COMPSCI 330: Design and Analysis of Algorithms Spring 2019

Lecture 18: Linear Programming Algorithms

Lecturer: Rong Ge *Scribe: Shweta Patwa*

1 Duality

Recall the following canonical forms:

$$(P): \min < c, x > \quad Ax \geq b \quad x \geq 0$$

$$(D): \max < b, y > \quad A^T y \leq c \quad y \geq 0$$

Number of variables in the primal LP will equal the number of constraints in the dual LP.
Number of constraints in the primal LP will equal the number of variables in the dual LP.

Here, a feasible solution to the primal gives an upper bound to the optimal solution, whereas a feasible solution to the dual gives a lower bound on the optimal solution.

Strong duality: Both LPs have same optimal value.
Thus, if you have a feasible solution to each primal and dual, and they have the same value, then they are also the optimal solutions to the respective LPs.

Note: The dual of the dual LP is the primal LP.

The example from last lecture is as follows:

$$\begin{aligned} \min \quad & 2x_1 - 3x_2 + x_3 \\ s.t. \quad & x_1 - x_2 \geq 1 \\ & x_2 - 2x_3 \geq 2 \\ & -x_1 - x_2 - x_3 \geq -7 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Optimal solution: x=(4, 3, 0)

$$\begin{aligned} \max \quad & y_1 + 2y_2 - 7y_3 \\ s.t. \quad & y_1 - y_3 \leq 2 \\ & -y_1 + y_2 - y_3 \leq -3 \\ & -2y_2 - y_3 \leq 1 \\ & y_1, y_2, y_3 \geq 0 \end{aligned}$$

Optimal solution: y=(2.5, 0, 0.5)

We can rewrite these as follows:

$$\min \begin{pmatrix} 2 & -3 & 1 \end{pmatrix} \cdot x$$
$$s.t. \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -2 \\ -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \geq \begin{pmatrix} 1 \\ 2 \\ -7 \end{pmatrix}$$
$$x_1, x_2, x_3 \geq 0$$

$$\max \begin{pmatrix} 1 & 2 & -7 \end{pmatrix} \cdot y$$
$$s.t. \begin{pmatrix} 1 & 0 & -1 \\ -1 & 1 & -1 \\ 0 & -2 & -1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \leq \begin{pmatrix} 2 \\ -3 \\ 1 \end{pmatrix}$$
$$y_1, y_2, y_3 \geq 0$$

The dual optimal solution is $y_1 = 2.5, y_2 = 0, y_3 = 0.5$. Thus, multiplying the primal constraints with y_1, y_2 and y_3, respectively gives us

$$\begin{aligned} & y_1(x_1 - x_2 \geq 1) + y_2(x_2 - 2x_3 \geq 2) + y_3(x_1, x_2, x_3 \geq 0) \\ \implies & 2x_1 - 3x_2 - 0.5x_3 \geq -1 \\ \implies & 2x_1 - 3x_2 + x_3 \geq 2x_1 - 3x_2 - 0.5x_3 \geq -1 \qquad (x_3 \geq 0) \end{aligned}$$

Thus, the primal objective is greater than or equal to -1.

We need the variables to be non-negative so that the above analysis goes through and we are able to relate the values of feasible solution for the dual to the primal optimal value, and vice-versa.

The primal optimal solution is $x_1 = 4, x_2 = 3$ and $x_3 = 0$.

$$\begin{aligned} & 4(y_1 - y_3) + 3(-y_1 + y_2 - y_3) + 0(-2y_2 - y_3) \\ \implies & y_1 + 3y_2 - 7y_3 \leq -1 \\ \implies & y_1 + 2y_2 - 7y_3 \leq y_1 + 3y_2 - 7y_3 \leq -1 \qquad (y_2 \geq 0) \end{aligned}$$

Thus, the dual objective is always less than or equal to -1.

1.1 Complementary slackness

We call a constraint tight if instead of the inequality, the equality holds true. Consider the primal LP and the optimal solution $(4, 3, 0)$ above. Then the first and third constraints are tight, but not the second constraint. In addition, $x_3 \geq 0$ is also tight.
Similarly, the first and second constraints in the dual LP are tight with respect to the optimal solution $(2.5, 0, 0.5)$. Also, $y_2 \geq 0$ is a tight constraint.
Let us look at dual variables corresponding to the tight primal constraints. In the primal we have exactly two tight constraints, and exactly the matching dual variables are non-zero. This is true of the dual constraints and primal variables as well. This is not a coincidence, and we formalize these as complementary slackness below.

(Def) Let x and y be optimal solutions of primal and dual LPs.

1. If the i^{th} constraint in the primal is **not** tight, then the i^{th} variable of the dual is equal to zero.
2. If the i^{th} constraint in the dual is **not** tight, then the i^{th} variable in the primal is equal to zero.

Primal slack of i = LHS of primal constraint i - RHS of primal constraint i.
Example. Primal slack (1) = $x_1 - x_2 - 1$. Then, primal slack $(i) * y_i = 0$.

What is the benefit of knowing these constraints?
The primal and dual specific problems, say the primal LP is for the shortest path problem (see hw7). Then, complementary slackness tells us which edges will be on the shortest path and which ones will not be.
When trying to find a dual LP solution, then knowing which primal constraints are not tight tells us which dual variables will be zero.
When we are trying to get a tight lower bound on the primal objective, and say the second constraint is not tight. This means that using such a constraint will not be very useful when proving the bound.

2 LP algorithms

Given an LP, how do we find its optimal solution?

Many algorithms (fairly complex and people end up using available packages/solvers):

1. Simplex
2. Ellipsoid
3. Interior point

2.1 Simplex

Let us revisit the geometric interpretation of LPs with two variables. Here, each constraint can be thought of as a half-space, and the set of feasible solutions is the intersection of all these half-spaces. The objective function works as the direction of gravity. The optimal solution then corresponds to the lowest point in this convex polygon/polytope if we were to rotate the feasible region so that the direction given by the objective, i.e., gravity, points vertically downwards.

Basic feasible solution: (Geometric) A vertex of the feasible region.
(Linear algebraic) A feasible solution where there are n tight constraints with linearly independent coefficients.

Example. $(4,3,0)$ is a basic feasible solution the the primal LP above because it is a solution of the three tight constraints: $x_1 - x_2 = 1, -x_1 - x_2 - x_3 = -7$ and $x_3 = 0$.

We can enumerate all basic feasible solutions and compute an optimal solution by solving the corresponding system of (tight) constraints and picking the solution that optimizes the objective. But this is not a very efficient method. In the above example, choosing 3 constraints out of the 6 given constraints already gives us 20 different systems of linear equations to solve.

Simplex algorithm gives you a *better* way to find an optimal solution.

Claim. There is always an optimal basic feasible solution.

However, it is not always true that an optimal solution must be a basic feasible solution. The reason is that if we change the objective function, i.e., the direction of gravity, then multiple points along an edge of the feasible region can correspond to optimal solutions.

Idea: Start with a basic feasible solution and follow an edge in the polytope.
(There are ways to find a basic feasible solution to start with. We skip the details here.)
Algorithmically: Follow an edge $\iff$ swap a constraint. Many ways to decide which constraint to swap.

Running time: each move takes polynomial time. How many times do we need to move to get to an optimal solution? In the worst-case with n variables and m constraints, it can take 2^n moves to reach a global optimal solution. In practice, the algorithm is very fast. It is one of the most popular algorithms in the packages solving LPs. Explanation/theoretical reasons for this are beyond the scope of the class.

2.2 Ellipsoid

Idea: We can add a constraint that specifies a bound on the value that the objective can take. If there are still feasible solutions, we can move this bound further in the direction of gravity until there are no more feasible solution to the systems of linear equations with this new, added constraint. We can perform a binary search in this range of bounds and try to identify the point where the intersection of the (new) half-space and the original set of feasible solutions is exactly one point.

How do we find a feasible solution?
Algorithmically: The algorithm starts with a big ball that contains the entire feasible region inside it. We look at the center of this ball and see if it is a feasible solution. If not, then we can identify the violated constraint. We know that the actual feasible solution must then lie in the intersection of the half-space specified by the violated constraint and the previous ellipsoid. The algorithm then finds a new ellipsoid that contains this entire intersection. Repeat!
The size/volume of the ellipsoid decreases by at least half in each step. Eventually we either find a feasible solution or the ellipsoid becomes so small that we can conclude that there are no feasible solutions.

Running time: This is faster than the Simplex algorithm. The running time is polynomial in the number of variables and constraints. Actual running time is slow, so this algorithm is often used for low dimensional problems.

2.3 Interior point algorithms

Idea: We try to smooth out the very hard threshold for constraints. Build barrier functions for constraints. These are functions that are small for the feasible part of the constraint, but go to infinity at the boundary (for a minimization problem).

Algorithmically: First find the optimal solution for the objective plus all the barrier functions. This will result in a point that cannot be on the boundary on the feasible region by definition of barrier functions. This point will lie in the interior of the feasible region.
Next, gradually decrease the barrier functions to make the point closer to the actual optimal. Thus, the influence of the objective becomes larger while that of the barrier functions goes down. We skip details here.

Running time: It is polynomial in the number of variables and constraints, and can be implemented very efficiently in practice.