236621 - Algorithms for Submodular Optimization

Roy Schwartz

April 15, 2019

Abstract

1 Introduction

We are looking on $f : 2^N \to \mathbb{R}$ for some set $N = \{1, \dots n\}$

Definition 1.1. f is submodular if

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \tag{1}$$

Definition 1.2. Return of u wrt A is $f(A \cup \{u\}) - f(A)$

Definition 1.3 (Diminishing returns)**.** f has diminishing returns if for $A \subseteq B$

$$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B) \tag{2}$$

Proposition 1.1. f is submodular iff f has diminishing returns

Proof. $\Rightarrow$:
Let $A \subseteq B \subseteq N$ and $u \notin B$. Lets use submodularity property on $A \cup \{u\}$ and B:

$$f(A \cup \{u\}) + f(B) \geq f(A \cup \{u\} \cup B) + f((A \cup \{u\}) \cap B) = f(B \cup \{u\}) + f(A) \tag{3}$$

Thus

$$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B) \tag{4}$$

□

$\Leftarrow$:
We'll proof by induction over $|A \cup B| - |A \cap B|$, i.e., size of symmetric difference.
Basis: $|A \cup B| - |A \cap B| = 0$, then $A = B$, and then submodular property is fulfilled.
Step: assume $|A \cup B| - |A \cap B| = k$. WLOG let $u \in A$ such that $u \notin B$.

$$f(A) + f(B) = f(A) - f(A \setminus \{u\}) + f(A \setminus \{u\}) + f(B) \geq \tag{5}$$
$$\geq f(A) - f(A \setminus \{u\}) + f(A \setminus \{u\} \cup B) + f(A \setminus \{u\} \cap B) \geq \tag{6}$$
$$\geq f(A \cup B) - f(A \cup B \setminus \{u\}) + f(A \cup B \setminus \{u\}) + f(A \cap B) = f(A \cup B) + f(A \cap B) \tag{7}$$

Definition 1.4 (Monotonous function)**.** f is non-decreasing monotonous if $\forall A \subseteq B \subseteq N$, $f(A) \leq f(B)$.

Definition 1.5 (Symmetric function)**.** f is symmetric if $\forall S \subseteq N$, $f(S) \leq f(N \setminus S)$.

Definition 1.6 (Normalized function)**.** f is normalized if $f(\emptyset) = 0$.

Examples

Linear function $\forall n \in N$ exists weight w_n and

$$f(S) = \sum_{u \in S} w_u + b \tag{8}$$

Such f is submodular.

Budget additive function (clipped linear function) $\forall n \in N$ exists weight w_n and

$$f(S) = \min\left\{\sum_{u \in S} w_u, b\right\} \tag{9}$$

Such f is submodular.

Coverage function Given set X and n subsets $S_1, S_2, \ldots, S_n \subset X$ define

$$f(S) = \left|\bigcup_{i \in S} S_i\right| \tag{10}$$

This f is obviously submodular.

Graph cuts Let $G + (V, E)$ be a graph and $w : E \to \mathbb{R}^+$ weights of edges. Given a cut $S \subseteq V$ define $\delta(S)$ to be sum of weights of all edges going through the cut. $\delta : 2^V \to \mathbb{R}^+$ is submodular, normalized, and symmetric.

Rank function Let $v_1, \ldots, v_n \in \mathbb{R}^d$ vectors, and

$$f(S) = \operatorname{rank}(S) = \dim \operatorname{span}\big(\{v_i | i \in S\}\big) \tag{11}$$

2 Submodular optimization

Given world N, submodular function $f : 2^N \to \mathbb{R}^+$, and a family of feasible solutions $\mathcal{I} \subseteq 2^N$

$$\max f(S) \tag{12}$$
$$\text{s.t. } S \in \mathcal{I} \tag{13}$$

Note Most of submodular functions (except for logarithm of determinant of submatrix) are nonnegative. We use the condition to have properly defined multiplicative approximation.

Note How f is given in input? Obviously, not as a list of values, since it's exponential in $|N|$. Thus we represent f with black box, and same applies for constraints. Usually, constraints are simple.

2.1 Examples of submodular optimization problems

Example f is submodular and there are no constraints. It generalizes MAX-CUT, MAX-DICUT

Example f is submodular and there is size constraint:

$$\max f(S) \tag{14}$$
$$\text{s.t. } |S| \leq k \tag{15}$$

. It generalizes MAX-K-COVER.

Submodular welfare

3 Maximization of the submodular function with cardinality constraints

$$\max f(S) \tag{16}$$
$$\text{s.t. } |S| \leq k \tag{17}$$

Algorithm 1 Nemhauser-Wolsey-Fisher

```
1: procedure GREEDY($N$)
2:     $A \leftarrow \emptyset$
3:     for $i = 1$ to $k$ do
4:         Let $u_i \in N$ maximize $f(A_{i-1} \cup \{u_i\}) - f(A_{i-1})$
5:         $A_i \leftarrow A_{i-1} \cup \{u_i\}$
6:     end for
7:     return $A_k$
8: end procedure
```

Greedy algorithm If f is monotonic, greedy algorithm is an optimal approximating algorithm.

Lemma 3.1. For submodular $f : 2^N \to \mathbb{R}_+$,

$$f(A \cup B) - f(A) \leq \sum_{b_i \in B} f(A \cup \{b_i\}) - f(A) \tag{18}$$

Proof.

$$f(A \cup B) - f(A) = \sum_i f(A \cup \{b_1, \dots b_{i-1}\} \cup \{b_i\}) - f(A \cup \{b_1, \dots b_{i-1}\}) \leq \sum_i f(A \cup \{b_i\}) - f(A) \tag{19}$$

□

Proposition 3.2 (Nemhauser et al. [1978]). Algorithm 1 is $1 - \frac{1}{e}$ optimal.

Proof. For optimal set S^*

$$f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) \geq \max_{u \in S^*} \{f(A_{i-1} \cup \{u\}) - f(A_{i-1})\} \geq \frac{1}{k} \sum_{u \in S^*} [f(A_{i-1} \cup \{u\}) - f(A_{i-1})] \geq \tag{20}$$

$$\geq \frac{1}{k}\Big(f(A_{i-1} \cup S^*) - f(A_{i-1})\Big) \geq \frac{1}{k}\Big[f(S^*) - f(A_{i-1})\Big] \tag{21}$$

We got a recursion equation:

$$f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) \geq \frac{1}{k}\Big[f(S^*) - f(A_{i-1})\Big] \tag{22}$$

We can solve the recursion and acquire

$$f(A_k) \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) f(S^*) + \left(1 - \frac{1}{k}\right)^k f(A_0) \geq \left(1 - \frac{1}{e}\right) f(S^*) \tag{23}$$

□

Theorem 3.3 (Nemhauser and Wolsey [1978]). For all constant $\epsilon > 0$ each algorithm acquiring $1 - \frac{1}{e} + \epsilon$ requires exponential number of requests to value oracle.

Theorem 3.4 (Feige [1998]). For MAX-K-COVER all constant $\epsilon > 0$ each algorithm acquiring $1 - \frac{1}{e} + \epsilon$ requires exponential number of requests to value oracle unless $P = NP$.

Note Runtime of Algorithm 1 is $\mathcal{O}(nk)$. It is possible to acquire $\mathcal{O}\big(n \lg(\frac{1}{\epsilon})\big)$ runtime and $1 - \frac{1}{e} - \epsilon$ optimality by looking on some subset of N at each step instead of the whole set.

3.1 Non-monotonic functions

What happens if f is not monotonic? First of all, does greed algorithm work? Not only it is not optimal approximation, it can be as bad as $\frac{2}{N}$. However, it can be fixed. The idea is to randomize algorithm to prevent it from "bad" choices.

Algorithm 2

```
1: procedure RANDOMIZED GREEDY($N$)
2:     $A \leftarrow \emptyset$
3:     for $i = 1$ to $k$ do
4:         $M_i \leftarrow \arg\max_{B \subseteq N : |B| \le k} \sum_{u \in B} f(A_{i-1} \cup \{u\}) - f(A_{i-1})$
5:         $A_i \leftarrow \begin{cases} A_{i-1} \cup \{u\} & \forall u \in M_i \text{ with } P = \frac{1}{k} \\ A_{i-1} & \text{with } P = 1 - \frac{|M_i|}{k} \end{cases}$
6:     end for
7:     return $A_k$
8: end procedure
```

Randomized greedy algorithm

Theorem 3.5 (Buchbinder et al. [2014]). In monotonic case, Algorithm 2 is $1 - \frac{1}{e}$ optimal in expectation.

Proof. Take a look at i^{th} iteration and condition on previous iterations, denote a chosen element from M_i as u_i:

$$\mathbb{E}\Big[f(A_{i-1} \cup \{u_i\}) - f(A_{i-1})|A_{i-1}\Big] = \frac{1}{k} \sum_{u_i \in M_i} f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) \ge \frac{1}{k}(f(A_{i-1} \cup S^*) - f(A_{i-1})) \ge \tag{24}$$

$$\ge \frac{1}{k}(f(S^*) - f(A_{i-1})) \ge \frac{1}{k}(f(S^*) - f(A_{i-1})) \tag{25}$$

If the inequality is right for any A_{i-1} it is right, from tower property, in expectation over A_{i-1}:

$$\mathbb{E}\Big[f(A_{i-1} \cup \{u_i\}) - f(A_{i-1})\Big] \ge \frac{1}{k}(f(S^*) - \mathbb{E}[f(A_{i-1})]) \tag{26}$$

And thus we can once again solve the recurrence and acquire same result as in Proposition 3.2 on the previous page. □

Lemma 3.6. Given set $B \subseteq N$ such that

$$\forall u \in N \quad P(u \in B) \le p \tag{27}$$

then

$$\mathbb{E}[f(B)] \ge (1-p)f(\emptyset) \tag{28}$$

Proof. WLoG $p(u_1 \in B) \ge p(u_2 \in B) \ge \cdots \ge p(u_n \in B)$. Denote

$$X_i = \mathbb{1}_{u_i \in B} N_i = \bigcup_{j=1}^{i} u_j \tag{29}$$

We can then rewrite

$$f(B) = f(N_0) + \sum_{i=1}^{n} X_i \Big(f(B \cap N_i) - f(B \cap N_{i-1})\Big) \tag{30}$$

$$\mathbb{E}[f(B)] = f(N_0) + \sum_{i=1}^{n} \mathbb{E}\Big[X_i \Big(f(B \cap N_i) - f(B \cap N_{i-1})\Big)\Big] \ge \tag{31}$$

$$\ge f(N_0) + \sum_{i=1}^{n} \Big(f(N_i) - f(N_{i-1})\Big)\mathbb{E}[X_i] = f(N_0) + \sum_{i=1}^{n} \Big(f(N_i) - f(N_{i-1})\Big)p_i = \tag{32}$$

$$= f(N_0)(1-p_1) + \sum_{i=1}^{n} f(N_i)\underbrace{(p_i - p_{i+1})}_{\le 0} \ge f(N_0)(1-p_1) \ge f(\emptyset)(1-p) \tag{33}$$

□

Lemma 3.7. Given set $A \subseteq N$ and set $B \subseteq N$ such that

$$\forall u \in N \quad P(u \in B) \leq p \tag{34}$$

$$\mathbb{E}[f(A \cup B)] \geq (1-p)f(A) \tag{35}$$

Proof. Define

$$g_A(S) = f(A \cup S) \tag{36}$$

Obviously, g_A is also submodular (from diminishing returns). Then, from Lemma 3.6 on the preceding page

$$\mathbb{E}[f(A \cup B)] = \mathbb{E}[g(B)] \geq (1-p)g(\emptyset) = (1-p)f(A) \tag{37}$$

□

Theorem 3.8 (Buchbinder et al. [2014])**.** In non-monotonic case, Algorithm 2 on the previous page is $\frac{1}{e}$ optimal in expectation.

Proof. Similarly to monotonic case, take a look at i^{th} iteration and condition on previous iterations, denote a chosen element from M_i as u_i:

$$\mathbb{E}\Big[f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) | A_{i-1}\Big] = \frac{1}{k} \sum_{u_i \in M_i} f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) \geq \frac{1}{k}(f(A_{i-1} \cup S^*) - f(A_{i-1})) \tag{38}$$

Since

$$P(u \in A_{i-1}) \leq 1 - \left(1 - \frac{1}{k}\right)^{i-1} \tag{39}$$

from Lemma 3.7

$$\mathbb{E}[f(A_{i-1} \cup S^*)] \geq \left(1 - \frac{1}{k}\right)^{i-1} f(S^*) \tag{40}$$

Thus, taking expectation

$$\mathbb{E}\Big[f(A_{i-1} \cup \{u_i\}) - f(A_{i-1})\Big] \geq \frac{1}{k}(f(A_{i-1} \cup S^*) - f(A_{i-1})) \geq \frac{1}{k}\left[\left(1 - \frac{1}{k}\right)^{i-1} f(S^*) - \mathbb{E}\Big[f(A_{i-1})\Big]\right] \tag{41}$$

$$\mathbb{E}\Big[f(A_i)\Big] \geq \frac{1}{k}(f(A_{i-1} \cup S^*) - f(A_{i-1})) \geq \frac{1}{k}\left[\left(1 - \frac{1}{k}\right)^{i-1} f(S^*) - \mathbb{E}\Big[f(A_{i-1})\Big]\right] \tag{42}$$

Solving the recurrence we get

$$\mathbb{E}[f(A)] \geq \frac{i}{k}\left(1 - \frac{1}{k}\right)^{k-1} f(S^*) \geq \frac{1}{e} f(S^*) \tag{43}$$

i.e.,

$$\mathbb{E}[f(A_k)] \geq \left(1 - \frac{1}{k}\right)^{k-1} f(S^*) \geq \frac{1}{e} f(S^*) \tag{44}$$

□

Note Algorithm 2 on the previous page is not optimal. In addition, the upper bound of the best approximation is 0.49.

Runtime Runtime of Algorithm 2 on the preceding page is $\mathcal{O}(nk)$.

4 Maximization of the submodular function without constraints

$$\max f(S) \tag{45}$$

Examples

- MAX-CUT
- MAX-DIRECTED-CUT
- Max Facility Location
- MAX-SAT (with all literals in a clause having same sign).

Proposition 4.1 ([Feige et al., 2011]). Algorithm which choose random solution as following: $u \in S$ with probability $\frac{1}{2}$ independently, is $\frac{1}{4}$ approximation in expectation:

$$\mathbb{E}[f(S)] \geq \frac{1}{4} f(S^*) \tag{46}$$

Proposition 4.2 ([Feige et al., 2011]). If f is symmetric, the same algorithm is $\frac{1}{2}$ approximation in expectation:

$$\mathbb{E}[f(S)] \geq \frac{1}{2} f(S^*) \tag{47}$$

Proposition 4.3 ([Feige et al., 2011]). For any constant $\epsilon > 0$ it is impossible to acquire $\left(\frac{1}{2} + \epsilon\right)$ approximation in polynomial time, even in symmetric case.

Note that for $\bar{f}(S) = f(\bar{S})$, we can use the same oracle. So a "conjugate" algorithm would be start from N and drop elements from it.

Algorithm 3

```
1:  procedure DOUBLE GREEDY($N$)
2:      $X \leftarrow \emptyset$, $Y \leftarrow N$
3:      for $i = 1$ to $n$ do
4:          $a_i = f(X_{i-1} \cup \{u_i\}) - f(X_i)$
5:          $b_i = f(Y_{i-1} \setminus \{u_i\}) - f(Y_i)$
6:          if $a_i > b_i$ then
7:              $X_i \leftarrow X_{i-1} \cup \{u_i\}$
8:              $Y_i \leftarrow Y_{i-1}$
9:          else
10:             $X_i \leftarrow X_{i-1}$
11:             $Y_i \leftarrow Y_{i-1} \setminus \{u_i\}$
12:         end if
13:     end for
14:     return $X_N$
15: end procedure
```

Algorithm 4

```
1:  procedure RANDOMIZED DOUBLE GREEDY($N$)
2:      $X \leftarrow \emptyset$, $Y \leftarrow N$
3:      for $i = 1$ to $n$ do
4:          $a_i = \max\{0, f(X_{i-1} \cup \{u_i\}) - f(X_i)\}$
5:          $b_i = \max\{0, f(Y_{i-1} \setminus \{u_i\}) - f(Y_i)\}$
6:          $(X_i, Y_i) \leftarrow \begin{cases} (X_{i-1} \cup \{u_i\}, Y_i) & \text{with } P = \frac{a_i}{a_i+b_i} \\ (X_{i-1}, Y_{i-1} \setminus \{u_i\}) & \text{with } P = \frac{b_i}{a_i+b_i} \end{cases}$
7:      end for
8:      return $X_N$
9:  end procedure
```

Proposition 4.4. It's impossible that both $f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}) < 0$ and $f(Y_{i-1} \setminus \{u_i\}) - f(Y_i) < 0$.

Proof. From diminishing returns:

$$f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}) \geq f(Y_i) - f(Y_{i-1} \setminus \{u_i\}) \tag{48}$$
$$f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1} \setminus \{u_i\}) \geq 0 \tag{49}$$

Thus at least one of $f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})$ and $f(Y_{i-1} \setminus \{u_i\}) - f(Y_i)$ is greater than 0. □

Lemma 4.5. Let S^* be an optimal solution and

$$S_i^* = S^* \cup X_i \cap Y_i \tag{50}$$

i.e., optimal solution to which we add everything Algorithm 3 added and drop everything it dropped.
For all i:

$$f(S_{i-1}^*) - f(S_i^*) \leq f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1}) \tag{51}$$

Lemma 4.6. Let S^* be an optimal solution and

$$S_i^* = S^* \cup X_i \cap Y_i \tag{52}$$

i.e., optimal solution to which we add everything Algorithm 4 added and drop everything it dropped.
For all i:

$$\mathbb{E}\Big[f(S_{i-1}^*) - f(S_i^*)\Big] \leq \frac{1}{2}\mathbb{E}\Big[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})\Big] \tag{53}$$

Proof. Take a look at i^{th} iteration and condition on previous iterations:

$$\mathbb{E}\Big[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})\Big| X_{i-1}, Y_{i-1}\Big] = \tag{54}$$
$$= \frac{a_i}{a_i+b_i}\underbrace{(f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}))}_{=a_i \text{ if } a_i \neq 0} + \frac{b_i}{a_i+b_i}\underbrace{(f(Y_{i-1} \cup \{u_i\}) - f(Y_{i-1}))}_{=b_i \text{ if } b_i \neq 0} = \frac{a_i^2+b_i^2}{a_i+b_i} \tag{55}$$

Now divide into two cases: $u_i \in S^*$ and $u_i \notin S^*$.

- If $u_i \notin S^*$, in particular, $u_i \notin S_{i-1}^*$:

$$\mathbb{E}\big[f(S_{i-1}^*) - f(S_i^*)\big] = \frac{a_i}{a_i+b_i}\big(f(S_{i-1}^*) - f(S_{i-1}^* \cup \{u_i\})\big) \overset{S_{i-1}^* \subseteq Y_{i-1} \setminus \{u_i\}}{\leq} \tag{56}$$
$$\leq \frac{a_i}{a_i+b_i}\big(f(Y_{i-1}^* \setminus \{u_i\}) - f(Y_{i-1}^*)\big) \leq \frac{a_i b_i}{a_i+b_i} \tag{57}$$

- If $u_i \in S^*$, in particular, $u_i \in S_{i-1}^*$:

$$\mathbb{E}\big[f(S_{i-1}^*) - f(S_i^*)\big] = \frac{b_i}{a_i+b_i}\big(f(S_{i-1}^*) - f(S_{i-1}^* \setminus \{u_i\})\big) \overset{X_{i-1} \subseteq S_{i-1}^* \setminus \{u_i\}}{\leq} \tag{58}$$
$$\leq \frac{b_i}{a_i+b_i}\big(f(X_{i-1}^* \cup \{u_i\}) - f(X_{i-1}^*)\big) \leq \frac{a_i b_i}{a_i+b_i} \tag{59}$$

And since $a_i^2 - 2a_ib_i + b_i^2 = (a_i - b_i)^2 \geq 0$ (and by tower property), we get the required. □

Theorem 4.7 ([Buchbinder et al., 2015])**.** Algorithm 4 on the previous page is $\frac{1}{2}$ approximation in expectation.

Proof. Denote

$$S_{alg} = S_n^* = X_n = Y_n \tag{60}$$

Then

$$\mathbb{E}\Big[f(S_0^*) - f(S_n^*)\Big] \leq \frac{1}{2}\mathbb{E}\Big[f(X_n) - f(X_0) + f(Y_n) - f(Y_0)\Big] \tag{61}$$

$$\mathbb{E}\Big[f(S^*) - f(S_{alg})\Big] \leq \frac{1}{2}\mathbb{E}\Big[2S_{alg} - f(X_0) - f(Y_0)\Big] \overset{f(S)\geq 0}{\leq} \mathbb{E}\Big[S_{alg}\Big] \tag{62}$$

Thus

$$\mathbb{E}[S_{alg}] \geq \frac{1}{2}\mathbb{E}[f(S^*)] \tag{63}$$

□

Collary 4.7.1. Algorithm 3 on page 6 is $\frac{1}{3}$ approximation.

Note Algorithms 3 and 4 on page 6 run in $\mathcal{O}(N)$ time.

5 Knapsack constraints

Let each element of set have price c_i and budget B, then

$$\max f(S) \tag{64}$$

$$\text{s.t.} \sum_{i\in S} c_i \leq B \tag{65}$$

Algorithm 5

```
1: procedure DENSITY GREEDY($N$)
2:     $S \leftarrow \emptyset$
3:     while $N \neq \emptyset$ do
4:         $x^* \leftarrow \arg\max \left\{ \frac{f(S\cup\{x\})-f(S)}{c_i} \right\}$
5:         if $c(S) + c_{x^*} \leq B$ then
6:             $S \leftarrow S \cup \{x^*\}$
7:         end if
8:         $N \leftarrow N \setminus \{x^*\}$
9:     end while
10:    return $S$
11: end procedure
```

Note that this is generalization of cardinality constraint.

Algorithm 6

```
1: procedure OPTIMIZED DENSITY GREEDY($N$)
2:     $S_1 \leftarrow$ output of Algorithm 5
3:     $S_2 \leftarrow \left\{ \arg\max_{i\in N, c_i \leq B} f(i) \right\}$
4:     return $\arg\max_{S\in\{S_1,S_2\}} f(S)$
5: end procedure
```

Proposition 5.1 ([Khuller et al., 1999]). Algorithm 6 is $\frac{1}{2}\left(1-\frac{1}{e}\right)$-optimal.

Proposition 5.2. Algorithm 6 is $\left(1-\frac{1}{\sqrt{e}}\right)$-optimal.

Theorem 5.3 ([Khuller et al., 1999, Sviridenko, 2004]). If a set of l most dense items in optimal solution S^*, it is possible to get good approximation to the optimal solution.
Enumerating all sets of up to 3 most dense items in optimal solution S^*, we can acquire $1-\frac{1}{e}$-approximation of optimal solution. Since cardinality constraint is a particular case of knapsack constraint, this is best polynomial approximation.

6 Introduction to matroids

Matroid is a basic concept in combinatorial optimization. It was first defined by Whitney [1935].

Definition 6.1 (matroid)**.** Matroid $\mathcal{M}$ is a pair $(E, \mathcal{I})$. E is is a finite set (called the ground set) and $\mathcal{I} \neq \emptyset$ is a family of subsets of E (called the independent sets) with the following properties:

1. If $Y \in \mathcal{I}$ then for all $X \subseteq Y$, $X \in \mathcal{I}$.
2. If $X, Y \in \mathcal{I}$ and $|Y| > |X|$, then exists $e \in Y \setminus X$, $X \cup \{e\} \int \mathcal{I}$.

Notes All maximal independent sets have same size. Those sets are called basis.

Examples

Uniform manifold

$$\mathcal{M}_k = \left(E, \left\{X \subseteq E \middle| |X| \leq k\right\}\right) \tag{66}$$

Linear manifold Let $A \in \mathbb{R}^{m \times n}$ be a matrix. Let E be a set of columns of A. The set $X \subseteq E$ is independent if its elements are independent. Alternatively, for sub-matrix A_X consisting of columns of A:

$$\mathcal{I} = \{X \subset E | \operatorname{rank}(A_x) = |X|\} \tag{67}$$

Graphic matroids Let $G = (V_G, E_G)$ be a graph, $E = E_G$ and

$$\mathcal{I} = \{X \subseteq E_G | X \text{ is forest}\} \tag{68}$$

Proposition 6.1. $M = (E_G, \mathcal{I})$ is matroid.

The basis is then spanning trees (or forests if graph is not connected).

Partition matroid For a set E let $E_1, \ldots E_k$ be some partition of E. Then

$$\mathcal{I} = \left\{X \subseteq E \middle| \forall i = 1..k \, |X| \cup E_i \leq 1\right\} \tag{69}$$

Proposition 6.2. $M = (E, \mathcal{I})$ is matroid.

Note that partition matroid encodes constraints of submodular welfare problem.
Constraint of matching in the bipartite graph can be defined as intersection of two partition matroids.

Definition 6.2 (Circuit)**.** Circuit in matroid $M = (E, \mathcal{I})$ is a dependent set X such that $X \notin \mathcal{I}$ and for all $x \in X$ $X \setminus \{x\} \in \mathcal{I}$.

Definition 6.3 (Rank function)**.** For matroid $M = (E, \mathcal{I})$ rank function $r : 2^{\mathbb{E}} \to \mathbb{N}$ is defined as

$$r(x) = \max\left\{|Y| \middle| Y \subset X, Y \in \mathcal{I}\right\} \tag{70}$$

Definition 6.4 (Rank of matroid)**.** For matroid $M = (E, \mathcal{I})$ rank of matroid is $\operatorname{rank}(E)$.

Proposition 6.3. Rank of matroid is submodular function.

Algorithm 7

```
1: procedure GREEDY($E$, $I$)
2:     $S \leftarrow \emptyset$
3:     for $e \in E$ do
4:         if $S \cup \{x\} \in \mathcal{I}$ then
5:             $S \leftarrow S \cup \{x\}$
6:         end if
7:     end for
8:     return $S$
9: end procedure
```

Proposition 6.4. Algorithm 7 returns basis of E.

Proof. Assume S is not a basis and let B be a basis. Exists $x \in B \setminus S$ such that $S \cup \{x\}$ is independent. However, since we have not added x to S, it got to be dependent with S. □

Question Given matroid over E (via independence oracle), let weight function $w : E \to \mathbb{R}$ and weight of set be $w(X) = \sum_{x \in X} w(x)$. We want to find independent set (pr basis) of maximal weight.

Algorithm 8

```
1: procedure GREEDY($E$, $I$)
2:     $S \leftarrow \emptyset$
3:     for $e \in E$ from heaviest to lightest do
4:         if $S \cup \{x\} \in \mathcal{I}$ then
5:             $S \leftarrow S \cup \{x\}$
6:         end if
7:     end for
8:     return $S$
9: end procedure
```

Proposition 6.5. Algorithm 8 solves the problem of maximal weight basis.

Proof. We know that for $k = \text{rank}(M)$, the size of the output of algorithm is k and so is size of optimal solution S^*. Lets assume S is not optimal, thus exists i such that $w(e_i^*) > w(e_i)$.
At iteration at which we added e_i to S. At this iteration $|S| = i - 1$. take a look at first i elements of S^*: this is independent set, and from definition of matroid, exists e' such that S is independent with e'. However, $w(e') \geq w(e^* - i) > w(e_i)$, thus we should have added it beforehand. □

7 Continuous extensions of submodular functions

Definition 7.1 (Continuous extensions of function). For $f : 2^N \to \mathbb{R}$ extension of f is $F : [0,1]^N \to \mathbb{E}$ such that for all $S \in N$ (denote by $\mathbb{1}_S \in \{0,1\}^N$ indicator of S) $F(\mathbb{1}_S) = f(S)$.

There exist many extensions of submodular functions. In particular, there exist convex and concave extensions of submodular functions.

Proposition 7.1. Evaluating concave extensions of submodular function in some point is NP-hard.

References

Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1433–1452. Society for Industrial and Applied Mathematics, 2014. (cited on pp. 4 and 5)

Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 44(5):1384–1402, 2015. (cited on p. 7)

Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998. (cited on p. 3)

Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011. (cited on p. 6)

Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information processing letters*, 70(1):39–45, 1999. (cited on p. 8)

George L Nemhauser and Laurence A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978. (cited on p. 3)

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions–i. *Mathematical programming*, 14(1):265–294, 1978. (cited on p. 3)

Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004. (cited on p. 8)

Hassler Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57(3):509–533, 1935. (cited on p. 9)