Double Greedy Algorithm for Submodular Maximization[1]

- Let V be a finite universe. A set function $f : 2^V \to \mathbb{R}$ is submodular if it satifies the following "diminishing marginal utilities" property.

$$\text{For any } A \subseteq B \text{ and } i \in V \setminus B, \quad f(A \cup i) - f(A) \geq f(B \cup i) - f(B) \tag{1}$$

  In this note, we describe a beautiful randomized algorithm which gives a $\frac{1}{2}$-approximation to the problem of finding a set S maximizing $f(S)$. Note that this problem is non-trivial as f need not be monotone. Indeed, it generalizes the maximum cut problem in graphs. The algorithm accesses the function via queries, and makes at most $O(n^2)$ queries.

- We first describe a deterministic $\frac{1}{3}$-approximation.

procedure Double Greedy(Submodular $f : 2^V \to \mathbb{R}_{\geq 0}$):
 ▷ *Find set S which maximizes $f(S)$*
 Order the elements of V **arbitrarily**, so we may assume it to be $\{1, 2, \ldots, n\}$.
 Initialize $A \leftarrow \emptyset$ and $B \leftarrow V$
 for $i = 1$ to n **do**:
  $a_i \leftarrow f(A \cup i) - f(A)$
  $b_i \leftarrow f(B \setminus i) - f(B)$
  if $a_i \geq b_i$ **then**:
   $A \leftarrow A \cup i$ ▷ *B remains unchanged.*
  else:
   $B \leftarrow B \setminus i$ ▷ *A remains unchanged.*
 ▷ *Note that at this point $A = B$.*
 return $A = B$.

Theorem 1. Double Greedy gives an $\frac{1}{3}$-approximation unconstrained submodular function minimization.

Proof. I must confess that this short and simple proof is still magical to me. For simplicity, let (A_i, B_i) denote the set (A, B) at the end of loop i with $(A_0, B_0) = (\emptyset, V)$. Thus, $a_i = f(A_{i-1} \cup i) - f(A_{i-1})$ and $b_i = f(B_{i-1} \setminus i) - f(B_{i-1})$.

Let the optimal set be $O \subseteq V$. For $0 \leq i \leq n$, define $C_i := A_i \cup (B_i \cap O)$. Observe that $A_i \subseteq C_i \subseteq B_i$ for all i. It is going to be important to understand how the set C_i behaves depending on where the element i goes. The following captures this.

$$\text{If } A_i = A_{i-1} \cup i, \;\; C_i = \begin{cases} C_{i-1} & \text{if } i \in O \\ C_{i-1} \cup i & \text{if } i \notin O \end{cases} \qquad \text{If } B_i = B_{i-1} \setminus i, \;\; C_i = \begin{cases} C_{i-1} \setminus i & \text{if } i \in O \\ C_{i-1} & \text{if } i \notin O \end{cases} \tag{2}$$

[1] *Lecture notes by Deeparnab Chakrabarty. Last modified : 16th Dec, 2021*
These have not gone through scrutiny and may contain errors. If you find any, or have any other comments, please email me at deeparnab@dartmouth.edu. Highly appreciated!

Finally, note $A_n = B_n = C_n$ which is the set the algorithm returns.

Now define

$$\Phi_i := f(A_i) + f(B_i) + f(C_i)$$

Observe: $\Phi_0 = f(V) + f(O) \geq \texttt{opt}$ and $\Phi_n = 3 \cdot \mathsf{alg}$. The proof of the theorem immediately follows from the following claim, for we get $\Phi_n \geq \Phi_0$.

Claim 1. For any $i \geq 1$, $\Phi_i \geq \Phi_{i-1}$.

Proof. We begin with a simple observation : $a_i + b_i \geq 0$ which follows from the submodularity of f. Thus, the larger of the two is ≥ 0. Let $\Delta\Phi_i := \Phi_i - \Phi_{i-1}$. There are two cases to consider: whether i enters A in loop i or i leaves B in loop i.

The first case occurs when $a_i \geq b_i$. In this case we see $\Delta\Phi_i = a_i + f(C_i) - f(C_{i-1})$. Referring to (2), if $i \in O$, then $\Delta\Phi_i = a_i \geq 0$. Otherwise, $C_i = C_{i-1} \cup i$. Since $C_{i-1} \subseteq B_{i-1}$, by submodularity $f(C_i) - f(C_{i-1}) \geq f(B_{i-1}) - f(B_{i-1} \setminus i) = -b_i$. Thus, $\Delta\Phi_i = a_i - b_i \geq 0$.

The second case occurs when $a_i < b_i$. In this case we see $\Delta\Phi_i = b_i + f(C_i) - f(C_{i-1})$. Again, referring to (2), if $i \notin O$, then $\Delta\Phi_i = b_i \geq 0$. Otherwise, $C_i = C_{i-1} \setminus i$. Since $A_{i-1} \subseteq C_{i-1}$, by submodularity $f(C_{i-1}) - f(C_{i-1} \setminus i) \leq f(A_{i-1} \cup i) - f(A_{i-1}) = a_i$. That is, $f(C_i) - f(C_{i-1}) \geq -a_i$ implying $\Delta\Phi_i = b_i - a_i \geq 0$. ☐

☐

- **Getting a $1/2$-approximation via Randomization.** A slight tweak to the above algorithm leads to an $1/2$-approximation. The algorithm is randomized and returns a distribution over subsets. The *expected* value of the subset returned is at least $\frac{\texttt{opt}}{2}$.

procedure RANDOMIZED DOUBLE GREEDY(Submodular $f : 2^V \to \mathbb{R}_{\geq 0}$):
  ▷ *Find set S which maximizes $f(S)$*
  Order the elements of V **arbitrarily**, so we may assume it to be $\{1, 2, \ldots, n\}$.
  Initialize $A \leftarrow \emptyset$ and $B \leftarrow V$
  for $i = 1$ to n **do**:
    $a_i \leftarrow \max(0, f(A \cup i) - f(A))$
    $b_i \leftarrow \max(0, f(B \setminus i) - f(B))$
    Toss a coin which comes heads with probability $p_i := \frac{a_i}{a_i + b_i}$
    If heads, $A \leftarrow A \cup i$; else $B \leftarrow B \setminus i$.
  ▷ *Note that at this point $A = B$.*
  return $A = B$.

Theorem 2. The expected function value of the set returned by RANDOMIZED DOUBLE GREEDY is at least $\frac{\texttt{opt}}{2}$.

Proof. Like the previous proof, this one is short and magical too. The potential is slightly different this time. It is

$$\Phi_i := f(A_i) + f(B_i) + 2f(C_i)$$

Now, $\Phi_0 \geq 2\texttt{opt}$ and $\mathbf{Exp}[\Phi_n] = 4\,\mathbf{Exp}[\mathsf{alg}]$. The proof of the theorem, therefore, follows from the following lemma.

Lemma 1. For any $i \geq 1$, $\mathbf{Exp}[\Phi_i - \Phi_{i-1} \mid A_{i-1}, B_{i-1}] \;\; \geq 0$

Proof. Note that $\mathbf{Exp}[f(A_i) - f(A_{i-1}) \mid A_{i-1}, B_{i-1}] = p_i \cdot (f(A_{i-1} \cup i) - f(A)) = \frac{a_i^2}{a_i+b_i}$. Similarly, $\mathbf{Exp}[f(B_i) - f(B_{i-1}) \mid A_{i-1}, B_{i-1}] = \frac{b_i^2}{a_i+b_i}$. Next, note that

$$\begin{aligned} \mathbf{Exp}[f(C_i) - f(C_{i-1}) \mid A_{i-1}, B_{i-1}] \;&=\; p_i \cdot [f(C_i) - f(C_{i-1}) \mid A_i = A_{i-1} \cup i] \\ &\quad + (1-p_i) \cdot [f(C_i) - f(C_{i-1}) \mid B_i = B_{i-1} \setminus i] \end{aligned} \tag{3}$$

Now comes the kicker. Using (2), one sees that if $i \in O$ then the expression multiplying p_i in (3) is 0 and if $i \notin O$, the expression multiplying $(1-p_i)$ is 0. Furthermore, if $i \in O$, as argued in Claim 1, the expression multiplying $(1-p_i)$ is at least $-a_i$, and if $i \notin O$, the expression multiplying p_i is $\geq -b_i$. In sum, we get that

$$\mathbf{Exp}[f(C_i) - f(C_{i-1}) \mid A_{i-1}, B_{i-1}] \;\; \geq \;\; \max\left(-p_i b_i, -(1-p_i)a_i\right) \;\; = \;\; -\frac{a_i b_i}{a_i + b_i}$$

Putting everything together, we get $\mathbf{Exp}[\Phi_i - \Phi_{i-1} \mid A_{i-1}, B_{i-1}] \geq (a_i - b_i)^2/(a_i+b_i) \geq 0$. □

□

Notes

The algorithms described here are from the paper [2] by Buchbinder, Feldman, Naor and Schwartz. My presentation follows a presentation by Jan Vondrák. The approximation factor is tight in the sense that any algorithm obtaining an $(\frac{1}{2} + \varepsilon)$-approximation must make exponentially many queries to the submodular function oracle. This result can be found in the paper [3] by Feige, Mirrokni, and Vondrák. The $\frac{1}{2}$-approximation algorithm above is randomized. A deterministic $\frac{1}{2}$-approximation algorithm was given later in the paper [1] by Buchbinder and Feldman.

References

[1] Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. *ACM Transactions on Algorithms (TALG)*, 14(3):1–20, 2018.

[2] 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 (SICOMP)*, 44(5):1384–1402, 2015.

[3] Uriel Feige, Vahab Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing (SICOMP)*, 40(4):1133 – 1153, 2011.