16 Linear programming, second-order cone programming, semidefinite programming

In this lecture we look at applications of the path-following method seen in the previous lecture. The purpose is to demonstrate that one can construct self-concordant barrier functions for convex sets Q with some particular structure. Recall that, for a closed bounded convex set Q, we say that F is a *self-concordant barrier* for Q if F is self-concordant with $\text{dom}(F) = \text{int}(Q)$, $\text{epi}(F)$ is closed, and $\theta_F = \max_{x\in\text{dom}(F)} \lambda_F(x)^2 < \infty$.

16.1 Linear programming

A linear program is an optimization problem of the form

$$\min_{x\in\mathbb{R}^n} \quad c^T x \quad \text{s.t.} \quad Ax \le b \tag{1}$$

where $c \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ and the inequality $Ax \le b$ is understood componentwise, i.e., $a_i^T x \le b_i$ for all $i = 1,\dots,m$, where $a_1^T,\dots,a_m^T$ are the rows of A. The feasible set

$$Q = \{x \in \mathbb{R}^n : Ax \le b\} \tag{2}$$

is a *polyhedron*. We can solve (1) using the path-following method described in the previous lecture. This is based on the following:

Theorem 16.1. *Let $Q = \{x \in \mathbb{R}^n : Ax \le b\}$. Assuming Q is bounded, the function F defined by $F(x) = -\sum_{i=1}^m \log(b_i - a_i^T x)$ is a self-concordant barrier for Q with parameter $\theta_F \le m$.*

Proof. Note that $F(x) = f(b - Ax)$ where $f(y) = -\sum_{i=1}^m \log(y_i)$ for $y \in \mathbb{R}^m_{++}$. We know that f is self-concordant (Exercise sheet 3, q7), and furthermore it is easy to check that $\theta_f = m$ (in fact, for any $y \in \mathbb{R}^m_{++}$ we have $\lambda_f(y)^2 = m$). Since Q is bounded, A is injective, i.e., $\ker(A) = \{0\}$. It thus follows that F is self-concordant and that $\theta_F \le \theta_f = m$ (see Exercise sheet 3, q6). $\square$

Thus the path-following method allows us to compute an ϵ-approximate value to (1) using $\approx \sqrt{m}\log(m/\epsilon)$ total number of Newton iterations, assuming an initial point on the central path is given.

Remark. *Note that each Newton iteration of the path-following method for (1) costs $\approx n^2m$ floating point iterations. Indeed: each Newton iteration requires the computation of the Newton step $\nabla^2 F_t(x)^{-1}\nabla F_t(x)$. Since $F_t(x) = tc^T x - \sum_{i=1}^m \log(b_i - a_i^T x)$, the Hessian of F_t is given by*

$$\nabla^2 F_t(x) = A^T \operatorname{diag}\left(\frac{1}{(b_i - a_i^T x)^2}\right)_{i=1,\dots,m} A.$$

Computing $\nabla^2 F_t(x)$ takes $\approx n^2m$ floating point operations since A is $n \times m$; and computing $\nabla^2 F_t(x)^{-1}$ will take $\approx n^3$ operations. Since $m > n$ (since A is injective) the cost is dominated by forming $\nabla^2 F_t(x)$.

It follows that the total number of floating point operations of the path-following method for the linear program (1) is $\approx n^2m^{1.5}\log(m/\epsilon)$.

Applications of linear programming We give some examples of problems that can be formulated using linear programming:

1. Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and consider the problem of minimizing $\|Ax-b\|_\infty$ over $x \in \mathbb{R}^n$. This a nonsmooth convex minimization problem that can be solved, e.g., using the subgradient method. We show here that it can be formulated as a linear program. Indeed, we have

$$\begin{aligned}\min_{x\in\mathbb{R}^n} \quad \|Ax-b\|_\infty &= \min_{x\in\mathbb{R}^n, t\in\mathbb{R}} \ t \quad \text{s.t.} \quad \|Ax-b\|_\infty \le t \\ &= \min_{x\in\mathbb{R}^n, t\in\mathbb{R}} \ t \quad \text{s.t.} \quad -t \le a_i^T x - b_i \le t \ \ \forall i = 1,\dots,m.\end{aligned}$$

   The latter is a linear program with $n+1$ variables and $2m$ inequality constraints. For moderate values of n,m solving the linear program is preferable over the subgradient method as it easily gives us a high-precision solution. For large (huge) values of n,m where the complexity of linear programming is prohibitive, one must rely on first-order methods (e.g., subgradient method).

2. Consider now the problem of minimizing $\|Ax-b\|_1$ over $x \in \mathbb{R}^n$. Using similar ideas as above one can formulate this problem as a linear program. Indeed we have

$$\begin{aligned}\min_{x\in\mathbb{R}^n} \quad \|Ax-b\|_1 &= \min_{x\in\mathbb{R}^n, t\in\mathbb{R}^m} \ \sum_{i=1}^m t_i \quad \text{s.t.} \quad |(Ax-b)_i| \le t_i \ \forall i = 1,\dots,m \\ &= \min_{x\in\mathbb{R}^n, t\in\mathbb{R}^m} \ \sum_{i=1}^m t_i \quad \text{s.t.} \quad -t_i \le a_i^T x - b_i \le t_i \ \forall i = 1,\dots,m.\end{aligned}$$

   This is a linear program with $n+m$ variables and $2m$ inequality constraints.

3. A linear program in *standard form* is an optimization problem of the form

$$\min_{x\in\mathbb{R}^n} \quad c^T x \quad \text{s.t.} \quad Ax = b, \ x \ge 0 \tag{3}$$

   where $x \ge 0$ means $x_i \ge 0$ for all $i = 1,\dots,n$. Clearly (3) can be put in the form (1) since the constraint $Ax = b$ can be rewritten as $Ax \le b$ and $-Ax \le -b$. Conversely any LP in inequality form can be written as (3). It suffices to introduce a slack variable $s = b - Ax$, and to note that any $x \in \mathbb{R}^n$ can be written as $x = x^+ - x^-$ with $x^+, x^- \ge 0$. At the end we get:

$$\min_{x\in\mathbb{R}^n} \left\{c^T x \ : \ Ax \le b\right\} \quad = \quad \begin{array}{ll} \displaystyle\min_{\substack{x^+\in\mathbb{R}^n \\ x^-\in\mathbb{R}^n \\ s\in\mathbb{R}^m}} & c^T(x^+ - x^-) \\ \text{s.t.} & A(x^+ - x^-) + s = b \\ & x^+, x^-, s \ge 0. \end{array}$$

   The right-hand side is of the form (3).

16.2 Second-order cone program

A *second-order cone program* (SOCP) is an optimization problem of the form

$$\min_{x\in\mathbb{R}^n} \ c^T x \ \text{ s.t. } \ \|A_i x + b_i\|_2 \le d_i^T x + e_i \ \ \forall i = 1,\dots,m \tag{4}$$

where $c \in \mathbb{R}^n$ and $A_i \in \mathbb{R}^{l_i \times n}, b_i \in \mathbb{R}^{l_i}, d_i \in \mathbb{R}^n$ and $e_i \in \mathbb{R}$ for each $i = 1, \ldots, m$. (The l_i are arbitrary integers for $i = 1, \ldots, m$.) Linear programming is a special case of (4) when $A_i = 0$ and $b_i = 0$. Assuming that $Q = \left\{x \in \mathbb{R}^n : \|A_i x + b_i\|_2 \leq d_i^T x + e_i \ \forall i = 1, \ldots, m\right\}$ is bounded, one can show that the function

$$F(x) \ = \ -\sum_{i=1}^{m} \log\left((d_i^T x + e_i)^2 - \|A_i x + b_i\|_2^2\right)$$

is a self-concordant barrier for Q with $\theta_F \leq 2m$. This can be used to solve problem (4) with path-following methods.

Applications of second-order cone programming A typical application of second-order cone programming is regularized least-squares. Consider the problem

$$\min_{x \in \mathbb{R}^n} \quad \|Ax - b\|_2 \ + \ \lambda \|Dx\|_1 \tag{5}$$

where $\lambda \geq 0$, $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, D \in \mathbb{R}^{p \times n}$. It is not hard to see that this problem can be formulated as a second-order cone problem. Indeed (5) can be rewritten as:

$$\begin{array}{ll} \displaystyle\min_{\substack{x \in \mathbb{R}^n \\ t \in \mathbb{R}, s \in \mathbb{R}^p}} & t + \lambda \sum_{i=1}^{p} s_i \\ \text{s.t.} & \|Ax - b\|_2 \leq t \\ & -s_i \leq (Dx)_i \leq s_i \ \forall i = 1, \ldots, p \end{array}$$

which is a second-order cone program.

16.3 Semidefinite programming

Let $\mathbf{S}^n$ be the vector space of $n \times n$ real symmetric matrices. A matrix $A \in \mathbf{S}^n$ is *positive semidefinite* (psd) if all its eigenvalues are nonnegative, or, equivalently, if $x^T A x \geq 0$ for all $x \in \mathbb{R}^n$. Let $\mathbf{S}^n_+ \subset \mathbf{S}^n$ denote the convex set of positive semidefinite matrices. Given $A \in \mathbf{S}^n$ we use the abbreviation $A \succeq 0$ to indicate that A is psd. Also if $A, B \in \mathbf{S}^n$ we write $A \succeq B$ if $A - B \succeq 0$. The trace inner product on $\mathbf{S}^n$ is defined by

$$\langle A, B \rangle = \mathrm{Tr}(AB) = \sum_{1 \leq i,j \leq n} A_{ij} B_{ij}.$$

A *semidefinite program* is an optimization problem of the form

$$\min_{x \in \mathbb{R}^n} \ c^T x \ \ \text{s.t.} \ \ A_0 + x_1 A_1 + \cdots + x_n A_n \succeq 0, \tag{6}$$

where $c \in \mathbb{R}^n$, and $A_0, \ldots, A_n \in \mathbf{S}^m$. Note that the linear program (1) is a special case of (6) when $A_0, \ldots, A_n$ are diagonal (more precisely, $A_0 = \mathrm{diag}(b)$, and $A_i = -\mathrm{diag}(\alpha_i)$ where α_i is the i'th column of A.) It can be shown that second-order cone programs can also be put in the form (6). As such, semidefinite programming generalizes linear, and second-order cone programming.

Assuming $Q = \{x \in \mathbb{R}^n : A_0 + x_1 A_1 + \cdots + x_n A_n \succeq 0\}$ is bounded, one can show that the function

$$F(x) = -\log\det(A_0 + x_1 A_1 + \cdots + x_n A_n)$$

is a self-concordant barrier for Q with parameter $\theta_F \leq m$. As such, path-following methods can be used to solve semidefinite programs.

Applications of semidefinite programming

- Eigenvalue minimization: Let $A(x) = A_0 + x_1 A_1 + \cdots + x_n A_n$ be a symmetric matrix that depends affinely on $x \in \mathbb{R}^n$, and consider the optimization problem

$$\min_{x \in \mathbb{R}^n} \ \lambda_{\max}(A(x)) \tag{7}$$

  where $\lambda_{\max}(A)$ is the largest eigenvalue of A. Using the fact that $\lambda_{\max}(A) \le t$ iff $tI - A \succeq 0$ we easily see that (7) can be formulated as a semidefinite program:

$$\min_{x \in \mathbb{R}^n} \ \lambda_{\max}(A(x)) \quad = \quad \min_{x \in \mathbb{R}^n, t \in \mathbb{R}} \ t \ \text{ s.t. } \ tI - A(x) \succeq 0.$$

- Operator norm minimization: The *operator norm* of a matrix A is defined as

$$\|A\|_2 = \max_{\|x\|_2 = 1} \|Ax\|_2.$$

  Let $A(x) = A_0 + x_1 A_1 + \cdots + x_n A_n$ be a symmetric matrix that depends affinely on $x \in \mathbb{R}^n$, and consider the problem of minimizing the operator norm of $A(x)$:

$$\min_{x \in \mathbb{R}^n} \quad \|A(x)\|_2. \tag{8}$$

  One can formulate this problem as a semidefinite program. Indeed, for a symmetric matrix A one can show that $\|A\|_2 = \max\{|\lambda_i(A)|, i = 1, \ldots, n\}$ where $\lambda_i(A)$ are the eigenvalues of A. It follows that $\|A\|_2 \le t$ iff $-tI \preceq A \preceq tI$. Thus problem (8) can be written as a semidefinite program:

$$\min_{x \in \mathbb{R}^n} \quad \|A(x)\|_2 \quad = \quad \min_{x \in \mathbb{R}^n, t \in \mathbb{R}} \ t \ \text{ s.t. } \ tI - A(x) \succeq 0, \ tI + A(x) \succeq 0.$$

  See [BDX04] for an application of (8) to find a Markov chain on a graph with the fastest mixing time.

Many applications of linear/second-order cone/semidefinite programming are provided in the book [BV04, Chapter 4].

16.4 Software

Implementation of path-following methods for linear/second-order/semidefinite programming are available online. These implementations use more sophisticated versions of the path-following method we have seen, but the main idea of following the central path using Newton's method is the same. To get started we recommend checking out CVX (`http://cvxr.com/cvx/`) which gives a user-friendly interface to these solvers on Matlab (interfaces for Python also exist, check CVXPY `https://www.cvxpy.org/`).

References

[BDX04] Stephen Boyd, Persi Diaconis, and Lin Xiao. Fastest mixing markov chain on a graph. *SIAM review*, 46(4):667–689, 2004. 4

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004. 4