m elements from $[n]$ and a threshold ε, output the elements i such that $f_i > \varepsilon\, L_2$...and no elements j such that $f_j < \frac{\varepsilon}{16}\, L_2$
- Motivation: DDoS prevention, iceberg queries

Distinct Elements

- Given a set S of m elements from $[n]$, let f_i be the frequency of element i. (How often it appears)
- Let F_0 be the frequency moment of the vector:

$$F_0 = |\{i : f_i \neq 0\}|$$

- Goal: Given a set S of m elements from $[n]$ and an accuracy parameter ε, output a $(1+\varepsilon)$-approximation to F_0
- Motivation: Traffic monitoring

$(1+\varepsilon)$-Approximation Streaming Algorithms

- Space $O\left(\frac{1}{\varepsilon^2}+\log n\right)$ algorithm for F_0 [KaneNelsonWoodruff10], [Blasiok20]
- Space $O\left(\frac{1}{\varepsilon^2}\log n\right)$ algorithm for F_p with $p \in (0,2]$ [BlasiokDingNelson17]
- Space $O\left(\frac{1}{\varepsilon^2} n^{1-2/p} \log^2 n\right)$ algorithm for F_p with $p > 2$ [Ganguly11, GangulyWoodruff18]
- Space $O\left(\frac{1}{\varepsilon^2}\log n\right)$ algorithm for L_2-heavy hitters [BravermanChestnutIvkinNelsonWangWoodruff17]

Model #2: Adversarially Robust Streaming

- Input: Elements of an underlying data set S, which arrives sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

1

1

Model #2: Adversarially Robust Streaming

- Input: Elements of an underlying data set S, which arrives sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

10

1

Model #2: Adversarially Robust Streaming

- Input: Elements of an underlying data set S, which arrives sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

101

2

Model #2: Adversarially Robust Streaming

- Input: Elements of an underlying data set S, which arrives sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

1010

3

Model #2: Adversarially Robust Streaming

- Input: Elements of an underlying data set S, which arrives sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

1010

3

Model #2: Adversarially Robust Streaming

- Input: Elements of an underlying data set S, which arrives sequentially and *adversarially*
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

- Adversarially robust: "Future queries may depend on previous queries"
- Motivation: Database queries, adversarial ML

“Attack” on AMS

- Can learn whether $s_i = s_j$ from $\langle s, e_i + e_j \rangle$
- Let $f_i = 1$ if $s_i = s_1$ and $f_i = -1$ if $s_i \neq s_1$
- $Z = \langle s, f \rangle = s_1 f_1 + \cdots + s_n f_n = m$ and $Z^2 = m^2$ deterministically

- What happened? Randomness of algorithm not independent of input

Reconstruction Attack on Linear Sketches

- Linear sketches are “not robust” to adversarial attacks, must use $\Omega(n)$ space [HardtWoodruff13]
- Approximately learn sketch matrix U, query something in the kernel of U
- Iterative process, start with $V_1, \dots, V_r$

- Correlation finding: Find vectors strongly correlated with vector v in orthogonal subspace to V_{i-1}
- Progress: Set $V_i = \text{span}(V_{i-1}, v)$

Insertion-Only Streams

- Key: Deletions are needed to perform this attack

- Assume insertion-only updates
- How do the previous results work?

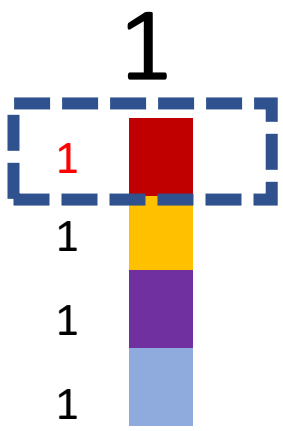
Robust Algorithms

- Suppose we are trying to approximate some given function
  - Suppose we have a streaming algorithm for this function
  - Suppose this function is monotonic and the stream is insertion-only

- Sketch switching framework [Ben-EliezerJayaramWoodruffYogev20] gives a robust for this function

Sketch Switching

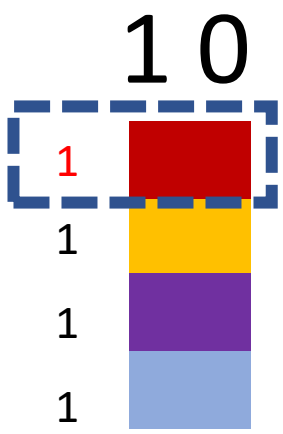
- Start many instances of the streaming algorithm at the beginning
- Use an instance of the algorithm but “freeze” the output
- Each time the next instance has value $(1 + O(\varepsilon))$ more than the “frozen” output, use the next instance and “freeze” its output



- Example: Number of ones in the stream (2-approximation)

Sketch Switching

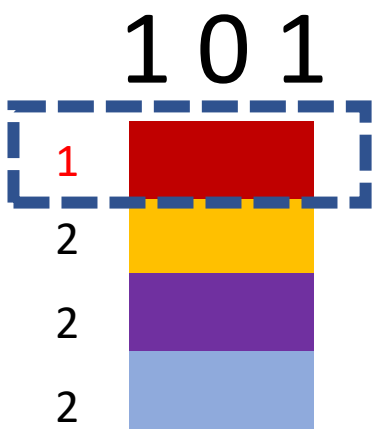
- Start many instances of the streaming algorithm at the beginning
- Use an instance of the algorithm but “freeze” the output
- Each time the next instance has value $(1 + O(\varepsilon))$ more than the “frozen” output, use the next instance and “freeze” its output



- Example: Number of ones in the stream (2-approximation)

Sketch Switching

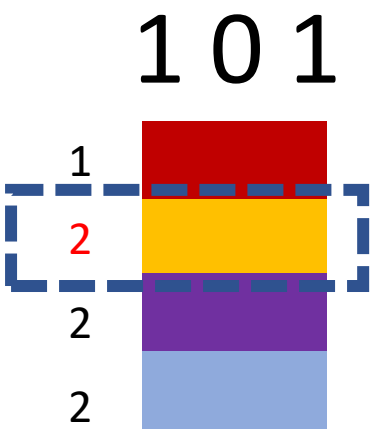
- Start many instances of the streaming algorithm at the beginning
- Use an instance of the algorithm but “freeze” the output
- Each time the next instance has value $(1 + O(\varepsilon))$ more than the “frozen” output, use the next instance and “freeze” its output



- Example: Number of ones in the stream (2-approximation)

Sketch Switching

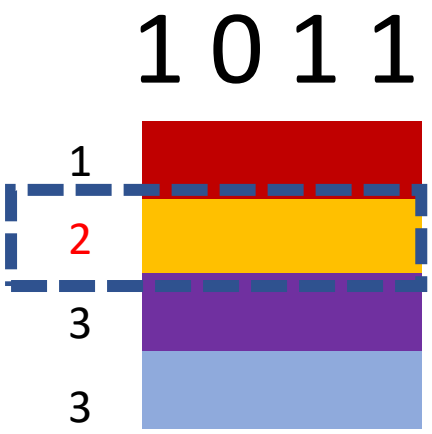
- Start many instances of the streaming algorithm at the beginning
- Use an instance of the algorithm but “freeze” the output
- Each time the next instance has value $(1 + O(\varepsilon))$ more than the “frozen” output, use the next instance and “freeze” its output



- Example: Number of ones in the stream (2-approximation)

Sketch Switching

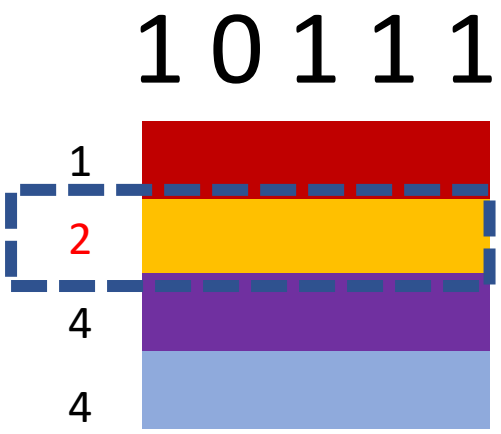
- Start many instances of the streaming algorithm at the beginning
- Use an instance of the algorithm but “freeze” the output
- Each time the next instance has value $(1 + O(\varepsilon))$ more than the “frozen” output, use the next instance and “freeze” its output



- Example: Number of ones in the stream (2-approximation)

Sketch Switching

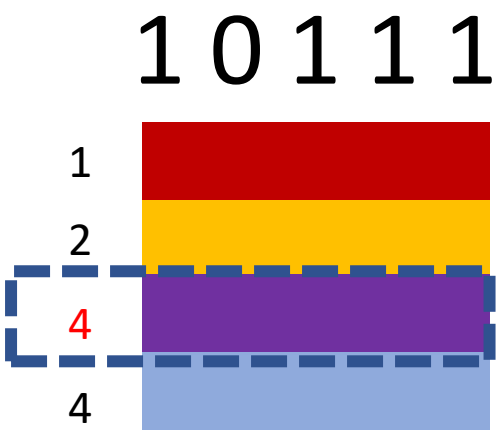
- Start many instances of the streaming algorithm at the beginning
- Use an instance of the algorithm but “freeze” the output
- Each time the next instance has value $(1 + O(\varepsilon))$ more than the “frozen” output, use the next instance and “freeze” its output



- Example: Number of ones in the stream (2-approximation)

Sketch Switching

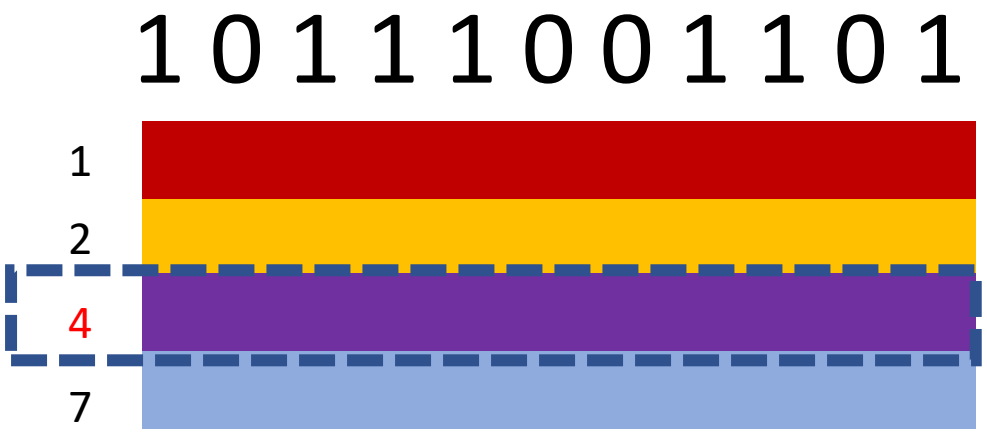
- Start many instances of the streaming algorithm at the beginning
- Use an instance of the algorithm but “freeze” the output
- Each time the next instance has value $(1 + O(\varepsilon))$ more than the “frozen” output, use the next instance and “freeze” its output



- Example: Number of ones in the stream (2-approximation)

Sketch Switching

- Start many instances of the streaming algorithm at the beginning
- Use an instance of the algorithm but “freeze” the output
- Each time the next instance has value $(1 + O(\varepsilon))$ more than the “frozen” output, use the next instance and “freeze” its output



- Example: Number of ones in the stream (2-approximation)
- Number of ones stream is at least 4 and at most 8
- 4 is a good approximation

Sketch Switching Summary

- Sketch switching for robust algorithms uses $\frac{1}{\varepsilon^2}$ space each time F_p increases by $(1+\varepsilon)$ and function increases $O\left(\frac{1}{\varepsilon}\log n\right)$ times

- How much space do “typical” algorithms use?

$(1+\varepsilon)$-Robust Algorithms [Ben-EliezerJayaramWoodruffYogev20]

- Space $\tilde{O}\left(\frac{1}{\varepsilon^3}\log n\right)$ algorithm for F_0
- Space $\tilde{O}\left(\frac{1}{\varepsilon^3}\log n\right)$ algorithm for F_p with $p \in (0,2]$
- Space $\tilde{O}\left(\frac{1}{\varepsilon^3}n^{1-2/p}\right)$ algorithm for F_p with $p > 2$
- Space $\tilde{O}\left(\frac{1}{\varepsilon^3}\log n\right)$ algorithm for L_2-heavy hitters

"A general framework that loses* nothing in n and only $\frac{1}{\varepsilon}$ "

Computation Paths

- Upper bound the number of total possible inputs
- Union bound over all of these inputs

- How many different streams can there be?
- Length m stream can increase by $(1+\varepsilon)$ at most $L = O\left(\frac{1}{\varepsilon}\log m\right)$ times, so there are roughly $\binom{m}{L} = m^{O(L)}$ possible inputs
- Failure probability $\frac{1}{m^{O(L)}}$ needed, set $\log\frac{1}{\delta} = O(L\log m)$

Computation Paths

- If space overhead for non-robust algorithm is $\log\frac{1}{\delta} = O(L \log m)$, this is $O\left(\frac{1}{\varepsilon}\log^2 m\right)$ space overhead for robust algorithm

- Some algorithms have space overhead $\log\log\frac{1}{\delta}$, which is only $O\left(\log\frac{1}{\varepsilon} + \log\log m\right)$ overhead!

“What’s an epsilon between friends?

- Statista: $\sim 300B$ e-mails sent per day
- Unsigned integer range: $n = 2^{32} \sim 4B$
- Accuracy: $\varepsilon = 0.01$
- Since $\frac{1}{\varepsilon} > \log n$, we should care about $\frac{1}{\varepsilon}$ factors!

Differential Privacy

- [DMNS06] Given $\varepsilon > 0$ and $\delta \in (0,1)$, a randomized algorithm $A: U^* \to Y$ is (ε, δ)-differentially private if, for every neighboring frequency vectors f and f' and for all $E \subseteq Y$,

$$\Pr[A(f) \in E] \leq e^{\varepsilon} \Pr[A(f') \in E] + \delta$$

$A($    $) \approx A($    $)$

Composition Theorems

- Basic composition: Suppose $\mathcal{M}_i$ is an $(\varepsilon_i, \delta_i)$-differentially private algorithm for $i \in [k]$. Then $(\mathcal{M}_1, \dots, \mathcal{M}_k)$ is $(\varepsilon_1 + \cdots + \varepsilon_k, \delta_1 + \cdots + \delta_k)$-differentially private

- Advanced composition: Suppose $\mathcal{M}_i$ is an $(\varepsilon_i, \delta_i)$-differentially private algorithm for $i \in [k]$. Then $(\mathcal{M}_1, \dots, \mathcal{M}_k)$ is $(\varepsilon', k\delta + \delta')$-differentially private, for

$$\varepsilon' = \sqrt{2k \ln(1/\delta')}\varepsilon + k\varepsilon(e^{\varepsilon} - 1)$$

Private Median

- Theorem: There exists an ε-DP algorithm that takes a dataset $S \subset X$ and outputs $x \in X$ such that with probability $1 - \delta$, the rank of x in S is $\frac{|S|}{2} \pm \Gamma$, where $\Gamma = O\left(\frac{1}{\varepsilon} \log \frac{|X|}{\delta}\right)$

DP for Adversarial Robustness

- Recall: sketch-switching needed $O\left(\frac{1}{\varepsilon}\log n\right)$ repetitions of the algorithm, one for each time the function increased by $(1+\varepsilon)$

- Run $\sqrt{\frac{1}{\varepsilon}}\cdot \mathrm{polylog}\left(n,\frac{1}{\varepsilon}\right)$ copies of the algorithm and output the private median of them each time the function increases by $(1+\varepsilon)$

$(1+\varepsilon)$-Robust Algorithms via DP [HassidimKaplanMansourMatiasStemmer20]

- Space $\tilde{O}\left(\frac{1}{\varepsilon^{2.5}}\log^4 n\right)$ algorithm for F_0
- Space $\tilde{O}\left(\frac{1}{\varepsilon^{2.5}}\log^4 n\right)$ algorithm for F_p with $p \in (0,2]$
- Space $\tilde{O}\left(\frac{1}{\varepsilon^{2.5}}n^{1-2/p}\right)$ algorithm for F_p with $p > 2$
- Space $\tilde{O}\left(\frac{1}{\varepsilon^{2.5}}\log^4 n\right)$ algorithm for L_2-heavy hitters

"$\frac{1}{\varepsilon}$ losses are not necessary"

A Curious Question…

- [Blanc23] showed that subsampling suffices for adaptive data analysis with the same rate, i.e., $\tilde{O}(k)$ samples suffice to handle k^2 adaptive queries

- To what extent can this replace differential privacy in adversarial robustness for the streaming model?

$(1+\varepsilon)$-Robust Algorithms via Difference Estimators [WoodruffZhou21]

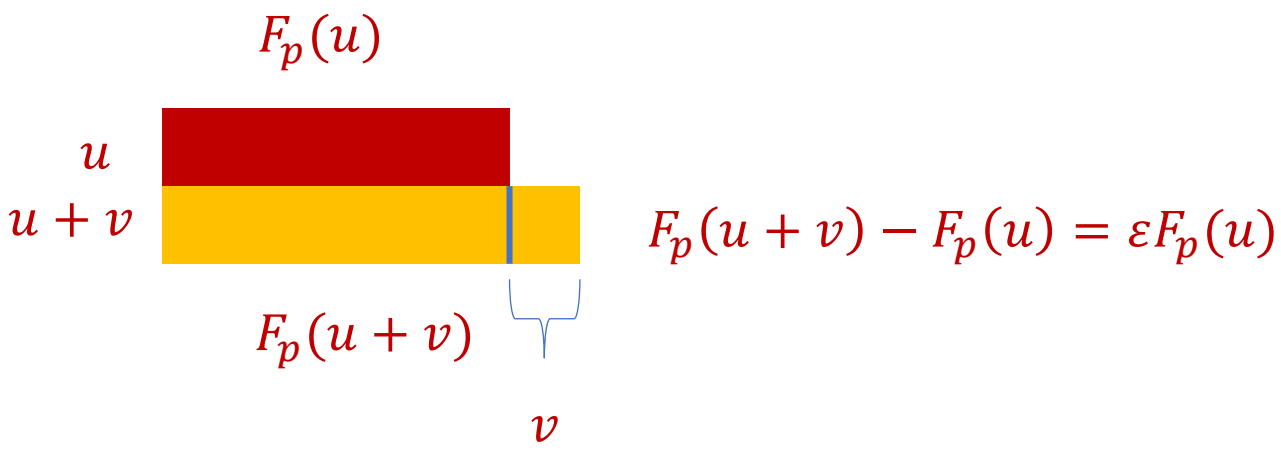
- Space $\tilde{O}\left(\frac{1}{\varepsilon^2}\log n\right)$ algorithm for F_0
- Space $\tilde{O}\left(\frac{1}{\varepsilon^2}\log n\right)$ algorithm for F_p with $p \in (0,2]$
- Space $\tilde{O}\left(\frac{1}{\varepsilon^2}n^{1-2/p}\right)$ algorithm for F_p with integer $p > 2$
- Space $\tilde{O}\left(\frac{1}{\varepsilon^2}\log n\right)$ algorithm for L_2-heavy hitters

"No losses* are necessary!"

Difference Estimators Intuition

- Do not need to pay $\frac{1}{\varepsilon^2}$ space each time F_p increases by $(1+\varepsilon)$?
- Only need constant factor approximation to $\varepsilon F_p(u)$
- Only need constant factor approximation to $F_p(u+v) - F_p(u)$

Use difference estimator



Summary

	[BJWY20]	[HKMMS20]	[WY21]
Distinct Elements	$\tilde{\mathcal{O}}\left(\frac{\log n}{\varepsilon^3}\right)$	$\tilde{\mathcal{O}}\left(\frac{\log^4 n}{\varepsilon^{2.5}}\right)$	$\tilde{\mathcal{O}}\left(\frac{\log n}{\varepsilon^2}\right)$
F_p Estimation, $p \in (0,2]$	$\tilde{\mathcal{O}}\left(\frac{\log n}{\varepsilon^3}\right)$	$\tilde{\mathcal{O}}\left(\frac{\log^4 n}{\varepsilon^{2.5}}\right)$	$\tilde{\mathcal{O}}\left(\frac{\log n}{\varepsilon^2}\right)$
Shannon Entropy	$\tilde{\mathcal{O}}\left(\frac{\log^6 n}{\varepsilon^5}\right)$	$\tilde{\mathcal{O}}\left(\frac{\log^4 n}{\varepsilon^{3.5}}\right)$	$\tilde{\mathcal{O}}\left(\frac{\log^3 n}{\varepsilon^2}\right)$
L_2-Heavy Hitters	$\tilde{\mathcal{O}}\left(\frac{\log n}{\varepsilon^3}\right)$	$\tilde{\mathcal{O}}\left(\frac{\log^4 n}{\varepsilon^{2.5}}\right)$	$\tilde{\mathcal{O}}\left(\frac{\log n}{\varepsilon^2}\right)$
F_p Estimation, integer $p > 2$	$\tilde{\mathcal{O}}\left(\frac{n^{1-2/p}}{\varepsilon^3}\right)$	$\tilde{\mathcal{O}}\left(\frac{n^{1-2/p}}{\varepsilon^{2.5}}\right)$	$\tilde{\mathcal{O}}\left(\frac{n^{1-2/p}}{\varepsilon^2}\right)$
F_p Estimation, $p \in (0,2]$, flip number λ	$\tilde{\mathcal{O}}\left(\frac{\lambda \log^2 n}{\varepsilon^2}\right)$	$\tilde{\mathcal{O}}\left(\frac{\log^3 n\sqrt{\lambda \log n}}{\varepsilon^2}\right)$	$\tilde{\mathcal{O}}\left(\frac{\lambda \log^2 n}{\varepsilon}\right)$

DP Strikes Back [AttiasCohenShechnerStemmer23]

- Differential privacy can be used with difference estimators for better bounds in the streaming model

- Parameterized in terms of the “twist number”, which is roughly how many times the difference vector has function value that is ε fraction of the prefix

Sampling

- Uniform sampling with a small overhead is robust for capturing the “density” of a dataset [Ben-EliezerYogev20]
- Importance sampling is robust with various overheads [BravermanHassidimMattiasSchainSilwalZhou22, JiangPengWeinstein23, Kenneth-MordochSapir25]
- Certain sampling algorithms are robust even to white-box adversaries

Separations

- Separation between non-adaptive and adaptive streaming model, for black-box adversaries in the insertion-only setting for problems of:
  - Streaming Adaptive Data Analysis Problem (SADA) [KaplanMansourNissimStemmer21]
  - Graph Coloring [ChakrabartiGhoshStoeckl22]

White-Box Adversaries

- The adversary has access not only to the previous outputs of the honest algorithm, but also to the *entire internal state* of the algorithm

- Reductions to deterministic communication complexity protocols [AjtaiBravermanJayramSilwalWoodruffZhou22]
- No non-trivial algorithms for many central problems
- Can use crypto against *time-bounded* white-box adversaries

Many Other Beautiful Works

- Attacks on classic data structures and algorithms:

[CohenLyuNelsonSarlosShechnerStemmer22], [CohenNelsonSarlosStemmer23], [CohenNelsonSarlosSinghalStemmer23], [AhmadianCohen24], [GribelyukLinWoodruffYuZhou24, 25]

- Adapting classic data structures:

[CherapanamjeriNelson20], [CohenLyuNelsonSarlosShechnerStemmer22], [NaorOved22],

Future Directions

- Capabilities and limitations for insertion-deletion streams?
- Refining adversary models and interactions, e.g., computationally restricted adversaries (bounded time, space, etc.), delayed observations, partial visibility of randomness
- Adversarial input in new problems and models, e.g., distributed monitoring, MPC

Any Questions?

Workshop Themes

- What attacks are possible against classical data structure and algorithms? How efficient can we make them?

  Xin, Mihir, Sara, Edith

- What are the capabilities and limitations of robust data structures and algorithms in various models? How do different adversarial models change what's possible?

  Omri, Amit, Guy, David, Elena, Honghao

- New problems and models of adversarial robustness

  Uri, Binghui, Sandeep, Chen, Shenghao, Yuhan