Linear Programming

GIAN Short Course on Optimization: Applications, Algorithms, and Computation

Sven Leyffer

Argonne National Laboratory

September 12-24, 2016

Outline

Introduction to Linear Programming

Simplest nonlinear optimization problem is a linear program (LP)

$$\begin{array}{lll} \underset{x}{\text{minimize}} & c^T x & \\ \text{subject to} & a_i^T x = b_i & i \in \mathcal{E} \\ & a_i^T x \geq b_i & i \in \mathcal{I}, \end{array}$$

where $\mathcal{E}, \mathcal{I}$ are equality and inequality constraints, and $x \in \mathbb{R}^n$.

- Name "linear program" dates back to when Dantzig used LPs to solve planning problems for US Air Force.
- Fundamental building block of nonlinear algorithms.
- Fundamental building block of mixed-integer algorithms.
- Efficient commercial and open-source solvers

Introduction to Linear Programming

Simplest nonlinear optimization problem is a linear program (LP)

$$\begin{array}{lll} \underset{x}{\text{minimize}} & c^T x & \\ \text{subject to} & a_i^T x = b_i & i \in \mathcal{E} \\ & a_i^T x \geq b_i & i \in \mathcal{I}, \end{array}$$

Text book standard form of linear program:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

... note A in constraints, not A^T

Our form makes it easier to explain certain methods ...

Introduction to Linear Programming

Simplest nonlinear optimization problem is a linear program (LP)

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & c^T x \\
\text{subject to} \quad & a_i^T x = b_i \quad i \in \mathcal{E} \\
& a_i^T x \geq b_i \quad i \in \mathcal{I},
\end{aligned}
$$

Solvers allow more flexible problem definitions:

- Bounds on variables: $l \leq x \leq u$
- Two-sided constraints: $l_c \leq A^T x \leq u_c$

Solvers exploit special structure

- Network constraints $\Rightarrow$ can form inverse explicitly

The Busy College Student Problem

How should a college student spend his/her time?

- Day is divided into regular tasks:
  'Study', 'Lecture', 'Tutorial', 'Sleep', 'Eat', 'Friends', & 'Beer'
- Student derives benefit from each of these tasks
- College and student's parents place constraints on tasks
- Student must decide how much time to spend on each task

Defining the Problem Vaiables

For set of tasks, $\mathcal{T}$, define $\mathrm{h(t)} \geq 0$ as hours spent on task $t \in \mathcal{T}$

Building the Objective

Each task $t \in \mathcal{T}$ has Value(t) to student; goal is to maximize value

$$\underset{\mathrm{h}}{\text{maximize}} \quad \sum_{t \in \mathcal{T}} \mathrm{Value}(t) \cdot \mathrm{h}(t)$$

The Busy College Student Problem

Constraints imposed by College regarding split of study times

- Must spend at least as much time in lectures as in study/tutorials

$$\text{h(study)} + \text{h(tutorial)} \leq \text{h(lecture)}$$

- Must study at least 8 hours per day

$$\text{h(study)} + \text{h(tutorial)} + \text{h(lecture)} \geq 8$$

- Must achieve minimum course credit (different for study, tutorial, lectures:

$$\text{h(study)} + \frac{3}{2}\text{h(tutorial)} + 2\text{h(lecture)} \geq 10$$

The Busy College Student Problem

Constraints imposed by the parents and universe:

- Parents rules for a healthy life style
  - Spend at least 10 hours sleeping or eating

$$\mathrm{h}(\mathrm{eat}) + \mathrm{h}(\mathrm{sleep}) \geq 10$$

  - Don't overeat and get enough sleep:

$$\mathrm{h}(\mathrm{sleep}) \geq 8\mathrm{h}(\mathrm{eat})$$

- Can only spend 24 hours in a day

$$\sum_{t \in \mathcal{T}} \mathrm{h}(t) \leq 24$$

Building the Student Model in AMPL

Create a txt file (e.g. called `Student.mod`) with ...

Definition the set of tasks, $\mathcal{T}$: 'Study', 'Lecture', 'Tutorial', 'Sleep', 'Eat', 'Friends', & 'Beer'

```
# ... set of Tasks student can perform
set Tasks := { 'Study', 'Lecture', 'Tutorial', 'Sleep',
               'Eat', 'Friends', 'Beer' };
```

Definition the model parameters (value)

```
# ... parameters: value of each task
param Value{Tasks} >= 0, default 1;
```

... default value of 0 (indifferent), and requiring nonnegativity

Definition the variables (hours per task)

```
# ... variables: hours per task
var h{Tasks} >= 0;
```

Building the Student Model in AMPL

Define the objective function:

$$\underset{\mathsf{h}}{\text{maximize}} \quad \sum_{t \in \mathcal{T}} \mathsf{Value}(t) \cdot \mathsf{h}(t)$$

```
# ... maximize total value to student
maximize fun: sum{t in Tasks} Value[t] * h[t];
```

Add the constraints, e.g. only 24 hours in day:

$$\sum_{t \in \mathcal{T}} \mathsf{h}(t) \leq 24$$

```
subject to
  # ... finite number of hours per day
  hoursPerDay: sum{t in Tasks} h[t] <= 24;
```

Building the Student Model in AMPL

Add the parent's rules for a healthy life style

- Spend at least 10 hours sleeping or eating

$$h(\text{eat}) + h(\text{sleep}) \geq 10$$

- Don't overeat and get enough sleep:

$$h(\text{sleep}) \geq 8h(\text{eat})$$

```
parentsRule1: h['Sleep'] + h['Eat'] >= 10;
parentsRule2: h['Sleep'] >= 8*h['Eat'];
```

NB: Only need one `subject to` in model file.

Building the Student Model in AMPL

Add the remaining constraints, and then define the data:

```
data;

param: Value :=    # ... international survey data
'Study'         3
'Lecture'       1
'Tutorial'      2
'Sleep'         2
 'Eat'          6
'Friends'      10
'Beer'          8 ;
```

... or create a separate data file, e.g. `Student001.dat`.

Running the Student Model in AMPL

Now open AMPL, load the model, select a solver, and sole:

```
% ampl
ampl: reset; model Student.mod;
ampl: option solver ipopt;
ampl: solve;
ampl: display h, fun;
```

... where last command shows the solution

Outline

Active-Set Method for Linear Programming

Introduce active-set method for linear programs (LPs)

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x \\ \text{subject to } & a_i^T x = b_i \quad i \in \mathcal{E} \\ & a_i^T x \geq b_i \quad i \in \mathcal{I}, \end{aligned}$$

where

- $\mathcal{E}, \mathcal{I}$ are equality and inequality constraints
- variables $x \in \mathbb{R}^n$.

Relationship to Simplex Methods

- Active-set methods are equivalent to Simplex method
- More intuitive, and generalizes to quadratic programs
- Dual active-set method is active-set applied to dual LP

Basic Facts About Linear Programming

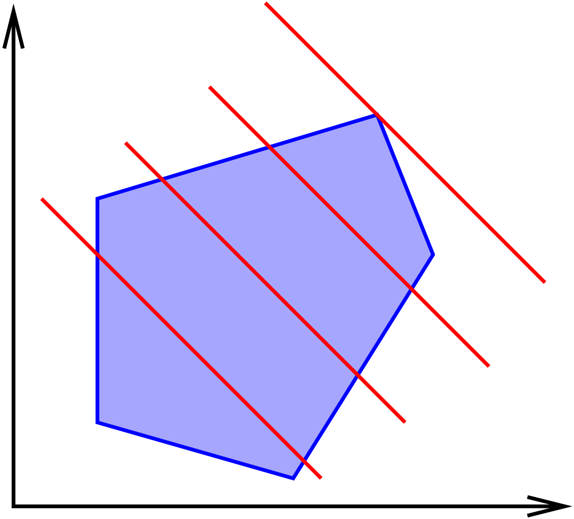
$$\underset{x}{\text{minimize}} \; c^T x \quad \text{subject to } A_{\mathcal{E}}^T x = b_{\mathcal{E}} \quad A_{\mathcal{I}}^T x \geq b_{\mathcal{I}}$$

- Feasible set may be empty ... detect in phase-I methods ...
- Feasible can be unbounded $\Rightarrow$ LP may be unbounded ... detect this situation during the line-search
- Feasible set is polyhedron; every vertex has n active constraints ... more, if vertex is degenerate
- If solution exists, then there exists a vertex solution

Active-Set Methods for LP

Moves from feasible vertex to another reducing $c^T x$.

Active-Set Method for Linear Programming



Move from vertex to vertex, reducing objective

Active-Set Method for LP

Active-Set Methods for LP

Moves from feasible vertex to another reducing $c^T x$.

Every iterate, $x^{(k)}$ is vertex of feasible set:

$$a_i^T x = b_i, \quad i \in \mathcal{W} \quad \Leftrightarrow \quad A_k^T x = b_k,$$

where

- $\mathcal{W} \subset \mathcal{A}(x)$ working set
  - If vertex is non-degenerate (exactly n active constraints), then $\mathcal{W} = \mathcal{A}(x)$
  - Make this non-degeneracy assumption from now on ... solvers can handle degeneracy
- Jacobian and right-hand-side

$$A_k := [a_i]_{i \in \mathcal{W}} \in \mathbb{R}^{n \times n} \quad \text{and} \quad b_k^T := (b_i)_{i \in \mathcal{W}} \in \mathbb{R}^n$$

Active-Set Method for LP

Active-Set Methods for LP

Moves from feasible vertex to another reducing $c^T x$.

Every iterate, $x^{(k)}$ is vertex of feasible set:

$$a_i^T x = b_i, \quad i \in \mathcal{W} \quad \Leftrightarrow \quad A_k^T x = b_k,$$

At $x^{(k)}$, the Lagrange multipliers are

$$y^{(k)} = A_k c.$$

Optimality Test for LP

$$y_i^{(k)} \geq 0, \forall i \in \mathcal{I} \cap \mathcal{W} \quad \Rightarrow \; x^{(k)} \text{ optimal.}$$

Active-Set Method for LP

Active-Set Methods for LP

Move from vertex to vertex along a common edge reducing $c^T x$.

Define feasible edges as

$$A_k^{-T} := [s_i]_{i \in \mathcal{W}} \in \mathbb{R}^{n \times n},$$

$\Rightarrow$ slope of objective along edge s_i is $y_i^{(k)} = s_i^T c$

If $x^{(k)}$ not optimal, then there exists $y_q^{(k)} < 0$
$\Rightarrow$ edge s_q is feasible descend direction

Possibly choice for q is most negative multiplier,

$$y_q := \min_{i \in \mathcal{I} \cap \mathcal{W}} y_i$$

... not good in practice ... take scaling into account!

Active-Set Method for LP

Active-Set Methods for LP

Move from vertex to vertex along a common edge reducing $c^T x$.

Given $x^{(k)}$ not optimal and $y_q^{(k)} < 0$
... search along the edge $s_q \Rightarrow$ move away from constraint q
Drop constraint q from working set, $\mathcal{W}$, move along line

$$x = x^{(k)} + \alpha s_q$$

Consider effect on inactive constraints, $i \in \mathcal{I} : i \notin \mathcal{W}$:

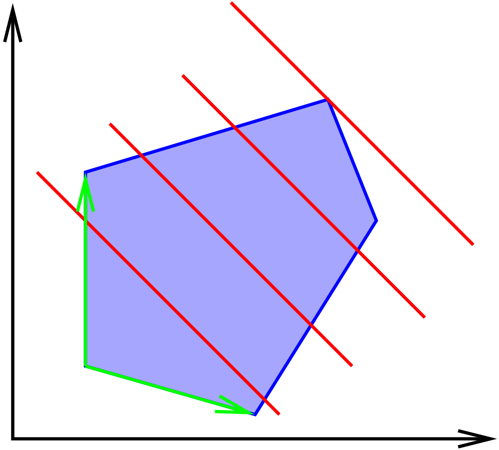
$$r_i := a_i^T x - b_i = a_i^T x^{(k)} + \alpha a_i^T s_q - b_i =: r_i^{(k)} + \alpha a_i^T s_q.$$

Inactive constraint only becomes active, if $a_i^T s_q < 0$, after step α:

$$0 = r_i = r_i^{(k)} + \alpha a_i^T s_q \quad \Leftrightarrow \quad \alpha = \frac{r_i^{(k)}}{-a_i^T s_q}$$

Active-Set Method for LP

- From vertex to vertex along common edge reducing $c^T x$.
- Given $x^{(k)}$ not optimal and $y_q^{(k)} < 0$
  ... search along the edge $s_q \Rightarrow$ move away from constraint q



Side-Track: Degeneracy in LP Active-Set

Active-Set Methods for LP

Move from vertex to vertex along a common edge reducing $c^T x$.

Move from $x^{(k)}$ along edge $x = x^{(k)} + \alpha s_q$ with $y_q^{(k)} < 0$

Inactive constraint $i \in \mathcal{I} : i \notin \mathcal{W}$ ...
... becomes active, if $a_i^T s_q < 0$, after step α:

$$0 = r_i = r_i^{(k)} + \alpha a_i^T s_q \quad \Leftrightarrow \quad \alpha = \frac{r_i^{(k)}}{-a_i^T s_q}$$

Degeneracy in LP

If vertex $x^{(k)}$ degenerate, then $\exists$ more than n active constraints
... can cause $\alpha = 0$, if $\exists i : \; r_i^{(k)} = 0$ ... may cycle

Active-Set Method for LP

Active-Set Methods for LP

Move from vertex to vertex along a common edge reducing $c^T x$.

Given $x^{(k)}$ not optimal and $y_q^{(k)} < 0$
... search along the edge $s_q \Rightarrow$ move away from constraint q
Drop constraint q from working set, $\mathcal{W}$, move along line

$$x = x^{(k)} + \alpha s_q$$

Consider effect on inactive constraints, $i \in \mathcal{I} : i \notin \mathcal{W}$:

$$r_i := a_i^T x - b_i = a_i^T x^{(k)} + \alpha a_i^T s_q - b_i =: r_i^{(k)} + \alpha a_i^T s_q.$$

Inactive constraint only becomes active, if $a_i^T s_q < 0$, after step α:

$$0 = r_i = r_i^{(k)} + \alpha a_i^T s_q \quad \Leftrightarrow \quad \alpha = \frac{r_i^{(k)}}{-a_i^T s_q}$$

Active-Set Method for LP

Active-Set Methods for LP

Move from vertex to vertex along a common edge reducing $c^T x$.

Drop constraint q from working set, $\mathcal{W}$, move along $x = x^{(k)} + \alpha s_q$

Inactive constraint becomes active, if $a_i^T s_q < 0$, after step α:

$$0 = r_i = r_i^{(k)} + \alpha a_i^T s_q \quad \Leftrightarrow \quad \alpha = -r_i^{(k)} / a_i^T s_q$$

Stay feasible wrt constraints $\Rightarrow$ find 1^{st} newly active constraint:

$$\alpha = \min_{i \in \mathcal{I} : i \notin \mathcal{W}, a_i^T s_q < 0} \quad -r_i^{(k)} / a_i^T s_q$$

If $\nexists i \in \mathcal{I} : i \notin \mathcal{W}$ such that $a_i^T s_q < 0 \Rightarrow \alpha = \infty$, LP unbounded

Otherwise, $\alpha < \infty$, constraint p becomes active

$\Rightarrow$ exchange p and q in working set, move new vertex, $x^{(k+1)}$

Active-Set Method for Linear Programming

Given initial feasible vertex, $x^{(0)}$, working set $\mathcal{W}^{(0)}$, set $k = 0$

repeat

Optimality Test: Let $A_k := [a_i]_{i \in \mathcal{W}^{(k)}}$ compute $y^{(k)} = A_k^{-1} c$

Find $y_q := \min \left\{ y_i : i \in \mathcal{W}^{(k)} \cap \mathcal{I} \right\}$

if $y_q \geq 0$ **then** $x^{(k)}$ **optimal** solution ;

else

Ratio Test: s_q be column of A^{-T} corresp. to y_q

$$\alpha = \min_{i \in \mathcal{I} : i \notin \mathcal{W}, a_i^T s_q < 0} \frac{b_i - a_i^T x^{(k)}}{-a_i^T s_q} =: \frac{b_p - a_p^T x^{(k)}}{-a_p^T s_q}$$

if $a_i^T s_q \geq 0, \ \forall i \in \mathcal{I} : i \notin \mathcal{W}$ **then** LP is **unbounded** ;

else

Pivot: p and q in $\mathcal{W}^{(k+1)} = \mathcal{W}^{(k)} - \{q\} \cup \{p\}$ Set $x^{(k+1)} = x^{(k)} + \alpha s_q$ and $k = k + 1$

end

end

until $x^{(k)}$ *is optimal or LP unbounded*;

Modern LP Solvers

Modern LP solvers more sophisticated

- Anti-cycling rules to handle degeneracy
- More sophisticated pivoting choice (leaving constraint)
- Using inverse A^{-1} inefficient and numerically unstable.
  - Use factors of active-set matrix $A_k = L_k U_k$, where L_k is lower and U_k is upper triangular matrix
  - Update factors after removing a_q and adding q_p
  - Efficient & numerically stable
- Dual active-set methods start from dual feasible point ... e.g. after changing RHS in branching $\Rightarrow$ great for MIP

LP Solvers for Huge LPs

Active-set solvers inefficient or very large problems ...
... interior-point methods are alternative with good complexity

Getting Initial Feasible Point for LPs

If no initial feasible vertex, then solve auxiliary LP

- Add surplus variables that measure infeasibility
- Solve resulting LP for initial feasible vertex ...
  ... or proof that LP is infeasible

$$
\begin{aligned}
\underset{x,s}{\text{minimize}} \quad & \sum_{i \in \mathcal{E}} (s_i^+ + s_i^-) + \sum_{i \in \mathcal{I}} s_i \\
\text{subject to } \; & a_i^T x - b_i = s_i^+ - s_i^- \qquad i \in \mathcal{E} \\
& a_i^T x - b_i \geq -s_i \qquad\qquad i \in \mathcal{I} \\
& s^+ \geq 0, \; s^- \geq 0, \; s \geq 0.
\end{aligned}
$$

Getting Initial Feasible Point for LPs

$$\begin{aligned} \underset{x,s}{\text{minimize}} \quad & \sum_{i \in \mathcal{E}} (s_i^+ + s_i^-) + \sum_{i \in \mathcal{I}} s_i \\ \text{subject to } \quad & a_i^T x - b_i = s_i^+ - s_i^- \qquad i \in \mathcal{E} \\ & a_i^T x - b_i \geq -s_i \qquad\qquad i \in \mathcal{I} \\ & s^+ \geq 0, \; s^- \geq 0, \; s \geq 0. \end{aligned}$$

For any x, initial feasible point for auxiliary LP is

$$s_i := \min\left(0, b_i - a_i^T x\right),$$

$$s_i^- := \min\left(0, b_i - a_i^T x\right), \quad s_i^+ := \min\left(0, -b_i + a_i^T x\right),$$

If solution $(s = 0, s^+ = 0, s^- = 0)$ then feasible, otherwise not.

Summary & Teaching Points

Simple model as LP

- From description to mathematical formulation
- Translated mathematical formulation into AMPL
  ... there exist open-source alternatives:
  - `JuMP` based on MIT's Julia project
  - `Zimpl` is AMPL clone developed at ZIB in Berlin
  - Can be used with open-source solvers
- Discussed active-set method for LP
  - Move from vertex to vertex, reducing objctive
  - Phase I method for intial feasible point