Linear programming

● Input: System of inequalities or equalities over the reals R

A linear cost function

● Output: Value for variables that minimizes cost function

Example: Minimize $6x+4y$

Subject to $3x + 2y + 5z \geq 1$
$z = 2$

● Note: Can equivalently maximize the negative of the cost function

- Max flow is a special case of LP

Variables: ?

- Max flow is a special case of LP

Variables: f(u,v)

Max ?

- Max flow is a special case of LP

Variables: $f(u,v)$

Max $f(s,V)$

subject to ?

- Max flow is a special case of LP

Variables: $f(u,v)$

Max $f(s,V)$

subject to $f(v,V)=0$

$f(u,v) \leq c(u,v)$

$f(u,v) = -f(v,u)$

- Standard form:
  Min c.x
  $Ax = b$
  $x \geq 0$

- Claim: The general form can be reduced to the standard

- Proof:

Express inequality $a_i \,.\, x \geq b_i$ as $a_i \,.\, x - s_i = b_i$ and $s_i \geq 0$

Express $x \in R$ as $(x^+ - x^-)$ with $x^+ \geq 0$, $x^- \geq 0$

■

- Definition: P := {x : Ax=b, x ≥ 0}, known as polytope
  x is vertex if not ∃ y ≠ 0 : x+y ∈ P and x-y ∈ P

- Claim: If min {c.x : x ∈ P} is finite, it is achieved at a vertex.

- Proof:

Suppose x not a vertex. Take y ≠ 0 : x+y ∈ P and x-y ∈ P.

Assume cy < 0, then x+y is a better solution

Assume cy > 0, then x-y is a better solution

- Definition: P := {x : Ax=b, x ≥ 0}, known as polytope
  x is vertex if not ∃ y ≠ 0 : x+y ∈ P and x-y ∈ P

- Claim: If min {c.x : x ∈ P} is finite, it is achieved at a vertex.

- Proof:

Suppose x not a vertex. Take y ≠ 0 : x+y ∈ P and x-y ∈ P.

Assume cy = 0. Can assume $y_j < 0$ for some j, since y ≠ 0.
Note x + λ y ∈ P since Ay = 0 as A(x-y) = A(x+y) = b
Increase λ from 0 until one more variable is 0.
Note that if a var. was zero it still is, because $x_i = 0$ ➔ $y_i = 0$
otherwise can't be x+y ∈ P and x-y ∈ P

Have new x' with one more zero variable, and no worse cost
Repeat whole argument on x'.
This will eventually stop as (0, 0, …, 0) is a vertex. ■

- This motivates looking for solutions at vertices

The simplex algorithm moves from vertex to vertex

- Most basic example of simplex algorithm:

min $x + y : 2x + 3y=1,\ x, y \geq 0$.
Vertices = (0,1/3), (1/2,0).

Somehow start at vertex $s_1 := (1/2,0)$
Write $x = 2^{-1} (1 - 3y)$ (var > 0 as function of var = 0)
Cost = $x + y = 2^{-1} (1 - 3y) + y = 0.5 - 0.5y$.

Can increment y and reduce the cost, as long as $x \geq 0$

This takes us to $s_2 := (0,1/3)$
Write $y = 3^{-1}(1-2x)$.
Cost = $x + y = x + 1/3-2x/3 = x/3 + 1/3$.

Note this holds for any solution. s_2 has $x = 0$ so is optimal.

- The simplex method:

min $c.x : Ax=b,\ x \geq 0$.

Start with some solution s

$B := \{j : s_j > 0\}$, let A_B = columns of A corresponding to B

N := the other coordinates

Assume A_B is invertible

$$\min c_B.x_B + c_N.x_N \ : A_B\, x_B + A_N\, x_N = b,\ x \geq 0.$$

Write $x_B = A_B^{-1} (b - A_N\, x_N)$

$$\text{Cost} = c_B\ A_B^{-1} (b - A_N\, x_N) + c_N\, x_N$$

$$= c_B\ A_B^{-1}\, b + x_N \underbrace{(c_N - c_B A_B^{-1} A_N)}_{r}$$

● Cost = $c_B A_B^{-1} b + x_N r$

● If $r_j < 0$ for some j, increase corresponding var in x_N in current solution s as much as possible, to arrive to s'.

Now the increased variable is > 0, and another will be 0

s' has smaller cost. Repeat.

● If $r \geq 0$, note this holds for every solution.
Current solution has $x_N = 0$, so it is optimal.

Note: New solution s is vertex, because if s+y and s-y $\in$ P then y_N is 0; but x_B is a function of x_N so that stays same too.

Note: We assumed A_B is invertible

This is not always the case, e.g. if you have too many zero variables. In general, this may make the simplex algorithm take exponential time or even never terminate.

There exist more complicated, polynomial-time algorithms

However the simplex works well in practice

Research in the area is still very active.

- Example:

$$\begin{aligned} \text{Minimize} \quad & 6x_1 + 4x_2 \\ \text{Subject to} \quad & 2x_1 + 3x_2 = 9 \\ & x_1 + x_3 = 3 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

min c.x : Ax=b, x ≥ 0.

$$A = \begin{bmatrix} 2 & 3 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad b = [9 \quad 3]^T \qquad c = [6 \quad 4 \quad 0]$$

Start with s = (3, 1, 0)

min c.x : Ax=b, x ≥ 0.

$$A = \begin{pmatrix} 2 & 3 & 0 \\ 1 & 0 & 1 \end{pmatrix} \qquad b = [9 \quad 3]^T \qquad c = [6 \quad 4 \quad 0]$$

Let's start with s = (3, 1, 0)

$B := \{j : s_j > 0\} = \{1, 2\}$

$N := \{3\}$

$$A_B = \begin{pmatrix} 2 & 3 \\ 1 & 0 \end{pmatrix} \quad A_N = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad A_B^{-1} = \begin{pmatrix} 0 & 1 \\ 1/3 & -2/3 \end{pmatrix}$$

$r = (c_N - c_B A_B^{-1} A_N)$ $\qquad c_B = [6 \quad 4]$ $\qquad c_N = [0]$

$= 0 - 10/3$

$= -10/3$

r < 0, so increase x_3 as much as possible,

arrive to s' = (0, 3, 3)

$\min c.x : Ax=b,\ x \geq 0.$

$$A = \begin{bmatrix} 2 & 3 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad b = [9 \quad 3]^T \qquad c = [6 \quad 4 \quad 0]$$

Now we are at s' = (0, 3, 3). Repeat.

$B := \{j : s_j > 0\} = \{2, 3\}$

$N := \{1\}$

$$A_B = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \quad A_N = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad A_B^{-1} = \begin{bmatrix} 1/3 & 0 \\ 0 & 1 \end{bmatrix}$$

$r = (c_N - c_B A_B^{-1} A_N)$ $\qquad c_B = [4 \quad 0] \qquad c_N = [6]$

$= 6$

Since $r \geq 0$, s' = (0, 3, 3) is an optimal solution,
and the optimal value is 12.

- Duality:

z : min $x_1 + 2x_2 + 4\,x_3 \quad = cx$

subject to $x_1 + x_2 + 2x_3 = 5$

$2x_1 + x_2 + 3x_3 = 8$

$x \geq 0$

Want: Best lower bound on z:

Can you think of any lower bound?

- Duality:

$z : \min\ x_1 + 2x_2 + 4\,x_3 \quad = cx$

subject to $x_1 + x_2 + 2x_3 = 5$

$2x_1 + x_2 + 3x_3 = 8$

$x \geq 0$

Want: Best lower bound on z:

1st equation ➔ $z \geq$?

- Duality:

$z : \min x_1 + 2x_2 + 4 x_3 \quad = cx$

subject to $x_1 + x_2 + 2x_3 = 5$

$2x_1 + x_2 + 3x_3 = 8$

$x \geq 0$

Want: Best lower bound on z:

1st equation ➔ $z \geq 5$

A better lower bound?

- Duality:

z : min $x_1 + 2x_2 + 4\,x_3 \quad = cx$

subject to $x_1 + x_2 + 2x_3 = 5$

$2x_1 + x_2 + 3x_3 = 8$

$x \geq 0$

Want: Best lower bound on z:

1st equation ➔ $z \geq 5$

3(1st) - 2nd ➔ $x_1 + 2x_2 + 3x_3 = 7 \leq z$

This process can be automated:

Find max $5\,y_1 + 8\,y_2$

subject to $y_1 + 2y_2 \leq 1$ (1st column $\leq c_1$)

$y_1 + y_2 \leq 2$ (2nd $\leq c_2$)

$2y_1 + 3y_2 \leq 4$ (3rd $\leq c_3$)

- Duality:

Consider primal: min cx : Ax=b, $x \geq 0$

Suppose we multiply each equation by a number y_j and sum:
We get y Ax = y b.

Now, if $yA \leq c$, then $yAx \leq cx$, because $x \geq 0$

Note yAx = yb.

So a generic lower bound is given by
dual: max yb : $yA \leq c$
equivalently,
max yb: $A^T y \leq c$

Note: the dual of the dual is the primal.

- Linear programming duality theorem:

$\min cx : Ax=b, x \geq 0 \quad = \quad \max by : A^T y \leq c$

when they are both finite.

A.k.a. min-max theorem, Hahn–Banach theorem

- Main tool in proving duality: Farkas' lemma.

- Recall fundamental theorem of linear algebra

$\exists\ x : Ax = b$ XOR $\exists\ y$ such that $yA = A^T y = 0$ and $b.y \neq 0$

i.e., b is in the span of the columns iff b is orthogonal to any vector that is orthogonal to the columns of A

- Farkas lemma:

$\exists\ x : Ax \leq b$ XOR $\exists\ y \geq 0 : yA = A^T y = 0$ but $b.y = -1$

- Given Farkas' lemma, proof of duality is mostly notation

Duality proof:

$p^* := \min \{cx : Ax=b, x \geq 0\}$

$d^* := \max \{by: A^T y \leq c \}$

Weak duality: $p^* \geq d^*$

Proof: Let $c x^* = p^*$, $by^* = d^*$, where x^* and y^* feasible.

Then $A^T y^* \leq c \Rightarrow A^T y^* x^* \leq c^*x = p^*$.

But $A^T y^* x^* = y^* b = d^*$.

Duality proof:

$p^* := \min \{cx : Ax=b, x \geq 0\}$

$d^* := \max \{by: A^T y \leq c \}$

Strong duality: $p^* \leq d^*$

We want $y : A^T y \leq c$ and $b.y \geq p^*$.

We express this as $\begin{bmatrix} A^T \\ -b \end{bmatrix} y \leq \begin{bmatrix} c \\ -p^* \end{bmatrix}$

If no such y, by Farkas $\exists\, z \geq 0 : [A\ -b]\, z = 0$, $[c\ -p^*]z = -1 < 0$

Write $z = [x, \lambda\,]$ $\quad Ax = \lambda\, b$, $\quad cx < \lambda\, p^*$

Case $\lambda > 0$. Let $x' := x/\lambda$. Then $Ax' = b$ and $cx' < p^*$

Case $\lambda = 0$. Then $Ax = 0$, $cx < 0$. So $A(x+x^*) = b$, $c(x+x^*) < p^*$

In either case we contradict the optimality of p^*. ■

Farkas: $\exists\, x : Ax \leq b$ XOR $\exists\, y \geq 0 : yA = A^T y = 0$ and $b.y = -1$

Proof By induction on number of variables

Base case: zero variables.

The system is of the form $0 \leq b$.

If $b_i \geq 0\ \forall\, i$ then we have a solution but can't get $b.y = -1$

Otherwise $b_i < 0$ for some i. In this case there is no solution.

Then letting $y_i = 1/b_i$ and the rest 0 we get $b.y = -1$.

Farkas: $\exists\, x : Ax \le b$ XOR $\exists\, y \ge 0 : yA = A^T y = 0$ and $b.y = -1$

Proof Induction step Write $x = (x',t)$.

Up to non-negative scaling, each equation in $Ax \le b$ is one of:

$a_i^+(x') + t \le b_i$, $a_i^-(x') - t \le b_i$, $a_i^0(x') \le b_i$

For any x', above system solvable in t iff

$a_i^0(x') \le b_i \;\forall\, i$, and $a_i^-(x') - b_i \le b_j - a_j^+(x') \;\forall\, i, j$

If such an x' exists, we have a solution $x = (x',t)$.

There is no $y \ge 0 : yA = 0$ and $b.y = -1$,

since otherwise, as seen before,

from $Ax \le b$ we obtain that $0 = yAx \le by = -1$

Farkas: $\exists\, x : Ax \le b$ XOR $\exists\, y \ge 0 : yA = A^T y = 0$ and $b.y = -1$

Proof Induction step Write $x = (x',t)$.

Up to non-negative scaling, each equation in $Ax \le b$ is one of:

$a_i^+(x') + t \le b_i$, $a_i^-(x') - t \le b_i$, $a_i^0(x') \le b_i$

For any x', above system solvable in t iff

$a_i^0(x') \le b_i\ \forall\, i$, and $a_i^-(x') - b_i \le b_j - a_j^+(x')\ \forall\, i, j$

If such an x' does not exist:

Above system is $A' x' \le b'$.

By induction, $\exists\, y' : y' A' = 0$ and $b' y' = -1$.

Since $a_i^-(x') + a_j^+(x') = a_i^+(x') + t + (a_i^-(x') - t)$ this gives a corresponding y such that $yA = 0$ and $b.y = -1$. ■

Problem (increasing generality)	Method
Linear programming	Simplex (P)
Semi-definite programming	Interior point (P) Multiplicative weights update (P) (A)
Convex programming	Ellipsoid (enough to have separator) Gradient descent (A)

(P) = Somewhat practical

(A) = Approximate solutions: runtime is poly(1/eps) to satisfy constraints within eps. (As opposed to log(1/eps) runtime, which allows for exact solutions.)

All problems admit duality formulations (strong duality for Lagrangian)

Reference: Convex Optimization – Boyd and Vandenberghe