Convex Optimization EE364a: Review Session 3
Using CVX

Stanford University

Winter Quarter, 1/29/2013
Outline

Disciplined convex programming and CVX

Declaring variables

Forming objectives and constraints

Using sets

Defining new functions

CVX hints/warnings
Outline

Disciplined convex programming and CVX

Declaring variables

Forming objectives and constraints

Using sets

Defining new functions

CVX hints/warnings
Convex optimization solvers

- **LP solvers**
  - lots available (GLPK, Excel, Matlab’s `linprog`, …)

- **cone solvers**
  - typically handle (combinations of) LP, SOCP, SDP cones
  - several available (SDPT3, SeDuMi, CSDP, …)

- **general convex solvers**
  - some available (CVXOPT, MOSEK, …)

- plus lots of special purpose or application specific solvers
- could write your own
Transforming problems to standard form

- there are lots of tricks for transforming a problem into an equivalent one that has a standard form (e.g., LP, SDP)
  - introducing slack variables
  - introducing new variables that upper bound expressions
- these tricks greatly extend the applicability of standard solvers
- writing code to carry out this transformation is often painful
- **modeling systems** can partly automate this step
Modeling systems

a typical modeling system

- automates most of the transformation to standard form;
supports
  - declaring optimization variables
  - describing the objective function
  - describing the constraints
  - choosing (and configuring) the solver
- when given a problem instance, calls the solver
- interprets and returns the solver’s status (optimal, infeasible, . . .)
- (when solved) transforms the solution back to original form
Disciplined convex programming

- describe objective and constraints using expressions formed from
  - a set of basic atoms (convex, concave functions)
  - a restricted set of operations or rules (that preserve convexity)
- modeling system keeps track of affine, convex, concave expressions
- rules ensure that
  - expressions recognized as convex (concave) are convex (concave)
  - but, some convex (concave) expressions are not recognized as convex (concave)
- problems described using DCP are convex by construction
CVX

- uses DCP
- runs in Matlab, between the cvx_begin and cvx_end commands
- relies on SDPT3 or SeDuMi (LP/SOCP/SDP) solvers
- refer to user guide, online help for more info
- the CVX example library has more than a hundred examples
Outline

Disciplined convex programming and CVX

Declaring variables

Forming objectives and constraints

Using sets

Defining new functions

CVX hints/warnings

Declaring variables
Declaring variables

- declare variables with
  variable name[(dims)] [attributes]
  - variable x(3);
  - variable C(4,3);
  - variable S(3,3) symmetric;
  - variable D(3,3) diagonal;
  - variables y z;
- mathematical interpretation
  - \( x \in \mathbb{R}^3 \)
  - \( C \in \mathbb{R}^{4 \times 3} \)
  - \( S \in \mathbb{S}^3 \)
  - \( D \in \mathbb{S}^3, D_{i,j} = 0 \) if \( i \neq j \)
  - \( y, z \in \mathbb{R} \)
Outline

Disciplined convex programming and CVX

Declaring variables

Forming objectives and constraints

Using sets

Defining new functions

CVX hints/warnings
Objectives and constraints

- **Objective** can be
  - minimize(convex expression)
  - maximize(concave expression)
  - omitted (feasibility problem)

- **Constraints** can be
  - convex expression <= concave expression
  - concave expression >= convex expression
  - affine expression == affine expression
  - omitted (unconstrained problem)
Composition rules

- can combine atoms using valid composition rules, e.g.:
  - a convex function of an affine function is convex
  - the negative of a convex function is concave
  - a convex, nondecreasing function of a convex function is convex
  - a concave, nondecreasing function of a concave function is concave
Composition rules — multiple arguments

- For convex $h$, $h(g_1, \ldots, g_k)$ is recognized as convex if, for each $i$,
  - $g_i$ is affine, or
  - $g_i$ is convex and $h$ is nondecreasing in its $i$th arg, or
  - $g_i$ is concave and $h$ is nonincreasing in its $i$th arg

- For concave $h$, $h(g_1, \ldots, g_k)$ is recognized as concave if, for each $i$,
  - $g_i$ is affine, or
  - $g_i$ is convex and $h$ is nonincreasing in its $i$th arg, or
  - $g_i$ is concave and $h$ is nondecreasing in its $i$th arg
Affine expressions

- declare constants and variables
  - $A = \text{randn}(4, 3)$;
  - variables $x(3)$ $y(4)$;
- then form affine expressions
  - $3 \times x + 4$
  - $A \times x - y$
  - $x(2:3)$
  - $\text{sum}(x)$
Example: Affine expressions

- Form affine constraint $x_1 \leq x_2 \leq \ldots \leq x_n$ in CVX
  - Variable $x(n)$
    - $x(2:n)-x(1:n-1) \geq 0$

- Form constraints for $p \in \mathbb{R}^n$ to be in probability simplex
  - Variable $p(n)$
    - $p \geq 0$
    - $\text{sum}(p) = 1$
### Some functions

<table>
<thead>
<tr>
<th>function</th>
<th>meaning</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>norm(x, p)</td>
<td>$|x|_p$</td>
<td>cvx</td>
</tr>
<tr>
<td>square(x)</td>
<td>$x^2$</td>
<td>cvx</td>
</tr>
<tr>
<td>square_pos(x)</td>
<td>$(x_+)^2$</td>
<td>cvx, nondecr</td>
</tr>
<tr>
<td>pos(x)</td>
<td>$x_+$</td>
<td>cvx, nondecr</td>
</tr>
<tr>
<td>sum_largest(x,k)</td>
<td>$x[1] + \cdots + x[k]$</td>
<td>cvx, nondecr</td>
</tr>
<tr>
<td>sqrt(x)</td>
<td>$\sqrt{x}$ ($x \geq 0$)</td>
<td>ccv, nondecr</td>
</tr>
<tr>
<td>inv_pos(x)</td>
<td>$1/x$ ($x &gt; 0$)</td>
<td>cvx, nonincr</td>
</tr>
<tr>
<td>max(x)</td>
<td>$\max{x_1, \ldots, x_n}$</td>
<td>cvx, nondecr</td>
</tr>
<tr>
<td>quad_over_lin(x,y)</td>
<td>$x^2/y$ ($y &gt; 0$)</td>
<td>cvx, nonincr</td>
</tr>
<tr>
<td>lambda_max(X)</td>
<td>$\lambda_{\max}(X)$ ($X = X^T$)</td>
<td>cvx</td>
</tr>
<tr>
<td>huber(x)</td>
<td>$\begin{cases} x^2, &amp;</td>
<td>x</td>
</tr>
</tbody>
</table>
Example: Some functions

- why define `square_pos(x)`?
  - useful for composition rules
  - function is monotone (nondecreasing)
  - `square_pos([cvx function])` is convex
Valid (recognized) examples

u, v, x, y are scalar variables; X is a symmetric $3 \times 3$ variable

- **convex:**
  - $\text{norm}(A\times x - y) + 0.1\times \text{norm}(x, 1)$
  - $\text{quad}\_\text{over}\_\text{lin}(u - v, 1 - \text{square}(v))$
  - $\lambda_{\text{max}}(2X - 4\times \text{eye}(3))$
  - $\text{norm}(2X - 3, \ 'fro')$

- **concave:**
  - $\min(1 + 2\times u, 1 - \text{max}(2, v))$
  - $\sqrt{v} - 4.55\times \text{inv}\_\text{pos}(u - v)$
Rejected examples

\[ |u|, v, x, y \] are scalar variables

- neither convex nor concave:
  - \( \text{square}(x) - \text{square}(y) \)
  - \( \text{norm}(A \cdot x - y) - 0.1 \cdot \text{norm}(x, 1) \)

- rejected due to limited DCP ruleset:
  - \( \sqrt{\text{sum}(\text{square}(x))} \) (is convex; could use \( \text{norm}(x) \))
  - \( \text{square}(1 + x^2) \) (is convex; could use \( \text{square}_\text{pos}(1 + x^2) \), or \( 1 + 2 \cdot \text{pow}_\text{pos}(x, 2) + \text{pow}_\text{pos}(x, 4) \))
Outline

Disciplined convex programming and CVX

Declaring variables

Forming objectives and constraints

Using sets

Defining new functions

CVX hints/warnings

Using sets
Sets

- some constraints are more naturally expressed with convex sets
- sets in CVX work by creating unnamed variables constrained to the set
- examples:
  - semidefinite(n)
  - nonnegative(n)
  - simplex(n)
  - lorentz(n)
- semidefinite(n), say, returns an unnamed (symmetric matrix) variable that is constrained to be positive semidefinite
Using the semidefinite cone

variables: $X$ (symmetric matrix), $z$ (vector), $t$ (scalar)

constants: $A$ and $B$ (matrices)

- $X == \text{semidefinite}(n)$
  - means $X \in S^n_+$ (or $X \succeq 0$)
- $A*X*A' - X == B*\text{semidefinite}(n)*B'$
  - means $\exists Z \succeq 0$ so that $AXA^T - X = BZB^T$
- $[X \ z; \ z' \ t] == \text{semidefinite}(n+1)$
  - means $\begin{bmatrix} X & z \\ z^T & t \end{bmatrix} \succeq 0$
Example: Using the semidefinite cone

- constrain eigenvalues of $X \in S^n$ to be between $\lambda_{\text{min}}$ and $\lambda_{\text{max}}$

- equivalent mathematical description
  - $\lambda_{\text{min}} I \preceq X \preceq \lambda_{\text{max}} I$

- CVX command
  - variable $X(n,n)$ symmetric
    - $X - \lambda_{\text{min}} \cdot \text{eye}(n) == \text{semidefinite}(n)$
    - $\lambda_{\text{max}} \cdot \text{eye}(n) - X == \text{semidefinite}(n)$
Outline

Disciplined convex programming and CVX

Declaring variables

Forming objectives and constraints

Using sets

Defining new functions

CVX hints/warnings

Defining new functions
Defining new functions

- can make a new function using existing atoms
- **example:** the convex deadzone function

\[
f(x) = \max\{|x| - 1, 0\} = \begin{cases} 
0, & |x| \leq 1 \\
x - 1, & x > 1 \\
1 - x, & x < -1
\end{cases}
\]

- create a file `deadzone.m` with the code

```matlab
function y = deadzone(x)
    y = max(abs(x) - 1, 0)
end
```

- `deadzone` makes sense both within and outside of CVX
Defining functions via incompletely specified problems

- Suppose $f_0, \ldots, f_m$ are convex in $(x, z)$
- Let $\phi(x)$ be optimal value of convex problem, with variable $z$ and parameter $x$

\[
\begin{align*}
\text{minimize} & \quad f_0(x, z) \\
\text{subject to} & \quad f_i(x, z) \leq 0, \quad i = 1, \ldots, m \\
& \quad A_1x + A_2z = b
\end{align*}
\]

- $\phi$ is a convex function
- Problem above sometimes called *incompletely specified* since $x$ isn’t (yet) given
- An incompletely specified concave maximization problem defines a concave function
CVX functions via incompletely specified problems

Implement in cvx with

\[
\begin{align*}
\text{function } & \text{ cvx_optval } = \phi(x) \\
\text{cvx_begin} & \\
& \text{variable } z; \\
& \text{minimize}(f0(x, z)) \\
& \text{subject to} \\
& \quad f1(x, z) \leq 0; \ldots \\
& \quad A1*x + A2*z == b; \\
\text{cvx_end}
\end{align*}
\]

- \(\text{function } \phi \text{ will work for numeric } x \text{ (by solving the problem)}\)
- \(\text{function } \phi \text{ can also be used inside a CVX specification, wherever a convex function can be used}\)
Simple example: Two element max

- create file max2.m containing

  function cvx_optval = max2(x, y)
  cvx_begin
    variable t;
    minimize(t)
    subject to
      x <= t;
      y <= t;
  cvx_end

- the constraints define the epigraph of the max function
- could add logic to return \( \max(x, y) \) when \( x, y \) are numeric (otherwise, an LP is solved to evaluate the max of two numbers!)
A more complex example

- $f(x) = x + x^{1.5} + x^{2.5}$, with $\text{dom} f = \mathbb{R}_+$, is a convex, monotone increasing function
- its inverse $g = f^{-1}$ is concave, monotone increasing, with $\text{dom} g = \mathbb{R}_+$
- there is no closed form expression for $g$
- $g(y)$ is optimal value of problem

\[
\begin{align*}
\text{maximize} & \quad t \\
\text{subject to} & \quad t_+ + t_+^{1.5} + t_+^{2.5} \leq y
\end{align*}
\]

(for $y < 0$, this problem is infeasible, so optimal value is $-\infty$)
implement as

function cvx_optval = g(y)
cvx_begin
    variable t;
    maximize(t)
    pos(t) + pow_pos(t, 1.5) + pow_pos(t, 2.5) <= y;
cvx_end

use it as an ordinary function, as in g(14.3), or within CVX as a concave function:

cvx_begin
    variables x y;
    minimize(quad_over_lin(x, y) + 4*x + 5*y)
    g(x) + 2*g(y) >= 2;
cvx_end
Outline

Disciplined convex programming and CVX

Declaring variables

Forming objectives and constraints

Using sets

Defining new functions

CVX hints/warnings
 CVX hints/warnings

- watch out for = (assignment) versus == (equality constraint)
- \( X \geq 0 \), with matrix \( X \), is an elementwise inequality
- \( X \geq \text{semidefinite}(n) \) means: \( X \) is elementwise larger than some positive semidefinite matrix (which is likely not what you want)
- writing subject to is unnecessary (but can look nicer)
- use brackets around objective functions:
  use \texttt{minimize (c'*x)}, not \texttt{minimize c'*x}
- double inequalities like \( 0 \leq x \leq 1 \) don’t work in some versions (older) of CVX; use \( 0 \leq x; x \leq 1 \) instead
CVX hints/warnings

- declare `cvx_begin quiet` to suppress output
- many problems traditionally stated using convex quadratic forms can be posed as norm problems (which can have better numerical properties):
  \[ x' P x \leq 1 \text{ can be replaced with } \text{norm}(\text{chol}(P)x) \leq 1 \]
- `log`, `exp`, entropy-type functions implemented using experimental successive approximation method, which can be slow, unreliable