Disciplined Convex Programming and CVX

- convex optimization solvers
- modeling systems
- disciplined convex programming
- CVX
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

(we’ll study, and write, solvers later in the quarter)
Transforming problems to standard form

- you’ve seen lots of tricks for transforming a problem into an equivalent one that has a standard form (e.g., LP, SDP)

- 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
Some current modeling systems

- AMPL & GAMS (proprietary)
  - developed in the 1980s, still widely used in traditional OR
  - no support for convex optimization

- YALMIP (‘Yet Another LMI Parser’)
  - first matlab-based object-oriented modeling system with special support for convex optimization
  - can use many different solvers; can handle some nonconvex problems

- CVXMOD/CVXOPT (in alpha)
  - python based, completely GPLed
  - cone and custom solvers

- CVX
  - matlab based, GPL, uses SDPT3/SeDuMi
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/SOC/SDP) solvers
- refer to user guide, online help for more info
- the CVX example library has more than a hundred examples
Example: Constrained norm minimization

A = randn(5, 3);
b = randn(5, 1);
cvx_begin
  variable x(3);
  minimize(norm(A*x - b, 1))
  subject to
    -0.5 <= x;
    x <= 0.3;
cvx_end

- between cvx_begin and cvx_end, x is a CVX variable
- statement subject to does nothing, but can be added for readability
- inequalities are interpreted elementwise
What CVX does

after `cvx_end`, CVX

- transforms problem into an LP
- calls solver SDPT3
- overwrites (object) \( x \) with (numeric) optimal value
- assigns problem optimal value to `cvx_optval`
- assigns problem status (which here is `Solved`) to `cvx_status`

(had problem been infeasible, `cvx_status` would be `Infeasible` and \( x \) would be `NaN`
Variables and affine expressions

• 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;

• form affine expressions
  - A = randn(4, 3);
  - variables x(3) y(4);
  - 3*x + 4
  - A*x - y
  - x(2:3)
  - sum(x)
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 in y</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>
Composition rules

• can combine atoms using valid composition rules, \textit{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

• 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 $i$th arg, or
  – $g_i$ is concave and $h$ is nondecreasing in $i$th arg
Valid (recognized) examples

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

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

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

\[ u, v, x, y \text{ are scalar variables} \]

- neither convex nor concave:
  - \( \text{square}(x) - \text{square}(y) \)
  - \( \|A\cdot x - y\| - 0.1\|x\|_1 \)

- rejected due to limited DCP ruleset:
  - \( \sqrt{\text{sum}(\text{square}(x))} \) (is convex; could use \( \|x\| \))
  - \( \text{square}(1 + x^2) \) (is convex; could use \( \text{square\_pos}(1 + x^2) \), or \( 1 + 2\cdot\text{pow\_pos}(x, 2) + \text{pow\_pos}(x, 4) \))
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 $\begin{bmatrix} X & z \\ z^T & t \end{bmatrix} \succeq 0$
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)
More involved example

A = randn(5);
A = A’*A;
cvx_begin
    variable X(5, 5) symmetric;
    variable y;
    minimize(norm(X) - 10*sqrt(y))
    subject to
        X - A == semidefinite(5);
        X(2,5) == 2*y;
        X(3,1) >= 0.8;
        y <= 4;
cvx_end
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)
```

• `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 \( x \)

\[
\begin{align*}
\text{minimize} & \quad f_0(x, z) \\
\text{subject to} & \quad f_i(x, z) \leq 0, \quad i = 1, \ldots, m \\
& \quad A_1 x + A_2 z = b
\end{align*}
\]

• \( \phi \) 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

function cvx_optval = phi(x)
cvx_begin
    variable z;
    minimize(f0(x, z))
    subject to
        f1(x, z) <= 0; ...
        A1*x + A2*z == b;
cvx_end

• function phi will work for numeric x (by solving the problem)

• function phi 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

```matlab
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
  
  $\text{maximize } t$
  
  $\text{subject to } t_+ + t_+^{1.5} + t_+^{2.5} \leq y$

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

```plaintext
function cvx_optval = g(y)
cvx_begin
    variable t;
    maximize(t)
    subject to
        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:

```plaintext
cvx_begin
    variables x y;
    minimize(quad_over_lin(x, y) + 4*x + 5*y)
    subject to
        g(x) + 2*g(y) >= 2;
cvx_end
```
Example

• optimal value of LP, \( f(c) = \inf \{ c^T x \mid Ax \leq b \} \), is concave function of \( c \)
• by duality (assuming feasibility of \( Ax \leq b \)) we have
  \[
  f(c) = \sup \{-\lambda^T b \mid A^T \lambda + c = 0, \lambda \geq 0\}
  \]
• define \( f \) in CVX as

```matlab
function cvx_optval = lp_opt_val(A,b,c)
  cvx_begin
    variable lambda(length(b));
    maximize(-lambda'*b);
    subject to
      A'*lambda + c == 0; lambda >= 0;
  cvx_end
end
```
• in \( \text{lp\_opt\_val}(A,b,c) \) \( A, b \) must be constant; \( c \) can be affine expression
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)

• make sure you include brackets around objective functions
  – yes: minimize($c' * x$)
  – no: minimize $c' * x$

• double inequalities like $0 \leq x \leq 1$ don’t work; use $0 \leq x; x \leq 1$ instead

• log, exp, entropy-type functions not yet implemented in CVX