Algorithmic Methods	25/10/2010

Lecture Notes 2: The Simplex Algorithm

Professor: Yossi Azar — *Scribe:Kiril Solovey*

1 Introduction

In this lecture we will present the Simplex algorithm, finish some unresolved linear programming related issues from the previous lecture, and present the Dual concept.

2 Linear Programming

Some basic definitions and theorems we covered in the previous lecture:

- *Polyhedron* - Intersection of half-spaces.
- *Vertex* - Feasible point which cannot be written as a convex combination of two feasible points.
- **Theorem A:** Let p be a polyhedron defined as $p = \{x|Ax \geq b\}$. Then x is a vertex of p iff x is a feasible solution that satisfies tightly n linearly independent constraints.
- **Theorem B:** Let p be a polytope (non-empty bounded polyhedron). Then for every $x \in p$ there exists a vertex x' such that $c^t x' \geq c^t x$.
- *Basic Solution*: The matrix A has m linearly independent columns. The total number of constraints is $m+n$. Denote $j_1 < j_2 < \ldots < j_m$ as the appropriate variable indices. Any other variable is set to 0. Let B be the matrix consist of the columns $j_1, \ldots, j_m$ of A and $x_B = x_{j_1}, \ldots, x_{j_m}$. The basic solution is defined as $x_B = B^{-1}b$. If $x_B \geq 0$ then x_B is a basic feasible solution (bfs).

Example 1:

$$\begin{cases} x_1 + x_2 + x_3 = 8 \\ x_2 + 2x_3 = 6 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

$(2, 6, 0)$ and $(5, 0, 3)$ are both bfs whereas $(0, 10, -2)$ is not feasible.
Example 2:

$$\begin{cases} x_1 + x_2 + x_3 = 6 \\ x_2 + 2x_3 = 6 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

$(0, 6, 0)$ is a mutual bfs for the basis x_1, x_2 and x_2, x_3. The other bfs is $(3, 0, 3)$.
Theorem C: x is a vertex iff x is bfs.
proof. If x is a vertex, then by Theorem A x satisfies tightly n linearly independent constraints. Let j_x be all the non-zero variable indices, and A_x be the induced matrix. Denote

k as the number of non zero variables. Then $n-k$ of the variables are 0 and therefore define $n-k$ tight constraints. $rank(A_x) = k$, otherwise we will not get n linearly independent constrains. Hence $k \leq m$. If $k = m$ then x is bfs for the columns j_x of A and we are done. If $k < m$ then by adding $m-k$ independent columns we form a basis. x satisfies all the constraints defined by the sub-matrix C of A of size $m \times m$, and the rest of the variables in x are set to 0. The matrix is regular and therefore x is the bfs of the column of C.

For the other direction we include an alternative proof to the one given in class. Let x be a bfs of $Ax = b$, $x \geq 0$, and $rank(A) = m$. Suppose that $x = \lambda u + (1-\lambda)v$ where $0 < \lambda < 1$ and $u, v \in p = \{x | Ax = b\}$, in other words, assume that x is not a vertex. Let $J \subseteq \{1, \ldots, n\}$ be the basis corresponding to x, and for $j \notin J$ $x_j = 0$. x is unique vector in p satisfying $x_j = 0$ if $j \notin J$. Therefore, there exists $j \notin J$ such that $u_j > 0$. Also, $v \geq 0$ since this a LPS system. Therefore $\lambda u_j + (1-\lambda)v_j > 0$ in contradiction to the fact that $x_j = 0$.

2.1 Yet, another application of linear programming

Let $\ell_1, \ldots, \ell_m$ be k half-spaces in $\mathbb{R}^n$. Find a maximal radius ball that doesn't intersect with any ℓ_i. ℓ_i are given in the following form,

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \geq b_1 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \geq b_m \end{cases}.$$

We would like that the center of such ball will be as far as it can from each half-space. We get the following linear programming system whose variables are $x_1, \ldots, x_n$ and R:

$$\forall i \quad \frac{a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n - b_i}{\sqrt{a_{i1}^2 + a_{i2}^2 + \ldots + a_{in}^2}} \geq R; \quad \max R.$$

3 The Simplex Algorithm [Danzig47]

In a nutshell, the algorithm traverses the vertices of the polytop, moving from one vertex to its neighbor, by modifying the basis of the bfs of the current vertex.

Example:

$$\begin{pmatrix} 1 & 0 & 2 & -1 & 0 \\ 0 & -1 & 1 & -1 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 7 \\ 3 \\ 6 \end{pmatrix}; \quad x_1, x_2, x_3, x_4, x_5 \geq 0$$

$$\min 2x_1 + x_2 + x_3 + x_4 + 2x_5.$$

Using elementary row operations we get

$$\begin{pmatrix} 1 & 2 & 0 & 1 & 0 \\ 0 & -1 & 1 & -1 & 0 \\ 0 & 2 & 0 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix}.$$

We will use x_1, x_3, x_5 as the basis and set the 2nd and 4th columns to 0. Note that the columns 1,3,5 form the identity matrix (or in general, permutation matrix). The appropriate bfs is $(1, 0, 3, 0, 3)$ with the target function 11. Let's increase the value of x_4 from 0 to ϕ. The values of x_1, x_3, x_5 would change accordingly,

$$\begin{pmatrix} x_1 \\ x_3 \\ x_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix} - \begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix} \cdot x_4.$$

We will get the following change in the target function, $-2\phi + 0 + \phi + \phi - 4\phi = -4\phi$. So, the target function diminishes as x_4 grows bigger, but we must keep in mind the fact that we have to satisfy the constraints $x_1, \ldots, x_5 \geq 0$ as well. It forces the constrain $\phi \leq 1$. If, on the other hand, we set $x_2 = \delta$ then $x_1 = 1 - 2\delta, x_3 = 3 + \delta, x_5 = 3 - 2\delta; \delta \leq 1/2$ and the change in the target function will be -6δ.

Let's extend the above example to general LPS.

3.1 Preliminaries

$$\min c^t x; \quad Ax = b; \quad x \geq 0$$

Let B be the basis, x_B basis variables, x_N non-basis variables, A_B the basis columns matrix (regular matrix), A_N non-basis columns matrix. We get the following (equivalent) linear programming problem,

$$\min(c_B^t x_B + c_N^t x_N); \quad A_B x_B + A_N x_N = b; \quad x_B, x_N \geq 0.$$

Moreover, $x_B = A_B^{-1} b - A_B^{-1} A_N x_N$, and the target function can be written as

$$\min(c_B^t (A_B^{-1} b - A_B^{-1} A_N x_N) + c_N^t x_N) = \min(c_B^t A_B^{-1} b + (c_N^t - c_B^t A_B^{-1} A_N) x_N).$$

If we set $x_N = 0$ then $x_B = A_B^{-1} b$ and the value of the target function will be $c_B^t A_B^{-1} b$.
We define a weight vector $\tilde{c} = c_N^t - c_B^t A_B^{-1} A_N$. If there exists $\tilde{c}_j < 0$ then increasing x_j (as long as $x_B \geq 0$) will result a decrease in the goal function value. In case there is no limit for increasing x_j, the target function is unbounded, i.e $-\infty$, otherwise we stop once some variable in the basis reaches 0. This process is called *pivot step*.
Theorem: If $\tilde{c} > 0$ then the algorithm is optimal.
Proof. For any feasible point y we have

$$c^t y = c_B^t A_B^{-1} b + (c_N^t - c_B^t A_B^{-1} A_N) y_N) = c_B^t A_B^{-1} b \quad + \tilde{c} y_N \geq c_B^t A_B^{-1} b$$

since $\tilde{c} \geq 0$ and $y_N \geq 0$. Hence x is optimal.

3.2 Algorithm

1. Select a basis B that defines a bfs.
2. Calculate the proper matrices (A_B^{-1}, A_N,...).
3. If $\tilde{c} > 0$ then the solution is optimal, STOP.
4. Otherwise, find an index j such that $\tilde{c}_j < 0$ and increase x_j until one of the basis variables is set to 0.
   If there is no such variable the target function is unbounded.
   Otherwise include j in the basis and remove the new zero variable.

Steps 2,3,4 are the *pivot steps*.
A few questions arise:

- How do we select j (among all j, such that $\tilde{c}_j < 0$) and what variable is removed from the basis (among all variables that are set to 0)?
- What happens when we cannot increase x_j at all ?
- Can the algorithm get stuck in a loop?

Lemma: The algorithm converges if the target function decreases.
Proof. The algorithm never returns to the same solution and the number of vertices is finite. If, on the other hand, the target function remains the same, we stay in the same vertex and might return to the same basis.
Definition (Bland Rule):

- Add to the basis a variable with minimal j such that $\tilde{c}_j < 0$.
- Remove from the basis the zero variable with minimal index j.

The following theorem is given without a proof.
Theorem: The algorithm converges if the Bland Rule is used.
Observation: In the worst case, the algorithm traverses all the vertices, which might be exponential (in m, n).
Typically, however, the number of iterations is small and even linear.
Complexity:

1. Initialization requires handling - deciding whether some solution exists is as hard as finding an optimal solution.
2. Gauss Elimination is used in the first iteration - $O(n^3)$, otherwise $O(n^2)$.
3. $O(n)$ (values are computed by the Gauss Elimination).
4. $O(n)$.

3.3 Algorithm Initialization

$\min c^t x, Ax = b \geq 0$ (if there's $b_i < 0$ we multiply the ith row of A by -1), $x \geq 0$. We define a new system

$$\begin{pmatrix} A & I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = b, \quad \min \sum y_i, \quad x \geq 0, y \geq 0.$$

Lemma: The original system is feasible $\iff$ the new system has an optimum value 0.
Proof. Let x be a solution to the original system. Set $y_i = 0$, observe that $(x\, y)$ satisfies the new system as well and the goal function attains it's minimum value. The other direction of the proof is similar.

Hence, we can find an initial solution to the original system by solving the new system using the simplex algorithm. However, we need also to initialize the new system. Notice that $x_i = 0, y_i = b_i \geq 0$ is a bfs of the new system.

4 The Dual

We start with an example. Observe the following linear programming system:

$$\min 2x_1 + 4x_2 - x_3; \quad \begin{array}{c} (1)\, x_1 + x_2 + x_3 \geq 10 \\ (2)\, x_1 + 3x_2 - 2x_3 \geq 4 \\ (3)\, 2x_2 - 3x_3 \geq -8 \end{array}$$

Notice that

$$2 \times (1) + 1 \times (3): \quad 2x_1 + 4x_2 - x_3 \geq 12$$

$$1 \times (1) + 1 \times (2): \quad 2x_1 + 4x_2 - x_3 \geq 14.$$

Thus, we found a lower bound on the goal function. To find to find a stronger lower bound, denote by y_1, y_2, y_3 the factors on the constraints (in the last example $y_1 = 1, y_2 = 1, y_3 = 0$). Now we want to solve the system $y^t A = c$; $\max y^t b$; $y \geq 0$. Each feasible solution of the new system is smaller then any feasible solution in the original system. It's clear that $\min c^t x \geq \max y^t b$.
The constraints $x \geq 0$ in the original system transform to the constraints $y^t A \leq c$ in the dual.
Yet, another example:

$$\min 3x_1 + 2x_2 + 3x_3 = G; \quad \begin{array}{c} x_1 + x_2 - x_3 = 4 \\ -2x_1 - 4x_3 = -7. \\ x_1, x_2, x_3 \geq 0 \end{array}$$

$$1 * (1) - 1 * (2): \quad 3x_1 + x_2 + 3x_3 = 11 \implies G \geq 11$$

$$2 * (1) - \frac{1}{2} * (2): \quad 3x_1 + 2x_2 + 0x_3 = 11.5 \implies G \geq 11.5$$

Again, to get a lower bound we need to solve $y^t A \leq c$; $y \lesseqgtr 0$; $\max y^t b$.