**MOTIVATION**

Are submodular functions in your backyard?

**THE COMPUTING MODEL**

We only require black-box access to the submodular function and the ability to store data set elements.

Formally, we assume the following:
- Elements from a ground set $T$ is streamed in to the algorithm, with order according to a uniformly random permutation.
- The algorithm chooses to store or throw away the element the moment it is seen.
- Only $O(k)$ elements can be stored.

**OUR CONTRIBUTIONS**

Our algorithms take exponentially less memory than current state-of-the-art while achieving better approximations in practice.

Our algorithm on real-world data. “Lazy greedy” is the best known offline solution.

Submodularity: $f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$ when $S \subseteq T$.

Monotone: $f(S \cup \{e\}) \geq f(S)$

**APPLICATION OUTLINE**

Our algorithm relies on two ingredients: a solution cascade and a random window partitioning.

**REQUIREMENTS OF MODERN SYSTEMS**

Modern data often comes in the form of a stream.

Characteristics of typical systems:
- Memory is limited
- Random access is not possible
- Data is sampled from some underlying distribution

Classical algorithms require storing the entire dataset. Can we do better?

Submodularity: $f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$ when $S \subseteq T$.

Monotone: $f(S \cup \{e\}) \geq f(S)$

Typical application: max $f(S)$ for a given $k$.

- “cardinality constrained submodular maximization”

---

Authors | Memory | Approx. | Notes |
--- | --- | --- | --- |
Sieve streaming, Badanidiyuru et al. '14 | $O\left(\frac{k \log k}{\epsilon}\right)$ | $\frac{1}{2} - \epsilon$ | Adversarial order |
Salsa, Norouzi-Fard et al. '18 | $O(k \log k)$ | $\frac{1}{2} + c$, $0 < c \leq 10^{-13}$ | Random order, also gives $1/2$ l.b. for adversarial order |
Agrawal et al. '19 | $O(k \exp(poly(1/\epsilon)))$ | $1 - \frac{1}{e} - \epsilon$ | Optimal approx.; constant of $\geq 2^{10^6}$ for $\epsilon < 0.2$ |
Our paper | $O\left(\frac{k}{\epsilon}\right)$ | $1 - \frac{1}{e} - \epsilon$ | Works in practice; also $\frac{1}{2}$ for non-monotone |

---

- Add $e$ (and replace $L_i$) if $f(L_i \cup \{e\}) \geq f(L_{i+1})$.
- A pyramid of solutions $\{L_1, L_2, \ldots, L_k\}$ with $|L_i| = i$.
- Greedily add one element to each level per window, choosing from window + elements added in previous windows.