Linear Programming Approximations

1 Generalization of Maximum Coverage Approximation

We will see that the same approximation we used last lecture to solve maximum coverage leads to a $(1-\frac{1}{e})$ approximation for any non-negative monotone submodular function.

MAXIMUM COVERAGE: Given a universe U where each element has a value $v(u)$, subsets $S_1,\ldots,S_m \subseteq U$, and an integer k, find a $T \subseteq \{1,\ldots,m\}$ with $|T| \leq k$ maximizing $\sum_{u\in\bigcup_{j\in T} s_j} v(u)$.

Let function $f : 2^U \to \mathbb{R}$; in other words, $f : T \to \sum_{u\in\bigcup_{j\in T} s_j} v(u)$. Assume $f(T) \geq 0\ \forall T$.

Monotonicity: $f(S) \geq f(T)$ when $S \supseteq T$.

Submodularity (diminishing returns): If $S \supseteq T$, then $f(T\cup\{x\}) - f(T) \geq f(S\cup\{x\}) - f(S)\ \forall x$. More elegantly, $f(S\cap T) + f(S\cup T) \leq f(S) + f(T)$.

Recall key insight 2 from last lecture: by adding the best set S_j for $j \in T^*$ to T_t, we increase the value by at least $\frac{1}{k}(OPT - V_t)$ because $|T^*| \leq k$.

Lemma 1. *If f is non-negative, submodular, and monotone, then this insight holds.*

Proof. Let $S = T^* \setminus T_t$; $S = \{u_1,\ldots,u_r\}$ with $r \leq k$. Define $S_j := \{u_1,\ldots,u_j\}$ and recall $f(T) = \sum_{u\in\bigcup_{j\in T} s_j} v(u)$. We are interested in:

$$f(T^*\cup T_t) - f(T_t)$$

$$= f(S\cup T_t) - f(T_t)$$

We use a telescoping sum:

$$= \sum_{j=0}^{r-1} f(S_{j+1}\cup T_t) - f(S_j\cup T_t)$$

$$= \sum_{j=0}^{r-1} f(S_j\cup T_t\cup\{u_{j+1}\}) - f(S_j\cup T_t)$$

Applying submodularity:

$$\leq \sum_{j=0}^{r-1} f(T_t\cup\{u_{j+1}\}) - f(T_t)$$

We use this fact to prove the previous insight:

$$\max_{j=1\ldots r} f(T_t \cup \{u_j\}) - f(T_t) \geq \frac{1}{r}\sum_{j=1}^{r} f(T_t \cup \{u_j\}) - f(T_t)$$

$$\geq \frac{1}{r}(f(T^* \cup T_t) - f(T_t))$$

And because $r \leq k$:

$$\geq \frac{1}{k}(f(T^* \cup T_t) - f(T_t))$$

$\square$

Theorem 2. *Let f be a non-negative, monotone, submodular function. Then, the greedy algorithm which for k iterations always adds the element giving largest increase in f is a $1 - \frac{1}{e}$ approximation (Nemhauser, Halsey, Fisher).*

In the greedy algorithms unit, we proved that if f is **modular** (i.e., linear), maximizing $f(S)$ subject to S being an independent set in a given matroid is solved optimally by the greedy algorithm. The implications of Theorem 2 are that if f is non-negative, monotone, and submodular, the greedy algorithm is a $1 - \frac{1}{e}$ approximation on the k-uniform matroid (where any set of $\leq k$ elements is independent).

Theorem 3. *The greedy algorithm for a non-negative, monotone, submodular function on an arbitrary matroid is a $\frac{1}{2}$-approximation (Nemhauser).*

Theorem 4. *There is a "continuous-greedy" (think gradient descent) algorithm which achieves a $1 - \frac{1}{e}$ approximation for any non-negative, monotone, submodular function on any matroid (Vondrak).*

2 Intro to Linear Programming Approximations

Many optimization problems can be encoded naturally as integer linear programs (ILPs). Solving ILPs is **NP**-hard because many problems are naturally encoded. The generic technique for achieving bounds on OPT:

1. Create the ILP encoding the problem.
2. "Relax" the ILP to an LP by removing integrality constraints.
3. Now the LP can be solved in polynomial time to give a fractional solution $\vec{x}$.
4. Round $\vec{x}$ to an integer solution $\hat{x}$ and ensure it is a valid solution (use $\vec{x}$ as guidance for computing $\hat{x}$).
5. Prove $\hat{x}$ is not too much worse than $\vec{x}$.

Important bounds:

$$OPT_{LP} \leq OPT_{ILP} \text{ for minimization}$$

$$OPT_{LP} \geq OPT_{ILP} \text{ for maximization}$$

Corollaries:

$$\text{If } cost(\hat{x}) \leq \alpha * cost(\vec{x}), \text{ then } \hat{x} \text{ is an } \alpha\text{-approximation for minimization.}$$

If $cost(\hat{x}) \geq \alpha * cost(\vec{x})$, then $\hat{x}$ is an α-approximation for maximization.

The **integrality gap** of an LP is, over all instances:

$$\min \frac{OPT_{ILP}}{OPT_{LP}} \text{ for minimization}$$

$$\max \frac{OPT_{ILP}}{OPT_{LP}} \text{ for maximization}$$

If analysis only uses LP bound, it can prove no better approximation guarantees than the integrality gap.

3 Weighted Vertex Cover

3.1 Building an LP

WEIGHTED VERTEX COVER: Given a graph $\mathcal{G} = (V, E)$ with vertex costs c_v, find a vertex cover S of minimum total cost $\sum_{u \in S} c_v$.

We write an ILP encoding the problem:

Maximize

$$\sum_v c_v x_v$$

subject to

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

We relax the ILP to an LP by removing the $x_v \in \{0, 1\}$ constraint. Solving this LP gives a fractional solution $\vec{x}$.

3.2 Rounding Algorithm

We include in S all nodes v with $x_v \geq \frac{1}{2}$.

(1). This is a vertex cover: because of the first constraint, at least one endpoint of e has $x_v \geq \frac{1}{2}$ $\forall e$. Such a v is in S.

(2). Cost analysis:

$$cost(\hat{x}) = \sum_{v \in S} c_v = \sum_{v : x_v \geq \frac{1}{2}} c_v$$

$$\leq \sum_{v : x_v \geq \frac{1}{2}} (2x_v) c_v$$

$$\leq \sum_v (2x_v)c_v$$

$$= 2 * cost(\vec{x})$$

So this is a 2-approximation.

3.3 Integrality Gap

The worst case scenario is a complete graph with all $c_v = 1$. Then, $LP-OPT \leq \frac{n}{2}$ and $ILP-OPT = n-1$. This ratio is about 2, so we know that the rounding can't be better than a 2-approximation.