The Simplex Algorithm

The simplex algorithm gets as an input a linear program of the form:

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax \geq b \\
& \quad x \geq 0,
\end{align*}
\]

where \( x \) is a dimension \( n \) vector, \( c \) is a real vector of dimension \( n \), \( A \) is a \( m \times n \) real matrix, and \( b \) is a real vector of dimension \( m \).

The algorithm finds the optimum of the program by visiting vertices of the feasible polytope in a way that improves the objective function.

1 Preparation

The simplex algorithm works by considering linear programs that satisfy two conditions:

- Have equalities \( Ax = b \) rather than inequalities \( Ax \geq b \). Non-negativity constraints \( x \geq 0 \) are as before.
- We know a vertex of their feasible polytope.

We will see how we can transform a general linear program into a linear program that satisfies those conditions.

From inequalities to equalities. Suppose we have an inequality

\[ a_{i1}x_1 + \ldots + a_{in}x_n \geq b_i. \]

We will add an auxiliary variable \( y_i \geq 0 \) to obtain

\[ a_{i1}x_1 + \ldots + a_{in}x_n - y_i = b_i. \]

We call \( y_i \) a slack variable.

Basic solutions. A vertex of the initial linear program (1) is at the intersection of \( n \) constraints, i.e., where \( n \) out of the \( m + n \) constraints (\( m \) constraints \( Ax \geq b \), and \( n \) constraints \( x \geq 0 \)) hold as equalities. In equality/slack form, a vertex corresponds to solution that sets \( n \) out of the \( m + n \) variables to 0. We call a solution which is both feasible, and sets all the variables, except for \( m \), to 0, a basic feasible solution. The \( m \) variables are the basic variables of the solution. The remaining variables are the non-basic variables. We call a solution degenerate if there is a basic variable that is set to 0.
**Obtaining a basic feasible solution.** Note that if \( b \leq 0 \), then the solution \( y_i = -b_i \geq 0 \) for \( i = 1, \ldots, m \), and \( x_i = 0 \) for \( i = 1, \ldots, n \), is a basic feasible solution. In the general case, we can use the following trick:

1. By the pset problem, to solve a minimization problem like we wish to do, it is enough to solve an appropriately constructed feasibility problem, i.e., given constraints \( Ax \geq b, x \geq 0 \), find a feasible \( x \).

   This transformation to a feasibility problem changes the variables, the matrix, the right hand side vector, and their dimensions, from the initial ones in the minimization problem. Yet, for convenience, let us stick to denoting the number of variables by \( n \), the number of constraints by \( m \), the constraints matrix by \( A \), the variables vector by \( x \), and the right hand side vector by \( b \).

2. We can transform the inequalities to equalities as we did before.

3. Now we can add to each equality \( a_{i1}x_1 + \ldots a_{in}x_n - y_i = b_i \) yet another slack variable \( z_i \geq 0 \), so we get an equality of the form

   \[
   a_{i1}x_1 + \ldots a_{in}x_n - y_i \pm z_i = b_i
   \]

   The coefficient of \( z_i \) is the sign of \( b_i \). The intention is that \( z_i = 0 \), but we allow \( z_i \) to be larger than 0 to ensure feasibility: Now the following solution is a basic feasible solution: \( z_i = |b_i| \geq 0 \) for \( i = 1, \ldots, m \), and \( x_i = 0 \) for \( i = 1, \ldots, n \), as well as \( y_i = 0 \) for \( i = 1, \ldots, m \).

   The final program minimizes over \( z_1 + \ldots + z_m \) subject to the aforementioned constraints. The optimum is 0 if and only if the initial program is feasible.

### 2 The Algorithm

To explain the algorithm, we use an example, for which all the preparations were already done.

\[
\begin{align*}
\text{minimize} & \quad -3x_1 - x_2 - 2x_3 \\
\text{subject to} & \quad x_1 + x_2 + 3x_3 + y_1 = 30 \\
& \quad 2x_1 + 2x_2 + 5x_3 + y_2 = 24 \\
& \quad 4x_1 + x_2 + 2x_3 + y_3 = 36 \\
& \quad x_1, x_2, x_3, y_1, y_2, y_3 \geq 0.
\end{align*}
\]

We use the following presentation of the program to distinguish the basic variables and the basic feasible solution we have in mind: Per equation we have one basic variable, which we put in the left hand side of the equation with coefficient 1. The rest of the variables of the equation...
we move to the right hand side. The basic variable does not appear anywhere else in the program. The basic solution is obtained by setting all the variables except for the basic variables to 0. This means that each basic variable is assigned the free coefficient in its equation.

In our example, we represent the basic feasible solution $y_1 = 30, y_2 = 24, y_3 = 36, x_1 = 0, x_2 = 0, x_3 = 0$, as follows:

$$\begin{align*}
\text{minimize} & \quad -3x_1 - x_2 - 2x_3 \\
\text{subject to} & \\
& y_1 = 30 - x_1 - x_2 - 3x_3 \\
& y_2 = 24 - 2x_1 - 2x_2 - 5x_3 \\
& y_3 = 36 - 4x_1 - x_2 - 2x_3 \\
& x_1, x_2, x_3, y_1, y_2, y_3 \geq 0.
\end{align*}$$

2.1 Pivot

The pivot operation moves from one basic feasible solution to a better basic feasible solution. It makes one of the non-basic variables basic, and one of the basic variables non-basic. Here is how:

1. Find a (non-basic) variable that appears in the objective function with a negative coefficient, i.e., increasing it would improve the objective function. If such does not exist, output the current solution.

2. To make the variable basic, we pick one of the equations the variable appears in, move the variable to the left hand side and normalize so the variable’s coefficient is 1. We pick the equation in a way that ensures we get a basic feasible solution:

For $i = 1, 2, \ldots, m$, let $\Delta_i = -b_i / a_{ij}$, where $a_{ij}$ denotes the coefficient of the variable in the $i$’th equation (So $\Delta_i$ is the assignment to the variable assuming the $i$’th equation is chosen). Then we pick $i$ with minimal $\Delta_i$ among the non-negative $\Delta_i$. If there is no non-negative $\Delta_i$, then declare that the linear program is unbounded.

- Negative $\Delta_i$ violate the non-negativity constraints.
- Taking the minimum over non-negative $\Delta_i$ ensures that none of the other basic variables is assigned a negative value.
- If all the $\Delta_i$ are negative, then we can set the value of the variable to be arbitrary large, making the objective function arbitrarily small.

3. By substitution, eliminate all other occurrences of the variable in the program.
Example.  Iteration 1: Pick $x_1$ that appears in the objective function with negative coefficient $-3$. There are three ways to make $x_1$ basic:

1. Substitute $x_1 = 30 - x_2 - 3x_3 - y_1$.

2. Substitute $x_1 = 12 - x_2 - 5x_3/2 - y_2/2$.


This corresponds to $\Delta_1 = 30$, $\Delta_2 = 12$, $\Delta_3 = 9$. We pick the third equation, which has the minimal non-negative $\Delta$. We then substitute $9 - x_2/4 - x_3/2 - y_3/4$ for $x_1$ to get the following program:

$$\begin{align*}
\text{minimize} & \quad -27 - x_2/4 - x_3/2 + 3y_3/4 \\
\text{subject to} & \quad y_1 = 21 - 3x_2/4 - 5x_3/2 + y_3/4 \\
& \quad y_2 = 6 - 3x_2/2 - 4x_3 + y_3/2 \\
& \quad x_1 = 9 - x_2/4 - x_3/2 - y_3/4 \quad x_1, x_2, x_3, y_1, y_2, y_3 \geq 0.
\end{align*}$$

This corresponds to the solution $y_1 = 21$, $y_2 = 6$, $x_1 = 9$, $x_2 = 0$, $x_3 = 0$, $y_3 = 0$.

Note that if we had picked the second equation instead, for instance, the third equation would have become:

$$y_3 = -12 + 3x_2 + 8x_3 + 2y_2$$

which corresponds to the infeasible assignment $y_3 = -12$.

Iteration 2: Pick $x_2$ that appears in the objective function with the negative coefficient $-1/4$. Pick the second equation and obtain:

$$\begin{align*}
\text{minimize} & \quad -28 + x_3/6 + y_2/6 + 2y_3/3 \\
\text{subject to} & \quad y_1 = 18 - x_3/2 + y_2/2 \\
& \quad x_2 = 4 - 8x_3/3 - 2y_2/3 + y_3/3 \\
& \quad x_1 = 8 + x_3/6 + y_2/6 - y_3/3 \quad x_1, x_2, x_3, y_1, y_2, y_3 \geq 0.
\end{align*}$$

Since there is no variable that appears in the objective function with negative coefficient, we end. The solution is $x_1 = 8$, $x_2 = 4$, $x_3 = 0$, $y_1 = 18$, $y_2 = 0$, $y_3 = 0$, which gives an objective $-28$.

### 2.2 Termination

We iterate the pivot operation as described above. Will the algorithm eventually terminate, and after how many iterations?
Observe: the set of basic variables determines uniquely the basic feasible solution.

Hence, either: (i) the algorithm returns to a set of basic variables it visited before, in which case it never terminates; or (ii) it has at most \( \binom{m+n}{m} \) iterations, assuming there are \( m \) constraints on \( n \) variables (\( \binom{m+n}{m} \) is the number of possible sets of basic variables).

One can eliminate the possibility that the algorithm never terminates, by using “Bland’s rule”: always break ties by picking the variable/constraint with the smallest index. We will not elaborate on this here.

2.3 Optimality

When there are no more variables appearing in the objective function with negative constraints, the objective function equals its free coefficient plus a non-negative. So the objective function of the linear program is at least as large as that free coefficient, which is the value of the solution we have.