Notes on Greedy Algorithms for Submodular Maximization

Thibaut Horel

February 26, 2015

1 Submodular Functions

All the functions we consider are set functions defined over subsets of a ground set N.

Definition 1. A function $f : 2^N \to \mathbf{R}$ is *monotone* iff:

$$\forall S \subseteq T \subseteq N,\ f(S) \leq f(T)$$

Definition 2. For $f : 2^N \to \mathbf{R}$ and $S \subseteq N$, the *marginal contribution* to S is the function f_S defined by:

$$\forall T \subseteq N,\ f_S(T) = f(S \cup T) - f(S)$$

When there is no ambiguity, we write $f_S(e)$ instead of $f_S(\{e\})$ for $e \in N$, $S + e$ instead of $S \cup \{e\}$ and $S - e$ instead of $S \setminus \{e\}$.

Definition 3. A function $f : 2^N \to \mathbf{R}$ is *submodular* iff:

$$\forall S \subseteq T \subseteq N, \forall e \in N \setminus T,\ f_T(e) \leq f_S(e)$$

This "decreasing marginal contribution" definition of submodular functions often leads to treating them as "discrete concave functions".

Proposition 4. *The following statements are equivalent:*

1. *f is submodular.*
2. *for all $S \subseteq N$, f_S is submodular.*
3. *for all $S \subseteq N$, f_S is subadditive.*

Proof. (*1.* $\Rightarrow$ *2.*) is immediate. To prove (*2.* $\Rightarrow$ *3.*), we show that any submodular function f is subadditive. Let f be a submodular function. Consider A and B two subets of N. Writing $B = \{e_1, \dots, e_n\}$ and $B_i = \{e_1, \dots, e_i\}$:

$$\begin{aligned} f(A \cup B) = f(A) &+ \sum_{i=1}^n f(A \cup B_i) - f(A \cup B_{i-1}) \\ &\leq f(A) + \sum_{i=1}^n f(B_i) - f(B_{i-1}) = f(A) + f(B) \end{aligned}$$

where the inequality uses the submodularity of f.

Finally, we prove (*3.* $\Rightarrow$ *1.*). Let f be a function satisfying *3.*, and let us consider $S \subseteq T \subseteq N$ and $e \in N \setminus T$. Writing $T' = T \setminus S$:

$$\begin{aligned} f_T(e) &= f_S(T' + e) - f_S(T') \\ &\leq f_S(T') + f_S(e) - f_S(T') = f_S(e) \end{aligned}$$

where the inequality used that f_S is subadditive. $\square$

Remark. Proposition 4 implies in particular that a submodular function is subadditive.

The following corollary will be very useful in analysing greedy algorithms involving submodular functions. It can be seen as the "integrated" version of Definition 3[1].

Corollary 5. *Let f be a submodular function, then:*

$$\forall S \subseteq T \subseteq N,\ f(T) \leq f(S) + \sum_{e \in T \setminus S} f_S(e)$$

Furthermore, if f is monotone submodular, S need not be a subset of T:

$$\forall S \subseteq N, T \subseteq N,\ f(T) \leq f(S) + \sum_{e \in T \setminus S} f_S(e)$$

Proof. If f is submodular, using that f_S is subadditive:

$$f_S(T) = f_S(T \setminus S) \leq \sum_{e \in T \setminus S} f_S(e)$$

which proves the first part of the corollary.

If f is monotone submodular, $f(T) \leq f(S \cup T)$ and applying the first part of the corollary to $S \cup T$ and T concludes the proof. $\square$

Remark. The two inequalities of Corollary 5 can be proved to be respectively equivalent to "f is submodular" and "f is monotone submodular".

2 Cardinality Constraint

Henceforth, f will be a monotone submodular function. Furthermore, we assume that f is *normalized*, that is, $f(\emptyset) = 0$. Consider the following maximization program:

$$S^* \in \arg\max_{S\,:\,|S| \leq K} f(S)$$

The choice of a representation for f has a big impact on the computational nature of the above program. We assume the *value query model*: for any $S \subseteq N$, the algorithm can query a black-box oracle for the value $f(S)$. An algorithm making $O\big(\text{poly}(|N|)\big)$ queries to the oracle is considered to have polynomial running time.

Proposition 6. *Let S_G be the set returned by Algorithm 1, then $f(S_G) \geq \big(1 - \frac{1}{e}\big) f(S^*)$.*

Proof. Let us denote by $S_i = \{e_1, \dots, e_i\}$, the value of S_G after the ith time line 4 of Algorithm 1 is executed. Then:

$$\begin{aligned} f(S^*) &\leq f(S_{i-1}) + \sum_{e \in S^* \setminus S_{i-1}} f_{S_{i-1}}(e) \\ &\leq f(S_{i-1}) + \sum_{e \in S^* \setminus S_{i-1}} f(S_i) - f(S_{i-1}) \\ &\leq f(S_{i-1}) + K\big(f(S_i) - f(S_{i-1})\big) \end{aligned}$$

[1] Note the analogy to $f(b) \leq f(a) + (b-a)f'(a)$, for f concave.

Algorithm 1 Greedy (Cardinality constraint)

```
Input: N, K, value query oracle for f
S_G ← ∅
while |S_G| < K do
    x* ← arg max_{x∈N} f_{S_G}(x)
    S_G ← S_G + x*
    N ← N − x*
end while
return S_G
```

where the first inequality used Corollary 5, the second inequality used the greediness of Algorithm 1 and the third inequality used that $|S^*| \leq K$.

Subtracting $K \cdot f(S^*)$ both sides gives:

$$f(S_i) - f(S^*) \geq \frac{K-1}{K}\big(f(S_{i-1}) - f(S^*)\big)$$

which in turn implies by induction:

$$f(S_i) \geq \left(1 - \left(1 - \frac{1}{K}\right)^i\right) f(S^*)$$

Taking $i = K$ and using $(1 - \frac{1}{K})^K \leq \frac{1}{e}$ concludes the proof. □

Remark. Feige [1998] proved that unless P = NP, no polynomial time algorithm can achieve an approximation ratio better than $1 - \frac{1}{e}$ for the cardinality constrained maximization of set cover functions.

3 Knapsack Constraint

There is now a cost function $c : N \to \mathbf{R}^+$ and a budget $B \in \mathbf{R}^+$. c is extended to 2^N by $c(S) = \sum_{e \in S} c(e)$. Consider the following Knapsack constrained optimization problem:

$$S^* \in \underset{S\,:\,c(S) \leq B}{\arg\max}\ f(S)$$

A natural way to extend Algorithm 1 to this case is Algorithm 2. The two main differences are that:

1. instead of maximizing the marginal contribution at each time step, Algorithm 2 maximizes the "bang-per-buck": the marginal contribution divided by the cost.

2. when adding an item would violate the budget constraint, the item is thrown away, and the iteration keeps inspecting possibly cheaper items.

Unfortunately, Algorithm 2 has unbounded approximation ratio. Similarly to the standard Knapsack problem (when f is additive), problems arise when there are high value items. Consider the case where f is additive and $N = \{e_1, e_2\}$ with $f(e_1) = v$, $f(e_2) = \varepsilon v$, $c(e_1) = B$ and $c(e_2) = \frac{\varepsilon B}{2}$. The best solution is clearly to pick $\{e_1\}$ for a value of v. In contrast, Algorithm 2 picks $\{e_2\}$ for a value of εv. As ε gets close to zero, the approximation ratio becomes arbitrarily large.

However, the following lemma will be useful to use Algorithm 2 as a building block for algorithms solving the Knapsack constrained submodular maximization.

Algorithm 2 Greedy (Knapsack constraint)

```
Input: N, B, value query oracle f, cost function c
1: S_G ← ∅
2: while N ≠ ∅ do
3:     x* ← arg max_{x∈N} f_{S_G}(x) / c(x)
4:     if c(S_G) + c(x*) ≤ B then
5:         S_G ← S_G + x*
6:     end if
7:     N ← N − x*
8: end while
9: return S_G
```

Lemma 7. *Whenever line 4 of Algorithm 2 evaluates to False,* $f(S_G + x^*) \geq \big(1 - \frac{1}{e}\big) f(S^*)$.

Proof. Let us denote by $S_i = \{e_1, \dots, e_i\}$, the value of S_G after the ith time line 5 of Algorithm 2 is executed. Then:

$$\begin{aligned}
f(S^*) &\leq f(S_{i-1}) + \sum_{e \in S^* \setminus S_{i-1}} f_{S_{i-1}}(e) \\
&= f(S_{i-1}) + \sum_{e \in S^* \setminus S_{i-1}} c(e) \frac{f_{S_{i-1}}(e)}{c(e)} \\
&\leq f(S_{i-1}) + \frac{f(S_i) - f(S_{i-1})}{c(e_i)} \sum_{e \in S^* \setminus S_{i-1}} c(e) \\
&\leq f(S_{i-1}) + \frac{B}{c(e_i)}\big(f(S_i) - f(S_{i-1})\big)
\end{aligned}$$

where the first inequality used Corollary 5, the second inequality used the greediness of Algorithm 2 and the last inequality used that $c(S^*) \leq B$.

Subtracting $\frac{B}{c_i}$ both sides and reordering the terms:

$$f(S_i) - f(S^*) \geq \left(1 - \frac{c(e_i)}{B}\right)\big(f(S_{i-1}) - f(S^*)\big)$$

Solving this recursive inequality yields:

$$f(S_i) \geq \left(1 - \prod_{k=1}^{i}\left(1 - \frac{c(e_i)}{B}\right)\right) f(S^*)$$

Finally, using that $1 - x \leq e^{-x}$:

$$f(S_i) \geq \left(1 - \exp\frac{-c(S_i)}{B}\right) f(S^*)$$

We are now ready to prove the lemma. Let us consider S_G at some iteration of Algorithm 2 where line 5 evaluates to false. The above analysis didn't assume that line 5 evaluated to True when element e_i was added to S_G, hence we can apply it to $S_G + x^*$:

$$f(S_G + x^*) \geq \left(1 - \exp\frac{-c(S_G) - c(x^*)}{B}\right) f(S^*)$$

and using that $c(S_G) + c(x^*) > B$ by assumption of Lemma 7 concludes the proof. □

We now present two algorithms which exploit Lemma 7 to obtain a constant approximation ratio to the optimal solution.

Algorithm 3 Greedy (Knapsack constraint), simple fix

Input: N, B, value query oracle f, cost function c

$e^* \leftarrow \arg\max_{e \in N,\, c(e) \leq B} f(e)$
$S_G \leftarrow$ result of Algorithm 2
return $\arg\max\{f(S_G), f(e^*)\}$

Proposition 8. *Let S be the set returned by Algorithm 3, then $f(S) \geq \frac{1}{2}\left(1 - \frac{1}{e}\right) f(S^*)$.*

Proof. Let us consider the value of S_G the first time line 4 of Algorithm 2 evaluated to False after the last element of S_G was added:

$$\begin{aligned} 2f(S) &\geq f(S_G) + f(e^*) \geq f(S_G) + f(x^*) \\ &\geq f(S_G + x^*) \geq \left(1 - \frac{1}{e}\right) f(S^*) \end{aligned}$$

where the first inequality used the definition of S, the second inequality used the definition of e^*, the third inequality used the subadditivity of f and the last inequality used Lemma 7. □

Remark. Khuller et al. [1999] noted that the above analysis can be refined to show that the approximation ratio of Algorithm 3 is at least $1 - \frac{1}{\sqrt{e}}$.

Algorithm 4 Greedy (Knapsack constraint), partial enumeration

Input: N, B, value query oracle f, cost function c

1: $S_1 \leftarrow \arg\max_{S \subseteq N,\, c(S) \leq B,\, |S| < d} f(S)$
2: $S_2 \leftarrow \emptyset$
3: **for all** $S \subseteq N$, $|S| = d, c(S) \leq B$ **do**
4: $\quad N' \leftarrow N \setminus S$
5: $\quad S_G \leftarrow$ Algorithm 2 for N' and initialization $S_G \leftarrow S$
6: $\quad$**if** $f(S_G) > f(S_2)$ **then**
7: $\quad\quad S_2 \leftarrow S_G$
8: $\quad$**end if**
9: **end for**
10: **return** $\arg\max\{f(S_1), f(S_2)\}$

For some constant d (fixed later in the analysis), Algorithm 4 first compute S_1, the set of maximum value among all sets of at most $d - 1$ elements. Then for all sets of S of d elements, the algorithm completes S greedily using Algorithm 2 where the initialisation $S_G \leftarrow \emptyset$ is replaced by $S_G \leftarrow S$. The best set obtained by such a greedy completion is S_2. Algorithm 4 then returns the best of S_1 and S_2.

Proposition 9. *For $d = 3$, let S be the set returned by Algorithm 4, then $f(S) \geq \left(1 - \frac{1}{e}\right) f(S^*)$.*

Proof. *Wlog*, assume that $|S^*| > d$, otherwise Algorithm 4 finds the optimal solution. Let us write $S^* = \{e_1^*, \dots e_n^*\}$ and $S_i^* = \{e_1^*, \dots, e_i^*\}$ where the elements of S^* where ordered such that:

$$e_i^* \in \arg\max_{e \in S^* \setminus S_{i-1}^*} f_{S_{i-1}^*}(e)$$

Let us now consider the iteration of Algorithm 4 where $S = S_d^*$. Then line 5 is equivalent to running Algorithm 2 for the function $f_{S_d^*}$ and set $N \setminus S_d^*$. Let us consider the first time line 4 of Algorithm 2 evaluated to false for some element x^* of $S^* \setminus S_d^*$[2]. Then by Lemma 7:

$$f(S_G + x^*) - f(S_d^*) \geq \left(1 - \frac{1}{e}\right)\left(f(S^*) - f(S_d^*)\right) \tag{1}$$

But by submodularity of f and the ordering of S_d^*:

$$f_{S_G}(x^*) \leq f_{S_i^*}(x^*) \leq f(S_i^*) - f(S_{i-1}^*), \quad 1 \leq i \leq d$$

Summing for $1 \leq i \leq d$:

$$f(S_G + x^*) \leq f(S_G) + \frac{1}{d} f(S_d^*) \tag{2}$$

Combining Equations (1) and (2) gives

$$f(S_G) \geq \left(1 - \frac{1}{e}\right) f(S^*) + \left(\frac{1}{e} - \frac{1}{d}\right) f(S_d^*)$$

which concludes the proof after observing that $\frac{1}{e} - \frac{1}{d} > 0$ for $d = 3$. □

4 Matroid Constraint

Definition 10. A *matroid* M is a pair $(N, \mathcal{I})$. N is a finite set called the *ground set* and $\mathcal{I}$ is family of subsets of N called the *independent sets* such that:

1. (downward closure) if $B \in \mathcal{I}$ and $A \subseteq B$ then $A \in \mathcal{I}$
2. (exchange property) if $A \in \mathcal{I}, B \in \mathcal{I}$ and $|A| < |B|$ then there exists $x \in B \setminus A$ such that $A + x \in \mathcal{I}$

Maximal independent sets of M are called *bases*.

Remark. It follows from the exchange property that all bases have the same cardinality. This cardinality is the *rank* of M.

Proposition 11 (bijective basis exchange)**.** *If B_1 and B_2 are two bases of a matroid M, then there exists a bijection $\phi : B_1 \setminus B_2 \to B_2 \setminus B_1$ such that:*

$$\forall x \in B_1 \setminus B_2,\ B_1 - x + \phi(x) \in \mathcal{I}$$

Proof. This is a standard result in matroid theory. See for example Corollary 39.12a in Schrijver [2003]. □

Remark. The bijection ϕ of Proposition 11 can be extended to $\phi : B_1 \to B_2$ by defining it to be the identity function on $B_1 \cap B_2$.

We now look at the problem of maximizing a monotone submodular function over a matroid $M = (N, \mathcal{I})$:

$$S^* \in \arg\max_{S \in \mathcal{I}} f(S)$$

From a computational perspective, we still assume a value query oracle for f. Furthermore, we assume an *independence oracle* for M: given $S \subseteq N$, the independence oracle tests whether or not $S \in \mathcal{I}$.

[2] This necessarily happens since all the elements in N are eventually considered in the while loop.

Algorithm 5 Greedy (Matroid constraint)

Input: N, value query oracle f, independence oracle for $\mathcal{I}$

$S_G \leftarrow \emptyset$
while $N \neq \emptyset$ **do**
  $x^* \leftarrow \arg\max_{x \in N} f_{S_G}(x)$
  if $S_G + x \in \mathcal{I}$ **then**
    $S_G \leftarrow S_G + x^*$
  end if
  $N \leftarrow N - x^*$
end while
return S_G

Remark. The set S_G constructed by Algorithm 5 is a base of M. When the rank K of M is known, the while loop can be stopped as soon as $|S_G| = K$ (cf. cardinality constrained submodular maximization).

Proposition 12. *Let S_G be the set returned by Algorithm 5, then* $f(S_G) \geq \frac{1}{2} f(S^*)$.

Proof. Wlog, assume that S^* is a base of M. Let $\phi : S^* \to S_G$ be a bijection as in Proposition 11. Le us write:

$$S^* = \{e_1^*, \dots, e_K^*\} \text{ and } S_G = \{e_1, \dots, e_K\}$$

where $e_i = \phi(e_i^*)$ and define $S_i = \{e_1, \dots, e_i\}$. Then:

$$\begin{aligned} f(S^*) - f(S_G) &\leq \sum_{i=1}^{K} f_{S_G}(e_i^*) \leq \sum_{i=1}^{K} f_{S_{i-1}}(e_i^*) \\ &\leq \sum_{i=1}^{K} f(S_i) - f(S_{i-1}) = f(S_G) \end{aligned}$$

where the first inequality used Corollary 5, the second inequality used submodularity of f, and the third inequality used the greediness of Algorithm 5 and that $S_{i-1} + e_i^* \in \mathcal{I}$ by Proposition 11. □

5 Bibliographical Notes

A systematic analysis of greedy algorithms for submodular maximization was made by Fisher et al. [1978], Nemhauser et al. [1978]. The results about submodular maximization under cardinality and matroid constraints can be found in these papers, even though some of them had already been obtained by Edmonds [1971]. The lower bound of $(1 - \frac{1}{e})$ for the approximation ratio of a polynomial time algorithm is due to Feige [1998].

For Knapsack constraints, Khuller et al. [1999] were the first to obtain an approximation ratio of $(1 - \frac{1}{e})$ using partial enumeration in the case of a set cover function. It was then noted by Sviridenko [2004], that the result extended to any submodular function.

It is possible to obtain a $(1 - \frac{1}{e})$ approximation ratio under matroid constaints. This result was first obtained by Calinescu et al. [2007] using continuous optimization. A combinatorial algorithm was later constructed by Filmus and Ward [2012].

More complex constraints can also be considered: intersection of independence systems, matroids, knapsack constraints. Nemhauser et al. [1978] summarize some results from the late 70's. A general framework to combine constraints can be found in Vondrák et al. [2011].

References

G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *Integer programming and combinatorial optimization*, pages 182–196. Springer, 2007.

J. Edmonds. Matroids and the greedy algorithm. *Mathematical programming*, 1(1):127–136, 1971.

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

Y. Filmus and J. Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. In *Foundations of Computer Science (FOCS)*, pages 659–668. IEEE, 2012.

M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. *An analysis of approximations for maximizing submodular set functions—II*. Springer, 1978.

S. Khuller, A. Moss, and J. S. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

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

J. Vondrák, C. Chekuri, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 783–792. ACM, 2011.