Lecture 22: Copositive Programming

The following claim was used in the proof of the Motzkin-Straus theorem.

Claim: $\sum_{i=1}^{k} x_i^2$ is minimized subject to $\sum_{i=1}^{k} x_i = 1$ when all the x_i-values are equal to $\frac{1}{k}$.

Proof: Minimizing $\sum_{i=1}^{k} x_i^2$ subject to $\sum_{i=1}^{k} x_i = 1$ is equivalent to minimizing $\sum_{i=1}^{k-1} x_i^2 + \left(1 - \sum_{i=1}^{k-1} x_i\right)^2$.

Let $h(x_1, \ldots, x_{k-1}) = \sum_{i=1}^{k-1} x_i^2 + \left(1 - \sum_{i=1}^{k-1} x_i\right)^2$

$$= 2\left(x_1^2 + \ldots + x_{k-1}^2\right) + 2\sum_{i<j} x_i x_j - 2\left(x_1 + \ldots + x_{k-1}\right) + 1$$

∇h = the gradient of h $= \left(\frac{\partial h}{\partial x_1}, \frac{\partial h}{\partial x_2}, \ldots, \frac{\partial h}{\partial x_{k-1}}\right)$.

Observe that $\frac{\partial h}{\partial x_i} = 4x_i + 2\sum_{j \neq i} x_j - 2$

$$= 2\left(x_i - x_k\right).$$

To find the minimizer, we need to solve for $\nabla h = 0$. So $x_i = x_k$ for $1 \leq i \leq k-1$.

So the only critical point for h is $\left(\frac{1}{k}, \frac{1}{k}, \ldots, \frac{1}{k}\right)$.

Easy to check that this point is a minimizer.

Thus $\sum_{i=1}^{k} x_i^2$ is minimized subject to $\sum_{i=1}^{k} x_i = 1$ when all the x_i-values are equal to $\frac{1}{k}$. ▨

This finishes the proof of the Motzkin-Straus thm. We'll now see a complexity result that follows from the ~~proof of the~~ Motzkin-Straus theorem.

Theorem. Given an integer matrix $M \in SYM_n$, it is coNP-complete to decide whether $M \in COP_n$.

Proof If $M \notin COP_n$ then there exists $x \in \mathbb{R}^n_+$ such that $x^T M x < 0$. However we should be careful here — we need to show there exists such an $x \in \mathbb{R}^n_+$ whose binary encoding length is polynomial in the binary encoding length of M. In other words, we need to show a compact certificate of M's non-membership in COP_n. (Please do this as an exercise.)

To show this problem $M \notin COP_n$ is NP-hard, we'll show a reduction from the independent set problem: given a graph G and an integer k, does G have an independent set of size $> k$?

Claim: The matrix $k I_n + k A_G - J_n \notin COP_n$ if and only if $\alpha_G > k$.

So G has an independent set of size larger than k $\iff$ the above matrix is not in COP_n. ▨

Proof of the claim. Let $M = k I_n + k A_G - J_n$. If $M \in COP_n$ then $x^T (k I_n + k A_G - J_n) x \geq 0$ for all $x \in \mathbb{R}^n_+$. Thus $k \cdot x^T (I_n + A_G) x \geq x^T J_n x = 1$ for all $x \in \mathbb{R}^n_+$ and $\sum_{i=1}^n x_i = 1$.

That is, $\min \left\{ x^T (I_n + A_G) x : x \in \mathbb{R}^n_+, \sum_i x_i = 1 \right\} \geq 1/k$.

Since the value on the left equals $\frac{1}{\alpha_G}$ (by Motzkin-Straus theorem), we have $k \geq \alpha_G$.

Thus $M \in COP_n \Rightarrow k \geq \alpha_G$.

Conversely, if $k \geq \alpha_G$ then $\min\{x^T(I_n + A_G)x : x \in \mathbb{R}^n_+ \text{ and } \sum_{i=1}^n x_i = 1\} = \frac{1}{\alpha_G} \geq \frac{1}{k}$.

As seen in the previous lecture, this implies that $x^T M x \geq 0$ for $x \in \mathbb{R}^n_+$.

Thus $M \in COP_n$.

Hence $M \notin COP_n \iff G$ has an independent set of size larger than k. (since $\alpha_G > k$) ▨

Colorings with Low Discrepancy

Let $V = \{1, 2, \ldots, n\}$ and let $\mathcal{F} = \{S_1, \ldots, S_m\}$ be a family of subsets of V.

The basic problem in combinatorial discrepancy theory is to color each $i \in V$ either red or blue so that each of the sets in $\mathcal{F}$ has roughly the same number of red elements and blue elements.

Suppose $\mathcal{F} = 2^V$, i.e, $\mathcal{F}$ consists of all subsets of V. Then no matter how we color the elements of V, there is always a completely monochromatic set of size $\geq n/2$.

Definition. Let $\chi : V \to \{-1, +1\}$ be a coloring and for any $S \subseteq V$, let $\chi(S) = \sum_{i \in S} \chi(i)$. For $\mathcal{F} \subseteq 2^V$, let $\text{disc}(\mathcal{F}, \chi) = \max_{S \in \mathcal{F}} |\chi(S)|$. The discrepancy of $\mathcal{F}$ is

$$\text{disc}(\mathcal{F}) = \min_{\chi} \text{disc}(\mathcal{F}, \chi).$$

If +1's are red and −1's are blue, then $\chi(S)$ is the number of red elements in S minus the number of blue elements in S — we will call this the imbalance of S under χ.

We will consider the algorithmic problem of computing a low-discrepancy coloring for a given set system. Discrepancy can be regarded as a measure of how complex a set system is. But it is not very well-behaved.

For example, let $\mathcal{F} = 2^V$ and let V' be a disjoint copy of V and let $\mathcal{F}' = 2^{V'}$. The set system $(V \cup V', \{S \cup S' : S \in \mathcal{F}\})$ has discrepancy 0, yet we feel that it is as complex as $\mathcal{F}$.

A better behaved measure is the hereditary discrepancy:

$$\text{herdisc}(\mathcal{F}) = \max_{A \subseteq V} \text{disc}(\mathcal{F}|_A),$$

where $\mathcal{F}|_A$ is the restriction of the set system $\mathcal{F}$ to the ground set A, i.e., $\mathcal{F}|_A = \{A \cap S : S \in \mathcal{F}\}$.

In other words, the adversary selects a subset $A \subseteq V$ and we must color its elements red or blue so that each set in $\mathcal{F}$ is balanced; the elements outside A do not count.

We'll see the following result here.

Theorem. There is a randomized polynomial-time algorithm which, for an input set system $\mathcal{F}$ on n elements, with m sets, and with hereditary discrepancy at most t, computes a coloring χ with $\text{disc}(\mathcal{F}, \chi) = O(t \cdot \log(mn))$.

The above algorithm, which will be called Bansal's algorithm, is based on SDP.

Bansal's algorithm. The desired coloring will be obtained through a sequence of semicolorings.

A semicoloring is a mapping $c: V \to [-1,1]$. So a semicoloring is like a coloring but by real numbers in $[-1,1]$. We define $\mathrm{disc}(\mathcal{F}, c)$ as $\max_{S \in \mathcal{F}} \left| \sum_{j \in S} c(j) \right|$.

- The algorithm starts with the semicoloring $x_0 = 0$.
- Then it produces a sequence
$$x_0, x_1, \ldots, x_\ell \in [-1,1]^n,$$
of semicolorings. We'll see later what ℓ should be.
- With probability close to 1, the final semicoloring x_ℓ is actually a coloring, i.e., all coordinates are ± 1's. This will be the output of the algorithm.
- If x_ℓ is not a coloring then we run the algorithm all over again.

The t-th step. Here x_{t-1} is updated to x_t as follows. First we generate an increment $\Delta_t \in \mathbb{R}^n$. A "tentative value" of x_t is $\tilde{x}_t = x_{t-1} + \Delta_t$. However we need to ensure that each coordinate is in $[-1,1]$. So we do the following:

$$(x_t)_j = \begin{cases} +1 & \text{if } (\tilde{x}_t)_j \geq 1, \\ -1 & \text{if } (\tilde{x}_t)_j \leq -1, \text{ and} \\ (\tilde{x}_t)_j & \text{otherwise.} \end{cases}$$

How do we generate the increment Δ_t? Let $A_t = \{j \in V : (x_{t-1})_j \neq \pm 1\}$ be the set of coordinates that are still active in the t-th step. We will make sure that $(\Delta_t)_j = 0$ for all $j \notin A_t$.

- Via semidefinite programming, we compute a coloring of the elements in A_t by unit vectors (rather than ± 1's).

  More explicitly, we compute $u_{t,j}$ where $j \in A_t$ so that

$$\left\| \sum_{j \in S_i \cap A_t} u_{t,j} \right\|^2 \leq D^2 \quad \text{for } i = 1, \dots, m$$

$$\text{and} \quad \|u_{t,j}\|^2 = 1,$$

  with $D \geq 0$ as small as possible.

  Note that $u_{t,j} = 0$ for $j \notin A_t$.

- Similar to the Goemans-Williamson algorithm, we generate a random vector r_t from the n-dimensional normal distribution. That is, the coordinates of r_t are independent $N(0,1)$ random variables.

- Then we set $(\Delta_t)_j = s \cdot r_t^T u_{t,j}$ for $j = 1, \dots, m$.

  Here s is a small parameter.

- The length l of the sequence of semicolorings $x_0, x_1, \dots, x_l$ will be set to $\frac{C \cdot \log n}{s^2}$ for a suitable constant C. (Note that $s = \Theta\left(\frac{1}{n\sqrt{\log n}}\right)$)

can be regarded as a random walk in the cube $[-1,1]^n$

This concludes the description of Bansal's algorithm.