236621 - Algorithms for Submodular Optimization

Roy Schwartz

March 30, 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

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

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:

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

. It generalizes MAX-K-COVER.

Submodular welfare

3 Maximization of the submodular function with cardinality constraints

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

Algorithm 1 Nemhauser-Wolsey-Fisher

```
procedure GREEDY(N)
    A ← ∅
    for i = 1 to k do
        Let u_i ∈ N maximize f(A_{i-1} ∪ {u_i}) − f(A_{i-1})
        A_i ← A_{i-1} ∪ {u_i}
    end for
    return A_k
end procedure
```

Greedy algorithm If f is monotonic, there exist an optimal 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 3.1 (?). 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 3.2 (?). For all constant $\epsilon > 0$ each algorithm acquiring $1 - \frac{1}{e} + \epsilon$ requires exponential number of requests to value oracle.

Theorem 3.3 (?). 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 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,
What happens if f is not monotonic? First of all, does greed algorithm work? Not only it is not optimal, it can be as bad as $\frac{2}{N}$. The idea is to randomize algorithm to prevent it from "bad" choices.

Greedy algorithm If f is monotonic, there exist an optimal algorithm.

Proposition 3.4. Given set S and set A such that each element is in A with probability less than p

$$\mathbb{E}[f(S \cup A)] \geq (1 - p) f(S) \tag{24}$$

Theorem 3.5 (?). In monotonic case, Algorithm 2 is $1 - \frac{1}{e}$ optimal in expectation.

Algorithm 2

procedure RANDOMIZED GREEDY(N)
  $A \leftarrow \emptyset$
  for $i = 1$ to k **do**
    $M_i \leftarrow \arg\max_{B \subseteq N \,:\, |B| \leq k} \sum_{u \in B} f(A_{i-1} \cup \{u\}) - f(A_{i-1})$
    $A_i \leftarrow \begin{cases} A_{i-1} \cup \{u\} & \forall u \in M_i \text{ with } P = \frac{1}{k} \\ A_{i-1} & \text{with } P = 1 - \frac{|M_i|}{k} \end{cases}$
  end for
  return A_k
end procedure

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

$$\mathbb{E}[f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) | A_{i-1}] = \frac{1}{k} \sum_{u_i \in M_i} f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) \geq \frac{1}{k} \sum_{u_i \in S^*} f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) \geq \tag{25}$$

$$\geq \frac{1}{k}(f(S^*) - f(A_{i-1})) \tag{26}$$

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

$$\mathbb{E}[f(A_{i-1} \cup \{u_i\}) - f(A_{i-1})] \geq \frac{1}{k}(f(S^*) - \mathbb{E}[f(A_{i-1})]) \tag{27}$$

And thus we can once again solve the recurrence and acquire same result as in Proposition 3.1. □