**Topics in Theoretical Computer Science** May 20, 2014

Lecture 13

*Lecturer: Moran Feldman* *Scribes: David Leydier and Samuel Grütter*

In this lecture we will talk about some concrete problems regarding submodular functions.

# 1 The Submodular Vertex Cover Problem

**Definition 1** *We have a given graph $G = (V, E)$ and a normalized, monotone and submodular function $f : 2^V \longrightarrow \mathbb{R}_+$. The objective is to find a vertex cover $C$ of $G$ which minimizes $f(C)$. So, we define the following problem:*

$$\min \hat{f}(x)$$

*Where $\hat{f}$ is the Lovasz extension of $f$. Subject to the following constraints:*

$$\begin{array}{ll} x_u + x_v \geq 1 & \forall e = (u, v) \in E \\ x_u \geq 0 & \forall u \in V \end{array}$$

**Remark** Since $f$ is monotone and normalized we can easily deduce that it's non-negative.

We now have to round this program. We will do it in the same way that we've done before for the linear version of Vertex Cover.

Let $y$ be an OPT solution of the above program and $S = \{ u \mid y_u \geq \frac{1}{2} \}$. For every edge $e = (u, v) \in E$, $y_u + y_v \geq 1$ therefore either $y_u \geq \frac{1}{2}$ or $y_v \geq \frac{1}{2}$, so S contains either $u$ or $v$: this proves that S is a VC for G.

**Theorem 2** *This algorithm is a $\frac{1}{2}$-approximation algorithm for SVC.*

**Proof** So, we want to show that $f(S) \leq 2 \cdot OPT$.

$$OPT \geq \hat{f}(y) = \int_0^1 f(T_\lambda(y))\,\mathrm{d}\lambda$$

Moreover, for $\lambda \leq \frac{1}{2}$ then we have $S \subseteq T_\lambda$, and so because of monotonicity of f: $f(T_\lambda(y)) \geq f(S)$.

$$\int_0^1 f(T_\lambda(y))\,\mathrm{d}\lambda \geq \int_0^{\frac{1}{2}} f(T_\lambda(y))\,\mathrm{d}\lambda \geq \int_0^{\frac{1}{2}} f(S)\,\mathrm{d}\lambda = \frac{f(S)}{2}$$

■

We now want to prove, that this approximation is more or less the best that we can do.

Let's consider a family of instances of SVC using a graph with $n$ isolated edges (and thus $2n$ vertices). The objective function $f_S(T)$ depends on a set of vertices $S$. Let $\delta > 0$ be a constant that we will determine later, and define $f_S$ as

$$f_S(T) = \min\{|\overline{S} \cap T| + \min\{|S \cap T|, \frac{(1+\delta)n}{2}\}, n\}$$

**Lemma 3** *For any fixed $T \subseteq N$,* $\Pr[f_S(T) \neq \min\{|T|, n\}] < e^{\frac{-\delta^2 n}{6}}$

**Remark** If $f_S$ returns $|T|$ or $n$ to the algorithm, that's not very useful for the algorithm in order to find the minimum, so this lemma tells us that the probability that the algorithm gets a useful answer is very low.

**Proof**

Let $T_1 = \{u \in T | (u,v) \in E \text{ and } v \notin T\}$ and $n_1 = n - |T \backslash T_1|$.

**Case 1:** $|T_1| \leq n_1$:

$$\begin{aligned}
& P_r[f_s(T) \neq \min\{n, |T|\}] \\
\leq\ & P_r[|S \cap T| > \tfrac{(1+\delta)n}{2}] \\
=\ & P_r[\ \tfrac{|T \backslash T_1|}{2} + |S \cap T_1\ | > \tfrac{(1+\delta)n}{2}] \\
=\ & P_r[\ |S \cap T_1| > \tfrac{n_1 + \delta n}{2}\ ] \\
=\ & P_r[\ |S \cap T_1| > \tfrac{n_1+\delta n}{|T_1|}\ \mathbb{E}[|S \cap T_1|]\ ] \\
\leq\ & e^{-\ (\frac{n_1+\delta n}{|T_1|} - 1)^2\ \frac{\mathbb{E}[|S \cap T_1|]}{3}} && \text{with the Chernoff bound} \\
=\ & e^{-\ \frac{(n_1 + \delta n - |T_1|)^2}{6\ |T_1|}} \\
\leq\ & e^{\frac{-\delta^2 n}{6}} && \text{because } |T_1| \text{ is bounded}
\end{aligned}$$

**Case 2:** $|T_1| \geq n_1$:

$$\begin{aligned}
& P_r[f_s(T) \neq min\{n, |T|\}] \\
\leq\ & P_r[|\overline{S} \cap T| > \tfrac{(1-\delta)n}{2}] \\
=\ & P_r[\ \tfrac{|T \backslash T_1|}{2} + |\overline{S} \cap T_1\ | > \tfrac{(1-\delta)n}{2}] \\
=\ & P_r[\ |\overline{S} \cap T_1| > \tfrac{n_1 - \delta n}{2}\ ] \\
=\ & P_r[\ |\overline{S} \cap T_1| > \tfrac{n_1-\delta n}{|T_1|}\ \mathbb{E}[|\overline{S} \cap T_1|]\ ] \\
\leq\ & e^{-\ (\frac{n_1+\delta n}{|T_1|} - 1)^2\ \frac{\mathbb{E}[|S \cap T_1|]}{3}} \\
=\ & e^{-\ \frac{(n_1 + \delta n - |T_1|)^2}{4\ |T_1|}} && \text{maximum when } |T_1| \text{ is minimal} \\
\leq\ & e^{\frac{-\delta^2 n}{4}}
\end{aligned}$$

■

**Theorem 4** *For every constant $\epsilon > 0$, every $(2-\epsilon)$-approximation algorithm for SVC must use an exponential number of oracle queries.*

**Proof** Assume that ALG is a deterministic algorithm which is making a subexponential number of queries $h$. Assume without loss of generality that ALG always returns a set that it queries. Let $T_1, ..., T_h$ be the sets that ALG queries when given $g(T) = \min\{n, |T|\}$. If $g(T_i) = f_S(T_i)$ for every $1 \leq i \leq h$ then ALG will return the set $T_i$ when given either $g$ or $f_S$. Moreover the value of this set will be at least $n$ since $g$ returns $n$ for every feasible VC.

The probability $g(T_i) = f_S(T_i)$ for $1 \leq i \leq h$ is at least $1 - he^{-\ \frac{\delta^2 n}{6}} \geq 1 - e^{-\ \frac{\delta^2 n}{7}}$.

Then the expected value of ALG is at least $n(1 - e^{-\frac{\delta^2 n}{7}})$ and on the other hand the OPT solution for $f_S$ is $S$ with the value $\frac{(1+\delta)n}{2}$.

The approximation ratio of ALG is no better than, $\frac{2(1-e^{-\frac{\delta^2 n}{7}})}{1+\delta}$ which is $\geq 2 - \epsilon$ for a large enough $n$ and some small enough $\delta$.

■

## 2 The Submodular Maximization Problem subject to cardinality constraints

**Definition 5** *In the* Submodular Maximization Problem subject to cardinality constraints*, we are given a normalized monotone submodular function $f : 2^N \to \mathbb{R}^+$ and an integer paramter $1 \leq k \leq |N|$. The objective is to find sets $S \subseteq N$ of size $k$ maximizing $f(S)$.*

**Remark** One might ask why we only accept sets of size exactly $k$, and just all sets of size at most $k$. That's because $f$ is monotone: If we have a solution of size smaller than $k$, we can always add some elements and its value will not decrease.

Let's consider the following greedy algorithm:

- Let $S_0$ be $\emptyset$.
- For $i = 1 \ldots k$ do
  - Select the element $u \in N$ maximizing the marginal contribution $f_u(S_{i-1})$
  - Let $S_i \leftarrow S_{i-1} + u$.
- Return $S_k$.

**Remark** Contrary to some LP and SDP approaches we saw in the last lectures, the algorithms we see in this lecture do not depend on the ellipsoid method (which cannot be implemented nicely and efficiently, even though it runs in polynomial time).

**Theorem 6** *The above algorithm is a $(1 - \frac{1}{e})$-approximation for SC.*

To prove this theorem, we will prove a lemma from which the theorem follows immediately:

**Lemma 7** *For every $0 \leq i \leq k$, $f(S_i) \geq [1 - (1 - \frac{1}{k})^i] f(OPT)$.*

**Proof** By induction on $i$: If $i = 0$, $f(S_0) = f(\emptyset) = 0 = [1 - (1 - \frac{1}{k})^0] f(OPT)$. Now we assume the lemma holds for $i - 1 \geq 0$, and we prove it for $i$:

$$\begin{array}{lll}
& f(S_i) = f(S_{i-1} + u_i) = f(S_{i-1}) + f_{u_i}(S_{i-1}) & \\
\geq & f(S_{i-1}) + \frac{1}{k} \sum_{u \in OPT} f_u(S_{i-1}) & \text{since } \forall u \in OPT, f_u(S_{i-1}) \leq f_{u_i}(S_{i-1}) \\
\geq & f(S_{i-1}) + \frac{1}{k}(f(S_{i-1} \cup OPT) - f(S_{i-1})) & \text{by submodularity} \\
\geq & f(S_{i-1}) + \frac{1}{k}(f(OPT) - f(S_{i-1})) & \text{by monotonicity} \\
= & (1 - \frac{1}{k}) f(S_{i-1}) + \frac{1}{k} \cdot f(OPT) & \\
\geq & (1 - \frac{1}{k})[1 - (1 - \frac{1}{k})^{i-1}] f(OPT) + \frac{1}{k} \cdot f(OPT) & \text{by induction hypothesis} \\
= & [1 - (1 - \frac{1}{k})^i] f(OPT) &
\end{array}$$

■

# 3 The Unconstrained Submodular Maximization Problem

**Definition 8** *In the* Unconstrained Submodular Maximization Problem (USMax)*, we are given a non-negative submodular function* $f : 2^N \to \mathbb{R}^+$*. The objective is to find a* $S \subseteq N$ *maximizing* $f(S)$*.*

**Remark** $f$ is not required to be monotone, because if it was, the problem would be trivial, because $N$ would always be an optimal solution.

## 3.1 A (too) naïve greedy algorithm

One could come up with a very simple greedy algorithm for this problem: Always start with the empty set, and repeatedly add the element with the largest marginal contribution, until there's no element which increases the value of $f$ if we add it. However, this algorithm performs very bad on the following example:

$$U = \{u_1, u_2, \dots u_n, v\}$$

$$f(S) = \begin{cases} 2 & \text{if } v \in S \\ |S| & \text{if } v \notin S \end{cases}$$

The optimum is to choose $S = \{u_1, u_2, \dots u_n\}$ of value $n$, but the greedy algorithm would choose $\{v\}$, of value 2.

## 3.2 A greedy deterministic $\frac{1}{3}$-approximation algorithm

So we need a better algorithm. We still choose a greedy algorithm, but one which is "greedy from two sides" (one side being $\emptyset$ and the other being $N$). Formally:

- Choose an arbitrary order $u_1, u_2, \dots u_n$ on the elements.
- Let $X_0 \leftarrow \emptyset$ and $Y_0 \leftarrow N$.
- For $i$ from 1 to $n$ do
  - Let $a_i \leftarrow f(X_{i-1} + u_i) - f(X_{i-1})$ and $b_i \leftarrow f(Y_{i-1} - u_i) - f(Y_{i-1})$.
  - If $a_i \geq b_i$, then $X_i \leftarrow X_{i-1} + u_i$ and $Y_i \leftarrow Y_{i-1}$.
  - Else $X_i \leftarrow X_{i-1}$ and $Y_i \leftarrow Y_{i-1} - u_i$.
- Return $X_n$ (which is equal to $Y_n$).

**Theorem 9** *The above algorithm is a* $\frac{1}{3}$*-approximation of USMax.*

Before proving the theorem, we need some lemmas:

**Lemma 10** *For every* $1 \leq i \leq n$*,* $a_i + b_i \geq 0$*.*

**Proof** Since we have $(X_{i-1}+u_i)\cap(Y_{i-1}-u_i)=X_{i-1}$ and $(X_{i-1}+u_i)\cup(Y_{i-1}-u_i)=Y_{i-1}$ we can apply submodularity as follows:

$$\begin{aligned} a_i+b_i = &\ f(X_{i-1}+u_i)-f(X_{i-1})+f(Y_{i-1}-u_i)-f(Y_{i-1}) && \text{by definition} \\ = &\ f(X_{i-1}+u_i)+f(Y_{i-1}-u_i)-[f(X_{i-1})+f(Y_{i-1})]\geq 0 && \text{by submodularity} \end{aligned}$$

■

**Definition 11** *Let* $OPT_i=(OPT\cup X_i)\cap Y_i$

The idea of this definition is to have a variant of OPT which we force to agree with what the algorithm does.

**Lemma 12** *If* $u_i\notin OPT$ *and the algorithm decides to pick* $u_i$*, then* $f(OPT_{i-1})-f(OPT_i)\leq b_i$*.*
*If* $u_i\in OPT$ *and the algorithm decides to reject* $u_i$*, then* $f(OPT_{i-1})-f(OPT_i)\leq a_i$*.*

**Proof** We prove only the first case. The other case is similar. First, note that we always have $X_i\subseteq OPT_i\subseteq Y_i$. So we also have $Y_{i-1}-u_i\supseteq Y_i-u_i\supseteq OPT_i-u_i$. And by submodularity, we have that the marginal contribution of $u_i$ to a bigger set is smaller than the marginal contribution to a smaller set:

$$f((Y_{i-1}-u_i)+u_i)-f(Y_{i-1}-u_i)\leq f((OPT_i-u_i)+u_i)-f(OPT_i-u_i)$$

Multiplying this inequality by $-1$, and using that $OPT_{i-1}=OPT_i-u_i$, we find that

$$f(OPT_{i-1})-f(OPT_i)=f(OPT_i-u_i)-f(OPT_i)\leq f(Y_{i-1}-u_i)-f(Y_{i-1})=b_i$$

■

**Lemma 13** *For* $1\leq i\leq n$*,* $f(OPT_{i-1})-f(OPT_i)\leq[f(X_i)-f(X_{i-1})]+[f(Y_i)-f(Y_{i-1})]$*.*

**Proof** We assume $u\notin OPT$, the case $u\in OPT$ is similar. Now we distinguish two cases:

- If the algorithm did not pick $u_i$: We need to show that $0\leq 0+b_i$. We know that $a_i+b_i\geq 0$, and since the algorithm did not pick $u_i$, we have $b_i\geq a_i$, so $b_i\geq 0$.
- If the algorithm decides to pick $u_i$ (i.e. if it made the wrong choice): We have $f(OPT_{i-1})-f(OPT_i)\leq b_i$ and $f(X_i)-f(X_{i-1})=a_i$ and $f(Y_i)-f(Y_{i-1})=0$, so we need to show $b_i\leq a_i+0$, which holds because we took $u_i$.

■

Now we have everything we need to prove the theorem:

**Proof** We sum up the relation of the above Lemma from 1 to $n$:

$$\sum_{i=1}^{n}f(OPT_{i-1})-f(OPT_i)\leq\sum_{i=1}^{n}f(X_i)-f(X_{i-1})+\sum_{i=1}^{n}f(Y_i)-f(Y_{i-1})$$

and simplify the sums:

$$f(OPT_0)-f(OPT_n)\leq f(X_n)-f(X_0)+f(Y_n)-f(Y_0)$$

and since $f$ is non-negative, we can discard $f(X_0)$ and $f(Y_0)$:

$$f(OPT_0)\leq f(X_n)+f(Y_n)+f(OPT_n)$$

and using that $OPT_0=OPT$ and $OPT_n=X_n=Y_n$:

$$f(OPT)\leq 3\cdot f(X_n)$$

■

### 3.3 A greedy randomized $\frac{1}{2}$-approximation algorithm

Let's consider the following variation of the above algorithm: At iteration $i$, we do the following:

- If $b_i \leq 0$, pick $u_i$.
- If $a_i < 0$, reject $u_i$.
- Otherwise pick $u_i$ with probability $\frac{a_i}{a_i+b_i}$.

**Theorem 14** *This variant of the algorithm is a $\frac{1}{2}$-approximation for USMax in expectation.*

The proof of this theorem works the same as the proof of theorem 9, except for lemma 13, which we have to replace by the following lemma:

**Lemma 15** *For $1 \leq i \leq n$, $\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)] \leq \frac{1}{2}\left(\mathbb{E}[f(X_i) - f(X_{i-1})] + \mathbb{E}[f(Y_i) - f(Y_{i-1})]\right)$.*

**Proof** We prove the lemma conditioned on an arbitrary history of what happened up to iteration $i$: We distinguish the three cases that can happen in each iteration:
**Case 1:** The algorithm picks $u_i$ because $b_i \leq 0$:
**Case 1.1:** $u_i \in OPT$: We need to show $0 \leq \frac{a_i+0}{2}$. We know that $a_i + b_i \geq 0$, so $a_i \geq 0$.
**Case 1.2:** $u_i \notin OPT$: We need to show $b_i \leq \frac{a_i+0}{2}$. We know that $a_i + b_i \geq 0$, so $a_i \geq 0 \geq b_i$.
**Case 2:** The algorithm rejects $u_i$ because $a_i < 0$: Analogous to Case 1.
**Case 3:** $a_i \geq 0$, $b_i > 0$: We assume $u \in OPT$, the case $u \notin OPT$ is similar. The probability that $u_i$ is picked is $\frac{a_i}{a_i+b_i}$, and the probability that it is not picked is $\frac{b_i}{a_i+b_i}$. So we have

$$\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)] = \frac{b_i}{a_i+b_i} a_i = \frac{a_i b_i}{a_i+b_i}$$

and

$$\mathbb{E}[f(X_i) - f(X_{i-1})] + \mathbb{E}[f(Y_i) - f(Y_{i-1})] = \frac{a_i}{a_i+b_i} a_i + \frac{b_i}{a_i+b_i} b_i = \frac{a_i^2 + b_i^2}{a_i+b_i}$$

So we need to show that $a_i b_i \leq \frac{1}{2}\left(a_i^2 + b_i^2\right)$, and we can easily show this by distinguishing the cases $a_i \geq b_i$ and $a_i < b_i$. ■