Algorithms for Massive Data Science	Lecture date: April 3, 2019
Instructor: Ben Moseley	Scribe: Yuyan Wang

1 Sample-Prune on Maximum Submodular Coverage Problem

1.1 Recap from Maximum Submodular Coverage Problem

Recall the maximum submodular coverage problem.

Given a universe U of elements as input, for a function f, define $f_A(e) := f(A \cup \{e\}) - f(A)$ to be the marginal increase of f when e is added to the current set A.

Definition 1.1. *A function $f : 2^{|U|} \to \mathbb{R}^+$ is said to be* submodular *if for every $S' \subseteq S \subseteq U$ and $u \in U \setminus S$, we have $f_{S'}(u) \geq f_S(u)$. A function f is said to be* monotone *if $S' \subseteq S \Rightarrow f(S) \geq f(S')$.*

In the maximum submodular coverage problem, the f is assumed to be non-negative. The goal is to choose an S, such that $|S| = k$, to maximize $f(S)$.

The greedy algorithm for this problem, which we mentioned in previous classes, is described as follows:

Algorithm 1 Greedy for Maximum Submodular Coverage Problem

Input: A set U, a monotone, submodular, nonnegative function f defined on all the subsets of U, a number k.

Output: A subset S of U, $|S| = k$, such that S maximizes $f(S)$.

1: Let $S = \emptyset$.
2: **while** $|S| < k$ **do**
2: Select an element e such that $f_S(e) = \max_{e \in U \setminus S} f_S(e)$. Add e to S.
3: **end while**

2 Using Sample-Prune to Solve the Problem

In previous lecture we proved greedy is constant-approximation for optimal. The following theorem shows we can adapt greedy to the MPC setting.

Theorem 2.1. *There is distributed algorithm that simulates an approximation of greedy using $O(kn^\epsilon)$ memory and enough machines to store the entire data set. The algorithm runs in $O(\log k)$ iterations.*

Prior to developing the distributed algorithm, we first give a $\frac{1}{2}$-approximation algorithm for the problem. Let's assume for now, by some "magic", we already know the optimal value beforehand. We start the analysis with the following lemma.

Lemma 2.2. *Fix any $\gamma > 0$. Let* OPT *be the value of an optimal solution S^* and $\tau = \text{OPT} \cdot \frac{\gamma}{2k}$. Consider any $S = \{s_1, \ldots, s_t\} \subseteq U$, $|S| = t \leq k$, with the following properties:*

(i) There is an ordering of elements in S such that for all $0 \leq i < t$, $f_{\{s_1,\ldots,s_i\}}(s_{i+1}) \geq \tau$.

(ii) If $t < k$, then $f_S(u) \leq \tau$, for all $u \in U$.

Then, $f(S) \geq \text{OPT} \cdot \min\left\{\frac{\gamma}{2}, 1 - \frac{\gamma}{2}\right\}$.

Proof. Let $S_i = \{s_1, ..., s_i\}$ for $i = 1, ..., t$. Let $S_0 = \emptyset$.

If $t = k$, then $f(S) = \sum_{i=0}^{k-1} f_{S_i}(s_{i+1}) \geq \sum_{i=0}^{k-1} \tau = k\tau = \text{OPT}\gamma/2$.

If $t < k$, we know that $f_S(u^*) \leq \tau$, for any element $u^* \in S^*$. Furthermore, since f is submodular, let $s_{t+1}, ..., s_m$ be a sequence of elements in $S^* \setminus S$. Let $S_i = S \cup S^* \setminus \{s_{i+1}, ..., s_m\}$ for $i = t, ..., m-1$. Then $f_{S_i}(s_{i+1}) \leq f_S(s_{i+1}) \leq \tau$, and $m \leq k$.

$$\begin{aligned} f(S \cup S^*) - f(S) &= \sum_{i=t}^{m-1} f_{S_i}(s_{i+1}) \\ &\leq \tau \cdot |S^* \setminus S| \leq \gamma \text{OPT}/2 \end{aligned}$$

We have $f(S) \geq f(S \cup (S^* \setminus S)) - \frac{\gamma \text{OPT}}{2} \geq \text{OPT} - \frac{\gamma \text{OPT}}{2} = \left(1 - \frac{\gamma}{2}\right) \text{OPT}$, where the third step follows from the monotonicity of f. □

We let $\gamma = 1$, then we have a $\frac{1}{2}$-approximation algorithm. Then we design the following algorithm:

Algorithm 2 Threshold-greedy

Input: A set U, a monotone, submodular, nonnegative function f defined on all the subsets of U, a number k.

Output: A subset S of U, such that $|S| = k$.

```
Let S = ∅, and γ = 1.
while |S| < k do
    if ∃e, f_S(e) ≥ OPT/(2k) then
        Add an arbitrary e such that f_S(e) ≥ OPT/(2k) to S.
    else
        Exit.
    end if
end while
```

It's easy to see that the solution Threshold-greedy returns satisfies the two conditions in Lemma 2.2, so it is a $\frac{1}{2}$-approximation. Now consider the following distributed algorithm.

Algorithm 3 Distributed Algorithm for Maximum Submodular Covering Problem

Input: A set U, a monotone, submodular, nonnegative function f defined on all the subsets of U, a number k.

Output: A subset S of U, such that $|S| = k$.

1: Sample each element with probability $1/\sqrt{n}$ to get a sample S.
2: Run threshold-greedy on S to get A.
3: Discard elements e such that $f_A(e) \leq \text{OPT}/2k$.
4: Return the remaining set R with A.

Now we want to show that R has small size w.h.p.

Lemma 2.3. $|R| = O(kn^{1/2} \log n)$ *with high probability.*

Proof. We show that $\Pr[|R| \geq 10kn^{1/2} \log n] = O(1/n)$. We refine this bad event depending on each realization of R.

Let $\mathcal{G}$ represent the Threshold-greedy algorithm. Note that sample S determines $A(S) := \mathcal{G}(S)$, which in turn determines remaining elements $R(S)$.

For each possible realization A' of A, we let $R(A')$ denote the resulting R, that is, $R(A') := \{u \in U \mid f_{A'}(u) \geq \text{OPT}/2k\}$. Note that the bad event with $A = A'$ can happen only when no points in $R(A')$ are included in the sample S since otherwise at least one of those points would have been added to the set A, contradicting to the condition $A = A'$.

Hence,

$$\begin{aligned}
\Pr[|R| \geq 10kn^{1/2}\log n] &\leq \Pr\Big[\bigcup_{|A'|<k}\Big(A = A'\Big) \wedge \Big(|R(A')| \geq 10kn^{1/2}\log n\Big) \wedge \Big(R(A') \cap S = \emptyset\Big)\Big] \\
&\leq \sum_{|A'|<k} \Pr\Big[\Big(|R(A')| \geq 10kn^{1/2}\log n\Big) \wedge \Big(R(A') \cap S = \emptyset\Big)\Big] \\
&\leq n^k \cdot (1 - \frac{1}{n^{1/2}})^{10kn^{1/2}\log n} \leq n^k \cdot \exp(-10k\log n) \leq n^{-9k}
\end{aligned}$$

The second inequality is an easy consequence of union bound over all possible realizations of A. As mentioned before, each realization A' of A determines R which we denote as $R(A')$. The third inequality follows from the fact that each point in $R(A')$ is sampled to be in S independently with probability $1/\sqrt{n}$ and because $|A| < k$. □

Lemma 2.3 tells us that after a 2-round Sample-Prune, the remaining set R can be stored in a machine with memory $\tilde{O}(k\sqrt{n})$ with high probability. This leads to the following theorem.

Theorem 2.4. *There is a* $1/2$ *approximation to the maximum submodular coverage problem in the MPC setting. The total memory needed is* $\tilde{O}(k\sqrt{n})$ *and the rounds is* 2.

The following algorithm is modified to exploit this framework better to use less memory with more rounds. Assume that we sample each element independently with $1/(kn^{1-\delta}\log n)$. Then, it is easy to see that $|S| = O(kn^{\delta}\log n)$ w.h.p. Let R_1 denote the U after the first iteration. Using the same argument we can show that $|R_1| = O(n^{1-\delta})$.

Finally, we discuss how we can estimate OPT to an arbitrary precision using one more round and $O(\log k)$ factor more memory.

Note that $\max_{e\in U} f(e) \leq \text{OPT} \leq k\max_{e\in U} f(e)$. Then, we guess different OPT_i's the form $(1+\epsilon)^i$, where ϵ is a constant chosen, such that $\max_{e\in U} f(e) \leq \text{OPT}_i \leq k\max_{e\in U} f(e)$. Apparently, there are $O((1/\epsilon)\log k)$ guesses in total.

For each OPT_i we run a separate Sample-Prune in parallel which takes a different greedy algorithm $\mathcal{G}$ based on a different guess $\text{OPT}_i/(2k)$. Then we use $O((1/\epsilon)\log k)$ factor more memory and one more round.

3 Approximate Greedy Algorithm and Simulation of It

Consider the following approximate greedy algorithm.

Algorithm 4 Greedy for Maximum Submodular Coverage Problem

Input: A set U, a monotone, submodular, nonnegative function f defined on all the subsets of U, a number k.

Output: A subset S of U, $|S| \leq k$, such that S maximizes $f(S)$.

1: Let $S = \emptyset$.
2: **while** $|S| < k$ **do**
2: Choose any e such that $f_S(e) \geq \frac{1}{1+\epsilon}\max_{e'} f_S(e')$. Add e to S.
3: **end while**

Exercise: Prove that this algorithm is a $1 - \frac{1}{e} - O(\epsilon)$ approximation.

Now we want to simulate this algorithm. Consider breaking it up as follows.

Algorithm 5 Simulated Greedy for Maximum Submodular Coverage Problem

Input: A set U, a monotone, submodular, nonnegative function f defined on all the subsets of U, a number k.

Output: A subset S of U, $|S| \leq k$, such that S maximizes $f(S)$.

```
Let S = ∅, i = 1.
Let v* = max_e f({e}) be max additive value.
while v*/(1+ε)^i > εv*/10k do
    for e such that f_S(e) ≥ v*/(1+ε)^i do
        Add e into S.
    end for
    i = i + 1.
end while
```

Exercise: Show Sample-Prune can simulate each iteration in $O(1)$ rounds.

Overall the algorithm runs in $O(\log_{1+\epsilon} k)$ rounds.

This algorithm perfectly simulates the approximate greedy algorithm. Consider e added in iteration i. We have $f_S(e) \geq \frac{v^*}{(1+\epsilon)^i}$, and every element e gives value at most $f_S(e) \leq \frac{v^*}{(1+\epsilon)^{i-1}}$ by sub-modularity (otherwise this element will be added in the previous iteration). These are the elements the approximate greedy would have picked in the same iteration. It stops when either $|S| = k$, or the value of $f_S(u)$ of $u \in U \setminus S$ is at most $\frac{\epsilon v^*}{10k}$. Total remaining value that can be obtain is at most k times this, which is no more than $\frac{\epsilon v^*}{10}$. This is negligible since we obtain value at least v^*. Therefore, there is no need to further add any element into the set.