week05-02–linprog-algorithms

March 13, 2024

1 George McNinch Math 87 - Spring 2024

2 Week 5

2.1 Course material (Week 5): Overview on the simplex algorithm

3 How does one solve a linear program??

When we first introduced linear programs, we observed that one might hope to solve them just by *brute force*. Let's consider a linear program $\mathcal{L}$ in standard form given by

$$(\mathbf{c} \in \mathbb{R}^{1\times n}, A \in \mathbb{R}^{r\times n}, \mathbf{b} \in \mathbb{R}^r);$$

thus the variable $\mathbf{x} \in \mathbb{R}^n$ satisfies $\mathbf{0} \le \mathbf{x}$ and $A\mathbf{x} \le \mathbf{b}$.

The rows of the matrix A – as well as the condition $\mathbf{x} \ge \mathbf{0}$ – correspond to inequalities involving linear equations in the coefficients of $\mathbf{x}$. One expects to optimize the objective function $\mathbf{cx}$ at intersection points of a collection of these r linear equations.

We pointed out that one would expect to have to check roughly $\binom{r}{n}$ intersection points to solve the linear program by *brute force*, and that this presents an unreasonably large number of required calculations even when r and n don't seem *so* big.

So far, our treatment of linear programs so far has mainly focused on other issues:

- how to reformulate them ("eliminate equality constraints")
- how to describe them via network flow diagrams
- how to understand the relation between a linear program and its dual linear program

so we haven't really considered the question "how are linear programs solved?".

A nice achievement of 20th century mathematics was a formulation of several algorithms for solving linear programs more efficiently than *brute force*. In fact, `python`'s `scipy` library uses one (actually: it can use several) of these algorithms – by default it uses the so-called simplex algorithm.

4 Augmented form

Recall that we've obsered that we can eliminate equality constraints in a linear program, and thus put it in what we've called *standard form* (a notion that so far was mainly interesting for us because of its role in defining the dual linear program).

We now observe that on the other hand we can replace inequality constraints with equality constraints, at the expense of increasing the number of variables we consider. In fact, we've already met this notion – some of these new variables are precisely the *slack variables* we mentioned in our discussion of *complementary slackness*.

Consider the linear program $\mathcal{L}$ from above. Recall that the slack variables are defined by

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_r \end{bmatrix} = \mathbf{b} - A\mathbf{x}.$$

We observe that

$$A\mathbf{x} \le \mathbf{b} \quad \text{if and only if} \quad \mathbf{z} \ge \mathbf{0}.$$

We now consider the "new" variable vector

$$\mathbf{w} = \begin{bmatrix} v \\ \mathbf{x} \\ \mathbf{z} \end{bmatrix} \in \mathbb{R}^{n+r+1}$$

and the $(1+r) \times (1+n+r)$ matrix

$$B = \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0}_{1\times r} \\ \mathbf{0} & A & I_{r\times r} \end{bmatrix}$$

We note that

$$B\mathbf{w} = \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} \\ \mathbf{0} & A & I_r \end{bmatrix} \cdot \begin{bmatrix} v \\ \mathbf{x} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} v - \mathbf{c}\mathbf{x} \\ A\mathbf{x} + \mathbf{z} \end{bmatrix}$$

so that the condition

$$B\mathbf{w} = \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}$$

means that the coefficient v of $\mathbf{w}$ is equal to the value $\mathbf{cx}$ of the objective function of $\mathcal{L}$, and that $\mathbf{z} = \mathbf{b} - A\mathbf{x}$. Moreover, insisting that $\mathbf{w} \ge 0$ amounts to the condition(s) that $v \ge 0$, that $\mathbf{x} \ge 0$ and that $\mathbf{z} \ge 0$ so that $A\mathbf{x} \le \mathbf{b}$.

Thus in trying to solve $\mathcal{L}$ we may – if we wish – replace $\mathcal{L}$ by an *equivalent* linear program with – in general – more variables and only equality constraints. Moreover, we may as well suppose that the objective function is *equal to* one of the variables (v in the above discussion).

5 Example

Consider the linear program `maximize` $\mathbf{cx}$ where

$\mathbf{c} = \begin{bmatrix} 5 & 4 & 3 \end{bmatrix}, \quad \mathbf{x} \ge 0$

$A = \begin{bmatrix} 2 & 3 & 1 \\ 4 & 1 & 2 \\ 3 & 4 & 2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 5 \\ 11 \\ 8 \end{bmatrix}$, and $A\mathbf{x} \le \mathbf{b}$.

Thus we form $\mathbf{w} = \begin{bmatrix} v \\ \mathbf{x} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} v & x_1 & x_2 & x_3 & z_1 & z_2 & z_3 \end{bmatrix}^T$, and

$$B = \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} \\ \mathbf{0} & A & I_r \end{bmatrix} = \begin{bmatrix} 1 & -5 & -4 & -3 & 0 & 0 & 0 \\ 0 & 2 & 3 & 1 & 1 & 0 & 0 \\ 0 & 4 & 1 & 2 & 0 & 1 & 0 \\ 0 & 3 & 4 & 2 & 0 & 0 & 1 \end{bmatrix}$$

and we consider the linear program: `maximize` v for $\mathbf{w} = \begin{bmatrix} v \\ \mathbf{x} \\ \mathbf{z} \end{bmatrix} \geq 0$ subject to $B\mathbf{w} = \begin{bmatrix} 0 & 5 & 11 & 8 \end{bmatrix}^T$.

6 "Pivoting"

We've just seen that we as well consider a linear program in the variable vector $\mathbf{x} \geq \mathbf{0}$ subject to an equality constraint $B\mathbf{x} = \mathbf{b}$ where our goal is to maximize the value of the variable x_1 – note that we can view the coefficients of $\mathbf{x}$ as "variables".

For a given feasible point, we say that the variables that take a value of zero are *non-basic*, and all the other variables are *basic*.

The goal is to perform an operation we'll call a *pivot*.

Rather than explain this pivot operation, I just want to quickly examine it in the context of the preceding example, for some choice of feasible point.

Note that $\mathbf{w}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 5 & 11 & 8 \end{bmatrix}^T$ is a feasible point since $B\mathbf{w}_0 = \begin{bmatrix} 0 & 5 & 11 & 8 \end{bmatrix}^T$.

For this choice of feasible point, the non-basic variables are (v and) x_1, x_2, x_3 and the basic variables are z_1, z_2, z_3.

Now, the idea is to exchange the point $\mathbf{w}_0$ for another feasible point $\mathbf{w}_1$ with properties: - the v-coordinate of $\mathbf{w}_1$ is larger than that of $\mathbf{w}_0$ (i.e. the objecive function increases) - the set of basic variables for $\mathbf{w}_1$ differs from those for $\mathbf{w}_0$ by exactly one variable.

Such an exchange is what is meant by a "pivot".

So roughly speaking, a "pivot" depends on the choice of a variable (i.e. an index).

It turns out (I'm not going to give the details of why!) that one possible $\mathbf{w}_1$ resulting from a pivot in this case is

$$\mathbf{w}_1 = \begin{bmatrix} 25/2 & 5/2 & 0 & 0 & 0 & 1 & 1/2 \end{bmatrix}^T$$

(note that z_1, x_2, x_3 are non-basic, x_1, z_2, z_3 are basic, and $v = 25/2$ increased from $\mathbf{w}_0$).

The important observation (which I'm not at all explaining here!) is that if such a choice $\mathbf{w}_1$ is not possible, then the optimal value of the objective function is already attained at $\mathbf{w}_0$.

So very roughly speaking: if one understands how to perform the pivot operation - and in particular, how to determine whether or not a pivot is possible – then one knows how to iterate and implement the simplex method.

And it turns out that the simplex method gives an effective way of solving a linear program. Indeed, on average the number of steps in the "algorithm"

```
start with a feasible point;
replace by pivoting;
iterate until no pivot is possible
```

is given by a quadratic polynomial in the quantity $\min(n, r)$.