Scribe: Trevor Mendez

Linear Programming: The Simplex Algorithm and Relaxations

The material covered in this lecture is based on the presentation in Chapter 2 of the text on linear programming by Howard Karloff [Kar91].

1 Simplex Algorithm

The *Simplex Algorithm* is based on the fact that the optimal solution to a feasible LP (linear program) can be found at one of the vertices of the polytope defined by the set of constraints. The algorithm starts from an arbitrary vertex represented by a *basic feasible solution* (bfs), and at each iteration uses a technique called pivoting to search for an adjacent vertex with an improved cost to the solution to move to. If no adjacent vertex has an improved cost, then it can be proved that the current vertex represents the optimal solution. The algorithm must search through a set of potentially exponentially many vertices, and as a result is not polynomial in the worst-case. Even so, it performs very well in practice, and was the algorithm of choice for several decades.

(Throughout this document, all LP's are given by: minimize $c^T x$ subject to $Ax = b$ and $x \geq 0$, unless explicitly defined otherwise.)

1.1 Pivoting

Pivoting is the mechanism used to manipulate the basis corresponding to a vertex v's bfs, to find the basis that corresponds to an adjacent vertex. To perform a pivot step we replace one of the m linearly independent columns of the basis corresponding to v with one of the nonbasic columns from A in such a way that we still have m linearly independent columns. This produces the basis of an adjacent vertex, v'. The Simplex Algorithm uses pivoting to examine the neighbors of v until a neighbor is found that corresponds to a feasible (non-negative) solution which is an improvement over the bfs for v. We illustrate how pivoting is done with an example.

Example 1 Assume that in LP_1,

$$A = \begin{bmatrix} 5 & 2 & -3 & 16 & 4 \\ 2 & 3 & 1 & 3 & 1 \\ 1 & 7 & 6 & -1 & 2 \end{bmatrix}$$

$$b = \begin{bmatrix} 8 \\ 8 \\ 25 \end{bmatrix}$$

and we have determined a basis,

$$B = \{1, 3, 5\}.$$

(B is specified here by identifying the *basic* columns. In other words, $x_2 = x_4 = 0$.) This gives us the following three linearly independent equations, which can be solved to find the corresponding bfs:

$$\begin{aligned} 5x_1 - 3x_2 + 4x_5 &= 8 \\ 2x_1 + x_2 + x_5 &= 8 \\ x_1 + 6x_3 + 2x_5 &= 25 \end{aligned}$$

The solution, $x_1 = 1$, $x_3 = 3$, $x_5 = 3$; can also be written as:

$$bfs = \begin{bmatrix} 1 \\ 0 \\ 3 \\ 0 \\ 3 \end{bmatrix}.$$

We now perform a pivot to find another basis, by adding a nonbasic column to B, and removing one of the basic columns from B. Assume that we choose to add column A_4. We can express A_4 as a linear combination of the columns that are in B as follows:

$$\begin{aligned} A_4 &= \alpha A_1 + \beta A_3 + \gamma A_5 \\ 16 &= 5\alpha - 3\beta + 4\gamma \\ 3 &= 2\alpha + \beta + \gamma \\ -1 &= \alpha + 6\beta + 2\gamma \end{aligned}$$

The solution is: $\alpha = 1$, $\beta = -1$, $\gamma = 2$. Therefore, $A_4 = A_1 - A_3 + 2A_5$. Since we want to find a new basis, B', with a feasible solution, we must find a basic column to remove, such that:

- $A_{B'} x_{B'} = b$,
- $x_{B'} \geq 0$,
- and $x_2 = 0$.

where $A_{B'}$ refers to the basic columns of A. We have already solved

$$A_1 x_1 + A_3 x_3 + A_5 x_5 = b$$

and want to modify that solution to include A_4 in the basis so that:

$$\begin{aligned} A_1 x'_1 + A_3 x'_3 + A_4 \Theta + A_5 x'_5 &= b \\ x'_1, x'_3, x'_5 &\geq 0 \end{aligned}$$

Since we know that $A_4 = A_1 - A_3 + 2A_5$, we simply need to find $\Theta \geq 0$ such that,

$$A_1(x'_1 + \Theta) + A_3(x'_3 - \Theta) + A_5(x'_5 + 2\Theta) = b.$$

But we know that:

$$A_1 x_1 + A_3 x_3 + A_5 x_5 = b$$

so we have:

$$\begin{array}{rcrcl} x_1 &=& x'_1 + \Theta = & 1, & \Rightarrow \quad \Theta \leq 1 \\ x_3 &=& x'_3 - \Theta = & 3, & \Rightarrow \quad \Theta \geq -3 \\ x_5 &=& x'_5 + 2\Theta = & 3, & \Rightarrow \quad \Theta \leq \frac{3}{2} \end{array}$$

We simply choose the smallest non-negative Θ that satisfies these conditions, namely $\Theta = 1 \Rightarrow x_1 = 0$. Thus our new basis is $B' = \{4, 3, 5\}$, and our bfs has $x'_4 = 1, x'_3 = 4, x'_5 = 1$, and we have completed a pivot. ■

Let us review the steps involved in pivoting. The idea is to start from a basis, $B = \{1, 2, \ldots, m\}$ and then find a suitable nonbasic column $j \in \{1, 2, \ldots, m\}$ to replace a basic column $l \notin \{1, 2, \ldots, m\}$ in order to arrive at an adjacent basis, B'. This is achieved using the following steps:

1. Assume that the starting basis $B = \{1, 2, \ldots, m\}$, is known.

   B contains m linearly independent columns of the n columns in A. We have already found the bfs for B, bfs_B which satisfies:

$$\sum_{i=1}^{m} A_i x_i = b \tag{1}$$

$$x \geq 0 \tag{2}$$

2. Choose a nonbasic column $j \notin \{1, 2, \ldots, m\}$ to add to B.

   (In the Simplex Algorithm, each $j \notin \{1, 2, \ldots, m\}$ can be tried in turn until one is found that results in a bfs with an *improved* cost.)

3. Find the linear dependence of column j on the basic columns, by finding $\alpha_{i,j}$ such that:

$$A_j = \sum_{i=1}^{m} \alpha_{i,j} A_i. \tag{3}$$

4. Find our options for $x'_j = \Theta$, so that

$$\sum_{i=1}^{m} A_i x'_i + A_j x'_j = b \tag{4}$$

   We can find the feasible range of values for Θ as follows:

$$\begin{array}{lcll} \sum_{i=1}^{m} A_i x_i + (A_j x'_j - A_j x'_j) &=& b & \text{(based on Equation 1)} \\ \sum_{i=1}^{m} A_i x_i + A_j x'_j - \Theta \sum_{i=1}^{m} \alpha_{i,j} A_i && & \\ &=& b & \text{(based on Equation 3)} \\ \sum_{i=1}^{m} A_i (x_i - \Theta \alpha_{i,j}) + A_j x'_j &=& b & \end{array}$$

   So $x'_i = x_i - \Theta \alpha_{i,j}$, and in order for $x'_i \geq 0$, we must have $\Theta \leq \frac{x_i}{\alpha_{i,j}}$.

5. Choose the basic column to $l \in \{1, 2, \ldots, m\}$ to remove from B.

   We want to choose the smallest value of Θ that will result in a feasible solution, so we take Θ to be the

   $$\underbrace{\min}_{\forall \alpha_{i,j} > 0} \frac{x_i}{\alpha_{i,j}}.$$

   l is the value of i associated with our choice of Θ.

   (N.B. If $\forall \alpha_{i,j} \leq 0$, then the solution is unbounded.)

1.2 Tableau Method

The *Tableau Method* provides an efficient means of performing the pivots required for the Simplex Algorithm. The idea is to manipulate A and b so that the basic columns of A become the identity matrix. When this happens, Step 3 of the pivoting process (above)—finding the $\alpha_{i,j}$ coefficients for A_j—becomes trivial, because (based on Equation 3) $\alpha_{i,j} = a_{i,j}$ (where $a_{i,j}$ is the (i, j) entry of A—the member in the *ith* row of A_j).

Example 2 Consider LP_2 defined by

$$\begin{aligned} 6 &= x_1 + x_2 + 4x_6 \\ 14 &= -2x_1 + 2x_2 + x_3 - x_4 + x_6 \\ -11 &= x_1 - 2x_2 + x_4 + 2x_6 \\ 7 &= x_1 - 3x_4 + x_5 - 5x_6 \end{aligned}$$

for which we have determined that a basis, $B = \{2, 3, 4, 5\}$. We can represent LP_2 by an $m \times (n+1)$ matrix, a *tableau*, as follows:

b	A_1	A_2	A_3	A_4	A_5	A_6
6	1	1	0	0	0	4
14	-2	2	1	-1	0	1
-11	1	-2	0	1	0	2
7	1	0	0	-3	1	-5

By performing row operations on the tableau, we can transfrom the columns that represent B into the identity matrix, without affecting the solution space. This transformation yields,

b	A_1	A_2	A_3	A_4	A_5	A_6
6	1	1	0	0	0	4
3	-1	0	1	0	0	3
1	3	0	0	1	0	10
10	10	0	0	0	1	25

There are two possible nonbasic columns that we may consider adding to our basis as part of a pivot. Assume that we choose column 1, so that $j = 1$. Then we have,

$$A_j = A_1 \quad = \quad 1A_2 - 1A_3 + 3A_4 + 10A_5$$

Θ is determined by

$$\Theta = \underbrace{\min}_{\forall \alpha_{i,1}>0} \frac{x_i}{\alpha_{i,j}} = \frac{1}{3}.$$

(since $\alpha_{i,j} = a_{i,j}$), so that $l = 4$, and the new basis, B', is $\{2,3,1,5\}$.

Row operations are once again performed in order to convert B' to the identity matrix (and update b to the new bfs b'), yielding

b'	A'_1	A'_2	A'_3	A'_4	A'_5	A'_6
$\frac{17}{3}$	0	1	0	$\frac{-1}{3}$	0	$\frac{2}{3}$
$\frac{10}{3}$	0	0	1	$\frac{1}{3}$	0	$\frac{19}{3}$
$\frac{1}{3}$	1	0	0	$\frac{1}{3}$	0	$\frac{10}{3}$
$\frac{20}{3}$	0	0	0	$\frac{-10}{3}$	1	$\frac{-25}{3}$

The cost function, $c^T x$, is what actually determines which of the nonbasic columns gets chosen to include in the new basis—we must find one which results in decrease in the cost. For any proposed replacement of some column l by some column j, we must simply check that the cost of the new solution is less that the cost of the previous solution. The cost of new solution is given by:

$$\begin{aligned}
\text{new cost} &= \sum_{i=1}^{m} x'_i c_i + x'_j c_j \\
&= \sum_{i=1}^{m} (x_i - \Theta\alpha_{i,j}) + \Theta c_j \\
&= \underbrace{\sum_{i=1}^{m} x_i c_i}_{\text{old cost}} + \Theta \underbrace{[c_j - \sum_{i=1}^{m} \alpha_{i,j} c_i]}_{\text{modified cost, } \bar{c}_j}
\end{aligned}$$

So if the modified cost, $\bar{c}_j$, is negative,, then the new cost will be reduced. ■

Let us review how pivoting is performed using the Tableau Method. A tableau, constructed as follows, is used to perform pivots:

$*$	$\bar{c}_1$	$\bar{c}_2$	$\dots$	$\bar{c}_n$
b_1	$A_{1,1}$	$A_{1,2}$	$\dots$	$A_{1,n}$
b_2	$A_{2,1}$	$A_{2,2}$	$\dots$	$A_{2,n}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
b_m	$A_{m,1}$	$A_{m,2}$	$\dots$	$A_{m,n}$

in which the basic columns of A form the identity matrix, and the each nonbasic column, j, contains the coefficients, $\alpha_{i,j}$. (An additional row, row 0, is used to store the modified costs.)

A pivot is made with the following steps:

1. Find a column j such that $\bar{c}_j < 0$, by examining row 0.

   (If no such column exists, we have found the optimal solution.)

2. Find column l with $\alpha_{l,j} > 0$, that minimizes Θ

   (If all $\alpha_{i,j} \leq 0$ then either the solution is unbounded–unless we didn't have a basis to begin with.)

3. Replace l by j

We have yet to specify how the starting basis is found an LP. Suppose we are given LP_A: minimize $c^T x$ subject to $Ax \leq b$, and $x \geq 0$. Then we can add one *surplus* variable per constraint, such that:

$$\sum_j a_{i,j} x_j + s_i = b_i, \quad i \in \{1, 2, \ldots, m\}$$

Then our bfs is: $\{x_j = 0, s_i = b_i\}$. If instead of $Ax \leq b$ we have $Ax = b$, then we can similarly add *artificial* variables, such that:

$$\sum_j a_{i,j} x_j + y_i = b_i i \in \{1, 2, \ldots, m\}$$

to form LP_B (for which we can easily construct a basis) for which we minimize $\sum y_i$, for $y_i \geq 0$. LP_A will be feasible iff the optimal solution for LP_B is 0. To get a bfs for LP_A we solve LP_B using the Simplex Algorithm by starting with the bfs, $\{x_j = 0, y_i = b_i\}$, and if the optimal solution for LP_B is 0, then all $y_i's$ are 0 and the x_i' form a bfs for LP_A.

2 LP Relaxations

Integer optimization problems can often be approximated using a technique known as LP Relaxation. Given an Integer Program (IP), relax the integers-only conditions and find an optimal solution to the resulting Linear Program. Then use "rounding" to construct an integers-only solution. In subsequent lectures we shall examine the application of LP Relaxation-techniques to approximate several NPC problems. We demonstrate this approach by showing how it can be used to solve the Weighted Set Cover problem.

3 LP Relaxation Approach to Weighted Set Cover

Definition 1 (Weighted Set Cover)
INSTANCE: *A finite set* $\mathcal{U} = \{1, 2, \ldots, n\}$*; a collection of subsets of* $\mathcal{U}$, $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$*; a weight,* w_i *for each* $S_i \in \mathcal{F}$.

PROBLEM: *To find a subset of* $\mathcal{F}$, C, *such that every element in* $\mathcal{U}$ *is covered by at least one set in* C, *and for which the total weight,* $\sum_{S_i \in C} w_i$, *is minimized.*

IP for Weighted Set Cover

$$x_i = \begin{cases} 1 & \text{if } S_i \in C \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} w_i x_i \\ \text{subject to} & \sum_{S_i : j \in S_i} x_i \geq 1, \forall j \in \mathcal{U} \\ & x_i \in 0, 1 \end{array}$$

Let W be the optimal solution to the IP.

LP Relaxation for Weighted Set Cover

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} w_i x_i \\ \text{subject to} & \sum_{S_i : j \in S_i} x_i \geq 1, \forall j \in \mathcal{U} \\ & 0 \leq x_i \geq 1, \forall i \end{array}$$

Let $\hat{W}$ be the optimal solution to the LP. Then $\hat{W} \leq W$.

Randomized Rounding of the LP Solution

The optimal solution to the LP, $\hat{W}$, describes the solution by specifying $\hat{x}_i$. We use Randomized Rounding to find a solution for the IP as follows: Select the set S_i with probability $\hat{x}_i$, independent of what is done with any other set.

There are two obvious concerns to using such an approach:

- Feasibility. Will rounding produce a set cover?

  Consider an element $j \in \mathcal{U}$. Suppose $j \in \{S_1, S_2, \ldots, S_k\}$. Then

  $$\Pr[\text{not covering } j] \quad = \quad (1 - \hat{x}_1)(1 - \hat{x}_2) \ldots (1 - \hat{x}_k)$$

  And since the LP had a solution, one of these $\hat{x}_i$'s must be non-zero, so that we have a probability less than 1. Since the geometric mean of $1 - \hat{x}_1$, $1 - \hat{x}_2$, $\ldots$, $1 - \hat{x}_k$ is at most their arithmetic mean, we have:

  $$((1 - \hat{x}_1)(1 - \hat{x}_2) \ldots (1 - \hat{x}_k))^{1/k} \leq (1 - \sum \frac{\hat{x}_i}{k}).$$

  This implies that

  $$\Pr[\text{not covering } j] \quad \leq \quad \left(1 - \sum \frac{\hat{x}_i}{k}\right)^k \leq \frac{1}{e}$$

- Cost. Will rounding blow up the value of the solution?

  The expected cost of the solution is:

  $$E[\text{cost}] \quad = \quad \sum w_i \hat{x}_i \quad = \quad \hat{W} \quad \leq W$$

So, after having solved the LP optimally, we can simply repeat the Randomized Rounding step t times and include any set, S_i that is selected during any of the t trials. We then have:

$$\begin{array}{rcl} \Pr[j \text{ not covered after 1 trial}] & \leq & \frac{1}{e} \\ \Pr[j \text{ not covered after } t \text{ trials}] & \leq & \frac{1}{e^t} \end{array}$$

If we select $t = 2\ln\ n$, then we have:

$$\Pr[j \text{ not covered after } t \text{ trials}] \leq \frac{1}{e^t} = \frac{1}{e^{2\ln\ n}} = \frac{1}{n^2}$$

And since Boole's inequality states that

$$\Pr[\bigcup_i E_i] \leq \sum \Pr[E_i]$$

we have:

$$\Pr[\exists j \text{ not covered}] \leq \frac{1}{n}.$$

If $\hat{W}_i$ is the cost of experiment (trial) i, then we have:

$$\begin{array}{rclcl} E[\hat{W}_i] & \leq & W & & \\ E[\hat{W}] & = & \sum_{i=1}^{t} \hat{W}_i & \leq & tW \\ E[\hat{W}] & \leq & tW & \leq & (2\ln\ n)W \end{array}$$

and with $\Pr \leq 1 - \frac{1}{n}$ we will have a feasible solution with $cost \leq (2\ln\ n)W$.

Definition 2 (Markov's Inequality)
Let X be a non-negative random variable. Then,

$$\begin{array}{rcll} \Pr[X \geq c] & \leq & \frac{E[X]}{c}, & c > 0. \\ E[X] & = & \sum_{\alpha} \alpha \Pr[X = \alpha] & \\ & \geq & \sum_{\alpha \geq c} \alpha \Pr[X = \alpha] & \\ & \geq & \sum_{\alpha \geq c} c \Pr[X = \alpha] & \\ & = & c \Pr[X \geq c] & \end{array}$$

By Markov's Inequality, we have:

$$\Pr[\hat{W} \geq 2c\ \ln\ n\ W] \leq \frac{1}{c}$$

Theorem 1 *For $c = 4$, the Randomized Rounding Algorithm returns a feasible solution with probability $\geq 1 - \frac{1}{n}$; and has cost $\leq 8 \ln W$, with probability $\geq \frac{3}{4}$.*

Note, the algorithm runs in $O(\log\ n)$ time.

References

[Kar91] H. Karloff. *Linear Programming.* Birkhäuser, Boston, MA, 1991.