Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding

B. O'Donoghue E. Chu N. Parikh **S. Boyd**

Convex Optimization and Beyond, Edinburgh, 11/6/2104

Outline

Cone programming

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

- variables $x \in \mathbf{R}^n$ and (slack) $s \in \mathbf{R}^m$
- $\mathcal{K}$ is a proper convex cone
  - $\mathcal{K}$ nonnegative orthant $\longrightarrow$ LP
  - $\mathcal{K}$ Lorentz cone $\longrightarrow$ SOCP
  - $\mathcal{K}$ positive semidefinite matrices $\longrightarrow$ SDP

self-dual cone

- the 'modern' canonical form for convex optimization
- popularized by Nesterov, Nemirovsky, others, in 1990s

Cone programming

Step 1 : modeling ① mathematical convex problems ⇒ ②

- parser/solvers like CVX, CVXPY, YALMIP translate or canonicalize to cone problems ②
- focus has been on symmetric self-dual cones 4
- for medium scale problems with enough sparsity, interior-point methods reliably attain high accuracy
- but they scale superlinearly in problem size
- open source software (SDPT3, SeDuMi, . . .) widely used

Step 2 : solving interior-point method based
(ADMM, SCS)

This talk

a new first order method that

- solves general cone programs
- finds primal and dual solutions, or certificate of primal/dual infeasibility
  ① KKT conditions
  ② homogeneous self-dual embedding (HSD)
- obtains modest accuracy quickly
- scales to large problems and is easy parallelized
- is matrix-free: only requires $z \to Az$, $w \to A^T w$

(first-order method, e.g., ADMM)

Some previous work

- projected subgradient type methods (Polyak 1980s)
- primal-dual subgradient methods (Chambelle-Pock 2011)
- matrix-free interior-point methods (Gondzio 2012)
- can use iterative linear solver (CG) in any interior-point method

Outline

Cone programming

Homogeneous embedding

Operator splitting

Numerical results

Conclusions

Primal-dual cone problem pair

primal and dual cone problems:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax + s = b \\ & (x, s) \in \mathbf{R}^n \times \mathcal{K} \end{array} \quad \overset{A}{\Longrightarrow} \quad \begin{array}{ll} \text{maximize} & -b^T y \\ \text{subject to} & -A^T y + r = c \\ & (r, y) \in \{0\}^n \times \mathcal{K}^* \end{array}$$

- primal variables $x \in \mathbf{R}^n$, $s \in \mathbf{R}^m$; dual variables $r \in \mathbf{R}^n$, $y \in \mathbf{R}^m$
- $\mathcal{K}^\star$ is dual of closed convex proper cone $\mathcal{K}$
- note that $\mathbf{R}^n \times \mathcal{K}$ and $\{0\}^n \times \mathcal{K}^*$ are dual cones

(self-dual cones)

Example cones

$\mathcal{K}$ is typically a Cartesian product of smaller cones, *e.g.*,

- $\mathbf{R}$, $\{0\}$, $\mathbf{R}_+$
- second-order cone $\mathcal{Q} = \{(x,t) \in \mathbf{R}^{k+1} \mid \|x\|_2 \leq t\}$
- positive semidefinite cone $\{X \in \mathbf{S}^k \mid X \succeq 0\}$
- exponential cone $\mathbf{cl}\{(x,y,z) \in \mathbf{R}^3 \mid y > 0,\ e^{x/y} \leq z/y\}$

?

these cones would handle almost all convex problems that arise in applications

Optimality conditions

KKT conditions (necessary and sufficient, assuming strong duality):

- primal feasibility: $Ax + s = b, \quad s \in \mathcal{K}$
- dual feasibility: $A^T y + c = r, \quad r = 0, \quad y \in \mathcal{K}^*$
- complementary slackness: $y^T s = 0$
  equivalent to zero duality gap: $\underbrace{c^T x + b^T y = 0}_{\text{strong duality}}$

Primal-dual embedding

- KKT conditions as feasibility problem: find

① $$(x, s, r, y) \in \mathbf{R}^n \times \mathcal{K} \times \{0\}^n \times \mathcal{K}^*$$

that satisfy

② $$\begin{bmatrix} r \\ s \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & A^T \\ -A & 0 \\ c^T & b^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ b \\ 0 \end{bmatrix}$$ affine set

- reduces solving cone program to finding point in intersection of cone and affine set
- no solution if primal or dual problem infeasible/unbounded

Homogeneous self-dual (HSD) embedding

(Ye, Todd, Mizuno, 1994)

- find **nonzero**

$$(x, s, r, y) \in \mathbf{R}^n \times \mathcal{K} \times \{0\}^n \times \mathcal{K}^*, \quad \tau \geq 0, \quad \kappa \geq 0$$

that satisfy

$$\begin{bmatrix} r \\ s \\ \kappa \end{bmatrix} = \begin{bmatrix} 0 & A^T & c \\ -A & 0 & b \\ -c^T & -b^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \tau \end{bmatrix}$$

- this feasibility problem is homogeneous and self-dual
- $\tau = 1, \kappa = 0$ reduces to primal-dual embedding
- due to skew symmetry, any solution satisfies

$$(x, y, \tau) \perp (r, s, \kappa), \qquad \tau\kappa = 0$$

Recovering solution or certificates

any HSD solution $(x, s, r, y, \tau, \kappa)$ falls into one of three cases:

1. $\tau > 0$, $\kappa = 0$: $(\hat{x}, \hat{y}, \hat{s}) = (x/\tau, y/\tau, s/\tau)$ is a solution
2. $\tau = 0$, $\kappa > 0$: in this case $c^T x + b^T y < 0$
   - if $b^T y < 0$, then $\hat{y} = y/(-b^T y)$ certifies primal infeasibility
   - if $c^T x < 0$, then $\hat{x} = x/(-c^T x)$ certifies dual infeasibility
3. $\tau = \kappa = 0$: nothing can be said about original problem (a pathology)

Homogeneous primal-dual embedding

check infeasibility

solving

HSD embedding

- obviates need for phase I / phase II solves to handle infeasibility/unboundedness
- is used in all interior-point cone solvers
- is a particularly nice form to solve (for reasons not completely understood)

Notation

- define

$$u = \begin{bmatrix} x \\ y \\ \tau \end{bmatrix}, \quad v = \begin{bmatrix} r \\ s \\ \kappa \end{bmatrix}, \quad Q = \begin{bmatrix} 0 & A^T & c \\ -A & 0 & b \\ -c^T & -b^T & 0 \end{bmatrix}$$

- HSD embedding is: find (u, v) that satisfy

$$v = Qu, \qquad (u, v) \in \mathcal{C} \times \mathcal{C}^*$$

with $\mathcal{C} = \mathbf{R}^n \times \mathcal{K}^* \times \mathbf{R}_+$

find u, v
s.t. $v = Qu$
$(u, v) \in \mathcal{C} \times \mathcal{C}^*$

Outline

Cone programming

Homogeneous embedding

Operator splitting

Numerical results

Conclusions

Consensus problem

- consensus problem:

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & x = z \end{array}$$

- f, g convex, not necessarily smooth, can take infinite values
- $p^\star$ is optimal objective value

Alternating direction method of multipliers

- ADMM is: for $k = 0, \ldots,$

$$
\begin{aligned}
x^{k+1} &= \operatorname*{argmin}_x \left(f(x) + (\rho/2)\|x - z^k - \lambda^k\|_2^2 \right) \\
z^{k+1} &= \operatorname*{argmin}_z \left(g(z) + (\rho/2)\|x^{k+1} - z - \lambda^k\|_2^2 \right) \\
\lambda^{k+1} &= \lambda^k - x^{k+1} + z^{k+1}
\end{aligned}
$$

- $\rho > 0$ step-size
- λ (scaled) dual variable for $x = z$ constraint
- same as many other operator splitting methods for consensus problem, *e.g.*, Douglas-Rachford method

Convergence of ADMM

under benign conditions ADMM guarantees:

- $f(x^k) + g(z^k) \to p^\star$
- $\lambda^k \to \lambda^\star$, an optimal dual variable
- $x^k - z^k \to 0$

ADMM applied to HSD embedding

minimize constant
subject to $v = Qu$
$(u, v) \in \mathcal{C} \times \mathcal{C}^*$
//

- HSD in consensus form

$$
\begin{array}{ll}
\text{minimize} & I_{\mathcal{C}\times\mathcal{C}^*}(u, v) + I_{Q\tilde{u}=\tilde{v}}(\tilde{u}, \tilde{v}) \\
\text{subject to} & (u, v) = (\tilde{u}, \tilde{v})
\end{array}
$$

2 3 1

$I_{\mathcal{S}}$ is indicator function of set $\mathcal{S}$

- ADMM is:

$$
\begin{array}{rcl}
(\tilde{u}^{k+1}, \tilde{v}^{k+1}) & = & \Pi_{Qu=v}(u^k + \lambda^k, v^k + \mu^k) \\
u^{k+1} & = & \Pi_{\mathcal{C}}(\tilde{u}^{k+1} - \lambda^k) \\
v^{k+1} & = & \Pi_{\mathcal{C}^*}(\tilde{v}^{k+1} - \mu^k) \\
\lambda^{k+1} & = & \lambda^k - \tilde{u}^{k+1} + u^{k+1} \\
\mu^{k+1} & = & \mu^k - \tilde{v}^{k+1} + v^{k+1}
\end{array}
$$

1

2 $\mathcal{C}\times\mathcal{C}^*$

3

dual update

$\Pi_{\mathcal{S}}(x)$ is Euclidean projection of x onto $\mathcal{S}$

Simplifications

(straightforward, but not immediate)

- if $\lambda^0 = v^0$ and $\mu^0 = u^0$, then 1 $\lambda^k = v^k$ and 2 $\mu^k = u^k$ for all k
- simplify projection onto $Qu = v$ using $\boxed{Q^T = -Q}$
- nothing depends on $\tilde{v}^k$, so can be eliminated

Final algorithm

- for $k = 0, \ldots,$

$$\begin{aligned} \tilde{u}^{k+1} &= (I+Q)^{-1}(u^k + v^k) \\ u^{k+1} &= \Pi_{\mathcal{C}}\left(\tilde{u}^{k+1} - v^k\right) \\ v^{k+1} &= v^k - \tilde{u}^{k+1} + u^{k+1} \end{aligned}$$

subspace projection

parallel cone projection

$(C = C_1 \times C_2 \times \cdots \times C_n)$

- parameter free
- homogeneous
- same complexity as ADMM applied to primal or dual alone

Variation: Approximate projection

- replace exact projection with any $\tilde{u}^{k+1}$ that satisfies

$$\|\tilde{u}^{k+1} - (I + Q)^{-1}(u^k + v^k)\|_2 \leq \mu^k,$$

  where $\mu^k > 0$ satisfy $\sum_k \mu_k < \infty$
- useful when an iterative method is used to compute $\tilde{u}^{k+1}$
- implied by the (more easily verified) inequality

$$\|(Q + I)\tilde{u}^{k+1} - (u^k + v^k)\|_2 \leq \mu^k$$

  by skew-symmetry of Q

Convergence

can show the following (even with approximate projection):

- for all iterations $k > 0$ we have

$$u^k \in \mathcal{C}, \quad v^k \in \mathcal{C}^*, \quad (u^k)^T v^k = 0$$

- as $k \to \infty$,

$$Qu^k - v^k \to 0$$

- with $\tau^0 = 1$, $\kappa^0 = 1$, (u^k, v^k) bounded away from zero

Solving the linear system

- in first step need to solve equations

$$\underbrace{\begin{bmatrix} I & A^T & c \\ -A & I & b \\ -c^T & -b^T & 1 \end{bmatrix}}_{Q} \begin{bmatrix} \tilde{u}_x \\ \tilde{u}_y \\ \tilde{u}_\tau \end{bmatrix} = \begin{bmatrix} w_x \\ w_y \\ w_\tau \end{bmatrix}$$

- let

$$M = \begin{bmatrix} I & A^T \\ -A & I \end{bmatrix}, \quad h = \begin{bmatrix} c \\ b \end{bmatrix}$$

so

$$I + Q = \begin{bmatrix} M & h \\ -h^T & 1 \end{bmatrix}$$

- it follows that

$$\begin{bmatrix} \tilde{u}_x \\ \tilde{u}_y \end{bmatrix} = (M + hh^T)^{-1} \left(\begin{bmatrix} w_x \\ w_y \end{bmatrix} - w_\tau h \right),$$

Solving the linear system, contd.

- applying matrix inversion lemma to $(M + hh^T)^{-1}$ yields

$$\begin{bmatrix} \tilde{u}_x \\ \tilde{u}_y \end{bmatrix} = \left(M^{-1} - \frac{M^{-1}hh^T M^{-1}}{(1 + h^T M^{-1} h)} \right) \left(\begin{bmatrix} w_x \\ w_y \end{bmatrix} - w_\tau h \right)$$

  and

$$\tilde{u}_\tau = w_\tau + c^T \tilde{u}_x + b^T \tilde{u}_y$$

- first compute and cache $M^{-1}h$
- so each iteration requires that we compute

$$M^{-1} \begin{bmatrix} w_x \\ w_y \end{bmatrix}$$

  and perform vector operations with cached quantities

Direct method

- to solve

$$\begin{bmatrix} I & -A^T \\ -A & -I \end{bmatrix} \begin{bmatrix} z_x \\ -z_y \end{bmatrix} = \begin{bmatrix} w_x \\ w_y \end{bmatrix}$$

- compute *sparse permuted LDL factorization* of matrix
- re-use cached factorization for subsequent solves
- factorization guaranteed to exist for all permutations, since matrix is symmetric *quasi-definite*

Indirect method

- by elimination

$$z_x = (I + A^T A)^{-1}(w_x - A^T w_y)$$
$$z_y = w_y + A z_x$$

- can apply *conjugate gradient (CG)* to first equation
- CG requires only multiplies by A and A^T
- terminate CG iterations when residual smaller than μ^k
- easily parallelized; can exploit warm-starting

Scaling / preconditioning

convergence greatly improved by scaling / preconditioning:

- replace original data A, b, c with $\hat{A} = DAE$, $\hat{b} = Db$, $\hat{c} = Ec$
- D and E are diagonal positive; D respects cone boundaries
- D and E chosen by equilibrating A (details in paper)
- stopping condition retains unscaled (original) data

Outline

SCS software package

- available from:

  `https://github.com/cvxgrp/scs`

- written in C with matlab and python hooks
- can be called from CVX and CVXPY
- solves LPs, SOCPs, ECPs, and SDPs
- includes sparse direct and indirect linear system solvers
- can use single or double precision, ints or longs for indices

Portfolio optimization

- $z \in \mathbf{R}^p$ gives weights of (long-only) portfolio with p assets
- maximize risk-adjusted portfolio return:

return

$$\begin{array}{ll} \text{maximize} & \mu^T z - \gamma(z^T \Sigma z) \\ \text{subject to} & \mathbf{1}^T z = 1, \quad z \geq 0 \end{array}$$

risk

- μ, Σ are return mean, covariance
- $\gamma > 0$ is risk aversion parameter
- Σ given as factor model $\Sigma = FF^T + D$
- $F \in \mathbf{R}^{q \times p}$ is factor loading matrix
- can be transformed to SOCP

Portfolio optimization results

assets p	5000	50000	100000
factors q	50	500	1000
SOCP variables n	5002	50002	100002
SOCP constraints m	10055	100505	201005
nonzeros in A	3.8×10^4	2.5×10^6	1.0×10^7
SDPT3: (interior-point)			
solve time	**1.14 sec**	**17836.7 sec** !	**OOM**
SCS direct: (ADMM)			
solve time	**0.17 sec**	**4.7 sec**	**37.1 sec**
iterations	420	340	760
SCS indirect:			
solve time	**0.23 sec**	**12.2 sec**	**101 sec**
average CG iterations	1.62	1.39	1.82
iterations	400	400	800

ℓ_1-regularized logistic regression

- fit logistic model, with ℓ_1 regularization
- data $z_i \in \mathbf{R}^p$, $i = 1, \ldots, q$ with labels $y_i \in \{-1, 1\}$
- solve

$$\text{minimize} \quad \sum_{i=1}^{q} \log(1 + \exp(y_i w^T z_i)) + \mu \|w\|_1$$

over variable $w \in \mathbf{R}^p$; $\mu > 0$ regularization parameter

- can be transformed to exponential cone program (ECP)

ℓ_1-regularized logistic regression results

	small	**medium**	**large**
features p	600	2000	6000
samples q	3000	10000	30000
ECP variables n	10200	34000	102000
ECP constraints m	22200	74000	222000
nonzeros in A	1.9×10^5	1.9×10^6	1.7×10^7
SCS direct:			
solve time	**22.1 sec**	**165 sec**	**1020 sec**
iterations	280	660	1240
SCS indirect:			
solve time	**24.0 sec**	**199 sec**	**1290 sec**
average CG iterations	2.00	2.49	2.82
iterations	300	760	1320

Large random SOCP

- randomly generated SOCP with known optimal value
- $n = 1.6 \times 10^6$ variables, $m = 4.8 \times 10^6$ constraints
- 2×10^9 nonzeros in A, 22.5Gb memory to store
- indirect solver, tolerance 10^{-3}, parallelized over 32 threads
- results:
  - 740 SCS iterations, about 5000 matrix multiplies
  - 10 hours wall-clock time
  - $|c^T x - c^T x^\star| / |c^T x^\star| = 7 \times 10^{-4}$
  - $|b^T y - b^T y^\star| / |b^T y^\star| = 1 \times 10^{-3}$

Outline

Conclusions

- HSD embedding is great for first-order methods
- diagonal preconditioning critical
- matrix-free algorithm: only $z \to Az$, $w \to A^T w$
- SCS is now standard large scale solver in CVXPY