CS261 Winter 2018 - 2019
Lecture 9: Complementary Slackness Theorems; Shortest Path

Instructor: Ashish Goel
Scribe: Kaidi Yan

February 6, 2019

1 Complementary Slackness Theorems

Let P and D denote the primal and dual linear program (in standard form) respectively. The Complementary Slackness[1] Theorems state the following:

Theorem 1. *If* $\mathbf{x}$ *and* $\mathbf{y}$ *are feasible solutions to* P *and* D *respectively and* $\mathbf{x}$, $\mathbf{y}$ *satisfy complementary slackness conditions, then* $\mathbf{x}$ *and* $\mathbf{y}$ *are optimum.*

Theorem 2. *If* $\mathbf{x}$ *and* $\mathbf{y}$ *are optimal solutions to* P *and* D *respectively, then* $\mathbf{x}$ *and* $\mathbf{y}$ *satisfy complementary slackness conditions.*

Here we present the proof to Theorem 2[2].

Proof. Since $\mathbf{x}$ and $\mathbf{y}$ are optimal solutions to P and D, we know they are also feasible solutions to P and D. Hence we have:

$$\mathbf{y}^T(\mathbf{b} - A\mathbf{x}) \geq 0 \tag{1}$$
$$\mathbf{x}^T(A^T\mathbf{y} - \mathbf{c}) \geq 0 \tag{2}$$

Adding these two inequalities together and rearranging terms, we get:

$$\mathbf{y}^T(\mathbf{b} - A\mathbf{x}) + \mathbf{x}^T(A^T\mathbf{y} - \mathbf{c}) \geq 0 \tag{3}$$
$$(\mathbf{y}^T\mathbf{b} - \mathbf{x}^T\mathbf{c}) - \mathbf{y}^TA\mathbf{x} + \mathbf{x}^TA^T\mathbf{y} \geq 0 \tag{4}$$

Now, notice that $\mathbf{y}^T\mathbf{b} = \mathbf{x}^T\mathbf{c}$ due to strong duality; also $\mathbf{y}^TA\mathbf{x} = \mathbf{x}^TA^T\mathbf{y}$ since we have $(\mathbf{y}^TA\mathbf{x})^T = \mathbf{x}^TA^T\mathbf{y}$ and both are scalar values. Hence inequality (4) must hold with equality. This implies that both inequality (1) and (2) must also hold with equality, which is exactly how complementary slackness conditions are defined in terms of matrix multiplication. □

[1] For definition of complementary slackness, please refer to Tim Roughgarden's Lecture #9 notes
[2] You can read the proof to Theorem 1 in Tim Roughgarden's Lecture #9 notes also

2 Complementary Slackness in Max-Flow/Min-Cut

Given a directed graph $G = (V, E)$ and let $\mathcal{P}$ be the set of all s-t paths on G. Recall that the linear program for a max-flow problem on G is:

$$\max \sum_{P \in \mathcal{P}} f_P$$

subject to

$$\sum_{P \in \mathcal{P}: e \in P} f_P \leq u_e \text{ for all } e \in E$$
$$f_P \geq 0 \text{ for all } P \in \mathcal{P}$$

Its dual program, which we show in previous lectures corresponding to a min-cut problem, is:

$$\min \sum_{e \in E} u_e l_e$$

subject to

$$\sum_{e \in P} l_e \geq 1 \text{ for all } P \in \mathcal{P}$$
$$l_e \geq 0 \text{ for all } e \in E$$

Let's see what complementary slackness conditions imply in this case. Let $\mathbf{f}^*$(indexed by $\mathcal{P}$) and $\mathbf{l}^*$(indexed by E) be optimal solutions to the primal and dual programs. If $f_P^* \neq 0$ for some $P \in \mathcal{P}$, then by complementary slackness conditions we know that $\sum_{e \in P} l_e = 1$ must hold for that P. What does this mean? For every optimal 0-1 $\mathbf{l}^*$, we know that an edge e is cut if and only if $l_e = 1$. Hence, $\sum_{e \in P} l_e = 1$ implies that if any max-flow has a path which carries flow, then exactly one edge in that path is cut in any integral min-cut (induced by the 0-1 optimal dual solution).

On the other hand, if $l_e^* \neq 0$ for some $e \in E$, then complementary slackness conditions tell us that $\sum_{P \in \mathcal{P}: e \in P} f_P = u_e$ for that e. This means that if an edge is cut in any min-cut (integral or fractional), then that edge must be saturated in any max-flow.

3 Shortest Path

In this section, we illustrate how LP duality can lead to algorithmic understanding, using the shortest path problem as an example.

In order to apply a linear programming model, we think of a shortest path problem on $G = (V, E)$ as a min-cost flow problem, where we send one unit of flow from starting vertex s to destination vertex t. Each edge $e \in E$ has infinite capacity and the cost c_e is equal to

the length of e. Each vertex v has demand $d_v = 0$ except for s with $d_s = -1$ and t with $d_t = 1$, since we send out one unit of flow from s and sink one unit of flow into t.

Hence the linear program of such a min-cost flow problem (equivalent to our original shortest path problem) is:

$$\min \sum_{e \in E} f_e c_e$$

subject to

$$\sum_{(w,v) \in E} f_{(w,v)} - \sum_{(v,w) \in E} f_{(v,w)} \geq d_v \text{ for all } v \in V \tag{5}$$

$$f_e \geq 0 \text{ for all } e \in E \tag{6}$$

Notice that constraint (5) should technically be

$$\sum_{(w,v) \in E} f_{(w,v)} - \sum_{(v,w) \in E} f_{(v,w)} = d_v \text{ for all } v \in V$$

We replace $=$ with $\geq$ here to write the LP in dual standard form[3]. We are able to do this since if we sum up (5) for all $v \in V$, then we get $0 \geq 0$, which implies that (5) must hold with equality for all $v \in V$. This is similar to what we did for proving Complementary Slackness Theorem 2.

Let this be the primal linear program. We now turn to the dual linear program. From (5) we define the dual decision variables to be l_v for all $v \in V$. Applying the recipe for primal-dual yields:

$$\max l_t - l_s$$

subject to

$$l_q - l_p \leq c_{(p,q)} \text{ for all } (p,q) \in E \tag{7}$$

$$l_v \geq 0 \text{ for all } v \in V \tag{8}$$

Notice that this dual program only deals with differences between l_v's — It's analogous to a potential field where we can define zero potential anywhere and all it matters is the potential differences. Hence we can safely set $l_s = 0$ and remove (11) in this case, yielding:

$$\max l_t$$

subject to

$$l_q - l_p \leq c_{(p,q)} \text{ for all } (p,q) \in E \tag{9}$$

$$l_s = 0 \tag{10}$$

[3]Here we write the primal in dual standard form and the dual in primal standard form. It's easy to verify that the dual of a LP in dual standard form is exactly a LP in primal standard form

Strong duality ensures that if we solve this dual program, then the optimal solution value must be equal to the primal optimal solution value, which is equal to the shortest path length.

Intuitively, the dual program models a scenario where every edge $e \in E$ is replaced with a spring with stationary length c_e. We fix s at zero level and try to "pull" the graph as far away as possible, with the constraint that we cannot stretch any spring (we are allowed to compress some of the springs). The furthest level t can reach is the length of a shortest path from s to t.

To propose a shortest path algorithm based on the dual program, let's first define the following function:

---

function RELAX(v, w)
  if $l_w > l_v + c_{(v,w)}$ **then**
    $l_w \leftarrow l_v + c_{(v,w)}$
  end if
end function

Essentially, RELAX(v, w) fixes the value of l_w such that constraint (12) is satisfied on edge (v, w). For convenience, we call RELAX(p, q) a "relaxation" of (p, q). To see how we can use relaxation to find the shortest path, let's consider the following example:

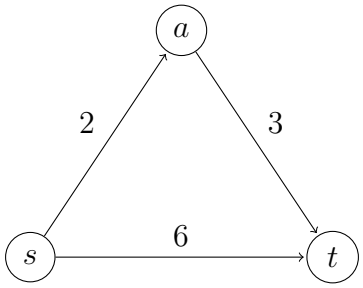

Figure 1: example G. The number next to each edge indicates the edge length.

In this simple graph, suppose we have an initial infeasible solution to the dual program $l_s = 0$, $l_a = 5$ and $l_t = 16$. It's easy to see that constraint (12) is violated for every edge in this graph. Now let's apply relaxations several times:

$$\begin{aligned}
\text{RELAX}(a, t) &\rightarrow l_s = 0, l_a = 5, l_t = 8 \\
\text{RELAX}(s, t) &\rightarrow l_s = 0, l_a = 5, l_t = 6 \\
\text{RELAX}(s, a) &\rightarrow l_s = 0, l_a = 2, l_t = 6 \\
\text{RELAX}(a, t) &\rightarrow l_s = 0, l_a = 2, l_t = 5
\end{aligned}$$

Each time after we relax some edge, there might still be edges in the graph where constraint (12) is violated — so we keep applying relaxations until all edges satisfy constraint (12). In

this case, after applying relaxations four times we arrive at a feasible solution $l_s = 0, l_a = 2, l_t = 5$ and indeed $l_t = 5$ is the optimal solution for the example. Note that we ended up relaxing edge (a, t) twice.

Ford's Algorithm

initalization: $l_s = 0$, $l_v = \infty$ for all $v \in V \setminus \{s\}$
repeat

1. Find an edge (p, q) with $l_q > l_p + c_{(p,q)}$
2. RELAX(p, q)

until: all edges (p, q) satisfy $l_q \leq l_p + c_{(p,q)}$

In fact, this is exactly what *Ford's algorithm* does. Ford's algorithm keeps finding edges (p, q) which violates constraint (12) and performing relaxation on (p, q) to fix the value of l_q. When the algorithm terminates, all edges must satisfy constraint (12) and l_t stores the shortest path length from s to t.

Notice that we didn't specify the way to choose which edge to relax in case there are many edges which violate the constraint. In CS161 we have learnt several shortest-path algorithms such as Bellman-Ford's algorithm and Dijkstra's algorithm. Both Bellman-Ford's algorithm and Dijkstra's algorithm use the idea of relaxation and they are both more specific versions of Ford's algorithm in terms of the way they choose which edge to relax.