Recap, and outline of Lecture 14

Previously
- Proved that the simplex method will terminate after a finite number of iterations
  - Using an anti-cycling pivoting rule if degenerate BFSs are encountered
- Proved that, at termination, the simplex method will either return an optimal (basic) solution to the LP, or evidence of problem unboundedness
- Showed how to assess problem feasibility by solving an auxiliary LP with simplex method

Today
- Obtaining a starting BFS and verify full-rank assumption from solution to the auxiliary problem (slides from lec. 13)
- Approaches to implementation of the simplex method: memory and running time

Implementations of the simplex method

- **Computational effort/running time of an algorithm:** number of iterations $\times$ work per iteration
- Things to consider when implementing (i.e., “coding up”) an iterative algorithm, such as the simplex method:
  - What information do we need to have available to start an iteration? (And how much memory will it take to store it?)
  - How much time will it take to perform an iteration (that is, to perform the required computations, including updating the information needed for the next iteration)?
    - One way to answer this is to consider how many “elementary” arithmetic operations we need to execute at each iteration.
  - If computer memory is not a “scarce resource,” can we perhaps store extra information from the current iteration, to reduce the computational effort in the next iteration?

There are several implementations of the simplex method, with differing answers to the above questions!
Preliminary facts

- We will estimate computational effort of a calculation by the number of elementary arithmetic operations involved.
- E.g., If \( x, y \in \mathbb{R}^n \), to compute \( x'y \), we need to perform (at most) \( n \) multiplications and \( n - 1 \) additions.
  - Total time — at most \( 2n - 1 = O(n) \).

Definition 1.2

Let \( f \) and \( g \) be functions that map positive numbers to positive numbers. We write \( f(n) = O(g(n)) \) (“order of”) if there exist positive numbers \( n_0 \) and \( c \) such that \( f(n) \leq cg(n) \) for all \( n \geq n_0 \).

- Above, \( f(n) \) = “time needed to take the inner product of two vectors in \( \mathbb{R}^n \)”; \( g(n) = n \); \( c = 2 \); \( n_0 = 1 \).
- If \( P, R \in \mathbb{R}^{n \times n} \), it takes \( O(n^3) \) time to compute \( PR \in \mathbb{R}^{n \times n} \).
  - Each element of \( PR \) computed via an inner product.
- Given \( B \in \mathbb{R}^{m \times m} \), computing \( B^{-1} \) takes \( O(m^3) \) operations.
- Given \( B \in \mathbb{R}^{m \times m} \), \( w \in \mathbb{R}^m \), solving \( Bw = w \), takes \( O(m^3) \) ops.
- Multiple methods: Gaussian elimination, various factorizations.

An iteration of the simplex method: “on paper”

1. Iteration starts with a basis of columns \( A_{B(1)}, \ldots, A_{B(m)} \), and an associated BFS \( x \).
2. Compute the reduced costs \( \bar{c}_j = c_j - c'B^{-1}A_j \) for all nonbasic indices \( j \). If \( \bar{c}_j \geq 0 \) for all \( j \), terminate; \( x \) is an optimal solution. Otherwise, pick some \( j \) with \( \bar{c}_j < 0 \).
3. Compute \( u = B^{-1}A_j \). If \( u \leq 0 \), we have \( \theta^* = \infty \). Terminate the algorithm and declare the problem unbounded.
4. If at least one component of \( u \) is positive, let

   \[
   \theta^* = \min_{i: u_i > 0} \frac{x_B(i)}{u_i}.
   \]

5. Let \( l \) be such that \( \theta^* = x_B(l)/u_l \). Form a new basis by replacing \( A_{B(l)} \) with \( A_j \). If \( y \) is the new basic feasible solution, the values of the new basic variables are \( y_j = \theta^* \) and \( y_B(i) = x_B(i) - \theta^* u_i , \ i \neq l. \)
Naive implementation
No information (other than current BFS) retained/updated from iteration to iteration

► At the beginning of the iteration have indices $B(1), \ldots, B(m)$ and vector $x_B$
► Form $B$ and compute $p' = c'_B B^{-1}$ (by solving system $B'p = c_B$ for $p$) \( (O(m^3) \text{ operations}) \)
► Compute reduced costs $\bar{c}' = c' - p'A$ \( (O(nm) \text{ operations}) \)
  ▶ Depending on the pivoting rule used, may compute them one at a time, or may have to compute them all
► Select $A_j$ to enter the basis and compute $u = B^{-1}A_j$ (by solving $Bu = A_j$) \( (O(m^3) \text{ operations}) \)
► Determine $\theta^*$, find the variable $l$ leaving the basis and update the basic indices and corresponding BFS \( (O(m) \text{ operations}) \)

► Memory requirements: $O(m)$
► Computational effort: $O(m^3 + mn)$
  ▶ Except for special cases with very simple special structure of $B$, e.g., network flow problems

Revised implementation (a.k.a. “revised simplex method”)
$B^{-1}$ retained/updated

► At the beginning of the iteration have basic columns $A_{B(1)}, \ldots, A_{B(m)}$, associated BFS $x$, and $B^{-1}$
► Compute $p' = c'_B B^{-1}$ (by matrix-vector multiplication) \( (O(m^2)) \)
► Compute reduced costs $\bar{c}' = c' - p'A$ \( (O(nm)) \)
► Select $A_j$ to enter the basis and compute $u = B^{-1}A_j$ \( (O(m^2)) \)
► Determine $\theta^*$, find the variable $l$ leaving the basis and update the basic columns and corresponding BFS \( (O(m)) \)
► Compute $\bar{B}^{-1}$, where $\bar{B}$ is the new basis matrix, from $B^{-1}$ and $u$ \( (O(m^2)) \)
  ▶ How? See following slides

► Memory requirements: $O(m^2)$
► Computational effort: $O(m^2 + mn) = O(mn)$ (since $m \leq n$)
Computing $\tilde{\mathbf{B}}^{-1}$ from $\mathbf{B}^{-1}$

Recall:

$\mathbf{B} = \begin{bmatrix} \mathbf{A}_B(1) & \cdots & \mathbf{A}_B(m) \end{bmatrix}$

$\tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{A}_B(1) & \cdots & \mathbf{A}_B(l-1) & \mathbf{A}_j & \mathbf{A}_B(l+1) & \cdots & \mathbf{A}_B(m) \end{bmatrix}$

By definition, $\tilde{\mathbf{B}}^{-1}$ is an $m$-by-$m$ matrix such that $\tilde{\mathbf{B}}^{-1} \cdot \tilde{\mathbf{B}} = \mathbf{I}$

We have

$\mathbf{B}^{-1}\tilde{\mathbf{B}} = \begin{bmatrix} 1 & u_1 \\ \vdots & \vdots \\ u_l & \vdots \\ \vdots & \vdots \\ u_m & 1 \end{bmatrix}$

where $\mathbf{u} = \mathbf{B}^{-1}\mathbf{A}_j$ and $u_l > 0$.

Intuition: $\tilde{\mathbf{B}}$ only differs from $\mathbf{B}$ by 1 column. So, $\tilde{\mathbf{B}}^{-1}$ should be not too different from $\mathbf{B}^{-1}$.

Elementary row operations on a matrix

**Definition 3.4**

Given a matrix (not necessarily square) the operation of adding a constant multiple of one row to the same or to another row of the same matrix is called an **elementary row operation**.

- Time taken by an elem. row op.: $O(\#\text{columns of the matrix})$
- **Goal:** Compute $\tilde{\mathbf{B}}^{-1}$ from $\mathbf{B}^{-1}$ by a sequence of at most $m$ elementary row operations.
- **Observation:** Performing an elementary row operation on $\mathbf{C}$ is equivalent to calculating the product $\mathbf{Q} \cdot \mathbf{C}$, where $\mathbf{Q}$ is a suitable matrix.
- More precisely, to multiply row $j$ of $\mathbf{C}$ by $\beta$ and add it to row $i$, set $\mathbf{Q} = \mathbf{I} + \mathbf{D}_{ij}$, where $\mathbf{D}_{ij}$ is a matrix with $\beta$ in position $(i,j)$, and 0 elsewhere.
Computing $\bar{B}^{-1}$ from $B^{-1}$

$$B^{-1}\bar{B} = \begin{bmatrix} 1 & u_1 \\ \vdots & \vdots \\ u_i & \vdots \\ \vdots & \vdots \\ u_m & 1 \end{bmatrix},$$

- Perform the following sequence of elem. row op’s on $B^{-1}\bar{B}$:
  - For each $i \neq l$, add the $l$th row times $-u_l/u_l$ to the $i$th row.
  - Then divide $l$th row by $u_l$.
  - The result is $I \in \mathbb{R}^{m \times m}$.
- Notice: if $Q_1, \ldots, Q_m$ are the matrices corresponding to the elem. row op’s above, them
  $$Q_m \cdots Q_1 B^{-1} \bar{B} = I,$$
  so $\bar{B}^{-1} = Q_m \cdots Q_1 B^{-1}$
- Applying a sequence of these $m$ elem. row op’s to $B^{-1}$ results in $\bar{B}^{-1}$

Full tableau implementation

- **Idea:** Simplify computations further by also maintaining $B^{-1}b$; $B^{-1}A_j$, $j = 1, \ldots, n$; $\bar{c}$ — that is, all the information you may need during the iteration!
- Information presented as the (full) simplex tableau for a basis $B$:

<table>
<thead>
<tr>
<th>row 0</th>
<th>$-c'_{B}B^{-1}b$</th>
<th>$c' - c'_{B}B^{-1}A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>rows 1 through m</td>
<td>$B^{-1}b$</td>
<td>$B^{-1}A$</td>
</tr>
</tbody>
</table>

- In more detail:

  | row 0 | $-c'_{B}x_B$ | $\bar{c}_1$ | $\cdots$ | $\bar{c}_n$ |
  |-------|---------------|-------------|----------|
  | row 1 | $x_B(1)$ | $\cdots$ | $B^{-1}A_1$ | $\cdots$ | $B^{-1}A_n$ |
  | row m | $x_B(m)$ | $\cdots$ | $\cdots$ | $\cdots$ |

- Note: basic columns of the tableau make up $I$
- This requires quite a bit of storage space, but updating this information for the next iteration is quite efficient using row operations.
Updating the tableau (pivoting)

- Let \([b \mid A] \in \mathbb{R}^{m \times (n+1)}\)
- Let \(\bar{B}\) be the new basis; recall: \(\bar{B}^{-1} = QB^{-1}\), where \(Q\) is the product of \(m\) matrices representing elementary row operations.
- To compute \(\bar{B}^{-1}[b \mid A]\):
  \[
  \bar{B}^{-1}[b \mid A] = QB^{-1}[b \mid A],
  \]
  i.e., need to apply the same \(m\) elem. row op’s to the entire tableau.
- That is, add to all the rows of the tableau a multiple of the “pivot row” \(l\) to set all entries of the pivot column to zero, with the exception of the pivot element, which is set to one.
- To update the zeroth row \([0 \mid c'] - c'_{\bar{B}} \bar{B}^{-1}[b \mid A]\): do a similar elem. row op:
  - Add to the zeroth row a multiple of the pivot row to set the element in the pivot column to 0.

An iteration of the full tableau implementation

1. A typical iteration starts with the tableau associated with a basis matrix \(B\) and the corresponding BFS \(x\).
2. Examine the reduced costs in the 0th row of the tableau. If they are all nonnegative, terminate — the current BFS is optimal; else, choose \(j\) for which \(\bar{c}_j < 0\).
3. Consider the vector \(u = B^{-1}A_j\) (\(j\)th column in the tableau, the pivot column). If \(u \leq 0\), terminate — the problem is unbounded.
4. For each \(i\) for which \(u_i > 0\), compute the ratio \(x_{B(i)}/u_i\). Let \(l\) be the index row that corresponds to the smallest ratio. \(A_{B(l)}\) exits, and \(A_j\) enters, the basis. \((O(m))\)
5. Add to each row of the tableau a constant multiple of the \(l\)th row (the pivot row) so that \(u_l\) (the pivot element) becomes one, and all other entries of the pivot column become zero. \((O(nm))\)
Comparison of naive, revised and full tableau simplex implementations

- Memory: storage space required
- Time: computational effort per iteration

<table>
<thead>
<tr>
<th></th>
<th>Naive</th>
<th>Revised</th>
<th>Full Tableau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>$O(m)$</td>
<td>$O(m^2)$</td>
<td>$O(mn)$</td>
</tr>
<tr>
<td>Worst-case time</td>
<td>$O(m^3 + mn)$</td>
<td>$O(mn)$</td>
<td>$O(mn)$</td>
</tr>
<tr>
<td>Best-case time</td>
<td>$O(m^3)$</td>
<td>$O(m^2)$</td>
<td>$O(mn)$</td>
</tr>
</tbody>
</table>

- Best-case time: if first computed reduced cost is negative