Linear Programming: Simplex

Stephen J. Wright[1]

[2]Computer Sciences Department,
University of Wisconsin-Madison.

IMA, August 2016

Linear Programming

Vector of continuous variables $x \in \mathbb{R}^n$, linear objective, linear constraints.

Standard form:

$$\min c^T x \text{ s.t. } Ax = b, \quad x \geq 0.$$

We assume that $A \in \mathbb{R}^{m \times n}$ (with $m < n$) has full row rank.

Any problem with linear objective and linear constraints can be converted to this form by adding / subtracting slacks, splitting variables.

Note: All variables are continuous — readl numbers! Problems in which some components x_i are required to be binary or integer are not covered. These binary linear programs or integer linear programs are much harder and require much different methodology (though simplex is a part of this methodology).

Main ref: (Nocedal and Wright, 2006, Chapter 13).

Basic Points

The feasible set is a polyhedron in $\mathbb{R}^n$: a set with flat faces, edges, and vertices. A *vertex* is a point that doesn't lie on a line between two other feasible points.

Vertices are important in linear programming because *if the LP has a solution, then at least one of its solutions is a vertex*. Thus, in seeking a solution, we can restrict our attention to vertices. But we can't look at all vertices — there are too many in general (up to $\binom{n}{m}$).

Each vertex can be represented as a basic point (traditionally known as a *basic feasible solution*) defined by a square nonsingular column submatrix of A that contains m columns. This matrix is called a basis, denoted by $B \in \mathbb{R}^{m \times m}$.

After a permutation, partition A into basic and nonbasic columns:

$$\begin{bmatrix} B & N \end{bmatrix}.$$

Basic Points

We can partition x accordingly as

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix}.$$

The key is to set $x_N = 0$ and let x_B be defined by the linear equalities:

$$b = Ax = Bx_B + Nx_N,\ x_N = 0 \quad \Leftrightarrow \quad x_B = B^{-1}b.$$

In order for this to be a *feasible* point, we require $x_B \geq 0$. Can't take just any set of m columns of A.

A single vertex can be defined by number of different choices of B. These happen when x_B has some zero components. Such vertices are called denegerate.

(Some pictures here....)

Simplex: One Step

Simplex moves from one basic point to an adjacent basic point, by sliding along one edge of the feasible polyhedron.

The new basic point differs from the current point in that one column is swapped between B and N. We choose the new point to have a lower function value than the current point.

How to choose the new point? Substitute for x according to the current (B, N) decomposition:

$$x_B = B^{-1}(b - Nx_N) = B^{-1}b - B^{-1}Nx_N.$$

Then transform the LP to

$$\min_{x_N} c_B^T B^{-1}(b - B^{-1}Nx_N) + c_N^T x_N \quad \text{s.t. } x_N \geq 0, \quad x_B = B^{-1}(b - Nx_N) \geq 0,$$

which is equivalent to

$$\min_{x_N} c_B^T B^{-1}b + (c_N^T - c_B^T B^{-1}N)x_N \quad \text{s.t. } x_N \geq 0 \quad x_B = B^{-1}b - B^{-1}Nx_N \geq 0.$$

Simplex: One Step

Strategy: Choose an index $j \in N$ such that the j component of $c_N^T - c_B^T B^{-1} N$ is negative. Let this component x_j increase away from 0. This has the effect of decreasing the objective.

BUT it also causes x_B to change. We need to stop increasing x_j when one of the components of x_B hits zero. Precisely, if $N_{(j)}$ denotes the column of N corresponding to x_j, we have

$$x_B = B^{-1} b - B^{-1} N_{(j)} x_j \geq 0.$$

We set the new value of x_j to be the maximum value for which this inequality still holds.

Column $N_{(j)}$ enters the basis matrix B. We move the column of B that corresponds to the component of x_B that just became zero into N.

At the end of this iteration we still have B square and nonsingular, the new x_B is nonnegative, and the new x_N is all zero. That's one step!

A Few Complications

- What if $(c_N^T - c_B^T B^{-1} N)$? Solution, baby!
- What if $B^{-1} N_{(j)} \leq 0$, so that we can increase x_j without limit while maintaining $x_B \geq 0$. Then the LP is unbounded. Trivial example: min $-x$ s.t. $x \geq 0$.
- What if more than one component of x_B reaches zero at the same time? Then just pick one of them to swap with $N_{(j)}$.
- What if one of the x_B components is already zero (degenerate), and we can't increase x_j away from zero *at all* without making it go negative? This is a degenerate pivot. There's no decrease in objective, but swap $N_{(j)}$ with the offending column of B anyway. The next pivot (or the one after...) may yield a decrease in objective.
- What if we do a string of degenerate pivots, and end up with some basis matrix B that we've encountered earlier. This is cycling. It can be overcome by applying some rules for choosing $N_{(j)}$ judiciously. (There is usually more than one candidate for $N_{(j)}$.)

Details: Maintaining a Factorization of B

For LP of practical size, we never want to compute B^{-1} directly — it's too large. Instead we can maintain LU factors of some permuted version of B, say $PB\bar{P}^T$.

Can exploit the fact that B changes in just one column during each step of simplex. Thus the same L factor almost works, but U now has a "spike" in the location of the replaced column.

By applying some permutations and making some small modifications to L, we can restore U to upper triangular form. See (Nocedal and Wright, 2006, Section 13.4).

Thus the L factor is stored in factored form. Occasionally, a fresh factorization of B is computed, to avoid possible buildup of error and blowup of storage.

Choosing $N_{(j)}$: Pricing

Almost any negative component of $(c_N^T - c_B^T B^{-1} N)$ will lead to a decrease in objective. Usually there is more than one. How do we choose?

This operation is called pricing and it's a critical operation in practical LP implementation. Possible strategies:

- Choose the most negative element of $(c_N^T - c_B^T B^{-1} N)$, as this leads to the biggest decrease in objective per unit increase in x_j. Problem: We may not be able to increase x_j very much before some component of x_B hits zero.
- Choose the j that yields the steepest decrease *per unit distance moved along the edge of the feasible polyhedron* (that is, considering the changes in x_B components as well as x_j). This "steepest-edge" strategy was proposed in the 1990s and is quite effective.
- Choose the component j with the smallest index among the possible options. This is guaranteed to prevent cycling, but is not otherwise the most practical.

Partial Pricing

In any case, it is impractical to maintain and update the full vector $(c_N^T - c_B^T B^{-1} N)$ — in fact this would be the most expensive operation in simplex calculation.

In practice, just a subvector is maintained — a subset of the full N. We ignore the rest of the matrix until we have exhausted this subset, i.e. this subvector of $c_N^T - c_B^T B^{-1} N$ becomes nonnegative. Then we move to a fresh subset of N.

Phase I

We need to find a starting point: An intial choice of B and N such that B is square and nonsingular and $B^{-1}b \geq 0$. How?

Answer: Construct a modified (but related) problem for which the initial basis B is easy to identify. Do simplex on this problem until we find a suitable basis for the original problem.

Specifically, add m extra variables (say $z \in \mathbb{R}^m$) and solve:

$$\min_{(x,z)} e^T z \quad \text{s.t.} \quad \begin{bmatrix} A & E \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = b, \quad \begin{bmatrix} x \\ z \end{bmatrix} \geq 0,$$

where $e = (1, 1, \ldots, 1)^T$ and E is a diagonal matrix with ± 1 on the diagonal. Note that we've discarded the original objective $c^T x$.

Now set $x = 0$ and $E_{ii} = \text{sign}(b_i)$, $z_i = |b_i|$ for $i = 1, 2, \ldots, m$. Initial basis is $B = E$, with $N = A$. This is a Phase I LP.

Phase I

Now apply simplex from this starting basis to Phase I. Two outcomes:

- Have objective strictly positive. Then the original problem is infeasible: it's not possible to find an x with $Ax = b$, $x \geq 0$.
- Have objective $e^T z = 0$, so that $z = 0$. Then proceed to the solution of the real problem by
  - replacing the Phase I objective with the original objective $c^T x$;
  - set upper bounds on z of zero. (We need to modify simplex slightly to handle upper bounds.) This ensures that from here on we solve the original problem, i.e. z stays at zero.

Why not simply remove z from the problem at the end of Phase I? Because some of the z components may still be degenerate components of the basis B.

Duality!

We've come a long way without discussing one of the most intriguing properties in linear programming: **DUALITY**.

Duality is a powerful mathematical theory that's also of great practical importance. It plays a vital role too in other areas of convex optimization.

Given the data objects A, b, c that define an LP, we can construct another LP called the dual LP from the same objects:

$$\text{(DUAL)} \quad \max_{\lambda,s} \; b^T\lambda \;\; \text{s.t.} \;\; A^T\lambda + s = c, \;\; s \geq 0.$$

Note that $\lambda \in \mathbb{R}^m$ and $s \in \mathbb{R}^n$. The original LP is called the "Primal" to distinguish it from the dual.

Weak Duality

$$\text{(P)} \quad \min\ c^T x \ \text{ s.t. } \ Ax = b, \ \ x \geq 0.$$

$$\text{(D)} \quad \max_{\lambda, s}\ b^T \lambda \ \text{ s.t. } \ A^T \lambda + s = c, \ \ s \geq 0.$$

The two problems say a great deal about each other. One simple but useful relationship is weak duality, which says:

> *If x is feasible for (P) and (λ, s) is feasible for (D), then $c^T x \geq b^T \lambda$.*

Proof.

$$c^T x = (A^T \lambda + s)^T x = \lambda^T (Ax) + s^T x \geq \lambda^T b.$$

□

Practical application: A feasible point for (D) is sometimes easy to find, and it gives a lower bound on the optimal value for (P).

Strong Duality

The other key duality result is strong duality and its proof requires a lot more than one line.

Theorem

Exactly one of these three statements is true:

(i) *(P) and (D) both have solutions, and their objectives are the same.*

(ii) *One of (P) and (D) is infeasible and the other is unbounded.*

(iii) *Both (P) and (D) are infeasible.*

It excludes some plausible possibilities. e.g. if we find that (D) is infeasible, it's impossible for (P) to have a solution. If we find that (P) is unbounded, then (D) must be infeasible — it can't be unbounded or have a solution.

Parts (ii) and (iii) are easy to prove (do it!) but part (i) is hard. The conventional proof argues that "the simplex method works, and it identifies solutions to both (P) and (D), with equal objectives."

KKT Conditions

We can use strong duality to derive a set of primal-dual optimality conditions. (These can be generalized to nonlinear programming, as we see later.)

These are sometimes called the Karush-Kuhn-Tucker (KKT) conditions, after their inventors in 1948 and 1951.

$$Ax = b, \quad A^T\lambda + s = c, \quad 0 \leq x \perp s \geq 0.$$

where $x \perp s$ indicates that $x^T s = 0$.

KKT conditions are just feasibility conditions for (P) and (D), together with $x^T s = 0$. This condition ensures that the duality gap is zero, i.e. no gap between primal and dual objectives. (See weak duality proof.)

The KKT conditions are the key to deriving primal-dual interior-point methods, which we'll discuss later.

Farkas Lemma

Strong Duality can be used to prove some powerful results. An important one is the Farkas Lemma, which is critical to optimality theory for constrained optimization.

Lemma

Given a collection of vectors $a_i \in \mathbb{R}^n$, $i = 1, 2, \ldots, m$ and a vector $b \in \mathbb{R}^n$, exactly one of the following claims is true:

(1) *b is a convex combination of the a_i, that is, there exist $x_i \in \mathbb{R}_+$, $i = 1, 2, \ldots, m$ such that $b = \sum_{i=1}^m x_i a_i$.*

(2) *there is a vector $\lambda \in \mathbb{R}^m$ such that $b^T \lambda > 0$ and $a_i^T \lambda \leq 0$ for all $i = 1, 2, \ldots, m$.*

Farkas Proof

Proof.

We set up a primal-dual pair of LP given the data. Define $A = [a_1 : a_2 : \ldots : a_m]$, $x = (x_1, x_2, \ldots, x_m)^T$ and

$$(\text{P}) \quad \min_x \ 0^T x \ \text{ s.t. } \ Ax = b, \ \ x \geq 0.$$

Note: zero objective vector ($c = 0$). The corresponding dual is:

$$(\text{D}) \quad \max_{\lambda, s} \ b^T \lambda \ \text{ s.t. } \ A^T \lambda + s = 0, \ \ s \geq 0.$$

If statement (1) is true, then (P) is feasible and has optimal objective zero. Thus (D) is also feasible with optimal objective zero. Thus (2) cannot be true, since if it were, λ would be a feasible point for (D) with positive objective.

If statement (1) is not true, then (P) is infeasible. But (D) is clearly *not* infeasible, since $\lambda = 0$, $s = 0$ is a feasible point. Thus by strong duality (case (ii)), (D) is feasible and unbounded. Thus (2) is true. □

Dual Simplex

An important variant of the simplex method works with the primal formulation (P), but is based on duality. The idea is to start with an initial basis that is dual feasible i.e. satisfies $c_N^T - c_B^T B^{-1} N \geq 0$, but does not satisfy the constraints of (P), that is, $B^{-1} b \not\geq 0$.

Simplex pivots maintain the dual feasiblity property $c_N^T - c_B^T B^{-1} N \geq 0$ and gradually iterate toward satisfying $B^{-1} b \geq 0$.

Why the term "dual feasible"? Because this choice of B can be used to construct a feasible point for the dual, defined by:

$$\lambda = B^{-T} c_B, \quad s_B = 0, \quad s_N = c_N - N^T \lambda = c_N - N^T B^{-T} c_B \geq 0.$$

Dual Simplex Iterations

Each step proceeds by:

- Choose an index i for which the component of x_B is negative. Consider the corresponding element of s_B.
- Allow s_i to increase away from zero. This changes all components of λ, since

$$\lambda = B^{-T}(c_B - s_i e_i),$$

  where $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)^T$, with the 1 in the location corresponding to s_i. This in turn changes the values of $s_N = c_N - N^T\lambda$. We increase s_i as much as possible away from 0 while maintaining $s_N \geq 0$.
- We switch the column corresponding to s_i out of B and replace it with the column of N that corresponds to the new zero component in s_N.

We can show that each iteration increases the value of $b^T\lambda$, thus moves (D) toward better and better points.

Keep track of values of x throughout the process.

Presolving

Practical LP codes have presolvers, which use "common sense" to eliminate some variables and constraints before actually applying simplex or interior-point methods to solve the problem.

There are many "tricks." Some very simple ones:

- Row singleton: $3x_{10} = 13$. Then (obviously), we can set $x_{10} = 13/3$ and substitute it out of the problem, thus eliminating one variable and one constraint.
- Forcing Constraints: A combination of a constraint and bounds can force variables to the bounds. Example: $x_1 + x_3 + x_7 = 3$, $x_1, x_3, x_7 \in [0, 1]$. Then we must have $x_1 = 1$, $x_3 = 1$, $x_7 = 1$. These variables can be fixed and removed from the problem.
- Dominated constraints. Given constraints

$$3x_1 + 2x_5 \geq 4, \quad 4x_1 + 3x_5 \geq 2, \quad x_1 \geq 0, \; x_5 \geq 0,$$

  we can remove the constraint $4x_1 + 3x_5 \geq 2$.

Presolving can be applied in rounds.

Topics

Complexity: exponential in worst case. There are examples. Is there a pricing rule that makes it polynomial?
Average-case analysis.
Rounded analysis.

References I

Ferris, M. C., Mangasarian, O. L., and Wright, S. J. (2007). *Linear Programming with* MATLAB. MOS-SIAM Series in Optimization. SIAM.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York.