Convex Optimization

Review Session 3

Before we begin, let's produce a quick summary of the 'standard' convex programs we consider in this course.

- Linear programs (LP)

$$\begin{aligned} \min \quad & \boldsymbol{c} \cdot \boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \end{aligned}$$

- Quadratic programs (QP)

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{x}^\top P\boldsymbol{x} + \boldsymbol{q} \cdot \boldsymbol{x} + \boldsymbol{r} \\ \text{subject to} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \end{aligned}$$

  Where P is positive semidefinite.

- Quadratically constrained quadratic programs (QCQP)

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{x}^\top P^0\boldsymbol{x} + \boldsymbol{q}^0 \cdot \boldsymbol{x} + r_0 \\ \text{subject to} \quad & \frac{1}{2}\boldsymbol{x}^\top P^i\boldsymbol{x} + \boldsymbol{q}^i \cdot \boldsymbol{x} + r_i \leq 0 \\ & A\boldsymbol{x} \leq \boldsymbol{b} \end{aligned}$$

  Where P is positive semidefinite.

- Second-order cone programs (SOCP)

$$\begin{aligned} \min \quad & \boldsymbol{f} \cdot \boldsymbol{x} \\ & \left\| A^i\boldsymbol{x} + \boldsymbol{b}^i \right\|_2 \leq \boldsymbol{c}^i \cdot \boldsymbol{x} + \boldsymbol{d}^i \\ & A\boldsymbol{x} \leq \boldsymbol{b} \end{aligned}$$

  Note that the sign of the inequality matters!

- Semi-definite programs (SDP)

$$\begin{aligned} \min \quad & \boldsymbol{c} \cdot \boldsymbol{x} \\ \text{subject to} \quad & G + \sum x_i F^i \preceq 0 \end{aligned}$$

  (Note that if an SDP contains more than one inequalities of the type above, they can be aggregated by forming a matrix whose block-diagonal elements consist of each of the LHSs of the inequalities concerned.)

- Geometric Program (GP)

$$\begin{array}{rl} \min & f_0(\boldsymbol{x}) \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 1 \\ & h_i(\boldsymbol{x}) = 1 \\ & \boldsymbol{x} > \mathbf{0} \end{array}$$

  Where the f_i are *posynomials* – ie: functions with domain $\mathbb{R}^n_{++}$ of the form

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} c_k x_1^{a_1^k} x_2^{a_2^k} x_3^{a_3^k} \cdots x_n^{a_n^k} \qquad c_k > 0, \boldsymbol{a}^k \in \mathbb{R}^n$$

  and the h_i are *monomials* – functions of the form above with $K = 1$. Note that

  - Posynomials are closed under addition and multiplication.
  - Monomials are closed under multiplication and division.
  - Posynoial $\times$ Monomial = Psynomial.
  - Posynomial $\div$ Monimial = Posynomial.

  A few notes on geometric programs

  - Constraints of the form $f(\boldsymbol{x}) \leq h(\boldsymbol{x})$ and $h_1(\boldsymbol{x}) = h_2(\boldsymbol{x})$ (as above, f are posynomials, h are monomials) can be re-cast by dividing both sides by the monomial.
  - The maximization of a *monomial* can be dealt with by minimzing its reciprocal, which is also a monomial. (In this case, we can't just minimize -1 times the original objective, because the constant coefficient in the definition of posynomials and monomials must be positive).
  - Geometric programs can be convexified by substituting $y_i = \log x_i \Rightarrow x_i = e^{y_i}$, and then taking logarithms of the objectives and constraints.

Another important concept we will be using in this review session (that is often confusing to many!) is the topic of *Schur Complements*. Consider a symmetric matrix

$$X = \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix}$$

where $A \in \mathbb{S}^k$ (the set of symmetric $k \times k$ matrices). If $\det A \neq 0$, then the matrix

$$S = C - B^\top A^{-1} B$$

is called the *Schur complement* of A in X. It is the case that

- $X \succ 0$ if and only if $A \succ 0$ and $S \succ 0$.
- If $A \succ 0$, then $X \succeq 0 \Leftrightarrow S \succeq 0$.

These results will prove super-super-useful in many parts of this course.

I promise you I've looked far and wide for some sort of interesting/intuitive/cute way of understanding/interpreting Schur complements and proofs relating to them, but I'm afraid I only succeeded in making myself mildly depressed. So just take these results an put them in a list of results that are super-useful but a pain to prove (another result like that is the fact that $\det(AB) = \det(A)\det(B)$). If anyone finds anything cool about Schur complements, let me know!

■ □ ■

Question 1 (Two Short Question)

1. Show that

$$\text{LP} \subset \text{QP} \subset \text{QCQP} \subset \text{SOCP} \subset \text{SDP}$$

   By which I mean that you should show every LP is a special case of a QP, etc...

2. Consider the convex optimization problem

$$\begin{array}{rll} \min & f_0(\boldsymbol{x}) & \\ \text{subject to} & f_i(\boldsymbol{x}) \leq 0 & i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = 0 & i = 1, \ldots, p \end{array}$$

   Assume f_0 is differentiable. Denote the feasible region of the problem by

$$\mathscr{X} = \{\boldsymbol{x} : f_i(\boldsymbol{x}) \leq 0, h_i(\boldsymbol{x}) = 0\}$$

   Show that $\boldsymbol{x}$ is optimal if and only if $\boldsymbol{x} \in \mathscr{X}$ and

$$\nabla f_0(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) \geq 0 \text{ for all } \boldsymbol{y} \in \mathscr{X}$$

   Use this result to show that in the program

$$\begin{array}{ll} \min & f_0(\boldsymbol{x}) \\ \text{s.t.} & A\boldsymbol{x} = \boldsymbol{b} \end{array}$$

   $\boldsymbol{x}$ is optimal if and only if it is feasible and there exists a $\boldsymbol{\nu}$ such that

$$\nabla f_0(\boldsymbol{x}) + A^\top \boldsymbol{\nu} = \boldsymbol{0} \tag{1}$$

   Interpret this condition.

Solution

1. LPs, QPs and QCQPs are trivial. Now, consider a QCQP – how can we express it as an SOCP? Let's consider both the objective and the constraints

   - The QCQP objective is of the form $\frac{1}{2}\boldsymbol{x}^\top P_0\boldsymbol{x}+\boldsymbol{q}_0\cdot\boldsymbol{x}+r_0$ whereas the SOCP objective is linear. This can easily be remedied, though, by using an epigraph formulation in which we minimize t subject to $t \geq \frac{1}{2}\boldsymbol{x}^\top P_0\boldsymbol{x} + \boldsymbol{q}_0 \cdot \boldsymbol{x}+r_0$. Thus, we have linearized our objective by adding another quadratic constraint to the problem.

   - Consider the following quadratic constraint

   $$\frac{1}{2}\boldsymbol{x}^\top P_i\boldsymbol{x} + \boldsymbol{q}^i \cdot \boldsymbol{x} + r_i \leq 0$$

   and note that it is equivalent to the constraint

   $$\left\| \begin{array}{c} \frac{1}{2}(1+\boldsymbol{q}^i\cdot\boldsymbol{x}+r_i) \\ (P^i)^{\frac{1}{2}}\boldsymbol{x} \end{array} \right\|_2 \leq \frac{1}{2}\left(1-\boldsymbol{q}^i\cdot\boldsymbol{x}-r_i\right)$$

   (Note that as per the definition of a QCQP, P^i must be positive definite, so the quantity $(P^i)^{\frac{1}{2}}$ exists).

   Now, let's start from an SOCP – it can easily be verified that setting $\boldsymbol{q^i} = \mathbf{0}$ in all the inequalities (and then squaring both sides of the inequality) results in a QCQP.

   Finally, let's consider SOCPs and SDPs. Things are now less obvious. How do we take an SOCP constraint

   $$\left\|A^i\boldsymbol{x}+\boldsymbol{b}^i\right\|_2 \leq \boldsymbol{c}^i\cdot\boldsymbol{x}+\boldsymbol{d}^i$$

   and make it into an SDP constraint? Schur complements come in useful here. Consider the matrix

   $$M = \left[\begin{array}{cc} I & A^i\boldsymbol{x}+\boldsymbol{b}^i \\ (A^i\boldsymbol{x}+\boldsymbol{b}^i)^\top & \boldsymbol{c}^i\cdot\boldsymbol{x}+\boldsymbol{d}^i \end{array}\right]$$

   The Schur Complement of this matrix is (in this case, this is a scalar quantity)

   $$\boldsymbol{c}^i\cdot\boldsymbol{x}+\boldsymbol{d}^i - \left\|A^i\boldsymbol{x}+\boldsymbol{b}^i\right\|_2$$

   By our results for Schur complements, requiring this quantity to be positive is equivalent to requiring M to be positive definite. Thus, our SOCP constraint can be written as

   $$\left[\begin{array}{cc} I & A^i\boldsymbol{x}+\boldsymbol{b}^i \\ (A^i\boldsymbol{x}+\boldsymbol{b}^i)^\top & \boldsymbol{c}^i\cdot\boldsymbol{x}+\boldsymbol{d}^i \end{array}\right] \succeq 0$$

2. The proof of both statements is provided in detail in Boyd. See the bottom of page 139 for the first, and the middle of page 141 for the second.

   The reason I wanted to highlight this question is because of the conclusion – namely, that $\boldsymbol{x}$ is optimal if and only if it can be expressed as[1]

   $$\nabla f_0(\boldsymbol{x}) = A^\top \boldsymbol{\nu}$$

   in other words, if it can be expressed as a linear combination of the rows of A. Why does this make sense? Simply because we can always move *perpendicular* to the rows of A without violating the constraint $A\boldsymbol{x} = \boldsymbol{b}$. However, as soon as we start moving in any direction that is not perpendicular to these rows, the constraint is violated. By ensuring that the direction of growth of the function is a linear combination of these rows, we are effectively stating that there is no way to 'grow' at that point without violating the constraint. Thus, we have reached an optimum.

   We will later see that a similar characterization exists for inequalities, except that in that case, we'll require the $\boldsymbol{\nu}$ to be positive or negative, depending on the direction of the inequality. This is because with an inequality, it *is* possible to move in a direction that is not perpendicular to the inequality, as long as it's in the right direction that doesn't result in a violation. Much more on this soon!

[1]My characterization differs slightly from that in Boyd in that I flipped the sign of $\boldsymbol{\nu}$. But since $\boldsymbol{\nu} \in \mathbb{R}^p$, it can take both positive and negative values, so the change doesn't make a difference.

■ □ ■

Question 2 (Semidefinite Programs)

In this question, we will be looking at three important examples of semidefinite programs. First, we'll look at various optimization problems involving eigenvalues can be solved using semidefinite programming. We'll then look at a brief introduction to the fascinating field of optimization over polynomials. Finally, we'll look at one example of a place in which Schur complements come in very useful.

Part A

[1] Suppose $A : \mathbb{R}^n \to \mathbb{S}^m$ ($\mathbb{S}^m$ is the set of symmetric matrices of size $m \times m$) is of the form

$$A(\boldsymbol{x}) = A^0 + x_1 A^1 + \cdots + x_n A^n$$

[1]Mostly based on Exercise 4.43 in Boyd. See also exercise 3.26 for more on the convexity of certain functions of eigenvalues.

where $A^i \in \mathbb{S}^m$. Let $\lambda_1(\boldsymbol{x}) \geq \cdots \geq \lambda_m(\boldsymbol{x})$ denote the eigenvalues of $A(\boldsymbol{x})$. Show how to pose the following problems as convex optimization problems

1. Minimize and maximize the sum of the eigenvalues.
2. Minimize the maximum eigenvalue $\lambda_1(\boldsymbol{x})$.
3. Minimize the spread of the eigenvalues $\lambda_1(\boldsymbol{x}) - \lambda_m(\boldsymbol{x})$.
4. Minimize the sum of the absolute eigenvalues, $|\lambda_1(\boldsymbol{x})| + \cdots + |\lambda_m(\boldsymbol{x})|$

Solution

1. The sum of eigenvalues is equal to the trace of the matrix[2]. In this case,

   $$\text{tr}(X) = \text{tr}(A^0) + \sum x_i \text{tr}(A^i)$$

   Thus, the sum of the eigenvalues is a linear function of $\boldsymbol{x}$. Maximizing and minimizing it is therefore just an LP.

2. First, consider that $\lambda_1(\boldsymbol{x}) \leq t$ if and only if $A(\boldsymbol{x}) - tI \preceq 0$. To see why, consider that

   $$\begin{aligned} \lambda_1(\boldsymbol{x}) &= \sup\left\{\boldsymbol{y}^\top A(\boldsymbol{x})\boldsymbol{y} : \|\boldsymbol{y}\|_2 \leq 1\right\} \\ &= \sup\left\{\boldsymbol{y}^\top \left[A(\boldsymbol{x}) - tI\right]\boldsymbol{y} + t : \|\boldsymbol{y}\|_2 \leq 1\right\} \end{aligned}$$

   If the the matrix in square brackets is negative semidefinite, $\lambda_1(\boldsymbol{x})$ is clearly $\leq t$. Similarly, if $\lambda_1(\boldsymbol{x}) \leq t$, there can be no vector $\boldsymbol{y}$ for which the first term in the supremum can be greater or equal to 0. Thus, the matrix in square brackets is negative semidefinite.

   As such, we can minimize the maximum eigenvalue by solving

   $$\begin{aligned} \min \quad & t \\ \text{subject to} \quad & A(x) - tI \preceq 0 \end{aligned}$$

   This is an SDP.

3. We also have that $\lambda_m(\boldsymbol{x}) \geq t$ if and only if $A(\boldsymbol{x}) - tI \succeq 0$. As such, we can find the minimum spread in eigenvalues by solving

   $$\begin{aligned} \min \quad & t_1 - t_2 \\ \text{subject to} \quad & A(\boldsymbol{x}) - t_1 I \preceq 0 \\ & A(\boldsymbol{x}) - t_2 I \succeq 0 \end{aligned}$$

[2] To see why, consider that a matrix X can be written as $U^\top \Lambda U$, where Λ is a diagonal matrix containing the eigenvalues of X and U is an orthonormal matrix containing the normalized eigenvectors of X. We then have that $\text{tr}(X) = \text{tr}(U^\top \Lambda U) = \text{tr}(\Lambda U^\top U) = \text{tr}(\Lambda) = \sum \lambda_i$

4. We can solve this problem by simply solving

$$\begin{array}{ll} \min & \mathrm{tr} Y \\ \text{subject to} & -Y \preceq A(\boldsymbol{x}) \preceq Y \end{array}$$

To see why this works, consider a fixed $\boldsymbol{x}$, and minimize over Y. By decomposing $A(\boldsymbol{x})$ into its eigenvalue decomposition $A(\boldsymbol{x}) = Q \Lambda Q^\top$, we can write our program as

$$\begin{array}{ll} \min & \mathrm{tr} Y \\ \text{subject to} & -Q^\top Y Q \preceq \Lambda \preceq Q^\top Y Q \end{array}$$

Then, making the change of variables $Z = Q^\top Y Q$ (and noting that $\mathrm{tr}(Z) = \mathrm{tr}(Q^\top Y Q) = \mathrm{tr}(Q^\top Q Y) = \mathrm{tr}(Y)$), we get

$$\begin{array}{ll} \min & \mathrm{tr} Z \\ \text{subject to} & -Z \preceq \Lambda \preceq Z \end{array}$$

Only the diagonal entries of Z enter into the objective function, so we can assume without loss of generality that Z is diagonal. In that case, the optimal Z is clearly $|Z|$, and so the optimal value of our program for fixed $\boldsymbol{x}$ is the sum of the absolute eigenvalues of $A(\boldsymbol{x})$. Minimizing over $\boldsymbol{x}$, we get our expected result.

Part B

[2]Write a program that finds that polynomial $p : \mathbb{R} \to \mathbb{R}$ of degree $2k$ that satisfies a set of bounds $\ell_i \leq p(t_i) \leq u_i$ for a specified set of points t_i, and, of all the polynomials that satisfy these bounds, has the greatest minimum value.

Solution

In solving this question, we will need to use the fact that any univariate polynomial of degree $2k$ is nonnegative on $\mathbb{R}$ if and only if it can be expressed as the sum of squares of two polynomials of degree k or less. In other words, the polynomial

$$p(t) = x_0 + x_1 t + x_2 t^2 + \cdots + x_{2k} x^{2k}$$

is ≥ 0 everywhere if and only if it can be expressed as

$$p(t) = r(t)^2 + s(t)^2$$

where r and s are polynomials of degree k.[3]

Given that fact, we will show that $p(t) \geq 0$ if and only if there is a matrix $Y \in \mathbb{S}_+^{k+1}$ (the set of size $k+1$ positive semidefinite matrices) such that

[2]Exercise 4.44 in Boyd, using Exercise 2.37

[3]This is *not* an easy result, and it actually relies on the Fundamental Theorem of Algebra, which states that every nonzero univariate polynomial of degree n has exactly n complex roots (counted with multiplicity), and that if all the coefficients of the polynomial are real, the roots exist in convex conjugate pairs – in other words, x is a root if and only if x^* is a root.

Now, consider that if $p(t)$ is nonegative everywhere, all its real roots must have even multiplicity. Thus, when factorizing $p(t)$, all the real roots produce terms of the form $(x-a)^2$, and since all the complex roots come in complex conjugate pairs $c \pm di$, the corresponding terms can be written as

$$(x-c-di)(x-c+di) = (x-c)^2 + d^2$$

Finally, the identity

$$(\alpha^2+\beta^2)(\gamma^2+\delta^2) = (\alpha\gamma-\beta\delta)^2(\alpha\delta+\beta\gamma)^2$$

allows us to combine every partial product as a sum of only two squares. Leading to the result. See `http://bit.ly/zOdwJn` for more details.

$$x_i = \sum_{(m,n):m+n=i+2} Y_{mn} \tag{2}$$

(Note that this matrix is such that the coefficient x_i ends up being equal to the $(i+1)^{\text{th}}$ top-right to bottom-left diagonal through the matrix Y).

The first step involves some book-keeping. We need to show that if $\boldsymbol{v}(t) = (1, t, t^2, \cdots, t^k)^\top$

$$p(t) = \boldsymbol{v}(\boldsymbol{t})^\top Y \boldsymbol{v}(t) = \sum_{i=0}^{2k} x_i t^i$$

with the x_i defined as in equation 2.

This is tedious but not particularly difficult to do.[4]

$$\begin{aligned}
\boldsymbol{v}(t)^\top Y \boldsymbol{v}(t) &= \boldsymbol{v}(t)^\top \left(\sum_{j=1}^{k+1} \boldsymbol{Y}_{\Diamond j} t^{j-1} \right) \\
&= \sum_{i=1}^{k+1} \left[t^{i-1} \left(\sum_{j=1}^{k+1} Y_{ij} t^{j-1} \right) \right] \\
&= \sum_{i,j=1}^{k+1} Y_{ij} t^{i+j-2} \\
&= \sum_{i=0}^{2k} \left[\left(\sum_{m+n=i+2} Y_{mn} \right) t^i \right]
\end{aligned}$$

We now need to show that $p(t) \geq 0$ if and only if it can be written in that form with $Y \succeq 0$. Let's do both directions

$\boxed{p(t) = \boldsymbol{t}_k^\top Y \boldsymbol{t}, Y \succeq 0 \Rightarrow p(t) \geq 0}$: This is not difficult. If $Y \succeq 0$, then for any $\boldsymbol{t}_k$ (ie: for any t) the quadratic form is positive (and therefore so is $p(t)$.

$\boxed{p(t) \geq 0 \Rightarrow \exists Y \succeq 0 \textbf{ s.t. } p(t) = \boldsymbol{t}_k^\top Y \boldsymbol{t}}$: To prove this direction, we need to use the theorem discussed above and write $p(t)$ as[5]

$$\begin{aligned}
p(t) &= r(t)^2 + s(t)^2 \\
&= (\boldsymbol{v}(t) \cdot \boldsymbol{r})^2 + (\boldsymbol{v}(t) \cdot \boldsymbol{s})^2 \\
&= \boldsymbol{v}(t)^\top \boldsymbol{r}\boldsymbol{r}^\top \boldsymbol{v}(t) + \boldsymbol{v}(t)^\top \boldsymbol{s}\boldsymbol{s}^\top \boldsymbol{v}(t) \\
&= \boldsymbol{v}(t)^\top \left(\boldsymbol{r}\boldsymbol{r}^\top + \boldsymbol{s}\boldsymbol{s}^\top \right) \boldsymbol{v}(t) \\
&= \boldsymbol{v}(t)^\top Y \boldsymbol{v}(t)
\end{aligned}$$

With $Y = \boldsymbol{r}\boldsymbol{r}^\top + \boldsymbol{s}\boldsymbol{s}^\top$. To show that this matrx is positive

[4] In the below, I use the notation $\boldsymbol{Y}_{\Diamond j}$ to denote the j^{th} row of Y.

[5] We let the vectors $\boldsymbol{r}$ and $\boldsymbol{s}$ contain the coefficients of the polynomials $r(t)$ and $s(t)$.

semidefinite, consider that for any $\boldsymbol{x}$:

$$\begin{aligned}\boldsymbol{x}^\top Y \boldsymbol{x} &= \boldsymbol{x}^\top \left(\boldsymbol{r}\boldsymbol{r}^\top + \boldsymbol{s}\boldsymbol{s}^\top\right) \boldsymbol{x} \\ &= \|\boldsymbol{r}^\top \boldsymbol{x}\|_2^2 + \|\boldsymbol{s}^\top \boldsymbol{x}\|_2^2 \\ &\geq 0\end{aligned}$$

Thus, Y must be positive semidefinite.

Having said all that, let us finally return to the problem in the question. It can be formulated in the form

$$\begin{array}{rll} \max & \inf_t p(t) & \\ \text{subject to} & \ell_i \leq p(t_i) \leq u_i & \forall i \end{array}$$

Our first step is to modify this to

$$\begin{array}{rll} \max & \gamma & \\ \text{subject to} & p(t) - \gamma \geq 0 & \forall t \in \mathbb{R} \\ & \ell_i \leq p(t_i) \leq u_i & \forall i \text{ specified} \end{array}$$

Using our formulation above, we ca re-state this as

$$\begin{array}{rll} \max & \gamma & \\ \text{subject to} & x_0 - \gamma = Y_{11} & \\ & x_j = \sum_{(m,n):m+n=j+2} Y_{mn} & j = 1, \cdots, 2k \\ & \ell_i \leq \sum_{j=0}^{2k} x_j t_i^j \leq u_i & \forall i \text{ specified} \\ & Y \succeq 0 & \end{array}$$

Part C

[3]Consider the following constraint on the vector $\boldsymbol{x}$

$$(A\boldsymbol{x} + \boldsymbol{b})^\top (P^0 + x_1 P^1 + \cdots + x_n P^n)^{-1} (A\boldsymbol{x} + \boldsymbol{b}) \leq t$$

This is a composition of a matrix fractional function (see Example 3.4, page 76 in Boyd) and an affine transformation, and so the LHS is convex.

Confirm this (and show that this constraint could form part of an SDP) by finding a matrix $F(\boldsymbol{x}, t)$ affine in $(\boldsymbol{x}, t)$ such that the condition above is true if and only if

$$F(\boldsymbol{x}, t) \succeq 0$$

Solution

[3]Additional Exercise 3.8 in Boyd

We use Schur complements. Consider the matrix

$$F(\boldsymbol{x}, t) = \begin{bmatrix} t & (A\boldsymbol{x}+b)^\top \\ A\boldsymbol{x}+b & P^0 + x_1 P^1 + \cdots + x_n P^n \end{bmatrix}$$

The Schur complement of this matrix is precisely the LHS of the constraint above minus t. Thus, the constraint above holds if and only if this matrix is positive semidefinite. Furthermore, this matrix is linear in $\boldsymbol{x}$ and t.

■ □ ■

Question 3 (Robust Linear Programming)

Robust optimization is a paradigm that immunizes solutions of convex problems against bounded uncertainty in the parameters of the problem.

As a typical example, consider the following linear program

$$\begin{array}{rll} \min_{\boldsymbol{x}} & \boldsymbol{c}\cdot\boldsymbol{x} & \\ \text{subject to} & \boldsymbol{a}^i\cdot\boldsymbol{x} \leq b^i & i = 1,\ldots,n \end{array}$$

If the parameters of the problem are known exactly, this is a standard LP. However, imagine that the individual vectors $\boldsymbol{a}^i$ are *not* known exactly. In this question, we will be considering methods for dealing with this uncertain in a number of cases

Part A

Suppose we know each $\boldsymbol{a}^i$ may be located anywhere inside some ellipsoid $\mathscr{E}^i$, where

$$\mathscr{E}^i = \left\{ \bar{\boldsymbol{a}}^i + P^i\boldsymbol{u} : \|\boldsymbol{u}\|_2 \leq 1 \right\}$$

And we want to solve this LP and get a solution that is feasible whatever values the $\boldsymbol{a}^i$ end up taking. Write an SOCP that does precisely that.

Solution

First, consider that our problem can be rewritten as (henceforth, I'll leave out the $i = 1,\ldots,n$ – let it be understood that's what I mean whenever I write this program!)

$$\begin{array}{rll} \min\limits_{\boldsymbol{x}} & \boldsymbol{c}\cdot\boldsymbol{x} & \\ \text{subject to} & \boldsymbol{a}^i\cdot\boldsymbol{x} \leq b^i & \forall \boldsymbol{a}^i \in \mathscr{E}^i \end{array}$$

This, in turn, can be written as

$$\begin{array}{rl} \min\limits_{\boldsymbol{x}} & \boldsymbol{c}\cdot\boldsymbol{x} \\ \text{subject to} & \bar{\boldsymbol{a}}^i\cdot\boldsymbol{x} + \sup\limits_{\boldsymbol{u}:\|\boldsymbol{u}\|\leq 1} \left\{ \boldsymbol{x}\cdot\left(P^i\boldsymbol{u}\right)\right\} \leq \boldsymbol{b}^i \end{array}$$

Consider, however, that by the Cauchy-Schwartz Inequality

$$\boldsymbol{x} \cdot (P^i \boldsymbol{u}) = \boldsymbol{u}^\top P^{i,\top} \boldsymbol{x} = \boldsymbol{u} \cdot (P^{i,\top} \boldsymbol{x}) \leq \|\boldsymbol{u}\|_2 \cdot \|(P^{i,\top} \boldsymbol{x})\|_2$$

This maximum is attained when $\boldsymbol{u}$ is aligned with $P^{i,\top}\boldsymbol{x}$, in which case the norm is equal to $\|\boldsymbol{u}\|_2 \|(P^{i,\top}\boldsymbol{x})\|_2$. Maximizing over all $\boldsymbol{u}$ in that direction with $\|\boldsymbol{u}\|_2 \leq 1$, we find that the maximizing norm is $\|(P^{i,\top}\boldsymbol{x})\|_2$. Thus, the program above is inequivalent to

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c} \cdot \boldsymbol{x} \\ \text{subject to} \quad & \bar{\boldsymbol{a}}^i \cdot \boldsymbol{x} + \|(P^{i,\top}\boldsymbol{x})\|_2 \leq \boldsymbol{b}^i \end{aligned}$$

This is an SOCP.

Part B

[4]Now suppose we now know that each component of each $\boldsymbol{a}^i$ may be located anywhere between two particular values. Expressed mathematically, we stack the $\boldsymbol{a}^i$, row by row, into a matrix A (to create a single constraint $A\boldsymbol{x} \leq \boldsymbol{b}$), and say the matrix A is know to lie inside the set

$$\mathscr{A} = \left\{ A : \bar{A} - V \leq A \leq \bar{A} + V \right\}$$

where $\bar{A}$ and V are known. Suppose we once again want to solve this LP to get a solution that is feasible whatever value A ends up taking. Write an LP that does exactly that.

Solution

In this case, we can write our problem as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c} \cdot \boldsymbol{x} \\ \text{subject to} \quad & \sup_{A \in \mathscr{A}} (A\boldsymbol{x}) \leq \boldsymbol{b} \end{aligned}$$

Now, consider the quantity $A\boldsymbol{x}$. Effectively, each column of A will be multiplied by an entry of $\boldsymbol{x}$. If that entry of $\boldsymbol{x}$ happens to be positive, then we should make that column of A as large as possible (ie: *add on* as much of V as we can). If that entry of $\boldsymbol{x}$ is negative, the opposite holds true.

As such, we find that[6]

$$\sup_{A \in \mathscr{A}} (A\boldsymbol{x}) = \bar{A}\boldsymbol{x} + V\boldsymbol{x}^+ - V\boldsymbol{x}^- = \bar{A}\boldsymbol{x} + V|\boldsymbol{x}|$$

[6]Here, we write $\boldsymbol{x}^+$ to denote the vector equal to $\boldsymbol{x}$ for very positive entry of $\boldsymbol{x}$ and equal to 0 otherwise, and we define $\boldsymbol{x}^-$ to be $-(-\boldsymbol{x})^+$ – ie: the vector equal to the positive part of $\boldsymbol{x}$ wherever $\boldsymbol{x}$ is negative, and equal to 0 otherwise.

As such, our problem becomes

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c} \cdot \boldsymbol{x} \\ \text{subject to} \quad & \bar{A}\boldsymbol{x} + V|\boldsymbol{x}| \leq \boldsymbol{b} \end{aligned}$$

[4]Boyd, Exercise 4.13

This is not an LP (because of the absolute value constraint), but it's easy to transform it into one as follows

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c} \cdot \boldsymbol{x} \\ \text{subject to} \quad & \bar{A}\boldsymbol{x} + V\boldsymbol{y} \leq \boldsymbol{b} \\ & \boldsymbol{y} \geq \boldsymbol{x} \\ & \boldsymbol{y} \geq -\boldsymbol{x} \end{aligned}$$

Consider this new program – clearly, the first constraint will try to reduce $\boldsymbol{y}$ as much as possible (to relax the program as much as possible), but the second two constraints will ensure $\boldsymbol{y}$ never becomes smaller than $|\boldsymbol{x}|$. Thus, the LP works as required.

Part C

Suppose, now that the $\boldsymbol{a}^i$ are known to be inside some polyhedron $\mathscr{P}^i$, given by

$$\mathscr{P}^i = \left\{\boldsymbol{a} : C^i \boldsymbol{a} \leq \boldsymbol{d}^i\right\}$$

Once again, we want to solve the LP to get a solution for any constraint inside the set. How might you do that?

Solution

Based on what we've covered so far in class, there's no particularly good way to achieve this. The only way, in fact, is to find the extreme points of each polyhedron $\mathscr{P}^i$ and then notice that in each case, the maximum of the RHS of each constraint will occur at one of these points. Thus, if polyhedron $\mathscr{P}^i$ has n extreme points, we simply need to replace constraint i by n constraints, each using one of the extreme points.

This method, however, leaves much to be desired. In particular, it requires pre-processing of the polyhedra to find their extreme points. When we study duality, we will discover that there is a much simpler, and more elegant, way of achieving the same outcome.

Part D

Finally, suppose that the $\boldsymbol{a}^i$ may take on *any* value, but that we consider this problem in a statistical framework in which each of the $\boldsymbol{a}^i$ are independent random vectors following a multivariate Gaussian distribution with mean $\bar{\boldsymbol{a}}^i$ and covariance matrix Σ^i.

Clearly, it no longer makes sense to require that our constraint hold for *every* possible value of $\boldsymbol{a}^i$, so instead we solve the program

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c} \cdot \boldsymbol{x} \\ \text{subject to} \quad & \mathbb{P}\left(\boldsymbol{a}^i \cdot \boldsymbol{x} \leq b^i\right) \geq \eta \end{aligned}$$

for some $\eta \geq \frac{1}{2}$.

Write an SOCP that solves this problem.

Solution

First, consider the quantity $\boldsymbol{a}^i \cdot \boldsymbol{x}$. We know that $\boldsymbol{a}^i \sim N(\hat{\boldsymbol{a}}^i, \Sigma^i)$, and so by standard results for the multivariate normal distribution,

$$\boldsymbol{a}^i \cdot \boldsymbol{x} \sim N(\mu = \hat{\boldsymbol{a}}^i \cdot \boldsymbol{x}, \sigma^2 = \boldsymbol{x}^\top \Sigma^i \boldsymbol{x})$$

As such, letting Z denote a random variable with $Z \sim N(0,1)$ and letting $\Phi(x) = \mathbb{P}(Z \leq x)$, we find that

$$\begin{aligned} \mathbb{P}\left(\boldsymbol{a}^i \cdot \boldsymbol{x} \leq b^i\right) &= \mathbb{P}\left(\frac{\boldsymbol{a}^i \cdot \boldsymbol{x} - \mu}{\sigma}\right) \leq \frac{b^i - \mu}{\sigma} \\ &= \mathbb{P}\left(Z \leq \frac{b^i - \mu}{\sigma}\right) \\ &= \Phi\left(\frac{b^i - \mu}{\sigma}\right) \end{aligned}$$

Thus, the constraint

$$\mathbb{P}\left(\boldsymbol{a}^i \cdot \boldsymbol{x} \leq b^i\right) \geq \eta$$

is equivalent to the requirement that

$$\frac{b^i - \mu}{\sigma} \geq \Phi^{-1}(\eta)$$

or equivalently

$$\mu + \Phi^{-1}(\eta)\sigma \leq b_i$$

Feeding in our values for μ and σ, this becomes

$$\bar{\boldsymbol{a}}^i \cdot \boldsymbol{x} + \Phi^{-1}(\eta) \left\| (\Sigma^i)^{\frac{1}{2}} \boldsymbol{x} \right\|_2 \leq b_i$$

This is indeed a second-order cone constraint.

■ □ ■

Question 4 (Sparsity: The LASSO and Robust Optimization)

[5]The LASSO (least absolute shrinkage and regression operator) is a name given to ℓ_1-regularized regression.

Consider a set of input vectors $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n$ (collected in the matrix X) and a set of associated responses $y_1, \ldots, y_n$, collected in the

[5]The material in this question was adapted from Chapter 14 of *Optimization for Machine Learning*, MIT University Press, 2011

vector $\boldsymbol{y}$. In regression, we attempt to *approximate* each response y_i as a linear combination in the corresponding input vector $\boldsymbol{x}^i$ – the weight given to each component of the input vector is given by β^i. The aim of regression is to find the $\boldsymbol{\beta}$ that results in an estimate $\hat{\boldsymbol{y}} = X\boldsymbol{\beta}$ that best estimates $\boldsymbol{y}$ itself. The standard way to do this is to solve the QP

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2^2$$

The resulting estimator $\boldsymbol{\beta}$ minimizes the squared error resulting from approximating $\boldsymbol{y}$ by $\hat{\boldsymbol{y}}$. This works great, but there are a few issues with standard regression

- The resulting $\boldsymbol{\beta}$ results in a $\hat{\boldsymbol{y}}$ that very tightly matches the data $\boldsymbol{y}$, but might not match *future* observations that are as-of-yet unobserved. In other words, $\boldsymbol{\beta}$ might be *overfit* to the existing data.

- If the number inputs is larger than the number of points we have (in other words, if X is short and wide), the solution to this problem might be undetermined (in other words, there may be lots of different vectors $\boldsymbol{\beta}$ resulting in an objective function of 0.

- Similarly, in a case with many inputs, the vector $\boldsymbol{\beta}$ we get is likely to have many very small but non-zero entries; each of the many factors contributes a small amount to the overall result. Unfortunately, this might make the model difficult to handle, and it also means that there's no way to really know which variables are 'more important'.

The LASSO is a quadratic program that solves the following problem (it is more customary to square the ℓ_2 norm in the formulation of the LASSO, but in this question, it will be easier to deal with this equivalent version)

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2 + \lambda\|\boldsymbol{\beta}\|_1 \tag{3}$$

Remarkably, it turns out that the LASSO resolves all three problems above (for many more details and proofs, see `http://bit.ly/ytxbb5`). In particular, the LASSO is *sparse* – this means that depending on the choice of λ, the resulting vector $\boldsymbol{\beta}$ might have many of its components set to 0 exactly. This is extremely useful in obtaining a more manageable model, and in performing variable selection.

In this question, we will explore an absolutely remarkable (and unexpected) relationship between the LASSO and robust optimization, and use this relationship to provide a surprisingly simple proof of the sparsity of the LASSO.

First, show that program 3 (the LASSO) is equivalent to the robust problem

$$\min_{\boldsymbol{\beta}} \max_{U \in \mathscr{U}} \|\boldsymbol{y} - (X+U)\boldsymbol{\beta}\|_2 \tag{4}$$

Where the uncertainty set $\mathscr{U}$ is given by

$$\mathscr{U} = \left\{ [\boldsymbol{u}^1, \cdots, \boldsymbol{u}^m] : \left\|\boldsymbol{u}^i\right\|_2 \le \lambda, i = 1, \cdots, m \right\}$$

Then, use this result to show that the LASSO is sparse. In particular let I be some subset of the inputs (columns of $\boldsymbol{x}$) involved, and let I be a subset of these inputs. We say a solution $\boldsymbol{\beta}$ is *supported* on I if and only if all elements of $\boldsymbol{\beta}$ indexed by I are nonzero, and all others are zero. Similarly, we define the set $\mathscr{U}^I$ to consist of that subset of $\mathscr{U}$ in which all columns corresponding to elements of I are non-zero, and all others are $\mathbf{0}$. Show that the robust problem in equation 4 has a solution that is supported on I *if* there is a perturbation $\tilde{U} \in \mathscr{U}^{I^c}$ that perturbs X to $\tilde{X} = X + \tilde{U}$ such that the program

$$\min_{\boldsymbol{\beta}} \max_{U \in \mathscr{U}^I} \left\|\boldsymbol{y} - (\tilde{X}+U)\boldsymbol{\beta}\right\|_2$$

has a solution that is supported on I.

How does this imply sparsity?

Solution

Let us begin by proving the equivalence of 3 and 4. We will show that for any given $\boldsymbol{\beta}$,

$$\max_{U \in \mathscr{U}} \|\boldsymbol{y} - (X+U)\boldsymbol{\beta}\|_2 = \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2 + \lambda\|\boldsymbol{\beta}\|_1$$

First, consider that using the triangle inequality twice

$$\begin{aligned}
\max_{U \in \mathscr{U}} \|\boldsymbol{y} - (X+U)\boldsymbol{\beta}\|_2 &\le \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2 + \max_{U \in \mathscr{U}} \|U\boldsymbol{\beta}\|_2 \\
&= \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2 + \max_{\boldsymbol{u}^1, \cdots, \boldsymbol{u}^m : \|\boldsymbol{u}^i\|_2 \le \lambda} \left\|\boldsymbol{u}^1\beta_1 + \cdots + \boldsymbol{u}^m\beta_m\right\|_2 \\
&\le \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2 + \sum_{i=1}^m \beta_i \max_{\boldsymbol{u} : \|\boldsymbol{u}\| \le \lambda} \|\boldsymbol{u}\|_2 \\
&= \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2 + \lambda\|\boldsymbol{\beta}\|_1
\end{aligned}$$

For the other inequality, let $\boldsymbol{u}$ be a vector of magnitude λ in the direction $\boldsymbol{y} - X\boldsymbol{\beta}$

$$\boldsymbol{u} = \begin{cases} \lambda \frac{\boldsymbol{y} - X\boldsymbol{\beta}}{\|\boldsymbol{y} - X\boldsymbol{\beta}\|_2} & \text{if } \boldsymbol{y} \ne X\boldsymbol{\beta} \\ \text{Any vector with } \ell_2\text{-norm } \lambda & \text{otherwise} \end{cases}$$

and take

$$U = [\boldsymbol{u}^1, \cdots, \boldsymbol{u}^m]$$

with every $\boldsymbol{u}^i$ corresponding to the vector $\boldsymbol{u}$ adjusted so that $\boldsymbol{u}^i\beta_i$ will always point in the a direction *opposite* to $\boldsymbol{y} - X\boldsymbol{\beta}$.

$$\boldsymbol{u}^i = \begin{cases} -\text{sgn}(\beta_i)\boldsymbol{u} & \text{if } \beta_i \neq 0 \\ -\boldsymbol{u} & \text{otherwise} \end{cases}$$

(You can trivially check that $U \in \mathscr{U}$). The sum total of these manipulations results in a U that makes

$$\begin{aligned}
\|\boldsymbol{y} + (X+U)\boldsymbol{\beta}\|_2 &= \|\boldsymbol{y} - X\boldsymbol{\beta} + U\boldsymbol{\beta}\|_2 \\
&= \left\|\boldsymbol{y} - X\boldsymbol{\beta} + \sum_{i=1}^m \boldsymbol{u}^i\beta_i\right\|_2 \\
&= \left\|\boldsymbol{y} - X\boldsymbol{\beta} + \lambda\sum_{i=1}^m \left(-\frac{\boldsymbol{y} - X\boldsymbol{\beta}}{\|\boldsymbol{y} - X\boldsymbol{\beta}\|_2}\right)|\beta_i|\right\|_2 \\
&= \left\|\left(1 - \lambda\sum_{i=1}^m \frac{|\beta_i|}{\|\boldsymbol{y} - X\boldsymbol{\beta}\|_2}\right)(\boldsymbol{y} - X\boldsymbol{\beta})\right\|_2 \\
&= \left(1 - \lambda\sum_{i=1}^m \frac{|\beta_i|}{\|\boldsymbol{y} - X\boldsymbol{\beta}\|_2}\right)\|\boldsymbol{y} - X\boldsymbol{\beta}\|_2 \\
&= \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2 + \lambda|\boldsymbol{\beta}|_1 \frac{\|\boldsymbol{y} - X\boldsymbol{\beta}\|_2}{\|\boldsymbol{y} - X\boldsymbol{\beta}\|_2} \\
&= \|\boldsymbol{y} - X\boldsymbol{\beta}\|_2 + \lambda|\boldsymbol{\beta}|_1
\end{aligned}$$

Thus, we have shown that there is *one* value of U for which the LHS = RHS. Since, in our program, we are *maximizing* the LHS over all $U \in \mathscr{U}$, the max of the LHS must be greater to equal to the RHS.

We have therefore shown that these two programs have the same optimal solution for the same $\boldsymbol{\beta}$. Thus, given the optimal $\boldsymbol{\beta}$ for either one, this solution is also optimal for the other. Thus, they are equivalent in the sense discussed by Boyd in section 4.1.3, page 130.

First, let's understand how the last part implies sparsity, because it's a bit of a mouthful! The original robust program perturbs *all* the inputs when finding its optimal result. What we're now saying is that we can restrict ourselves to only perturbing a few indices (those in I) while *fixing* a given perturbation for the other indices (those in I^c). The statement at the end of the question then says that if we find *any* such *fixed* perturbation of the I^c that results in their being irrelevant in the solution, then they'll also be irrelevant in the fully robust solution.

Let's now prove this result. TODO.

This, of course, proves that the LASSO is sparse – it simply removes all inputs that have a perturbation in $\mathscr{U}$ that would make them irrelevant. It also gives us unparalleled insight into *how* the LASSO is sparse, and provides an intriguing avenue of research – perhaps robustness is generally a good way to induce sparsity? This is an active avenue of research today.

■ □ ■