Linear programming

Huy L. Nguyễn

In the course of our studies, we have encountered many optimizations problems: finding the shortest paths, finding the minimum spanning trees, finding the minimum cuts, etc. We can model the decisions we need to make using variables and the constraints of the problems as equations involving these variables. Amazingly enough, the objective and constraints of many problems can be expressed as a system of linear inequalities.

The feasible region (the set of points satisfying the linear inequalities) is called a *polytope*. The region is *convex*: for any two points x and y in it, the line segment connecting x and y lies entirely in the region.

In *linear programming*, the goal is to optimize (maximize or minimize) a linear objective function over the feasible region. The general form of a linear program is

$$\begin{aligned} \min c^T x \\ Ax \geq b \\ x \geq 0 \end{aligned}$$

Here $\geq$ denotes component-wise greater than or equal.

This general form can express all types of linear programs. To maximize rather than minimize an objective, simply negate the coefficients. To include an inequality such as $5x + y \leq 3$, rewrite it as $-5x - y \geq -3$. To include an equality $3x = y + 1$, rewrite it as two inequalities $3x - y - 1 \geq 0$ and $-3x + y + 1 \geq 0$. To include an unconstrained variable x, one can replace x with $x^+ - x^-$ where x^+, x^- are two new non-negative variables.

An important fact about linear programs is that they can be solved in polynomial time and in fact, there are efficient softwares for this task. We will mostly use this fact as a tool to solve our problems but in due time, we will also see a glimpse of the theory behind algorithms for linear programs.

1 Modeling tasks as linear programs

1.1 Assignment Problem

Suppose we have n jobs and n machines. Since the machines are different and jobs have different requirements (memory, disk space, CPU, etc), the running time of different jobs on different machines are different. Let $c_{i,j}$ be the running time of job i on machine j. We would like to assign one job per machine so as to minimize the sum of the processing times across all machines.

Let $x_{i,j}$ be the variable indicating whether we assign job i to machine j. We hope that this variable will be either 0 or 1 but that is not expressible in an LP so we relax it to the constraints

$$0 \leq x_{i,j} \leq 1 \ \forall i, j$$

Since each machine j can process at most 1 job, we have the constraint:

$$\sum_{i=1}^{n} x_{i,j} \leq 1 \ \forall j$$

Each job i is assigned to at most one machine so we have the constraint:

$$\sum_{j=1}^{n} x_{i,j} \leq 1 \ \forall u$$

Finally, we would like to minimize the total processing time:

$$\min \sum_{i,j} c_{i,j} x_{i,j}$$

Interestingly this linear program always has an integral optimal solution i.e. all variables $x_{i,j}$ are either 0 or 1. Thus, solving the LP actually gives the optimal assignment.

1.2 Shortest path

Suppose we have a graph $G = (V, E)$. The edge from i to j has weight $w_{i,j}$. We would like to find the shortest path from vertex s to vertex t.

We use a variable $x_{i,j}$ to indicate whether an edge (i, j) is used in the shortest path. Again we hope that it is either 0 or 1 but we need to relax it to the constraints

$$0 \leq x_{i,j} \leq 1 \ \forall (i,j) \in E$$

Exactly one edge going out of s must be used in the shortest path:

$$\sum_{(s,i) \in E} x_{s,i} = 1$$

Exactly one edge going into t must be used in the shortest path:

$$\sum_{(i,t) \in E} x_{i,t} = 1$$

For all other vertices u, the number of incoming and outgoing edges in the shortest path must be the same:

$$\sum_{(i,u) \in E} x_{i,u} - \sum_{(u,j) \in E} x_{u,j} = 0 \ \forall u \neq s, t$$

Interestingly this LP also has an integral optimal solution. Thus, solving the LP gives the shortest path from s to t.

1.3 ℓ_1 regression

Suppose you are trying to build a model to explain student's performance. The performance of a student in a course is a function of 1) the difficulty of the course, 2) the aptitude of the student, and 3) random noise. Thus, we hypothesize that the student i's grade in course j is modeled by the following equation:

$$G_{i,j} = a_i + d_j + \varepsilon_{i,j}$$

where a_i is the aptitude of student i, d_j is the difficulty of course j and $\varepsilon_{i,j}$ is a noise term.

We would want the model to have as small noise as possible. Thus we have the following optimization problem, which is an LP.

$$\min \sum_{i,j} |\varepsilon_{i,j}| \ :$$
$$G_{i,j} = a_i + d_j + \varepsilon_{i,j} \ \forall i, j$$

Note that we cannot write $|\varepsilon_{i,j}|$ directly in an LP. Instead we have a new variable $z_{i,j}$ to represent the absolute value and include the constraints $-z_{i,j} \leq \varepsilon_{i,j} \leq z_{i,j}$.