Thursday, Sept 15

1. Welcome!
2. HW 1 due tonight
3. Masking Survey
4. Topics / HW2 up soon
5. Questions?
6. Small work
7. Simplex!

Today we are digging into the simplex method, an algorithm for solving linear programs that are in SEF. We've already worked on one step, and we'll do a few more examples to build intuition and support some of the vocabulary that comes with this method.

Example: In our last example, we increased one of the decision variables and improved our objective function. Try to improve the other variable.

$$\begin{bmatrix}3\\2\\2\end{bmatrix} = t\begin{bmatrix}1\\1\\0\end{bmatrix} + \begin{bmatrix}2\\0\\2\end{bmatrix} + \begin{bmatrix}s_1\\s_2\\s_3\end{bmatrix} \qquad \begin{bmatrix}1&1&1&0&0\\1&0&0&1&0\\0&1&0&0&1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\s_1\\s_2\\s_3\end{bmatrix} = \begin{bmatrix}3\\2\\2\end{bmatrix}$$

$$\begin{bmatrix}1\\2\\0\end{bmatrix} - t\begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}s_1\\s_2\\s_3\end{bmatrix}$$

$\boxed{t=1}$

$$\tfrac12 x_1 + x_2 = \tfrac12(3-s_1) + 1 - \tfrac12 s_3 = \boxed{2.5 - \tfrac12 s_1 - \tfrac12 s_3}$$

$$x_1 + x_2 + s_1 = 3 \qquad x_2 + s_3 = 2$$

$$\tfrac12 x_1 + \tfrac12 x_2 = \tfrac12(3-s_1) \qquad \tfrac12 x_2 = 1 - \tfrac12 s_3$$

feasible: $\begin{bmatrix}1\\2\\0\\1\\0\end{bmatrix}$, obj $= 2.5$

Question: Are we done? How can you tell?

Example: Solve the linear program below by putting it into standard equality form, finding a good starting point, and incrementally improving the objective.

$$\begin{cases} \max & x_1 + 2x_2 \\ \text{s.t.} & x_1 + x_2 \le 5 \\ & x_1 \le 4 \\ & -x_1 + x_2 \le 1 \\ & x_i \ge 0 \end{cases} \rightarrow \begin{bmatrix}1&1&1&0&0\\1&0&0&1&0\\-1&1&0&0&1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\s_1\\s_2\\s_3\end{bmatrix} = \begin{bmatrix}5\\4\\1\end{bmatrix} \rightarrow \text{start: } \begin{bmatrix}0\\0\\5\\4\\1\end{bmatrix}, \ \text{obj} = 0$$

1 increase x_2: $\begin{bmatrix}5\\4\\1\end{bmatrix} = 0\begin{bmatrix}1\\1\\-1\end{bmatrix} + t\begin{bmatrix}1\\0\\1\end{bmatrix} + \begin{bmatrix}s_1\\s_2\\s_3\end{bmatrix} \rightarrow \boxed{t=1}$ new feas: $\begin{bmatrix}0\\1\\4\\4\\0\end{bmatrix}$ new obj $= 2$

2 increase x_1: $\begin{bmatrix}5\\4\\1\end{bmatrix} = t\begin{bmatrix}1\\1\\-1\end{bmatrix} + \begin{bmatrix}1\\0\\1\end{bmatrix} + \begin{bmatrix}s_1\\s_2\\s_3\end{bmatrix} \rightarrow \boxed{t=4}$ new feas: $\begin{bmatrix}4\\1\\0\\0\\4\end{bmatrix}$ new obj. $= 6$

check objective: $x_1 + 2x_2 = 2x_1 + 2x_2 - x_1$
$= (10 - 2s_1) - (4 - s_2)$
$= 6 - 2s_1 + s_2$

$$\begin{bmatrix}5\\4\\1\end{bmatrix} = s_1\begin{bmatrix}1\\0\\0\end{bmatrix} + t\begin{bmatrix}0\\1\\0\end{bmatrix} + x_1\begin{bmatrix}1\\1\\-1\end{bmatrix} + x_2\begin{bmatrix}1\\0\\1\end{bmatrix} + s_3\begin{bmatrix}0\\0\\1\end{bmatrix}$$

$$\begin{bmatrix}5\\4\\1\end{bmatrix} = t\begin{bmatrix}0\\1\\0\end{bmatrix} + \begin{bmatrix}x_1+x_2\\ \\ \end{bmatrix}$$

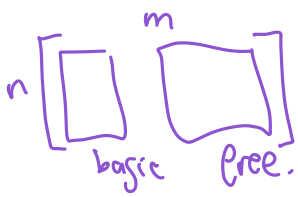
So we have some intuition about how this process works: write an LP in SEF, use our slack to find a starting solution, and improve our objective by moving our feasible solution around. This is going to tie back to our idea of basic feasible solutions and how optimal values occur at corners.

Putting it all together:

Basic feasible solutions occur when ...

we have n lin. ind. tight constraints!

but when we have a $n \times m$ matrix, linear algebra tells us...



this should have inf. many soln $\to$ n basic, m-n free

but for us, we'll want to treat all variables as candidates to be basic. To do this, we'll consider *square* $n \times n$ *submatrices* of our constraint matrix. If the submatrix is nonsingular, the solution is unique! These are our basic solutions! We'll lift them to the appropriate dimension for A by... setting free variables = 0

$\boxed{\text{ex}}$ $\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 1 \end{bmatrix}$

$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 1 \end{bmatrix} \to \text{lift} \begin{bmatrix} 0 \\ 0 \\ 5 \\ 4 \\ 1 \end{bmatrix}$

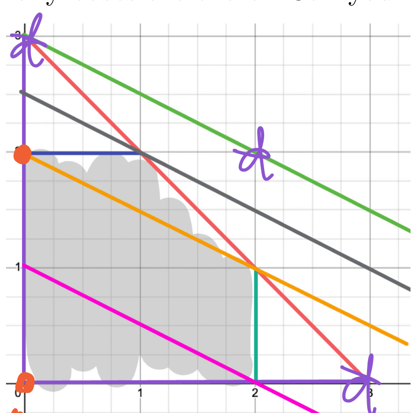
$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 1 \end{bmatrix}$ $\quad \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 4 \end{bmatrix} \to \begin{bmatrix} 0 \\ 1 \\ 4 \\ 4 \\ 0 \end{bmatrix}$

Definition. *(basis, basic solution, basic feasible solution) Given a polyhedron defined $Ax = b$ for $n \times m$ matrix A with linearly independent rows, a basis B is a $n \times n$ nonsingular submatrix. Each basis has a unique solution $x = B^{-1}b$, which can lifted to a basic solution for A. If $x \geq 0$, solution is basic feasible.*

Example: Below is the linear program that we were working with when we started today, along with our graph that we built a few days ago. Put the program into SEF. How many bases are there? Can you identify them on the graph?

{2,3,4}

$\begin{bmatrix} 0 \\ 2 \\ 1 \\ 2 \\ 0 \end{bmatrix}$



$$\begin{cases} \max & \frac{1}{2}x_1 + x_2 \\ \text{s. t.} & x_1 + x_2 \leq 3 \\ & x_1 \leq 2 \\ & x_2 \leq 2 \\ & x_i \geq 0 \end{cases}$$

$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 2 \end{bmatrix}$

{3,4,5}

{1,3,4}

{2,3,5}

Example: When we have an unbounded variable x_i, we introduce two new variables, $x_i^+ \geq 0$ and $x_i^- \geq 0$ such that $x_i = x_i^+ - x_i^-$. We can also use this to help define $|x_i|$, which can be useful.

a.) Why is it *not necessarily true* that $|x_i| = x_i^+ + x_i^-$?

$1 = \underset{x_i^+}{3} - \underset{x_i^-}{2}$ $\quad |x_i| = x_i^+ + x_i^- = 5$ ῠ

b.) Why *will it be true* that $|x_i| = x_i^+ + x_i^-$ for basic solutions?

$$\begin{array}{cc} x_i^+ & x_i^- \\ 3 & -3 \\ 5 & -5 \\ -1 & 1 \end{array}$$

when we introduce x_i^+ and x_i^-

the col for x_i^- is $-$ col for x_i^+

So we have that not both can be in a basis

So let's get back formulating the simplex method. it turns out that the process we've been doing "walks between" adjacent basic feasible solutions.

- Suppose we're at a basic feasible solution from basis B.
- We pick which variable x_i we want to add to the basis and increase it by t.
- With a bit of algebra, we solve for the t that allows that constraint:

$$t = \min\left\{\frac{b_j}{A_{j,i}} : A_{j,i} > 0\right\}$$

What happens when there is no t?

- We allow x_i to increase by that much. Notice that $x_i > 0$ and $x_j = 0$. As such, x_i *enters* the basis and x_j *leaves* the basis.

Question: How do we know which variable to swap into the basis?

Question: How do we know when we're done?

Example: Solve the linear program below.

$$\max\{10 + c^T x : Ax = b, x \geq 0\}$$

$$A = \begin{bmatrix} 1 & 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 1 & -1/2 & 0 \\ 0 & 3/2 & 0 & 1/2 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 5 \\ 1 \\ 9 \end{bmatrix} \quad c = \begin{bmatrix} 0 \\ 2 \\ 0 \\ -1 \\ 0 \end{bmatrix}$$

However, this is a lot to keep track of: all the values of t for increases, what the active basis is, the objective in the form we want it to be in... There's a more convenient, if less illuminating, way to keep this information! It's called a tableau.

For a linear program $\max\{z + c^T x : Ax = b, x \geq 0\}$, we build a tableau

$$T = \left(\begin{array}{c|c|c} 1 & -c^T & z \\ \hline 0 & A & b \end{array}\right)$$

and with careful manipulations (row operations), we can get our optimal. We'll try one out.

Suppose we have the linear program below:

$$\begin{cases} \max & 0 + [2,3,0,0,0]^T x \\ \text{s.t.} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & 1 \end{bmatrix} x = \begin{bmatrix} 6 \\ 10 \\ 4 \end{bmatrix} \\ & x \geq 0 \end{cases}$$

We'll write out the tableau.

Where should we *pivot*? ____________. Think of making pivot positions in lin alg, we're basically putting this tableau into rref, just not with the typical shape.

And again... and maybe again...?

This is great! We have an algorithm that will give solutions to our linear programs, and even a convenient way to note it. But, remember, we had big dreams. We wanted a program that would not only give solutions when they existed, but also (1) test for infeasibility and unboundedness and (2) even give those certificates when it could.

Turns out, this still can! However, we haven't tackled a big question yet. Can you spot it?