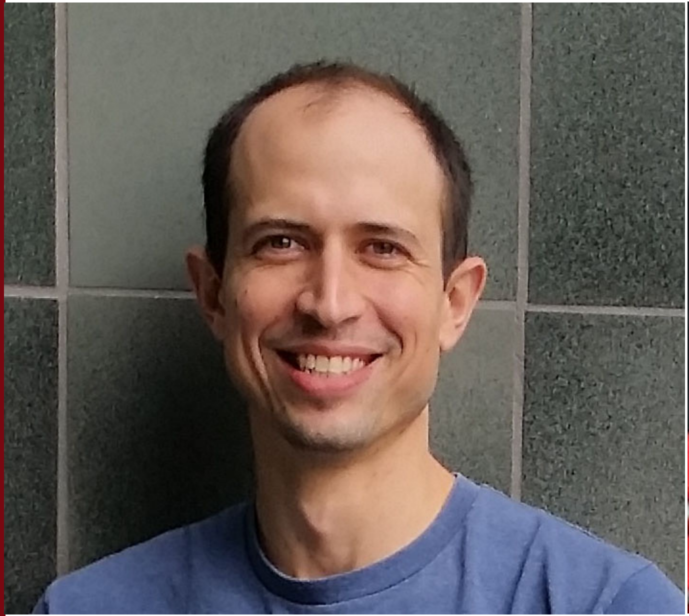
Stanford
Computer Science

Submodular maximization for random streams

Paul Liu, Aviad Rubinstein,
Jan Vondrák, Junyao Zhao

NeurIPS 2021



Aviad Rubinstein

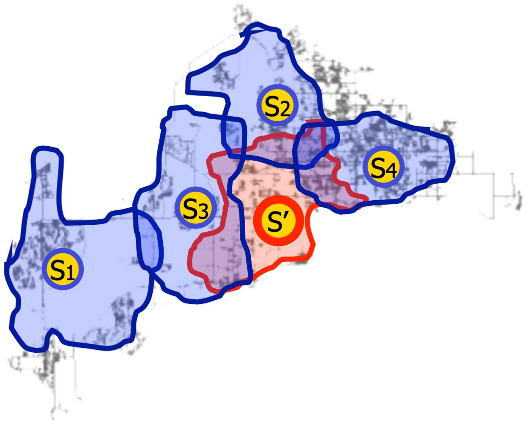
Jan Vondrák

Junyao Zhao

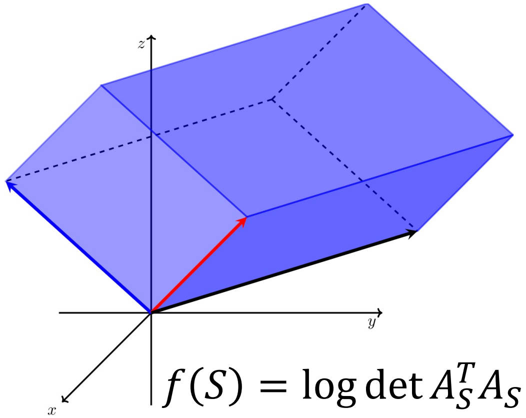
Submodular functions

- A function f is *submodular* if
$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$$
when $S \subseteq T$.
- ..and *monotone* if
$$f(S) \leq f(T) \text{ when } S \subseteq T.$$
- Captures the notion of *convexity* and *diminishing returns* for discrete functions

McDonald's Deals



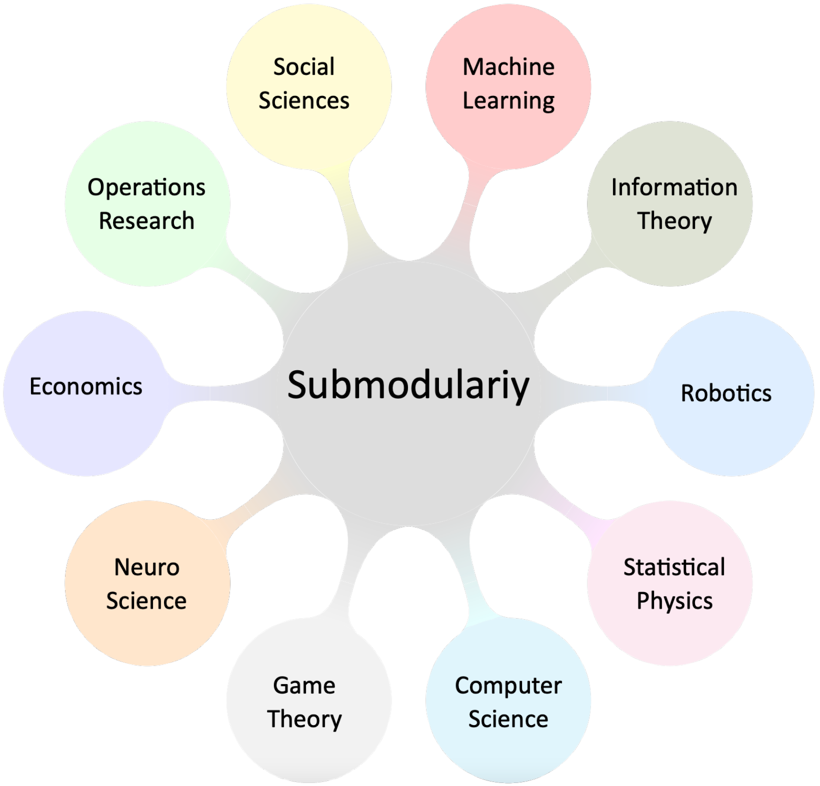
Sensor placement



Determinantal Point Processes

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

Submodularity in Everyone's Backyard

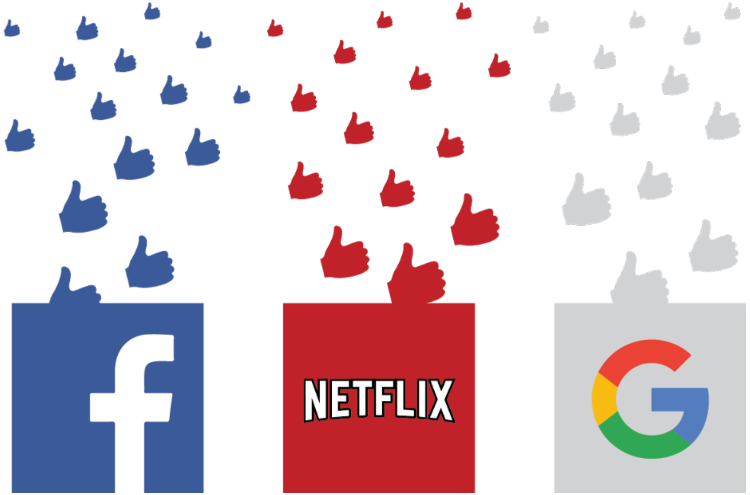


Slide courtesy of the 2020 ICML tutorial on submodularity.

Submod. Optimization on Modern Datasets

- **Our goal:** cardinality constrained submodular maximization on *modern datasets*

  $\max_{|S|\leq k} f(S)$ for submodular f



- Offline setting: NP-Hard
  - Can approximate within $1 - 1/e$ for monotone f, $\approx 1/e$ for non-monotone

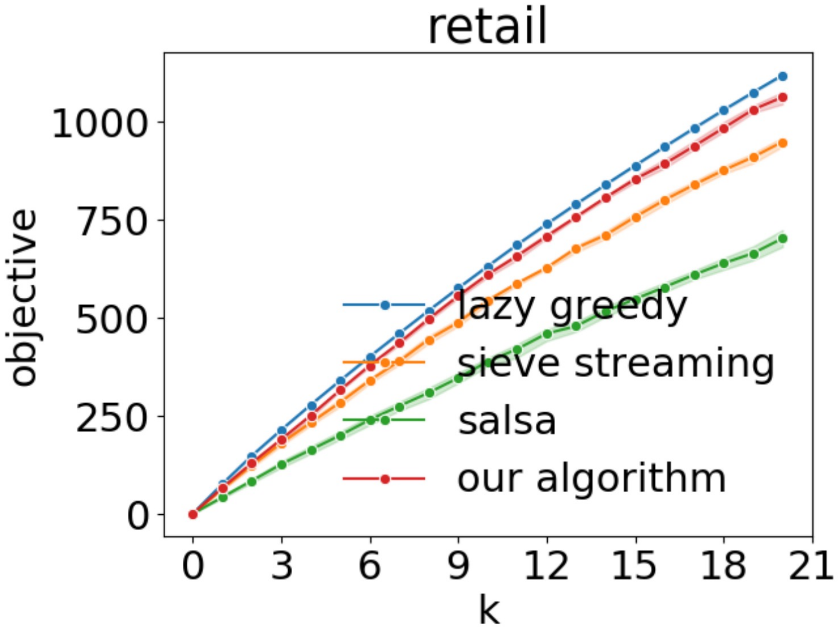
- Modern data often comes in the form of a *stream* ← Our focus

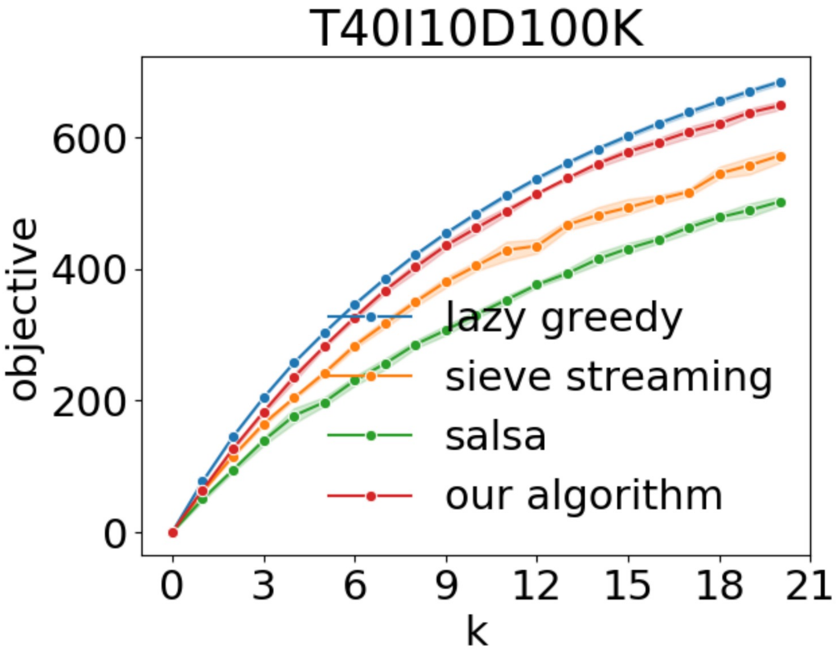
Streaming Submodular Optimization

- Memory is limited
  - $\tilde{O}(k)$ memory (elements stored)
- Random access is not possible
  - Algorithm chooses to store or ignore an element the moment that it is seen
- Assume a *random ordering* of the stream
  - A generalization of drawing i.i.d. data

Prior Work

Authors	Memory	Approximation	Notes
Badanidiyuru et al. '14	$O\left(\frac{k \log k}{\epsilon}\right)$	$\frac{1}{2} - \epsilon$	Adversarial order
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	$O(k\exp(\mathrm{poly}(1/\epsilon)))$	$1 - \frac{1}{e} - \epsilon$	Optimal approx.; constant of $> 2^{100}$ for $\epsilon < 0.2$.
Our paper	$O(k/\epsilon)$	$1 - \frac{1}{e} - \epsilon$	Works in practice; also $\frac{1}{e}$ for non-mono.

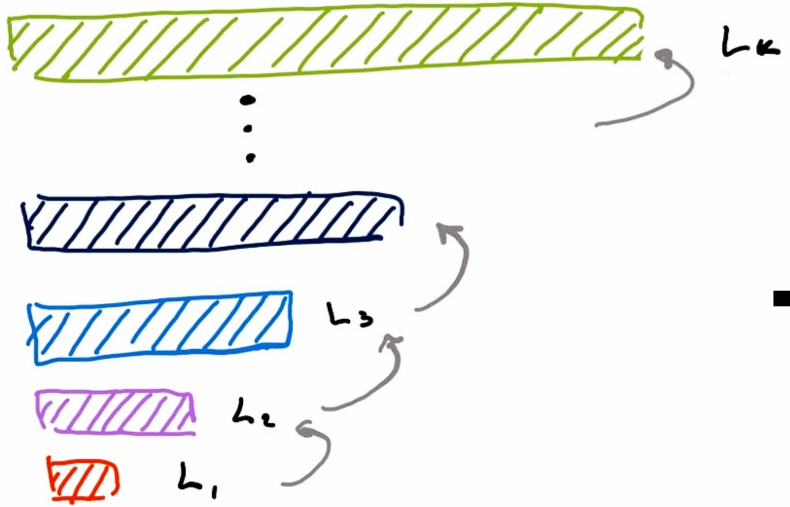




Our algorithm vs prior work. Salsa is the $\frac{1}{2} + c$ approximation by Norouzi-Fard et al.

Lazy greedy is an optimal ***offline*** algorithm.

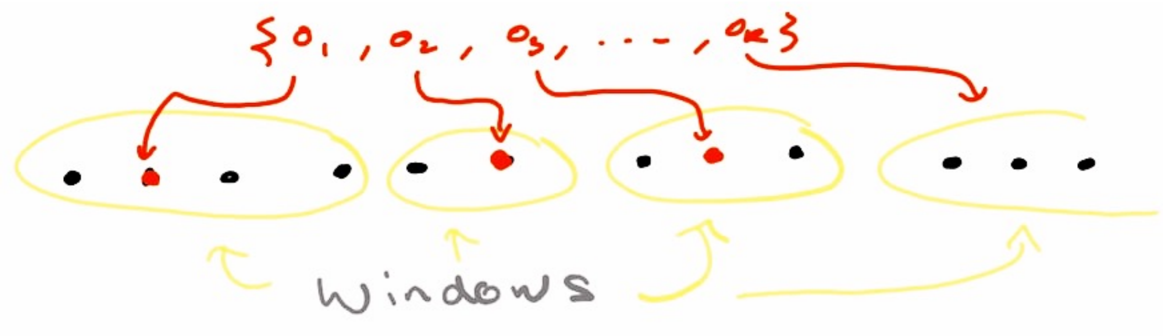
Our algorithm



Solution Cascade + Random window partitioning

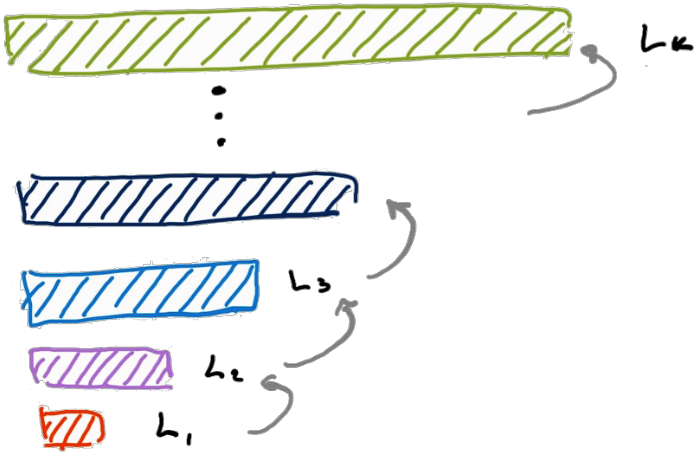
Random window partitioning

- Pick a random partition of stream E into $m = O(k/\epsilon)$ windows
  - i.e. window sizes have $\frac{|E|}{m}$ elements on average and sum to $|E|$
  - Windows with optimal elements are called *active windows*
  - Choice of m ensures $(1-\epsilon)k$ active windows



Solution Cascade

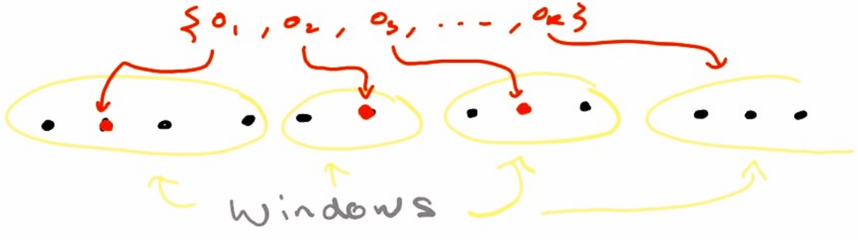
- A pyramid of solutions $\{L_1, L_2, \ldots, L_k\}$ where $|L_i| = i$.
- Greedily add one element to each level per window, choosing from window + H (initially empty).
- At the end of each window, the elements of the pyramid are added to H.

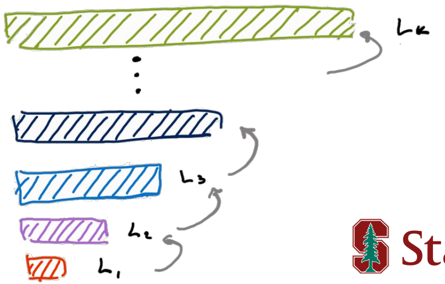


Add $\boldsymbol{e}$ (and replace L_{i+1}) if $f(L_i \cup \{e\}) \geq f(L_{i+1})$.

Intuition

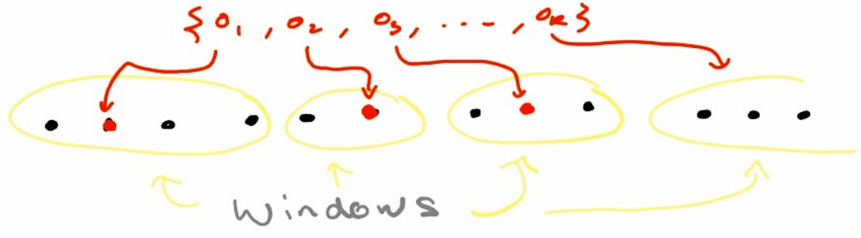
- Good solutions percolate up the cascade, until it reaches the top
- On each *active window*, elements at least as good as an average optimal element are added
  - On i-th active window, we get $\frac{1}{k}(f(O) - f(L_i))$.
  - After k active windows, $f(L_k) \geq 1 - \left(1 - \frac{1}{k}\right)^k \approx 1 - 1/e$.

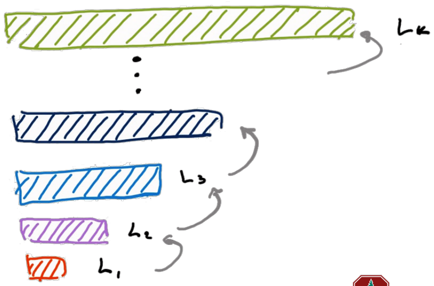




Non-monotone algorithm

- The same algorithm essentially works for non-monotone submodular functions
- However, a subsampling of the elements is needed for theoretical guarantees
- $1/e$-approximation in $O(k/\epsilon)$ memory.





Open questions / Future work

- Extension of our framework to more general constraints
  - E.g. what is the best algorithm for matroid constraints? (Possible to do slightly worse than 1/2)
  - Alternatively, can we prove lower bounds?

- Multi-pass algorithms for various constraints (upcoming paper!)

Stanford
Computer Science

Thanks for listening!