CSE 599: Interplay between Convex Optimization and Geometry **Winter 2018**

Lecture 3: Composite Problem via Duality

Lecturer: Yin Tat Lee

Disclaimer: *Please tell me any mistake you noticed.*

Some materials here are taken from [21] and some are new and might be submitted somewhere. If you have any interesting applications, please do not hesitate to contact me.

3.1 Composite Problem via Duality

3.1.1 Motivations

Up to now, duality is still magic to me and I can only explain some of its power through applications. Here, we focus on the algorithmic power of duality. Suppose we have a difficult convex problem $\min_x f(x)$. Often, we can split the difficult problem into $f(x) = g(x) + h(Ax)$ such that $\min_x g(x)$ and $\min_x h(Ax)$ are easy. This form is usually called composite problem. We can compute its dual as follows:

$$\begin{aligned}\min_x g(x) + h(Ax) &= \min_x \max_\theta g(x) + \theta^\top Ax - h^*(\theta)\\ &= \max_\theta \min_x g(x) + (A^\top\theta)^\top x - h^*(\theta)\\ &= \max_\theta -g^*(-A^\top\theta) - h^*(\theta)\end{aligned}$$

where we used the following minimax theorem:

Theorem 3.1.1 (Sion's minimax theorem)**.** *Let $X \subset \mathbb{R}^n$ be a compact convex set and $Y \subset \mathbb{R}^m$ be a convex set. If $f : X \times Y \to \mathbb{R} \cup \{+\infty\}$ such that $f(x, \cdot)$ is upper semi-continuous and quasi-concave on Y for all $x \in X$ and $f(\cdot, y)$ is lower semi-continuous and quasi-convex on X for all $y \in Y$. Then, we have*

$$\min_{x\in X}\sup_{y\in Y} f(x,y) = \sup_{y\in Y}\min_{x\in X} f(x,y).$$

Remark. Compactness is necessary. Consider $f(x, y) = x + y$.

We call "$g(x) + h(Ax)$" the primal problem and "$-g^*(-A^\top\theta) - h^*(\theta)$" the dual problem. Often, the dual problem gives us some insight on the primal problem. However, we note that there are many ways to separate a problem and hence many dual problems.

As an example, consider the flow problem on a graph $G = (V, E)$:

$$\max_{Af=d,-1\leq f\leq 1} c^\top f$$

where $f \in \mathbb{R}^E$ is the flow vector, $A \in \mathbb{R}^{V\times E}$ is the flow conversation constraints, d is the demand vector and c is the cost vector. We can take the dual as follows:

$$\begin{aligned}\max_{Af=d,-1\leq f\leq 1} c^\top f &= \max_{-1\leq f\leq 1}\min_\phi c^\top f - \phi^\top(Af - d)\\ &= \min_\phi \max_{-1\leq f\leq 1} \phi^\top d + (c - A^\top\phi)^\top f\\ &= \min_\phi \phi^\top d + \sum_{e\in E} |c - A^\top\phi|_e\end{aligned}$$

For the case $c = 0$ and $d = F \cdot 1_{st}$ (the maximum flow problem), the dual problem is the minimum $s - t$ cut problem with the cut given by $\{v \in V \text{ such that } \phi(v) \geq t\}$. Note that there are $|E|$ variables in primal and $|V|$ variables in dual. In this sense, the dual problem is easier to solve for dense graph.

Although we do not have a way to turn a minimum $s - t$ cut to a maximum $s - t$ flow in general, we will teach various tricks to reconstruct the primal solution from the dual solution by perturbing the problem.

3.1.2 Example: Semidefinite Programming

Here, we illustrate how cutting plane methods can be used to obtain both primal and dual solutions via a concrete problem - semidefinite programming. Consider the semidefinite programming (SDP) problem:

$$\max_{X \succeq 0} C \bullet X \text{ s.t. } A_i \bullet X = b_i \text{ for } i = 1, 2, \cdots, m \tag{3.1}$$

and its dual

$$\min_{y} b^\top y \text{ s.t. } \sum_{i=1}^{m} y_i A_i \succeq C \tag{3.2}$$

where X, C, A_i are $n \times n$ symmetric matrices and $b, y \in \mathbb{R}^m$. If we apply the current best cutting plane method naively on the primal problem, we would get $O^*(n^2(Z+n^4))$ time algorithm for the primal (because there is n^2 variables) and $O^*(m(Z+m^2))$ for the dual where Z is the total number of non-zeros in A_i, and O^* indicates we are ignoring the log terms in the run time. In general, $n^2 \gg m$ and hence it takes much less time to solve the dual.

We note that

$$\min_{\sum_{i=1}^m y_i A_i \succeq C} b^\top y = \min_{v^\top (\sum_{i=1}^m y_i A_i - C) v \geq 0 \ \forall \|v\|_2 = 1} b^\top y.$$

In each step of the cutting plane method, the gradient oracle either outputs b or outputs one of the cutting planes

$$v^\top (\sum_{i=1}^m y_i A_i - C) v \geq 0.$$

Let K be the set of all cutting planes used in the algorithm. Then, the proof of the cutting plane method shows that

$$\min_{\sum_{i=1}^m y_i A_i \succeq C} b^\top y = \min_{v^\top (\sum_{i=1}^m y_i A_i - C) v \geq 0 \ \forall v \in \mathcal{S}} b^\top y \pm \varepsilon. \tag{3.3}$$

The crux of the proof is to take the dual of the right hand side, then we have that

$$\begin{aligned}
\min_{v^\top (\sum_{i=1}^m y_i A_i - C) v \geq 0 \ \forall v \in \mathcal{S}} b^\top y &= \min_{y} \max_{\lambda_v \geq 0} b^\top y - \sum_{v \in \mathcal{S}} \lambda_v v^\top (\sum_{i=1}^m y_i A_i - C) v \\
&= \max_{\lambda_v \geq 0} \min_{y} C \bullet \sum_{v \in \mathcal{S}} \lambda_v v v^\top + b^\top y - \sum_{i=1}^m y_i \sum_{v \in \mathcal{S}} \lambda_v v v^\top \bullet A_i \\
&= \max_{X = \sum_{v \in \mathcal{S}} \lambda_v v v^\top, \lambda_v \geq 0} \min_{y} C \bullet X + \sum_{i=1}^m y_i (b_i - X \bullet A_i) \\
&= \max_{X = \sum_{v \in \mathcal{S}} \lambda_v v v^\top, \lambda_v \geq 0, X \bullet A_i = b_i} C \bullet X.
\end{aligned}$$

Note that this is exactly the primal SDP problem, except that we restrict the set of solution to the form $\sum_{v \in \mathcal{S}} \lambda_v v v^\top$ with $\lambda_v \geq 0$. Also, we can write the problem as a linear program:

$$\max_{\sum_v \lambda_v v^\top A_i v = b_i \text{ for all } i, \lambda_v \geq 0} \sum_v \lambda_v v^\top C v. \tag{3.4}$$

Therefore, we can simply solve this linear program and recovers an approximate solution for SDP $\sum_{v\in\mathcal{S}}\lambda_v vv^\top$. By (3.3), we know that this is an approximate solution with the same guarantee as the dual SDP.

Now, we analyze the runtime of this algorithm. This algorithm contains two phases: solve the dual SDP via cutting plane method, and solve the primal linear program. Note that each step of the cutting plane method involves finding a separation hyperplane of $\sum_{i=1}^m y_i A_i \succeq C$.

Exercise 3.1.2. Let $\Omega = \{y \in \mathbb{R}^m : \sum_{i=1}^m y_i A_i \succeq C\}$. Show that one can implement the separating oracle in time $O^*(Z + n^\omega)$ via eigenvalue calculations.

Therefore, the first phase takes $O^*(m(Z + n^\omega + m^2))$ time in total. Since the cutting plane method takes $O^*(m)$ steps, we have $|\mathcal{S}| = O^*(m)$. In the second phase, we need to solve a linear program (3.4) with $O^*(m)$ variables with $O(m)$ constraints. It is known how to solve such linear programs in time $O^*(m^{2.5})$ [20]. Hence, the total cost is dominated by the first phase

$$O^*(mZ + mn^\omega + m^3).$$

Problem 3.1.3. In the first phase, each step involves computing an eigenvector of a similar matrix. Is it natural to ask if some matrix update formulas are useful to decrease the cost per step in the cutting plane to $O^*(Z + n^2)$? Namely, can we solve SDP in time

$$O^*(mZ + m^3)?$$

3.1.3 Duality and Convex Hull

The composite problem $\min_x g(x) + h(Ax)$ in general can be solved by the similar trick. To make the geometric picture clear, we consider its "convex programming" version: $\min_{(x,t_1)\in \text{epi} g, (Ax,t_2)\in \text{epi} h} t_1 + t_2$. To simplify the notation, we consider the problem

$$\min_{x\in K_1, Mx\in K_2} c^\top x$$

where $M \in \mathbb{R}^{m\times n}$, convex sets $K_1 \subset \mathbb{R}^n$ and $K_2 \subset \mathbb{R}^m$.

To be concrete, let us consider the following example. Let V_1 be the set of students, V_2 be the set of schools. Each edge $e \in E$ represent a choice of a student. Let w_e be some happiness of school/student if the student is assigned to that school. Suppose that every student can only be assigned to one school and school b can accept c_b many students. Then, the problem can be formulated

$$\min_{x_e\geq 0} \sum_{e\in E} w_e x_e \text{ subjects to } \sum_{(a,b)\in E} x_{(a,b)} \leq 1 \ \forall a \in V_1, \sum_{(a,b)\in E} x_{(a,b)} \leq c_b \ \forall b \in V_2.$$

This is called weighted b-matching problem. Obviously, the number of students is much more than the number of schools. Therefore, an algorithm with running time linear to the number of students is preferable. To apply our framework, we let

$$K_1 = \{x \in \mathbb{R}^E, x_e \geq 0, \sum_{(a,b)\in E} x_{(a,b)} \leq 1 \ \forall a \in V_1\},$$
$$K_2 = \{y \in \mathbb{R}^{V_2}, y_b \leq c_b\},$$

and $M \in \mathbb{R}^E \to \mathbb{R}^{V_2}$ is the map $(Tx)_b = \sum_{(a,b)\in E} x_{(a,b)}$.

To further emphasize its importance, let me give some general examples here:

- Linear programming $\min_{Ax=b,x\geq 0} c^\top x$: $K_1 = \{x \geq 0\}$, $K_2 = \{b\}$ and $M = A$.
- Semidefinite programming $\min_{A_i \bullet X = b_i, X\succeq 0} C \bullet X$: $K_1 = \{X \succeq 0\}$, $K_2 = \{b\}$ and $M : \mathbb{R}^{n\times n} \to \mathbb{R}^m$ defined by $(MX)_i = A_i \bullet X$.

- Matroid intersection $\min_{x\in M_1\cap M_2} 1^\top x$: $K_1 = M_1$ and $K_2 = M_2$, $M = I$.
- Submodular minimization: $K_1 = \{y \in \mathbb{R}^n : \sum_{i\in S} y_i \le f(S) \text{ for all } S \subset [n]\}$, $K_2 = \{y \le 0\}$, $M = I$.
- Submodular flow: $K_1 = \{\varphi \in \mathbb{R}^E, \ell_e \le \varphi_e \le u_e \text{ for all } e \in E\}$, $K_2 = \{y \in \mathbb{R}^V : \sum_{i\in S} y_i \le f(S) \text{ for all } S \subset [n]\}$, M is the incidence matrix.

In all of these examples, it is easy to compute the gradient of $\ell^*_{K_1}$ and $\ell^*_{K_2}$. For the last three examples, it is not clear how to compute gradient of ℓ_{K_1} and/or ℓ_{K_2} directly. Furthermore, in all examples, M maps from a larger space to the same or smaller space. Therefore, it is good to take advantage of the smaller space.

Before our result in [21], the standard way is to use the equivalence of $\nabla\ell^*_{K_1}$ and $\nabla\ell_{K_1}$, and apply cutting plane methods. Worsen by the running time of cutting plane methods at that time, such algorithm usually has theoretical running time at least n^5 with terrible practical performance.

Now, we start rewrite the problem as we do in the beginning:

$$\begin{aligned}\min_{x\in K_1, Mx\in K_2} c^\top x &= \min_{x\in\mathbb{R}^n} c^\top x + \ell_{K_1}(x) + \ell_{K_2}(Mx)\\ &= \min_{x\in\mathbb{R}^n}\max_{\theta\in\mathbb{R}^m} c^\top x + \ell_{K_1}(x) + \theta^\top Mx - \ell^*_{K_2}(\theta)\\ &= \max_{\theta\in\mathbb{R}^m}\min_{x\in\mathbb{R}^n} c^\top x + \ell_{K_1}(x) + \theta^\top Mx - \ell^*_{K_2}(\theta)\\ &= \max_{\theta\in\mathbb{R}^m} -\ell^*_{K_1}(-c - M^\top\theta) - \ell^*_{K_2}(\theta).\end{aligned}$$

Taking dual has two benefits. First, the number of variables is smaller. Second, the gradient oracle is something we can compute efficiently. Hence, cutting plane methods can be used to solve it in $O^*(m\mathcal{T}+m^3)$ where $\mathcal{T}$ is the time to evaluate $\nabla\ell^*_{K_1}$ and $\nabla\ell^*_{K_2}$. The only problem left is to recover the primal x.

The key observation is the following lemma:

Lemma 3.1.4. *Let $x_i \in K_1$ be the set of points outputted by the oracle $\nabla\ell^*_{K_1}$ during the cutting plane method. Define $y_i \in K_2$ similarly. Suppose that the cutting plane method ends with the guarantee that the additive error is less than ε. Then, we have that*

$$\min_{x\in K_1, Tx\in K_2} c^\top x \le \min_{x\in\widetilde{K_1}, Tx\in\widetilde{K_2}} c^\top x \le \min_{x\in K_1, Tx\in K_2} c^\top x + \varepsilon$$

where $\widetilde{K_1} = \mathrm{Conv}(x_i)$ and $\widetilde{K_2} = \mathrm{Conv}(y_i)$.

Proof. Let θ_i be the set of directions queried by the oracle for $\nabla\ell^*_{K_1}$ and φ_i be the directions queried by the oracle for $\nabla\ell^*_{K_2}$. We claim that $x_i \in \nabla\ell^*_{\widetilde{K_1}}(\theta_i)$ and $y_i \in \nabla\ell^*_{\widetilde{K_2}}(\varphi_i)$. Having this, the algorithm cannot distinguish between $\widetilde{K_1}$ and K_1, and between $\widetilde{K_2}$ and K_2. Hence, the algorithm runs exactly the same, namely, the same sequence of points. Therefore, we get the same value $c^\top x$. However, by the guarantee of cutting plane method, we have that

$$\min_{x\in\widetilde{K_2}, Tx\in\widetilde{K_2}} c^\top x \le c^\top x \le \min_{x\in K_1, Tx\in K_2} c^\top x + \varepsilon.$$

To prove the claim, we note that $x_i \in \nabla\ell^*_{\widetilde{K_1}}(\theta_i)$. Note that $\widetilde{K_1} \subset K_1$ and hence $\min_{x\in\widetilde{K_1}} \theta_i^\top x \ge \min_{x\in K_1} \theta_i^\top x$. Also, note that

$$x_i = \arg\min_{x\in K_1} \theta_i^\top x \in \widetilde{K_1}$$

and hence $\min_{x\in\widetilde{K_1}} \theta_i^\top x \le \min_{x\in K_1} \theta_i^\top x$. Therefore, we have that $\min_{x\in\widetilde{K_1}} \theta_i^\top x = \min_{x\in K_1} \theta_i^\top x$. Therefore, $x_i \in \arg\min_{x\in K_1} \theta_i^\top x$. This proves the claim for $\widetilde{K_1}$. The proof for $\widetilde{K_2}$ is the same. □

This reduces the problem into the form $\min_{x\in\widetilde{K_1},Tx\in\widetilde{K_2}} c^\top x$. For the second phase, we let $z_i = Mx_i \in \mathbb{R}^m$. Then, we have

$$\begin{aligned}\min_{x\in\widetilde{K_1},Mx\in\widetilde{K_2}} c^\top x &= \min_{t_i\geq 0,s_i\geq 0,M\sum_i t_i x_i=\sum_i s_i y_i} c^\top(\sum_i t_i x_i)\\ &= \min_{t_i\geq 0,s_i\geq 0,\sum_i t_i z_i=\sum_i s_i y_i} \sum t_i\cdot c^\top x_i.\end{aligned}$$

Note that it takes $O^*(mZ)$ time to write down this linear program where Z is the number of non-zeros in M. Next, we note that this linear program has $O^*(m)$ variables and m constraints. Therefore, we can solve it in $O^*(m^{2.5})$ time.

Therefore, the total running time is

$$O^*(m(\mathcal{T}+m^2)+(mZ+m^{2.5})).$$

To conclude, we have the following theorem:

Theorem 3.1.5. *Given convex sets $K_1\subset\mathbb{R}^n$ and $K_2\subset\mathbb{R}^m$ with $m\leq n$ and a matrix $M:\mathbb{R}^n\to\mathbb{R}^m$ with Z non-zeros. Let $\mathcal{T}$ be the cost to compute $\nabla\ell^*_{K_1}$ and $\nabla\ell^*_{K_2}$. Then, we can solve the problem*

$$\min_{x\in K_1,Mx\in K_2} c^\top x$$

in time

$$O^*(m\mathcal{T}+mZ+m^3).$$

Remark. We hided all sorts of terms in the log term hidden in O^* such as the diameter of the set.

Going back to the school/student problem, this algorithm gives a running time of

$$O^*(|V_2||E|+|V_2|^3)$$

which is linear to the number of students!

In general, this statement says that if we can split a convex problem into two part, both easy to solve and one part has less variables, then we can solve it in time proportionally to the smaller dimension.

Exercise 3.1.6. How about the complexity of the problem $\min_{x\in\cap_{i=1}^k K_i} c^\top x$ given the oracle $\nabla\ell_{K_i}$.

References

[20] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in o(sqrt(rank)) iterations and faster algorithms for maximum flow. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 424–433. IEEE, 2014.

[21] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1049–1065. IEEE, 2015.