Improved Algorithms for White-Box Adversarial Streams

Ying Feng, David P. Woodruff
Carnegie Mellon University

White-box Adversarial Streaming Model

The sequence of stream updates is chosen adaptively by an adversary who sees the full internal state of the algorithm at all times, including the parameters and the previous randomness used by the algorithm [ABJ+22].

Modeled as a multi-round game between StreamAlg and Adversary:

① Adversary computes an update for the stream, which **depends on all previous stream updates, all previous internal states, and randomness** used by StreamAlg.

② StreamAlg acquires a fresh batch of random bits and uses it to update its data structures, and (if asked) outputs a response to the query.

③ Adversary observes the response, the internal state of StreamAlg, and the random bits.

The goal of Adversary is to make StreamAlg output an incorrect response to the query at some time throughout the stream.

Motivations and Applications

- **Database Coordination**: a central server may send internal state to the remote users to optimize the communication
  - Optimal Selection of Query Plans
  - Data Integration
  - OLAP
  - Data Warehousing
- **Adversarial Machine Learning**: protection against Perfect Knowledge Adversaries [BCM+13], i.e., attacks that use information of the trained model
- **Persistent Data Structures**: transparent version controls
- **Dynamic Algorithms**: adaptive adversaries with access to the internal randomness of the algorithm

Technical Ingredients

The **Short Integer Solution (SIS)** Problem:

- Setup: $n, m, p, \beta \in \mathbb{Z}$, $p \gg \beta$,
  A uniformly random matrix $A \in \mathbb{Z}_p^{n\times m}$
- Task: Find a non-zero $\boldsymbol{x} \in \mathbb{Z}_\beta^m$ so that $A\boldsymbol{x} = 0 \bmod p$.

Assumption. The SIS problem is hard against $2^{c\cdot n}$ - time adversaries, for $m, p, \beta \in \mathrm{poly}(n)$, $p \gg \beta$.

Lemma 1. (Under the SIS assumption,) given a uniformly random matrix $A \in \mathbb{Z}_p^{n\times m}$ for $m, p, \beta \in \mathrm{poly}(n)$, $p \gg \beta$.
If a vector $\boldsymbol{x} \in \mathbb{Z}_\beta^m$ is generated by an $O(2^{c\cdot n})$ - time adversary, then with probability at least 1-negl(n), there does not exist a k-sparse vector $\boldsymbol{y} \in \mathbb{Z}_\beta^m$, for which $\boldsymbol{x} \neq \boldsymbol{y} \bmod p$ yet $A\boldsymbol{x} = A\boldsymbol{y} \bmod p$, for $k \in O(n / \log n)$.

Main Problems

- **K-Sparse Vector Recovery with K-Sparsity Detection**:
  compute $f(x) = \begin{cases} x & \text{if } x \text{ is } k\text{-sparse} \\ \perp & \text{otherwise} \end{cases}$
- Extensions to matrix and tensor recovery
- Applications to numerical linear algebra and graph problems

Results

Problem	[ABJ+22]	Our Space
K-sparse recovery	–	$\tilde{O}(k)$
L_0-norm estimation	$\tilde{O}(n^{1-\varepsilon+c\varepsilon})$	$\tilde{O}(n^{1-\varepsilon})$
Low-rank matrix recovery	–	$\tilde{O}(nk)$
Low-rank tensor recovery	–	$\tilde{O}(k(n_1+\cdots+n_d))$
Robust PCA	–	$\tilde{O}(nk+r)$
Rank-decision	$\tilde{O}(nk^2)$	$\tilde{O}(nk)$
Maximum matching	–	$\tilde{O}(nk)$

(Suppressing $\log n$ factors)

Recovery Algorithm

We run two algorithms in parallel:

1. A **tester** algorithm to detect if the input is drawn from a small family of inputs, such as those which are sparse or low rank or both.
   - Sketch the data using a uniformly random matrix; by **Lemma 1.**, the probability of colliding with a sparse vector is low.
2. A time-efficient **recovery** algorithm, which only guarantees correct recovery if the input is indeed drawn from a small family.
   - A deterministic, thus adversarially robust algorithm. E.g. Iterative methods in Compressed Sensing

To respond to a query:

- Reconstructs a candidate of the data using the **recovery** sketch, which can be done quickly.
- Then uses the **tester** sketch to verify if this candidate matches the actual input (i.e., the input was from the small family), or this candidate is a garbage output.

Norm Estimation

L_0-Norm: Setting k = $n^{1-\varepsilon}$ in k-sparse recovery.

- Exact recovery when $||\boldsymbol{x}||_0 \leq n^{1-\varepsilon}$
- Otherwise $n^{1-\varepsilon} < ||\boldsymbol{x}||_0 \leq n$: estimate $||\boldsymbol{x}||_0 \approx n^{1-\varepsilon}$
  - n^{ε} - approximation

Future Work: Improve the approximation factor for the L_0 norm
L_p norms for p > 0,
as well as other statistics of vectors

References

[ABJ+22] Ajtai, M., Braverman, V., Jayram, T., Silwal, S., Sun, A., Woodruff, D. P., and Zhou, S. The white-box adversarial data stream model. PODS'22.

[BCM+13] Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. ECML PKDD'13.