CAS CS 599 B: Mathematical Methods for TCS

Lecturer: Mark Bun Spring 2022

Lecture Notes 22:

Duality, Semidefinite Programming, Max Cut

Reading.

- Useful introduction to linear programming: "Understanding and using linear programming," Matoušek and Gärtner.
- Goemans and Williamson, .879-approximation algorithms for Max Cut and Max 2SAT.

1 Duality

A key part of the theory of linear programming is duality. The clearest way to see what's going on is in the case of feasibility (decision) LPs, for example: Given a constraint region $K = \{\vec{x} \mid A\vec{x} \leq \vec{b}\}$ is K nonempty? As an example, consider the system

$$\begin{aligned} 2x_1 - 2x_2 + x_3 &\leq 1 \\ 2x_1 + x_2 + x_3 &\leq 3 \\ -2x_1 + x_2 - x_3 &\leq -2 \end{aligned}$$

I want to convince you that this LP is infeasible. To do so, suppose we multiply the first inequality by $y_1 = 2$, the second inequality by $y_2 = 1$, and the third by $y_3 = 3$. Then adding up the inequalities, we get

$$0x_1 + 0x_2 + 0x_3 \leq -1$$

which is a contradiction. In other words, the linear combination of constraints specified by $\vec{y} = (2, 1, 3)$ certifies that the LP is infeasible. Amazingly, *every* infeasible LP can be certified by such a linear combination of its constraints.

Theorem 1 (Farkas' Lemma)**.** *Let $A \in \mathbb{R}^{m \times n}$. Exactly one of the following holds:*

- *The system $A\vec{x} \leq \vec{b}$ has a solution $x \in \mathbb{R}^n$, or*
- *There exists $\vec{y} \geq 0$ such that $A^T\vec{y} = \vec{0}$ and $\langle \vec{b}, \vec{y} \rangle < \vec{0}$.*

This captures the general idea of LP duality, but note that there are many different statements corresponding to the various possible formulations of the original LP. If you understand the statement of Farkas' Lemma, you understand what's going on, but there's no shame in just looking up the fairly mechanical procedure for converting from an LP to its dual.

For example,

Theorem 2. *The following linear programs have the same objective value:*

$$\begin{aligned} L := \max \quad & \langle \vec{c}, \vec{x} \rangle \\ \text{s.t.} \quad & A\vec{x} \leq \vec{b} \\ & \vec{x} \geq \vec{0} \end{aligned} \qquad\qquad \begin{aligned} L^{\perp} := \min \quad & \langle \vec{b}, \vec{y} \rangle \\ \text{s.t.} \quad & A^T\vec{y} \geq \vec{c} \\ & \vec{y} \geq \vec{0} \end{aligned}$$

It's usually easy to verify the optimum of the dual LP is an *upper bound* on the upper bound of the primal LP. That is, every feasible solution to $L^{\perp}$ certifies an upper bound on the objective of L. For the formulation above, we can see this by calculating, for every $\vec{x}$ and $\vec{y}$ that are primal and dual feasible, respectively:

$$\langle \vec{c}, \vec{x} \rangle \leq \langle A^T\vec{y}, \vec{x} \rangle = \langle \vec{y}, A\vec{x} \rangle \leq \langle \vec{y}, \vec{b} \rangle.$$

Given an LP, you usually uncover something interesting (or get something for free) by taking the dual. Recall our example of an LP for bipartite matching.

$$\begin{aligned} R := \max \quad & \sum_{(\ell,r)\in E} x_{\ell,r} & \\ \text{s.t.} \quad & \sum_{r\sim\ell} x_{\ell,r} \leq 1 & \forall \ell \in L \\ & \sum_{\ell\sim r} x_{\ell,r} \leq 1 & \forall r \in R \\ & x_{\ell,r} \geq 0 & \forall (\ell, r) \in E \end{aligned}$$

The dual (optimization) LP, after some simplification, is the minimization problem

$$\begin{aligned} R^{\perp} := \min \quad & \sum_{\ell\in L} y_\ell + \sum_{r\in R} y_r & \\ \text{s.t.} \quad & y_\ell + y_r \geq 1 & \forall (\ell, r) \in E \\ & \vec{y} \geq \vec{0}. & \end{aligned}$$

Here, (strong) LP duality tells is that the minimum value of the dual LP $R^{\perp}$ is exactly the maximum value of the primal LP R. If we had integrality constraints on $R^{\perp}$, it would have a natural combinatorial interpretation: Think of y_ℓ or $y_r = 1$ as corresponding to including that vertex in a set of vertices. The constraints require that every edge in the graph is incident to one fo the vertices in this set. So the problem here is exactly to find a minimum size vertex cover for the bipartite graph.

The general fact that totally unimodular matrices have integral basic feasible solutions means that the minimum of $R^{\perp}$ is attained at an integral point as well. So we conclude that the minimum size of vertex cover of a bipartite graph is exactly the maximum size of a matching.

2 Ellipsoid Algorithm

Khachiyan showed that linear programming can be solved in polynomial time by analyzing the ellipsoid algorithm. It solves the following promise version of LP feasibility. Given a feasible region K under the promise that $B_r \subseteq K \subseteq B_R$ for some balls B_r and B_R of radius r and R, the goal is to find a point in K. The algorithm works roughly as follows.

- Initialize an ellipsoid E containing B_R with center $\vec{x_0}$.
- Test if $\vec{x_0} \in K$. If so, output $\vec{x_0}$.
- Otherwise, let H be a hyperplane that separates $\vec{x_0}$ from K. Update E to be the smallest ellipsoid containing the part of E on the correct side of H.

It can be shown that the shrinking step reduces the volume of E by a multiplicative factor of $1 - \Omega(1/n)$, so after running for $\text{poly}(n) \cdot \log(R/r)$ steps, the algorithm either finds a point or shrinks the bounding ellipsoid so small it couldn't contain a ball of radius r.

So how do we find the separating hyperplane H? If $\vec{x_0}$ is infeasible, then it must violate some constraint, i.e., $\langle \vec{a}, \vec{x_0} \rangle > b$, but $\langle \vec{a}, \vec{x} \rangle \leq b$ for every $x \in K$. This constraint specifies the separating hyperplane.

Note that this algorithm solves a more general problem than linear programming. The only thing it needs is a "separation oracle" for K, i.e., a subroutine that takes as input a point $x \in \mathbb{R}^n$ and either determines whether $x \in K$, or outputs a hyperplane separating x from K.

3 Max Cut, Revisited

Given an undirected graph $G = (V = [n], E)$ and a subset S of vertices, recall $\partial S = \{(i,j) \mid i \in S, j \notin S\}$. The MAX-CUT problem is to find a set S maximizing $|\partial S|$.

Way back in Lecture 8, we saw that including each vertex u in S independently with probability $1/2$ cuts half the edges in expectation, so it gives a $1/2$-approximation to the largest cut. We also saw how to use pairwise independence to give a poly-time deterministic $1/2$-approximation algorithm.

One can try to design an approximation algorithm for Max Cut by formulating an LP relaxation. Here's one potential way to do it:

$$\begin{aligned} \max \quad & \sum_{(i,j) \in E} x_{ij} & \\ \text{s.t.} \quad & 0 \leq x_{ij} \leq 1 & \forall (i,j) \in E. \end{aligned}$$

This doesn't quite work. The standard relaxation actually adds constraints, and corresponds to optimization over the so-called metric polytope:

$$\begin{aligned} \max \quad & \sum_{(i,j) \in E} x_{ij} & \\ \text{s.t.} \quad & 0 \leq x_{ij} \leq 1 & \forall (i,j) \in E \\ & x_{ij} \leq x_{ik} + x_{kj} & \forall i,j,k \\ & x_{ij} + x_{ik} + x_{kj} \leq 2 & \forall i,j,k. \end{aligned}$$

This can be shown to give a $1/2$-approximation. It turns out that linear programming relaxations get stuck at $1/2$: Chan, Lee, Raghavendra, and Steurer showed that any polynomial-size LP for Max Cut has an integrality gap of $1/2$.

Delorme and Poljak had the idea to look at a different formulation and relaxation of the problem. First, here is a quadratic integer program that exactly captures the Max Cut problem.

$$\begin{aligned} \max \quad & \sum_{(i,j)\in E} \frac{1}{2} - \frac{1}{2} x_i x_j \\ & \text{s.t. } x_{ij} \in \{-1, 1\} \forall (i,j) \in E. \end{aligned}$$

Here, you should think of variables $x_i = -1$ as corresponding to including vertex i in the cut. Equivalently, this can be formulated as

$$\begin{aligned} \max \quad & \sum_{(i,j)\in E} \frac{1}{2} - \frac{1}{2} y_{ij} & \\ \text{s.t.} \quad & y_{ii} = 1 & \forall i \in V \\ & \exists (x_i)_{i\in V} \qquad \text{s.t. } y_{ij} = x_i x_j \forall (i,j) \in E. & \end{aligned}$$

The idea of the relaxation is now to try to enforce the last condition using (infinitely many) linear constraints. Let Y be the symmetric matrix such that $Y_{ij} = y_{ij}$. The last condition implies that $Y = x^T x$ which is a rank-one PSD matrix. The rank-one condition seems hard to enforce, but we can at least enforce positive semidefiniteness: The condition $v^T Y v \geq 0$ for all v translates into the (infinitely many) linear inequalities $\sum_{i,j} v_i v_j y_{ij} \geq 0$ for all $v \in \mathbb{R}^n$.

This suggests the relaxation

$$\begin{aligned} \max \quad & \sum_{(i,j)\in E} \frac{1}{2} - \frac{1}{2} y_{ij} & \\ \text{s.t.} \quad & y_{ii} = 1 & \forall i \in V \\ & Y = (y_{ij}) & \text{is PSD.} \end{aligned}$$

Even though we can think of this as an LP with infinitely many linear constraints, it is still solvable in polynomial time using the ellipsoid algorithm. This is because we can design a separation oracle that either reports that Y is PSD, or finds a vector v such that $v^T Y v < 0$, corresponding to a separating hyperplane. In general, a linear program over $n(n+1)/2$ variables y_{ij} with the extra constraint that Y is PSD is called a semidefinite program (SDP).

Now using the characterization that a matrix $Y \in \mathbb{R}^{n\times n}$ is PSD iff there exist vectors $\vec{u}_1, \ldots \vec{u}_n \in \mathbb{R}^n$ such that $y_{ij} = \langle \vec{u}_i, \vec{u}_j \rangle$, we get the equivalent formulation

$$\begin{aligned} \max \quad & \sum_{(i,j)\in E} \frac{1}{2} - \frac{1}{2} \langle \vec{u}_i, \vec{u}_j \rangle & \\ \text{s.t.} \quad & \langle \vec{u}_i, \vec{u}_i \rangle = 1 & \forall i \in V \\ & \vec{u}_i \in \mathbb{R}^n & \end{aligned}$$

or equivalently

$$\begin{aligned} \max \quad & \sum_{(i,j)\in E} \frac{1}{2} - \frac{1}{2} \langle \vec{u}_i, \vec{u}_j \rangle \\ \text{s.t.} \quad & \|\vec{u}_i\|_2 = 1. \end{aligned}$$

Note that this is the original quadratic program capturing Max Cut, but where we've replaced each Boolean variable x_i with a unit vector $\vec{u}_i \in \mathbb{R}^n$. Even though this program looks more complicated, it can be solved in polynomial time, even though the original problem is NP-hard.

The final ingredient in the Goemans-Williamson algorithm for Max Cut is to round the SDP solution back to a cut. This is done via the following hyperplane rounding strategy. Pick a random halfspace $h_{\vec{r}}(\vec{v}) = \operatorname{sgn}(\langle \vec{r}, \vec{v} \rangle)$ through the origin and set $S = \{i \mid h_{\vec{r}}(\vec{u}_i) = -1\}$. Now we have

$$\begin{aligned}
\mathbb{E}[|\partial S|] &= \sum_{(i,j) \in E} \Pr_{\vec{r}}[h_{\vec{r}}(\vec{u}_i) \neq h_{\vec{r}}(\vec{u}_j)] \\
&= \sum_{(i,j) \in E} \frac{\theta(\vec{u}_i, \vec{u}_j)}{\pi} \\
&\geq \sum_{(i,j) \in E} C \cdot \left(\frac{1}{2} - \frac{1}{2} \langle \vec{u}_i, \vec{u}_j \rangle \right) \\
&= C \cdot \text{SDPOpt}
\end{aligned}$$

where θ is the angle between two vectors and

$$C = \min_{0 \leq y \leq \pi} \frac{y}{\pi} \frac{2}{1 - \cos y} \approx 0.87856.$$

Thus, the Goemans-Williamson relaxation gives a 0.878-approximation to Max Cut.