Advanced Algorithms December 10, 2024

Lecture 23: Submodularity and Maximization

Notes by Ola Svensson

(These notes are based on notes written by Moran Feldman and Justin Ward who were postdocs at EPFL and are now faculty at the Open University and Queens Mary University, respectively.)

In this lecture, we consider the problem of maximizing a submodular function. We use the same notation as last lecture. We assume that all functions $f : 2^N \to \mathbb{R}$ we deal with in this lecture are

- non-negative: $f(S) \geq 0$ for all $S \subseteq N$,
- and normalized: $f(\emptyset) = 0$.

Suppose also that $f : 2^N \to \mathbb{R}$ is *monotone*: $f(S) \leq f(T)$ for all $S \subseteq T \subseteq N$. Then, we must have $f(N) \geq f(S)$ for any other $S \subseteq N$, so unconstrained maximization is trivial in this case. However, the following constrained problem arises naturally in many settings: we are given a parameter k, and must find a set $S \subseteq N$ of size at most k that maximizes f. Natural applications of this setting is e.g. data summarization and influence maximization.

1 Cardinality Constrained Monotone Maximization

Recall that for weighted maximization problems with linear objective, this problem is solved optimally by the greedy algorithm. This result in fact holds even if we replace the constraint that $|S| \leq k$ by a general matroid constraint. The natural generalization of the greedy algorithm to submodular functions is as follows: at each step, we choose the element $u \notin S$ with maximal *marginal gain* $f(u \mid S)$ and add u to S.

Input: Ground set N, value oracle for monotone submodular $f : 2^N \to \mathbb{R}$, parameter $0 \leq k \leq |N|$.
Output: $S \subseteq N$ with $|S| \leq k$
$S \leftarrow \emptyset$;
for $i = 1$ **to** k **do**
  Let $u_i = \arg\max_{u \in N \setminus S} f(u \mid S)$;
  $S \leftarrow S \cup \{u_i\}$;
return S;

Notice that if $f(S) = \sum_{e \in S} w_e$ for some set of weights w, this is indeed exactly the standard greedy algorithm. Here, however, we find that the algorithm can produce a suboptimal solution. Consider a coverage function with sets:

$$T_1 = \{1, 2, 3, 4\}$$
$$T_2 = \{1, 2, 5\}$$
$$T_3 = \{3, 4, 6\}$$

Suppose $k = 2$. Then, the greedy algorithm first selects T_1 and then takes either T_2 or T_3. This covers 5 total elements. However, taking T_2 and T_3 covers 6 elements!

Despite this, we can show that the greedy algorithm gives a constant factor approximation. Let's start by proving a simple claim that holds for any submodular function:

Lemma 1 *Let $f : 2^N \to \mathbb{R}$ be a submodular function and let $S, T \subseteq N$. Then, $\sum_{e \in T \setminus S} f(e \mid S) \geq f(T \cup S) - f(S)$.*

Proof Order the elements of $T \setminus S$ as $e_1, e_2, \ldots, e_{|T \setminus S|}$ and let T_i be the set containing the first i elements in this ordering. Observe that $T_0 = \emptyset$ and $T_{|T \setminus S|} = T \setminus S$. Then,

$$\begin{aligned}
\sum_{e \in T \setminus S} f(e \mid S) &= \sum_{i=1}^{|T \setminus S|} f(e_i \mid S) \\
&\geq \sum_{i=1}^{|T \setminus S|} f(e_i \mid S \cup T_{i-1}) \\
&= \sum_{i=1}^{|T \setminus S|} f(T_i \cup S) - f(T_{i-1} \cup S) \\
&= f(T_{|T \setminus S|} \cup S) - f(T_0 \cup S) \\
&= f(T \cup S) - f(S)\,.
\end{aligned}$$

Here, we have used submodularity for the inequality, and then noted that the sum is telescoping. ■

We can now prove the following result:

Theorem 2 *Let S be the set produced by the greedy algorithm for maximizing a monotone submodular function f subject to a cardinality constraint k. Let O be any set of at most k elements. Then, $f(S) \geq (1 - 1/e) f(O) \approx 0.632 f(O)$.*

Proof Notice that since f is monotone, we can assume that $|O| = k$, since we can always add elements to it if this is not the case without decreasing its value.

Let u_i be the ith element selected by the greedy algorithm and let S_i be the set $\{u_j \ : \ j \leq i\}$ containing the first i elements selected. Note that $S_0 = \emptyset$ and $S_k = S$. Consider an iteration i and let e be any element in $O \setminus S_{i-1}$. Then, by our greedy choice we have:

$$f(u_i \mid S_{i-1}) \geq f(e \mid S_{i-1})$$

We have one such inequality for each $e \in O \setminus S_{i-1}$. Adding them all together and applying Lemma 1, we get:

$$|O \setminus S_{i-1}| \cdot f(u_i \mid S_{i-1}) \geq \sum_{e \in O \setminus S_{i-1}} f(e \mid S_{i-1}) \geq f(S_{i-1} \cup O) - f(S_{i-1}) \geq f(O) - f(S_{i-1}),$$

where the last inequality follows from the fact that f is monotone. Now, note that $|O \setminus S_{i-1}| \leq k$ and also $f(u_i \mid S_{i-1}) \geq 0$ since f is monotone. Thus,

$$k \cdot f(u_i \mid S_{i-1}) \geq |O \setminus S_{i-1}| \cdot f(u_i \mid S_{i-1}) \geq f(O) - f(S_{i-1})\,.$$

Dividing by k, and recalling the definition of S_{i-1} and S_i, we get:

$$f(S_i) - f(S_{i-1}) = f(u_i \mid S_{i-1}) \geq \tfrac{1}{k} f(O) - \tfrac{1}{k} f(S_{i-1}).$$

Let's rearrange this to:

$$f(S_i) \geq \left(1 - \tfrac{1}{k}\right) f(S_{i-1}) + \tfrac{1}{k} f(O). \tag{1}$$

Notice that this gives a recurrence for $f(S_i)$. We now show by induction on i.

$$f(S_i) \geq \left(1 - \left(1 - \tfrac{1}{k}\right)^i\right) f(O) \tag{2}$$

For $i = 0$, we have $f(S_0) = f(\emptyset) = 0 = (1 - (1 - \frac{1}{k})^0) f(O)$. For $i > 0$, we have:

$$\begin{aligned} f(S_i) &\geq \left(1 - \tfrac{1}{k}\right) f(S_{i-1}) + \tfrac{1}{k} f(O) \\ &\geq \left(1 - \tfrac{1}{k}\right) \left(1 - (1 - \tfrac{1}{k})^{i-1}\right) f(O) + \tfrac{1}{k} f(O) \\ &= \left(1 - \left(1 - \tfrac{1}{k}\right)^i\right) f(O), \end{aligned}$$

where we used (1) and the induction hypothesis in the first and second lines, respectively.

To complete the proof, it suffices to plug in $i = k$ in (2) and then note that $(1 - \frac{1}{k})^k \leq e^{-1}$ (since $1 + x \leq e^x$, as is seen by considering Taylor series expansion of the exponential function). ■

We briefly remark that the approximation ratio of $(1 - 1/e)$ is in fact *the best possible* for the general problem. If f is somehow given to us explicitly, it is NP-hard to do better. Similarly, if f is given as a value oracle (and so the only way to get information about it is to query values) then one can show that it is impossible to do better without making a super-polynomial number of value queries. This latter result is information-theoretic and holds *even if* $P = NP$.

2 Unconstrained Submodular Maximization

Now, suppose that f is not monotone. Then, even without constraints, it makes sense to ask what set S maximizes $f(S)$. One idea is to just run the greedy algorithm, and stop when no element gives any positive marginal gain (i.e. when $f(e \,|\, S) \leq 0$ for all $e \in N \setminus S$.

However, this algorithm performs very badly on the following example function. Let $N = \{u_1, u_2, \ldots u_n, v\}$. Then, define:

$$f(S) = \begin{cases} 2 & \text{if } v \in S \\ |S| & \text{if } v \notin S \end{cases}$$

One can verify that f is indeed submodular. The optimal solution chooses $S = \{u_1, u_2, \ldots u_n\}$ and has value n, but the greedy algorithm would choose $\{v\}$, and of value 2.

2.1 A greedy deterministic $\frac{1}{3}$-approximation algorithm

So we need a better algorithm. We still use a greedy approach, but one which is "greedy from two sides" (one side being $\emptyset$ and the other being N):

Input: ground set N, value oracle for submodular function $f : 2^N \to \mathbb{R}$
Choose an arbitrary order $u_1, u_2, \ldots u_n$ on the elements.;
$X_0 \leftarrow \emptyset$;
$Y_0 \leftarrow N$;
for $i = 1$ **to** n **do**
  Let $a_i = f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})$;
  Let $b_i = f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1})$;
  if $a_i \geq b_i$ **then**
    $X_i \leftarrow X_{i-1} \cup \{u_i\}$ and $Y_i \leftarrow Y_{i-1}$
  else
    $X_i \leftarrow X_{i-1}$ and $Y_i \leftarrow Y_{i-1} \setminus \{u_i\}$
return X_n (which is equal to Y_n).

Theorem 3 *The above algorithm is a $\frac{1}{3}$-approximation for unconstrained submodular maximization.*

Before proving the theorem, we need some lemmas:

Lemma 4 *For every* $1 \leq i \leq n$, $a_i + b_i \geq 0$.

Proof Notice that for all i, we have $X_{i-1} \subseteq Y_{i-1}$ and also $u_i \in Y_{i-1}$. Thus, we have:

$$(X_{i-1} \cup \{u_i\}) \cap (Y_{i-1} \setminus \{u_i\}) = X_{i-1}$$
$$(X_{i-1} \cup \{u_i\}) \cup (Y_{i-1} \setminus \{u_i\}) = Y_{i-1}\,.$$

We now apply submodularity:

$$\begin{aligned} a_i + b_i &= f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}) + f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1}) \\ &= f(X_{i-1} \cup \{u_i\}) + f(Y_{i-1} \setminus \{u_i\}) - [f(X_{i-1}) + f(Y_{i-1})] \\ &\geq 0\,. \end{aligned}$$

■

Let OPT be the optimal solution of the problem, and define:

$$OPT_i = (OPT \cup X_i) \cap Y_i.$$

Then, note that OPT_i is obtained from OPT by adding all of the elements added to X_i by the algorithm and removing all of the elements removed from Y_i by the algorithm. The idea of this definition is that we transform OPT into a set that agrees with what the algorithm has already done. We show the following.

Lemma 5 *At each step i:*

$$[f(X_i) + f(Y_i)] - [f(X_{i-1}) + f(Y_{i-1})] \geq f(OPT_{i-1}) - f(OPT_i)$$

Proof We prove the lemma in the case that $a_i \geq b_i$. The other case is similar. In this case, by Lemma 4 we must have $a_i \geq 0$. The algorithm sets $X_i = X_{i-1} \cup \{u_i\}$, and $Y_i = Y_{i-1}$, so the left-hand side of the inequality is just $f(u_i \,|\, X_{i-1}) = a_i \geq 0$. We also have $OPT_i = OPT_{i-1} \cup \{u_i\}$. If $u_i \in OPT$, then we have $OPT_i = OPT_{i-1}$, so the right-hand side is 0 and we are done. Suppose that $u_i \notin OPT$. Then, note that $OPT_{i-1} \subseteq Y_{i-1} \setminus \{u_i\}$. Thus, by submodularity:

$$\begin{aligned} f(OPT_i) - f(OPT_{i-1}) = f(u_i \,|\, OPT_{i-1}) &\geq f(u_i \,|\, (Y_{i-1} \setminus \{u_i\})) \\ &= f(Y_{i-1}) - f(Y_{i-1} \setminus \{u_i\}) = -b_i \geq -a_i. \end{aligned}$$

multiplying by -1 then gives

$$f(OPT_{i-1}) - f(OPT_i) \leq a_i.$$

as required. ■

Now we have everything we need to prove Theorem 3. Initially we have $OPT_0 = OPT$ and at the end we have $OPT_n = X_n = Y_n$. Intuitively Lemma 5 shows that each step our version OPT_i of OPT decrease in value, but this decrease is always matched by an increase in the value of either X_i or Y_i, which should contribute the value of the solution we return. To make this formal, we sum the inequality of Lemma 5 over $i = 1$ to n:

$$\sum_{i=1}^{n} [f(OPT_{i-1}) - f(OPT_i)] \leq \sum_{i=1}^{n} [f(X_i) + f(Y_i) - f(X_{i-1}) - f(Y_{i-1})]\,.$$

Simplifying, and using that f is non-negative we get:

$$f(OPT_0) - f(OPT_n) \leq f(X_n) + f(Y_n) - f(X_0) - f(Y_0) \leq f(X_n) + f(Y_n).$$

Finally, since $OPT_0 = OPT$ and $OPT_n = X_n = Y_n$ we get:

$$f(OPT) \leq 3 \cdot f(X_n)\,.$$

3 Some Remarks

- In Section 1, we saw that the greedy algorithm gives a (1-1/e)-approximation algorithm for maximizing a monotone submodular function subject to a cardinality constraint. This result has been generalized (via more complicated algorithm called continuous greedy) to achieve the same guarantee subject to *any* matroid constraint. The result appeared in

  > "Maximizing a Monotone Submodular Function Subject to a Matroid Constraint", Gruia Calinescu, Chandra Chekuri, Martin Pàl, and Jan Vondràk, SIAM J. Comput, 40(6): 1740-1766 (2011).

- In Section 2, we considered the problem of maximizing a general (non-monotone) submodular function. The best known (and tight) result is a randomized version of the algorithm we presented. It achieves an approximation guarantee of 1/2. This result appeared in

  > "A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization", Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz, SIAM J. Comput. 44(5): 1384-1402 (2015).

  It is still an open problem to design tight algorithms for maximizing a non-monotone submodular function subject to a matroid (or even cardinality) constraint.