Lecture 21: Linear Programming Algorithms

Scribe: Weiyao Wang

November 16, 2017

1 Lecture Overview

Last time we were talking about linear programming duality. Today we are going to solve linear programming slower.

2 Problem Continued: Duality of Linear Programs

Consider the following linear program:

- min $2x_1 - 3x_2 + x_3$
- $x_2 - x_2 \geq 1$
- $x_2 - 2x_3 \geq 2$
- $-x_1 - x_2 - x_3 \geq -7$
- $x_1, x_2, x_3 \geq 0$

The problem we ask for these linear programs are like "How can I prove th you that optimal solution is **at most** -1" and "How can I prove th you that optimal solution is **at least** -1".

For the first question, we can given an answer $(4, 3, 0)$ that gives the objective function with value -1 and satisfies the constraints.

But to show that the optimal solution is exactly -1, we need to prove the other direction: the optimal solution is at least -1. This requires us to think about the dual of the program. In this case, we want to show $2x_1 - 3x_2 + x_3 \geq -1$. Our strategy here is to use a linear combination of the constraints.

2.1 How to prove new inequalities?

There are two ways to prove additional inequalities based on the given constraints.

The first way, say we want to prove $2x_1 - x_2 + x_3 \geq 1$. We can break the inequalities into the constraints:

$$2x_1 - x_2 + x_3 \geq x_1 - x_2 \geq 1$$

where the first inequality comes from $x_1, x_2, x_3 \geq 0$, and the second inequality comes from the first constraint.

General Form: if $a_1 \geq b_1, a_2 \geq b_2, a_3 \geq b_3, x_1, x_2, x_3 \geq 0$

$a_1x_1 + a_2x_2 + a_3x_3 \geq b_1x_1 + b_2x_2 + b_3x_3$

The second way, for example, we want to prove $x_1 - 2x_3 \geq 3$, we can add the first two constraints, where we have $x_1 - 2x_3 = x_1 - x_2 + x_2 - 2x_3 \geq 3$. Thus, we can take a linear combination of these constraints with positive coefficients to find new inequalities.

These two are essentially the same, since for example, for the first method, we are just adding the first constraint with the fact that $x_1, x_2, x_3 \geq 0$. It is a default constraint for linear program of canonical form, and thus, is a special case for the second method.

Now, using these two methods, we can show the objective $2x_1 - 3x_2 + x_3 \geq -1$.

2.2 Proving the lower bound for objective function

Suppose the first condition $x_2 - x_2 \geq 1$ is $x(1)$ and the third condition $-x_1 - x_2 - x_3 \geq -7$ is $x(3)$. Then we have $2.5 \times x(1) + 0.5 \times x(3)$, which gives

$$2.5x_1 - 2.5x_2 + 0.5(-x_1 - x_2 - x_3) \geq 2.5 + 0.5(-7) = -1$$

After simplification, we have

$$2x_1 - 3x_2 - 0.5x_3 \geq -1$$

The above method is the second method. Recall we want $2x_1 - 3x_2 + x_3 \geq -1$. We will use the first method to prove it:

$$2x_1 - 3x_2 + x_3 \geq 2x_1 - 3x_2 - 0.5x_3 \geq -1$$

We have finish the prove, but this is not satisfying, since the choice of coefficients we use for the second method seems to be lucky. So the remaining question is how we can choose the correct coefficients.

2.3 How to choose the coefficients to multiply the constraints with?

What we are going to see is if you want choose the best coefficients in order to prove the lower bound of the objective function that is as strong as possible, you can formulate the problem of choosing coefficients as another linear program, and it is the dual. Suppose the coefficients for the first, second, third conditions are y_1, y_2, y_3. The constraints for y_1, y_2, y_3 are given by

- $y_1, y_2, y_3 \geq 0$, which makes sure that the direction of inequality is preserved
- $y_1 x(1) + y_2 x(2) + y_3 x(3)$, where $x(i)$ stands for the inequality of the i^{th} constraint, grouping them, we have

$$(y_1 - y_3)x_1 + (-y_1 + y_2 - y_3)x_2 + (-2y_2 - y_3)x_3 \geq y_1 + 2y_2 - 7y_3$$

What we want is that $2x_1 - 3x_2 + x_3 \geq LHS \geq y_1 + 2y_2 - 7y_3$, so we need to do so by focusing on condition where the first inequality holds, which is given by

1. $y_1 - y_3 \leq 2$
2. $-y_1 + y_2 - y_3 \leq -3$
3. $-2y_2 - y_3 \leq 1$
4. Want the strongest inequality $\rightarrow$ maximize the right hand side of the inequality: $y_1 + 2y_2 - 7y_3$

This again is a linear program, so to find the coefficients y, we need to solve this linear program. That's why this is called a dual linear program, and to prove the lower bound for the original objective program, we need to solve the dual linear program.

2.4 Summary of Primal and Dual Linear Program

Original problem:

- min $2x_1 - 3x_2 + x_3$
- $x_2 - x_2 \geq 1$
- $x_2 - 2x_3 \geq 2$
- $-x_1 - x_2 - x_3 \geq -7$
- $x_1, x_2, x_3 \geq 0$

Dual problem:

- max $y_1 + 2y_2 - 7y_3$
- $y_1 - y_3 \leq 2$
- $-y_1 + y_2 - y_3 \leq -3$
- $-2y_2 - y_3 \leq 1$
- $y_1, y_2, y_3 \geq 0$

where the solution for y is $(2.5, 0, 0.5)$. A remark here is that the number of variables for dual program is the same as the number of constraints for primal program, but necessarily the number of variables. Every constraint for dual program also corresponds to a variable in the primal program. Another note is that the dual of the dual is again the primal.

Also we notice that in the solution for the primal $x = (4, 3, 0)$, where the first and the third constraints are tight. The second is not tight. We see that the second variable for dual is 0, which is generally true: a non-tight constraint of the primal gives a variable in the dual with value 0. Intuitively, this makes sense since we do not want to use a non-tight constraint. Similarly, a non tight constraint of dual gives a variable with value 0 in primal (complimentary slackness).

2.5 Two Theorems Related to Linear Programming

We also give two theorems that we do not have time to prove in class for linear programming:

Theorem (Strong Duality): The optimal solution for the primal LP is the same as the solution for the dual LP. Therefore, we can always use the dual to prove the solution of the primal is optimal.

Theorem (Complimentary Slackness): if $x = (x_1, ..., x_n)$ is a primal optimal solution and $y = (y_1, ..., y_m)$ is a dual solution. If $a_1x_1+...+a_nx_n \geq b$ is the i^{th} constraint and $a_1x_1+...+a_nx_n > b$ (constraint not tight), then $y_i = 0$.

The reason why I thought that I could solve this by staring is that I know the complimentary slackness. I can use it to know that the second constraint is not tight, and thus know the second variable for dual is zero. Therefore, I only have two variables to deal with.

2.6 Formula for Primal and Dual

Given a primal linear program:

- $\min\langle c, x\rangle$
- $Ax \geq b$
- $x \geq 0$

The dual can be given by

- $\max\langle b, y\rangle$
- $A^T y \leq c$
- $y \geq 0$

The two have the following correspondence:

1. Constraint of primal $\rightarrow$ Variables of dual
2. Variables of primal $\rightarrow$ Constraint of dual
3. Feasible solution of the primal gives an upper bound $\rightarrow$ feasible solution of the dual gives a lower bound

The strong duality gives that the two linear programs have the same optimal value.

3 Applications of Linear Programs

3.1 Using LP to solve graph problems

In the first lecture, we talked about using linear programming to solve bipartite matching, but linear program can be used to solve more problems.

For example, we can write shortest path as a linear program.

- variables: for each edge (u, v), $x_{u,v}$ signals if $edge(u, v)$ is in the shortest path ($x_{u,v} = 1$ if it's in the path and 0 if not). Since it's not a linear constraint, we relax it to $0 \leq x_{u,v}\ leq1$
- Constraints: $x_{u,v}$ form a path from s to t.
  - For each intermediate node, the number of incoming edges equals the number of outgoing edges: for any intermediate vertex $v \neq s, t$, $\sum_u x_{u,v} = \sum_u x_{v,u}$.
  - For s, $\sum_u x_{s,u} = 1$ and $\sum_u x_{u,s} = 0$
  - For t, $\sum_u x_{t,u} = 0$ and $\sum_u x_{u,t} = 1$
- Objective function: min $\sum_{(u,v)} x_{u,v} w_{u,v}$, where w is the weight

A key point in this formulation is how to formulate intermediate nodes. The reason why we did not simply set the number incoming and outgoing edge for a node to be 1 is that we do not know if the node is going to be in the path.

4 Solving Linear Programs

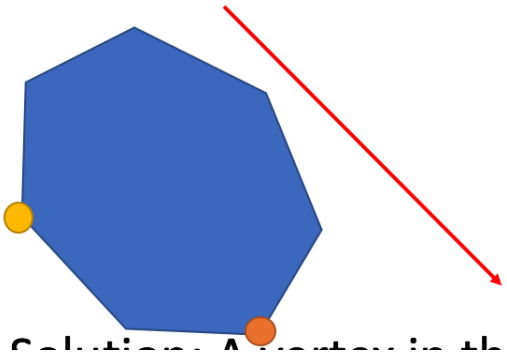
In the rest of the lecture we will briefly talk about how to solve linear programs. There are many existing packages that you may use, and they are highly optimized, so it's recommended to use these packages rather than implementing by yourself.

There are mainly three methods:

- Simplex
- Ellipsoid
- Interior Point

4.1 Simplex Algorithm

Recall the geometric interpretation of the polytope characterizing the constraints.



ible Solution: A vertex in the p

A basic feasible solution is a vertex in the polytope. The reason we want to consider basic feasible solution is that the number of basic feasible solution is finite, and we can interprete it as solution of n tight constraints.

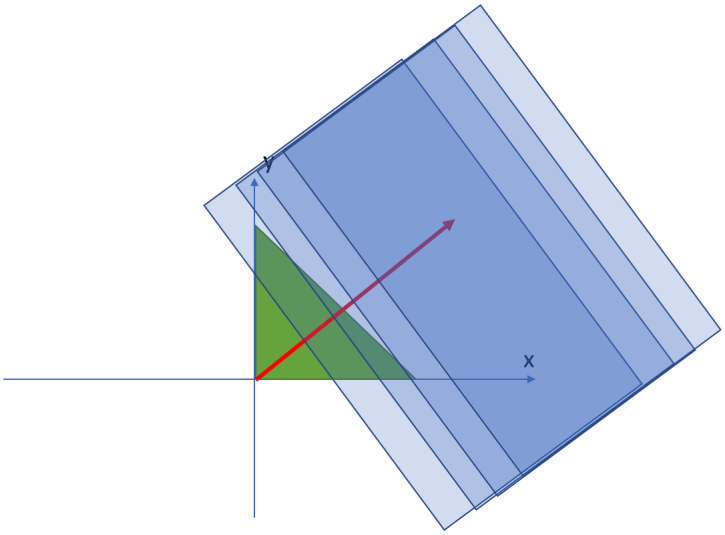
And we claim: There is always an optimal basic feasible solution.

Simplex used this idea. We start from a basic feasible solution, follow an edge in the polytope. Algorithmically, follow an edge is equivalent to swap constraints. There are many ways to determine the process.

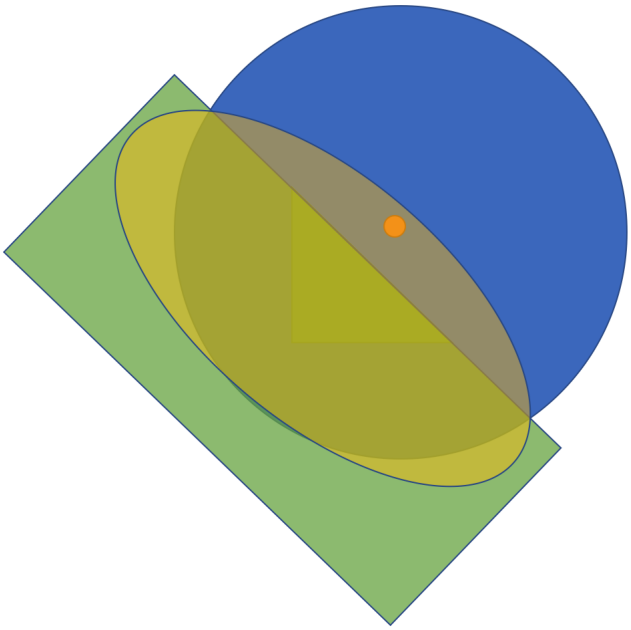
Running Time: Each move takes polynomial time, but how many moves do we need is unknown. In the worst case can require 2^n moves. But it is very fast in practice and is widely used in solving LP.

4.2 Ellipsoid Algorithm

First idea is that we can go from finding a feasible solution to finding optimal solution.



We gradually 'move' the feasible region to touch the optimal solution. The whole process can be illustrated below:



Running Time: The running time is polynomial in the number of variables and constraints, but in practice it is not very fast and often only used in low dimensional problems.