#project_tcs [2021-03-01]

[K. Han et al. Neurips 2020]

Deterministic Approximation for Submodular Maximization over a Matroid in Nearly Linear Time

Presenter: Jongseo Lee (leejseo, siroo fanclub)

Introduction

Definitions

- For a ground set $\mathcal{N}$, $f : 2^{\mathcal{N}} \to \mathbb{R}$ is
  - submodular if $\forall X, Y \in \mathcal{N}, f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$
  - non-negative if $\forall X \subseteq N, f(X) \geq 0$
  - non-monotone if $\exists X \subset Y, f(X) > f(Y)$
- For sake of convenience, let
  - $f(X \mid Y) = f(X \cup Y) - f(Y)$, called *marinal gain* of X w.r.t. Y
  - $f(x \mid Y) = f(\{x\} \mid Y)$

Definitions

- For a ground set $\mathcal{N}$ and $\mathcal{I} \subseteq 2^{\mathcal{N}}$, $(\mathcal{N}, \mathcal{I})$ is an independence system if i, ii holds. An independence system is a matroid if iii holds.
    i. $\varnothing \in \mathcal{I}$
    ii. $A \subseteq B, B \in \mathcal{I} \implies A \in \mathcal{I}$ (hereditary property)
    iii. $A, B \in \mathcal{I}, |A| < |B| \implies \exists x \in B - A, A \cup \{x\} \in \mathcal{I}$ (exchange property)
- For an independence system $(\mathcal{N}, \mathcal{I})$ and $X \subseteq \mathcal{N}$, a subset $Y \subseteq X$ is called *base* if Y is a maximally independent subset of X.

Some Properties

- For $X \subseteq Y \subseteq \mathcal{N}$ and $Z \subset \mathcal{N} - Y$, $f(Z \mid Y) \leq f(Z \mid X)$.
- For $X \subseteq Y \subseteq \mathcal{N}$, and a partition $Z_1, Z_2, \cdots, Z_t$ of $Y - X$,

$$f(Y \mid X) = \sum_{j=1}^{t} f(Z_j \mid Z_1 \cup \cdots \cup Z_{j-1} \cup X) \leq \sum_{j=1}^{t} f(Z_j \mid X).$$

Notations

- f represents a (non-monotone,) non-negative submodular functions
- $[n] = \{1, 2, \cdots, n\}$
- $r = \max\{|S| : S \in \mathcal{I}\}$ (i.e. rank)
- O: optimal solution to the problem (which will be presented in the next section)

Main Goal

Problem. Given a (non-monotone) submodular function f and a matroid $(\mathcal{N}, \mathcal{I})$, compute $\max\{f(S) : S \in \mathcal{I}\}$.

Compared to Previous Works:

Table 1: Approximation for Non-monotone Submodular Maximization over a Matroid

Algorithms	Ratio	Time Complexity	Type
Lee et al. [38]	$1/4 - \epsilon$	$\mathcal{O}((n^4 \log n)/\epsilon)$	Deterministic
Mirzasoleiman et al. [44]	$1/6 - \epsilon$	$\mathcal{O}(nr + r/\epsilon)$	Deterministic
Feldman et al. [25]	$1/4$	$\mathcal{O}(nr)$	Randomized
Buchbinder and Feldman [10]	0.385	$\text{poly}(n)$	Randomized
TwinGreedy (Alg. 1)	$1/4$	$\mathcal{O}(nr)$	Deterministic
TwinGreedyFast (Alg. 2)	$1/4 - \epsilon$	$\mathcal{O}((n/\epsilon) \log(r/\epsilon))$	Deterministic

TwinGreedy Algorithm

TwinGreedy Algorithm

Algorithm 1: $\mathsf{TwinGreedy}(\mathcal{N}, \mathcal{I}, f(\cdot))$

$S_1 \leftarrow \emptyset; S_2 \leftarrow \emptyset;$

repeat

  $\mathcal{M}_1 \leftarrow \{e \in \mathcal{N} \backslash (S_1 \cup S_2) : S_1 \cup \{e\} \in \mathcal{I}\}$

  $\mathcal{M}_2 \leftarrow \{e \in \mathcal{N} \backslash (S_1 \cup S_2) : S_2 \cup \{e\} \in \mathcal{I}\}$

  $C \leftarrow \{j \mid j \in \{1, 2\} \wedge \mathcal{M}_j \neq \emptyset\}$

  if $C \neq \emptyset$ **then**

    $(i, e) \leftarrow \arg\max_{j \in C, u \in \mathcal{M}_j} f(u \mid S_j)$; (ties broken arbitrarily)

    if $f(e \mid S_i) \leq 0$ **then Break**;

    $S_i \leftarrow S_i \cup \{e\}$;

until $\mathcal{M}_1 \cup \mathcal{M}_2 = \emptyset$;

$S^* \leftarrow \arg\max_{X \in \{S_1, S_2\}} f(X)$

return S^*

TwinGreedy Algorithm - Proof

Definition 1 *Consider the two solution sets* S_1 *and* S_2 *when TwinGreedy returns. We can write* $S_1 \cup S_2$ *as* $\{v_1, v_2, \cdots, v_k\}$ *where* $k = |S_1 \cup S_2|$*, such that* v_t *is added into* $S_1 \cup S_2$ *by the algorithm before* v_s *for any* $1 \leq t < s \leq k$*. With this ordered list, given any* $e = v_j \in S_1 \cup S_2$*, we define*

$$\mathrm{Pre}(e, S_1) = \{v_1, \cdots, v_{j-1}\} \cap S_1; \quad \mathrm{Pre}(e, S_2) = \{v_1, \cdots, v_{j-1}\} \cap S_2; \tag{2}$$

That is, $\mathrm{Pre}(e, S_i)$ *denotes the set of elements in* S_i *(*$i \in \{1, 2\}$*) that are added by the TwinGreedy algorithm before adding e. Furthermore, we define*

$$O_1^+ = \{e \in O \cap S_1 : \mathrm{Pre}(e, S_2) \cup \{e\} \in \mathcal{I}\}; \qquad O_1^- = \{e \in O \cap S_1 : \mathrm{Pre}(e, S_2) \cup \{e\} \notin \mathcal{I}\}$$
$$O_2^+ = \{e \in O \cap S_2 : \mathrm{Pre}(e, S_1) \cup \{e\} \in \mathcal{I}\}; \qquad O_2^- = \{e \in O \cap S_2 : \mathrm{Pre}(e, S_1) \cup \{e\} \notin \mathcal{I}\}$$
$$O_3 = \{e \in O \backslash (S_1 \cup S_2) : S_1 \cup \{e\} \notin \mathcal{I}\}; \qquad O_4 = \{e \in O \backslash (S_1 \cup S_2) : S_2 \cup \{e\} \notin \mathcal{I}\}$$

We also define the marginal gain of any $e \in S_1 \cup S_2$ *as* $\delta(e) = f(e \mid \mathrm{Pre}(e, S_1)) \cdot \mathbf{1}_{S_1}(e) + f(e \mid \mathrm{Pre}(e, S_2)) \cdot \mathbf{1}_{S_2}(e)$*, where* $\mathbf{1}_{S_i}(e) = 1$ *if* $e \in S_i$ *and* $\mathbf{1}_{S_i}(e) = 0$ *otherwise (*$\forall i \in \{1, 2\}$*).*

TwinGreedy Algorithm - Proof

Lemma 1. There's an injective function $\pi_1 : O_1^+ \cup O_1^- \cup O_2^- \cup O_3 \mapsto S_1$ such that

1. For any $e \in O_1^+ \cup O_1^- \cup O_2^- \cup O_3$, $Pre(\pi_1(e), S_1) \cup \{e\} \in \mathcal{I}$
2. For each $e \in O_1^+ \cup O_1^-$, $\pi_1(e) = e$

Similarly there's $\pi_2 : O_1^- \cup O_2^+ \cup O_2^- \cup O_4 \mapsto S_2$ s.t. $Pre(\pi_2(e), S_2) \cup \{e\} \in \mathcal{I}$ for each $e \in O_1^- \cup O_2^+ \cup O_2^- \cup O_4$ and $\pi_2(e) = e$ for each $e \in O_2^+ \cup O_2^-$.

Instead of giving whole proof to this lemma, we will just show some sketch including constructing π_1.

TwinGreedy Algorithm - Proof

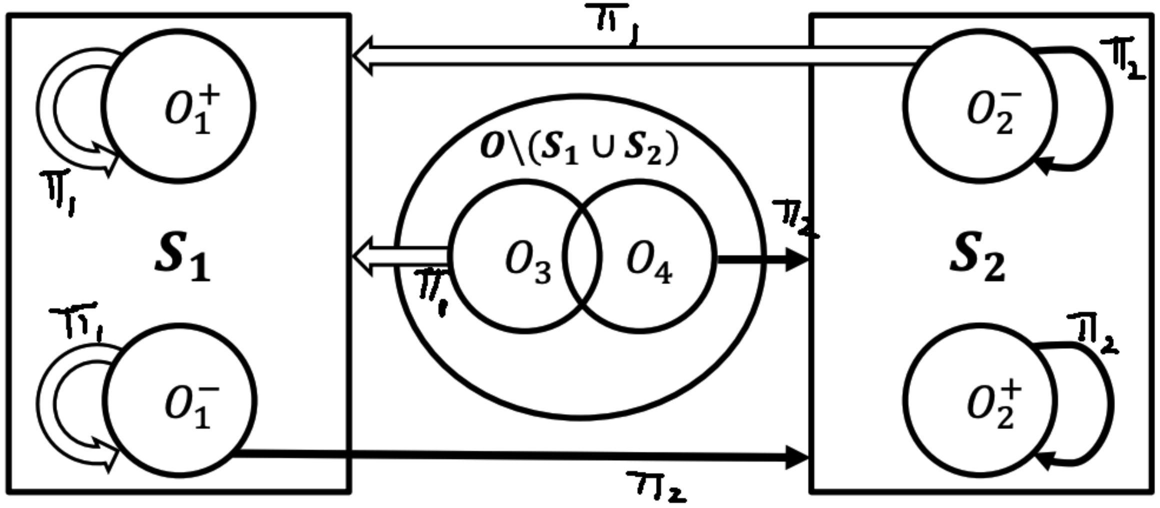
Let $S_1 = \{u_1, \cdots, u_s\}$ where u_i is added prior to u_j when $i < j$. Define $L_s = O_1^+ \cup O_1^- \cup O_2^- \cup O_3$. Run following algorithm.

for $j := s$ to 1

- $A_j := \{x \in L_j - \{u_1, \cdots, u_{j-1}\} | \{u_1, \cdots u_{j-1}, x\} \in \mathcal{I}\}$
- if $A_j = \varnothing$: continue
- if $u_j \in O_1^+ \cup O_1^-$ // (Note that $u_j \in A_j$)
  - $\pi_1(u_j)$ = u_j, $L_{j-1} = L_j - \{u_j\}$
- else
  - $e :=$ arbitrary element of A_j
  - $\pi_1(e) = u_j$, $L_{j-1} = L_j - \{e\}$

TwinGreedy Algorithm - Proof

Visualization of sets and mappings:



TwinGreedy Algorithm - Proof

Lemma 2.

- $f(O_1^+ \mid S_2) \leq \sum_{e \in O_1^+} \delta(\pi_1(e)); \quad f(O_2^+ \mid S_1) \leq \sum_{e \in O_2^+} \delta(\pi_2(e))$
- $f(O_1^- \mid S_2) \leq \sum_{e \in O_1^-} \delta(\pi_2(e)); \quad f(O_2^- \mid S_1) \leq \sum_{e \in O_2^-} \delta(\pi_1(e))$
- $f(O_4 \mid S_2) \leq \sum_{e \in O_4} \delta(\pi_2(e)); \quad f(O_3 \mid S_1) \leq \sum_{e \in O_3} \delta(\pi_1(e))$

TwinGreedy Algorithm - Proof

The authors decomposed Lemma 2 into 3 lemmas. Since I tried to follow the authors' numbering, I also decomposed it into 3 lemmas.

Lemma 4. $f(O_1^+ \mid S_2) \leq \sum_{e \in O_1^+} \delta(\pi_1(e)); \quad f(O_2^+ \mid S_1) \leq \sum_{e \in O_2^+} \delta(\pi_2(e))$

Lemma 5. $f(O_1^- \mid S_2) \leq \sum_{e \in O_1^-} \delta(\pi_2(e)); \quad f(O_2^- \mid S_1) \leq \sum_{e \in O_2^-} \delta(\pi_1(e))$

Lemma 6. $f(O_4 \mid S_2) \leq \sum_{e \in O_4} \delta(\pi_2(e)); \quad f(O_3 \mid S_1) \leq \sum_{e \in O_3} \delta(\pi_1(e))$

TwinGreedy Algorithm - Proof

Lemma 4. $f(O_1^+ \mid S_2) \leq \sum_{e \in O_1^+} \delta(\pi_1(e)); \quad f(O_2^+ \mid S_1) \leq \sum_{e \in O_2^+} \delta(\pi_2(e))$

At the exact moment $e \in O_1^+$ is added to S_1, one can also add e into S_2. So from greedy choice of the algorithm, we have $f(e \mid Pre(e, S_2)) \leq \delta(e)$.

From submodularity, we get:

$$f(e \mid S_2) \leq f(e \mid Pre(e, S_2)) \leq \delta(e) = \delta(\pi_1(e)),$$

$$\sum_{e \in O_1^+} f(O_1^+ \mid S_2) \leq \sum_{e \in O_1^+} f(e \mid S_2) \leq \sum_{e \in O_1^+} \delta(\pi_1(e)). \square$$

TwinGreedy Algorithm - Proof

Lemma 5. $f(O_1^- \mid S_2) \leq \sum_{e\in O_1^-} \delta(\pi_2(e)); \quad f(O_2^- \mid S_1) \leq \sum_{e\in O_2^-} \delta(\pi_1(e))$

Consider the exact moment $\pi_2(e)$ is added to S_2 (where $e \in O_1^-$):

- one can add e to S_2.
- e has not been inserted to S_1 yet.

TwinGreedy Algorithm - Proof

Lemma 5. $f(O_1^- \mid S_2) \leq \sum_{e \in O_1^-} \delta(\pi_2(e)); \quad f(O_2^- \mid S_1) \leq \sum_{e \in O_2^-} \delta(\pi_1(e))$

Consider the exact moment $\pi_2(e)$ is added to S_2 (where $e \in O_1^-$):

- one can add e to S_2.
- e has not been inserted to S_1 yet.

So, since $\pi_2(e)$ has been added to S_2 (*instead of* e) by the algorithm, from greedy-choice property, we have $f(e \mid Pre(\pi_2(e), S_2)) \leq \delta(\pi_2(e))$.

Since $Pre(\pi_2(e), S_2) \subseteq S_2$, we also have $f(e \mid S_2) \leq f(e \mid Pre(\pi_2(e), S_2))$.

TwinGreedy Algorithm - Proof

Lemma 5. $f(O_1^- \mid S_2) \leq \sum_{e \in O_1^-} \delta(\pi_2(e)); \quad f(O_2^- \mid S_1) \leq \sum_{e \in O_2^-} \delta(\pi_1(e))$

So, since $\pi_2(e)$ has been added to S_2 (*instead of* e) by the algorithm, from greedy-choice property, we have $f(e \mid Pre(\pi_2(e), S_2)) \leq \delta(\pi_2(e))$.

Since $Pre(\pi_2(e), S_2) \subseteq S_2$, we also have $f(e \mid S_2) \leq f(e \mid Pre(\pi_2(e), S_2)$.

Therefore,

$$f(O_1^- \mid S_2) \leq \sum_{e \in O_1^-} f(e \mid S_2) \leq \sum_{e \in O_1^-} f(e \mid Pre(\pi_2(e), S_2)) \leq \sum_{e \in O_1^-} \delta(\pi_2(e)). \square$$

TwinGreedy Algorithm - Proof

Lemma 6. $f(O_4 \mid S_2) \le \sum_{e \in O_4} \delta(\pi_2(e)); \quad f(O_3 \mid S_1) \le \sum_{e \in O_3} \delta(\pi_1(e))$

At the moment $\pi_1(e)$ is added to S_1 (for $e \in O_3$), one can add e to S_1.

From the property of greedy choice, we have $f(e \mid Pre(\pi_1(e), S_1)) \le \delta(\pi_1(e))$. (If not, e should be added to S_1 instead of $\pi_1(e)$.)

Therefore,

$$f(O_3 \mid S_1) \le \sum_{e \in O_3} f(e \mid S_1) \le \sum_{e \in O_3} f(e \mid Pre(\pi_1(e), S_1)) \le \sum_{e \in O_3} \delta(\pi_1(e)). \square$$

TwinGreedy Algorithm - Proof

Theorem 1. When $(\mathcal{N}, \mathcal{I})$ is a matroid, the algorithm returns a solution S^* with $\frac{1}{4}$ approximation ratio, under time complexity $\mathcal{O}(nr)$.

TwinGreedy Algorithm - Proof

Theorem 1. When $(\mathcal{N}, \mathcal{I})$ is a matroid, the algorithm returns a solution S^* with $\frac{1}{4}$ approximation ratio, under time complexity $\mathcal{O}(nr)$.

1. When $S_1 = \varnothing$ or $S_2 = \varnothing$, we get an optimal solution, so assume $S_1, S_2 \neq \varnothing$.
   - WLOG, suppose S_2 is empty.
   - From greedy choice,

$$f(O \cap S_1 \mid \varnothing) \leq \sum_{e \in O \cap S_1} f(e \mid \varnothing) \leq \sum_{O \cap S_1} \delta(e) \leq \sum_{e \in S_1} \delta(e) = f(S_1 \mid \varnothing)$$

   - and $f(O - S_1 \mid \varnothing) \leq \sum_{e \in O - S_1} f(e \mid \varnothing) \leq 0$. (If not, they would have been added to S_2.)
   - From $f(O - S_1) + f(O \cap S_1) \geq f(O) + f(\varnothing)$, $f(S_1) \geq f(O)$. $\square$

TwinGreedy Algorithm - Proof

Theorem 1. When $(\mathcal{N}, \mathcal{I})$ is a matroid, the algorithm returns a solution S^* with $\frac{1}{4}$ approximation ratio, under time complexity $\mathcal{O}(nr)$.

2. Define $O_5 := O - (S_1 \cup S_2 \cup O_3)$, $O_6 := O - (S_1 \cup S_w \cup O_4)$.
    - $f(O \cup S_1) - f(S_1) \leq f(O_2^+ \mid S_1) + f(O_2^- \mid S_1) + f(O_3 \mid S_1) + f(O_5 \mid S_1)$ and its variant for $f(O \cup S_2) - f(S_2)$.

Because

- For $X \subseteq Y \subseteq \mathcal{N}$, and a partition $Z_1, Z_2, \cdots, Z_t$ of $Y - X$,

$$f(Y \mid X) = \sum_{j=1}^{t} f(Z_j \mid Z_1 \cup \cdots \cup Z_{j-1} \cup X) \leq \sum_{j=1}^{t} f(Z_j \mid X).$$

TwinGreedy Algorithm - Proof

Theorem 1. When $(\mathcal{N}, \mathcal{I})$ is a matroid, the algorithm returns a solution S^* with $\frac{1}{4}$ approximation ratio, under time complexity $\mathcal{O}(nr)$.

2. Define $O_5 := O - (S_1 \cup S_2 \cup O_3)$, $O_6 := O - (S_1 \cup S_w \cup O_4)$.
    - $f(O \cup S_1) - f(S_1) \leq f(O_2^+ \mid S_1) + f(O_2^- \mid S_1) + f(O_3 \mid S_1) + f(O_5 \mid S_1)$ and its variant for $f(O \cup S_2) - f(S_2)$.
    - $f(O_2^+ \mid S_1) + f(O_2^- \mid S_1) + f(O_3 \mid S_1) + f(O_1^+ \mid S_2) + f(O_1^- \mid S_2) + f(O_4 \mid S_2) \leq f(S_1) + f(S_2)$.

TwinGreedy Algorithm - Proof

Theorem 1. When $(\mathcal{N}, \mathcal{I})$ is a matroid, the algorithm returns a solution S^* with $\frac{1}{4}$ approximation ratio, under time complexity $\mathcal{O}(nr)$.

Some computations...

$$\begin{aligned}
f(O_2^+ \mid S_1) + f(O_2^- \mid S_1) + f(O_3 \mid S_1) + f(O_1^+ \mid S_2) + f(O_1^- \mid S_2) + f(O_4 \mid S_2) \\
\leq \sum_{e \in O_1^+ \cup O_2^- \cup O_3} \delta(\pi_1(e)) + \sum_{e \in O_1^- \cup O_2^+ \cup O_4} \delta(\pi_2(e)) \\
\leq \sum_{e \in S_1} \delta(e) + \sum_{e \in S_2} \delta(e) \\
\leq f(S_1) + f(S_2)
\end{aligned}$$

TwinGreedy Algorithm - Proof

Theorem 1. When $(\mathcal{N}, \mathcal{I})$ is a matroid, the algorithm returns a solution S^* with $\frac{1}{4}$ approximation ratio, under time complexity $\mathcal{O}(nr)$.

2. Define $O_5 := O - (S_1 \cup S_2 \cup O_3)$, $O_6 := O - (S_1 \cup S_w \cup O_4)$.
   - $f(O \cup S_1) - f(S_1) \leq f(O_2^+ \mid S_1) + f(O_2^- \mid S_1) + f(O_3 \mid S_1) + f(O_5 \mid S_1)$ and its variant for $f(O \cup S_2) - f(S_2)$.
   - $f(O_2^+ \mid S_1) + f(O_2^- \mid S_1) + f(O_3 \mid S_1) + f(O_1^+ \mid S_2) + f(O_1^- \mid S_2) + f(O_4 \mid S_2) \leq f(S_1) + f(S_2)$.
   - $f(O_5 \mid S_1) \leq \sum_{e \in O_5} f(e \mid S_1) \leq 0$, $f(O_6 \mid S_2) \leq \sum_{e \in O_6} f(e \mid S_2) \leq 0$.

TwinGreedy Algorithm - Proof

Theorem 1. When $(\mathcal{N}, \mathcal{I})$ is a matroid, the algorithm returns a solution S^* with $\frac{1}{4}$ approximation ratio, under time complexity $\mathcal{O}(nr)$.

2. Define $O_5 := O - (S_1 \cup S_2 \cup O_3)$, $O_6 := O - (S_1 \cup S_w \cup O_4)$.
    - $f(O \cup S_1) - f(S_1) \leq f(O_2^+ \mid S_1) + f(O_2^- \mid S_1) + f(O_3 \mid S_1) + f(O_5 \mid S_1)$ and its variant for $f(O \cup S_2) - f(S_2)$.
    - $f(O_2^+ \mid S_1) + f(O_2^- \mid S_1) + f(O_3 \mid S_1) + f(O_1^+ \mid S_2) + f(O_1^- \mid S_2) + f(O_4 \mid S_2) \leq f(S_1) + f(S_2)$.
    - $f(O_5 \mid S_1) \leq \sum_{e \in O_5} f(e \mid S_1) \leq 0$, $f(O_6 \mid S_2) \leq \sum_{e \in O_6} f(e \mid S_2) \leq 0$.
    - $f(O) \leq f(O) + f(O \cup S_1 \cup S_2) \leq f(O \cup S_1) + f(O \cup S_2)$ (from submodularity and $S_1 \cap S_2 = \varnothing$)

TwinGreedy Algorithm - Proof

Theorem 1. When $(\mathcal{N}, \mathcal{I})$ is a matroid, the algorithm returns a solution S^* with $\frac{1}{4}$ approximation ratio, under time complexity $\mathcal{O}(nr)$.

- $f(O \cup S_1) - f(S_1) \leq f(O_2^+ \mid S_1) + f(O_2^- \mid S_1) + f(O_3 \mid S_1) + f(O_5 \mid S_1)$ and its variant for $f(O \cup S_2) - f(S_2)$.
- $f(O_2^+ \mid S_1) + f(O_2^- \mid S_1) + f(O_3 \mid S_1) + f(O_1^+ \mid S_2) + f(O_1^- \mid S_2) + f(O_4 \mid S_2) \leq f(S_1) + f(S_2)$.
- $f(O_5 \mid S_1) \leq \sum_{e \in O_5} f(e \mid S_1) \leq 0, f(O_6 \mid S_2) \leq \sum_{e \in O_6} f(e \mid S_2) \leq 0$.
- $f(O) \leq f(O) + f(O \cup S_1 \cup S_2) \leq f(O \cup S_1) + f(O \cup S_2)$

Now let's put them all together!

TwinGreedy Algorithm - Proof

Theorem 1. When $(\mathcal{N}, \mathcal{I})$ is a matroid, the algorithm returns a solution S^* with $\frac{1}{4}$ approximation ratio, under time complexity $\mathcal{O}(nr)$.

- $f(O \cup S_1) - f(S_1) \leq f(O_2^+ \mid S_1) + f(O_2^- \mid S_1) + f(O_3 \mid S_1) + f(O_5 \mid S_1)$ and its variant for $f(O \cup S_2) - f(S_2)$.
- $f(O_2^+ \mid S_1) + f(O_2^- \mid S_1) + f(O_3 \mid S_1) + f(O_1^+ \mid S_2) + f(O_1^- \mid S_2) + f(O_4 \mid S_2) \leq f(S_1) + f(S_2)$.
- $f(O_5 \mid S_1) \leq \sum_{e \in O_5} f(e \mid S_1) \leq 0, f(O_6 \mid S_2) \leq \sum_{e \in O_6} f(e \mid S_2) \leq 0.$
- $f(O) \leq f(O) + f(O \cup S_1 \cup S_2) \leq f(O \cup S_1) + f(O \cup S_2)$

Now let's put them all together!

$$f(O) \leq 2f(S_1) + 2f(S_2) \leq 4 \cdot \max(f(S_1), f(S_2)) \square$$

TwinGreedy Algorithm - Proof

Theorem 1. When $(\mathcal{N}, \mathcal{I})$ is a matroid, the algorithm returns a solution S^* with $\frac{1}{4}$ approximation ratio, under time complexity $\mathcal{O}(nr)$.

Remark. If monotone condition of f is given, the algorithm guarantees $1/2$ approximation ratio.

TwinGreedyFast Algorithm

TwinGreedyFast Algorithm

Algorithm 2: $\mathsf{TwinGreedyFast}(\mathcal{N}, \mathcal{I}, f(\cdot), \epsilon)$

$\tau_{max} \leftarrow \max\{f(e) : e \in \mathcal{N} \wedge \{e\} \in \mathcal{I}\}$;
$S_1 \leftarrow \emptyset; S_2 \leftarrow \emptyset$;
for $(\tau \leftarrow \tau_{max};\ \ \tau > \epsilon\tau_{max}/[r(1+\epsilon)];\ \ \tau \leftarrow \tau/(1+\epsilon))$ **do**
  foreach $e \in \mathcal{N} \backslash (S_1 \cup S_2)$ **do**
    $\Delta_1 \leftarrow -\infty; \Delta_2 \leftarrow -\infty$ /*two signals*/
    if $S_1 \cup \{e\} \in \mathcal{I}$ **then** $\Delta_1 \leftarrow f(e \mid S_1)$;
    if $S_2 \cup \{e\} \in \mathcal{I}$ **then** $\Delta_2 \leftarrow f(e \mid S_2)$;
    $i \leftarrow \arg\max_{j \in \{1,2\}} \Delta_j$; (ties broken arbitrarily)
    if $\Delta_i \geq \tau$ **then** $S_i \leftarrow S_i \cup \{e\}$;
$S^* \leftarrow \arg\max_{X \in \{S_1, S_2\}} f(X)$
return S^*

TwinGreedyFast Algorithm - Proof

This algorithm has similar properties compared to TwinGreedy but proofs are somewhat more time-consuming, so I will omit the proof.

Algorithm 2: TwinGreedyFast$(\mathcal{N}, \mathcal{I}, f(\cdot), \epsilon)$

$\tau_{max} \leftarrow \max\{f(e) : e \in \mathcal{N} \wedge \{e\} \in \mathcal{I}\}$;
$S_1 \leftarrow \emptyset; S_2 \leftarrow \emptyset$;
for $(\tau \leftarrow \tau_{max};\ \tau > \epsilon\tau_{max}/[r(1+\epsilon)];\ \tau \leftarrow \tau/(1+\epsilon))$ **do**
  foreach $e \in \mathcal{N} \backslash (S_1 \cup S_2)$ **do**
    $\Delta_1 \leftarrow -\infty; \Delta_2 \leftarrow -\infty$ /*two signals*/
    if $S_1 \cup \{e\} \in \mathcal{I}$ **then** $\Delta_1 \leftarrow f(e \mid S_1)$;
    if $S_2 \cup \{e\} \in \mathcal{I}$ **then** $\Delta_2 \leftarrow f(e \mid S_2)$;
    $i \leftarrow \arg\max_{j \in \{1,2\}} \Delta_j$; (ties broken arbitrarily)
    if $\Delta_i \geq \tau$ **then** $S_i \leftarrow S_i \cup \{e\}$;
$S^* \leftarrow \arg\max_{X \in \{S_1, S_2\}} f(X)$
return S^*

TwinGreedyFast Algorithm - Proof

Lemma 3. The algorithm satisfies:

- $f(O_1^+ \mid S_2) \leq \sum_{e \in O_1^+} \delta(\pi_1(e))$,
  $f(O_2^+ \mid S_1) \leq \sum_{e \in O_2^+} \delta(\pi_2(e))$
- $f(O_1^- \mid S_2) \leq (1+\epsilon) \sum_{e \in O_1^-} \delta(\pi_2(e))$,
  $f(O_2^- \mid S_1) \leq (1+\epsilon) \sum_{e \in O_2^-} \delta(\pi_1(e))$
- $f(O_4 \mid S_2) \leq (1+\epsilon) \sum_{e \in O_4} \delta(\pi_2(e))$,
  $f(O_3 \mid S_1) \leq (1+\epsilon) \sum_{e \in O_3} \delta(\pi_1(e))$

TwinGreedyFast Algorithm - Proof

Theorem 2. When $(\mathcal{N}, \mathcal{I})$ is a matroid, the algorithm returns a solution with $1/4 - \epsilon$ approximation ratio, with time complexity of $\mathcal{O}(\frac{n}{\epsilon} \log \frac{r}{\epsilon})$.

Extensions

Extensions

For a ground set $\mathcal{N}$ and $\mathcal{I} \subseteq 2^{\mathcal{N}}$, $(\mathcal{N}, \mathcal{I})$ is an independence system if 1, 2 holds. An independence system is a p-set system if 3 holds.

1. $\varnothing \in \mathcal{I}$
2. $A \subseteq B, B \in \mathcal{I} \implies A \in \mathcal{I}$ (hereditary property)
3. For every $Y \subseteq N$ and two base X_1, X_2 of Y, $|X_1| \leq p|X_2|$ where $p \geq 1$.

Extensions

Theorem 3. TwinGreedyFast algorithm can be used to handle the problem of submodular maximization over a p-set system. It returns a solution with $\frac{1}{2p+2} - \epsilon$ approximation ratio, with time complexity of $\mathcal{O}(\frac{n}{\epsilon}\log\frac{r}{\epsilon})$.

Applications

Applications - Social Network Monitoring

Problem. Suppose given a graph $G = (V, E, w)$. Let $\{V_1, V_2, \cdots, V_h\}$ be the partition of V. We want to select $S \subseteq V$ to maximize total amount of monitored content: $f(S) = \sum_{(u,v)\in E, u\in S, v\notin S} w(u, v)$. We also want $|S \cap V_i| \leq k$ for every i, where k is some constant.

Applications - Multi-Product Viral Marketing

Problem. Suppose given a graph $G = (V, E, c)$ where $c(u)$ represents some cost for node u and non-negative submodular functions $f_1, f_2, \cdots, f_m$ with domain 2^V. We have two constants B, k and want to maximize $\sum_{i\in[m]} f_i(S_i) + (B - \sum_{i\in[m]}\sum_{v\in S_i} c(v))$.

No two of S_i share elements.

Applications - Multi-Product Viral Marketing

Problem. Suppose given a graph $G = (V, E, c)$ where $c(u)$ represents some cost for node u and non-negative submodular functions $f_1, f_2, \cdots, f_m$ with domain 2^V. We have two constants B, k and want to maximize $\sum_{i\in[m]} f_i(S_i) + (B - \sum_{i\in[m]}\sum_{v\in S_i} c(v))$.

No two of S_i share elements.

Lemma. Define the ground set $\mathcal{N} = V \times [m]$ and $\mathcal{I} = \{X \subseteq \mathcal{N} : |X| \le k \wedge \forall u \in V, |X \cap N_u| \le 1\}$, where $N_u = \{(u, i) : i \in [m]\}$. Then, $(\mathcal{N}, \mathcal{I})$ is a matroid.

Lemma. For any $S \subseteq \mathcal{N}$ and $S \neq \varnothing$, define $f(S) = \sum_{i\in[m]} f_i(S_i) + (B - \sum_{i\in[m]}\sum_{v\in S_i} c(v))$, where $S_i := \{u : (u, i) \in S\}$ and f_i is a non-negative submodular with domain 2^V. Define $f(\varnothing) = 0$. Then, f is a submodular function defined on $2^{\mathcal{N}}$.

Conclusion

Conclusion

- We have discovered two simple deterministic greedy algorithms maximizing (non-monotone) non-negative submodular function over a matroid with $1/4(-\epsilon)$ approximation ratio.
- TwinGreedy algorithm achieves $1/2$ approximation ratio with monotone constraint.
- TwinGreedyFast algorithm can also be directly used to handle more general p-set system constraint.
- We have discovered two real-world problems that the paper's algorithms can be used.