Optimality of Linear Sketching under Modular Updates

Shachar Lovett (UCSD)

Kaave Hosseini (UCSD → CMU), Grigory Yaroslavtsev (Indiana)
Streaming and sketching
Streaming with binary updates

• Counters $x_1, \ldots, x_n \in \mathbb{F}_2$

• Stream of updates: $x_i \leftarrow x_i \oplus 1$

• At the end, want to compute function $f(x_1, \ldots, x_n)$

• For which functions can we do it using $\ll n$ memory?
Example

• Initially 000000
• Flip $x_1$ 100000
• Flip $x_5$ 100010
• Flip $x_2$ 110010
• Flip $x_5$ 100000
• ...

• Compute $f(x_1, \ldots, x_n)$
Linear sketching

• Linear sketching is a useful primitive for streaming

• Let \( f : \mathbb{F}_2^n \to \{0, 1\} \)

• \( f \) has a linear sketch of size \( k \) if it factors as \( f(x) = p(L(x)) \) where:
  (i) \( L : \mathbb{F}_2^n \to \mathbb{F}_2^k \) linear function
  (ii) \( p : \mathbb{F}_2^k \to \{0, 1\} \) post-processing function

• Equivalently, the “Fourier dimension” of \( f \) is \( k \)
Linear sketching implies streaming

• Assume $f: \mathbb{F}_2^n \rightarrow \{0,1\}$ factors as $f(x) = p(L(x))$ where
  (i) $L: \mathbb{F}_2^n \rightarrow \mathbb{F}_2^k$ linear function 
  (ii) $p: \mathbb{F}_2^k \rightarrow \{0,1\}$ post-processing function

• To compute $f$ in the streaming model, maintain $L(x) \in \mathbb{F}_2^k$
  • Easy to maintain under updates $x_i \leftarrow x_i \oplus 1$
  • Requires only $k$ bits of memory
Randomized linear sketching

• Randomization makes linear sketching more powerful

• \( f : \mathbb{F}_2^n \rightarrow \{0,1\} \) has a randomized linear sketch of size \( k \) if it can be approximated by a distribution over linear sketches of size \( k \)

• That is, if exists a distribution over \((L, p)\), where:

  (i) \( L : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^k \) linear function

  (ii) \( p : \mathbb{F}_2^k \rightarrow \{0,1\} \) post-processing function

Such that \( \Pr_{L,p}[f(x) = p(L(x))] \geq 1 - \epsilon \)
Randomized sketching gives additional power

• Consider the OR function: $OR(x_1, \ldots, x_n) = x_1 \lor \cdots \lor x_n$

• *Deterministic sketching* requires size $n$

• *Randomized sketching* can be done in size $O(\log(1/\epsilon))$
  (random parities)
Is linear sketching universal?

• Linear sketching seems like a very useful primitive for streaming
• Is it universal?

• That is: given a streaming algorithm that computes $f$ using $k$ bits of memory, can we extract from it a linear sketch for $f$ of size $\approx k$?
Universality of linear sketching
Universality of linear sketching

• Let $f : \mathbb{F}_2^n \to \{0,1\}$
• Assume: randomized streaming algorithm supporting $N$ updates and using $k$ bits of memory
• Goal: extract a randomized linear sketch of size $\approx k$

• True if $N \geq 2^{2^n}$ [Li-Nguyen-Wooldruff ‘14, Ai-Hu-Li-Woodruff ‘16]

• True if $N = \Omega(n)$ for random inputs [Kannan-Mossell-Sanyal-Yaroslavtsev ‘18]

• True if $N = \Omega(n^2)$ [This work]
Main theorem: streaming

• Let $f : \mathbb{F}_2^n \to \{0, 1\}$

• Assume there exists a randomized streaming algorithm for $f$ supporting $N = \Omega(n^2)$ updates which uses $k$ bits of memory

• Then there exists a randomized linear sketch for $f$ of size $O(k)$
Extensions (that I will not talk about)

• Extends to approximate real-valued functions $f: \mathbb{F}_2^n \rightarrow [0,1]$

• Extends to functions over other fields

• Assuming only $N = \Omega(n)$ updates are supported, we can still extract a randomized linear sketch, but its size will be $\text{poly}(k)$ instead of $O(k)$
One-way communication complexity
One way communication complexity

• Model a streaming algorithm as a one-way communication protocol

• Break $N$ updates into $M = N/n$ chunks of size $n$ each

• Setup: M players, holding inputs $x^1, ..., x^M \in \mathbb{F}_2^n$ ($x^i$ is the aggregate of the $n$ updates in the $i$-th chunk)

• Goal: compute $f(x^1 + \cdots + x^M)$

• Communication model: one-way
One way communication complexity

- M players, holding inputs $x^1, \ldots, x^M \in \mathbb{F}_2^n$
- Model: one-way communication with shared randomness
- Goal: $\text{output} = f(x^1 + \cdots + x^M)$ w.h.p over shared randomness

![Diagram of one-way communication with shared randomness]
Main theorem: one way communication

• Let \( f : \mathbb{F}_2^n \rightarrow \{0,1\} \)

• Assume there exists a one-way communication protocol for computing \( f(x^1 + \cdots + x^M) \) for \( M = \Omega(n) \) players with \( k \)-bit messages

(recall: this corresponds to \( N = Mn = \Omega(n^2) \) binary updates)

• Then there exists a randomized linear sketch for \( f \) of size \( O(k) \)

• For \( M = \Omega(1) \) players, get linear sketch of size \( \text{poly}(k) \)
Proof
Proof

• The proof uses
1. Standard techniques in communication complexity
2. Additive combinatorics
Proof step 1: Yao’s minimax principle

• Let \( f: \mathbb{F}_2^n \rightarrow \{0,1\} \)

• Fix a “hard distribution” \( \mu \) over inputs

• Goal: linear sketch for \( f(x) \) where \( x \sim \mu \)

• Embed hard distribution to the M players:
  • First M-1 players inputs \( x^1, \ldots, x^{M-1} \) are uniform in \( \mathbb{F}_2^n \)
  • Last player input \( x^M \) is set so that \( x^1 + \cdots + x^M = x \)

• Intuition: protocol has no information on \( x \) until the last player
Proof step 2: protocol structure

- Target: $x \sim \mu$
- Players inputs: $x^1, \ldots, x^{M-1} \in \mathbb{F}_2^n$ uniformly, $x^M = x^1 + \cdots + x^{M-1} + x$

- We may assume the protocol is deterministic
- Messages: $m_1(x^1), m_2(m_1, x^2), m_3(m_1, m_2, x^3), \ldots$
- Output: $\text{out}(m_1, \ldots, m_{M-1}, x^M)$

- With good probability $\text{out} = f(x^1 + \cdots + x^M) = f(x)$

- Can fix the messages (of the first $M-1$ players) to “typical messages”, without hurting the success probability too much
Proof step 3: fixing to typical messages

• Fix typical messages $m_1^*, m_2^*, \ldots, m_{M-1}^*$

• Corresponds to the first M-1 players inputs:
  • $A_1 = \{x^1 \in \mathbb{F}_2^n : m_1(x^1) = m_1^*\}$
  • $A_2 = \{x^2 \in \mathbb{F}_2^n : m_2(m_1^*, x^2) = m_2^*\}$
  • ...

• Sets are big: if the protocol uses k bits, then $|A_i| \geq 2^{n-k}$

• After conditioning on $x^1 \in A_1, \ldots, x^{M-1} \in A_{M-1}$, protocol output is a function of only $x^M = x^1 + \cdots + x^{M-1} + x$
Proof step 4: mixing

• Large sets $A^1, \ldots, A^{M-1} \subset \mathbb{F}_2^n$ of density $2^{-k}$

• If we sample $x^1 \in A_1, \ldots, x^{M-1} \in A_{M-1}$ and $x \sim \mu$, then with high probability
  \[
  \text{out}(x^1 + \cdots + x^{M-1} + x) = f(x)
  \]

• Technical lemma: for $M = \Omega(N)$, the sum $x^1 + \cdots + x^{M-1}$ mixes in $\mathbb{F}_2^n$

• More precisely, there exists a subspace $V \subset \mathbb{F}_2^n$ of co-dimension $O(k)$, such that the sum is near invariant to a random shift from $V$
Proof step 5: extracting linear sketch

• We found a large subspace $V$ of co-dimension $O(k)$

• If we sample $x^1 \in A_1, \ldots, x^{M-1} \in A_{M-1}, x \sim \mu$ and $v \in V$, then with high probability

\[
\text{out}(x^1 + \cdots + x^{M-1} + x + v) = f(x)
\]

• This allows to “factor out” $V$ from the output function, and extract a linear sketch for $f(x)$
Open problems
Linear sketching for modular updates

• For binary updates (or more general, modular updates), we prove that linear sketching is universal

• Any streaming algorithm which supports $N = \Omega(n^2)$ updates implies a randomized linear sketch with similar guarantees

• **Open problem 1:** can this be improved to $N = \Omega(n)$?

• [Kannan-Mossell-Sanyal-Yaroslavtsev ‘18] proved a partial result in this regime, giving a linear sketch for $f$ on random inputs

• Our results in this regime incur a polynomial loss in the sketch size
Integer updates

• Streaming if often considered in the integer case
• Integer counters $x_1, \ldots, x_n$
• Updates $x_i \pm 1$ or $x_i = 1$
• Sketching corresponds to linear functions over the integers

• The results of [Li-Nguyen-Wooldruff ‘14, Ai-Hu-Li-Woodruff ‘16] work in this regime as well, but require assuming $N \geq 2^{2^n}$

• **Open problem 2:** can our techniques be imported to this regime?
• Challenge: not clear what “mixing” should mean here
Thank you!