Do Less, Get More: Streaming Submodular Maximization with Subsampling

Moran Feldman\textsuperscript{1} \quad Amin Karbasi\textsuperscript{2} \quad Ehsan Kazemi\textsuperscript{2}

\textsuperscript{1}Open University of Israel and \textsuperscript{2}Yale University
Data Summarization

- Large set of images
- Videos
- Sensor data
- fMRI parcellation
Submodularity

- Diminishing returns property for set functions.

\[ V = \{ \text{Rose, Sunset, Airplane, Sun, Vine, Flower, Sunflower} \} \]

\[ f \left( \{ \text{Rose} \} \right) - f \left( \{ \text{Rose} \} \right) \geq \]

\[ f \left( \{ \text{Rose, Airplane} \} \right) - f \left( \{ \text{Rose, Airplane} \} \right) \]

\[ \forall A \subseteq B \subseteq V \text{ and } x \notin B \]

\[ f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B) \]
Submodularity

- Diminishing returns property for set functions.

\[ V = \{ \text{rose, airplane, sunset, airplane, sunset, dandelion, sunflower} \} \]

\[ f \left( \{ \text{rose, airplane} \} \right) - f \left( \{ \text{rose} \} \right) \geq \]

\[ f \left( \{ \text{rose, airplane, sunset} \} \right) - f \left( \{ \text{rose, airplane} \} \right) \]

\[ \forall A \subseteq B \subseteq V \text{ and } x \not\in B \]

\[ f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B) \]
Streaming Algorithms

• 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
Streaming Algorithms

• 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

Key challenge:

Extract small, representative subset out of a massive stream of data

Surveillance camera
Constrained Non-Monotone Submodular Maximization

$$S^* = \arg \max_{S \in \mathcal{I}} f(S)$$

- **Set system**: a pair $\mathcal{N}, \mathcal{I}$, where $\mathcal{N}$ is the ground set and $\mathcal{I} \subseteq 2^\mathcal{N}$ is the set of independent sets

- **$p$-matchoid**: a set system $(\mathcal{N}, \mathcal{I})$ where there exist $m$ matroids $(\mathcal{N}_i, \mathcal{I}_i)$ such that every element of $\mathcal{N}$ appears in the ground set of at most $p$ matroids and

$$\mathcal{I} = \{ S \subseteq 2^\mathcal{N} \mid \forall 1 \leq i \leq m \ S \cap \mathcal{N}_i \in \mathcal{I}_i \}$$

[Chekuri et al., 2015]
The Sample-Streaming Algorithm

Data Stream

Keep with probability \( q = \frac{1}{p + \sqrt{p(p+1)+1}} \)

\( U_i \leftarrow \text{Exchange-Candidate}(S_{i-1}, u_i) \)

\[
\text{if } f(u_i \mid S_{i-1}) \geq (1 + c) \cdot f(U_i : S_{i-1}) \\text{ then } S_i \leftarrow S_{i-1} \setminus U_i + u_i.
\]
### Theorem 1: Non-monotone Submodular Maximization

- The **Sample-Streaming** algorithm provides a solution for the problem of maximizing a non-negative submodular function $f$ subject to a $p$-matchoid constraint with a $(2p + 2\sqrt{p(p + 1)} + 1)$-approximation guarantee.
- The space complexity of this algorithm is $O(k)$.
- The algorithm uses, in expectation, $O(km/p)$ value and independence oracle queries per each arriving element.

### Theorem 2: Monotone Submodular Maximization

- The **Sample-Streaming** algorithm provides a solution for the problem of maximizing a non-negative monotone submodular function $f$ subject to a $p$-matchoid constraint with a $4p$-approximation guarantee.
- The space complexity of this algorithm is $O(k)$.
- The algorithm uses, in expectation, $O(km/p)$ value and independence oracle queries per each arriving element.
### Theorem 1: Non-monotone Submodular Maximization

- The **Sample-Streaming** algorithm provides a solution for the problem of maximizing a **non-negative submodular function** $f$ subject to a $p$-matchoid constraint with a $(2p + 2\sqrt{p(p + 1)} + 1)$-approximation guarantee.
- The space complexity of this algorithm is $O(k)$.
- The algorithm uses, in expectation, $O(km/p)$ value and independence oracle queries per each arriving element.

### Theorem 2: Monotone Submodular Maximization

- The **Sample-Streaming** algorithm provides a solution for the problem of maximizing a **non-negative monotone submodular function** $f$ subject to a $p$-matchoid constraint with a $4p$-approximation guarantee.
- The space complexity of this algorithm is $O(k)$.
- The algorithm uses, in expectation, $O(km/p)$ value and independence oracle queries per each arriving element.
Constrained Submodular Maximization

**Theorem 1: Non-monotone Submodular Maximization**

- The **Sample-Streaming** algorithm provides a solution for the problem of maximizing a **non-negative submodular function** \( f \) subject to a \( p \)-matchoid constraint with a \( (2p + 2\sqrt{p(p+1)} + 1) \)-approximation guarantee.
- The space complexity of this algorithm is \( O(k) \).
- The algorithm uses, in expectation, \( O(km/p) \) value and independence oracle queries per each arriving element.

**Theorem 2: Monotone Submodular Maximization**

- The **Sample-Streaming** algorithm provides a solution for the problem of maximizing a **non-negative monotone submodular function** \( f \) subject to a \( p \)-matchoid constraint with a \( 4p \)-approximation guarantee.
- The space complexity of this algorithm is \( O(k) \).
- The algorithm uses, in expectation, \( O(km/p) \) value and independence oracle queries per each arriving element.
## Conclusion

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Function</th>
<th>Approx. Ratio</th>
<th>Memory</th>
<th>#Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chekuri et al., 2015</td>
<td>Monotone</td>
<td>$4p$</td>
<td>$O(k)$</td>
<td>$O(nkm)$</td>
</tr>
<tr>
<td>Chekuri et al., 2015 (R)</td>
<td>Non-monotone</td>
<td>$\frac{5p+2+1/p}{1-\varepsilon}$</td>
<td>$O\left(\frac{nk^2}{\varepsilon^2}\log\frac{k}{\varepsilon}\right)$</td>
<td>$O\left(\frac{nk^2m}{\varepsilon^2}\log\frac{k}{\varepsilon}\right)$</td>
</tr>
<tr>
<td>Chekuri et al., 2015</td>
<td>Non-monotone</td>
<td>$\frac{9p+O(\sqrt{p})}{1-\varepsilon}$</td>
<td>$O\left(\frac{k}{\varepsilon}\log\frac{k}{\varepsilon}\right)$</td>
<td>$O\left(\frac{nk^m}{\varepsilon}\log\frac{k}{\varepsilon}\right)$</td>
</tr>
<tr>
<td>LOCAL-SEARCH</td>
<td>Non-monotone</td>
<td>$4p + 4\sqrt{p} + 1$</td>
<td>$O(k\sqrt{p})$</td>
<td>$O(n\sqrt{pk}m)$</td>
</tr>
<tr>
<td>Sample-Streaming (R)</td>
<td>Monotone</td>
<td>$4p$</td>
<td>$O(k)$</td>
<td>$O(nkm/p)$</td>
</tr>
<tr>
<td>Sample-Streaming (R)</td>
<td>Non-monotone</td>
<td>$4p + 2 - o(1)$</td>
<td>$O(k)$</td>
<td>$O(nkm/p)$</td>
</tr>
</tbody>
</table>

- Our algorithm provides the best of three worlds:
  - the **tightest approximation guarantees** in various settings
  - **minimum memory** requirement
  - **fewest queries** per element

**Poster:** Today (Thu Dec 6th) 10:45 AM-12:45 PM @ Room 210 & 230 AB #75