Semidefinite Programs

Theo Diamandis

February 4, 2023

In this lecture, we will very briefly look at semidefinite programs (SDPs) and one of their important use cases: relaxations of nonconvex problems. However, these problems come up in a variety of contexts that we don't have time to cover in this course. Convex functions that can be represented using positive semidefinite constraints include the largest and smallest eigenvalues of a matrix, the nuclear and operator norms of a matrix, and the quantum entropy function.[1] This lecture draws from MIT's 6.256 (which I highly recommend if you're interested in state-of-the-art SDP use cases, like sum-of-squares programming) and Stanford's EE364b.

1 Positive Semidefinite (PSD) Matrices

First, We briefly review some properties of positive semidefinite matrices. All of the following are equivalent:

- The (symmetric) matrix $X \in \mathbf{S}^n$ is positive semidefinite (PSD), which we also denote as $X \succeq 0$.
- All eigenvalues of X are nonnegative
- For any vector $y \in \mathbf{R}^n$, $y^T X y \geq 0$
- There exists a factorization $X = B^T B$ for some $B \in \mathbf{R}^{r \times n}$, where $r \leq n$.

Recall from the lecture on convex sets and functions that the set of positive semidefinite matrices is a convex set. Additionally, we can show that the set of positive semidefinite matrices is a cone[2] directly from the fact that X is PSD if and only if $y^T X y \geq 0$ for all vectors y. Finally, we will define the inner product on the PSD matrices as the trace:

$$\langle X, Y \rangle = \mathbf{tr}(X^T Y) = \mathbf{tr}(XY),$$

[1] You can use these function directly in `Convex.jl`, and it will form an SDP under the hood.

[2] A set S is a cone if for every $x \in S$ and $\theta \in \mathbf{R}_+$, $\theta x \in S$.

where the last equality follows from the fact that X and Y are symmetric. Importantly, every linear function of a PSD matrix X can be written as a trace:

$$\sum_{ij} C_{ij} X_{ij} = \mathbf{tr}(CX),$$

where we take C to be symmetric without loss of generality.

2 Semidefinite programming

A semidefinite program (SDP) is simply a convex optimization problem with a linear objective, linear constraints, and a semidefinite constraint:

$$\begin{array}{ll} \text{minimize} & b^T y \\ \text{subject to} & \sum_{i=1}^{m} A_i y_i + C \preceq 0. \end{array}$$

The form above is called the linear matrix inequality form of an SDP. The more common 'standard form' is

$$\begin{array}{ll} \text{maximize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\ & X \succeq 0. \end{array}$$

(Recall how similar this looks to the 'standard form' of a linear program.) Sometimes the maximization and minimization in the definitions are flipped. Although the 'standard form' of an SDP only has affine constraints, the constraint $X \succeq 0$ is convex and tractable, so we can compose this PSD constraint with any other convex constraint.

3 Semidefinite relaxations

One important use case of SDPs is in the relaxation of nonconvex problems. As an example, we will revisit the two-way partitioning problem from the duality lecture.

Two way partitioning problem. Recall that two way partitioning problem attempts to partition n items into two sets. Let W_{ij} be the cost of assigning items i and j to the same set and $-W_{ij}$ be the cost of assigning them to different sets. The MAXCUT problem[3] is a famous example of this problem. The optimization problem is

$$\begin{array}{ll} \text{minimize} & x^T W x \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \ldots, n. \end{array} \tag{1}$$

[3] https://en.wikipedia.org/wiki/Maximum_cut

This problem is nonconvex due to the binary constraint on x. However, we can still write the dual function (see the duality lecture for details)

$$g(\nu) = \begin{cases} -\mathbf{1}^T \nu & W + \mathbf{diag}(\nu) \in \mathbf{S}_+^n \\ -\infty & \text{otherwise.} \end{cases}$$

The associated dual problem is

$$\begin{array}{ll} \text{maximize} & -\mathbf{1}^T \nu \\ \text{subject to} & W + \mathbf{diag}(\nu) \succeq 0. \end{array} \tag{2}$$

We recognize the dual problem as a semidefinite program.

Semidefinite relaxation. Another approach to bounding the optimal value of the original problem is to construct a convex relaxation directly. Define $X = xx^T$. We will rewrite the objective as

$$x^T W x = \mathbf{tr}(x^T W x) = \mathbf{tr}(W x x^T) = \mathbf{tr}(WX),$$

where we used the cyclic property of the trace. The constraint that $x_i^2 = 1$ is equivalent to $\mathbf{diag}\left(xx^T\right) = \mathbf{diag}(X) = \mathbf{1}$. Thus, the original problem is equivalent to

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(WX) \\ \text{subject to} & \mathbf{diag}(X) = \mathbf{1} \\ & X \succeq 0 \\ & \mathbf{rank}(X) = 1. \end{array}$$

The rank one constraint means that X has the form xx^T. Unfortunately, this constraint is not convex. We form the convex relaxation by simply dropping it:

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(WX) \\ \text{subject to} & \mathbf{diag}(X) = \mathbf{1} \\ & X \succeq 0. \end{array} \tag{3}$$

Since the feasible set for this problem is no smaller than the feasible set for the original, its objective value is a lower bound on the original problem. In fact, if an optimal solution $X^\star$ is rank one, then it is also a solution to the original problem. In addition, note that the dual of the nonconvex problem is exactly the dual of the semidefinite relaxation.

Recovering a feasible solution. In general, the solution X to the semidefinite relaxation (3) is not feasible in the original problem (1). One way to recover a solution is to generate samples $i = 1, \ldots, N$ as

$$x^{(i)} \sim \mathcal{N}(0, X^\star),$$

then take $\hat{x}^{(i)} = \mathbf{sign}(x^{(i)})$. We use the solution that has the lowest cost among the generated solutions. For some particular $\hat{x}$, we have

$$\mathbb{E}[\hat{x}_i \hat{x}_j] = \frac{2}{\pi} \arcsin(X_{ij}).$$

This means that

$$\mathbb{E}\left[\hat{x}^T W \hat{x}\right] = \frac{2}{\pi} \mathbf{tr}(W \arcsin(X_{ij})).$$

With a modest number of samples, one of our samples will have an objective value below the expectation with high probability. Naturally, we may wonder how good this Gaussian randomization does. Let's define a few quantities:

- $p^{\text{lb}} = \mathbf{tr}(WX^\star)$, the optimal solution to the SDP, which lower bounds (1).
- $p^{\text{expected}} = \mathbb{E}\left[\hat{x}^T W \hat{x}\right] = \frac{2}{\pi} \mathbf{tr}(W \arcsin(X_{ij}))$, where we generate $\hat{x}$ as described above.
- $p^\star$, the true optimal value for (1).
- $p^{\text{ub}} = \hat{x}^T W \hat{x}$, a feasible value for (1) for a particular $\hat{x}$.

We have that

$$p^{\text{lb}} \leq p^\star \leq p^{\text{expected}}.$$

This means that the true optimal value is between $\frac{2}{\pi} \mathbf{tr}(W \arcsin(X^\star))$ and $\mathbf{tr}(WX^\star)$, and we can generate a $\hat{x}$ that gives a p^{ub} below the expected value p^{expected} without too much effort. For more guarantees about these types of results, see Goemans and Williamson's work [GW95], which provides a very good constant factor bound for the MAXCUT problem, and Nesterov's [Nes98] generalization to a larger class of problems.

4 Cones and generalized inequalities

In this section, we (very briefly) discuss convex cones, which allows us to write *conic formulations* of convex optimization problems. This is the problem form handled by most convex optimization solvers; `Convex.jl` is essentially just a tool to translate problems from a 'natural' formulation into a conic one (and helps us avoid making mistakes in the process!).

Convex cones. Recall that a convex cone is a set K such that for all $x, y \in K$ and θ_1, $\theta_2 \in \mathbf{R}_+$, the point $\theta_1 x + \theta_2 y$ is also in K. Some (somewhat silly) examples are

- reals $K = \mathbf{R}^n$.
- zero cone $K = \{0\}^n$.

We call a convex cone *proper* if it is closed (contains its boundary), solid (non-empty interior), and pointed (contains no line). Some examples are

- nonnegative orthant $K = \mathbf{R}^n_+$.
- second-order cone $K = \{(x,t) \in \mathbf{R}^n \times \mathbf{R} \mid \|x\|_2 \le t\}$.
- positive semidefinite cone $K = \mathbf{S}^n_+$.

In some sense, the proper cones correspond to our intuition of what a cone should look like.

Generalized inequalities. Equipped with a proper cone K, we can define a *generalized inequality*

$$x \preceq_K y \iff y - x \in K.$$

For example, if $K = \mathbf{R}^n_+$, then

$$x \preceq_{\mathbf{R}^n_+} y \text{ means that } x_i \le y_i \text{ for all } i.$$

Since this case is so common, we usually just write $x \le y$. As another example,

$$X \preceq_{S^n_+} Y \text{ means that } Y - X \text{ is PSD.}$$

Again, since this case is so common, we usually just write $X \preceq Y$. Many properties of generalized inequalities are analogous to those of inequalities, but they do not generally define a linear ordering: we can have both $x \not\preceq y$ and $y \not\preceq x$.

General convex optimization problems. We can use generalized inequalities to define convex problems more generally. The problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \preceq_{K_i} 0 \text{ for all } i = 1, \dots, m \\ & Ax = b \end{array}$$

is a convex optimization problem if $f_0 : \mathbf{R}^n \to \mathbf{R}$ is convex and $f_i : \mathbf{R}^n \to \mathbf{R}^{k_i}$ is K_i-convex with respect to proper cone K_i (replace the inequality in the definition of convexity with a generalized inequality using K_i). These problems have the same properties as convex problems, *i.e.,* they can (usually) be efficiently solved, strong duality (often) holds, a locally optimal point is globally optimal, and so on.

Conic formulations. A special case of the above formulation is

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Fx + g \preceq_K 0 \\ & Ax = b, \end{array}$$

where K is potentially a product cone, *i.e.,* $K = K_1 \times K_2 \times \cdots K_p$. This is the 'general form' used by most conic solvers, which support a standard set of cones. Almost all convex optimization solvers support the nonnegative orthant and the second-order cone. Many also support the semidefinite cone, the exponential cone, and the power cone. The job of `Convex.jl` is to transform your problem an equivalent problem of this form and then to transform it back to give you your variable values and optimal value. In terms of the problem classes we've seen,

$$\text{LPs} \subset \text{QPs} \subset \text{SOCPs} \subset \text{SDPs} \subset \text{Conic programs.}$$

Conic programs with the cones listed above cover most of the problems we want to solve in practice. However, it can be advantageous from a modeling and an algorithmic perspective to use additional 'exotic cones', which is an area of current research (see, for example, the `Hypatia.jl` solver [CKV22]).

References

[CKV22] Chris Coey, Lea Kapelevich, and Juan Pablo Vielma. "Solving natural conic formulations with Hypatia. jl". In: *INFORMS Journal on Computing* 34.5 (2022), pp. 2686–2699.

[GW95] Michel X Goemans and David P Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM (JACM)* 42.6 (1995), pp. 1115–1145.

[Nes98] Yu Nesterov. "Semidefinite relaxation and nonconvex quadratic optimization". In: *Optimization methods and software* 9.1-3 (1998), pp. 141–160.