Lecture 10

So far we have seen aspects of the geometry of linear programming, discussed the feasible sets (polyhedra) and presented a useful duality theorem: the optimal value of

$$\text{maximize} \quad \langle \boldsymbol{c}, \boldsymbol{x} \rangle \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \tag{P}$$

coincides with the optimal value of

$$\text{minimize} \quad \langle \boldsymbol{b}, \boldsymbol{y} \rangle \quad \text{subject to} \quad \boldsymbol{A}^\top \boldsymbol{y} = \boldsymbol{c},\ \boldsymbol{y} \geq \boldsymbol{0}, \tag{D}$$

provided both (P) and (D) have a finite solution.

We now turn attention to algorithms for linear programming. We present the main idea behind the classical Simplex Algorithm and then discuss the more modern class of Interior Point Algorithms in more detail. The latter can be shown to be efficient in a very precise sense (polynomial time) and generalize to non-linear convex optimization.

10.1 A first algorithm for linear programming

For $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$ let $P = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$ be the polyhedron of feasible points for (P). Recall that a vertex of P is a zero-dimensional face of P, or equivalently, a point that cannot be written as convex combination of two distinct points of P.

If the optimization problem (P) has a solution, then it can be shown (Problem Set 4) that there exists a vertex $\boldsymbol{x}^*$ such that

$$\max_{\boldsymbol{x}\in P} \langle \boldsymbol{c}, \boldsymbol{x} \rangle = \langle \boldsymbol{c}, \boldsymbol{x}^* \rangle.$$

Intuitively this is clear by looking at a picture, as in Figure 10.1: Move a hyperplane orthogonal to the objective $\boldsymbol{c}$ along this direction to the highest level that still intersects a polyhedron P, and try to imagine that this intersection *does not* contain a vertex.

This crucial observation turns linear programming into a *finite* problem: we only have to test the objective function on the finite set of vertices to determine the maximizer. We have also seen (Problem Set 3) that the vertices can be identified as the solutions of the systems of equations

$$\boldsymbol{A}_I \boldsymbol{x} = \boldsymbol{b}_I,$$

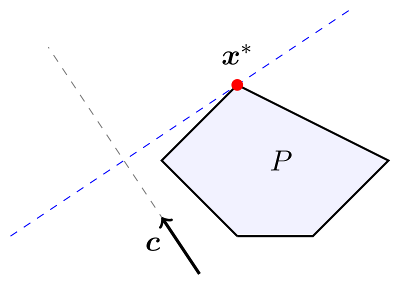
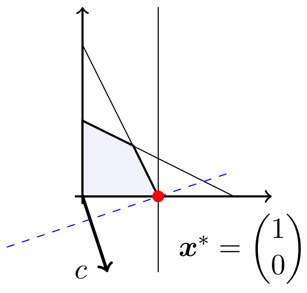


Figure 10.1: Geometry of linear programming

for which $\boldsymbol{A}_I$ has full rank and $\boldsymbol{x} \in P$, where $I \subset \{1,\dots,m\}$ is a subset of indices with $|I| = d$, and $\boldsymbol{A}_I, \boldsymbol{b}_I$ are the matrix and vector arising from $\boldsymbol{A}$ and $\boldsymbol{b}$ by taking the rows indexed by I. This gives a first primitive algorithm for solving the optimization problem (P).

Example 10.1. Consider the linear programming problem

$$\begin{aligned} \text{maximize} \quad & x_1 - 3x_2 \\ \text{subject to} \quad & 0.5x_1 + x_2 \leq 1 \\ & x_1 + 0.5x_2 \leq 1 \\ & -x_1 \leq 0 \\ & -x_2 \leq 0 \\ & x_1 \leq 1 \end{aligned}$$



The matrix $\boldsymbol{A}$ and vectors $\boldsymbol{b}$ and $\boldsymbol{c}$ are

$$\boldsymbol{A} = \begin{pmatrix} 0.5 & 1 \\ 1 & 0.5 \\ -1 & 0 \\ 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{c} = \begin{pmatrix} 1 \\ -3 \end{pmatrix}.$$

We see that the minor $\boldsymbol{A}_{\{3,5\}} = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}$, corresponding to the two parallel hyperplanes $x_1 = 0$ and $x_1 = 1$, is singular and does not define a vertex, while all other minors are invertible. For the minor $I = \{1,2\}$, $\boldsymbol{A}_I\boldsymbol{x} = \boldsymbol{b}_I$ has the form

$$\begin{aligned} 0.5x_1 + x_2 &= 1 \\ x_1 + 0.5x_2 &= 1, \end{aligned}$$

which has the solution $\boldsymbol{x} = (2/3, 2/3)^\top$. Since $\boldsymbol{x}$ also satisfies all the other inequalities, it is a vertex. In a similar fashion we can find all the vertices as the points

$$\boldsymbol{x}_1 = \begin{pmatrix} 2/3 \\ 2/3 \end{pmatrix}, \ \boldsymbol{x}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ \boldsymbol{x}_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \ \boldsymbol{x}_4 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

The values of the objective function at these vertices are

$$\langle \boldsymbol{c}, \boldsymbol{x}_1 \rangle = -\frac{4}{3},\ \langle \boldsymbol{c}, \boldsymbol{x}_2 \rangle = 1,\ \langle \boldsymbol{c}, \boldsymbol{x}_3 \rangle = -3,\ \langle \boldsymbol{c}, \boldsymbol{x}_4 \rangle = 0.$$

It follows that the optimization problem has the optimal value 1 at $\boldsymbol{x}^* = (1,0)^\top$.

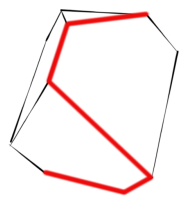
Unfortunately, this method requires solving up to $\binom{m}{n}$ systems of linear equations, which can be exponential in n in the worst case, making this method not practical.

Example 10.2. Consider the hypercube

$$\begin{gathered} -1 \leq x_1 \leq 1 \\ \cdots \\ -1 \leq x_n \leq 1. \end{gathered}$$

It has 2^n vertices that need to be checked. Even though there are so many vertices, one can travel along edge between any two vertices in at most n steps.

The simplex algorithm is a way to search through the vertices in a more clever way than just listing all of them. The idea is to start with a vertex and see if there is a vertex connected to it by an edge that has a bigger objective value. If not, we are done. If there is a neighbouring vertex, we move there and continue the process. We keep walking along the vertices of the feasible polytope until we find an optimal one. The



simplex algorithm is one of the most successful algorithms around and usually very fast, even though there examples that show that its worst case running time can be exponential.