Approximation Algorithms **Spring 2019**

Lecture 4: Submodular Maximization Part 2

Scribe: Antares Chen 3/6/2019

We conclude our discussion of maximizing submodular functions with the double greedy algorithm for unconstrained submodular maximization due to Buchbinder, Feldman, Naor, and Schwartz [2].

4.1 Unconstrained Submodular Maximization

Previously, we provided two examples for problems that are modeled by maximizing a submodular function under cardinality constraints: maximizing float among bank accounts, and maximizing social influence. However, many more problems are captured when cardinality constraints and guaranteed monotonicity are not provided.

Recall the problem of maxcut. Given $G = (V, E)$ where each edge has weight $w_e \geq 0$, our goal is to choose a cut such that the total weight of edges crossing the cut is maximized. More precisely, we wish to find $S \subseteq V$ such that the following function is maximized.

$$f(S) = \sum_{e \in E(S, V-S)} w_e \tag{4.1}$$

where $E(S, V - S) = \{(u, v) \in E : u \in S \text{ and } v \in V - S\}$. The objective function for maxcut is submodular.

Claim 4.1. *For a graph $G = (V, E)$ with edge weights w_e, the function f defined in equation 4.1 is submodular.*

Proof. Recall that f is submodular if and only if for all $S, T \subseteq V$ we have that $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$. Now, a sum of submodular functions is also submodular, thus to demonstrate that $f(S)$ is submodular, we will show that it is a sum of submodular functions. Define $f_e : 2^V \to \mathbb{R}_{\geq 0}$ for each edge $e \in E$ as

$$f_e(S) = \begin{cases} w_e & \text{if } e \in E(S, V - S) \\ 0 & \text{otherwise} \end{cases}$$

Suppose $e = (u, v)$ and consider any $S, T \subseteq V$. To see that f_e is submodular, we verify three cases.

1. S, T either contain none or both of u, v. In this case we have that

$$f(S \cup T) + f(S \cap T) = 0 = f(S) + f(T)$$

2. S, T contain the same one of u, v. Without loss of generality, suppose $u \in S$ and $u \in T$. Observe that

$$f(S \cup T) + f(S \cap T) = f(\{u\}) + f(\{u\}) = 2 \cdot w_e = f(S) + f(T)$$

3. S, T contain one of but different u, v. Without loss of generality, suppose $u \in S$ and $v \in T$. Observe that

$$f(S \cup T) + f(S \cap T) = f(\{u, v\}) + f(\varnothing) = 0 \leq 2 \cdot w_e = f(\{u\}) + f(\{v\}) = f(S) + f(T)$$

In all cases, f_e satisfies submodularity. Finally, because $f(S) = \sum_{e \in E} f_e(S)$, we have that f is submodular. □

However, the maxcut objective is certainly not monotone as the edge weights may be non-zero but $f(V) = 0$.

4.2 Maximizing Non-monotone Submodular Functions

Let us precisely define the problem of maximizing a non-monotone submodular function, sometimes called *unconstrained submodular maximization*. Given $E = [n]$ a ground set of elements, we wish to choose $S \subseteq E$ such that a submodular function $f : 2^E \to \mathbb{R}_{\geq 0}$ is maximized. We now present the double greedy algorithm due to Buchbinder, Feldman, Naor, and Schwartz. The algorithm provided a simple resolution to a problem that had been studied since the 1960s. We further discuss the history of this problem in the *Concluding Remarks* section.

4.2.1 The Double Greedy Algorithm

The double greedy algorithm iterates through elements of E deciding whether or not to add an element i to a maintained solution by computing what is gained by adding i or ignoring it. More precisely, if $E = [n]$ is our ground set and $f : 2^E \to \mathbb{R}_{\geq 0}$ our submodular function, then the algorithm proceeds as follows:

Double Greedy
Given ground set E and submodular function $f : 2^E \to \mathbb{R}_{\geq 0}$, initialize $X_0 = \varnothing$. Do for $i = 1, \ldots, n$:

1. Compute a_i and r_i according to the following:

$$a_i = f(X_{i-1} \cup \{i\}) - f(X_{i-1}) \qquad r_i = f(\hat{X}_{i-1} - \{i\}) - f(\hat{X}_{i-1})$$

2. If $a_i \geq 0$ and $r_i < 0$, then update:

$$X_i = X_{i-1} \cup \{i\}$$

3. If $r_i \geq 0$ and $a_i < 0$, then update:

$$X_i = X_{i-1}$$

4. If $a_i \geq 0$ and $r_i \geq 0$, then randomly update X_i according to the following probability:

$$X_i = \begin{cases} X_{i-1} \cup \{i\} & \text{w.p. } \frac{a_i}{a_i + r_i} \\ X_{i-1} & \text{w.p. } \frac{r_i}{a_i + r_i} \end{cases}$$

This algorithm iteratively builds a solution X_i and it decides whether or not to add i based on computing a_i and r_i. These two values measure what is gained in f if i is either added to X_{i-1} or ignored.

- If one gains more by adding i, i.e. $a_i \geq 0$ while $r_i < 0$, then i should be added to X_{i-1}.
- If one gains more by ignoring i, i.e. $r_i \geq 0$ while $a_i < 0$, then i should be ignored.

But what should one do when adding and ignoring i both increase the value of the solution the algorithm maintains? Why not flip a biased coin. Add i to X_{i-1} with probability weighted by how much is gained by either action. Indeed, this is why the algorithm is called "double greedy." It greedily chooses either to add or ignore i depending on how much it benefits the cost of the maintained solution. If either choice yields benefit, then tiebreak randomly.

One may notice that there is a forth case. What if both actions decrease the value of the maintained solution, i.e $a_i < 0$ and $r_i < 0$. This is not possible! Doing something will always increase the cost of X_i. Before showing this, let us first define the set $\hat{X}_i$ as follows:

$$\hat{X}_i = X_i \cup \{i+1, \ldots, n\}$$

Think of $\hat{X}_i$ as the set of items that are in X_i along with what could potentially be added in future iterations. It is not always the case that $\hat{X}_i \neq X_i$ always, but it does hold in general that

$$\hat{X}_0 = E \qquad\qquad X_i \subseteq \hat{X}_i \qquad\qquad X_n = \hat{X}_n$$

Let's also think of how $\hat{X}_i$ changes based on whether step 2 or 3 is executed in the double greedy algorithm.

- If $a_i \geq 0$ and $r_i < 0$, then $\hat{X}_i = \hat{X}_{i-1}$ as i is added to X_{i-1}
- If $a_i < 0$ and $r_i \geq 0$, then $\hat{X}_i = \hat{X}_{i-1} - \{i\}$ as i is will never be considered for addition in the future.

We now prove that doing anything always benefits the cost of the maintained solution.

Claim 4.2. *$a_i + r_i \geq 0$ for all i.*

Proof. Notice that $X_{i-1} \subseteq \hat{X}_{i-1} - \{i\}$. By submodularity, we have that

$$f(\hat{X}_{i-1}) - f(\hat{X}_{i-1} - \{i\}) \leq f(X_{i-1} \cup \{i\}) - f(X_i)$$

However, $\hat{X}_{i-1} = (\hat{X}_{i-1} - \{i\}) \cup \{i\}$ meaning the above inequality becomes

$$f((\hat{X}_{i-1} - \{i\}) \cup \{i\}) - f(\hat{X}_{i-1} - \{i\}) \leq f(X_{i-1} \cup \{i\}) - f(X_i)$$

The LHS of that above is $-r_i$ while the RHS is a_i. Thus $-r_i \leq a_i$ which is equivalent to that required. □

4.2.2 Analysis

Since maxcut is an NP-hard problem, we cannot expect to determine the optimal set $S \subseteq E$ that maximizes a submodular f efficiently unless P = NP. Thus we seek to bound the approximation ratio of a solution produced from this algorithm. Let $\textsc{opt} \subseteq E$ be the optimal solution and define $\textsc{opt}_i$ as follows:

$$\textsc{opt}_i = X_i \cup (\textsc{opt} \cap \{i+1, \ldots, n\})$$

Consider $\textsc{opt}_i$ as a sliding window of elements from $\textsc{opt}$ to X_i since we have $\textsc{opt}_0 = \textsc{opt}$ while $\textsc{opt}_n = X_n$. Because the double-greedy algorithm is stochastic, our statements will hold in expectation. More precisely, we will require the following lemma.

Lemma 4.3. *For all $i = 1, \ldots, n$, the following holds*

$$\mathbb{E}\big[f(\textsc{opt}_{i-1}) - f(\textsc{opt}_i)\big] \leq \frac{1}{2} \cdot \mathbb{E}\big[(f(X_i) - f(X_{i-1})) + (f(\hat{X}_i) - f(\hat{X}_{i-1}))\big] \tag{4.2}$$

We defer the proof of this statement for now and show how this implies a bound on the approximation ratio.

Theorem 4.4. *The double greedy algorithm provides a $\frac{1}{2}$-approximation for non-monotone submodular maximization on expectation.*

Proof. Consider the sum $\sum_{i=1}^{n} \mathbb{E}\big[f(\textsc{opt}_{i-1}) - f(\textsc{opt}_i)\big]$ and notice lemma 4.3 implies the following bound

$$\begin{aligned}\sum_{i=1}^{n} \mathbb{E}\big[f(\textsc{opt}_{i-1}) - f(\textsc{opt}_i)\big] &\leq \frac{1}{2} \cdot \sum_{i=1}^{n} \mathbb{E}\big[(f(X_i) - f(X_{i-1})) + (f(\hat{X}_i) - f(\hat{X}_{i-1}))\big] \\ &= \frac{1}{2} \cdot \left(\sum_{i=1}^{n} \big(\mathbb{E}[f(X_i)] - \mathbb{E}[f(X_{i-1})]\big) + \sum_{i=1}^{n} \big(\mathbb{E}[f(\hat{X}_i)] - \mathbb{E}[f(\hat{X}_{i-1})]\big) \right)\end{aligned}$$

Both sums are telescoping thus we have

$$\sum_{i=1}^{n} \mathbb{E}\big[f(\textsc{opt}_{i-1}) - f(\textsc{opt}_i)\big] \leq \frac{1}{2} \cdot \big(\mathbb{E}[f(X_n)] - \mathbb{E}[f(X_0)] + \mathbb{E}[f(\hat{X}_n)] - \mathbb{E}[f(\hat{X}_0)]\big)$$

Notice that $f(X_0) = f(\varnothing) = 0$ while $f(\hat{X}_n) = f(X_n)$ implying $\mathbb{E}[f(X_n)]$ is also 0. Meanwhile, $f(\hat{X}_0) = f(E)$ meaning $\mathbb{E}[f(\hat{X}_0)]$ is a constant that is at least 0. We can conclude that

$$\begin{aligned}\frac{1}{2} \cdot \big(\mathbb{E}[f(X_n)] - \mathbb{E}[f(X_0)] + \mathbb{E}[f(\hat{X}_n)] - \mathbb{E}[f(\hat{X}_0)]\big) &\leq \frac{1}{2} \cdot \big(\mathbb{E}[f(X_n)] + \mathbb{E}[f(\hat{X}_n)]\big) \\ &= \frac{1}{2} \cdot \big(\mathbb{E}[f(X_n)] + \mathbb{E}[f(X_n)]\big) \\ &= \mathbb{E}[f(X_n)]\end{aligned}$$

However, $\sum_{i=1}^{n} \mathbb{E}\big[f(\textsc{opt}_{i-1}) - f(\textsc{opt}_i)\big]$ is telescoping and equal to

$$\sum_{i=1}^{n} \mathbb{E}\big[f(\textsc{opt}_{i-1}) - f(\textsc{opt}_i)\big] = \mathbb{E}\big[f(\textsc{opt}_0)\big] - \mathbb{E}\big[f(\textsc{opt}_n)\big] = f(\textsc{opt}) - \mathbb{E}\big[f(X_n)\big]$$

Thus we have

$$f(\textsc{opt}) - \mathbb{E}\big[f(X_n)\big] \leq \mathbb{E}[f(X_n)] \qquad \Longrightarrow \qquad \mathbb{E}[f(X_n)] \geq \frac{1}{2} \cdot f(\textsc{opt})$$

as required. □

The final step in our analysis is to demonstrate lemma 4.3. The proof of this lemma proceeds by checking that inequality 4.2 holds for every conceivable case of the algorithm. Our case work splits based on whether or not $i \in \text{OPT}$, $a_i \geq 0$ and $r_i \geq 0$ for every iteration i. As a broad overview of the proof, notice for the cases where $a_i \geq 0$ and $r_i < 0$ as well as $a_i < 0$ and $r_i \geq 0$ the element i is added / ignored deterministically, thus we need not concern ourselves with computing an expectation. Additionally, we would have that $a_i \geq r_i$ or $r_i \geq a_i$ respectively. The brunt of the proof is thus use submodularity to appropriately relate the left and right side of inequality 4.2 to either a_i or r_i then use the fact that either $a_i \geq r_i$ or $r_i \geq a_i$ to conclude the inequality.

In the case where we need to flip a biased coin, we fall back to the first two cases with a certain weighted probability. Consequently, the expectation that we need to compute recycles the same computation performed in the first two cases of our analysis. We will provide a sketch of the argument for the case when $i \notin \text{OPT}$. The analysis for $i \in \text{OPT}$ is similar.

Proof sketch of lemma 4.3. Assume that $i \notin \text{OPT}$. We verify inequality 4.2 for the following cases.

1. Suppose $a_i \geq 0$ and $r_i < 0$: element i is added to the maintained solution X_{i-1} deterministically. Since $i \notin \text{OPT}$, we have that

   $$\text{OPT}_{i-1} = X_{i-1} \cup (\text{OPT} \cap \{i, \ldots, n\}) \subseteq (X_i \cup \{i+1, \ldots, n\}) - \{i\} = \hat{X}_i - \{i\}$$

   thus by submodularity of f, we can conclude that

   $$f((\hat{X}_{i-1} - \{i\}) \cup \{i\}) - f(\hat{X}_{i-1} - \{i\}) \leq f(\text{OPT}_{i-1} \cup \{i\}) - f(\text{OPT}_{i-1})$$

   Now, there are a couple things to notice about this inequality. Focusing on the LHS, we have that $\text{OPT}_{i-1} \cup \{i\} = \text{OPT}_i$ because i is added to X_{i-1}. Next consider the RHS. The set $(\hat{X}_{i-1} - \{i\}) \cup \{i\}$ is just $\hat{X}_{i-1}$ meaning the RHS is equivalent to $f(\hat{X}_{i-1}) - f(\hat{X}_{i-1} - \{i\})$, but $f(\hat{X}_{i-1}) - f(\hat{X}_{i-1} - \{i\}) = -r_i$ by definition. This reasoning allows us to conclude:

   $$f(\text{OPT}_{i-1}) - f(\text{OPT}_i) \leq r_i \tag{4.3}$$

   Let us now expand $\frac{1}{2} \cdot a_i$.

   $$\frac{1}{2} \cdot a_i = \frac{1}{2} \cdot \big(f(X_i \cup \{i\}) - f(X_{i-1})\big) = \frac{1}{2} \cdot \big((f(X_i \cup \{i\}) - f(X_{i-1})) + (f(\hat{X}_i) - f(\hat{X}_{i-1}))\big) \tag{4.4}$$

   The addition of term $f(\hat{X}_i) - f(\hat{X}_{i-1})$ follows as $f(\hat{X}_i) - f(\hat{X}_{i-1}) = 0$. This is because $\hat{X}_i = \hat{X}_{i-1}$ since i is added to X_{i-1}. By assumption $r_i < 0$ and $a_i \geq 0$ thus $r_i \leq \frac{1}{2} \cdot a_i$. The above calculations combine to derive the required inequality.

2. Suppose $a_i < 0$ and $r_i \geq 0$: element i is ignored from the maintained solution meaning $X_{i-1} = X_i$. Consequently $\text{OPT}_{i-1} = \text{OPT}_i$. We thus have that

   $$f(\text{OPT}_{i-1}) - f(\text{OPT}_i) = 0 \tag{4.5}$$

   Certainly, $0 \leq \frac{1}{2} \cdot r_i$ thus expanding out r_i derives

   $$\frac{1}{2} \cdot r_i = \frac{1}{2} \cdot \big(f(\hat{X}_{i-1} \cup \{i\}) - f(\hat{X}_{i-1})\big) = \frac{1}{2} \cdot \big((f(X_i \cup \{i\}) - f(X_{i-1})) + (f(\hat{X}_i) - f(\hat{X}_{i-1}))\big) \tag{4.6}$$

   The addition of term $f(X_i \cup \{i\}) - f(X_{i-1})$ follows as $f(X_i \cup \{i\}) - f(X_{i-1}) = 0$ because $X_i = X_{i-1}$. Combining our calculations for this case derives the required inequality.

3. Suppose $a_i \geq 0$ and $r_i \geq 0$: With probability $\frac{a_i}{a_i+r_i}$ we fall into case 1, and with probability $\frac{r_i}{a_i+r_i}$ into case 2. However, in case 1, we have by inequality 4.3

$$f(\text{OPT}_{i-1}) - f(\text{OPT}_i) \leq r_i$$

while in case 2, inequality 4.5 implies

$$f(\text{OPT}_{i-1}) - f(\text{OPT}_i) \leq 0$$

We can thus compute the expectation $\mathbb{E}\big[f(\text{OPT}_{i-1}) - f(\text{OPT}_i)\big]$ as follows

$$\mathbb{E}\big[f(\text{OPT}_{i-1}) - f(\text{OPT}_i)\big] \leq \frac{a_i}{a_i+r_i} \cdot r_i + \frac{r_i}{a_i+r_i} \cdot 0 = \frac{a_i r_i}{a_i+r_i} \tag{4.7}$$

Now the RHS of the required inequality 4.2 can be written as

$$\frac{1}{2} \cdot \mathbb{E}\big[f(X_i) - f(X_{i-1})\big] + \frac{1}{2} \cdot \mathbb{E}\big[f(\hat{X}_i) - f(\hat{X}_{i-1})\big]$$

Let's look at these two expectations separately. For $\mathbb{E}\big[f(X_i) - f(X_{i-1})\big]$, we are in case 1 with probability $\frac{a_i}{a_i+r_i}$ and by equation 4.4

$$f(X_i) - f(X_{i-1}) = a_i$$

With probability $\frac{r_i}{a_i+r_i}$ we are in case 2, and because $\hat{X}_{i-1} = \hat{X}_i$, we have

$$f(\hat{X}_i) - f(\hat{X}_{i-1}) = 0$$

Consequently, we derive

$$\frac{1}{2} \cdot \mathbb{E}\big[f(X_i) - f(X_{i-1})\big] = \frac{1}{2} \cdot \left(\frac{a_i}{a_i+r_i} \cdot a_i + \frac{r_i}{a_i+r_i} \cdot 0\right) = \frac{1}{2} \cdot \frac{a_i^2}{a_i+r_i}$$

For $\mathbb{E}\big[f(\hat{X}_i) - f(\hat{X}_{i-1})\big]$, case 1 admits $\hat{X}_i = \hat{X}_{i-1}$ thus

$$f(\hat{X}_i) - f(\hat{X}_{i-1}) = 0$$

while case 2 admits

$$f(\hat{X}_i) - f(\hat{X}_{i-1}) = r_i \tag{4.8}$$

by equation 4.6. The expectation is thus:

$$\frac{1}{2} \cdot \mathbb{E}\big[f(X_i) - f(X_{i-1})\big] = \frac{1}{2} \cdot \left(\frac{a_i}{a_i+r_i} \cdot 0 + \frac{r_i}{a_i+r_i} \cdot r_i\right) = \frac{1}{2} \cdot \frac{r_i^2}{a_i+r_i}$$

Combining these two expectations, we derive

$$\frac{1}{2} \cdot \mathbb{E}\big[(f(X_i) - f(X_{i-1})) + (f(\hat{X}_i) - f(\hat{X}_{i-1}))\big] = \frac{1}{2} \cdot \frac{a_i^2 + r_i^2}{a_i+r_i} \tag{4.9}$$

Now for any a_i, r_i we have that

$$\frac{a_i r_i}{a_i+r_i} \leq \frac{1}{2} \cdot \frac{a_i^2 + r_i^2}{a_i+r_i}$$

since $(a_i - r_i)^2 \geq 0$. By equations 4.9 and 4.7 we can conclude that

$$\mathbb{E}\big[f(\text{OPT}_{i-1}) - f(\text{OPT}_i)\big] \leq \frac{1}{2} \cdot \mathbb{E}\big[(f(X_i) - f(X_{i-1})) + (f(\hat{X}_i) - f(\hat{X}_{i-1}))\big]$$

as required.

By claim 4.2, we need not consider the case $a_i < 0$ and $r_i < 0$. In all cases, at least for $i \notin \text{OPT}$, the required inequality 4.2 holds thus completing the proof sketch. □

4.3 Concluding Remarks

The search for an optimal approximation algorithm for maximizing a non-monotone submodular function has a storied history. The study of this problem first began in the 1960s. A number of algorithms discovered during this period either focused on special cases, or provided outputs without provable guarantees [6, 8, 5]. It wasn't until 2007, when Feige, Mirrokni and Vondrák first provided a rigorous study of the problem [3]. In their paper, they prove the following hardness result:

Theorem 4.5. *For any $\epsilon > 0$, there does not exist $(\frac{1}{2} + \epsilon)$-approximation algorithm for maximizing a non-monotone submodular function using polynomial number of queries to an oracle access.*

Think of an access oracle as some black-box that provides the value of the submodular function f given $S \subseteq E$. Additionally, they give a number of constant factor approximation algorithms for maximizing a non-montone submodular function – notably a 0.4-approximation using local search.

For the next five years, improvements to the approximation algorithm seemed to add complexity while only making incremental gains in the approximation ratio. Gharan and Vondrák [7] first improved the algorithm to provide a 0.41-approximation using a technique called simulating annealing. Feldman, Naor, and Schwartz [4] then provided a more nuanced analysis to increase the approximation ratio to 0.42. Finally in 2012, Buchbinder, Feldman, Naor, and Schwartz [2] published the 0.5-approximation discussed above, providing a remarkably simple algorithm that achieved optimality without requiring complex analysis seen in preceding work. The problem of derandomizing the double greedy algorithm was resolved only in the past year by [1].

References

[1] N. Buchbinder, & M. Feldman. "Deterministic algorithms for submodular maximization problems." In *ACM Transactions on Algorithms* (2018), 14(3), 32.

[2] N. Buchbinder, M. Feldman, J. Seffi, & R. Schwartz. "A tight linear time (1/2)-approximation for unconstrained submodular maximization." In *SIAM Journal on Computing* (2015), 44(5), 1384-1402.

[3] U. Feige, V. Mirrokni & J. Vondrák. "Maximizing non-monotone submodular functions." In *FOCS* (2007), 461–471.

[4] M. Feldman, J. Naor, and R. Schwartz. "Nonmonotone submodular maximization via a structural continuous greedy algorithm." In *ICALP* (2011), 342–353.

[5] Goldengorin, B., & Ghosh, D. (2005). "A multilevel search algorithm for the maximization of submodular functions applied to the quadratic cost partition problem." In *Journal of Global Optimization*, 32(1), 65-82.

[6] Goldengorin, B., Sierksma, G., Tijssen, G. A., & Tso, M. (1999). "The data-correcting algorithm for the minimization of supermodular functions." In *Management Science*, 45(11), 1539-1551.

[7] S. O. Gharan & J. Vondrák. "Submodular maximization by simulated annealing." In *SODA* (2011), 1098–1117

[8] Khachaturov, V. R. (1968). "Some problems of the consecutive calculation method and its applications to location problems" (Doctoral dissertation, Ph. D. thesis, Central Economics & Mathematics Institute, Russian Academy of Sciences, Moscow, 1968 (in Russian)).