Semidefinite Programming

Pekka Orponen

T-79.7001 Postgraduate Course on Theoretical Computer Science

24.4.2008

Outline

- 1. Strict Quadratic Programs and Vector Programs
  - Strict quadratic programs
  - The vector program relaxation
- 2. Semidefinite Programs
  - Vector programs as matrix linear programs
  - Properties of semidefinite matrices
  - From vector programs to semidefinite programs
  - Notes on computation
- 3. Randomised Rounding of Vector Programs
  - Randomised rounding for MAX CUT

1. Strict Quadratic Programs and Vector Programs

- A quadratic program concerns optimising a quadratic function of integer variables, with quadratic constraints.
- A quadratic program is strict if it contains no linear terms, i.e. each monomial appearing in it is either constant or of degree 2.
- E.g. a strict quadratic program for weighted MAX CUT:
  - Given a weighted graph $G = (N, E, w)$, $N = [n] = \{1, \dots, n\}$.
  - Associate to each vertex $i \in N$ a variable $y_i \in \{+1, -1\}$. A cut $(S, \bar{S})$ is determined as $S = \{i \mid y_i = +1\}$, $\bar{S} = \{i \mid y_i = -1\}$.
  - The program:

$$
\begin{aligned}
\max \quad & \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - y_i y_j) & \\
\text{s.t.} \quad & y_i^2 = 1, & i \in N \\
& y_i \in \mathbb{Z}, & i \in N.
\end{aligned}
$$

The vector program relaxation

- Given a strict quadratic program on n variables y_i, relax the variables into n-dimensional vectors $v_i \in \mathbb{R}^n$, and replace quadratic terms by inner products of these.
- E.g. the MAX CUT vector program:

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij} (1 - v_i^T v_j) \\ \text{s.t.} \quad & v_i^T v_i = 1, & i \in N \\ & v_i \in \mathbb{R}^n, & i \in N. \end{aligned}$$

- Feasible solutions correspond to families of points on the n-dimensional unit sphere S_{n-1}.
- Original program given by restriction to 1-dimensional solutions, e.g. all points along the x-axis: $v_i = (y_i, 0, \ldots, 0)$.
- We shall see that vector programs can in fact be solved in polynomial time, and projections to 1 dimension yield nice approximations for the original problem.

2. Semidefinite Programming

- A vector program on n n-dimensional vectors $\{v_1, \ldots, v_n\}$ can also be viewed as a linear program on the $n \times n$ matrix Y of their inner products, $Y = [v_i^T v_j]_{ij}$.
- However there is a "structural" constraint on the respective matrix linear program: the feasible solutions must be specifically *inner product* matrices.
- This turns out to imply (cf. later) that the feasible solution matrices Y are symmetric and positive semidefinite, i.e.

$$x^T Yx \geq 0, \quad \text{for all } x \in \mathbb{R}^n.$$

- Thus vector programming problems can be reformulated as semidefinite programming problems.

- Define the Frobenius (inner) product of two $n \times n$ matrices $A, B \in \mathbb{R}^{n\times n}$ as

$$A \bullet B = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{ij} = \mathrm{tr}(A^T B).$$

- Denote the family of symmetric $n \times n$ real matrices by M_n, and the condition that $Y \in M_n$ be positive semidefinite by $Y \succeq 0$.
- Let $C, D_1, \ldots, D_k \in M_n$ and $d_1, \ldots, d_k \in R$. Then the general semidefinite programming problem is

$$\begin{aligned} \max / \min \;\; & C \bullet Y \\ \text{s.t. } & D_i \bullet Y = d_i, \qquad i = 1, \ldots, k \\ & Y \succeq 0, \\ & Y \in M_n. \end{aligned}$$

- E.g. the MAX CUT semidefinite program relaxation:

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - y_{ij}) \\ \text{s.t.} \quad & y_{ii} = 1, \qquad\qquad i \in N \\ & Y \succeq 0, \\ & Y \in M_n. \end{aligned}$$

- Or, equivalently:

$$\begin{aligned} \min \quad & W \bullet Y \\ \text{s.t.} \quad & D_i \bullet Y = 1, \qquad\qquad i \in N \\ & Y \succeq 0, \\ & Y \in M_n. \end{aligned}$$

where $Y = [y_{ij}]_{ij}$, $W = [w_{ij}]_{ij}$, $D_i = [1]_{ii}$.

Properties of positive semidefinite matrices

Let A be a real, symmetric $n \times n$ matrix. Then A has n (not necessarily distinct) real eigenvalues, and associated n linearly independent eigenvectors.

Theorem 1. Let $A \in M_n$. Then the following are equivalent:

1. $x^T A x \geq 0$ for all $x \in \mathbb{R}^n$.
2. All eigenvalues of A are nonnegative.
3. $A = W^T W$ for some $W \in \mathbb{R}^{n \times n}$.

Proof ($1 \Rightarrow 2$*).*

- Let λ be an eigenvalue of A, and v a corresponding eigenvector.
- Then $Av = \lambda v$ and $v^T Av = \lambda v^T v$.
- By assumption (1), $v^T Av \geq 0$, and since $v^T v > 0$, necessarily $\lambda \geq 0$.

Proof ($2 \Rightarrow 3$).

- Decompose A as $A = Q\Lambda Q^T$, where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, with $\lambda_1, \ldots, \lambda_n \geq 0$ the n eigenvalues of A.
- Since by assumption (2), $\lambda_i \geq 0$ for each i, we can further decompose $\Lambda = DD^T$, where $D = \mathrm{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n})$.
- Denote $W = (QD)^T$. Then $A = Q\Lambda Q^T = QDD^T Q = W^T W$.

Proof ($3 \Rightarrow 1$).

- By assumption (3), A can be decomposed as $A = W^T W$.
- Then for any $x \in \mathbb{R}^n$:

$$x^T A x = x^T W^T W x = (Wx)^T (Wx) \geq 0. \qquad \square$$

From vector programs to semidefinite programs

Given a vector program $\mathcal{V}$, define a corresponding semidefinite program $\mathcal{S}$ on the inner product matrix of the vector variables, as described earlier.

Corollary 2. Vector program $\mathcal{V}$ and semidefinite program $\mathcal{S}$ are equivalent (have essentially the same feasible solutions).

Proof. Let $v_1, \ldots, v_n$ be a feasible solution to $\mathcal{V}$. Let W be a matrix with columns $v_1, \ldots, v_n$. Then $Y = W^T W$ is a feasible solution to $\mathcal{S}$ with the same objective function value as $v_1, \ldots, v_n$.

Conversely, let Y be a feasible solution to $\mathcal{S}$. By Theorem 1 (iii), Y can be decomposed as $Y = W^T W$. Let $v_1, \ldots, v_n$ be the columns of W. Then $v_1, \ldots, v_n$ is a feasible solution to $\mathcal{V}$ with the same objective function value as Y. □

Notes on computation

- Using Cholesky decomposition, a matrix $A \in M_n$ can be decomposed in polynomial time as $A = U\Lambda U^T$, where Λ is a diagonal matrix whose entries are the eigenvalues of A.
- By Theorem 1 (ii), this gives a polynomial time test for positive semidefiniteness.
- The decomposition of Theorem (iii), $A = WW^T$, is not in general polynomial time computable, because W may contain irrational entries. It may however be approximated efficiently to arbitrary precision. In the following this slight inaccuracy is ignored.
- Note also that any convex combination of positive semidefinite matrices is again positive semidefinite.

- Semidefinite programs can be solved (to arbitrary accuracy) by the ellipsoid algorithm.
- To validate this, it suffices to show the existence of a polynomial time separation oracle.

Theorem 3. Let $\mathcal{S}$ be a semidefinite program and $A \in \mathbb{R}^n$. One can determine in polynomial time whether A is feasible for S and, if not, find a separating hyperplane.

Proof. A is feasible for $\mathcal{S}$ if it is symmetric, positive semidefinite, and satisfies all of the linear constraints. Each of these conditions can be tested in polynomial time. In the case of infeasible A, a separating hyperplane can be determined as follows:

- If A is not symmetric, then $a_{ij} > a_{ji}$ for some i, j. Then $y_{ij} \leq y_{ji}$ is a separating hyperplane.
- If A is not positive semidefinite, then it has a negative eigenvalue, say λ. Let v be a corresponding eigenvector. Then $(vv^T) \bullet Y = v^T Y v \geq 0$ is a separating hyperplane.
- If any of the linear constraints is violated, it directly yields a separating hyperplane. □

3. Randomised Rounding of Vector Programs

- Recall the outline of the present approximation scheme:

  1. Formulate the problem of interest as a strict quadratic program $\mathcal{P}$.
  2. Relax $\mathcal{P}$ into a vector program $\mathcal{V}$.
  3. Reformulate $\mathcal{V}$ as a semidefinite program $\mathcal{S}$ and solve (approximately) using the ellipsoid method.
  4. Round the solution of $\mathcal{V}$ back into $\mathcal{P}$ by projecting it on some 1-dimensional subspace.

- We shall now address the fourth task, using the MAX CUT program as an example.

Randomised rounding for MAX CUT

- Let $v_1, \ldots, v_n \in S_{n-1}$ be an optimal solution to the MAX CUT vector program, and let $\underline{\mathrm{OPT}}_v$ be its objective function value. We want to obtain a cut $(S, \bar{S})$ whose weight is a large fraction of OPT_v.
- The contribution of a pair of vectors v_i, v_j $(i < j)$ to OPT_v is

$$\frac{w_{ij}}{2}(1 - \cos\theta_{ij}),$$

where θ_{ij} denotes the (unsigned) angle between v_i and v_j.
- We would like vertices i, j to be separated by the cut if $\cos\theta_{ij}$ is large (close to π).

Here is an idea: pick a vector r on the unit sphere S_{n-1} uniformly at random, and define the cut by:

$$S = \{i \mid v_i^T r \geq 0\}, \quad \bar{S} = \{i \mid v_i^T r < 0\}.$$

Theorem 4. For any pair of vertices i, j:

$$\Pr[i \text{ and } j \text{ are separated by the cut}] = \frac{\theta_{ij}}{\pi}.$$

Proof. Let r' be the projection of r onto the plane containing vectors v_i and v_j. Vertices i and j are separated iff v_i and v_j have “different orientation” w.r.t. r', i.e. are on opposite sides of the normal line determined by r', i.e. the normal line falls in the angle of width θ_{ij} between v_i and v_j. Since r has been picked from a spherically symmetric distribution, r' will determine a random direction in the plane. The lemma follows. □

A technical issue: how to generate n-dimensional unit vectors u.a.r.?

Lemma 5. Let $x_1, \dots, x_n$ be independent $N(0,1)$ distributed random variables, and let $d = (x_1^2 + \dots + x_n^2)^{1/2}$. Then the random vector $r = (x_1/d, \dots, x_n/d)$ has uniform distribution on S_{n-1}.

Proof. Random vector $x = (x_1, \dots, x_n)$ has density

$$f(x_1, \dots, x_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-x_i^2/2} = \frac{1}{(2\pi)^{n/2}} e^{-\frac{1}{2}\sum_i x_i^2}.$$

Since the density depends only on the distance from the origin, the distribution of x is spherically symmetric. Hence, dividing by the length of x, i.e. d, yields a uniformly distributed random vector on S_{n-1}. □

- Now let us consider how close to OPT_v the weight of our random cut is likely to be.
- Let W be a random variable denoting the weight of the cut, i.e.

$$W = \sum_{1 \le i < j \le n} w_{ij} I[i \text{ and } j \text{ are separated by the cut}]$$

- Also, denote

$$\alpha = \frac{2}{\pi} \min_{0 \le \theta \le \pi} \frac{\theta}{1 - \cos\theta}.$$

By elementary calculus, $\alpha > 0.87856$.

Theorem 6. $E[W] \ge \alpha \cdot \mathrm{OPT}_v$.

Proof. By the definition of α,

$$\frac{\theta}{\pi} \geq \alpha \left(\frac{1 - \cos\theta}{2} \right),$$

for any θ, $0 \leq \theta \leq \pi$.

Thus, by Lemma 4:

$$\begin{aligned}
E[W] &= \sum_{1 \leq i < j \leq n} w_{ij} \Pr[i \text{ and } j \text{ are separated by the cut}] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \frac{\theta_{ij}}{\pi} \\
&\geq \alpha \cdot \sum_{1 \leq i < j \leq n} w_{ij} \frac{1}{2}(1 - \cos\theta_{ij}) \\
&= \alpha \cdot \mathrm{OPT}_v. \quad \square
\end{aligned}$$

By using repeated trials, this result can be strengthened:

Theorem 7. There is a randomised approximation algorithm for MAX CUT that with "arbitrarily high probability" achieves approximation factor > 0.87856.