Lecture 3

Review of Linear programs and duality*

In the next few lectures we will study special classes of optimization problems that can be solved efficiently (in time polynomial in the input size), linear programs (LP) and semidefinite programs (SDP). SDPs are more general than LPs (any LP is an SDP) but are a special case of more general convex programs. One of the most attractive features of LPs and SDPs, in addition to being efficiently solvable, is that there is a rich theory of duality. This makes LPs and SDPs useful not only as algorithms, but also as a proof technique.

3.1 Administrative Things

- Students should sign up for Piazza
- Recitations are highly recommended. Identical recitations will be given Friday 10am - 11am and 11am - noon in 214 Annenberg
- The first homework set is due Thursday 12:59pm at Annenberg 214, as instructed on the homework set.

3.2 Linear Programs

Any LP can be brought into the following standard primal form $\mathcal{P}$

$$\begin{align*}
\text{max} & \quad c^\top x \\
\text{s.t.} & \quad Ax \leq b \\
& \quad x \geq 0
\end{align*}$$

(3.1)

where $x, c, b \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$.

*Lecturer: Thomas Vidick. Scribe: Fabian Boemer.*
For example,

$$\begin{align*}
\max & \quad x_1 + 2x_2 \\
\text{s.t.} & \quad 0 \leq x_1 \leq 2 \\
& \quad 0 \leq x_2 \leq 1 \\
& \quad x_1 + x_2 \leq 2
\end{align*}$$

(3.2)

has standard form with

$$c = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}, b = \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix}.$$  

For equality constraints, we can simply include two inequality constraints. For example $x = y$ is equivalent to $x \leq y, y \leq x$.

We can represent a linear program geometrically. We represent constraints by half-spaces. The feasible region $K$ is the set of all $x$ such that $x \geq 0$ and $Ax \leq b$. $K$ is a convex polytope $P_K$. For any target value $k$ the objective function defines a hyperplane $c^\top x = k$. The optimal solution can be found geometrically by increasing the target value as long as the corresponding level set, $c^\top x = k$ for increasing $k$, intersects $P_K$.

---

1If we take any $x, x' \in K$, $\forall \lambda \in [0, 1], \lambda x + (1 - \lambda)x' \in K$. This is the definition of convex. A polytope is any region that has flat faces, i.e. can be defined by linear equalities.
Figure 3.1: The region $K$ is the feasible region.

Figure 3.1 shows this geometric interpretation for our example 3.2 above.

**Types of LPs**  An LP $\mathcal{P}$ can be
- *infeasible*, i.e. the polytope $P$ is empty. For example,
  \[
  \begin{align*}
  \text{max} & \quad x_1 \\
  \text{s.t.} & \quad x_1 \geq 0 \\
  & \quad x_1 \leq -1
  \end{align*}
  \]
  \quad (3.3)

- *feasible*, but the polytope $P$ unbounded. For example,
  \[
  \begin{align*}
  \text{max} & \quad x_1 \\
  \text{s.t.} & \quad x_1 \geq 0
  \end{align*}
  \]
  \quad (3.4)
• feasible, and the polytope \( P \) bounded. For example, 3.2.

Suppose we are given an LP that is both feasible and bounded. A \textit{vertex} of \( K \) is defined as any \( x \in K \) such that \( n \) linearly independent constraints are satisfied with equality. Thus a vertex is always the unique solution of a system of equations

\[ A'x = b' \]

where \( A' \) is an \( n \times n \) submatrix of \( A \).

**Theorem 3.1.** Given an LP that is both feasible and bounded, there is always an optimum that is a vertex.

\begin{proof}
We sketch a proof. Consider an optimal point \( x \in K \). If \( x \) is not a vertex, it satisfies at most \( n - 1 \) of the constraints with equality. Let \( e \) be a unit vector that is orthogonal to the rows of \( A \) corresponding to these \( n - 1 \) constraints. Then it is possible to add or remove a small multiple of \( e \) to \( x \) while still satisfying all constraints. At least one of the two possibilities, adding a positive or a negative multiple, will not decrease the objective value. Moving in that direction as much as possible will lead to an additional constraint being satisfied with equality. Iterating the process eventually leads to a vertex whose objective value is at least as large as that of the initial \( x \).
\end{proof}

3.2.1 Bounding the size of optima to an LP

The \textit{size} of an LP is the total number of bits required to completely specify the LP, including \( A, b, c, m, n \). How large of a size can the solution \( x \) be in the size of the LP? Consider the following LP:

\[
\begin{align*}
\min & \quad x_n \\
\text{s.t.} & \quad x_1 \geq k \\
& \quad x_2 - kx_1 \geq 0 \\
& \quad \vdots \\
& \quad x_n - kx_{n-1} \geq 0
\end{align*}
\] (3.5)

The solution is \( x_n = k^n \). As an integer this is exponentially larger than the integers used to specify the LP. In terms of the number of bits required to represent it, \( n \log k \) bits are required, which is of the same order as the number of bits required to specify the LP in the first place.

**Theorem 3.2.** Any feasible, bounded LP has an optimum that can be specified using a number of bits that is polynomial in the size of the LP.

\begin{proof}
We know the optimum can be attained at a vertex, which will always be the unique solution to a system of equations \( A'x = b' \) for \( A' \) a square submatrix of \( A \) and \( b' \) the
corresponding subvector of \( b \). This system of equations can be solved in \( O(n^3) \) steps using Gaussian elimination. The algorithm for Gaussian elimination, if it is properly implemented, will not increase the size of the coefficients manipulated by too much at each step. Doing the analysis carefully, one can verify that the solution \( x \) can always be specified exactly using a number of bits polynomial in the size of \( A' \) and \( b' \).

Note that an LP can have as many as \( \binom{m}{n} \) vertices, since each vertex is a set of \( n \) constraints satisfied from \( m \) possible, so knowing that one of the vertices is optimal is not too helpful.

### 3.3 Solving LPs (in polynomial time)

“LPs can be solved in polynomial time.” What does this mean? We need to specify the input size. This is the number of bits require to write down the LP. Note this allows exponential-sized coefficients. The following algorithms can be used to solve LPs.

- **Simplex algorithm.** The simplex algorithm is the most efficient algorithm in practice. Intuitively, the algorithm performs a walk along the edges of the feasible region in attempt to keep improving the objective value. The choice of which edge to take at each step is called the *pivot rule*. Unfortunately, for almost all pivot rules known one can construct examples on which the use of that rule will take exponential time before reaching an optimum. It is a major open problem whether there exists a pivot rule which always converges in polynomial time. Nevertheless, in practice the algorithm works very well and is the most widely used.

- **Ellipsoid algorithm** [Khachiyan, 1979] The ellipsoid algorithm is the first provably polynomial-time algorithm for solving LPs. In fact ellipsoid solves the related feasability problem, which is easily seen to be equivalent to solving the LP (hint: binary search!). The assumptions required by the ellipsoid algorithm are
  
  - \( \exists r, R \in \mathbb{R} \) such that either
    * \( K \) is empty, or
    * \( B(O_1, r) \subseteq K \subseteq B(O_2, R) \), where \( B(O_1, r) \) is a ball centered at \( O_1 \), with radius \( r \). Note, \( O_2 \) can always be made the origin for sufficiently large \( R \), but \( O_1 \) cannot always be made the origin.
  
  - There exists a separation oracle for \( K \) that can be implemented efficiently. A (weak) separation oracle for a convex \( K \subseteq \mathbb{R}^n \) is a procedure whose inputs are \( x \in \mathbb{R}^N \) and \( \epsilon > 0 \), and either outputs “\( d(x, K) \leq \epsilon \)” or returns \( y \in \mathbb{R}^n \setminus \{0\} \) such that \( \|y\| = 1 \) and \( y^\top x > y^\top z - \epsilon \ \forall z \in K \).

Under these assumptions, the ellipsoid algorithm is guaranteed to terminate in polynomial time. The main idea for the algorithm is as follows:

1. Start with an ellipsoid \( E_0 \) containing \( K \).
2. Check if the center \( c_0 \) of the ellipsoid is in \( K \). If so, we are done. If not, use the separation oracle to determine a hyperplane separating \( K \) and \( c_0 \).

3. Set \( E_1 \) to be the smallest-volume ellipsoid containing the half-ellipsoid that \( (\text{may}) \) contain \( K \). (Showing how this can be done, and analyzing the resulting volume, consists of the “meat” of the algorithm and its analysis.)

4. Continue setting \( E_i \) in this manner. If we ever have \( c_i \in K \), we are done. If \( E_i \subset B(O_1, r) \), we are done, since this contradicts our given condition \( B(O_1, r) \subseteq K \).

Figure 3.2 shows a simple example of the ellipsoid algorithm.

![Figure 3.2: An illustration of the ellipsoid algorithm. The region \( K \) is the feasible region. \( E_0 \) is the first, blue ellipsoid, centered at \( c_0 \). Since \( c_0 \notin K \), we find a separating hyperplane, and let \( E_1 \) entered at \( c_1 \) be the smallest-volume ellipsoid covering the region of \( E_0 \) separated by the hyperplane. \( c_1 \in K \), so we are done.](image_url)

One can show that at each step the volume of the ellipsoid decreases by a constant factor, and it is at most exponential at first. Using the procedure for reducing the optimization problem to the feasibility question, one can show the following:
Theorem 3.3. Let $K \subseteq \mathbb{R}^n$ be convex such that $B(O_1, r) \subseteq K \subseteq B(O_2, R)$, $r, R \in \mathbb{R}$, and assume we have a weak separation oracle for $K$. Let $c \in \mathbb{R}^n$ and $\epsilon > 0$. Then we can compute an $x \in \mathbb{R}^n$ such that $x \in K$ and $y^\top x > y^\top z - \epsilon \forall z \in K$. The number of operations and oracle calls is $\text{poly}(\log R/r + \log 1/\epsilon + n)$.

The ellipsoid algorithm is much more general than just solving LPs, and it is worth knowing when it can be applied. We will see in a later lecture that it also works for SDPs.

- **Interior point methods** Interior point methods start with a solution in $K$, and make incremental steps toward a better solution while staying inside the polytope $K$. The expert weights algorithm can be interpreted as a variant of this class of algorithms, and we will see in a later lecture how it can be applied to efficiently solve certain LPs and SDPs.

### 3.4 Dual of a Linear Program

Any primal LP $\mathcal{P}$ in standard form has an associated dual $\mathcal{D}$ in the following form

$$
\begin{align*}
\min & \quad y^\top b \\
\text{s.t.} & \quad y^\top A \geq c^\top \\
& \quad y \geq 0
\end{align*}
$$

(3.6)

where $x, c, \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$.

**Weak duality.** Any feasible solution $y$ to the dual LP has an interpretation as a proof that the optimum of the primal LP is at most $y^\top b$. This is weak duality. More formally,

**Theorem 3.4.** Suppose an LP $\mathcal{P}$ has dual $\mathcal{D}$ such that both are feasible and bounded, and let $c^\top x = \alpha, y^\top b = \beta$ be optimal values of the objective functions for $\mathcal{P}, \mathcal{D}$ respectively. Then, $\alpha \leq \beta$.

**Proof.**

$$
\alpha = c^\top x = y^\top A x \leq y^\top b = \beta
$$

where the inequality uses $y \geq 0$.

**Strong duality.** Exactly one of the following conditions always holds:

- Both $\mathcal{P}$ and $\mathcal{D}$ are infeasible
- $\mathcal{P}$ is feasible and unbounded, and $\mathcal{D}$ is infeasible
• $\mathcal{D}$ is feasible and unbounded, and $\mathcal{P}$ is infeasible

• Both $\mathcal{P}$ and $\mathcal{D}$ are feasible and bounded, in which case $\alpha = \beta$. In this case the \textit{complementary slackness} condition holds: if a pair $(x, y)$ is optimal, then for every $i$ such that $y_i > 0$, the constraint $A_i x = b$ holds ($A_i$ is the $i$-th row of $A$). Conversely, whenever $A_i x < b$ it holds that $y_i = 0$. 

8