AM 221: Advanced Optimization Spring 2016

Prof. Yaron Singer *Problem Set 11 — Due Wed, Apr. 20th at 23:59*

Instructions: All your solutions should be prepared in LaTeX and the PDF and .tex should be submitted to canvas. For each question, the best and correct answers will be selected as sample solutions for the entire class to enjoy. If you prefer that we do not use your solutions, please indicate this clearly on the first page of your assignment.

The programming parts can be written in the programming language of your choice and the code should be submitted alongside your solutions.

1. Local search for constrained monotone submodular maximization. In this problem, we are interested in the following optimization problem:

$$\max_{|S|\leq k} f(S) \tag{1}$$

where $f : 2^N \to \mathbb{R}^+$ is a monotone submodular function and $k \in \mathbb{N}$ is the budget. We have already seen in class that the greedy algorithm gets an approximation ratio of $(1 - \frac{1}{e})$ for this problem. Here, we will analyze an alternative algorithm based on the idea of local search.

Definition 1. *Consider a set $S \subseteq N$. We say that a pair (u, v) with $u \in S$ and $v \in N \setminus S$ is a **good exchange** if:*

$$f(S \setminus \{u\} \cup \{v\}) > f(S)$$

In other words, we can improve the value of $f(S)$ by removing u and adding v instead. Consider the following local search algorithm

Algorithm 1 Local search for constrained monotone optimization

1: compute $v^* \in \text{argmax}_{v \in N} f(v)$
2: add $k - 1$ elements arbitrarily to construct S with $|S| = k$ and $v^* \in S$.
3: **while** there exists a good exchange (u, v) **do**
4: $\quad S \leftarrow S \setminus \{u\} \cup \{v\}$
5: **end while**
6: **return** S

a. Let S be the set returned by Algorithm 1. Let us write $S = \{e_1, \ldots, e_k\}$ and $S_i = \{e_1, \ldots, e_i\}$ (with the convention $S_0 = \emptyset$). Let O be an optimal solution to (1). Show that:

$$f_S(o) \leq f(S_i) - f(S_{i-1}), \quad i \in [k],\, o \in O$$

b. Recall Lemma 8 from lecture 18:

$$f_S(o) \geq \frac{1}{k}[f(O) - f(S)]$$

and conclude that $f(S) \geq \frac{1}{2} f(O)$.

The problem of Algorithm 1 is that the while loop could loop for exponentially many steps. One way to fix this is to only consider exchanges which improve f by a fixed margin. This motivates the following definition.

Definition 2. *Consider a set $S \subseteq N$ and $\alpha > 0$. We say that a pair (u, v) with $u \in S$ and $v \in N \setminus S$ is a α-**good exchange** if:*

$$f(S \setminus \{u\} \cup \{v\}) \geq (1 + \alpha) f(S)$$

and the modified local search algorithm.

Algorithm 2 Local search for constrained monotone optimization with parameter α

1: compute $v^* \in \text{argmax}_{v \in N} f(v)$
2: add $k - 1$ elements arbitrarily to construct S with $|S| = k$ and $v^* \in S$.
3: **while** there exists an α-good exchange (u, v) **do**
4: $\quad S \leftarrow S \setminus \{u\} \cup \{v\}$
5: **end while**
6: **return** S

c. Show that the set S obtained at line 2. of Algorithm 2 is such that:

$$f(S) \geq \frac{1}{n} f(O)$$

d. Show that the number m of iterations of the while loop in Algorithm 2 is upper-bounded by:

$$m \leq \frac{\log n}{\log(1 + \alpha)}$$

e. Adapt the analysis of parts a. and b. to show that if S is the set returned by Algorithm 2 then:

$$f(O) \leq (2 + \alpha n) f(S)$$

f. Conclude that for any $\varepsilon > 0$, there exists a $(\frac{1}{2} - \varepsilon)$-approximation algorithm for (1) and whose running time is polynomial in n and $\frac{1}{\varepsilon}$.

2. Monotone submodular maximization under budget constraint. In this problem we will consider a variant of the submodular maximization problem in which every element $u \in N$ is associated with some cost $c_u \in \mathbb{R}^+$. There is now a budget B on the total cost of the selected set S. We want to solve:

$$\begin{aligned} &\max_S f(S) \\ &\text{s.t.} \sum_{u \in S} c_u \leq B \end{aligned} \tag{2}$$

We will consider the greedy algorithm described in Algorithm 3 for this problem. The idea of this algorithm is similar to the greedy algorithm for the Knapsack problem: at each iteration of the algorithm, we add to the current solution the element which has the largest density. Here, the density of an element, is defined to be the ratio of the marginal contribution of the element divided

Algorithm 3 Greedy algorithm for budgeted submodular optimization

$S \leftarrow \emptyset$
while True **do**
  $u^* \leftarrow \text{argmax}_{u \in N} \frac{f_S(u)}{c_u}$
  if $c_{u^*} + \sum_{u \in S} c_u \leq B$ **then**
    $S \leftarrow S + u^*$
  else
    return S
  end if
  $N \leftarrow N - u^*$
end while

by its cost. Without loss of generality we will assume that for all $u \in N$, $c_u \leq B$ (otherwise we can simply remove this element from N since it will never be added to any solution).

Let S be the set returned by Algorithm 3 and let us write $S = \{e_1, \ldots, e_m\}$ where the elements are numbered in the order in which they where added to S in Algorithm 3. Let us also write $S_i = \{e_1, \ldots, e_i\}$ (with the convention $S_i = \emptyset$). Finally let O be an optimal solution to (2).

a. Adapt the analysis of the greedy algorithm for submodular maximization done in class to show that:
$$f(S_i) \geq \left(1 - \exp\left(\frac{-\sum_{u \in S_i} c_u}{B}\right)\right) f(O)$$

b. Let u^* be the element which would have been added to S just before the algorithm terminates. In other words, we have $u^* = \text{argmax}_{u \in N} \frac{f_S(u)}{c_u}$ and $c_{u^*} + \sum_{u \in S} c_u > B$ (the element could not be added because of the budget constraint). Show that:
$$f(S \cup \{u^*\}) \geq \left(1 - \frac{1}{e}\right) f(O)$$

c. Let us consider $v^* \in \text{argmax}_{v \in N} f(v)$. Show that:
$$\text{argmax}\{f(S), f(v^*)\} \geq \frac{1}{2}\left(1 - \frac{1}{e}\right) f(O)$$

3. Randomized-Rounding for Max-Cover. Recall the input to the Max-Cover problem:

- universe $\mathcal{U} = \{1, \ldots, m\}$
- sets $T_1, \ldots, T_n$ with $T_i \subseteq \mathcal{U}$
- budget $k \in \mathbb{N}$

The goal is to solve:
$$\max_{|S| \leq k} \left| \bigcup_{j \in S} T_j \right| \tag{3}$$

In the problem we will give a randomized algorithm for MAX-COVER using randomized rounding. Recall the piecewise linear relaxation for Max-Cover seen in class where we associate a variable x_j to each set T_j:

$$\begin{aligned} \max_{\mathbf{x}} & \sum_{i=1}^{m} \min\left\{1, \sum_{j:i\in T_j} x_j\right\} \\ \text{s.t.} & \sum_{i=1}^{n} x_i \leq K, \\ & x_i \in [0,1], \ \forall i \in [n] \end{aligned}$$

a. Let $\mathbf{x}^* = (x_1, x_2, \ldots, x_n)$ be the optimal solution to the optimization problem above. Argue that $\sum_{i=1}^{m} \min\{1, \sum_{j:i\in T_j} x_j\}$ is an upper bound on the optimal solution to the MAX-COVER problem.

b. Prove that selecting every set T_i with probability x_i will have value $(1-1/e)OPT$ and respect the cardinality constraint, *in expectation*. That is, if we start with $S = \emptyset$ and add every set T_i to S independently with probability x_i we will have $\mathbb{E}[f(S)] \geq (1-1/e)OPT$ and $\mathbb{E}[|S|] = K$, where $f : 2^{[n]} \to \mathbb{R}$ is the objective function in (3). [**Hint:** use the randomized-rounding proof technique used in SET-COVER from class].

4. Document Clustering via Submodular Maximization. In this problem we will use submodular maximization to cluster a corpus of documents. A standard practice in the field of natural language processing is to manipulate documents as *bags of words*. Formally:

- there is a corpus of D documents: $\{1, \ldots, D\}$,
- there is a vocabulary of W words: $\{1, \ldots, W\}$,
- each document $i \in [D]$ is represented as a sparse vector $d_i \in \mathbb{N}^W$ where the jth entry d_{ij} is the number of times word $j \in [W]$ appeared in document i. The sparse vector d_i is the *bag of words* associated with document i.

A simple but widely used similarity metric between documents is the *weighted Jaccard similarity coefficient*. For two bag of words, d and e with $d, e \in \mathbb{N}^W$, the coefficient $J(d, e)$ is defined by:

$$J(d,e) = \frac{\sum_{1\leq j\leq W} \min(d_j, e_j)}{\sum_{1\leq j\leq W} \max(d_j, e_j)}$$

a. Show that for any two bag of words $0 \leq J(d, e) \leq 1$. What can you say when $J(d, e) = 0$? $J(d, e) = 1$?

b. We will cluster the documents in k clusters by solving the following optimization problem:

$$\max_{|S|\leq k} \sum_{i=1}^{D} \max_{j\in S} J(d_i, d_j) \tag{4}$$

Show that the objective function in this optimization problem is submodular.

We will solve use this optimization problem to cluster the dataset available at http://thibaut.horel.org/enron.zip. This dataset contains two files `dict.txt` and `corpus.txt`. The file `dict.txt` is the dictionary which contains all the words in the vocabulary. The file `corpus.txt` contains a sparse representation of the famous Enron Email corpus; each line has the following format:

```
docID  wordID  count
```

where `docID` identifies a document (an email) in the corpus, `wordID` identifies a word in the vocabulary (a line number in the file `dict.txt`) and `count` is the number of times this word appeared in this document.

c. Write some code which solves the optimization problem (4) on the Enron corpus for $k = 10$[1]. Given a solution S, the clustering of the documents is obtained by associating each document d in the corpus with the document e in S having the largest similarity to d. For each document e in S, there is a corresponding cluster of all the documents associated with e.

d. For each cluster, compile the list of the top 100 words which appeared the most frequently in this cluster's documents. By visually inspecting these lists can you identify a label/theme for each cluster?

[1]While it is possible to handle the entire corpus by writing efficient code, it is possible that you will run into computational bottlenecks if you are not really careful. If this is the case, feel free to truncate the dataset to only keep a number of documents that your code can handle.