Adversarially Robust Submodular Maximization under Knapsack Constraints

Dmitry Avdyukhin
Slobodan Mitrović
Grigory Yaroslavtsev
Samson Zhou
Submodular Functions

- Ground Set $V$ (items, sets, vertices)
- Set function $f : 2^V \rightarrow \mathbb{R}$ with diminishing returns property

Oracle access to $f$. Given a subset $S \subseteq V$ returns $f(S)$.

\[
\forall A \subseteq B \subseteq V, e \notin B \\
f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)
\]
Submodular Functions

\[ V = \{ \}

\[ f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B) \]
Applications of Submodular Functions

- Viral marketing [Kempe et al., 2003]
- Feature selection [Krause & Guestrin, 2005]
- Clustering [Narasimhan & Bilmes, 2005]
- Search result diversification [Agrawal et al., 2009]
- Recommender systems [El-Arini & Guestrin, 2011]
- Active learning [Golovin & Krause, 2011]
- Document summarization [Lin & Bilmes, 2011]
- Data subset selection [Wei, Iyer & Bilmes, 2015]
- etc
Clustering
Clustering

\[ C(S) = \frac{1}{|V|} \sum_{e \in V} \min_{v \in S} d(e, v) \]

\[ f(S) = C(\{e_0\}) - C(S \cup \{e_0\}) \]
Coverage

- $E = \{e_1, e_2, ..., e_n\}$
- $V \subseteq 2^E$
- $S = \{s_{i_1}, s_{i_2}, ..., s_{i_k}\} \in V$
- $f(S) = |\bigcup_{s_i \in S} s_i|$

$f$ is a submodular function

$S^* = \arg \max_{|S| \leq k} f(S)$
Viral Marketing

- $E = \{p_1, p_2, \ldots, p_n\}$
- $V \subseteq 2^E$

$$S^* = \arg \max_{c(S) \leq K} f(S)$$

$$S = \{s_{i_1}, s_{i_2}, \ldots, s_{i_K}\} \in V$$

$$c(s_i) \geq 0$$
First Objective

Submodular maximization under cardinality constraint

\( f \) is submodular, monotone, and \( f(\emptyset) = 0 \)

Extract *small* representative subset out of a big dataset

\[
S^* = \arg \max_{|S| \leq k} f(S)
\]

Solving this problem exactly is NP-hard
**Greedy** [Nemhauser, Wolsey, Fisher, ’78]

\[ V = \{ \text{food items} \} \]

**Goal:** Find \( S^* = \arg \max_{|S| \leq k} f(S) \)

*Marginal gain:* 

\[ f(\text{item}) \]

\[ f(\text{food}) \]

\[ f(\text{drink}) \]
Greedy [Nemhauser, Wolsey, Fisher, ‘78]

\[ V = \{ \} \]

Goal: Find \( S^* = \arg \max_{|S| \leq k} f(S) \)

\[ f(S) \geq (1 - 1/e)OPT \]
Algorithms performed sequentially.
Traditional

Algorithms performed sequentially.

Modern

distributed
Thresholding [Badanidiyuru, Mirzasoleiman, Karbasi, Krause, ‘14]

Stream:

\[ f \left( \frac{\text{OPT}}{2k} \right) \geq \ ? \]
Thresholding [Badanidiyuru, Mirzasoleiman, Karbasi, Krause, ‘14]

Stream:

\[ f(\text{Stream}) \geq \frac{\text{OPT}}{2k} \]
Thresholding [Badanidiyuru, Mirzasoleiman, Karbasi, Krause, ‘14]

Stream:

\[ f(\text{Stream}) \geq \frac{\text{OPT}}{2k} \]
Thresholding [Badanidiyuru, Mirzasoleiman, Karbasi, Krause, ‘14]

Stream:

\[ f \geq \frac{\text{OPT}}{2k} \]
Thresholding [Badanidiyuru, Mirzasoleiman, Karbasi, Krause, ‘14]

Stream:

\[ OPT \in \{ (1 + \epsilon)^i | i \in \mathbb{N} \} \]

\[ f(\vdots) \geq \frac{1}{2} \OPT \]

\[ \geq \frac{\OPT}{2k} \]
Second Objective

Submodular maximization under knapsack constraint

\( f \) is submodular, monotone, and \( f(\emptyset) = 0 \)

Extract **small** representative subset out of a big dataset

\[
S^* = \arg \max_{c(S) \leq K} f(S)
\]

Solving this problem exactly is NP-hard
Thresholding in Review

❖ Key concept: marginal gain $f(e | S) = f(e \cup S) - f(S)$
❖ If marginal gain exceeds threshold, add item to $S$.
❖ Else, discard item.

❖ $f(\text{OPT} \cup S) \geq f(\text{OPT}), \ |\text{OPT} \cup S| \leq 2k$

❖ What about for knapsack constraints?
❖ Could have item with good marginal gain, but really large size
Knapsack Optimization

Stream:
Knapsack Optimization

- Key concept: marginal density $\rho(e \mid S) = \frac{f(e \cup S) - f(S)}{c(e)}$
- If marginal density exceeds threshold, add item to $S$.
- Else, discard density. Does it work?
Knapsack Optimization

- **ALG 1:**
  - If marginal density exceeds threshold, add item to $S$.
  - Else, discard density.

- **ALG 2:**
  - Keep “best” element

- **ALG:** Return max(ALG 1, ALG 2)
Adversarial Robust Submodular Optimization
Adversarial Robust Submodular Optimization
Adversarial Robust Submodular Optimization
Adversarial Robust Submodular Optimization
Adversarial Robust Submodular Optimization
Adversarial Robust Submodular Optimization
Adversarial Robust Submodular Optimization

- $V = \{e_1, e_2, \ldots, e_n\}$
- Monotone submodular function $f$
- $c(e_i) \geq 0$

- See all the data and make summary $Z$.
- Given set $E$ that is removed from $V$.

- **Goal:** $S^* = \arg \max_{c(S) \leq K, S \cap E = \emptyset} f(S)$
Results

❖ Streaming algorithm for single knapsack, robust to the removal of $m$ items.
❖ Better streaming algorithm for single knapsack, robust to the removal of size $M$.
❖ Streaming algorithm for multiple knapsack, robust to the removal of $m$ items.
❖ Distributed algorithm for multiple knapsack, robust to the removal of $m$ items.
❖ Size of our summaries are almost optimal.
Approach

Stream:

- Algorithm to produce summary $S$
- $Z = S \setminus E$
- Run Greedy on $Z$
Partitions and Buckets Data Structure [Bogunovic, Mitrovic, Scarlett, and Cevher ‘17]

Partition 0
Partition 1
Partition 2
Partition 3

\( \tau \)

\( \tau / 2 \)

\( \tau / 4 \)

\( \tau / 8 \)
Partitions and Buckets Data Structure [Bogunovic, Mitrovic, Scarlett, and Cevher ‘17]

Partition 3

Partition 2

Partition 1

Partition 0

\[ \tau \]

\[ \tau /2 \]

\[ \tau /4 \]

\[ \tau /8 \]

\[ \geq ? \tau \]
Partitions and Buckets Data Structure [Bogunovic, Mitrovic, Scarlett, and Cevher ‘17]

Partition 4

Partition 3

Partition 2

Partition 1

Partition 0

\( \tau \)

\( \tau /2 \)

\( \tau /4 \)

\( \geq \tau /2 \)

\( \tau /8 \)
Partitions and Buckets Data Structure [Bogunovic, Mitrovic, Scarlett, and Cevher ‘17]

Partition 3

Partition 2

Partition 1

Partition 0

\[ \tau \]

\[ \tau /2 \]

\[ \geq \tau /4 \]

\[ \tau /4 \]

\[ \tau /8 \]
Partitions and Buckets Data Structure [Bogunovic, Mitrovic, Scarlett, and Cevher ‘17]

- Items in high partitions are more valuable
- Bad approximation if they are deleted, so we need more buckets

- Items in low partitions are not as valuable
- Still have good approximation if many buckets are full

- If many items are deleted from high partitions, but buckets in low partitions are not full, must still have captured “good” items
Towards Knapsack Constraints

- Initial idea: replace marginal gain with marginal density
- Problem: big items can’t fit
- Hotfix: double the size of each bucket
Towards Knapsack Constraints

- Problem: number of buckets is based on the threshold, not size

Partition 0

Remains good approximation

Partition 0

Bad approximation!
Towards Knapsack Constraints

❖ Problem: number of buckets is based on the threshold, not size
❖ Main idea: *Dynamic* bucketing scheme

❖ Each time element is added, allocate space proportional to its size

❖ Cap total number of items
Towards Knapsack Constraints

- Items in high partitions are more valuable
- Bad approximation if they are deleted, so we need more buckets

- Items in low partitions are not as valuable unless they are large
- Large items allocate more buckets
- Still have good approximation if many buckets are full

- If many items are deleted from high partitions, but buckets in low partitions are not full, must still have captured “good” items
ARMSM: Multiple Knapsacks

- $V = \{e_1, e_2, \ldots, e_n\}$
- Monotone submodular function $f$
- $c_1(e_i) \geq 0, c_2(e_i) \geq 0, \ldots, c_d(e_i) \geq 0$

- See all the data and make summary $Z$.
- Given set $E$ that is removed from $V$.

- **Goal:** $S^* = \text{arg} \max_{c_1(S) \leq b_1, \ldots, c_d(S) \leq b_d, S \cap E = \emptyset} f(S)$
Normalization

- Rescale each row $i$ in cost matrix by $b_1/b_i$ so that all knapsack constraints are $K := b_1$.
- Rescale all entries in cost matrix and constraint vector by minimum entry so that all costs are at least 1.
ARMSM: Multiple Knapsacks

- Recall algorithm: partitions and buckets, add item if marginal density exceeds threshold.
- What is the marginal density here?

- Marginal gain divided by the largest cost (across all knapsacks).
- Lose a factor of $\sim \frac{1}{2d}$ in the approximation guarantee.
Distributed Algorithm

- Send partition and buckets data structure and data across multiple machines
- With high probability, “bad” cases will be split across multiple machines
Results

- First constant factor approximation algorithms for submodular maximization robust to a number of removals

<table>
<thead>
<tr>
<th>Model</th>
<th>Removal</th>
<th>Approximation</th>
<th>Constraint</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streaming</td>
<td>$m$ items</td>
<td>$O(1)$</td>
<td>Single knapsack</td>
<td>Nearly optimal summary size</td>
</tr>
<tr>
<td>Streaming</td>
<td>$M$ space</td>
<td>$O(1)$</td>
<td>Single knapsack</td>
<td>Better guarantees, nearly optimal summary size and algorithm space</td>
</tr>
<tr>
<td>Streaming</td>
<td>$m$ items</td>
<td>$O\left(\frac{1}{d}\right)$</td>
<td>$d$ knapsacks</td>
<td>Nearly optimal summary size</td>
</tr>
<tr>
<td>Distributed</td>
<td>$m$ items</td>
<td>$O\left(\frac{1}{d}\right)$</td>
<td>$d$ knapsacks</td>
<td>2 rounds of communication</td>
</tr>
</tbody>
</table>
Empirical Evaluations

- Social network graphs from Facebook (4K vertices, 81K edges) and Twitter (88K vertices, 1.8M edges) collected by the Stanford Network Analysis Project (SNAP).
- Dominating set

- MovieLens (27K movies, 200K ratings)
- Coverage

- Baselines: Offline Greedy, “Robustified” versions of streaming algorithms
Empirical Evaluations

(a) n=20, 1 knapsack
(b) ego-Facebook, 1 knapsack
(c) ego-Twitter, 1 knapsack
(d) n=20, 2 knapsacks
(e) ego-Facebook, 2 knapsacks
(f) ego-Twitter, 2 knapsacks

Color Key:
- **Red**: AlgMult
- **Black**: MarginalRatio
- **Blue**: Multidimensional
- **Dotted Black**: Greedy
Empirical Evaluations

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>ml-20, 1 knapsack</th>
<th>fb, 1 knapsack</th>
<th>twitter, 1 knapsack</th>
<th>ml-20, 2 knapsacks</th>
<th>fb, 2 knapsacks</th>
<th>twitter, 2 knapsacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlgMult</td>
<td>641</td>
<td>378</td>
<td>401</td>
<td>1350</td>
<td>2745</td>
<td>4208</td>
</tr>
<tr>
<td>MarginalRatio</td>
<td>641</td>
<td>377</td>
<td>402</td>
<td>1350</td>
<td>2745</td>
<td>4209</td>
</tr>
<tr>
<td>Multidimensional</td>
<td>87</td>
<td>18</td>
<td>435</td>
<td>72</td>
<td>22</td>
<td>4221</td>
</tr>
<tr>
<td>Greedy</td>
<td>647</td>
<td>393</td>
<td>493</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: Sizes of robust summaries produced by the algorithms ($K = 10$).
Related Questions?

- Non-monotone robust submodular maximization
- Other constraints
- Better approximation guarantee
- Streaming algorithms with less space
Questions?