The Primal-Dual Method

Eric Bach
CS 787
Fall 2018

The primal-dual method is a "master plan" for the design of combinatorial algorithms. It applies to problems that can be expressed as linear programs. It works by reducing the given problem to a sequence of simpler LP's, which can often be given purely combinatorial interpretations.

Start with an LP in equality form:

$$\begin{aligned} & \min cx \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0. \end{aligned}$$

We also assume $b \geq 0$, as in the simplex algorithm.

The dual of this is

$$\begin{aligned} & \max yb \\ \text{s.t.} \quad & yA \leq c. \end{aligned}$$

Note that the dual vector is not sign-constrained.

One can show that if both problems have finite optima, the solutions are characterized by complementary slackness. This says that each vector is orthogonal to the other's residual, i.e.

$$y(Ax - b) = 0 \tag{1}$$

$$(yA - c)x = 0 \tag{2}$$

Note that the first always holds when x is primal feasible. So the second one is what concerns us.

The main idea is to find a sequence of better and better dual-feasible vectors. If this is successful, we will reach optimality in a way that reveals the desired primal-optimal vector:

$$\begin{array}{ccc} \text{dual} & & \text{primal} \\ & & \\ \cdots y \longrightarrow y' \rightarrow y'' \rightarrow \cdots & | & XXXXXXXXXXX \cdots \\ & \uparrow & \\ & x & \end{array}$$

We now describe the basic cycle of the procedure.

Let y be dual feasible. Plug it into (2), the second complementary slackness condition, and get

$$(yA - c)x = (00\cdots 0\underbrace{**\cdots *}_{<0})(x_1 x_2 \cdots x_r 00\cdots 0)^T = 0.$$

As indicated, some components of x must be 0, but the others, say the first r, need not be. Call this "free" part x'. We want to find x', i.e. solve

$$A'x' = b, \quad x' \geq 0, \tag{3}$$

where A' denotes the "interesting" part of A (its first r columns). (The rest of A doesn't matter since we multiply those columns by zero.) Thus, we seek a feasible point of (3). This might be easier than our initial problem, for two reasons:

i) The matrix is smaller;

ii) The cost vector c has disappeared, or as some say, "been combinatorialized."

The standard attack on (3) is to introduce slack variables (a vector w, say), and solve the LP

$$\begin{array}{rl} & \min \sum_i w_i \\ \text{s.t.} & (A'I)(x'w)^T = b \\ & x', w \geq 0. \end{array} \tag{4}$$

Traditionally this is called a "restricted primal," although it isn't really a restriction (we added some variables). Note that we are trying to minimize some measure of infeasibility.

If the min is 0, we have got the x' we want, and we're done.

Otherwise, consider the dual of (4). It is

$$\begin{array}{rl} & \max zb \\ \text{s.t.} & z(A'I) \leq (00\cdots 011\cdots 1). \end{array}$$

Equivalently, the dual is

$$\begin{array}{rl} & \max zb \\ \text{s.t.} & zA' \leq 0 \\ & z_i \leq 1, \ \text{all } i. \end{array} \tag{5}$$

Suppose we can solve this. The solution z is a dual vector that we can use to improve y. Why? zb is positive! (This is a consequence of strong duality.) Now we need to worry about dual feasibility when we modify y by adding some multiple of z. Consider

$$zA = z(A'A'')$$

There is no problem with the first part, since $zA' \leq 0$. However, zA'' could have some positive components. But we remember that in these components, yA is strictly less than c, so there will be *some* $\lambda > 0$ for which

$$y' := y + \lambda z$$

is an improved dual vector, in the sense of making yb larger. We take λ as large as possible subject to the requirement of dual feasibility.

At this point we replace y by y' and the cycle begins anew.

Comments.

1. The spirit of this is similar to iterative algorithms like Newton iteration:

   $$x' := x - f(x)/f'(x).$$

   Here, failure ($f(x) \neq 0$) suggests a way to improve x for the next try.
2. Often this works because we can find combinatorial interpretations, and hence efficient algorithms, for the "inner" LPs. The book of Papadimitriou and Steiglitz on combinatorial optimization has many examples, including shortest path, maximum flow, job assignment, optimal transportation, and min-cost flow.
3. We didn't prove termination, just that the dual objective improves. In cases of interest there is usually an independent way to show termination.