IE 495 Lecture 25

November 30, 2000

Reading for This Lecture

- Primary
  - Bazaraa, Sherali, and Sheti, Chapter 2.
  - Chvatal, Chapters 6 and 7.

Linear Programming

Introduction

- Consider again the system $Ax = b$, $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^{m}$.
- In this problem, there are either
  - no solutions
  - one solution
  - infinitely many solutions (if n > m)
- The problem of *linear programming* is

$$
\begin{array}{ll}
\min & c^{\mathrm{T}}x \\
\text{s.t.} & Ax = b \\
 & x \geq 0
\end{array}
$$

The Simplex Algorithm

- Note that $x_B = B^{-1}b - B^{-1}Nx_N$
- Hence, $c^T x = c_B^T x_B + c_N^T x_N = c_B^T B^{-1} b + (c_N^T - c_B^T B^{-1} N) x_N$
- So if $c_N^T - c_B^T B^{-1} N \geq 0$, we have found the optimal solution (why?).
- Otherwise, suppose some component of $c_N^T - c_B^T B^{-1} N$ is negative.
- Then we raise the value of the corresponding variable as much as possible while maintaining feasibility.

Summary of the Simplex Algorithm

- Simplex algorithm
  - Compute $yB = c_B^{\mathrm{T}}$
  - Choose a column of a_j of N such $ya_j < c_j$
  - Compute $Bd = a_j$
  - Find the largest t such that $x_B^* - td \geq 0$
  - Set the value of x_j to t and the values of the basic variables to $x_B^* - td$.
  - Update the basis.
- The only hard part is implementing the last step.

Implementing the Algorithm

- Let B_k be the basis after the k^{th} iteration.
- Note that $B_k = B_{k-1}E_k$ where
  - E_k is the identity matrix with the p^{th} column replaced by $d = B_{k-1}^{-1}\, a_j$ (already computed).
  - p is the "leaving column"
- So, we have $B_k = B_0E_1 \ldots\ldots E_k = LUE_1 \ldots\ldots E_k$
- To update at each iteration, we merely append the next eta matrix to the list.
- Often, B_0 is the identity matrix.

Refactorizing the Basis

- After many iterations, it can become inefficient to solve these systems.
- Periodically, throw away all the eta files and calculate a brand new LU factorization.
- How often should this be done?
- It depends on the matrix.
- Under some fairly reasonable assumptions, the "break-even" point seems to be $\approx$ 15 iterations.

Another Approach

- Update the LU factorization directly
- We have $B_k = L_k U_k$.
- We also have $B_{k+1} = B_k E_{k+1}$.
- Hence, $B_{k+1} = L_k U_k E_{k+1}$.
- We can permute the rows and columns of $V = U_k E_{k+1}$. such that V differs from an upper-triangular matrix in at most one row.
- It is then easy to perform an LU factorization of V.
- This can easily be made into an LU factorization of B_{k+1}.

Issues to be addressed

- Ensuring numerical accuracy
  - Conditioning
  - Stability
  - Zero tolerances
- Ensuring efficiency
  - Maintaining sparsity
  - Updating basis factorization

Dealing with Large Matrices

- Recall this step from the Simplex Algorithm:
  - Choose a column of a_j of N such $ya_j < c_j$
- This step is called *pricing*.
- One approach is to choose the quantity $c_j - ya_j$ to be as large as possible.
- If the number of columns of A is large, then the pricing step can be cumbersome.
- Partial pricing is the practice of only pricing out a small subset of possible columns.

Column Generation

- Notice that the problem max $\{c_j - ya_j\}$ is an optimization problem.
- Notice also that it is not necessary to have all the columns present in the matrix.
- Suppose the columns of the matrix have a special structure that allows us to generate them "automatically".
- We can sole the above optimization problem to determine the next column to be pivoted in.
- All we really need is the columns of the optimal basis.

Constraint Generation

- Consider an LP specified as follows

$$\begin{array}{ll} \min & c^{\mathrm{T}}x \\ \text{s.t.} & Ax \leq b \end{array}$$

- In this case, we can sometimes have $m >> n$.
- Constraints (rows) can also be automatically generated.
- This is called *separation*.

Deleting Columns and Rows

- If the slack variable for a particular row is basic, then that row is "inactive".
- Inactive rows can be deleted from the problem without changing the optimal solution.
- Similarly, there are methods of proving that a particular column can never be basic in an optimal solution.
- While solving large LP's by column and constraint generation, we can simultaneously purge ineffective rows and columns and generate new ones.
- This technique can be very effective.

Integer Linear Programs

Integer Linear Programs

- Now one more layer of complication. . . .
- Suppose that we have an LP in which some of the variables are constrained to be integer-valued.

$$
\begin{array}{ll}
\text{min} & c^{T}x \\
\text{s.t.} & Ax = b \\
 & x \geq 0 \\
 & x \in \mathbf{Z}^{n}
\end{array}
$$

- The Simplex Algorithm can't handle this.

LP-based Branch and Bound

- Basic Method
  - Formulate and solve the *LP relaxation*.
  - If the optimal solution is integral, STOP.
  - Otherwise, branch on some fractional variable
  - Iterate
- Notice that solving the LP serves a three-fold purpose
  - Generates a lower bound
  - Possibly generates a feasible solution
  - Indicates how to branch

Example: Traveling Salesman Problem