

# 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

$$\begin{array}{ll}\text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \dots, m \\ & Ax = b\end{array}$$

with variable  $x \in \mathbf{R}^n$

- ▶ objective and inequality constraints  $f_0, \dots, f_m$  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:

$$\begin{array}{ll}\text{minimize} & c^T x \\ \text{subject to} & Ax = b, \quad x \in \mathcal{K}\end{array}$$

with variable  $x \in \mathbf{R}^n$

- ▶ 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):

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}x^T Px + q^T x \\ \text{subject to} & l \leq Ax \leq u\end{array}$$

- ▶ smooth optimization:

$$\text{minimize } f(x)$$

where  $f : \mathbf{R}^n \rightarrow \mathbf{R}$  is smooth

- ▶ linearly constrained least squares:

$$\begin{array}{ll}\text{minimize} & \|Ax - b\|_2^2 \\ \text{subject to} & Fx = g\end{array}$$

- ▶ prox-affine:

$$\begin{array}{ll}\text{minimize} & \sum_{i=1}^N f_i(H_i x_i) \\ \text{subject to} & \sum_{i=1}^N A_i x_i = b.\end{array}$$

## 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*



# Outline

Convex Optimization

**Constructive Convex Analysis**

Disciplined Convex Programming

Modeling Frameworks

Conclusions

## Curvature: Convex, concave, and affine functions



- $f$  is *concave* if  $-f$  is convex, 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 *affine* if it is convex and concave, 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)
- ▶  $\max(x_1, \dots, x_n)$

## Concave functions: Basic examples

- ▶  $x^p$  ( $0 \leq p \leq 1$ ), e.g.,  $\sqrt{x}$
- ▶  $\log x$
- ▶  $\sqrt{xy}$
- ▶  $x^T P x$  ( $P \preceq 0$ )
- ▶  $\min(x_1, \dots, x_n)$

## Convex functions: Less basic examples

- ▶  $x^2/y$  ( $y > 0$ ),  $x^T x/y$  ( $y > 0$ ),  $x^T Y^{-1}x$  ( $Y \succ 0$ )
- ▶  $\log(e^{x_1} + \dots + e^{x_n})$
- ▶  $f(x) = x_{[1]} + \dots + x_{[k]}$  (sum of largest  $k$  entries)
- ▶  $f(x, y) = x \log(x/y)$  ( $x, y > 0$ )
- ▶  $\lambda_{\max}(X)$  ( $X = X^T$ )

## Concave functions: Less basic examples

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

## Calculus rules

- ▶ **nonnegative scaling:**  $f$  convex,  $\alpha \geq 0 \implies \alpha f$  convex
- ▶ **sum:**  $f, g$  convex  $\implies f + g$  convex
- ▶ **affine composition:**  $f$  convex  $\implies f(Ax + b)$  convex
- ▶ **pointwise maximum:**  $f_1, \dots, f_m$  convex  $\implies \max_i f_i(x)$  convex
- ▶ **composition:**  $h$  convex increasing,  $f$  convex  $\implies h(f(x))$  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:  $\max_{i=1,\dots,k} (a_i^T x + b_i)$
- ▶  $\ell_1$ -regularized least-squares cost:  $\|Ax - b\|_2^2 + \lambda \|x\|_1$ , with  $\lambda \geq 0$
- ▶ sum of largest  $k$  elements of  $x$ :  $x_{[1]} + \dots + x_{[k]}$
- ▶ 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

$h(f_1(x), \dots, f_k(x))$  is convex when  $h$  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
  - ▶ *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

- ▶  $\log(e^{u_1} + \dots + e^{u_k})$  is convex, increasing
- ▶ so if  $f(x, \omega)$  is convex in  $x$  for each  $\omega$  and  $\gamma > 0$ ,

$$\log \left( \left( e^{\gamma f(x, \omega_1)} + \dots + e^{\gamma f(x, \omega_k)} \right) / k \right)$$

is convex

- ▶ this is  $\log \mathbf{E} e^{\gamma f(x, \omega)}$ , where  $\omega \sim \mathcal{U}(\{\omega_1, \dots, \omega_k\})$
- ▶ arises in stochastic optimization via bound

$$\log \mathbf{Prob}(f(x, \omega) \geq 0) \leq \log \mathbf{E} e^{\gamma f(x, \omega)}$$

## 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  $x < 1, y < 1$

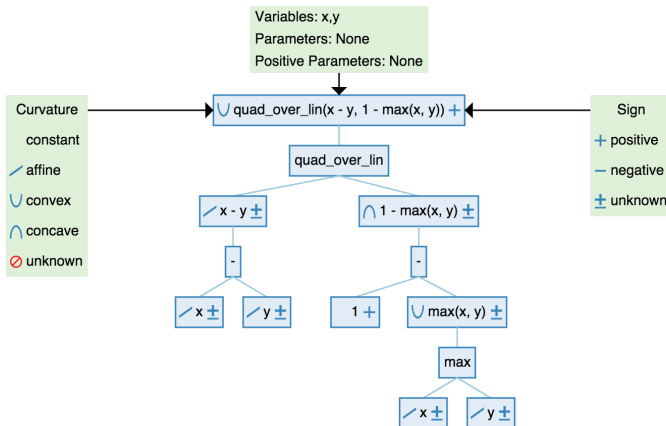
$$\frac{(x - y)^2}{1 - \max(x, y)}$$

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*)



## 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

# Outline

Convex Optimization

Constructive Convex Analysis

Disciplined Convex Programming

**Modeling Frameworks**

Conclusions

## 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

```
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

function	meaning	attributes
<code>norm(x, p)</code>	$\ x\ _p, p \geq 1$	cvx
<code>square(x)</code>	$x^2$	cvx
<code>pos(x)</code>	$x_+$	cvx, nondecr
<code>sum_largest(x,k)</code>	$x_{[1]} + \cdots + x_{[k]}$	cvx, nondecr
<code>sqrt(x)</code>	$\sqrt{x}, x \geq 0$	ccv, nondecr
<code>inv_pos(x)</code>	$1/x, x > 0$	cvx, nonincr
<code>max(x)</code>	$\max\{x_1, \dots, x_n\}$	cvx, nondecr
<code>quad_over_lin(x,y)</code>	$x^2/y, y > 0$	cvx, nonincr in $y$
<code>lambda_max(X)</code>	$\lambda_{\max}(X), X = X^T$	cvx

## DCP analysis in CVX

```
cvx_begin
    variables x y
    square(x+1) <= sqrt(y) % accepted
    max(x,y) == 1 % not DCP
    ...
```

Disciplined convex programming error:

Invalid constraint: {convex} == {real constant}

## CVXPY

```
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

function	meaning	attributes
<code>abs(x)</code>	$ x $	cvx, nondecr for $x \geq 0$ , nonincr for $x \leq 0$
<code>huber(x)</code>	$\begin{cases} x^2, &  x  \leq 1 \\ 2 x  - 1, &  x  > 1 \end{cases}$	cvx, nondecr for $x \geq 0$ , nonincr for $x \leq 0$
<code>norm(x, p)</code>	$\ x\ _p, p \geq 1$	cvx, nondecr for $x \geq 0$ , nonincr for $x \leq 0$
<code>square(x)</code>	$x^2$	cvx, nondecr for $x \geq 0$ , nonincr for $x \leq 0$

## DCP analysis in CVXPY

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

```
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:

```
x_values = []  
for val in numpy.logspace(-4, 2, 100):  
    gamma.value = val  
    prob.solve()  
    x_values.append(x.value)
```



## Parallel style trade-off curve

```
# 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))
```

## Convex.jl

```
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)
- ▶ CVX: <http://cvxr.com/>
- ▶ CVXPY: <https://www.cvxpy.org/>
- ▶ Convex.jl: <http://convexjl.readthedocs.org/>
- ▶ CVXR: <https://cvxr.rbind.io/>
- ▶ DCP tools: <https://dcp.stanford.edu/>