Very Fast Streaming Submodular Function Maximization

Sebastian Buschjäger, Jan-Philipp Honysz, Lukas Pfahler and Katharina Morik

ECML PKDD 2021
VIRTUAL
13-17 September

Submodular Function Maximization arises in many different applications fields in Machine Learning and Data Science, e.g.

- *Selecting the most informative items from a collection*
- *Maximizing the coverage of objects in an area*
- *Estimating the parameters of Determinantal Point Processes*

Submodular Function Maximization

$$\max_{S \subseteq V, |S| \leq K} f(S)$$

where $f: 2^V \to \mathbb{R}_+$ is a set-function and K is a cardinality constraint

Gain of item e: $\Delta(e \mid S) = f(S \cup \{e\}) - f(S)$

Submodularity: $\Delta(e \mid A) \geq \Delta(e \mid B)$ for $A \subseteq B \subseteq V$

Maximum singleton: $m = \max \{ f(\{e\}) \mid e \in V \}$

Streaming Submodular Function Maximization

The groundset V is often large. Therefore, a line of research studies streaming maximization algorithms which consume one item at a time. By submodularity we can estimate the optimal function value as

$$m \leq f(S) \leq K m$$

Thus, many streaming algorithms compute the gain of an element and add it to S once it exceeds a given novelty threshold depending on an estimation of the true f(S) from [m, Km]. However, the optimal threshold is unknown beforehand so that multiple thresholds must be used in parallel for sufficient performance.

The ThreeSieves Algorithm

Using more thresholds leads to a better maximization performance, but also requires more memory and computations. Our core idea is to maintain a single, carefully calibrated threshold which leads to similar performance while using fewer resources. To do so, we start with a very large threshold (e.g. assuming f (S) = K m) and gradually decrease it once there is enough evidence that no future item in the data stream will 'out-value' the current threshold.

The Rule Of Three

We estimate the probability p (e | S, f, v) that the gain of e exceeds the threshold v given the current summary S. There are two cases:

1) *If e is not added to S, update p given the negative outcome*
2) *If e is added to S, then S changed. Re-start the estimation of p*

Note that p is estimated from data and hence comes with its own confidence interval. The Rule of Three states that the α=0.95 confidence interval after T negative tries is

$$0 \leq p \leq 3/T$$

E.g.: After T = 1000 rejections $p \leq 0.003$ with 95% confidence

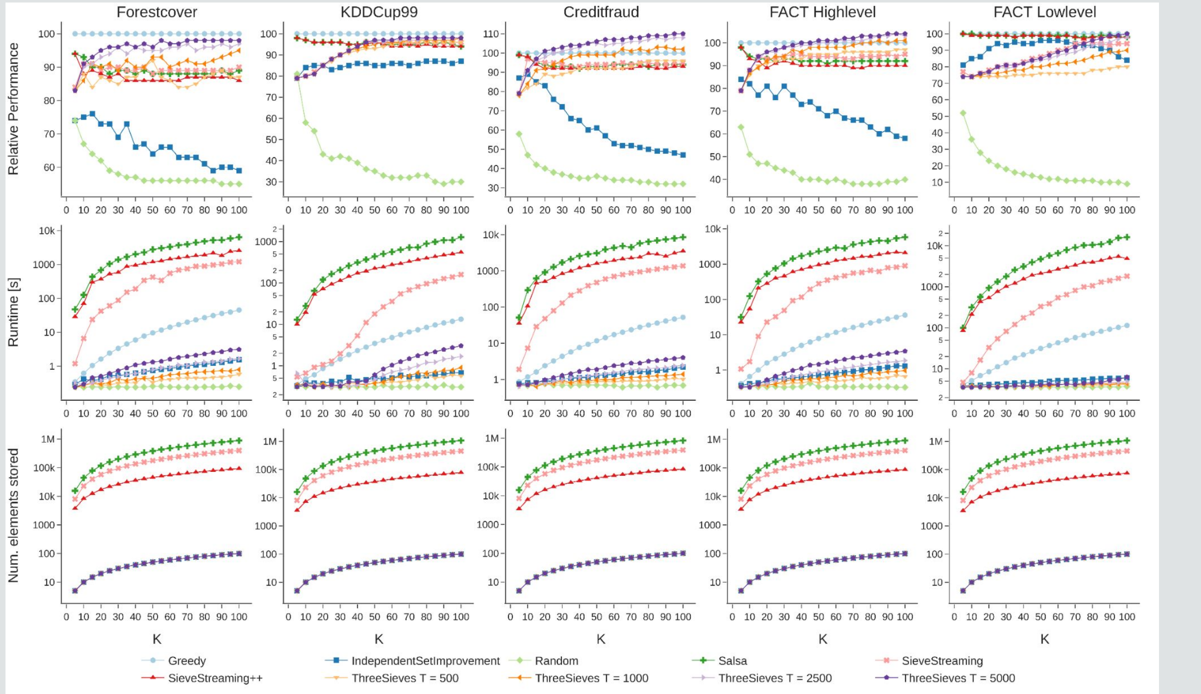
Algorithmic Idea

- *Start with largest available threshold and set t = 0*
- *If the gain exceeds the threshold, add e to S and set t = 0*
- *If the gain does not exceed the threshold, increase t by one*
- *If $t \geq T$, lower threshold and set t = 0*

Results

Algorithm	Approximation Ratio	Memory	Queries per Element	Stream	Ref.
Greedy	$1 - 1/\exp(1)$	$\mathcal{O}(K)$	$\mathcal{O}(1)$	✗	[23]
StreamGreedy	$1/2 - \varepsilon$	$\mathcal{O}(K)$	$\mathcal{O}(K)$	✗	[13]
PreemptionStreaming	$1/4$	$\mathcal{O}(K)$	$\mathcal{O}(K)$	✓	[4]
IndependentSetImprovement	$1/4$	$\mathcal{O}(K)$	$\mathcal{O}(1)$	✓	[8]
Sieve-Streaming	$1/2 - \varepsilon$	$\mathcal{O}(K \log K/\varepsilon)$	$\mathcal{O}(\log K/\varepsilon)$	✓	[2]
Sieve-Streaming++	$1/2 - \varepsilon$	$\mathcal{O}(K/\varepsilon)$	$\mathcal{O}(\log K/\varepsilon)$	✓	[16]
Salsa	$1/2 - \varepsilon$	$\mathcal{O}(K \log K/\varepsilon)$	$\mathcal{O}(\log K/\varepsilon)$	(✓)	[24]
QuickStream	$1/(4c) - \varepsilon$	$\mathcal{O}(cK \log K \log(1/\varepsilon))$	$\mathcal{O}(\lceil 1/c \rceil + c)$	✓	[18]
ThreeSieves	$(1-\varepsilon)(1 - 1/\exp(1))$ with prob. $(1-\alpha)^K$	$\mathcal{O}(K)$	$\mathcal{O}(1)$	✓	this paper

ThreeSieves: Smaller resource consumption with better approximation-ratio to existing work. However, the approximation-ratio now holds with high probability $(1-\alpha)^K$



ThreeSieves: Speed and memory is comparable with a random selection. However, the performance is comparable (or sometimes better) than existing work.

Sebastian Buschjäger, Artificial Intelligence Group, TU Dortmund University, Germany (sebastian.buschjaeger@tu-dortmund.de)

Submodular Streaming Maximization https://github.com/sbuschjaeger/SubmodularStreamingMaximization