Convex Optimization: from Real-Time Embedded to Large-Scale Distributed

Stephen Boyd
Neal Parikh, Eric Chu, Yang Wang, Jacob Mattingley

Electrical Engineering Department, Stanford University

Springer Lectures, UC Berkeley, 3/30/2015
Outline

Convex Optimization

Real-Time Embedded Optimization

Large-Scale Distributed Optimization

Summary
Outline

Convex Optimization

Real-Time Embedded Optimization

Large-Scale Distributed Optimization

Summary
Convex optimization — Classical form

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

- variable \( x \in \mathbb{R}^n \)
- \( f_0, \ldots, f_m \) are **convex**: for \( \theta \in [0, 1] \),
  \[
  f_i(\theta x + (1 - \theta)y) \leq \theta f_i(x) + (1 - \theta)f_i(y)
  \]
  i.e., \( f_i \) have nonnegative (upward) curvature
Convex optimization — Cone form

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

- variable \( x \in \mathbb{R}^n \)
- \( K \subseteq \mathbb{R}^n \) is a proper cone
  - \( K \) nonnegative orthant \( \rightarrow \) LP
  - \( K \) Lorentz cone \( \rightarrow \) SOCP
  - \( K \) positive semidefinite matrices \( \rightarrow \) SDP
- the ‘modern’ canonical form
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
Applications — Machine learning

- parameter estimation for regression and classification
  - least squares, lasso regression
  - logistic, SVM classifiers
  - ML and MAP estimation for exponential families

- modern $\ell_1$ and other sparsifying regularizers
  - compressed sensing, total variation reconstruction

- $k$-means, EM, auto-encoders (bi-convex)
Example — Support vector machine

- data \((a_i, b_i), i = 1, \ldots, m\)
  - \(a_i \in \mathbb{R}^n\) feature vectors; \(b_i \in \{-1, 1\}\) Boolean outcomes
- prediction: \(\hat{b} = \text{sign}(w^T a - v)\)
  - \(w \in \mathbb{R}^n\) is weight vector; \(v \in \mathbb{R}\) is offset
Example — Support vector machine

- data \((a_i, b_i), \ i = 1, \ldots, m\)
  - \(a_i \in \mathbb{R}^n\) feature vectors; \(b_i \in \{-1, 1\}\) Boolean outcomes
- prediction: \(\hat{b} = \text{sign}(w^T a - v)\)
  - \(w \in \mathbb{R}^n\) is weight vector; \(v \in \mathbb{R}\) is offset

- SVM: choose \(w, v\) via (convex) optimization problem

\[
\text{minimize} \quad L + \frac{\lambda}{2} \|w\|_2^2
\]

\[
L = \frac{1}{m} \sum_{i=1}^m \left(1 - b_i(w^T a_i - v)\right)_+ \quad \text{is avg. loss}
\]
SVM

\[ w^T z - v = 0 \text{ (solid); } \quad |w^T z - v| = 1 \text{ (dashed)} \]
Sparsity via $\ell_1$ regularization

- adding $\ell_1$-norm regularization

$$\lambda \| x \|_1 = \lambda (|x_1| + |x_2| + \cdots + |x_n|)$$

...to objective results in \textbf{sparse} $x$

- $\lambda > 0$ controls trade-off of sparsity versus main objective

- \textbf{preserves convexity, hence tractability}

- used for many years, in many fields
  - sparse design
  - feature selection in machine learning (lasso, SVM, \ldots)
  - total variation reconstruction in signal processing
  - compressed sensing
Example — Lasso

- regression problem with $\ell_1$ regularization:

$$\text{minimize} \quad \frac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_1$$

with $A \in \mathbb{R}^{m \times n}$

- useful even when $n \gg m$ (!!); does **feature selection**
Example — Lasso

- regression problem with $\ell_1$ regularization:

$$\text{minimize } (1/2)\|Ax - b\|_2^2 + \lambda\|x\|_1$$

with $A \in \mathbb{R}^{m \times n}$

- useful even when $n \gg m$ (!!); does feature selection

- cf. $\ell_2$ regularization (‘ridge regression’):

$$\text{minimize } (1/2)\|Ax - b\|_2^2 + \lambda\|x\|_2^2$$
Example — Lasso

- regression problem with $\ell_1$ regularization:

$$\text{minimize } (1/2) \|Ax - b\|_2^2 + \lambda \|x\|_1$$

with $A \in \mathbb{R}^{m \times n}$

- useful even when $n \gg m$ (!!); does **feature selection**

- cf. $\ell_2$ regularization (‘ridge regression’):

$$\text{minimize } (1/2) \|Ax - b\|_2^2 + \lambda \|x\|_2^2$$

- lasso, ridge regression have **same computational cost**
Example — Lasso

- $m = 200$ examples, $n = 1000$ features
- examples are noisy linear measurements of true $x$
- true $x$ is sparse (30 nonzeros)
Example — Lasso

true \( x \)  
\[ \ell_1 \text{ (lasso) reconstruction} \]
State of the art — Medium scale solvers

- 1000s–10000s variables, constraints
- reliably solved by interior-point methods on single machine
- exploit problem sparsity
- not quite a technology, but getting there
State of the art — Modeling languages

- (new) high level language support for convex optimization
  - describe problem in high level language
  - description is automatically transformed to cone problem
  - solved by standard solver, transformed back to original form
State of the art — Modeling languages

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

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

- parser/solver written in Matlab (M. Grant, 2005)
- SVM:

\[
\text{minimize } \quad L + \left( \frac{\lambda}{2} \right) \| w \|^2_2
\]

\[
L = \left( \frac{1}{m} \right) \sum_{i=1}^{m} \left( 1 - b_i(w^T a_i - v) \right)_+ \quad \text{is avg. loss}
\]

- CVX specification:

```matlab
cvx_begin
variables w(n) v  % weight, offset
L = (1/m)*sum(pos(1-b.*(A*w-v)));  % avg. loss
minimize (L+(lambda/2)*sum_square(w))

cvx_end
```
Outline

Convex Optimization

Real-Time Embedded Optimization

Large-Scale Distributed Optimization

Summary
Motivation

- in many applications, need to solve the same problem repeatedly with different data
  - control: update actions as sensor signals, goals change
  - finance: rebalance portfolio as prices, predictions change
- used now when solve times are measured in minutes, hours
  - supply chain, chemical process control, trading
Motivation

- in many applications, need to solve the same problem repeatedly with different data
  - control: update actions as sensor signals, goals change
  - finance: rebalance portfolio as prices, predictions change
- used now when solve times are measured in minutes, hours
  - supply chain, chemical process control, trading
- (using new techniques) can be used for applications with solve times measured in milliseconds or microseconds
Example — Disk head positioning

- force $F(t)$ moves disk head/arm modeled as 3 masses (2 vibration modes)
- goal: move head to commanded position as quickly as possible, with $|F(t)| \leq 1$
- reduces to a (quasi-) convex problem
Optimal force profile

position

force $F(t)$
Embedded solvers — Requirements

▶ high speed
  ▶ hard real-time execution limits

▶ extreme reliability and robustness
  ▶ no floating point exceptions
  ▶ must handle poor quality data

▶ small footprint
  ▶ no complex libraries
Embedded solvers

- (if a general solver works, use it)
Embedded solvers

- (if a general solver works, use it)
- otherwise, develop custom code
  - by hand
  - automatically via code generation

- can exploit known sparsity pattern, data ranges, required tolerance at solver code development time
Embedded solvers

▸ (if a general solver works, use it)

▸ otherwise, develop custom code
  ▸ by hand
  ▸ automatically via code generation

▸ can exploit known sparsity pattern, data ranges, required tolerance at solver code development time

▸ typical speed-up over general solver: \(100-10000\times\)
Parser/solver vs. code generator

Problem instance  \[\xrightarrow{\text{Parser/solver}}\]  \(x^*\)
Parser/solver vs. code generator

Problem instance \rightarrow \text{Parser/solver} \rightarrow x^*

Problem family description \rightarrow \text{Generator} \rightarrow \text{Source code} \rightarrow \text{Compiler} \rightarrow \text{Custom solver}

\begin{align*}
\text{Problem instance} & \quad \rightarrow \text{Custom solver} \\
& \quad \rightarrow x^*
\end{align*}
CVXGEN code generator

- handles small, medium size problems transformable to QP (J. Mattingley, 2010)
- uses primal-dual interior-point method
- generates flat library-free C source
CVXGEN example specification — SVM

dimensions
m = 50 \% training examples
n = 10 \% dimensions
end
parameters
a[i] (n), i = 1..m \% features
b[i], i = 1..m \% outcomes
lambda positive
end
variables
w (n) \% weights
v \% offset
end
minimize
\( \frac{1}{m} \sum_{i=1}^{m} \text{pos}(1 - b[i]*(w'*a[i] - v)) + \frac{\lambda}{2} \text{quad}(w) \)
end
### CVXGEN sample solve times

<table>
<thead>
<tr>
<th>Problem</th>
<th>SVM</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>61</td>
<td>590</td>
</tr>
<tr>
<td>Constraints</td>
<td>100</td>
<td>742</td>
</tr>
<tr>
<td>CVX, Intel i3</td>
<td>270 ms</td>
<td>2100 ms</td>
</tr>
<tr>
<td>CVXGEN, Intel i3</td>
<td>230 μs</td>
<td>4.8 ms</td>
</tr>
</tbody>
</table>
Outline

Convex Optimization

Real-Time Embedded Optimization

Large-Scale Distributed Optimization

Summary
Motivation and goal

motivation:

- want to solve arbitrary-scale optimization problems
  - machine learning/statistics with huge datasets
  - dynamic optimization on large-scale networks
Motivation and goal

motivation:

- want to solve arbitrary-scale optimization problems
  - machine learning/statistics with huge datasets
  - dynamic optimization on large-scale networks

goal:

- ideally, a system that
  - has CVX-like interface
  - targets modern large-scale computing platforms
  - scales arbitrarily

...not there yet, but there’s promising progress
Distributed optimization

- devices/processors/agents coordinate to solve large problem, by passing relatively small messages
- can split variables, constraints, objective terms among processors
- variables that appear in more than one processor called ‘complicating variables’ (same for constraints, objective terms)
Example — Distributed optimization

minimize \quad f_1(x_1, x_2) + f_2(x_2, x_3) + f_3(x_1, x_3)

\[ f_1 \quad f_2 \quad f_3 \]

\[ x_1 \quad x_2 \quad x_3 \]
Distributed optimization methods

- dual decomposition (Dantzig-Wolfe, 1950s–)
- subgradient consensus
  (Tsitsiklis, Bertsekas, Nedić, Ozdaglar, Jadbabaie, 1980s–)
Distributed optimization methods

- dual decomposition (Dantzig-Wolfe, 1950s–)
- subgradient consensus
  (Tsitsiklis, Bertsekas, Nedić, Ozdaglar, Jadbabaie, 1980s–)

- alternating direction method of multipliers (1980s–)
  - equivalent to many other methods
    (e.g., Douglas-Rachford splitting)
  - well suited to modern systems and problems
Consensus optimization

- want to solve problem with $N$ objective terms

$$\text{minimize } \sum_{i=1}^{N} f_i(x)$$

e.g., $f_i$ is the loss function for $i$th block of training data

- consensus form:

$$\text{minimize } \sum_{i=1}^{N} f_i(x_i)$$
$$\text{subject to } x_i - z = 0$$

- $x_i$ are local variables
- $z$ is the global variable
- $x_i - z = 0$ are consistency or consensus constraints
Consensus optimization via ADMM

with \( \bar{x}^k = (1/N) \sum_{i=1}^{N} x_i^k \) (average over local variables)

\[
\begin{align*}
    x_i^{k+1} &:= \arg\min_{x_i} \left( f_i(x_i) + \left( \frac{\rho}{2} \right) \|x_i - \bar{x}^k + u_i^k\|_2^2 \right) \\
    u_i^{k+1} &:= u_i^k + (x_i^{k+1} - \bar{x}^{k+1})
\end{align*}
\]

- get **global** minimum, under very general conditions
- \( u^k \) is running sum of inconsistencies (PI control)
- minimizations carried out independently and in parallel
- coordination is via averaging of local variables \( x_i \)
Statistical interpretation

- $f_i$ is negative log-likelihood (loss) for parameter $x$ given $i$th data block

- $x_i^{k+1}$ is MAP estimate under prior $\mathcal{N}(\bar{x}^k - u_i^k, \rho I)$

- processors only need to support a Gaussian MAP method
  - type or number of data in each block not relevant
  - consensus protocol yields global ML estimate

- privacy preserving: agents never reveal data to each other
Example — Consensus SVM

- baby problem with $n = 2$, $m = 400$ to illustrate
- examples split into $N = 20$ groups, in worst possible way: each group contains only positive or negative examples
Iteration 1
Iteration 5
Iteration 40
Example — Distributed lasso

- example with dense $A \in \mathbb{R}^{400000 \times 8000}$ ($\sim$30 GB of data)
  - distributed solver written in C using MPI and GSL
  - no optimization or tuned libraries (like ATLAS, MKL)
  - split into 80 subsystems across 10 (8-core) machines on Amazon EC2

- computation times
  
<table>
<thead>
<tr>
<th>Task</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>loading data</td>
<td>30s</td>
</tr>
<tr>
<td>factorization (5000 × 8000 matrices)</td>
<td>5m</td>
</tr>
<tr>
<td>subsequent ADMM iterations</td>
<td>0.5–2s</td>
</tr>
<tr>
<td>total time (about 15 ADMM iterations)</td>
<td>5–6m</td>
</tr>
</tbody>
</table>
Outline

Convex Optimization

Real-Time Embedded Optimization

Large-Scale Distributed Optimization

Summary
Summary

convex optimization problems

- arise in many applications

- can be solved effectively
  - small problems at microsecond/millisecond time scales
  - medium-scale problems using general purpose methods
  - arbitrary-scale problems using distributed optimization
References

- *Convex Optimization* (Boyd & Vandenberghe)

- *CVX: Matlab software for disciplined convex programming* (Grant & Boyd)

- *CVXGEN: A code generator for embedded convex optimization* (Mattingley & Boyd)

- *Distributed optimization and statistical learning via the alternating direction method of multipliers* (Boyd, Parikh, Chu, Peleato, & Eckstein)

all available (with code) from stanford.edu/~boyd