Introduction of CVX

Xu Shen and Qianghuai Jia

University of Science and Technology of China

July 26, 2013

Outline

1. About CVX

- CVX is an open source MATLAB-based modeling tool.
- CVX supports a number of standard problem types, including LPs/QPs, SOCPs, and SDPs.
- CVX can also solve much more complex convex optimization problems, including many involving nondifferentiable functions, such as L_1 norms.

About CVX

- Core solvers used in CVX:
  1. SeDuMi (http://sedumi.ie.lehigh.edu/)
  2. SDPT3 (http://www.math.nus.edu.sg/ mattohkc/sdpt3.html)
- Both are open-source interior-point solvers based on MATLAB.
- CVX converts the problem into a format accepted by those solvers and call them to solve the problem.

The history and development of CVX

Version 1.0

It was issued on June 9, 2006. In this version of cvx, SeDuMi is the only core solver supported. This version of cvx can only handle functions that can be represented using LPs, SOCPs and SDPs, or problems that can be reduced to these problem forms.

Version 1.21

It was issued in August, 2010. This version of cvx added a new solver SDPT3, which is the default. This version added some functions which could not be expressed accurately by LP, SOCP or SDP, such as log, exp, entropy and so on.

Version 2.0

Since the end of 2012, Version 2.0 has several test versions. Firstly, it has supported Gurobi and MOSEK. Secondly, it now supports mixed integer disciplined convex programs (MIDCPs).

What CVX is not

- CVX is not meant to be a tool for checking if your problem is convex.
- CVX is not meant for very large problems, so if your problem is very large, CVX is unlikely to work well.

2. Disciplined Convex Programming (DCP)

- DCP is a methodology for constructing convex optimization problems in a suitable format for CVX.
- DCP imposes a set of rules.
- Problems has to be written such that those rules are satisfied. Otherwise, problem will be rejected, even when the problem is convex.

A Quick Start

- Here I will take least-squares for the example. In a least-squares problem, we seek $x \in R^n$ that minimizes $\|Ax - b\|_2$.
- Using CVX, the problem can be sovlved as follows:

```
CVX_begin
    variable x(n)
    minimize (norm(Ax-b))
CVX_end
```

1. *CVX_begin* creates a placeholder for the new CVX specification, and prepares Matlab to accept variable declarations, constraints, an objective function, and so forth.
2. variable x(n) declares x to be an optimization variable of dimension n. CVX requires that all problem variables be declared before they are used in the objective function or constraints.
3. minimize(norm(Ax-b)) specifies the objective function to be minimized.
4. *CVX_end* signals the end of the CVX specification, and causes the problem to be solved.

3. Introduction to SDP

- Why semidefinite programming?
  SDP is solvable via interior-point methods and usually requires about the same amount of computational resources as linear optimization.

- Facts about the Semidefinite Cone
  1. If X is an $n \times n$ matrix, then X is a symmetric positive semidefinite (SPSD) matrix if $X = X^T$ and $v^T X v \geq 0 \quad for \quad any \quad v \in \Re^n$.
  2. $C \bullet X := \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$
  3. Consider the matrix M defined as follows:

$$M = \begin{pmatrix} P & v \\ v^T & d \end{pmatrix}$$

Then $M \geq 0$ if and only if $d - v^T P^{-1} v \geq 0$ and $P \geq 0$.

Semidefinite Programming

- A semidefinite program (SDP) is an optimization problem of the form:

$$SDP: \quad minimize_X \quad C \bullet X$$

$$s.t. \quad A_i \bullet X = b_i, i = 1, 2, ..., m$$

$$X \geq 0$$

The variable X must lie in the (closed convex) cone of positive semidefinite symmetric matrices S_+^n.

- The dual form of SDP is:

$$\max_y \quad b^T y$$

$$s.t. \quad C - \sum_{i=1}^{m} y_i A_i \geq 0$$

General convex problems to SDP

- LP to SDP
- QCQP to SDP
- SOCP to SDP
- ℓ_1 norm to SDP
- nuclear norm to SDP

LP to SDP

- The standard-form LP problem is:

$$\begin{aligned} \min_{x} \quad & c^T x \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

- when C and A_i X are all diagonal matrices, i.e., $C = diag\{c\}, A_i = diag\{a_i\}, X = diag\{x\}$

  LP is transformed to SDP:

$$SDP: \quad minimize_X \quad C \bullet X$$

$$s.t. \quad A_i \bullet X = b_i, i = 1, 2, ..., m$$

$$X \geq 0$$

QCQP to SDP

- The standard-form QCQP problem is:

$$\begin{aligned} \min_{x} \quad & x^T H x \\ s.t. \quad & x^T Q x + q^T x + r \leq 0 \\ & x \geq 0 \end{aligned}$$

- we can first introduce a variable δ:

$$\begin{aligned} \min_{x,\delta} \quad & \delta \\ s.t. \quad & x^T H x \leq \delta \\ & x^T Q x + q^T x + r \leq 0 \\ & x \geq 0 \end{aligned}$$

QCQP to SDP

- The above QCQP problem can be converted into the SDP problem with:

$$\hat{c} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \hat{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \delta \end{pmatrix} \quad and \quad E(\hat{x}) = diag\{G(\delta, x), F(x)\}$$

$$G(\delta, x) = \begin{pmatrix} I_4 & \hat{H}x \\ (\hat{H}x)^T & \delta \end{pmatrix}, F(x) = \begin{pmatrix} I_2 & \hat{Q}x \\ (\hat{Q}x)^T & -r - q^T x \end{pmatrix}$$

where $H = \hat{H}^T \hat{H}, Q = \hat{Q}^T \hat{Q}$

- Then the QCQP is converted to the following SDP:

$$\min_{x,\delta} \quad \delta \quad s.t. \quad E(\hat{x}) \geq 0$$

QCQP to SDP

- The above SDP problem is equivalent to the standard SDP problem:

$$b = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{pmatrix}, y = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \delta \end{pmatrix}$$

- Matrices A_i for $1 \leq i \leq 5$ and C are given by

$$A_i = -diag\{\begin{pmatrix} \mathbf{0_4} & h_i \\ h_i^T & 0 \end{pmatrix}, \begin{pmatrix} \mathbf{0_2} & q_i \\ q_i^T & -q(i) \end{pmatrix}\}, 1 \leq i \leq 4$$

$$A_5 = -diag\{\begin{pmatrix} \mathbf{0_4} & 0 \\ \mathbf{0} & 1 \end{pmatrix}, \mathbf{0_3}\}, \mathbf{C} = -\mathbf{diag}\{\begin{pmatrix} \mathbf{I_4} & \mathbf{0} \\ \mathbf{0} & 0 \end{pmatrix}, \begin{pmatrix} \mathbf{I_2} & \mathbf{0} \\ \mathbf{0} & -r \end{pmatrix}\}$$

SOCP to SDP

- A second-order cone optimization problem (SOCP) is an optimization problem of the form:

$$SOCP : \min_x c^T x$$
$$s.t. \quad Ax = b$$
$$||Q_i x + d_i|| \leq (g_i^T x + h_i), i = 1, ..., k.$$

$$Property : ||Qx+d|| \leq (g^T x+h) \iff \begin{pmatrix} (g^T x + h)I & (Qx + d) \\ (Qx + d)^T & g^T x + h \end{pmatrix} \geq 0$$

- Therefore we can rewrite the SOCP to SDP:

$$SDP : \min_x \quad c^T x$$
$$s.t. \quad Ax = b; \quad \begin{pmatrix} (g_i^T x + h)I & (Q_i x + d) \\ (Q_i x + d)^T & g_i^T x + h \end{pmatrix} \geq 0, i = 1, ..., k.$$

ℓ_1 norm to SDP

- Consider the ℓ_1 norm minimization problem:

$$\min_x \quad ||Ax - c||_1$$

where $A \in C^{mn}, c \in C^{m1}, x \in C^{n1}$, this problem can be solved as a LP:

$$\begin{aligned} LP: minimize \quad & \mathbf{1^T y} \\ s.t. \quad & -y \leq Ax - c \leq y, \end{aligned}$$

which can be converted to a standard SDP.

nuclear norm to SDP

- Consider the nuclear norm minimization problem:

$$\min_{x} \quad ||A(x) - B||_*$$

where $A(x) = x_1A_1 + x_2A_2 + ... + x_nA_n$ and $B \in \Re^{p\times q}$ is a given matrix.

- This problem can be cast as a semidefinite program (SDP):

$$SDP: \; minimize \quad (trU + trV)/2$$
$$s.t. \quad \begin{pmatrix} U & (A(x) - B)^T \\ A(x) - B & V \end{pmatrix} \geq 0,$$

with variables $x \in \Re^n$, $U \in S^q$, $V \in S^p$. (We use S^n to denote the set of symmetric matrices of order n.)

4. Interior Point Algorithms for SDPs

- The standard primal and dual SDPS are

$$(P) \qquad\qquad (D)$$

$$\min_X \ C \bullet X \qquad\qquad \max_y \ b^T y$$

$$s.t.\ A_i \bullet X = b_i, i = 1, ..., m \qquad\qquad s.t.\ C - \sum_{i=1}^{m} y_i A_i \succeq 0$$

$$X = X^T \succeq 0$$

- Their penalized barrier problems of parameter $\nu > 0$ are

$$BP(\nu) \qquad\qquad BD(\nu)$$

$$\min_X \ C \bullet X - \nu logdetX \qquad\qquad \max_{y,S} \ b^T y + \nu logdetS$$

$$s.t.\ A(X) = b(X \succ 0) \qquad\qquad s.t.\ A^*(y) + S = C(S \succ 0)$$

Interior Point Algorithms for SDPs

- The central path is

$$CP(\nu)$$

$$A(X) = b$$

$$A^*(y) + S = C$$

$$XS = \nu I, (X, S \succ 0).$$

- As $\nu \to 0$, the solution $(X(\nu), y(\nu), S(\nu))$ to $CP(\nu)$ convergence to the primal-dual optimal solution (X^*, y^*, S^*) of (P) and (D). Path-following methods are in the type of interior-point algorithms that track points near central path as $\nu \to 0$.

Primal-Dual Path-Following Interior-Point methods

- The optimal conditions for (P) and (D) are

$$A(X) = b,$$

$$A^*(y) + S = C,$$

$$XS = \nu I,$$

$$X = X^T, S = S^T \succ 0.$$

- The basic idea is to apply damped Newton's method to solve the above equation for a given $\nu > 0$, decrease ν, and then repeat. The only trouble is that when we linearize the last equation $XS = \nu I$, the resulting direction $\triangle X$, $\triangle S$ might not be symmetric. So an overcoming technique is to replace the last equation by its symmetrization

$$XS + SX = 2\nu I$$

Primal-Dual Path-Following Interior-Point Algorithm

- Choose a strictly feasible X_0 for (P) and (y_0, S_0) for (D), $\nu > 0$, $\tau \in (0,1)$, $\theta \in (0,1)$. Set k=0.
  1. Step 1 Compute Newton's Direction $(\triangle X, \triangle y, \triangle S)$ from

$$A(\triangle X) = b - A(X_k),$$

$$A^*(\triangle y) + \triangle S = C - A^*(y_k) - S_k,$$

$$\triangle XS + S\triangle X + X\triangle S + \triangle SX = 2\nu I - (X_k S_k - S_k X_k).$$

  2. Step 2 Compute the step-length

$$\alpha_k = max\alpha \in (0,1] : X_k + \alpha\triangle X \succ 0, S_k + \alpha\triangle S \succ 0$$

  3. Step 3 Update $S_{k+1} = S_k + \alpha_k \triangle S$ and $y_{k+1} = y_k + \alpha_k \triangle y$.
  4. Step 4 Set k := k + 1. If the following holds,

$$\|XS - (X_k \bullet S_k/n)I\|_F > \tau X_k \bullet S_k/n$$

     then go to step 1. Otherwise, go to the next step.
  5. Step 5 If $\nu > \epsilon$, update $\nu = \theta\nu$ and go to Step 1. Otherwise stop.

How cvx works

We introduce the process of how cvx tool box works based on the following example:

$$\min_{x_1,x_2,x_3} \quad -4x_1 - x_2 - 3x_3$$

$$x_1 + 4x_2 \leq 1$$

$$3x_1 - x_2 + x_3 \leq 3$$

$$x_1, x_2, x_3 \geq 0$$

```
A = [1 4 0 ;3 -1 1]; b = [1 3]'; c = [-4
-1 -3 ]';
cvx_begin sdp
cvx_solver sedumi;
variable x(3,1);
minimize(c'*x);
subject to
A * x ≤ b;
x ≥ 0;
cvx_end
```

read in and store the cvx problem

1. initialize the **cvx** struct.
2. *function* $z = mtimes(x, y, oper)$
   return $z = x * y$ or $z = x.*y$ based on 'oper'.

How cvx works

Extract the whole cvx problem

- function [dbcA, cones, Q, P, ineqs] = extract(pp,doineqs)
  To store all the data in matrix **dbcA** and convex set in **cones**.

$$dbcA = \begin{pmatrix} 0 & 1 & 3 & 0 & 0 & 0 \\ -4 & -1 & -3 & 1 & 0 & 0 \\ -1 & -4 & 1 & 0 & 1 & 0 \\ -3 & 0 & -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

cones.indices=[5,6,7,8,9] which is the index of slack variables.

$$dbcA^T * [1, x_1, x_2, x_3, s_1, ..., s_5]^T = 0$$

How cvx works

Extract the whole cvx problem

- function [dbCA, cones,Q, P] = eliminate(prob, can_dual) provides a smaller [d, b ; c, A]problem with no more nonzeros.

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -4 & 1 & 3 \\ -1 & 4 & -1 \\ -3 & 0 & 1 \end{pmatrix} \qquad Q = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$x = Q * [1; xx], y = P * [1; yy]$$

$$dbcA \quad is \quad converted \quad to \quad A$$

How cvx works

change to standard SDP

$$A = \begin{pmatrix} 0 & 1 & 3 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ -4 & -1 & -3 \\ -1 & -4 & 1 \\ -3 & 0 & -1 \end{pmatrix}$$

- function solve(prob)
  extract At,c,and b from A.

$$At = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -3 \\ -4 & 1 \\ 0 & -1 \end{pmatrix} \qquad c = \begin{pmatrix} 0 \\ 0 \\ -4 \\ -1 \\ -3 \end{pmatrix} \qquad b = \begin{pmatrix} -1 \\ -3 \end{pmatrix}$$

How cvx works

use sedumi to solve the problem

- function[x,y,info] = sedumi(A,b,c,K,pars)
  The main algorithm is a PREDICTOR-CORRECTOR loop
  1. Compute ADA.
  2. Block Sparse Cholesky: ADA(L.perm,L.perm) = L.L*diag(L.d)*L.L'.
  3. Factorization of self-dual embedding.
  4. Compute and take IPM-step.
  5. Primal-Dual transformation.
  6. optimality check.
  7. Interpret the solution as feasible

References


CVX Official Website
http://cvxr.com/cvx/


Interior Point Algorithms fpor SDP
http://www.math.ucsd.edu/ njw/Teaching/Math271C/Lecture_09.pdf


Duality Theorey in SDP
http://www.math.ucsd.edu/ njw/Teaching/Math271C/Lecture_06.pdf


Interior-point method for nuclear norm approximation with application to system identification
http://www.seas.ucla.edu/ vandenbe/publications/nucnrm.pdf


Introduction to Semidefinite Programming (SDP)
http://www-personal.umich.edu/ mepelman/teaching/IOE511/Handouts/511notes07-16.pdf


semidefinite and second-order cone programming
http://shmathsoc.org.cn/lu/core%20part/Chap14.pdf


Use SeDuMi to Solve LP, SDP and SCOP Problems: Remarks and Examples
http://sms.math.ecnu.edu.cn/lu/supplementary%20part/SeDuMi-Remarks.pdf

Thank you