Submodular Streaming in All Its Glory:
Tight Approximation, Minimum Memory and
Low Adaptive Complexity

Ehsan Kazemi¹, Marko Mitrovic¹, Morteza Zadimoghaddam²,
Silvio Lattanzi², Amin Karbasi¹
¹Yale University and ²Google
Many practical scenarios we need to use streaming algorithms:

- the data arrives at a very fast pace
- there is only time to read the data once
- random access to the entire data is not possible and only a small fraction of the data can be loaded to the main memory

Video from “Britain’s Got Talent”

Summary
Many practical scenarios we need to use streaming algorithms:

- the data arrives at a very fast pace
- there is only time to read the data once

Is it possible to summarize a massive data set “on the fly”, i.e., when at any point of time we have access only to a small fraction of data?

Video from “Britain’s Got Talent”

Summary
Submodularity
Submodularity

\[ A = \{s_1, s_2\} \]

\[ B = \{s_1, s_2, s_3, s_4\} \]
Submodularity

$A = \{s_1, s_2\}$

$B = \{s_1, s_2, s_3, s_4\}$
Submodularity

\[ A = \{s_1, s_2\} \]

\[ B = \{s_1, s_2, s_3, s_4\} \]

\[ \forall A \subseteq B \subseteq V \text{ and } s \in V \setminus B \]
Submodularity

\[ A = \{s_1, s_2\} \]

\[ B = \{s_1, s_2, s_3, s_4\} \]

\[ \forall A \subseteq B \subseteq V \text{ and } s \in V \setminus B \]
Submodularity

\[ A = \{ s_1, s_2 \} \]
\[ B = \{ s_1, s_2, s_3, s_4 \} \]

Submodularity

\[ \forall A \subseteq B \subseteq V \text{ and } s \in V \setminus B \]

\[ f(A \cup \{ s \}) - f(A) \geq f(B \cup \{ s \}) - f(B) \]
A = \{s_1, s_2\}

B = \{s_1, s_2, s_3, s_4\}

\forall A \subseteq B \subseteq V \text{ and } s \in V \setminus B

f(A \cup \{s\}) - f(A) \geq f(B \cup \{s\}) - f(B)

\forall A \subseteq B \subseteq V

f(A) \leq f(B)
Choosing elements with marginal gain at least \( \tau^* = \frac{\text{OPT}}{2k} \) returns a set \( S \) with an objective value of at least \( f(S) \geq \frac{\text{OPT}}{2} \).

Can we find a good approximation of \( \text{OPT} \)?

- **Submodularity**: \( m \leq \text{OPT} \leq k \cdot m, m = \max_{e \in V} f(\{e\}) \)
- Find better lower bounds for \( \text{OPT} \) as more elements arrive

\[
\tau = (1 + \varepsilon)^i
\]

\[ m \quad \tau^* \quad k \cdot m \]

[Kazemi, Mitrovic, Zadimoghaddam, Lattanzi, Karbasi]

For monotone submodular functions, SIEVE-STREAMING++ with a \( O(k) \) memory gives constant factor approximation using only \( O(\log(k)/\varepsilon) \) function evaluation per element

\[
f(S_{\text{SIEVE-STREAMING++}}) \geq \frac{1}{2} \text{OPT}
\]

“Submodular Streaming in All Its Glory: Tight Approximation, Minimum Memory and Low Adaptive Complexity”, ICML’19
Streaming Algorithms: Cardinality Constraint

![Graph showing the relationship between Memory Complexity and Approximation Factor, with regions labeled Impossible and Hard, and a reference to Norouzi-Fard et al.]

- Memory Complexity: \( \Omega(n/k) \) to \( O(k) \)
- Approximation Factor: \( 1/2 \)
Streaming Algorithms: Cardinality Constraint

Memory Complexity

$\Omega(n/k)$

$O(k)$

Impossible

Hard

[Buchbinder et al., SODA'2015]

[Norouzi-Fard et al]

Approximation Factor

$\frac{1}{4}$  $\frac{1}{2}$
Streaming Algorithms: Cardinality Constraint

Memory Complexity

Approximation Factor

$\Omega(n/k)$

$O(k \log k)$

$O(k)$

Hard

[Norouzi-Fard et al]

Impossible

[Buchbinder et al., SODA'2015]

[Badanidiyuru et al., KDD'2014]
Streaming Algorithms: Cardinality Constraint

Approximation Factor

Memory Complexity

$\Omega(n/k)$

$O(k \log k)$

$O(k)$

Hard

Impossible

$\frac{1}{4}$

$\frac{1}{2} - \epsilon$

[Badanidiyuru et al., KDD’2014]

[Buchbinder et al., SODA’2015]

[Norouzi-Fard et al]

[Kazemi et al., ICML’2019]
BATCH-SIEVE-STREAMING++

Data Stream

For each threshold $\tau$

Filter

No

Yes

Sample set $T$ of size 4

Buffer

Full

$T$

$S_\tau$

For $\forall \tau$, let $f(T | S_\tau) \geq (1 - \varepsilon)\tau$
Multi-source Setting

Buffers $\mathcal{B}$

Distributed Sampling $T$

Central Machine

Threshold $\%$ Full

Total Size $B$

Characteristics:

- $\mathcal{B}$
- $T$

Mathematical Expression:

$$f(\{S_\tau\} \mid T) \geq (1 - \varepsilon) \tau |T|$$

Yale
Thank You!