Further motivation for SDP

Michael Kompatscher

25.10.2024

Let us consider the following optimization problem:

$$\text{minimize } \mathbf{v}_1^T\mathbf{v}_1 - \mathbf{v}_1^T\mathbf{v}_3 + 3\mathbf{v}_2^T\mathbf{v}_3, \tag{1}$$
$$\text{subject to } \mathbf{v}_1^T\mathbf{v}_2 + \mathbf{v}_3^T\mathbf{v}_3 \leq 4. \tag{2}$$
$$\mathbf{v}_2^T\mathbf{v}_2 = 9, \tag{3}$$

for optimization variables $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^3$. This problem, as stated is not a convex optimization problem (for instance, the equality constraint $\mathbf{v}_2^T\mathbf{v}_2 = 9$ is not affine). In fact, the inner product $(\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x}^T\mathbf{y}$ is neither convex nor concave (for $n = 1$ we showed this in the practicals).

Nevertheless, our example has a very nice structure, since all the objective and constraint functions are linear combinations of inner products. We are going to show that this allows us to find a *semidefinite program (SDP)* that is equivalent to the problem. To prove find it, first recall following characterization of symmetric, positive semidefinite matrices:

Lemma 1. *Let $X \in S^n$. Then, the following are equivalent:*

(a) $X \in S^n_+$

(b) $X = V^TV$, *for some* $p \geq 1$ *and* $V \in \mathbb{R}^{p\times n}$

(c) $X = V^TV$, *for some* $V \in \mathbb{R}^{n\times n}$ *(we can even assume* $V \in S^n_+$*)*

Proof. (c)→(b) holds trivially, (b)→(a) holds since $\mathbf{z}^TX\mathbf{z} = \mathbf{z}^TV^TV\mathbf{z} = \|V\mathbf{z}\|^2 \geq 0$ for all $\mathbf{z} \in \mathbb{R}^n$. To see (a)→(c), recall that every symmetric matrix has a diagonal decomposition UDU^T, where $D = diag(\lambda_1, \ldots, \lambda_n)$ contains the eigenvalues, and U is an orthogonal matrix. Then, the square-root $V = X^{1/2} = U^T diag(\lambda_1^{1/2}, \ldots, \lambda_n^{1/2})U$ has the desired properties. □

Note that, in the setting of Lemma 1 (3), if $\mathbf{v}_i$ denote the columns of V, then $x_{ij} = \mathbf{v}_i^T\mathbf{v}_j$. Thus the entries of the matrix $X = (x_{ij})_{i,j=1}^n \in S^n_+$ are exactly the inner products of columns of V with each other. Thus our example is equivalent to the optimization problem with optimization variable $X \in S_3$, given by

$$\text{minimize } x_{11} - x_{13} + 3x_{23}, \tag{4}$$
$$\text{subject to } x_{12} + x_{33} \leq 4. \tag{5}$$
$$x_{22} = 9, \tag{6}$$
$$X \succeq 0 \tag{7}$$

This is clearly an SDP. To describe it in normal form recall that for symmetric matrices C, X we have $\mathrm{tr}(CX) = \sum_{i,j} c_{i,j}x_{i,j}$. So the problem can be also written as:

$$\begin{aligned} \text{minimize } & \mathrm{tr}(CX) && (8)\\ \text{subject to } & \mathrm{tr}(B_1X) \leq 4. && (9)\\ & \mathrm{tr}(B_2X) = 9, && (10)\\ & X \succeq 0 && (11) \end{aligned}$$

with

$$C = \begin{bmatrix} 1 & 0 & -1/2 \\ 0 & 0 & 3/2 \\ -1/2 & 3/2 & 0 \end{bmatrix}, B_1 = \begin{bmatrix} 0 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, B_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

As shown in the lecture this can further be rewritten into the standard form SDP by introducing slack variable for the $\leq$ constraint. (c.f. Chapter 4.6.2 in the book). Clearly this example can be generalized to all optimization problems with objective/constraint functions of the form $\sum_{i,j=1}^{n} a_{i,j}\mathbf{v}_i^T\mathbf{v}_j$, for optimization variables $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^n$.

1 The SDP-relaxation of Max-Cut

For a (finite) undirected graph $G = (V, E)$, a *cut* is a partition of the vertex set into two sets $V = V_{-1} \dot\cup V_1$. A cut of G is called *maximal*, if the number of edges between the two partition sets is maximal. Then, Max-Cut is the computational problem to find a maximal cut for a given input graph. There are no known polynomial time algorithms for Max-Cut, and the corresponding decision problem (for given $G, n \in \mathbb{N}$, is there a cut of size $\geq n$?) is NP-complete.

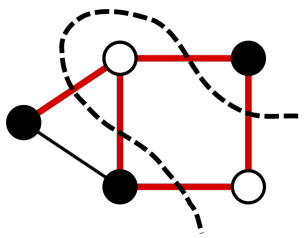

Figure 1: A maximum cut of the "house graph" (picture taken from Wikipedia)

If we identify every vertex of G with a (real valued) variable, then we can phrase Max-Cut as the following optimization problem:

$$\begin{aligned} \text{maximize } & \sum_{(v_i,v_j)\in E} \frac{1 - v_iv_j}{2} && (12)\\ \text{subject to } & v_i^2 = 1 \text{ for all } v_i \in V. && (13) \end{aligned}$$

Note that, by the constraint $v_i^2 = 1$, every variable v_i is either assigned -1 or 1, giving us a partition of V into two sets. Furthermore $\frac{1-v_iv_j}{2}$ is equal to 0 if $v_i = v_j \in \{1, -1\}$ (i.e. the corresponding vertices are in the same partition set), and equal to 1 if $v_i = -v_i \in \{1, -1\}$ (i.e. the corresponding vertices are in different partition set). Thus, the objective function $\sum_{(v_i,v_j)\in E} \frac{1-v_i,v_j}{2}$ outputs the total number of edges across the cut (respectively twice the number, since we count the edges (v_i, v_j) and (v_j, v_i) separately).

Since the objective/constraint functions of MAX-CUT only consist of products of variables and their sums, we can try to phrase it as an SDP, similar to the example in the previous section. We then obtain the so called *SDP-relaxation* of MAX-CUT, which is defined as the following SDP with optimization variable $X \in S_{|V|}$:

$$\text{maximize} \sum_{(v_i,v_j)\in E} \frac{1 - x_{ij}}{2} \tag{14}$$

$$\text{subject to } x_{ii} = 1 \text{ for all } v_i \in V \tag{15}$$

$$X \succeq 0. \tag{16}$$

SDPs are convex optimization problems (with generalized inequalities) and can thus be solved efficiently. Without going into too much details now, this means that, for fixed ϵ, we can approximate the solution of the SDP (up to an ϵ-mistake) in polynomial time. Thus, one might think that we can solve MAX-CUT by solving the above SDP (up to some $\epsilon < 1/2$) and then rounding the resulting value to a whole number. However, this would be a polynomial time algorithm that solves an NP-hard problem - So, did we prove that P = NP?!

No! The important detail that we are missing here, is that Lemma 1 (3) only guarantees that we can write the matrix X as the product V^TV for matrices V of *the same dimension* $|V| \times |V|$ (in general we can not get $p < n$ in Lemma 1 (2)). Thus the SDP-relaxation of MAX-CUT is only equivalent to the problem:

$$\text{maximize} \sum_{(v_i,v_j)\in E} \frac{1 - \mathbf{v}_i^T\mathbf{v}_j}{2} \tag{17}$$

$$\text{subject to } \mathbf{v}_i^T\mathbf{v}_i = 1 \text{ for all } v_i \in V, \tag{18}$$

for *vectors* $\mathbf{v}_i \in \mathbb{R}^{|V|}$. Nevertheless, in practise, solving this SDP often leads very good approximation to solution of the actual MAX-CUT problem. Since the feasibility set for the SDP-relaxation is bigger than the feasibility set for MAX-CUT, the optimal value of the SDP-relaxation will always be an over-estimate of the optimal value of the Max-Cut problem. For the house-graph you can obtain an optimal value ≈ 5.185486 (verify it yourself in CVXPY!). This is a slight overestimate of the number for an actual max-cut: 5 (see Figure 1).

It is a common strategy in general to approximate (or sometimes even accurately solve) discrete optimization problems by "relaxing" them to a convex optimization problems. Another important example is the *LP-relaxation* of $\{0, 1\}$-linear programs. The SDP relaxation of MAX-CUT was first described by Goemans and Williamson in 1995 (together with a rounding scheme to obtain actual cuts of G from the, in general, not integer-valued matrix solution X). Their procedure achieves an expected approximation ratio of ≈ 0.87856. This is thought to be optimal in some sense that we are not going to discuss here.