AM 221: Advanced Optimization Spring 2016

Prof. Yaron Singer *Lecture 18 — April 4th*

1 Overview

In the last lecture we studied the Knapsack problem which is an NP-complete optimization problem and we gave an algorithm which can solve within an approximation of $(1-\varepsilon)$ for any $\varepsilon > 0$ in time $O(\frac{n^3}{\varepsilon})$.

Today, we will study the Max Cover problem and submodular optimization which are generalizations of the Knapsack problem.

2 Max Cover

Input: sets $T_1, \dots, T_n$ that cover some universe.

Goal: Find k sets whose union is maximal, *i.e.* find:

$$S \in \text{argmax}_{R:|R|\leq k} \left| \bigcup_{i \in R} T_i \right|$$

Equivalent formulation. There is a bipartite graph. Elements of the universe are the vertices on one side of the graph, and each set is a vertex on the other side. There is an edge between a set and an element of the universe iff the element is contained in the set. The sets are usually called *parents* and the elements they contain are their *children*. The goal is to select a set of k parents which are connected to as many children as possible.

A greedy algorithm for Max Cover. It is possible to show that Max Cover is an NP-complete to show. However, we can hope to construct an approximation algorithm for this problem. A natural candidate algorithm is the Greedy Algorithm presented in Algorithm 1. The analysis of this algorithm will be done later after we introduce some new terminology.

Algorithm 1 Greedy algorithm for Max Cover

$S \leftarrow \varnothing$
while $|S| \leq k$ **do**
    $T \leftarrow$ set that covers the most elements that are not yet covered by S
    $S \leftarrow S \cup \{T\}$
end while
return S

3 Submodular functions

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

$$f(S \cup T) \leq f(S) + f(T) - f(S \cap T), \quad S, T \subseteq N$$

Example. Here are a few examples of classes of submodular functions:

- *Additive functions:* $f(S) = \sum_{a \in S} f(a)$. Indeed if $S \cap T \neq \emptyset$ we have:

$$f(S \cup T) = \sum_{a \in S \cup T} f(a) = \sum_{a \in S} f(a) + \sum_{a \in T} f(a)$$

  If not, we can write $S \cup T = (S \setminus (S \cap T)) \cup T$ and use that:

$$f(S \setminus (S \cap T)) = \sum_{a \in S} f(a) - \sum_{a \in S \cap T} f(a)$$

- *Unit-demand functions:* $f(S) = \max_{a \in S} f(a)$.
- *Coverage functions:* $f(S) = \left|\bigcup_{i \in S} T_i\right|$ given sets $T_1, \ldots, T_n$.

So the Knapsack problem and the Max Cover problem are specific examples of submodular optimization problems. Our goal is now to analyze Algorithm 1 for a general submodular function.

4 Properties of submodular functions

Definition 2. *For a function $f : 2^N \to \mathbb{R}$ and set a $S \subseteq N$, the **marginal contribution** of $T \subseteq N$ to S is:*

$$f_S(T) = f(T \cup S) - f(S)$$

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

$$f_S(a) \geq f_T(a), \quad S \subseteq T,\ a \in N \setminus T$$

Definition 4. *A function $f : 2^N \to R$ is **subadditive** iff:*

$$f(S \cup T) \leq f(S) + f(T), \quad S, T \subseteq N$$

Definition 5. *A function $f : 2^N \to R$ is **monotone** iff:*

$$f(S) \leq f(T), \quad S \subseteq T$$

Proposition 6. *If a function is monotone and submodular then f_S is subadditive for any $S \subseteq N$. You will do this in the problem set.*

Proof. Use that $f_S(T) = f(S \cup T) - f(S)$ $\square$

Algorithm 2 Greedy algorithm for any submodular function

$S \leftarrow \varnothing$
while $|S| \leq k$ **do**
  $S \leftarrow S \cup \text{argmax}_{a \notin S} f_S(a)$
end while
return S

5 An algorithm for Submodular Maximization

With this new terminology, we can rewrite Algorithm 1 for a general submodular function: adding the set which covers the most elements that are not yet covered by S is equivalent to choosing the set which maximizes the marginal contribution to the current solution.

Theorem 7. *For any monotone submodular function $f : 2^N \to \mathbb{R}$, Algorithm 2 returns a set S such that:*

$$f(S) \geq \left(1 - \frac{1}{e}\right) \max_{T:|T| \leq k} f(T)$$

Remark. $1 - \frac{1}{e} \simeq 0.63$, so the greedy algorithm gets to 63% of the optimal value.

Let us define $\text{OPT} = \max_{|T| \leq k} f(T)$. The proof of this theorem will rely on the following lemma.

Lemma 8. *Let S be the set selected by the greedy algorithm at some stage and let $a \notin S$ be the element added to S at this stage. Then:*

$$f_S(a) \geq \frac{1}{k}\big(OPT - f(S)\big)$$

Proof. Let O be the optimal solution and let $o^* \in \text{argmax}_{o \in O} f_S(o)$. Because f_S is subadditive:

$$f_S(O) \leq \sum_{o \in O} f_S(o) \leq k \cdot f_S(o^*) \leq k \cdot f_S(a)$$

where the first inequality used that the marginal contribution is subadditive (Lemma 6), and the last inequality used that by definition a is the element which maximizes the marginal contribution.

This implies:

$$f_S(a) \geq \frac{1}{k} f_S(O) = \frac{1}{k}\big(f(S \cup O) - f(S)\big) \geq \frac{1}{k}\big(f(O) - f(S)\big)$$

where the last inequality used the monotonicity of f. □

We are now ready to prove the theorem.

Proof. The proof is by induction. Let $S_i = \{a_1, \ldots, a_i\}$ be the set of elements selected by greedy after iteration i for $i \in \{1, \ldots, k\}$. We will prove:

$$f(S_i) \geq \left(1 - \left(1 - \frac{1}{k}\right)^i\right) f(O), \quad 1 \leq i \leq k \tag{1}$$

First note that Lemma 8 can be rewritten as:

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

Base case. For $i = 1$, we have $S_0 = \emptyset$, hence:

$$f(S_1) = f(a_1) \geq \frac{1}{k} f(O) = \left(1 - \left(1 - \frac{1}{k}\right)\right) f(O)$$

Inductive step. Assume the result holds $i = \ell$, we will prove for $i = \ell + 1$:

$$f(S_{\ell+1}) \geq \frac{1}{k}\left(f(O) - f(S_\ell)\right) + f(S_\ell) = \frac{1}{k}\left(f(O)\right) + \left(1 - \frac{1}{k}\right) f(S_\ell)$$

Now, by applying the inductive hypothesis:

$$\begin{aligned} f(S_{\ell+1}) &\geq \frac{1}{k}\left(f(O)\right) + \left(1 - \left(1 - \frac{1}{k}\right)^{\ell}\right)\left(1 - \frac{1}{k}\right) f(O) \\ &= \left(1 - \left(1 - \frac{1}{k}\right)^{\ell+1}\right) f(O) \end{aligned}$$

We can now conclude by using Equation 1 for $i = k$ and using that for $k \geq 1$, $(1 - 1/k)^k \leq \frac{1}{e}$, hence:

$$f(S_{\ell+1}) \geq \left(1 - \frac{1}{e}\right) f(O) \qquad \square$$

Should we be happy about this result? Yes, it was proven in 1998 that unless P=NP, no polynomial-time algorithm can obtain an approximation ratio better than $1 - 1/e$.

6 Maximizing influence in Social Networks

A nice application of submodularity is to the problem of Influence Maximization in social networks: an company wants to run a marketing campaign on a social network and wants to target a few influential individuals who will then spread awareness of the product being targeted to the rest of the network.

Goal: Select a subset of individuals who will be most influential.

Of course, this problem depends a lot on how to define and quantify the influence of individuals. A possible model of influence is to assume that there is a probability attached to each edge in the network. Once a node gets infected, it spreads the infection to its neighbors according to the edges' probabilities.

See more details in the section notes for this week.