

CS 49/149: 21st Century Algorithms (Fall 2018): Lecture 3

Date: 20th September, 2018

Topic: MWU Application: Linear Programming


Scribe: Rui Liu

Disclaimer: These notes have not gone through scrutiny and in all probability contain errors. Please email errors to rliu@cs.dartmouth.edu.

1 The Linear Programming

Linear programming is a technique for the optimization of a linear objective function, subject to linear inequality constraints, as the following equations.

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^m c_i x_i \\ & \text{subject to } m \text{ constraints} && a_i^T x \geq b_i, i = 1 \dots m \\ & \text{where} && x \in \mathbb{R}^n \text{ and } c \in \mathbb{R}^n \end{aligned} \tag{1}$$

The goal is to minimize the linear combination of x in the feasible region, subject to constraints given by inequalities. The feasible region is $\{x \in \mathbb{R}^n : \text{satisfy all constraints}\}$. 

Question: What is the largest total of corners in a n dimensional space?

Answer: In the n dimensional space, the largest total number of corners is $\binom{n}{m}$, because m non-parallel hyper-plane are needed to create a corner.

Instead of traversing the complete feasible region for opt , we could approach the linear programming problem by approximation. We will return \hat{x} within $1/\epsilon^2$ time, s.t. the following equations hold for all $i = 1 \dots m$.

$$\begin{aligned} c^T \hat{x} &\leq \text{opt} \\ a_i^T \hat{x} &\geq b_i - \epsilon \end{aligned}$$

Spacial case when $m = 1$. When $m = 1$, $x_j \in [0, 1]$, the problem becomes a variant of *Knapsack problem*. Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. This problem could solved by a greedy algorithm.

$$\begin{aligned} & \text{minimize} && c^T \hat{x} \\ & \text{subject to} && a^T \hat{x} \geq b \end{aligned}$$

GREEDY ALGORITHM FOR KNAPSACK

- Order items by decreasing $\frac{v_j}{c_j}$
- Keep putting $x_j = 1$ until the reaching the constraint b
- Maybe the last item $x_j < 1$

The time complexity of the greedy algorithm is $O(n \log n)$.



Question: *Is the outcome of the greedy algorithm guaranteed to be optimal?*

Answer: Yes, the optimization is guaranteed and it could be proved by *exchange argument*.

1.1 Appy MWU

ORACLE Instead of finding an x satisfying all the constraints, we could find an $x \in [0, 1]$ satisfying only the following equation (by average all the constraints).

$$\begin{aligned} & \text{minimize} && c^T \hat{x} \\ & \text{subject to} && \frac{1}{m} \sum_i^m a_i^T x \geq \frac{1}{m} \sum_i^m b_i \end{aligned}$$

Since the feasible region of our ORACLE is the super set of the feasible region given all the m constraints, we know $c^T \hat{x} \leq \text{opt}$.

With randomization, we modify the constraint to $\sum_i^m P_i a_i^T x \geq \sum_i^m P_i b_i$. Assume ORACLE give a distribution $\{P_i(t)\}$ and it returns $x(t)$ subject to the following constraints.

$$\begin{aligned} & c^T x(t) \leq \text{opt} \\ & \sum_i^m P_i(t) a_i^T x(t) \geq \sum_i^m P_i(t) b_i \end{aligned} \tag{2}$$

We assume $a_i^T x_i - b_i$ is bounded by some value $\rho > 0$. That is $\forall i \in \{1, \dots, m\}, a_i^T x_i - b_i \in [-\rho, \rho]$. We define loss function as the following equation.

$$\ell_i(t) := \frac{a_i^T x(t) - b_i}{\rho} \tag{3}$$

MULTIPLICATIVE WEIGHT UPDATE LINEAR PROGRAMMING

- Maintain weights $w_i()$ for each expert with $w_i(1) = 1$ for all i .
- On days $t = 1, \dots, T$:
 - Feed $P_i(t) \propto w_i(t)$ to ORACLE and get $x(t)$.

– For constraints i that are violated increase $w_i(t)$

$$w_i(t+1) = w_i(t) \cdot (1 - \eta \ell_i(t)) \quad (4)$$

Recall Theorem 1 in the first lecture.

Theorem 1. The expected ($\mathbf{Exp}[\text{loss}]$) of the *Randomized Weighted Majority* is bounded.

$$\mathbf{Exp}[\text{loss}_{\text{alg}}] \leq \mathbf{Exp}[\text{loss}_{\text{opt}}] + \frac{\ln m}{n} + \eta \sum_t^T \sum_i \ell_i^2(t) \quad (5)$$

Theorem 2. Given any $\epsilon > 0$, we could get $a_i^T \hat{x} - b_i \geq -\epsilon$ within time subject to ϵ .

Proof. With 1, we know

$$\frac{1}{\rho} \sum_t^T \sum_i^m P_i(t)(a_i^T x(t) - b_i) \leq_{i^*} \sum_t^T \ell_{i^*}(t) + \frac{\ln m}{\eta} + \eta * \sum_t^T |\ell_{i^*}(t)|$$

As the ORACLE defines, we know

$$\frac{1}{\rho} \sum_t^T \sum_i^m P_i(t)(a_i^T x(t) - b_i) \geq 0$$

With the two equations above, we get

$$\sum_t^T \ell_{i^*}(t) + \frac{\ln m}{\eta} + \eta * \sum_t^T |\ell_{i^*}(t)| \geq 0$$

Move things around and get

$$\sum_t^T (a_{i^*}^T x(t) - b_{i^*}) \geq -\left(\frac{\ln m}{\eta} + \eta * \sum_t^T |\ell_{i^*}(t)|\right)$$

Let $\hat{x} = \frac{1}{T} \sum_t^T x(t)$, we get

$$C^T \hat{x} = \frac{1}{T} \sum_t^T C^T x(t) \leq \text{opt} \quad (6)$$

Because the definition of loss in Equation 3, the term $|\ell_{i^*}| \in [0, 1]$, s.t. we have

$$a_i^T \hat{x} - b_i \geq -\rho \left(\frac{\ln m}{\eta T} + \eta \right)$$

The equation above reaches upper bound when $\eta = \sqrt{\frac{\ln m}{T}}$.

We will get $2\rho \sqrt{\frac{\ln m}{T}} = \epsilon$ by setting $T = \frac{4\rho^2 \ln m}{\epsilon^2}$ and $\eta = \frac{\epsilon}{2\rho}$.

Total time approximated by $O\left(\frac{4\rho^2 n \ln m \ln n}{\epsilon^2}\right) = \tilde{O}\left(\frac{4\rho^2 n}{\epsilon^2}\right)$.

□

1.2 Linear programming example: vertex cover

A **vertex cover** of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex of the set.

We formulate this problem to the following linear programming problem. For a graph $G = (V, E)$, set $x_v = 1$ if placed, otherwise $x_v = 0$.

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} c_v x_v \\ \text{subject to} & x_u + x_v \geq 1 \quad \forall e = (u, v) \\ & x_v \in [0, 1] \quad \forall v \in V \end{array}$$

In this setting, we have $x_u + x + v \in [0, 2]$ so $\rho = 1$. Thus, the total time for MWU is $\tilde{O}(\frac{n}{\epsilon^2})$, where $n = |V|$.