Lecture 15

Minimization of Submodular Functions in Polynomial Time; Edmond’s Theorem

Nov 19, 2004
Lecturer: Kamal Jain
Notes: Neva Cherniavsky

15.1 Submodular function minimization

\( f : 2^U \rightarrow \{i | i \text{ is a } k\text{-bit integer}\} \)
\( |U| = n \)
\( \forall S, T f(S) + f(T) \geq f(S \cap T) + f(S \cup T) \)
A strongly polynomial algorithm would be polynomial in \( n \). But instead we will have polynomial in \( (n, k) \).

Write integer program. \( S \) is represented by its characteristic 0-1 vector. \( f(x_1, x_2, \ldots, x_n) \), where ever \( f \) is 1 compute \( f \) on that particular \( S \).

\[
\min \quad f(x_1, x_2, \ldots, x_n) \\
\text{s.t.} \quad x_i \in \{0, 1\}
\]

Relax to: \( 0 \leq x_i \leq 1 \). The minimum we’ll give by another program because we don’t know how to interpret \( x_i = 0.8 \) for example.

\[
\min \quad \sum_S \lambda_S f(S) \\
\forall i \quad \sum_{S : i \in S} \lambda_S = x_i \\
\sum_S \lambda_S = 1 \\
\lambda_S \geq 0
\]

Write the \( x \) vector as a convex combination over integer vectors. I.e. \( (0.5, 0.5) \) becomes \( \frac{1}{2}(0, 0) + \frac{1}{2}(1, 1) \). Think of the unit cube. This function is defined on all corner points of the cube initially. Any point inside can be written as a convex combination of the corner points. We try to minimize the convex combination.
Claim 1. The solution of this LP is optimized at integral $x_is$.

In fact, this is true even if $f$ is not submodular.

Proof. Suppose not, i.e., suppose it is minimized at some $f(x_1, x_2, \ldots, x_n)$ and $x_i$ is fractional. Then by definition $f(x_1, x_2, \ldots, x_n) = \sum \lambda_S f(S)$. One of the $f(S)$’s must be smaller than $f(x_1, x_2, \ldots, x_n)$, so $f$ wasn’t minimized. $\square$

Given any number $b$, $f(x_1, x_2, \ldots, x_n) \leq b$. $0 \leq x_i \leq 1 \Rightarrow$ convex.

Since this is a convex program, we can run the ellipsoid algorithm. If we have a feasible solution for $x$, $y$, then we have a solution for $(x + y)/2$. The convex constraints are:

\[
\begin{align*}
f(x_1, x_2, \ldots, x_n) & \leq b \\
f(y_1, y_2, \ldots, y_n) & \leq b \\
0 & \leq x_i \leq 1 \\
0 & \leq y_i \leq 1
\end{align*}
\]

Then there exist $\lambda_S$’s satisfying

\[
\begin{align*}
\sum_S \lambda_S^x f(S) &= f(x_1, x_2, \ldots, x_n) \\
\forall i \sum_{S:i \in S} \lambda_S^x &= x_i \\
\sum_S \lambda_S^x &= 1 \\
\lambda_S^x &\geq 0
\end{align*}
\]

True for $y$ as well. So:

\[
\begin{align*}
\sum_S \lambda_S^x f(S) &\leq b \\
\sum_S \lambda_S^y f(S) &\leq b \\
\sum_{S:i \in S} \lambda_S &= (x_i + y_i)/2
\end{align*}
\]

The minimum $(\lambda_S^x + \lambda_S^y)/2$ will be $\leq b$. So it is also convex.

Claim 2. There exists an optimal solution to this program such that $\forall \; S, T : S \not\subseteq T$ and $T \not\subseteq S$ then either $\lambda_S = 0$ or $\lambda_T = 0$.

In other words, the only valid picture for $S$ is concentric circles; you would never have two separate, non-overlapping circles, or two circles with an intersection.

Proof. Among all optimum solutions of this program, pick the one which maximizes $\sum_S \lambda_S |S|^2$. 

2
\[ |S|^2 + |T|^2 \leq |S \cap T|^2 + |S \cup T|^2 \]

Always true. Take maximum, which separates S and T more extremely. We claim this particular optimum solution will satisfy property.

Suppose not. Then \( T_1 \not\subset T_2, T_2 \not\subset T_1 \), and \( \lambda_{T_1} > 0 \) and \( \lambda_{T_2} > 0 \). Take \( \epsilon = \min(\lambda_{T_1}, \lambda_{T_2}) > 0 \).

\[
\lambda_S = \begin{cases} 
\lambda_S & S \neq T_1, S \neq T_2, S \neq T_1 \cap T_2, S \neq T_1 \cup T_2 \\
\lambda_S - \epsilon & S = T_1, S = T_2 \\
\lambda_S + \epsilon & S = T_1 \cap T_2, S = T_1 \cap T_2 
\end{cases}
\]

Can check that all properties of the LP are still satisfied. The objective function is

\[
\min \sum_S \lambda_S f(S) = \min \sum_S \lambda_S f(S) - \epsilon(f(T_1) + f(T_2)) + \epsilon(f(T_1 \cap T_2) + f(T_1 \cup T_2))
\]

The loss must be negative because the objective function was minimal. In other words, \( f(T_1) + f(T_2) \geq f(T_1 \cap T_2) + f(T_1 \cup T_2) \). By submodularity, \( f(T_1) + f(T_2) \leq f(T_1 \cap T_2) + f(T_1 \cup T_2) \). So they must be equal, so \( \lambda' \) also minimized the objective function.

But \( \sum_S \lambda_S|S|^2 \leq \sum_S \lambda'_S|S|^2 \), which contradicts the assumption that \( \lambda_S \) was the solution to the LP that maximized \( \sum_S \lambda_S|S|^2 \).

\( S : \lambda_S > 0 \) are contained in each other. We can write this explicitly:

\[
\begin{align*}
z_1 &= \min_{x_i > 0} x_i & S_1 &= \{i | x_i \geq z_1\} & \lambda_{S_1} &= z_1 \\
z_2 &= \min_{x_i > z_1} x_i & S_2 &= \{i | x_i \geq z_2\} & \lambda_{S_2} &= z_2 - z_1 \\
&\vdots
\end{align*}
\]

\[
\begin{align*}
z_k &= \min_{x_i > z_{k-1}} x_i & S_k &= \{i | x_i \geq z_k\} & \lambda_{S_k} &= z_k - z_{k-1} \\
z_{k+1} &= 1 & S_{k+1} &= \{i | x_i \geq 1\} & \lambda_{S_{k+1}} &= 1 - z_k
\end{align*}
\]

This is a unique solution. We can check the constraints easily.

The process is to choose a \( b \) and pick \( x_i \) to be all 0. If this is infeasible then can run ellipsoid again with smaller bounds (on the ellipsoid). Do a binary search and keep calling ellipsoid until you find a \( b \) such that the program is feasible at \( b \) but not at \( b - 1 \).

We have an oracle that answers at integer points. Put it inside another that answers for fractional values. If the minimization of the program happens at a fractional value, then it happens at an integral value. So if the ellipsoid algorithm returns a fractional value, we know there’s an integral solution and can find it.

### 15.2 Edmond’s theorem

Given a directed graph \( G \) and a root \( r \), an arborescence (branching, rooted directed spanning tree) is a spanning tree that has all edges pointing away from \( r \). You might want an arborescence if you have information
at the root \( r \) and want to send it to all nodes on the graph. The capacity on all edges is one, and \( G \) is allowed to be a multigraph.

An arborescence packing is the maximum number of arborescences \( A_1, A_2, \ldots A_k \) such that all are edge disjoint. They all share the same root.

Define \( \lambda_{ru} \) to be the number of edge disjoint paths from \( r \) to \( u \) in \( G \). \( k \leq \min_u(\lambda_{ru}). \) Or, in other words, if \( \delta_{OUT}(S) \) is the number of outgoing edges of \( S \), then \( k \leq \min_{r \in S, u \in S} \left| \delta_{OUT}(S) \right| \).

**Theorem 15.1.** Edmond’s Theorem: The maximum number of arborescences \( k \) is equal to the minimum cut.

\[
k = \min_u(\lambda_{ru}) = \min_{r \in S, u \notin S} \left| \delta_{OUT}(S) \right|
\]

**Proof.** (Lovasz) Assume \( \min_{r \in S, u \notin S} \left| \delta_{OUT}(S) \right| = c. \) Initially take \( G \). It has minimum cut \( c_G. \) Pick an arborescence \( A_1 \) such that \( C(G - A_1) = c_G - 1. \) By induction we can keep going down, creating \( A_{C(G)} \) number of arborescences. Then \( k = C(G). \)

We need to prove the inductive step, that we can find \( A_1 \) with this property. Initially \( A_1 \) has a single vertex, root \( r. \) We create edges, maintaining the property that \( C(G - A) \geq C(G) - 1. \) We can keep picking edges and unless \( A_1 \) becomes spanning, we can always maintain this property. Call vertices spanned by \( A \) \( V(A). \) Need to maintain

\[
\forall S \neq S \neq V(G) \quad |\delta_{OUT}^{G-A}(S)| \geq C(G) - 1
\]

**Definition 15.1.** A critical set \( S \) satisfies

1. \( \delta_{OUT}^{G-A}(S) = C(G) - 1 \)
2. \( V(G) - V(A) \notin S; \exists u \in S \) but \( u \in V(G) - V(A) \)
3. \( r \in S \)

\( V(G) - V(A) \) are points outside the arborescence. We don’t want to pick an edge for which \( d_{OUT}^{G-A}(S) = C(G) - 1, \) because then we’ll have a problem.

Take any maximal critical set \( S. \) Can it contain all of \( A? \) No, because that would violate 1. It must leave some vertices outside. \( \exists \) point \( v, v \in V(A) \) and \( v \notin S. \) Because of 2, \( \exists \) point \( u, u \notin S, u \in V(G) - V(A). \)

We will show something stronger: \( \exists u, u \notin S, u \in V(G) - V(A), \) and \( v - u \) is an edge in \( G. \) This is the edge we will pick. We prove this existence using maximality.

\[
\delta_{OUT}^{G-A}(S \cup \{v\}) = C(G)
\]

Since \( S \) is maximal, it must violate one of 1, 2, 3; it still satisfies 2, 3 so it must violate 1. When we include \( v \) inside, there must be an edge in \( G - A \) that it goes to. If it doesn’t go to \( G - A, \) throw it away. If it goes to \( G - A, \) then include it and the induction works.

This won’t harm any other critical set \( T. \) Why? Suppose not. Assume \( T \) is hurt by removing \( v. \)

\[
|\delta_{OUT}(S)| + |\delta_{OUT}(T)| \geq |\delta_{OUT}(S \cap T)| + |\delta_{OUT}(S \cup T)| \text{ by submodularity.}
\]

\[
|\delta_{OUT}(S)| = C(G) - 1
|\delta_{OUT}(T)| = C(G) - 1
\]

The other ones must also be \( C(G) - 1 \) because that’s what we’re maintaining inductively (can’t be \( C(G) - 2. \) So \( T \) cannot be violated, because \( S \cup T \) is also a critical set. But \( S \) was maximal, so can’t hurt it. \( \square \)