Constructive Convex Analysis
and Disciplined Convex Programming

Stephen Boyd    Steven Diamond
Akshay Agrawal  Junzi Zhang
EE & CS Departments
Stanford University
Outline

Convex Optimization

Constructive Convex Analysis

Disciplined Convex Programming

Modeling Frameworks

Conclusions
Outline

Convex Optimization

Constructive Convex Analysis

Disciplined Convex Programming

Modeling Frameworks

Conclusions
Convex optimization problem — standard form

minimize \( f_0(x) \)
subject to \( f_i(x) \leq 0, \quad i = 1, \ldots, m \)
\( Ax = b \)

with variable \( x \in \mathbb{R}^n \)

- objective and inequality constraints \( f_0, \ldots, f_m \) are convex

for all \( x, y, \theta \in [0, 1]\),

\[ f_i(\theta x + (1 - \theta)y) \leq \theta f_i(x) + (1 - \theta)f_i(y) \]

i.e., graphs of \( f_i \) curve upward

- equality constraints are linear
Convex optimization problem — conic form

cone program:

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax = b, \quad x \in \mathcal{K}
\end{align*}
\]

with variable \( x \in \mathbb{R}^n \)

- linear objective, equality constraints; \( \mathcal{K} \) is convex cone
- special cases:
  - linear program (LP)
  - semidefinite program (SDP)

- the modern canonical form
- *there are well developed solvers for cone programs*
Other canonical forms

➤ quadratic program (QP):

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} x^T P x + q^T x \\
\text{subject to} & \quad l \leq A x \leq u
\end{align*}
\]

➤ smooth optimization:

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{where} & \quad f : \mathbb{R}^n \rightarrow \mathbb{R} \text{ is smooth}
\end{align*}
\]

➤ linearly constrained least squares:

\[
\begin{align*}
\text{minimize} & \quad \|A x - b\|_2^2 \\
\text{subject to} & \quad F x = g
\end{align*}
\]

➤ prox-affine:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{N} f_i(H_i x_i) \\
\text{subject to} & \quad \sum_{i=1}^{N} A_i x_i = b.
\end{align*}
\]
Why convex optimization?

▶ beautiful, fairly complete, and useful theory
▶ solution algorithms that work well in theory and practice
  ▶ convex optimization is actionable
▶ many applications in
  ▶ control
  ▶ combinatorial optimization
  ▶ signal and image processing
  ▶ communications, networks
  ▶ circuit design
  ▶ machine learning, statistics
  ▶ finance

... and many more
How do you solve a convex problem?

- use an existing custom solver for your specific problem

- develop a new solver for your problem using a currently fashionable method
  - requires work
  - but (with luck) will scale to large problems

- transform your problem into a cone program, and use a standard cone program solver
  - can be automated using domain specific languages
Curvature: Convex, concave, and affine functions

- $f$ is \textit{concave} if $-f$ is convex, \textit{i.e.}, for any $x, y, \theta \in [0, 1],$

$$f(\theta x + (1 - \theta)y) \geq \theta f(x) + (1 - \theta)f(y)$$

- $f$ is \textit{affine} if it is convex and concave, \textit{i.e.},

$$f(\theta x + (1 - \theta)y) = \theta f(x) + (1 - \theta)f(y)$$

for any $x, y, \theta \in [0, 1]$

- $f$ is affine $\iff$ it has form $f(x) = a^T x + b$
Verifying a function is convex or concave

(Verifying affine is easy)

approaches:

- via basic definition (inequality)
- via first or second order conditions, e.g., $\nabla^2 f(x) \succeq 0$

- via convex calculus: construct $f$ using
  - library of basic functions that are convex or concave
  - calculus rules or transformations that preserve convexity
Convex functions: Basic examples

- $x^p$ ($p \geq 1$ or $p \leq 0$), e.g., $x^2$, $1/x$ ($x > 0$)
- $e^x$
- $x \log x$
- $a^T x + b$
- $x^T P x$ ($P \succeq 0$)
- $\|x\|$ (any norm)
- $\max(x_1, \ldots, x_n)$
Concave functions: Basic examples

- $x^p$ $(0 \leq p \leq 1)$, e.g., $\sqrt{x}$
- $\log x$
- $\sqrt{xy}$
- $x^T P x$ $(P \preceq 0)$
- $\min(x_1, \ldots, x_n)$
Convex functions: Less basic examples

- \( x^2/y \) (\( y > 0 \)), \( x^T x/y \) (\( y > 0 \)), \( x^T Y^{-1} x \) (\( Y \succ 0 \))
- \( \log(e^{x_1} + \cdots + e^{x_n}) \)
- \( f(x) = x_{[1]} + \cdots + x_{[k]} \) (sum of largest \( k \) entries)
- \( f(x, y) = x \log(x/y) \) (\( x, y > 0 \))
- \( \lambda_{\text{max}}(X) \) (\( X = X^T \))
Concave functions: Less basic examples

- \( \log \det X, \ (\det X)^{1/n} \) \( (X \succ 0) \)
- \( \log \Phi(x) \) (\( \Phi \) is Gaussian CDF)
- \( \lambda_{\text{min}}(X) \) \( (X = X^T) \)
Calculus rules

- **nonnegative scaling**: \( f \text{ convex}, \alpha \geq 0 \implies \alpha f \text{ convex} \)

- **sum**: \( f, g \text{ convex} \implies f + g \text{ convex} \)

- **affine composition**: \( f \text{ convex} \implies f(Ax + b) \text{ convex} \)

- **pointwise maximum**: \( f_1, \ldots, f_m \text{ convex} \implies \max_i f_i(x) \text{ convex} \)

- **composition**: \( h \text{ convex increasing, } f \text{ convex} \implies h(f(x)) \text{ convex} \)

... and similar rules for concave functions

(there are other more advanced rules)
Examples

from basic functions and calculus rules, we can show convexity of . . .

▶ piecewise-linear function: \( \max_{i=1,...,k} (a_i^T x + b_i) \)
▶ \( \ell_1 \)-regularized least-squares cost: \( \|Ax - b\|^2_2 + \lambda \|x\|_1 \), with \( \lambda \geq 0 \)
▶ sum of largest \( k \) elements of \( x \): \( x_{[1]} + \cdots + x_{[k]} \)
▶ log-barrier: \( - \sum_{i=1}^m \log(-f_i(x)) \) (on \( \{x \mid f_i(x) < 0\} \), \( f_i \) convex)
▶ KL divergence: \( D(u, v) = \sum_i (u_i \log(u_i/v_i) - u_i + v_i) \) (\( u, v > 0 \))
A general composition rule

\[ h(f_1(x), \ldots, f_k(x)) \text{ is convex when } h \text{ is convex and for each } i \]

- \( h \) is increasing in argument \( i \), and \( f_i \) is convex, or
- \( h \) is decreasing in argument \( i \), and \( f_i \) is concave, or
- \( f_i \) is affine

- there's a similar rule for concave compositions (just swap convex and concave above)
- this one rule subsumes all of the others
- \textit{this is pretty much the only rule you need to know}
Example

let’s show that

\[ f(u, v) = (u + 1) \log((u + 1)/\min(u, v)) \]

is convex

▶ \( u, v \) are variables with \( u, v > 0 \)
▶ \( u + 1 \) is affine; \( \min(u, v) \) is concave
▶ since \( x \log(x/y) \) is convex in \( (x, y) \), decreasing in \( y \),

\[ f(u, v) = (u + 1) \log((u + 1)/\min(u, v)) \]

is convex
Example

- $\log(e^{u_1} + \cdots + e^{u_k})$ is convex, increasing
- so if $f(x, \omega)$ is convex in $x$ for each $\omega$ and $\gamma > 0,
  \log \left( \left( e^{\gamma f(x, \omega_1)} + \cdots + e^{\gamma f(x, \omega_k)} \right) / k \right)
  \right)$ is convex
- this is $\log E e^{\gamma f(x, \omega)}$, where $\omega \sim U \{\omega_1, \ldots, \omega_k\}$
- arises in stochastic optimization via bound
  \[
  \log \operatorname{Prob}(f(x, \omega) \geq 0) \leq \log E e^{\gamma f(x, \omega)}
  \]
Constructive convexity verification

- start with function given as expression
- build parse tree for expression
  - leaves are variables or constants/parameters
  - nodes are functions of children, following general rule
- tag each subexpression as convex, concave, affine, constant
  - variation: tag subexpression signs, use for monotonicity
    e.g., $(\cdot)^2$ is increasing if its argument is nonnegative
- sufficient (but not necessary) for convexity
Example

for \( x < 1, y < 1 \)

\[
\frac{(x - y)^2}{1 - \max(x, y)}
\]

is convex

- (leaves) \( x, y, \) and 1 are affine expressions
- \( \max(x, y) \) is convex; \( x - y \) is affine
- \( 1 - \max(x, y) \) is concave
- function \( u^2/v \) is convex, monotone decreasing in \( v \) for \( v > 0 \)
  hence, convex with \( u = x - y, \ v = 1 - \max(x, y) \)
Example

analyzed by dcp.stanford.edu (*Diamond 2014*)

Constructive Convex Analysis
Example

- $f(x) = \sqrt{1 + x^2}$ is convex

- but cannot show this using constructive convex analysis
  - (leaves) 1 is constant, $x$ is affine
  - $x^2$ is convex
  - $1 + x^2$ is convex
  - but $\sqrt{1 + x^2}$ doesn’t match general rule

- writing $f(x) = \|(1, x)\|_2$, however, works
  - $(1, x)$ is affine
  - $\|(1, x)\|_2$ is convex
Outline

Convex Optimization

Constructive Convex Analysis

Disciplined Convex Programming

Modeling Frameworks

Conclusions
Disciplined convex programming (DCP)

(Grant, Boyd, Ye, 2006)

- framework for describing convex optimization problems
- based on constructive convex analysis
- sufficient but not necessary for convexity
- basis for several domain specific languages and tools for convex optimization
Disciplined convex program: Structure

a DCP has

- zero or one **objective**, with form
  - minimize \{scalar convex expression\} or
  - maximize \{scalar concave expression\}

- zero or more **constraints**, with form
  - \{convex expression\} \(\leq\) \{concave expression\} or
  - \{concave expression\} \(\geq\) \{convex expression\} or
  - \{affine expression\} \(==\) \{affine expression\}
Disciplined convex program: Expressions

- expressions formed from
  - variables,
  - constants/parameters,
  - and functions from a library

- library functions have known convexity, monotonicity, and sign properties

- all subexpressions match general composition rule
Disciplined convex program

- A valid DCP is
  - convex-by-construction (cf. posterior convexity analysis)
  - ‘syntactically’ convex (can be checked ‘locally’)

- Convexity depends only on *attributes* of library functions, and not their meanings
  - E.g., could swap $\sqrt{\cdot}$ and $\sqrt[4]{\cdot}$, or $\exp \cdot$ and $(\cdot)_+$, since their attributes match
Canonicalization

- easy to build a DCP parser/analyzer
- not much harder to implement a canonicalizer, which transforms DCP to equivalent cone program
- then solve the cone program using a generic solver
- yields a modeling framework for convex optimization
Optimization modeling languages

- domain specific language (DSL) for optimization
- express optimization problem in high level language
  - declare variables
  - form constraints and objective
  - solve
- long history: AMPL, GAMS, ...
  - no special support for convex problems
  - very limited syntax
  - callable from, but not embedded in other languages
Modeling languages for convex optimization

all based on DCP

YALMIP Matlab Löfberg 2004
CVX Matlab Grant, Boyd 2005
CVXPY Python Diamond, Boyd; Agrawal et al. 2013; 2018
Convex.jl Julia Udell et al. 2014
CVXR R Fu, Narasimhan, Boyd 2017

some precursors

► SDPSOL (Wu, Boyd, 2000)
► LMITOOL (El Ghaoui et al., 1995)
cvx_begin
    variable x(n)  % declare vector variable
    minimize sum(square(A*x-b)) + gamma*norm(x,1)
    subject to norm(x,inf) <= 1
cvx_end

▶ A, b, gamma are constants (gamma nonnegative)
▶ variables, expressions, constraints exist inside problem
▶ after cvx_end
    ▶ problem is canonicalized to cone program
    ▶ then solved
Some functions in the CVX library

<table>
<thead>
<tr>
<th>function</th>
<th>meaning</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>norm(x, p)</td>
<td>$|x|_p$, $p \geq 1$</td>
<td>cvx</td>
</tr>
<tr>
<td>square(x)</td>
<td>$x^2$</td>
<td>cvx</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>
</tbody>
</table>
Disciplined convex programming error:
  Invalid constraint: \{convex\} == \{real constant\}
import cvxpy as cp
x = cp.Variable(n)

cost = cp.sum_squares(A@x-b) + gamma*cp.norm(x,1)
prob = cp.Problem(cp.Minimize(cost),
                  [cp.norm(x,"inf") <= 1])

opt_val = prob.solve()
solution = x.value

▶ A, b, gamma are constants (gamma nonnegative)
▶ variables, expressions, constraints exist outside of problem
▶ solve method canonicalizes, solves, assigns value attributes
## Signed DCP in CVXPY

<table>
<thead>
<tr>
<th>function</th>
<th>meaning</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(x)</td>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>huber(x)</td>
<td>$\begin{cases} x^2, &amp;</td>
<td>x</td>
</tr>
<tr>
<td>norm(x, p)</td>
<td>$|x|_p$, $p \geq 1$</td>
<td>cvx, nondecr for $x \geq 0$, nonincr for $x \leq 0$</td>
</tr>
<tr>
<td>square(x)</td>
<td>$x^2$</td>
<td>cvx, nondecr for $x \geq 0$, nonincr for $x \leq 0$</td>
</tr>
</tbody>
</table>
DCP analysis in CVXPY

\[
expr = \frac{(x - y)^2}{1 - \max(x, y)}
\]

```python
x = cp.Variable()
y = cp.Variable()
expr = cp.quad_over_lin(x - y, 1 - cp.maximum(x, y))
expr.curvature # CONVEX
expr.sign # POSITIVE
expr.is_dcp() # True
```
Parameters in CVXPY

- symbolic representations of constants
- can specify sign
- change value of constant without re-parsing problem

- for-loop style trade-off curve:
  ```python
  x_values = []
  for val in numpy.logspace(-4, 2, 100):
      gamma.value = val
      prob.solve()
      x_values.append(x.value)
  ```
# Use tools for parallelism in standard library.
from multiprocessing import Pool

# Function maps gamma value to optimal x.
def get_x(gamma_value):
    gamma.value = gamma_value
    result = prob.solve()
    return x.value

# Parallel computation with N processes.
pool = Pool(processes = N)
x_values = pool.map(get_x, numpy.logspace(-4, 2, 100))
using Convex
x = Variable(n);
cost = sum_squares(A*x-b) + gamma*norm(x,1);
prob = minimize(cost, [norm(x,Inf) <= 1]);
opt_val = solve!(prob);
solution = x.value;

- A, b, gamma are constants (gamma nonnegative)
- similar structure to CVXPY
- solve! method canonicalizes, solves, assigns value attributes
Outline

Convex Optimization

Constructive Convex Analysis

Disciplined Convex Programming

Modeling Frameworks

Conclusions
Conclusions

► DCP is a formalization of constructive convex analysis
  ► simple method to certify problem as convex
    (sufficient, but not necessary)
  ► basis of several DSLs/modeling frameworks for convex optimization

► modeling frameworks make rapid prototyping of convex optimization based methods easy
References

- *Disciplined Convex Programming* (Grant, Boyd, Ye)
- *Graph Implementations for Nonsmooth Convex Programs* (Grant, Boyd)
- *Matrix-Free Convex Optimization Modeling* (Diamond, Boyd)
- *A Rewriting System for Convex Optimization Problems* (Agrawal, Verschueren, Diamond, Boyd)

- CVXPY: [https://www.cvxpy.org/](https://www.cvxpy.org/)
- CVXR: [https://cvxr.rbind.io/](https://cvxr.rbind.io/)
- DCP tools: [https://dcp.stanford.edu/](https://dcp.stanford.edu/)