Convex Optimization Overview

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

Mathematical Optimization

Convex Optimization

Solvers & Modeling Languages

Examples

Summary
Outline

Mathematical Optimization

Convex Optimization

Solvers & Modeling Languages

Examples

Summary
Optimization problem

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

- $x \in \mathbb{R}^n$ is (vector) variable to be chosen
- $f_0$ is the \textit{objective function}, to be minimized
- $f_1, \ldots, f_m$ are the \textit{inequality constraint functions}
- $g_1, \ldots, g_p$ are the \textit{equality constraint functions}

- variations: maximize objective, multiple objectives, \ldots
Finding good (or best) actions

- $x$ represents some action, e.g.,
  - trades in a portfolio
  - airplane control surface deflections
  - schedule or assignment
  - resource allocation
  - transmitted signal

- constraints limit actions or impose conditions on outcome
- the smaller the objective $f_0(x)$, the better
  - total cost (or negative profit)
  - deviation from desired or target outcome
  - fuel use
  - risk
Engineering design

- $x$ represents a design (of a circuit, device, structure, ...)
- constraints come from
  - manufacturing process
  - performance requirements
- objective $f_0(x)$ is combination of cost, weight, power, ...
Finding good models

- $x$ represents the *parameters* in a model
- constraints impose requirements on model parameters (e.g., nonnegativity)
- objective $f_0(x)$ is the prediction error on some observed data (and possibly a term that penalizes model complexity)
Inversion

- $x$ is something we want to estimate/reconstruct, given some measurement $y$
- constraints come from prior knowledge about $x$
- objective $f_0(x)$ measures deviation between predicted and actual measurements
Worst-case analysis (pessimization)

- variables are actions or parameters out of our control (and possibly under the control of an adversary)
- constraints limit the possible values of the parameters
- minimizing $-f_0(x)$ finds worst possible parameter values
- if the worst possible value of $f_0(x)$ is tolerable, you’re OK
- it’s good to know what the worst possible scenario can be
Optimization-based models

- model an entity as taking actions that solve an optimization problem
  - an individual makes choices that maximize expected utility
  - an organism acts to maximize its reproductive success
  - reaction rates in a cell maximize growth
  - currents in an electric circuit minimize total power
Optimization-based models

- model an entity as taking actions that solve an optimization problem
  - an individual makes choices that maximize expected utility
  - an organism acts to maximize its reproductive success
  - reaction rates in a cell maximize growth
  - currents in an electric circuit minimize total power

- (except the last) these are very *crude* models
- and yet, they often work very well
Summary

- **summary**: optimization arises *everywhere*
Summary

- **summary**: optimization arises *everywhere*

- **the bad news**: most optimization problems are *intractable* i.e., *we cannot solve them*
Summary

- **summary**: optimization arises everywhere

- **the bad news**: most optimization problems are *intractable* i.e., we cannot solve them

- **an exception**: convex optimization problems are *tractable* i.e., we (generally) *can* solve them
Convex optimization problem

convex optimization problem:

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

- variable \( x \in \mathbb{R}^n \)
- equality constraints are linear
- \( f_0, \ldots, f_m \) are convex: for \( \theta \in [0, 1] \),

\[
\begin{align*}
  f_i(\theta x + (1 - \theta)y) &\leq \theta f_i(x) + (1 - \theta)f_i(y) \\
\end{align*}
\]

i.e., \( f_i \) have nonnegative (upward) curvature
Why

- beautiful, nearly complete theory
  - duality, optimality conditions, ...
Why

- beautiful, nearly complete theory
  - duality, optimality conditions, ... 

- effective algorithms, methods (in theory and practice)
  - get **global solution** (and optimality certificate)
  - polynomial complexity
Why

- beautiful, nearly complete theory
  - duality, optimality conditions, . . .

- effective algorithms, methods (in theory and practice)
  - get **global solution** (and optimality certificate)
  - polynomial complexity

- conceptual unification of many methods
Why

- beautiful, nearly complete theory
  - duality, optimality conditions, . . .

- effective algorithms, methods (in theory and practice)
  - get **global solution** (and optimality certificate)
  - polynomial complexity

- conceptual unification of many methods

- **lots of applications** (many more than previously thought)
Application areas

- machine learning, statistics
- finance
- supply chain, revenue management, advertising
- control
- signal and image processing, vision
- networking
- circuit design
- combinatorial optimization
- quantum mechanics
- flux-based analysis
The approach

▶ try to formulate your optimization problem as convex
▶ if you succeed, you can (usually) solve it (numerically)

▶ using generic software if your problem is not really big
▶ by developing your own software otherwise
The approach

▶ try to formulate your optimization problem as convex
▶ if you succeed, you can (usually) solve it (numerically)
  ▶ using generic software if your problem is not really big
  ▶ by developing your own software otherwise

▶ some tricks:
  ▶ change of variables
  ▶ approximation of true objective, constraints
  ▶ relaxation: ignore terms or constraints you can’t handle
Outline

Mathematical Optimization

Convex Optimization

Solvers & Modeling Languages

Examples

Summary
Medium-scale solvers

- 1000s–10000s variables, constraints
- reliably solved by interior-point methods on single machine (especially for problems in standard cone form)
- exploit problem sparsity
- not quite a technology, but getting there
- used in control, finance, engineering design, ...
Large-scale solvers

- 100k – 1B variables, constraints
- solved using custom (often problem specific) methods
  - limited memory BFGS
  - stochastic subgradient
  - block coordinate descent
  - operator splitting methods
- require custom implementation, tuning for each problem
- used in machine learning, image processing, . . .
Modeling languages

▶ (new) high level language support for convex optimization
  ▶ describe problem in high level language
  ▶ description automatically transformed to a standard form
  ▶ solved by standard solver, transformed back to original form

▶ implementations:
  ▶ YALMIP, CVX (Matlab)
  ▶ CVXPY (Python)
  ▶ Convex.jl (Julia)
  ▶ CVXR (R)
CVX

(Grant & Boyd, 2005)

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)
▶ after cvx_end
    ▶ problem is converted to standard form and solved
    ▶ variable x is over-written with (numerical) solution
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)
solve method converts problem to standard form, solves, assigns value attributes
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)
- solve! method converts problem to standard form, solves, assigns value attributes
Modeling languages

- enable rapid prototyping (for small and medium problems)
- ideal for teaching (can do a lot with short scripts)

- slower than custom methods, but often not much
- current work focuses on extension to large problems
Outline

Mathematical Optimization

Convex Optimization

Solvers & Modeling Languages

Examples

Summary
Radiation treatment planning

- radiation beams with intensities $x_j \geq 0$ directed at patient
- radiation dose $y_i$ received in voxel $i$
- $y = Ax$
- $A \in \mathbb{R}^{m \times n}$ comes from beam geometry, physics
- goal is to choose $x$ to deliver prescribed radiation dose $d_i$
  - $d_i = 0$ for non-tumor voxels
  - $d_i > 0$ for tumor voxels
- $y = d$ not possible, so we’ll need to compromise
- typical problem has $n = 10^3$ beams, $m = 10^6$ voxels
**Radiation treatment planning via convex optimization**

\[
\text{minimize} \quad \sum_i f_i(y_i) \\
\text{subject to} \quad x \geq 0, \quad y = Ax
\]

- variables \( x \in \mathbb{R}^n, \ y \in \mathbb{R}^m \)
- objective terms are
  \[
  f_i(y_i) = w_i^{\text{over}} (y_i - d_i)^+ + w_i^{\text{under}} (d_i - y_i)^+
  \]
  - \( w_i^{\text{over}} \) and \( w_i^{\text{under}} \) are positive weights
  - \( i.e., \) we charge linearly for over- and under-dosing
  - a convex problem
Example

▶ real patient case with $n = 360$ beams, $m = 360000$ voxels
▶ optimization-based plan essentially the same as plan used
Example

- real patient case with $n = 360$ beams, $m = 360000$ voxels
- optimization-based plan essentially the same as plan used
  - but we computed the plan in a few seconds on a GPU
  - original plan took hours of least-squares weight tweaking
Image in-painting

- guess pixel values in obscured/corrupted parts of image
- total variation in-painting: choose pixel values $x_{ij} \in \mathbb{R}^3$ to minimize total variation

$$TV(x) = \sum_{i,j} \left\| \begin{bmatrix} x_{i+1,j} - x_{ij} \\ x_{i,j+1} - x_{ij} \end{bmatrix} \right\|_2$$

- a convex problem
Example

512 × 512 grayscale image \( (n \approx 300000 \text{ variables}) \)

Original Image

Corrupted Image

This is the Loki test image. Can you recover the original image using convex optimization? This problem is called inpainting.
Example

In-Painted Image

Difference Image

This is the Loki test image. Can you recover the original image using convex optimization? This problem is called inpainting.
Example

512 $\times$ 512 color image ($n \approx 800000$), 80% of pixels removed

Original Image

Corrupted Image
Example

80% of pixels removed

In-Painted Image

Difference Image
Control

\[ \begin{align*}
\text{minimize} & \quad \sum_{t=0}^{T-1} \ell(x_t, u_t) + \ell_T(x_T) \\
\text{subject to} & \quad x_{t+1} = Ax_t + Bu_t \\
& \quad (x_t, u_t) \in C, \quad x_T \in C_T
\end{align*} \]

- variables are
  - system states \( x_1, \ldots, x_T \in \mathbb{R}^n \)
  - inputs or actions \( u_0, \ldots, u_{T-1} \in \mathbb{R}^m \)
- \( \ell \) is stage cost, \( \ell_T \) is terminal cost
- \( C \) is state/action constraints; \( C_T \) is terminal constraint
- convex problem when costs, constraints are convex
- applications in many fields
Example

- \( n = 8 \) states, \( m = 2 \) inputs, horizon \( T = 50 \)
- randomly chosen \( A, B \) (with \( A \approx I \))
- input constraint \( \| u_t \|_\infty \leq 1 \)
- terminal constraint \( x_T = 0 \) (‘regulator’)
- \( \ell(x, u) = \| x \|_2^2 + \| u \|_2^2 \) (traditional)
- random initial state \( x_0 \)
Example
Support vector machine classifier with $\ell_1$-regularization

- given data $(x_i, y_i)$, $i = 1, \ldots, m$
  - $x_i \in \mathbb{R}^n$ are feature vectors
  - $y \in \{-1, 1\}$ are associated boolean outcomes
- linear classifier $\hat{y} = \text{sign}(\beta^T x - v)$
- find parameters $\beta, v$ by minimizing (convex function)

\[
\frac{1}{m} \sum_i \left(1 - y_i(\beta^T x_i - v)\right)_+ + \lambda \|\beta\|_1
\]

- first term is average hinge loss
- second term shrinks coefficients in $\beta$ and encourages sparsity
- $\lambda \geq 0$ is (regularization) parameter
- simultaneously selects features and fits classifier
Example

- $n = 20$ features
- trained and tested on $m = 1000$ examples each
Example

$\beta_i$ vs. $\lambda$ (regularization path)
Summary

- convex optimization problems arise in many applications

- convex optimization problems can be solved effectively
  - using generic methods for not huge problems
  - by developing custom methods for huge problems

- high level language support (CVX/CVXPY/Convex.jl/CVXR) makes prototyping easy
many researchers have worked on the topics covered

- **Convex Optimization** (book)
- **EE364a** (course slides, videos, code, homework, . . .)
- software **CVX**, **CVXPY**, **Convex.jl**, **CVXR**

all available online