Linear programs

10-725 Optimization
Geoff Gordon
Ryan Tibshirani
Review

• Newton w/ equality constraints

• Examples:
  ‣ bundle adjustment
  ‣ MLE in exponential families

• Convergence: damped phase, quadratic phase

• Compare: Newton, FISTA, (stoch) (sub)gradient

• Variations: trust region, quasi-Newton, Gauss-Newton, Levenberg-Marquardt
Variations: Fisher scoring

- Recall Newton in exponential family
  \[ \text{Var}\[x \mid \theta]\text{d}\theta = E[x \mid \theta] - \bar{x} \]

- Can use this formula in place of Newton, even if not an exponential family
  - descent direction, even w/ no regularization
  - “Hessian” is independent of data
  - often a wider radius of convergence than Newton
  - can be superlinearly convergent
Administrivia

- HW2 due now
- HW3 out tonight (hopefully)
- Final project update
  - project milestone report requirements on web site
  - final poster session (3:30–6:30 12/12 NSH Atrium, 3PM setup)
  - set up meetings w/ TA mentors
Linear programs

• n variables: $x = (x_1, x_2, \ldots, x_n)^T$
  ‣ ranges: $[l_i, u_i]$

• Objective:

• m constraints (equality or inequality):

• Example:
Sketching an LP

\[
\begin{align*}
\text{max } & \quad 2x + 3y \\
\text{s.t.} & \quad x + y \leq 4 \\
& \quad 2x + 5y \leq 12 \\
& \quad x + 2y \leq 5 \\
& \quad x, y \geq 0
\end{align*}
\]
Did the prof get it right?
Polyhedra

- hull({points}) or \(\cap\)({halfspaces})

- Vertices, edges, faces
  - in general: d-faces
    - n vars: d-face = set of feasible points that make independent halfspace constrs tight
    - therefore, dimensionality =
  - n vars, \(m \geq n\) halfspaces: can have \(-\)-faces thru \(-\)-faces
Matrix notation

- For a vector of variables $v$ and a constant matrix $A$ and vector $b$,
  - $Av \leq b$ [componentwise]
- Objective: $c^Tv$
- E.g.:
  - $A =$
  - $b =$
  - $c =$

$max\ 2x+3y\ s.t.\ 
\begin{align*}
  x + y &\leq 4 \\
  2x + 5y &\leq 12 \\
  x + 2y &\leq 5 \\
  x,\ y &\geq 0
\end{align*}$
Finding the optimum

max $2x + 3y$ s.t.
$x + y \leq 4$
$2x + 5y \leq 12$
$x + 2y \leq 5$
$x, y \geq 0$
Where’s my ball?
Unhappy ball

\[ \text{max} \ 2x + 3y \ \text{subject to} \]
\[ x \geq 5 \]
\[ x \leq 1 \]
Convention

- min over empty set =
- max over empty set =
- Adding an element always
Linear feasibility

- find \((x, y)\) s.t.
  - \(x + y \leq 4\)
  - \(2x + 5y \leq 12\)
  - \(x + 2y \leq 5\)
  - \(x, y \geq 0\)

- Any easier than LP?
Binary search

- find \((x, y)\) s.t.
  - \(x + y \leq 4\)
  - \(2x + 5y \leq 12\)
  - \(x + 2y \leq 5\)
  - \(x, y \geq 0\)

vs. max \(2x + 3y\) s.t. \(\uparrow\)
Transforming LPs

- Getting rid of inequalities (except variable bounds)
  \[ x + y \leq 4 \]

- Getting rid of equalities
  \[ x + 2y = 4 \]
Transforming LPs

- Getting rid of free vars

\[
\begin{align*}
\text{max } x + y \text{ s.t. } & \\
2x + y & \leq 3 \\
y & \geq 0
\end{align*}
\]

- Getting rid of bounded vars

\[x \in [2, 5]\]
Standard form LP

- all variables are nonnegative
- all constraints are equalities
- E.g.: max $c^Tq$ s.t. $Aq = b$, $q \geq 0$
  - $q = [x \ y \ u \ v \ w]^T$

max $2x + 3y$ s.t.
  - $x + y \leq 4$
  - $2x + 5y \leq 12$
  - $x + 2y \leq 5$
  - $x, y \geq 0$

tableau
Objective in tableau

- Add an extra variable $z$
  - constrain it to $=$ the objective

\[
\begin{array}{ccccccc}
& x & y & u & v & w & z & \text{RHS} \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 4 \\
2 & 5 & 0 & 1 & 0 & 0 & 0 & 12 \\
1 & 2 & 0 & 0 & 1 & 0 & 0 & 5 \\
-2 & -3 & 0 & 0 & 0 & 1 & 0 & 0 \\
\end{array}
\]

max $2x + 3y$ s.t.
- $x + y \leq 4$
- $2x + 5y \leq 12$
- $x + 2y \leq 5$
- $x, y \geq 0$
Standard v. inequality forms

- \[ \text{max } 2x + 3y \text{ s.t.} \]
  - \[ x + y \leq 4 \]
  - \[ 2x + 5y \leq 12 \]
  - \[ x + 2y \leq 5 \]
  - \[ x, y \geq 0 \]

- or s.t.
  - \[ x + y + u = 4 \]
  - \[ 2x + 5y + v = 12 \]
  - \[ x + 2y + w = 5 \]
  - \[ x, y, u, v, w \geq 0 \]

If std fm has \( n \) vars, \( m \) eqns then ineq form has \( n - m \) vars and \( m + (n - m) = n \) ineqs

(here \( m = 3, n = 5 \))
Faces in standard form

- Inequality form
  - n vars, \( m \geq n \) halfspaces: can have 0-faces thru n-faces
  - d-face makes n–d inequalities tight

- Standard form
  - n nonneg. vars, \( m \leq n \) equalities: -faces thru -faces
Why is standard form useful?

- Can take linear combinations of constraints
- E.g., $x + 2y = 4$ & $2x + 3y = 5$
- Easy to manipulate via row operations
- Easy to find corners by Gaussian elimination
Example

```
1 1 1 0 0  4   set x, y = 0
2 5 0 1 0  12
1 2 0 0 1  5

1 1 1 0 0  4   set v, w = 0
2 5 0 1 0  12
1 2 0 0 1  5

1 1 1 0 0  4   set x, u = 0
2 5 0 1 0  12
1 2 0 0 1  5
```
What happened?
Row operations

- Can replace any row with linear combination of existing rows
  - as long as we don’t lose independence
- Eliminate x from 2nd and 3rd rows

\[
\begin{array}{cccccc}
A & b \\
1 & 1 & 1 & 0 & 0 & 4 \\
2 & 5 & 0 & 1 & 0 & 12 \\
1 & 2 & 0 & 0 & 1 & 5 \\
\end{array}
\]
Presto change-o

- Which are the slacks now?
  -
  -

- Eliminating $x$ from all but one constr:

- Constraint we used to eliminate $x$:
The “new” LP

Objective: was $z = 2x + 3y$

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>u</th>
<th>v</th>
<th>w</th>
<th>z</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-2</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>.</td>
</tr>
</tbody>
</table>

$y + u \leq 4$
$3y - 2u \leq 4$
$y - u \leq 1$
$y, u \geq 0$
Sketching standard form

- Drop the slacks, sketch in inequality form
- May be several ways
What if there aren’t slacks?

• Use row ops to \textit{make} some:
  
  \begin{itemize}
    \item u, v, w \geq 0
    \item u + 2v + w = 3
    \item 3u + v - w = 5
  \end{itemize}
Matlab version

\[ A = \begin{bmatrix} 1 & 2 & 1; \\ 3 & 1 & -1 \end{bmatrix}; \]
\[ b = \begin{bmatrix} 3; 5 \end{bmatrix}; \]
\[ A(:,1:2) \backslash A \]
\[ \text{ans} = \begin{bmatrix} 1.0000 & 0 & -0.6000 \\ 0 & 1.0000 & 0.8000 \end{bmatrix} \]
\[ A(:,1:2) \backslash b \]
\[ \text{ans} = \begin{bmatrix} 1.4000 \\ 0.8000 \end{bmatrix} \]
Matlab with \( z \)

- always pick \( z \)'s column (here, col 6)
- remember \( z \) is unconstrained

\[
\begin{align*}
\text{max } \quad z &= 2x + 3y \\
\text{s.t. } &\quad x + y + u = 4 \\
&\quad 2x + 5y + v = 12 \\
&\quad x + 2y + w = 5 \\
&\quad x, y, u, v, w \geq 0
\end{align*}
\]

\[
\begin{array}{cccccc}
\text{A and } b & \text{result } &= & A(:,[1\ 4\ 5\ 6]) & \backslash & [A\ b] \\
\hline \hline
x & y & u & v & w & z & \text{RHS} \\
1 & 1 & 1 & 0 & 0 & 0 & 4 \\
2 & 5 & 0 & 1 & 0 & 0 & 12 \\
1 & 2 & 0 & 0 & 1 & 0 & 5 \\
-2 & -3 & 0 & 0 & 0 & 1 & 0 \\
\hline \hline
x & y & u & v & w & z & \text{RHS} \\
1 & 1 & 1 & 0 & 0 & 0 & 4 \\
0 & 3 & -2 & 1 & 0 & 0 & 4 \\
0 & 1 & -1 & 0 & 1 & 0 & 1 \\
0 & -1 & 2 & 0 & 0 & 1 & 8
\end{array}
\]
Basis

- \{u, v, w, z\} and \{x, v, w, z\} are bases
  - elements = basic variables (always m of them)
  - easy to write values of basic variables in terms of non-basic ones
  - e.g., set x=y=0
  - e.g., set y=u=0

\[
\begin{array}{ccccccc}
 x & y & u & v & w & z & \text{RHS} \\
 1 & 1 & 1 & 0 & 0 & 0 & 4 \\
 2 & 5 & 0 & 1 & 0 & 0 & 12 \\
 1 & 2 & 0 & 0 & 1 & 0 & 5 \\
 -2 & -3 & 0 & 0 & 0 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
 x & y & u & v & w & z & \text{RHS} \\
 1 & 1 & 1 & 0 & 0 & 0 & 4 \\
 0 & 3 & -2 & 1 & 0 & 0 & 4 \\
 0 & 1 & -1 & 0 & 1 & 0 & 1 \\
 0 & -1 & 2 & 0 & 0 & 1 & 8 \\
\end{array}
\]
Basic solutions
Bases ↔ corners

- Std form: n vars, m equations
  - fix ineq form: n–m vars, n ineqs
- Pick a basis B for std form
  - m basic vars (≥ 0), n–m nonbasic (set to 0)
- Each nonbasic var yields a tight inequality
  - var is either a slack or explicit in ineq fm
    - explicit: one of n–m “trivial” (x≥0) ineqs tight
    - slack: one of m “real” ineqs tight
- Ineq fm: n–m vars and n–m tight ineqs → corner
What if we can’t pick basis?

- E.g., suppose A doesn’t have full row rank
  - can’t pick m linearly independent cols

- Ex:
What if we can’t pick basis?

- E.g., suppose fewer vars than constraints
  - A taller than it is wide, $m \geq n$
  - can’t pick enough cols of $A$ to make a square matrix

- Ex:
Nonsingular

- We can assume
  - $n \geq m$ (at least as many vars as constrs)
  - $A$ has full row rank
- Else, drop rows (maintaining rank) until it’s true
- Called *nonsingular* standard form LP
**Naive (sloooow) algorithm**

- Put in nonsingular standard form
- Iterate through all subsets of n vars
  - if m constraints, how many subsets?
- Check each for
  - full rank ("basis-ness")
  - feasibility (RHS $\geq 0$)
- If pass both tests, compute objective
- Maintain running winner, return at end
Improving our search

- Naive: enumerate all possible bases
- Smarter: maybe neighbors of good bases are also good?
- Simplex algorithm: repeatedly move to a neighboring basis to improve objective
  - continue to assume nonsingular standard form LP
Neighboring bases

- Two bases are **neighbors** if they share \((m-1)\) variables.
- Neighboring feasible bases correspond to vertices connected by an edge.

\[
\begin{array}{ccccccc}
 x & y & z & u & v & w & \text{RHS} \\
1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 \\
\end{array}
\]

def’n: pivot, enter, exit
Example

$$\text{max } z = 2x + 3y \text{ s.t.}$$
$$x + y \leq 4$$
$$2x + 5y \leq 12$$
$$x + 2y \leq 5$$
$$x \leq 4$$

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>s</th>
<th>t</th>
<th>u</th>
<th>v</th>
<th>z</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>-2</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>z</td>
<td>RHS</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>0.4</td>
<td>1</td>
<td>0</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.4</td>
</tr>
<tr>
<td>0.6</td>
<td>0</td>
<td>1</td>
<td>-0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.6</td>
</tr>
<tr>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>-0.4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>-0.8</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.2</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>y</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>z</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x$</td>
<td>$y$</td>
<td>$s$</td>
<td>$t$</td>
<td>$u$</td>
<td>$v$</td>
<td>$z$</td>
<td>RHS</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>
Initial basis

- So far, assumed we started w/ feasible basic solution—in fact, it was trivial to find one
- Not always so easy in general

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>u</th>
<th>v</th>
<th>w</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>
**Big M**

0 ≤ x, y, s1..s6
max x - 2y

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>slacks</th>
<th>z</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1 0 0 0 0 0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-2</td>
<td>0 1 0 0 0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>0 0 1 0 0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td>-2</td>
<td>0 0 0 1 0</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>2</td>
<td>0 0 0 0 1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

- Can make it easy: variant of slack trick
  - For each violated constraint, add var w/ coeff −1
  - Penalize in objective
Simplex in one slide
(skipping degeneracy handling)

• Given a nonsingular standard-form max LP
• Start from a feasible basis and its tableau
  ‣ big-M if needed
• Pick non-basic variable w/ coeff in objective ≤ 0
• Pivot it into basis, getting neighboring basis
  ‣ select exiting variable to keep feasibility
• Repeat until all non-basic variables have objective ≥ 0
Degeneracy

- Not every set of m variables yields a corner
  - some have rank < m (not a basis)
  - some are infeasible
- Can the reverse be true? Can two bases yield the same corner?
Degeneracy

\[
\begin{array}{cccccc}
\text{x} & \text{y} & \text{u} & \text{v} & \text{w} & \text{RHS} \\
1 & 1 & 1 & 0 & 0 & 4 \\
2 & 5 & 0 & 1 & 0 & 12 \\
1 & 2 & 0 & 0 & 1 & 16/3 \\
0 & 1 & 0 & -2 & 5 & 8/3 \\
0 & 1 & 0 & 1 & -2 & 4/3 \\
0 & 0 & 1 & 1 & -3 & 0 \\
1 & 0 & 2 & 0 & -1 & 8/3 \\
0 & 1 & -1 & 0 & 1 & 4/3 \\
0 & 0 & 1 & 1 & -3 & 0 \\
\end{array}
\]
Degeneracy in 3D
**Bases & degeneracy**

- How many bases for vertex A?
- Are they all neighbors of one another?
- Are they all neighbors of B?
Dual degeneracy

- More than \( m \) entries in objective row = 0
  - so, a nonbasic variable has reduced cost = 0
  - objective orthogonal to a \( d \)-face for \( d \geq 1 \)
Handling degeneracy

- Sometimes have to make pivots that don’t improve objective
  - stay at same corner (exiting variable was already 0)
  - move to another corner w/ same objective (coeff of entering variable in objective was 0)

- Problem of cycling
  - need an anti-cycling rule (there are many…)
  - e.g.: add tiny random numbers to obj, RHS