1 Constraints on the dual program

We saw last time that the nonlinear program

\[
(P) \quad \begin{cases} 
\text{minimize} & f(x) \\
\text{subject to} & g(x) \leq 0
\end{cases}
\]

has a dual program

\[
(D) \quad \begin{cases} 
\text{maximize} & h(\lambda) = \inf \{ f(x) + \lambda \cdot g(x) : x \in S \} \\
\text{subject to} & \lambda \geq 0
\end{cases}
\]

As written, this dual program is unconstrained apart from the requirement that \( \lambda \geq 0 \).

In practice, there are many cases where \( h(\lambda) = -\infty \) outside a specific region. If we’re maximizing \( h \), then values of \( \lambda \) for which \( h(\lambda) = -\infty \) are not very useful to us. So we may add constraints to \( D \) that cut off “bad” values of \( \lambda \), where \( h(\lambda) = -\infty \).

You may feel bad about adding constraints: doesn’t that just make the program harder? But really, we’re just bringing hidden constraints into the open when we do this. Dealing with a function \( h \) that can be \(-\infty\) sometimes is no easier, especially since this guarantees that \( h \) is not continuous where it switches to \(-\infty\).

2 Linear programming

The nicest example of such introduced constraints occurs when \( P \) is a linear program:

\[
(P) \quad \begin{cases} 
\text{minimize} & c \cdot x \\
\text{subject to} & Ax \geq b.
\end{cases}
\]

where \( A \) is an \( m \times n \) matrix, \( b \in \mathbb{R}^m \), and \( c \in \mathbb{R}^n \). (In standard form, the constraint looks like \( b - Ax \leq 0 \).)

Here, the dual objective function is \( h(\lambda) = \inf \{ c \cdot x + \lambda \cdot (b - Ax) : x \in \mathbb{R}^n \} \). We can rewrite the Lagrangian as

\[
c \cdot x + \lambda \cdot (Ax - b) = (c - A^T \lambda) \cdot x + \lambda \cdot b.
\]

---

1 This document comes from the Math 484 course webpage: https://faculty.math.illinois.edu/~mlavrov/courses/484-fall-2018.html
We’re minimizing a linear function of \( x \) to determine \( h(\lambda) \), and this pretty much always results in \(-\infty\). If \( c - A^T\lambda \) has any negative component, just make the corresponding component of \( x \) be arbitrarily large. If \( c - A^T\lambda \) has any positive component, just make the corresponding component of \( x \) be arbitrarily negative.

So we deduce a constraint on \( D \): to have \( h(\lambda) > -\infty \), we must have \( c - A^T\lambda = 0 \). As a bonus, when \( \lambda \) satisfies this constraint, the Lagrangian no longer depends on \( x \), and we get

\[
h(\lambda) = \inf \{ \lambda \cdot b : x \in \mathbb{R}^n \} = \lambda \cdot b.
\]

The dual program is therefore

\[
(D) \begin{cases}
\text{maximize} & \lambda \cdot b \\
\text{subject to} & A^T\lambda = c \\
 & \lambda \geq 0.
\end{cases}
\]

Progress? Kind of. We haven’t necessarily obtained a dramatically simpler dual here: it’s another linear program. When \( P \) has lots of variables and very few constraints, \( D \) has very few variables and lots of constraints, and vice versa; sometimes, one is easier than the other.

Another convenient feature is that when \( P \) is infeasible (there are no points \( x \) satisfying \( Ax \geq b \)), \( D \) will be unbounded. In such a case, we can give a short “certificate” for \( P \)’s infeasibility by showing a way to make the dual objective value arbitrarily large.

### 3 Constrained geometric programming

Recall that a posynomial term in variables \( t_1, t_2, \ldots, t_m \) is a product \( C t_1^{\alpha_1} t_2^{\alpha_2} \cdots t_m^{\alpha_m} \), where \( \alpha_1, \alpha_2, \ldots, \alpha_m \) are arbitrarily real numbers and \( C > 0 \). A posynomial is a sum of one or more posynomial terms.

Earlier in this course, we learned to solve unconstrained geometric programs: to minimized an arbitrary posynomial, subject to all the variables remaining positive.

Now, we can also look at constrained geometric programs. The constraints we allow will have the form \( P(t_1, t_2, \ldots, t_m) \leq 1 \), where \( P \) can be any posynomial. (In the next lecture, we will discuss slightly more general constraints.)

The general procedure for deriving the dual of such programs is painful and involves lots of notation. So we will consider a specific example: the geometric program

\[
(GP) \begin{cases}
\text{minimize} & \{ t_1^2 + 3 t_2^{-2} t_3^{-1} \} \\
\text{subject to} & t_1^{-1} t_2^2 + 2 t_1^{-1} t_3^2 \leq 1.
\end{cases}
\]

(Here, \( \mathbb{R}_+ \) stands for the set of positive real numbers: so our domain is \( S = \{(t_1, t_2, t_3) : t_1 > 0, t_2 > 0, t_3 > 0\} \).

The ideas we see here apply to all constrained geometric programs, and we will see the general form of the dual in the next lecture.
We modify \( GP \) slightly. First, since we know \( t_i > 0 \), we can let \( x_i = \log t_i \) and work with the variables \( x_1, x_2, x_3 \) instead. Now, the domain of \( (x_1, x_2, x_3) \) is all of \( \mathbb{R}^3 \), and the objective function and constraints are convex function of \( x \). We get:

\[
(GP') \begin{cases}
\text{minimize} & 2e^{2x_1} + 3e^{-2x_2-x_3} \\
\text{subject to} & e^{-x_1+2x_2} + 2e^{-x_1+2x_3} \leq 1.
\end{cases}
\]

Second, we want to be able to look at each of the posynomial terms in this problem (both the ones in the constraint and the ones in the objective function) separately. So we replace the exponents by \( z_1, z_2, z_3, z_4 \) and write down the problem

\[
(GP'') \begin{cases}
\text{minimize} & e^{z_1} + e^{z_2} \\
\text{subject to} & e^{z_3} + e^{z_4} \leq 1 \\
& z_1 \geq \log 2 + 2x_1 \\
& z_2 \geq \log 3 - 2x_2 - x_3 \\
& z_3 \geq -x_1 + 2x_2 \\
& z_4 \geq \log 2 - x_1 - 2x_3.
\end{cases}
\]

(If’s okay to have the constraint \( z_1 \geq \log 2 + 2x_1 \) rather than \( z_1 = \log 2 + 2x_1 \) because we want \( e^{z_1} \) as small as possible. This reasoning applies to all the terms.)

There are 5 constraints, so there are 5 dual variables. Call them \( \mu, \lambda_1, \lambda_2, \lambda_3, \lambda_4 \) in order of the constraints; \( \mu \) is special because it goes with the \( e^{z_1} \) constraint.

The Lagrangian is

\[
L(x, z, \mu, \lambda) = e^{z_1} + e^{z_2} + \mu(e^{z_3} + e^{z_4} - 1) + \lambda_1(\log 2 + 2x_1 - z_1) + \lambda_2(\log 3 - 2x_2 - x_3 - z_2) + \lambda_3(-x_1 + 2x_2 - z_3) + \lambda_4(\log 2 - x_1 - 2x_3 - z_4).
\]

Remember, the dual objective function is \( h(\mu, \lambda) = \inf \{ L(x, z, \mu, \lambda) : x \in \mathbb{R}^3, z \in \mathbb{R}^4 \} \).

This is messy, but by deducing some constraints on the dual variables, we can make the Lagrangian look simpler.

- The coefficient of \( x_1 \) is \( 2\lambda_1 - \lambda_3 - \lambda_4 \). Unless this is 0, \( L \) is unbounded and \( h(\mu, \lambda) = -\infty \). So we add the constraint \( 2\lambda_1 - \lambda_3 - \lambda_4 = 0 \) and can now forget all about \( x_1 \).
- Similarly, we add the constraints \(-2\lambda_2 + 2\lambda_3 = 0 \) and \(-\lambda_2 - 2\lambda_4 = 0 \), which takes care of the variables \( x_2 \) and \( x_3 \).
- For \( i = 1, 2 \), the part of Lagrangian depending on \( z_i \) is \( e^{z_i} - \lambda_i z_i \), which is minimized when \( z_i = \log \lambda_i \). So we make this substitution.
- For \( i = 3, 4 \), the part of the Lagrangian depending on \( z_i \) is \( \mu e^{z_i} - \lambda_i z_i \), which is minimized when \( z_i = \log \frac{\lambda_i}{\mu} \). So we make this substitution.
Having cleaned everything up, we get

\[ h(\mu, \lambda) = \lambda_1 + \lambda_2 + \mu \left( \frac{\lambda_3}{\mu} + \frac{\lambda_4}{\mu} - 1 \right) + \lambda_1 \log \frac{2}{\lambda_1} + \lambda_2 \log \frac{2}{\lambda_2} + \lambda_3 \log \frac{\mu}{\lambda_3} + \lambda_4 \log \frac{2\mu}{\lambda_4} \]

with the constraint

\[ \lambda_1 - \lambda_3 - \lambda_4 = -2\lambda_2 + 2\lambda_3 = -\lambda_2 - 2\lambda_4 = 0. \]

Let’s do one more step: since \( \mu \) is unconstrained, we can optimize with respect to \( \mu \) while ignoring \( \lambda \). The part of \( h(\mu, \lambda) \) that depends on \( \mu \) is

\[ \lambda_3 \log \lambda + \lambda_4 \log \mu - \mu \]

which is a concave function maximized at its only critical point, \( \mu = \lambda_3 + \lambda_4 \). Setting \( \mu \) to this value, we get

\[ h(\lambda) = \lambda_1 + \lambda_2 + \lambda_1 \log \frac{2}{\lambda_1} + \lambda_2 \log \frac{3}{\lambda_2} + \lambda_3 \log \frac{\lambda_3 + \lambda_4}{\lambda_3} + \lambda_4 \log \frac{2(\lambda_3 + \lambda_4)}{\lambda_4} \]

We can combine all the terms with log in them into a single log to make things look a bit nicer.

Summarizing what we have so far: the dual program looks like

\[
\begin{align*}
(D) \quad & \text{maximize} & h(\lambda) &= \lambda_1 + \lambda_2 + \log \left( \frac{2}{\lambda_1} \right)^{\lambda_1} \left( \frac{3}{\lambda_2} \right)^{\lambda_2} \left( \frac{1}{\lambda_3} \right)^{\lambda_3} \left( \frac{2}{\lambda_4} \right)^{\lambda_4} (\lambda_3 + \lambda_4)^{\lambda_3 + \lambda_4} \\
& \text{subject to} & \lambda_1 - \lambda_3 - \lambda_4 &= 0 \\
& & -2\lambda_2 + 2\lambda_3 &= 0 \\
& & -\lambda_2 - 2\lambda_4 &= 0 \\
& & \lambda_1, \lambda_2, \lambda_3, \lambda_4 &\geq 0.
\end{align*}
\]

There is still a bit more work to be done here to make this nicer, but we have a dual problem now. We’ll leave the cleanup for the next lecture.

There’s also some awkwardness that occurs when \( \lambda_i = 0 \) for any \( i \), since then we’re dividing by 0. We’ll also address this later, if we have time.

The other thing to remember is that in the process of determining \( h(\lambda) \), we decided that \( z_i = \log \lambda_i \) is the best value of \( z_i \) to use. This will come in handy when we want to understand how to use an optimal dual solution to get an optimal primal solution.