

## 18.433 组合最优化

### 原始-对偶算法

October 28

授课教师: Santosh Vempala

在这一讲中, 我们介绍互补松弛性条件并利用它们得到求解线性规划的原始-对偶方法。

### 1 互补松弛性

由前面的强对偶定理我们已经知道, 下面两个线性规划都有可行解时其最优值是相等的, 即  $u = w$ ,

$$u = \max\{c^T x : Ax \leq b, x \geq 0\} \quad (P)$$

$$w = \min\{b^T y : A^T y \geq c, y \geq 0\} \quad (D)$$

利用上面的结论, 我们可以验证原始和/或其对偶问题的最优性。

**定理 1.** 设  $x$  和  $y$  分别是 (P) 和 (D) 的可行解, 那么  $x$  和  $y$  是最优解当且仅当下面的条件成立:

$$\forall i \quad (b_i - \sum_j a_{ij} x_j) y_i = 0;$$

$$\forall j \quad (\sum_i a_{ij} y_i - c_j) x_j = 0.$$

**证明:** 首先, 由于  $x$  和  $y$  是可行解, 故  $(b_i - \sum_j a_{ij} x_j) y_i \geq 0$  且  $(\sum_i a_{ij} y_i - c_j) x_j \geq 0$ . 对下标  $i$  和  $j$  做加和, 可得

$$\sum_i (b_i - \sum_j a_{ij} x_j) y_i \geq 0 \quad (1)$$

$$\sum_j (\sum_i a_{ij} y_i - c_j) x_j \geq 0 \quad (2)$$

把上面两式相加并利用强对偶定理, 可得,

$$\sum_i b_i y_i - \sum_{i,j} a_{ij} x_j y_i + \sum_{j,i} a_{ij} y_i x_j - \sum_j c_j x_j = \sum_i b_i y_i - \sum_j c_j x_j = 0.$$

因此, 不等式 (1) 和 (2) 一定为等式。故结论得证。  $\square$

### 2 原始-对偶算法

定理 1 主要蕴含的结论是: 如果  $x$  和  $y$  是可行解且满足互补松弛性条件, 则他们是最优解。这个结论产生了原始-对偶算法: 从某个可行解  $x$  和  $y$  出发, 使之越来越满足互补松弛性

条件。

方便起见，我们考虑如下原始和对偶规划：

$$\min\{c^T x : Ax = b, x \geq 0\} \quad (P)$$

$$\max\{b^T y : A^T y \leq c\} \quad (D)$$

在这种形式下，互补松弛性条件可简化为：

$$\forall j \quad (c_j - \sum_i a_{ij} y_i) x_j = 0. \quad (3)$$

原始-对偶算法步骤如下：

- 1、从(D)的一个可行解  $y$  开始。在多数情况下得到这样的一个可行解  $y$  要比求解线性规划简单得多。

$$\text{令 } J = \{j : \sum_i a_{ij} y_i = c_j\}.$$

现在我们需要利用(3)得到(P)的一个可行解  $x$  满足  $\forall j \notin J, x_j = 0$ . 问题是有没有一个满足这种性质的可行解  $x$ 。

- 2、写出限定原始规划(RP)如下：

$$\begin{aligned} \min \quad & \sum_{i=1}^m X_i \\ \forall i \quad & \sum_{j \in J} a_{ij} x_j + X_i = b_i \\ & X_i, x_j \geq 0 \\ & \forall j \notin J, x_j = 0 \end{aligned}$$

事实上，(RP)的可行解即满足上述提到的性质(3)。这里，变量  $X_i$ 's 为人工变量。

如果  $\min \sum_{i=1}^m X_i$  为 0，那么  $x_j$ 's 即为(P)的最优解。

- 3、如果  $Opt(RP) = 0$ ，那么  $x$  和  $y$  是最优的。否则  $Opt(RP) > 0$ ，这时我们写出(RP)的对偶形式，称为(DRP)，并求其解  $\bar{y}$ 。

$$\begin{aligned} \max \quad & \sum_{i=1}^m b_i y_i \\ \forall j \in J \quad & \sum_i a_{ij} y_i \leq 0 \\ & y_i \leq 1 \end{aligned}$$

- 4、令  $y' = y + \epsilon \bar{y}$ . 来改进 (D) 的解，其中  $\epsilon$  的取值需满足  $y'$  是可行的，而且

$$\sum_i b_i y'_i > \sum_i b_i y_i. \quad \text{由可行性可知, } \forall j \quad \sum_i a_{ij} y'_i \leq c_j. \quad \text{对 } j \in J, \text{ 有}$$

$$\sum_i a_{ij} y_i + \epsilon \sum_i a_{ij} \bar{y}_i \leq c_j. \quad \text{又因为任意 } j \in J, \text{ 均有 } \sum_i a_{ij} \bar{y}_i \leq 0,$$

所以当  $j \in J$  时  $\epsilon$  可取任意正数。故取

$$\epsilon = \min_{\{j \notin J \text{ s.t. } \sum_i a_{ij} \bar{y}_i > 0\}} \frac{c_j - \sum_i a_{ij} y_i}{\sum_i a_{ij} \bar{y}_i}$$

则满足  $\epsilon > 0$  且  $y'$  是可行的。

又因为  $Opt(DRP) = Opt(RP) > 0$  且  $\epsilon > 0$ ,

$$\sum_i b_i y'_i = \sum_i b_i y_i + \epsilon \sum_i b_i \bar{y}_i > \sum_i b_i y_i.$$

注意，在上面的原始一对偶算法中，求解(DRP)通常要比求解(P)或(D)简单。实际上，在这种方法中，(P)和(RP)都是临时规划，我们真正想解的是(D)。为此，我们先解出(DRP)再用这个解来反复改进  $y$ 。

## 2.1 实例

考虑下面形式的最大流问题：

$$\begin{aligned} & \max f \\ & \sum_j x_{sj} - \sum_j x_{js} - f \leq 0 \\ & f - \sum_j x_{jt} + \sum_j x_{tj} \leq 0 \\ & \forall i \neq s, t \quad \sum_j x_{ij} - \sum_j x_{ji} \leq 0 \\ & x_{ij} \leq u_{ij} \\ & -x_{ij} \leq 0 \end{aligned}$$

值得一提的是，在初始的最大流问题中，前三组约束为等式。但是我们将这三组不等式相加，得到  $0 \leq 0$ ，这些不等式的弱集蕴涵着等式。我们把上述表示形式作为(D)，取  $x$  为零向量即可得到它的一个可行解。现在我们直接给出(DRP)：

$$\begin{aligned}
& \max f \\
& \sum_j x_{sj} - \sum_j x_{js} - f \leq 0 \\
& f - \sum_j x_{jt} + \sum_j x_{tj} \leq 0 \\
& \forall i \neq s, t \quad \sum_j x_{ij} - \sum_j x_{ji} \leq 0 \\
& x_{ij} \leq 0 \quad \forall i, j \text{ where } x_{ij} = u_{ij} \text{ in } (D) \\
& -x_{ij} \leq 0 \quad \forall i, j \text{ where } x_{ij} = 0 \text{ in } (D) \\
& x_{ij} \leq 1 \\
& f \leq 1
\end{aligned}$$

可以看出(DRP)有如下释义。寻找一条从  $s$  到  $t$  的路 (流值为 1) 且路上只能经过下列弧: 饱和的后向弧, 零流的前向弧和任意方向的其它弧。换句话说, 我们需要在剩余图中找一条路。这种观察说明最大流算法实际上是一个原始-对偶算法。

最后, 需要注意, 原始-对偶算法不一定具有多项式执行时间。