Scalable Semidefinite Programming

Madeleine Udell

Operations Research and Information Engineering
Cornell University

Based on joint work with
Lijun Ding (Cornell), Alp Yurtsever (MIT),
Olivier Fercoq (Télécom ParisTech), Volkan Cevher (EPFL),
and Joel Tropp (Caltech)

ORIE 7391, April 2022

Quiz

1. What is the memory required to solve an underconstrained SDP with $n \times n$ decision variable $X \in \mathbf{S}_+^n$ and $O(n)$ linear constraints, assuming the rank of the solution $\operatorname{rank}(X^\star) \leq 5$?
   A. $O(\log n)$
   B. $O(n)$
   C. $O(n \log(n))$
   D. $O(n^2)$
   E. $O(n^6)$
2. There exist optimization algorithms that can be run without explicit access to the optimization variable. (e.g., with access only to a linear image of that variable.)
   A. true
   B. false

Outline

Motivation

Large scale SDP

Complementary slackness

SketchyCGM

SketchyCGAL

*

Goal

Can we develop algorithms that provably solve a problem using *storage* controlled by the size of the *problem data* and the size of the *solution*?

Problem data: $\mathcal{O}(n)$

$\downarrow$

Working memory: $\mathcal{O}(???)$

$\downarrow$

Solution: $\mathcal{O}(n)$

Target problem: semidefinite program (SDP)

consider primal SDP with decision variable $X \in \mathbf{S}^n_+$:

$$\begin{array}{ll} \text{minimize} & \mathrm{tr}(CX) \\ \text{subject to} & \mathcal{A}X = b \\ & \mathrm{tr}\, X \le \alpha, \quad X \succeq 0 \end{array} \tag{$\mathcal{P}$}$$

problem data:

- cost matrix $C \in \mathbf{S}^n$
- linear map $\mathcal{A} : \mathbf{S}^n \to \mathbf{R}^m$
- righthand side $b \in \mathbf{R}^m$

Are desiderata achievable?

suppose $(\mathcal{P})$ has

- *compact specification*: problem data use $\mathcal{O}(n)$ storage
- *compact solution*: solution $X_\star$ has constant rank $r_\star$
  $\implies$ solution uses $\mathcal{O}(n)$ storage

Are desiderata achievable?

suppose $(\mathcal{P})$ has

- *compact specification*: problem data use $\mathcal{O}(n)$ storage
- *compact solution*: solution $X_\star$ has constant rank $r_\star$
  $\implies$ solution uses $\mathcal{O}(n)$ storage

$(\mathcal{P})$, using any first order method:

$$\boxed{\text{Problem data: } \mathcal{O}(n)}$$
$$\downarrow$$
$$\boxed{\text{Working memory: } \mathcal{O}(n^2)}$$
$$\downarrow$$
$$\boxed{\text{Solution: } \mathcal{O}(n)}$$

Are desiderata achievable?

$(\mathcal{P})$, using methods from this talk:

Problem data: $\mathcal{O}(n)$

$\downarrow$

Working memory: $\mathcal{O}(n)$

$\downarrow$

Solution: $\mathcal{O}(n)$

Outline

Motivation

Large scale SDP

Complementary slackness

SketchyCGM

SketchyCGAL

*

Large scale SDP

Max-Cut [Goemans and Williamson 1995]	Matrix Completion [Srebro and Shraibman 2005]
$\begin{array}{ll} \text{minimize} & \mathrm{tr}(-LX) \\ \text{subject to} & \mathbf{diag}(X) = \mathbf{1} \\ & X \succeq 0 \end{array}$	$\begin{array}{ll} \text{minimize} & \mathrm{tr}(W_1) + \mathrm{tr}(W_2) \\ \text{subject to} & X_{ij} = \bar{X}_{ij},\ (i,j) \in \Omega \\ & \begin{bmatrix} W_1 & X \\ X^* & W_2 \end{bmatrix} \succeq 0 \end{array}$

Large scale SDP

Max-Cut [Goemans and Williamson 1995]	**Matrix Completion** [Srebro and Shraibman 2005]
$\begin{array}{ll} \text{minimize} & \mathrm{tr}(-LX) \\ \text{subject to} & \mathbf{diag}(X) = \mathbf{1} \\ & X \succeq 0 \end{array}$	$\begin{array}{ll} \text{minimize} & \mathrm{tr}(W_1) + \mathrm{tr}(W_2) \\ \text{subject to} & X_{ij} = \bar{X}_{ij},\ (i,j) \in \Omega \\ & \begin{bmatrix} W_1 & X \\ X^* & W_2 \end{bmatrix} \succeq 0 \end{array}$

- Matrix completion:
  - 10^9 users, 10^9 products
  - $\implies$ SDP with 10^{18} variables

Large scale SDP

Max-Cut [Goemans and Williamson 1995]	Matrix Completion [Srebro and Shraibman 2005]
$\begin{array}{ll} \text{minimize} & \mathbf{tr}(-LX) \\ \text{subject to} & \mathbf{diag}(X) = \mathbf{1} \\ & X \succeq 0 \end{array}$	$\begin{array}{ll} \text{minimize} & \mathbf{tr}(W_1) + \mathbf{tr}(W_2) \\ \text{subject to} & X_{ij} = \bar{X}_{ij},\ (i,j) \in \Omega \\ & \begin{bmatrix} W_1 & X \\ X^* & W_2 \end{bmatrix} \succeq 0 \end{array}$

- Matrix completion:
  - 10^9 users, 10^9 products
  - $\implies$ SDP with 10^{18} variables
- MaxCut:
  - 10^9 people in social network
  - $\implies$ SDP with 10^{18} variables

Large scale SDP

Max-Cut [Goemans and Williamson 1995]	Matrix Completion [Srebro and Shraibman 2005]
$\begin{array}{ll} \text{minimize} & \mathrm{tr}(-LX) \\ \text{subject to} & \mathbf{diag}(X) = \mathbf{1} \\ & X \succeq 0 \end{array}$	$\begin{array}{ll} \text{minimize} & \mathrm{tr}(W_1) + \mathrm{tr}(W_2) \\ \text{subject to} & X_{ij} = \bar{X}_{ij},\ (i,j) \in \Omega \\ & \begin{bmatrix} W_1 & X \\ X^* & W_2 \end{bmatrix} \succeq 0 \end{array}$

- Matrix completion:
  - 10^9 users, 10^9 products
  - $\implies$ SDP with 10^{18} variables
- MaxCut:
  - 10^9 people in social network
  - $\implies$ SDP with 10^{18} variables
- Phase retrieval:
  - $10^3 \times 10^3$ discretization of sample
  - $\implies$ SDP with 10^{12} variables

Why compact?

why a $\mathcal{O}(n)$ specification?

- data is expensive
- collect constant data per column (=user or sample)
- if solution is compact, compact specification should suffice

why a $\mathcal{O}(n)$ solution?

- the world is simple and structured
- nice latent variable models are of log rank
- given d observations, there is a solution with rank $\mathcal{O}(\sqrt{d})$
  Barvinok 1995, Pataki 1998

Optimal Storage

What kind of storage bounds can we hope for?

- Assume black-box implementation of primitives

$$v \mapsto Cv, \qquad v \mapsto \mathcal{A}(vv^*), \text{and} \qquad (v, y) \mapsto (\mathcal{A}^* y)v,$$

where $v \in \mathbf{R}^n$, $y \in \mathbf{R}^m$.

Optimal Storage

What kind of storage bounds can we hope for?

- Assume black-box implementation of primitives

$$v \mapsto Cv, \qquad v \mapsto \mathcal{A}(vv^*), \text{and} \qquad (v, y) \mapsto (\mathcal{A}^* y)v,$$

  where $v \in \mathbf{R}^n$, $y \in \mathbf{R}^m$.
- Need $\Omega(n)$ storage to apply problem data.

Optimal Storage

What kind of storage bounds can we hope for?

- Assume black-box implementation of primitives

$$v \mapsto Cv, \qquad v \mapsto \mathcal{A}(vv^*), \text{and} \qquad (v, y) \mapsto (\mathcal{A}^* y)v,$$

  where $v \in \mathbf{R}^n$, $y \in \mathbf{R}^m$.
- Need $\Omega(n)$ storage to apply problem data.
- Need $\Theta(rn)$ storage for a rank-r approximate solution.

Optimal Storage

What kind of storage bounds can we hope for?

- Assume black-box implementation of primitives

$$v \mapsto Cv, \qquad v \mapsto \mathcal{A}(vv^*), \text{and} \qquad (v, y) \mapsto (\mathcal{A}^* y)v,$$

  where $v \in \mathbf{R}^n$, $y \in \mathbf{R}^m$.
- Need $\Omega(n)$ storage to apply problem data.
- Need $\Theta(rn)$ storage for a rank-r approximate solution.

Definition. An algorithm to return a rank r (approximate) solution to $(\mathcal{P})$ has **optimal storage** if it uses working storage

$$\Theta(rn).$$

Related work: convex methods

- **Interior point methods**: storage $\Theta(n^2 + m^2)$
  - [Nesterov and Nemirovski 1989, Nesterov and Nemirovskii 1994, Alizadeh 1991; 1995]; ...

Related work: convex methods

- **Interior point methods**: storage $\Theta(n^2 + m^2)$
  - [Nesterov and Nemirovski 1989, Nesterov and Nemirovskii 1994, Alizadeh 1991; 1995]; ...
- **Primal-dual first order methods**: storage $\Theta(n^2 + m)$
  - [Rockafellar 1970, Auslender and Teboulle 2006, O'Donoghue et al. 2016, Yurtsever et al. 2018]; ...

Related work: convex methods

- **Interior point methods**: storage $\Theta(n^2 + m^2)$
  - [Nesterov and Nemirovski 1989, Nesterov and Nemirovskii 1994, Alizadeh 1991; 1995]; ...
- **Primal-dual first order methods**: storage $\Theta(n^2 + m)$
  - [Rockafellar 1970, Auslender and Teboulle 2006, O'Donoghue et al. 2016, Yurtsever et al. 2018]; ...
- **Storage-efficient first order methods**: storage $\mathcal{O}(nk)$ for k iters
  - multiplicative weights [Arora et al. 2005]; CGM [Hazan 2008, Jaggi 2013]; truncated CGM [Rao et al. 2013]; spectral bundle method [Helmberg and Rendl 2000]; spectral low rank optimization [Friedlander and Macedo 2016]; ...

Related work: convex methods

- **Interior point methods**: storage $\Theta(n^2 + m^2)$
  - [Nesterov and Nemirovski 1989, Nesterov and Nemirovskii 1994, Alizadeh 1991; 1995]; ...
- **Primal-dual first order methods**: storage $\Theta(n^2 + m)$
  - [Rockafellar 1970, Auslender and Teboulle 2006, O'Donoghue et al. 2016, Yurtsever et al. 2018]; ...
- **Storage-efficient first order methods**: storage $\mathcal{O}(nk)$ for k iters
  - multiplicative weights [Arora et al. 2005]; CGM [Hazan 2008, Jaggi 2013]; truncated CGM [Rao et al. 2013]; spectral bundle method [Helmberg and Rendl 2000]; spectral low rank optimization [Friedlander and Macedo 2016]; ...
  - control storage or guarantee convergence
  - not both

Related work: non-convex methods

Burer-Monteiro approach: Solve

$$\begin{array}{ll} \text{minimize} & \operatorname{tr}(CFF^*) \\ \text{subject to} & \mathcal{A}(FF^*) = b \end{array} \tag{BM}$$

with variable $F \in \mathbf{R}^{n \times r}$

Related work: non-convex methods

Burer-Monteiro approach: Solve

$$\begin{array}{ll} \text{minimize} & \operatorname{tr}(CFF^*) \\ \text{subject to} & \mathcal{A}(FF^*) = b \end{array} \qquad \text{(BM)}$$

with variable $F \in \mathbf{R}^{n \times r}$

- $(+)$: if
$$\frac{r(r+1)}{2} > m,$$
any second order stationary point is globally optimal [Burer and Monteiro 2003, Boumal et al. 2016]; can solve locally with Riemannian optimization when $\mathcal{A}(FF^*) = b$ is smooth manifold [Absil et al. 2009, Boumal et al. 2014]

Related work: non-convex methods

Burer-Monteiro approach: Solve

$$\begin{array}{ll} \text{minimize} & \operatorname{tr}(CFF^*) \\ \text{subject to} & \mathcal{A}(FF^*) = b \end{array} \tag{BM}$$

with variable $F \in \mathbf{R}^{n \times r}$

- (+): if
$$\frac{r(r+1)}{2} > m,$$
any second order stationary point is globally optimal [Burer and Monteiro 2003, Boumal et al. 2016];
can solve locally with Riemannian optimization when $\mathcal{A}(FF^*) = b$ is smooth manifold [Absil et al. 2009, Boumal et al. 2014]
- (-): [Waldspurger and Waters 2018] show that (for some $\mathcal{A}, b, C$)) (BM) admits bad local minima when
$$\frac{r(r+1)}{2} + r \leq m$$

Related work: non-convex methods

Burer-Monteiro approach: Solve

$$\begin{array}{ll} \text{minimize} & \mathrm{tr}(CFF^*) \\ \text{subject to} & \mathcal{A}(FF^*) = b \end{array} \qquad \text{(BM)}$$

with variable $F \in \mathbf{R}^{n \times r}$

- (+): if
$$\frac{r(r+1)}{2} > m,$$
any second order stationary point is globally optimal [Burer and Monteiro 2003, Boumal et al. 2016]; can solve locally with Riemannian optimization when $\mathcal{A}(FF^*) = b$ is smooth manifold [Absil et al. 2009, Boumal et al. 2014]
- (-): [Waldspurger and Waters 2018] show that (for some $\mathcal{A}, b, C$)) (BM) admits bad local minima when
$$\frac{r(r+1)}{2} + r \leq m$$

$\implies$ BM approach is not storage optimal

Storage optimal methods for SDP

two approaches (so far) to storage-optimal SDP:

- **Complementary slackness.** [Ding et al. 2019]
  1. Use any dual solver to solve dual problem
  2. Insight: optimality conditions identify range of primal
  3. Recover primal by solving smaller SDP
  4. Requires strict complementarity, exact low rank solution

Storage optimal methods for SDP

two approaches (so far) to storage-optimal SDP:

- **Complementary slackness.** [Ding et al. 2019]
  1. Use any dual solver to solve dual problem
  2. Insight: optimality conditions identify range of primal
  3. Recover primal by solving smaller SDP
  4. Requires strict complementarity, exact low rank solution
- **SketchySDP.** [Yurtsever et al. 2017; 2019b]; ...
  1. Use primal-dual solver to solve dual problem
  2. Insight: can sketch primal during iteration
  3. Recover from sketch
  4. Can find low rank approximation to high rank solution

Approximate eigenvectors

Definition

For a symmetric matrix $M \in \mathbf{S}^n_+$, we say a unit vector v is an ϵ-*approximate minimum eigenvector* if

$$v^* M v \le \lambda_{\min}(M) + \epsilon \|M\| .$$

Approximate eigenvectors

Definition

For a symmetric matrix $M \in \mathbf{S}_+^n$, we say a unit vector v is an ϵ-*approximate minimum eigenvector* if

$$v^* M v \leq \lambda_{\min}(M) + \epsilon \|M\| .$$

how to find approximate minimum eigenvector (efficiently)?

- (-) Krylov methods (e.g., ARPACK `eigs`)
  - unstable, hard to control precision
- (+) shifted power method
  - converges in $\mathcal{O}(\epsilon^{-1})$ iterations, needs $\mathcal{O}(n)$ storage
- (+) randomized Lanczos method
  - converges in $q = \mathcal{O}(\epsilon^{-1/2})$ iterations, needs $\mathcal{O}(nq)$ storage

Approximate eigenvectors: shifted power method

- use power iteration to find max eigenvalue
- min eigenvalue of M is max eigenvalue of $\|M\| I - M$

Algorithm ApproxMinEvec via randomized shifted power method

Input: $M \in \mathbf{S}_n$, and maxiters q

Output: Approximate minimum eigenpair $(\xi, v) \in \mathbb{R} \times \mathbb{R}^n$ of M

function $\textsc{ApproxMinEvec}(M; q)$

 $\sigma \leftarrow \|M\|$

 $v \leftarrow \texttt{randn}(n, 1)/\sqrt{n}$

 for $i \leftarrow 1, 2, 3, \ldots, q$ **do**

  $v \leftarrow \sigma v - Mv$

  $v \leftarrow v / \|v\|$

 return $(v^*(Mv), v)$

Power method: guarantees

Fact (Randomized shifted power method (Kuczyński and Woźniakowski 1992))

Let $M \in \mathbf{S}_n$. For $\epsilon \in (0,1]$ and $\delta \in (0,1]$, the shifted power method computes a unit vector $u \in \mathbb{R}^n$ that satisfies

$$u^* M u \le \lambda_{\min}(M) + \epsilon \|M\| \quad w/prob \ge 1 - \delta$$

after $q \ge \frac{1}{2} + \epsilon^{-1} \log(n/\delta^2)$ iterations.[1]

- arithmetic cost is $\mathcal{O}(q)$ matrix–vector multiplies with M and $\mathcal{O}(qn)$ extra operations
- working storage is about $2n$ numbers

[1] All logarithms are base-e.

Outline

Motivation

Large scale SDP

Complementary slackness

SketchyCGM

SketchyCGAL

*

Dual problem

consider dual SDP with decision variable $y \in \mathbf{R}^m$:

$$\begin{array}{ll} \text{maximize} & b^* y \\ \text{subject to} & C - \mathcal{A}^* y \succeq 0 \end{array} \qquad (\mathcal{D})$$

Dual problem

consider dual SDP with decision variable $y \in \mathbf{R}^m$:

$$\begin{array}{ll} \text{maximize} & b^* y \\ \text{subject to} & C - \mathcal{A}^* y \succeq 0 \end{array} \qquad (\mathcal{D})$$

for sufficiently large $\alpha > 0$, equivalent to *penalized dual*

$$\text{maximize} \quad b^* y + \alpha \min(\lambda_{\min}(C - \mathcal{A}^* y), 0)$$

Dual problem

consider dual SDP with decision variable $y \in \mathbf{R}^m$:

$$\begin{array}{ll} \text{maximize} & b^* y \\ \text{subject to} & C - \mathcal{A}^* y \succeq 0 \end{array} \qquad (\mathcal{D})$$

for sufficiently large $\alpha > 0$, equivalent to *penalized dual*

$$\text{maximize} \quad b^* y + \alpha \min(\lambda_{\min}(C - \mathcal{A}^* y), 0)$$

- any first order method for $(\mathcal{D})$ uses optimal storage
- (just compute minimal eigenvalue with iterative method)
- e.g., subgradient method, AdaGrad [Duchi et al. 2011], AdaNGD [Levy 2017], AccelGrad [Levy et al. 2018], ...

Assumptions

Assumption (Regular SDP)

- primal SDP has a unique solution $X_\star$
- dual SDP has a unique solution $y_\star$
- primal and dual SDP satisfy strong duality

$$0 = \mathrm{tr}(CX_\star) - b^* y_\star = \mathrm{tr}(X_\star(C - \mathcal{A}^* y_\star)) = X_\star(C - \mathcal{A}^* y_\star)$$

- (for storage optimality) strict complementary slackness

$$\mathrm{rank}(X_\star) + \mathrm{rank}(C - \mathcal{A}^* y_\star) = n$$

Assumptions

Assumption (Regular SDP)

- primal SDP has a unique solution $X_\star$
- dual SDP has a unique solution $y_\star$
- primal and dual SDP satisfy strong duality

$$0 = \operatorname{tr}(CX_\star) - b^* y_\star = \operatorname{tr}(X_\star(C - \mathcal{A}^* y_\star)) = X_\star(C - \mathcal{A}^* y_\star)$$

- (for storage optimality) strict complementary slackness

$$\operatorname{rank}(X_\star) + \operatorname{rank}(C - \mathcal{A}^* y_\star) = n$$

Note: these conditions hold generically [Alizadeh et al. 1997, Ding and Udell 2020]

Primal recovery via exact complementarity

suppose $y_\star \in \mathbf{R}^m$ solves $(\mathcal{D})$

Primal recovery via exact complementarity

suppose $y_\star \in \mathbf{R}^m$ solves $(\mathcal{D})$

- define dual slack matrix $Z_\star = C - \mathcal{A}^* y_\star$

Primal recovery via exact complementarity

suppose $y_\star \in \mathbf{R}^m$ solves $(\mathcal{D})$

- define dual slack matrix $Z_\star = C - \mathcal{A}^* y_\star$
- strict complementarity holds between $X_\star$ and $Z_\star$

$$\operatorname{rank}(X_\star) + \operatorname{rank}(Z_\star) = n \quad \Longrightarrow \quad \operatorname{range}(X_\star) = \mathbf{null}(Z_\star)$$

Primal recovery via exact complementarity

suppose $y_\star \in \mathbf{R}^m$ solves $(\mathcal{D})$

- define dual slack matrix $Z_\star = C - \mathcal{A}^* y_\star$
- strict complementarity holds between $X_\star$ and $Z_\star$

$$\mathrm{rank}(X_\star) + \mathrm{rank}(Z_\star) = n \quad \Longrightarrow \quad \mathrm{range}(X_\star) = \mathbf{null}(Z_\star)$$

- let $V_\star$ be a basis for nullspace of dual slack matrix $Z_\star$

Primal recovery via exact complementarity

suppose $y_\star \in \mathbf{R}^m$ solves $(\mathcal{D})$

- define dual slack matrix $Z_\star = C - \mathcal{A}^* y_\star$
- strict complementarity holds between $X_\star$ and $Z_\star$

$$\operatorname{rank}(X_\star) + \operatorname{rank}(Z_\star) = n \quad \Longrightarrow \quad \operatorname{range}(X_\star) = \mathbf{null}(Z_\star)$$

- let $V_\star$ be a basis for nullspace of dual slack matrix $Z_\star$
- constrain $X = V_\star S V_\star^*$ in primal SDP for some $S \in \mathbf{S}_\star^{r_\star}$.
  $\Longrightarrow$ solution is preserved!

SDP via exact complementarity: algorithm

Algorithm SDP via exact complementarity

Given: problem data C, $\mathcal{A}$, b

SDP via exact complementarity: algorithm

Algorithm SDP via exact complementarity

Given: problem data C, $\mathcal{A}$, b

1. compute solution $y_\star \in \mathbf{R}^m$ to $(\mathcal{D})$

SDP via exact complementarity: algorithm

Algorithm SDP via exact complementarity

Given: problem data C, $\mathcal{A}$, b

1. compute solution $y_\star \in \mathbf{R}^m$ to $(\mathcal{D})$
2. compute basis $V_\star$ for nullspace of dual slack matrix $Z_\star = C - \mathcal{A}^* y_\star$

SDP via exact complementarity: algorithm

Algorithm SDP via exact complementarity

Given: problem data C, $\mathcal{A}$, b

1. compute solution $y_\star \in \mathbf{R}^m$ to $(\mathcal{D})$
2. compute basis $V_\star$ for nullspace of dual slack matrix $Z_\star = C - \mathcal{A}^* y_\star$
3. solve **reduced primal** $(\mathcal{P}_{V_\star})$ with variable $S \in \mathbf{S}_+^{r_\star}$

$$\begin{array}{ll} \text{minimize} & \mathrm{tr}(C V_\star S V_\star^*) \\ \text{subject to} & \mathcal{A}(V_\star S V_\star^*) = b \\ & S \succeq 0 \end{array} \qquad (\mathcal{P}_{V_\star})$$

to find primal solution $X_\star = V_\star S V_\star^*$

SDP via exact complementarity: algorithm

Algorithm SDP via exact complementarity

Given: problem data C, $\mathcal{A}$, b

1. compute solution $y_\star \in \mathbf{R}^m$ to $(\mathcal{D})$
2. compute basis $V_\star$ for nullspace of dual slack matrix $Z_\star = C - \mathcal{A}^* y_\star$
3. solve **reduced primal** $(\mathcal{P}_{V_\star})$ with variable $S \in \mathbf{S}_+^{r_\star}$

$$\begin{array}{ll} \text{minimize} & \mathrm{tr}(C V_\star S V_\star^*) \\ \text{subject to} & \mathcal{A}(V_\star S V_\star^*) = b \\ & S \succeq 0 \end{array} \qquad (\mathcal{P}_{V_\star})$$

to find primal solution $X_\star = V_\star S V_\star^*$

Reduced primal is easy: e.g., when $r_\star$=1, $(\mathcal{P}_{V_\star})$ solves a 1D problem over $S \in \mathbf{R}_+$

SDP via exact complementarity: storage complexity

SDP via exact complementarity uses optimal storage:

1. compute $y_\star$ with any subgradient method

   ($\mathcal{O}(m+n)$ storage)

SDP via exact complementarity: storage complexity

SDP via exact complementarity uses optimal storage:

1. compute $y_\star$ with any subgradient method
   ($\mathcal{O}(m+n)$ storage)
2. compute $V_\star$ using randomized linear algebra (e.g., [Halko et al. 2011]) ($\Theta(nr_\star)$ storage)

SDP via exact complementarity: storage complexity

SDP via exact complementarity uses optimal storage:

1. compute $y_\star$ with any subgradient method
   ($\mathcal{O}(m+n)$ storage)
2. compute $V_\star$ using randomized linear algebra (e.g., [Halko et al. 2011]) ($\Theta(nr_\star)$ storage)
3. solve $(\mathcal{P}_{V_\star})$ via first order method (e.g., [O'Donoghue et al. 2016]) ($\Theta(m+n+r_\star^2)$ storage)

SDP via exact complementarity: storage complexity

SDP via exact complementarity uses optimal storage:

1. compute $y_\star$ with any subgradient method
   ($\mathcal{O}(m+n)$ storage)
2. compute $V_\star$ using randomized linear algebra (e.g., [Halko et al. 2011]) ($\Theta(nr_\star)$ storage)
3. solve $(\mathcal{P}_{V_\star})$ via first order method (e.g., [O'Donoghue et al. 2016]) ($\Theta(m+n+r_\star^2)$ storage)

$= (\mathcal{O}(m+nr_\star)$ storage

SDP via exact complementarity: storage complexity

SDP via exact complementarity uses optimal storage:

1. compute $y_\star$ with any subgradient method
   ($\mathcal{O}(m+n)$ storage)
2. compute $V_\star$ using randomized linear algebra (e.g., [Halko et al. 2011]) ($\Theta(nr_\star)$ storage)
3. solve $(\mathcal{P}_{V_\star})$ via first order method (e.g., [O'Donoghue et al. 2016]) ($\Theta(m+n+r_\star^2)$ storage)

$= (\mathcal{O}(m+nr_\star)$ storage

(in practice, use an IPM to solve $(\mathcal{P}_{V_\star})$; it's tiny!)

Primal recovery via exact complementarity: picture

Approximate complementarity?

Problem: subgradient methods do not (exactly) compute $y^\star$

Approximate complementarity?

Problem: subgradient methods do not (exactly) compute $y^\star$

try the obvious fix:

Approximate complementarity?

Problem: subgradient methods do not (exactly) compute $y^\star$

try the obvious fix:

- ($y \approx y_\star$): compute approximate dual solution y

Approximate complementarity?

Problem: subgradient methods do not (exactly) compute $y^\star$

try the obvious fix:

- $(y \approx y_\star)$: compute approximate dual solution y
- $(V \approx V_\star)$: compute r-dimensional eigenspace V of $C - \mathcal{A}^* y$ with smallest eigenvalues

Approximate complementarity?

Problem: subgradient methods do not (exactly) compute $y^\star$

try the obvious fix:

- $(y \approx y_\star)$: compute approximate dual solution y
- $(V \approx V_\star)$: compute r-dimensional eigenspace V of $C - \mathcal{A}^* y$ with smallest eigenvalues
- $(X \approx X_\star)$? solve reduced SDP $(\mathcal{P}_V)$ with variable $S \in \mathbf{S}_+^r$

$$\begin{array}{ll} \text{minimize} & \operatorname{tr}(CVSV^*) \\ \text{subject to} & \mathcal{A}(VSV^*) = b \\ & S \succeq 0 \end{array} \qquad (\mathcal{P}_V)$$

to find primal solution $X = VSV^*$

Approximate complementarity?

Problem: subgradient methods do not (exactly) compute $y^\star$

try the obvious fix:

- $(y \approx y_\star)$: compute approximate dual solution y
- $(V \approx V_\star)$: compute r-dimensional eigenspace V of $C - \mathcal{A}^* y$ with smallest eigenvalues
- $(X \approx X_\star)$? solve reduced SDP $(\mathcal{P}_V)$ with variable $S \in \mathbf{S}_+^r$

$$\begin{array}{ll} \text{minimize} & \operatorname{tr}(CVSV^*) \\ \text{subject to} & \mathcal{A}(VSV^*) = b \\ & S \succeq 0 \end{array} \qquad (\mathcal{P}_V)$$

to find primal solution $X = VSV^*$

```
Status:  Infeasible
Optimal value (cvx_optval):  +Inf
```

Primal recovery via approximate complementarity: picture

SDP via approximate complementarity

Algorithm SDP via approximate complementarity

Given: problem data C, $\mathcal{A}$, b; target rank $r \geq r_\star$

SDP via approximate complementarity

Algorithm SDP via approximate complementarity

Given: problem data C, $\mathcal{A}$, b; target rank $r \geq r_\star$

1. Compute approximate dual solution y.

SDP via approximate complementarity

Algorithm SDP via approximate complementarity

Given: problem data C, $\mathcal{A}$, b; target rank $r \geq r_\star$

1. Compute approximate dual solution y.
2. Compute basis V for eigenspace of $C - \mathcal{A}^* y$ with r smallest eigenvalues.

SDP via approximate complementarity

Algorithm SDP via approximate complementarity

Given: problem data C, $\mathcal{A}$, b; target rank $r \geq r_\star$

1. Compute approximate dual solution y.
2. Compute basis V for eigenspace of $C - \mathcal{A}^* y$ with r smallest eigenvalues.
3. To find primal solution $X_\star = V_\star S V_\star^*$, $S \in \mathbf{S}_r^+$, minimize infeasibility

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\left\|\mathcal{A}(VSV^*) - b\right\|^2 \\ \text{subject to} & S \succeq 0 \end{array} \qquad \text{(MinFeasSDP)}$$

SDP via approximate complementarity

Algorithm SDP via approximate complementarity

Given: problem data C, $\mathcal{A}$, b; target rank $r \geq r_\star$

1. Compute approximate dual solution y.
2. Compute basis V for eigenspace of $C - \mathcal{A}^* y$ with r smallest eigenvalues.
3. To find primal solution $X_\star = V_\star S V_\star^*$, $S \in \mathbf{S}_r^+$, minimize infeasibility

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\left\|\mathcal{A}(VSV^*) - b\right\|^2 \\ \text{subject to} & S \succeq 0 \end{array} \qquad \text{(MinFeasSDP)}$$

or accept infeasibility δ and minimize loss

$$\begin{array}{ll} \text{minimize} & \mathrm{tr}(CVSV^*) \\ \text{subject to} & \|\mathcal{A}(VSV^*) - b\| \leq \delta \\ & S \succeq 0. \end{array} \qquad \text{(MinObjSDP)}$$

Exact vs approximate complementarity

Exact complementarity	**Approximate complementarity**
Compute dual solution $y_\star$	Compute approximate dual solution y
Compute basis $V_\star$ for $\textbf{null}(C - \mathcal{A}^* y_\star)$	Compute basis V for eigenspace of $C - \mathcal{A}^* y$ with r smallest eigenvalues
Solve $\mathcal{P}_{V_\star}$	Solve MinFeasSDP or MinObjSDP.

Theoretical guarantees: approximate complementarity

Theorem (Recovery guarantees)

Suppose $(\mathcal{P})$ and $(\mathcal{D})$ satisfy genericity assumptions.
If $r = r_\star$ (MinFeasSDP) or $r \geq r_\star$ (MinObjSDP),
primal recovery via approximate complementarity from an
ϵ-suboptimal dual solution y produces a $\sqrt{\epsilon}$-suboptimal primal solution X:

	MinFeasSDP	MinObjSDP
suboptimality $\mathrm{tr}(CX) - \mathrm{tr}(CX_\star)$	$\mathcal{O}(\kappa\sqrt{\epsilon})$	$\mathcal{O}(\sqrt{\epsilon})$
infeasibility $\|\mathcal{A}X - b\|_2$	$\mathcal{O}(\kappa\sqrt{\epsilon})$	$\mathcal{O}(\sqrt{\epsilon})$
distance to solution $\|X - X_\star\|_F$	$\mathcal{O}(\kappa\sqrt{\epsilon})$	$\mathcal{O}(\epsilon^{\frac{1}{4}})$

where the condition number $\kappa = \frac{\sigma_{\max}(\mathcal{A})}{\sigma_{\min}(\mathcal{A}|_{V_\star})}$.

Core Lemma

Lemma (Projected solution)

Suppose

- $(\mathcal{P})$ and $(\mathcal{D})$ admit solutions and satisfy strong duality;
- $y \in \mathbf{R}^m$ is feasible and ϵ-suboptimal for $(\mathcal{D})$;
- and the threshold $T := \lambda_{n-r}(C - \mathcal{A}^* y) > 0$.

Then for any solution $X_\star$ of the primal SDP $(\mathcal{P})$,

$$\left\| X_\star - V \underbrace{V^* X_\star V}_{\text{feasible } S \text{ for MinFeasSDP}} V^* \right\| \leq \frac{\epsilon}{T} + \sqrt{\frac{2r\epsilon}{T} \|X_\star\|_{\text{op}}}$$

where $\|\cdot\|$ is either the nuclear or Frobenius norm.

A practical algorithm for SDP

how to choose parameters?

A practical algorithm for SDP

how to choose parameters?

- infeasibility tolerance δ
  - solve MinFeasSDP first, then solve MinObjSDP

A practical algorithm for SDP

how to choose parameters?

- infeasibility tolerance δ
  - solve MinFeasSDP first, then solve MinObjSDP
- rank target r
  - bigger is better
  - use spectrum of slack matrix $C - \mathcal{A}^* y$:
    need $T = \lambda_{n-r}(C - \mathcal{A}^* y) > 0$

A practical algorithm for SDP

how to choose parameters?

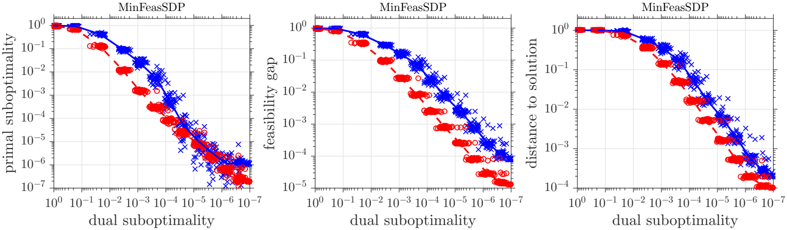
- infeasibility tolerance δ
  - solve MinFeasSDP first, then solve MinObjSDP
- rank target r
  - bigger is better
  - use spectrum of slack matrix $C - \mathcal{A}^* y$:
    need $T = \lambda_{n-r}(C - \mathcal{A}^* y) > 0$

Algorithm Primal recovery via approximate complementarity

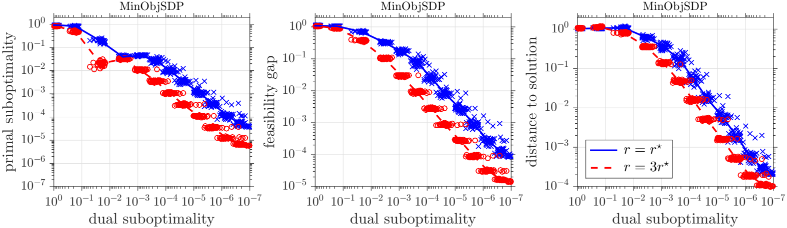
Given: approximate dual solution y, rank target r

- compute basis V for eigenspace of $C - \mathcal{A}^* y$ with r smallest eigenvalues
- solve (MinFeasSDP) to obtain a solution X_{infeas},
- then solve (MinObjSDP) with $\delta = 1.1 \, \|\mathcal{A} X_{\text{infeas}} - b\|_2$

Primal recovery works



Max-Cut



Matrix Completion

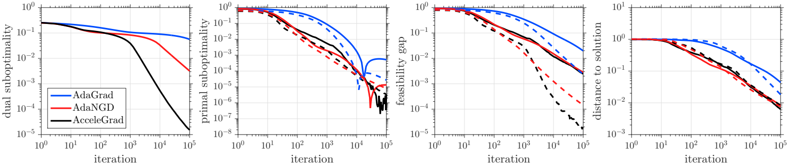
Don't try this at home

Algorithm for plot:

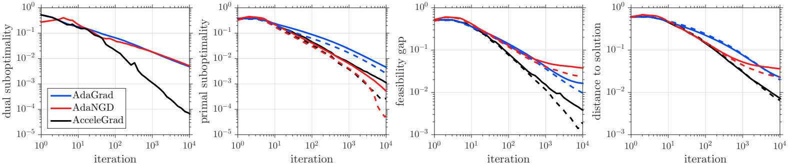
For $k = 1, 2, \ldots,$

- compute k-th dual iterate y_k using dual solver (here, AdaGrad [Duchi et al. 2011], AdaNGD [Levy 2017], and AccelGrad [Levy et al. 2018])
- recover a primal iterate X_k from y_k using RobustPrimalRecovery

Approximate complementarity solves primal SDP



Max-Cut



Matrix Completion

((dashed) $r = r_\star$ (solid) $r = 3r_\star$)

Outline

*

Model problem: low rank matrix optimization

consider a convex problem with decision variable $X \in \mathbf{S}^n_+$

compact matrix optimization problem:

$$\begin{array}{ll} \text{minimize} & f(\mathcal{A}X) \\ \text{subject to} & X \in \alpha\Delta_n \end{array} \qquad \text{(CMOP)}$$

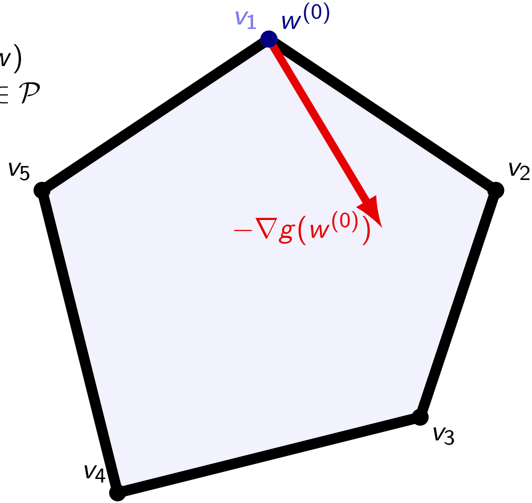
- $\mathcal{A} : \mathbf{S}^n \to \mathbb{R}^m$
- $f : \mathbb{R}^m \to \mathbb{R}$ convex and smooth
- $\Delta_n = \{X : \operatorname{tr} X \leq 1, X \succeq 0\}$

assume

- *compact specification*: problem data use $\mathcal{O}(n)$ storage
- *compact solution*: $\operatorname{rank} X_\star = r$ constant

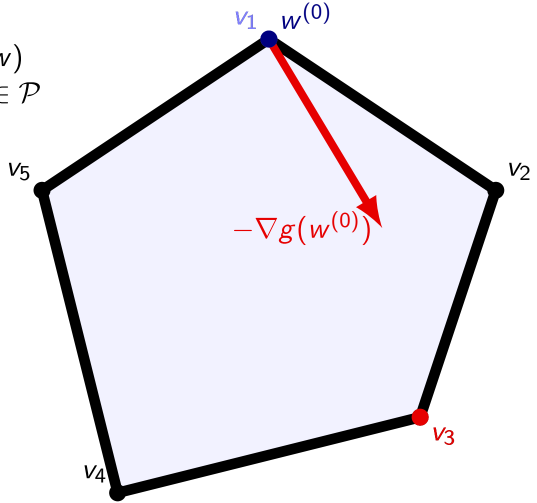
Conditional gradient method (Frank-Wolfe)

$$\begin{array}{ll} \text{minimize} & g(w) \\ \text{subject to} & w \in \mathcal{P} \end{array}$$



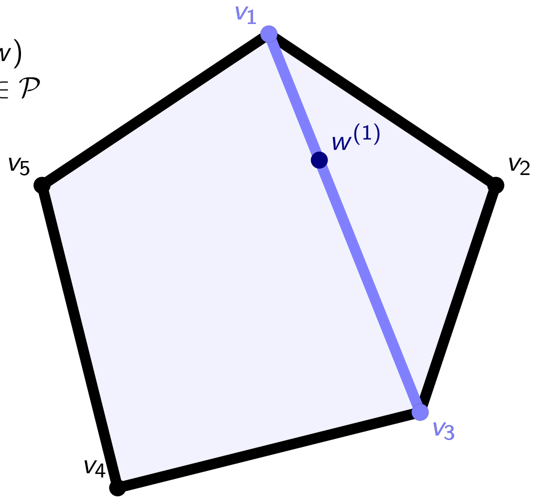
Conditional gradient method (Frank-Wolfe)

$$\begin{array}{ll} \text{minimize} & g(w) \\ \text{subject to} & w \in \mathcal{P} \end{array}$$



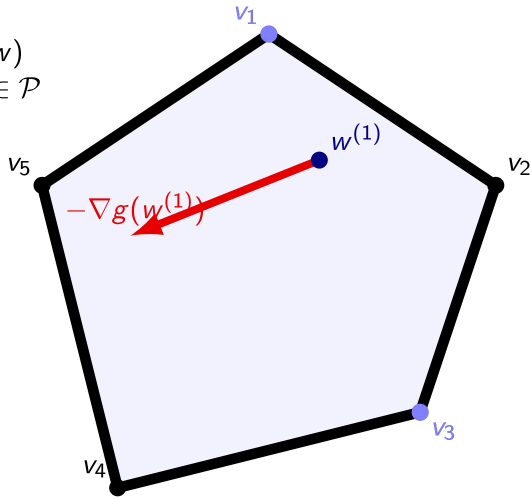
Conditional gradient method (Frank-Wolfe)

$$\begin{array}{ll} \text{minimize} & g(w) \\ \text{subject to} & w \in \mathcal{P} \end{array}$$



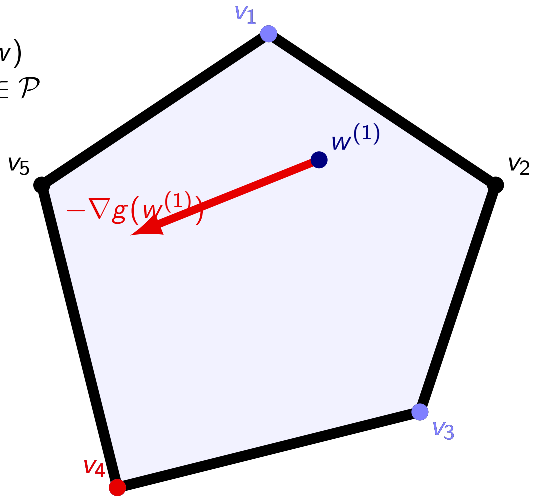
Conditional gradient method (Frank-Wolfe)

$$\begin{array}{ll} \text{minimize} & g(w) \\ \text{subject to} & w \in \mathcal{P} \end{array}$$



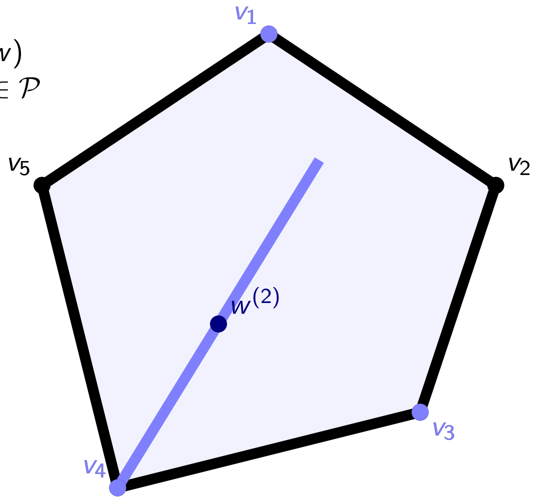
Conditional gradient method (Frank-Wolfe)

$$\begin{array}{ll} \text{minimize} & g(w) \\ \text{subject to} & w \in \mathcal{P} \end{array}$$



Conditional gradient method (Frank-Wolfe)

$$\begin{array}{ll} \text{minimize} & g(w) \\ \text{subject to} & w \in \mathcal{P} \end{array}$$



Linear optimization oracle for MOP

compute search direction

$$\underset{H \in \alpha\Delta_n}{\operatorname{argmin}} \langle H, G \rangle$$

- solution given by minimum eigenvector of G:

$$G = \sum_{i=1}^{n} \lambda_i v_i v_i^* \qquad \Longrightarrow \qquad H = \alpha v_1 v_1^*$$

- use shifted power method or randomized Lanczos method: apply $G = \mathcal{A}^*(\nabla f(\mathcal{A}X))$ and its conjugate to compute

$$\texttt{ApproxMinEvec}(G)$$

Linear optimization oracle for MOP

compute search direction

$$\underset{H \in \alpha \Delta_n}{\operatorname{argmin}} \langle H, G \rangle$$

- solution given by minimum eigenvector of G:

$$G = \sum_{i=1}^{n} \lambda_i v_i v_i^* \quad \Longrightarrow \quad H = \alpha v_1 v_1^*$$

- use shifted power method or randomized Lanczos method: apply $G = \mathcal{A}^*(\nabla f(\mathcal{A}X))$ and its conjugate to compute

$$\texttt{ApproxMinEvec}(G)$$

  (without forming G)

Conditional gradient descent

Algorithm CGM for the model problem (CMOP)

Input: Problem data for (CMOP); suboptimality ϵ
Output: Approximate solution X

```
function CGM
    X ← 0
    for t ← 0, 1, ... do
        (ξ, v) ← ApproxMinEvec(𝒜*(∇f(𝒜X)))
        H ← −α vv*
        if ⟨𝒜X − 𝒜H, ∇f(𝒜X)⟩ ≤ ε then break for
        η ← 2/(t + 2)
        X ← (1 − η)X + ηH
    return X
```

Two crucial ideas

To solve the problem using optimal storage:

- Use the low-dimensional "dual" variable

$$z_t = \mathcal{A}X_t \in \mathbb{R}^m$$

  to drive the iteration.
- Recover solution from small (randomized) sketch.

Never write down X until it has converged to low rank.

Conditional gradient descent

Algorithm CGM for the model problem (CMOP)

Input: Problem data for (CMOP); suboptimality ϵ
Output: Approximate solution X

function CGM
  $X \leftarrow 0$
  for $t \leftarrow 0, 1, \ldots$ **do**
    $(\xi, v) \leftarrow \texttt{ApproxMinEvec}(\mathcal{A}^*(\nabla f(\mathcal{A}X)))$
    $H \leftarrow -\alpha\, vv^*$
    if $\langle \mathcal{A}X - \mathcal{A}H, \nabla f(\mathcal{A}X) \rangle \leq \epsilon$ **then break for**
    $\eta \leftarrow 2/(t+2)$
    $X \leftarrow (1-\eta)X + \eta H$
  return X

Conditional gradient descent

Introduce "dual variable" $z = \mathcal{A}X \in \mathbb{R}^m$; eliminate X.

Algorithm Dual CGM for the model problem (CMOP)

Input: Problem data for (CMOP); suboptimality ϵ
Output: Approximate dual solution z^*

function $\textsc{dual}\mathrm{CGM}$
  $z \leftarrow 0$
  for $t \leftarrow 0, 1, \ldots$ **do**
    $(\xi, v) \leftarrow \texttt{ApproxMinEvec}(\mathcal{A}^*(\nabla f(z)))$
    $h \leftarrow \mathcal{A}(-\alpha v v^*)$
    if $\langle z - h, \nabla f(z) \rangle \leq \epsilon$ **then break for**
    $\eta \leftarrow 2/(t+2)$
    $z \leftarrow (1-\eta)z + \eta h$

Conditional gradient descent

Introduce "dual variable" $z = \mathcal{A}X \in \mathbb{R}^m$; eliminate X.

Algorithm Dual CGM for the model problem (CMOP)

Input: Problem data for (CMOP); suboptimality ϵ
Output: Approximate dual solution z^*

function $\textsc{dual}\mathrm{CGM}$
    $z \leftarrow 0$
    for $t \leftarrow 0, 1, \ldots$ **do**
        $(\xi, v) \leftarrow \texttt{ApproxMinEvec}(\mathcal{A}^*(\nabla f(z)))$
        $h \leftarrow \mathcal{A}(-\alpha v v^*)$
        if $\langle z - h, \nabla f(z) \rangle \leq \epsilon$ **then break for**
        $\eta \leftarrow 2/(t+2)$
        $z \leftarrow (1-\eta)z + \eta h$

---

we've solved the problem... but where's the solution?

Two crucial ideas

1. Use the low-dimensional “dual” variable

$$z_t = \mathcal{A}X_t \in \mathbb{R}^d$$

   to drive the iteration.
2. Recover solution from small (randomized) sketch.

How to catch a low rank matrix

if $\hat{X}$ has the same rank as $X^\star$,
and $\hat{X}$ acts like $X^\star$ (on its range and co-range),
then $\hat{X}$ is $X^\star$

use single-pass randomized sketch [Tropp et al. 2019; 2017a;b]

- see a series of additive updates
- remember how the matrix acts on random subspace
- reconstruct a low rank matrix that acts like $X^\star$
- storage cost for sketch and arithmetic cost of update are $\mathcal{O}(rn)$; reconstruction is $\mathcal{O}(r^2 n)$

Single-pass randomized sketch

- Draw and fix an standard normal matrix

$$\Omega \in \mathbb{R}^{n \times R}$$

Single-pass randomized sketch

- Draw and fix an standard normal matrix

$$\Omega \in \mathbb{R}^{n \times R}$$

- The sketch tracks the range and the trace of X

$$S = X\Omega \in \mathbb{R}^{n \times R} \quad \text{and} \quad \tau = \operatorname{tr} X \in \mathbb{R}$$

Single-pass randomized sketch

- Draw and fix an standard normal matrix
$$\Omega \in \mathbb{R}^{n\times R}$$

- The sketch tracks the range and the trace of X
$$S = X\Omega \in \mathbb{R}^{n\times R} \quad \text{and} \quad \tau = \operatorname{tr} X \in \mathbb{R}$$

- The sketch supports linear rank-one updates to X:
$$X' = (1-\eta)X + \eta vv^*$$
$$\Downarrow$$
$$S' = (1-\eta)S + \eta v(v^*\Omega)$$
$$\text{and} \quad \tau = (1-\eta)\tau + \eta \|v\|^2$$

Single-pass randomized sketch

- Draw and fix an standard normal matrix
$$\Omega \in \mathbb{R}^{n \times R}$$

- The sketch tracks the range and the trace of X
$$S = X\Omega \in \mathbb{R}^{n \times R} \quad \text{and} \quad \tau = \operatorname{tr} X \in \mathbb{R}$$

- The sketch supports linear rank-one updates to X:
$$X' = (1-\eta)X + \eta vv^*$$
$$\Downarrow$$
$$S' = (1-\eta)S + \eta v(v^*\Omega)$$
$$\text{and} \quad \tau = (1-\eta)\tau + \eta \|v\|^2$$

- Cost $\mathcal{O}(rn)$ to store and update sketch

Recovery from sketch

To recover approximation $\hat{X}$ from the sketch, compute

$$\hat{X} = S(\underbrace{\Omega^* S}_{R \times R})^\dagger S^* = X\Omega(\Omega^* X\Omega)^\dagger X\Omega^*$$

Recovery from sketch

To recover approximation $\hat{X}$ from the sketch, compute

$$\hat{X} = S(\underbrace{\Omega^* S}_{R \times R})^\dagger S^* = X\Omega(\Omega^* X \Omega)^\dagger X \Omega^*$$

For a rank $r < R$ approx $\hat{X}_r$ to X, use rank r approx $[(\Omega^* S)^\dagger]_r$

Recovery from sketch

To recover approximation $\hat{X}$ from the sketch, compute

$$\hat{X} = S(\underbrace{\Omega^* S}_{R\times R})^\dagger S^* = X\Omega(\Omega^* X\Omega)^\dagger X\Omega^*$$

For a rank $r < R$ approx $\hat{X}_r$ to X, use rank r approx $[(\Omega^* S)^\dagger]_r$

Error estimate: use the trace τ to determine the error

$$\left\| X - \hat{X}_r \right\|_* = \tau - \operatorname{tr} \hat{X}_r$$

[Yurtsever et al. 2019b]

Recovery from sketch

To recover approximation $\hat{X}$ from the sketch, compute

$$\hat{X} = S(\underbrace{\Omega^* S}_{R\times R})^\dagger S^* = X\Omega(\Omega^* X\Omega)^\dagger X\Omega^*$$

For a rank $r < R$ approx $\hat{X}_r$ to X, use rank r approx $[(\Omega^* S)^\dagger]_r$

Error estimate: use the trace τ to determine the error

$$\left\| X - \hat{X}_r \right\|_* = \tau - \operatorname{tr} \hat{X}_r$$

[Yurtsever et al. 2019b]

Theorem (A priori error [Tropp et al. 2017a])

For each $r < R$, the approximation above yields a rank-r matrix $\hat{X}_r$ with

$$\mathbb{E}_\Omega \|X - \hat{X}_r\|_* \leq \left(1 + \frac{r}{R - r - 1}\right) \|X - [X]_r\|_* .$$

Similar bounds hold with high probability.

SketchyCGM

Algorithm SketchyCGM for the model problem (CMOP)

Input: Problem data; suboptimality ϵ; target rank r
Output: Rank-r approximate solution $\hat{X} = V \Lambda V^*$

function $\textsc{SketchyCGM}$
  $\textsc{Sketch.Init}(n, r)$
  $z \leftarrow 0$
  for $t \leftarrow 0, 1, \ldots$ **do**
    $(\xi, v) \leftarrow \texttt{ApproxMinEvec}(\mathcal{A}^*(\nabla f(z)))$
    $h \leftarrow \mathcal{A}(-\alpha v v^*)$
    if $\langle z - h, \nabla f(z) \rangle \leq \epsilon$ **then break for**
    $\eta \leftarrow 2/(t+2)$
    $z \leftarrow (1-\eta) z + \eta h$
    $\textsc{Sketch.CGMUpdate}(\alpha, v, \eta)$
  $(\Lambda, V) \leftarrow \textsc{Sketch.Reconstruct}(\,)$
  return (Λ, V)

Guarantees

Suppose

- $X_{\mathrm{cgm}}^{(t)}$ is tth CGM iterate
- $\lfloor X_{\mathrm{cgm}}^{(t)} \rfloor_r$ is best rank r approximation to CGM solution
- $\hat{X}^{(t)}$ is SketchyCGM reconstruction after t iterations

Theorem (Convergence to CGM solution)

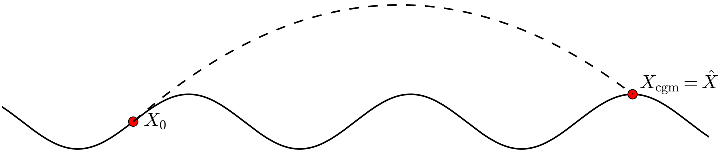
After t iterations, the SketchyCGM reconstruction satisfies

$$\mathbb{E} \, \| \hat{X}^{(t)} - X_{\mathrm{cgm}}^{(t)} \|_{\mathrm{F}} \leq 2 \, \| \lfloor X_{\mathrm{cgm}}^{(t)} \rfloor_r - X_{\mathrm{cgm}}^{(t)} \|_{\mathrm{F}} \, .$$

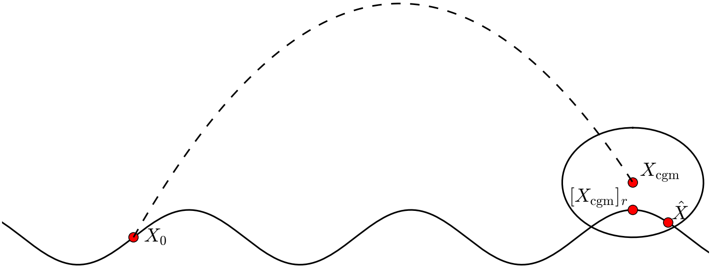
If in addition $X^\star = \lim_{t \to \infty} X_{\mathrm{cgm}}^{(t)}$ has rank r, then RHS $\to 0$!

[Yurtsever et al. 2017]

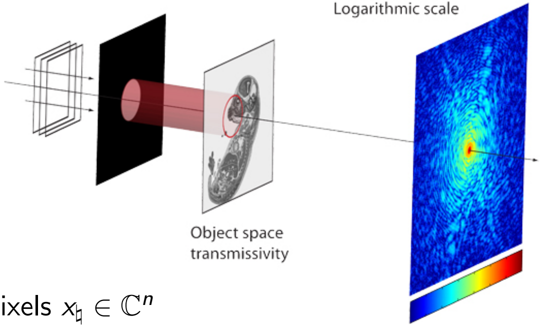
Convergence when rank($X_{\mathbf{cgm}}$) $\leq r$



Convergence when $\text{rank}(X_{\textbf{cgm}}) > r$



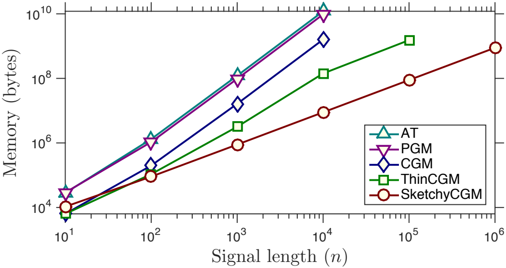
Application: Phase retrieval



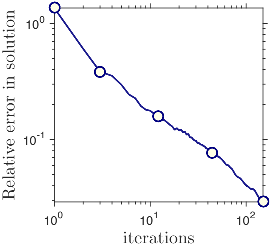
- image with n pixels $x_\natural \in \mathbb{C}^n$
- acquire noisy measurements $b_i = |\langle a_i, x_\natural \rangle|^2 + \omega_i$
- recover image by solving

$$\begin{array}{ll} \text{minimize} & f(\mathcal{A}X; b) \\ \text{subject to} & \operatorname{tr} X \le \alpha \\ & X \succeq 0. \end{array}$$

SketchyCGM is scalable



(a) Memory usage for five algorithms



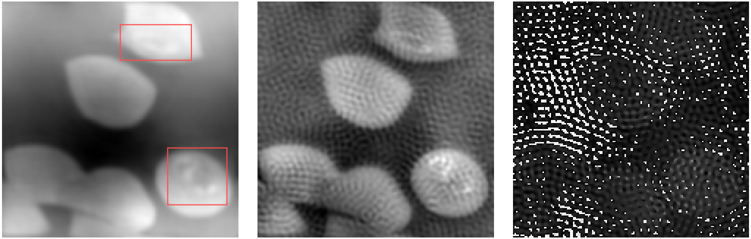
(b) Convergence for $n = 8 \cdot 10^6$.

PGM	=	proximal gradient (via TFOCS [Becker et al. 2011])
AT	=	accelerated PGM [Auslender and Teboulle 2006] (via TFOC
CGM	=	conditional gradient method [Jaggi 2013]
ThinCGM	=	CGM with thin SVD updates [Yurtsever et al. 2015]
SketchyCGM	=	ours, using $r = 1$

SketchyCGM is reliable

Fourier ptychography:

- imaging blood cells with $\mathcal{A}$ = subsampled FFT
- $n = 25,600$, $d = 185,600$
- $\text{rank}(X_\star) \approx 5$ (empirically)



(a) SketchyCGM (b) Burer–Monteiro (c) Wirtinger Flow

- brightness indicates phase of pixel (thickness of sample)
- red boxes mark malaria parasites in blood cells

Outline

Motivation

Large scale SDP

Complementary slackness

SketchyCGM

SketchyCGAL

*

Augmented Lagrangian

$$L_\beta(X; y) = \mathrm{tr}(CX) + \langle y, \mathcal{A}X - b\rangle + \frac{\beta}{2}\left\|\mathcal{A}X - b\right\|^2$$

- large β penalizes violations of constraint $\mathcal{A}X = b$
- dual variable y

Augmented Lagrangian

$$L_\beta(X; y) = \mathrm{tr}(CX) + \langle y, \mathcal{A}X - b \rangle + \frac{\beta}{2} \|\mathcal{A}X - b\|^2$$

- large β penalizes violations of constraint $\mathcal{A}X = b$
- dual variable y

saddle point of $L_\beta(X; y)$ gives solution to $(\mathcal{P})$:

- $\frac{\partial}{\partial y} L_\beta(X; y) = 0 \iff \mathcal{A}X = b$
- X is optimal for $(\mathcal{P})$ if in addition

$$X \in \operatorname*{argmin}_{X \in \alpha\Delta_n} L_\beta(X; y)$$

Conditional gradient augmented Lagrangian method

Augmented Lagrangian method: repeat

- $X \leftarrow \mathrm{argmin}_{X \in \alpha\Delta_n} L_\beta(X; y)$
- $y \leftarrow y + \eta(\mathcal{A}X - b)$

downside: minimization over X is too expensive

Conditional gradient augmented Lagrangian method

Augmented Lagrangian method: repeat

- $X \leftarrow \operatorname{argmin}_{X \in \alpha\Delta_n} L_\beta(X; y)$
- $y \leftarrow y + \eta(\mathcal{A}X - b)$

downside: minimization over X is too expensive

Conditional gradient augmented Lagrangian method:

- update X with approx CGM step on (smooth) $L_\beta(X; y)$ over constraint set $\alpha\Delta_n$:

$$
\begin{aligned}
(\xi, v) &\leftarrow \mathrm{ApproxMinEvec}(C + \mathcal{A}^*(y + \beta(\mathcal{A}X - b)); q_t) \\
X &\leftarrow (1-\eta)X + \eta\alpha vv^*
\end{aligned}
$$

- $y \leftarrow y + \eta(\mathcal{A}X - b)$

Conditional gradient augmented Lagrangian method

Algorithm CGAL [Yurtsever et al. 2019a]

Input: Problem data $\mathcal{A}$, b, C; maxiters T

Output: Approximate solution X_T

function $\mathrm{CGAL}(T)$
  $X \leftarrow 0_{n \times n}$ and $y \leftarrow 0_m$
  for $t \leftarrow 1, 2, 3, \ldots, T$ **do**
    $\beta \leftarrow \sqrt{t+1}$ and $\eta \leftarrow 2/(t+1)$
    $(\xi, v) \leftarrow \text{ApproxMinEvec}(C + \mathcal{A}^*(y + \beta(\mathcal{A}X - b)); q_t)$
    $X \leftarrow (1-\eta)\, X + \eta\, (\alpha\, vv^*)$
    $y \leftarrow y + \gamma(\mathcal{A}X - b)$
  return X

Conditional gradient augmented Lagrangian method

Algorithm CGAL [Yurtsever et al. 2019a]

Input: Problem data $\mathcal{A}$, b, C; maxiters T

Output: Approximate solution X_T

function $\mathrm{CGAL}(T)$
- $X \leftarrow 0_{n\times n}$ and $y \leftarrow 0_m$
- **for** $t \leftarrow 1, 2, 3, \ldots, T$ **do**
  - $\beta \leftarrow \sqrt{t+1}$ and $\eta \leftarrow 2/(t+1)$
  - $(\xi, v) \leftarrow \text{ApproxMinEvec}(C + \mathcal{A}^*(y + \beta(\mathcal{A}X - b)); q_t)$
  - $X \leftarrow (1-\eta)\,X + \eta\,(\alpha\, vv^*)$
  - $y \leftarrow y + \gamma(\mathcal{A}X - b)$
- **return** X

- set $T \approx \epsilon^{-1}$ to achieve ϵ-optimal solution
- CGAL provably works with $\epsilon \sim 1/\sqrt{t}$-approx eigenvectors:
  - with shifted power method, set $q_t = 8t^{1/2}\log n$
  - with randomized Lanczos method, set $q_t = t^{1/4}\log n$

CGAL: convergence rate

Fact (CGAL: Convergence [Yurtsever et al. 2019b])

Assume the SDP satisfies strong duality. CGAL with approximate eigenvector computations yields a sequence $\{X_t : t = 1, 2, 3, \ldots\} \subset \alpha\Delta_n$ *that satisfies*

$$\|\mathcal{A}X_t - b\| \leq \frac{\mathrm{Const}}{\sqrt{t}} \quad \textit{and} \quad |\langle C, X_t\rangle - \langle C, X_\star\rangle| \leq \frac{\mathrm{Const}}{\sqrt{t}}.$$

- ϵ-optimal iterate X_T after $\mathcal{O}(\epsilon^{-2})$ iterations
- ...which requires $\mathcal{O}(\epsilon^{-3})$ calls to primitives (mat-vec)
- constant depends on problem data $(C, \mathcal{A}, b, \alpha)$ and norm of dual solution $\|y_\star\|$

Two crucial ideas

1. Use the low-dimensional "dual" variable

$$z_t = \mathcal{A}X_t \in \mathbb{R}^d$$

   to drive the iteration.

2. Recover solution from small (randomized) sketch.

How to catch a low rank matrix

if $\hat{X}$ has the same rank as $X^\star$,
and $\hat{X}$ acts like $X^\star$ (on its range and co-range),
then $\hat{X}$ is $X^\star$

use single-pass randomized sketch [Tropp et al. 2019; 2017a;b]

- see a series of additive updates
- remember how the matrix acts on random subspace
- reconstruct a low rank matrix that acts like $X^\star$
- storage cost for sketch and arithmetic cost of update are $\mathcal{O}(rn)$; reconstruction is $\mathcal{O}(r^2 n)$

Single-pass randomized sketch

- Draw and fix an standard normal matrix

$$\Omega \in \mathbb{R}^{n \times R}$$

Single-pass randomized sketch

- Draw and fix an standard normal matrix

$$\Omega \in \mathbb{R}^{n\times R}$$

- The sketch tracks the range and the trace of X

$$S = X\Omega \in \mathbb{R}^{n\times R} \quad \text{and} \quad \tau = \operatorname{tr} X \in \mathbb{R}$$

Single-pass randomized sketch

- Draw and fix an standard normal matrix
$$\Omega \in \mathbb{R}^{n \times R}$$

- The sketch tracks the range and the trace of X
$$S = X\Omega \in \mathbb{R}^{n \times R} \quad \text{and} \quad \tau = \operatorname{tr} X \in \mathbb{R}$$

- The sketch supports linear rank-one updates to X:
$$X' = (1-\eta)X + \eta vv^*$$
$$\Downarrow$$
$$S' = (1-\eta)S + \eta v(v^*\Omega)$$
$$\text{and} \quad \tau = (1-\eta)\tau + \eta \|v\|^2$$

Single-pass randomized sketch

- Draw and fix an standard normal matrix

$$\Omega \in \mathbb{R}^{n \times R}$$

- The sketch tracks the range and the trace of X

$$S = X\Omega \in \mathbb{R}^{n \times R} \quad \text{and} \quad \tau = \operatorname{tr} X \in \mathbb{R}$$

- The sketch supports linear rank-one updates to X:

$$\begin{aligned} X' &= (1-\eta)X + \eta v v^* \\ &\Downarrow \\ S' &= (1-\eta)S + \eta v(v^*\Omega) \\ \text{and} \quad \tau &= (1-\eta)\tau + \eta \|v\|^2 \end{aligned}$$

- Cost $\mathcal{O}(rn)$ to store and update sketch

Recovery from sketch

To recover approximation $\hat{X}$ from the sketch, compute

$$\hat{X} = S(\underbrace{\Omega^* S}_{R \times R})^\dagger S^* = X\Omega(\Omega^* X\Omega)^\dagger X\Omega^*$$

Recovery from sketch

To recover approximation $\hat{X}$ from the sketch, compute

$$\hat{X} = S(\underbrace{\Omega^* S}_{R \times R})^\dagger S^* = X\Omega(\Omega^* X \Omega)^\dagger X \Omega^*$$

For a rank $r < R$ approx $\hat{X}_r$ to X, use rank r approx $[(\Omega^* S)^\dagger]_r$

Recovery from sketch

To recover approximation $\hat{X}$ from the sketch, compute

$$\hat{X} = S(\underbrace{\Omega^* S}_{R \times R})^\dagger S^* = X\Omega(\Omega^* X \Omega)^\dagger X \Omega^*$$

For a rank $r < R$ approx $\hat{X}_r$ to X, use rank r approx $[(\Omega^* S)^\dagger]_r$

Error estimate: use the trace τ to determine the error

$$\left\| X - \hat{X}_r \right\|_* = \tau - \operatorname{tr} \hat{X}_r$$

[Yurtsever et al. 2019b]

Recovery from sketch

To recover approximation $\hat{X}$ from the sketch, compute

$$\hat{X} = S(\underbrace{\Omega^* S}_{R \times R})^\dagger S^* = X\Omega(\Omega^* X\Omega)^\dagger X\Omega^*$$

For a rank $r < R$ approx $\hat{X}_r$ to X, use rank r approx $[(\Omega^* S)^\dagger]_r$

Error estimate: use the trace τ to determine the error

$$\left\| X - \hat{X}_r \right\|_* = \tau - \operatorname{tr} \hat{X}_r$$

[Yurtsever et al. 2019b]

Theorem (A priori error [Tropp et al. 2017a])

For each $r < R$, the approximation above yields a rank-r matrix $\hat{X}_r$ with

$$\mathbb{E}_\Omega \|X - \hat{X}_r\|_* \leq \left(1 + \frac{r}{R - r - 1}\right) \|X - [X]_r\|_* .$$

Similar bounds hold with high probability.

Sketchy CGAL

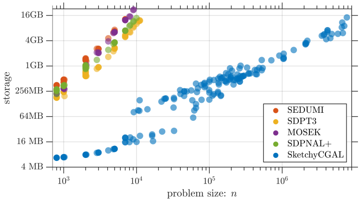
Algorithm SketchyCGAL Yurtsever et al. 2019a

Input: Problem data $\mathcal{A}$, b, C; maxiters T

Output: Approximate solution X_T

function $\mathrm{CGAL}(T)$
  $\textsc{Sketch.Init}(n, r)$
  $z \leftarrow 0_m$ and $y \leftarrow 0_m$
  for $t \leftarrow 1, 2, 3, \ldots, T$ **do**
    $\beta \leftarrow \sqrt{t+1}$ and $\eta \leftarrow 2/(t+1)$
    $(\xi, v) \leftarrow \mathrm{ApproxMinEvec}(C + \mathcal{A}^*(y + \beta(z - b)); q_t)$
    $\textsc{Sketch.CGMUpdate}(\alpha, v, \eta)$
    $z \leftarrow (1-\eta)z + \eta\alpha\mathcal{A}(vv^*)$
    $y \leftarrow y + \gamma(z - b)$
  $(\Lambda, V) \leftarrow \textsc{Sketch.Reconstruct}(\,)$
  return (Λ, V)

SketchyCGAL is scalable



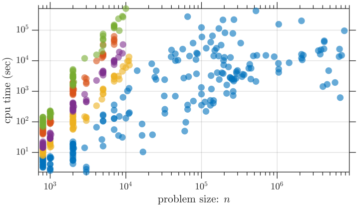


Figure: **MaxCut SDP: Scalability.** Storage cost [left] and runtime [right] of SketchyCGAL with sketch size $R = 10$ as compared with four standard SDP solvers. Each marker is one dataset.

SketchyCGAL for Quadratic Assignment Problem

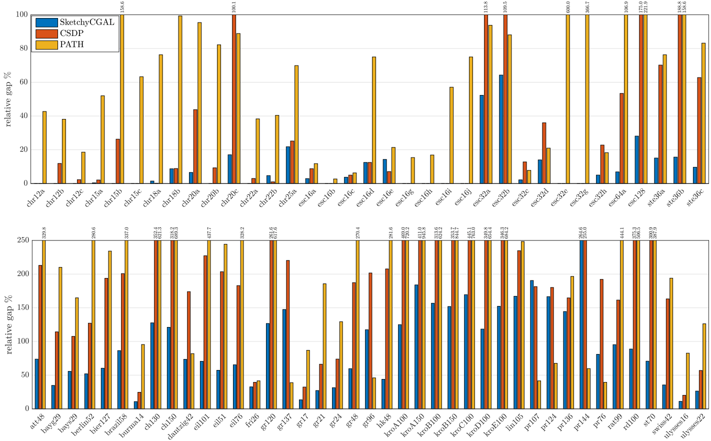


Figure: Using SketchyCGAL, CSDP [Bravo Ferreira et al. 2018], and PATH [Zaslavskiy et al. 2009], to solve SDP relaxations of QAPs from QAPLIB [top] and TSPLIB [bottom]. Bars compare the cost of the computed solution against the (known) optimal value; shorter is better.

Conclusion

Approximate complementarity approach provably solves (regular) SDP with optimal storage

- (+) uses $\mathcal{O}(nr)$ storage to find rank r solution
- (+) recovers approximate primal from approximate dual
- (+) parameters are easy to choose
- (+/-) relies on any subgradient solver for dual

Sketching methods provably solve large scale SDP efficiently

- (+) uses $\mathcal{O}(nr)$ storage to find rank r solution
- (+/-) requires compatible primal-dual solver (CGM, CGAL)
- (+) SketchyCGAL obtains ϵ-approximate solution after $\mathcal{O}(\epsilon^{-3})$ matrix-vector products

References

- A. Yurtsever, J. A. Tropp, O. Fercoq, M. Udell, and V. Cevher. Scalable Semidefinite Programming. SIMODS 2020. http://www.arxiv.org/abs/1912.02949
- L. Ding, A. Yurtsever, V. Cevher., J. A. Tropp, and M. Udell. An Optimal-Storage Approach to Semidefinite Programming using Approximate Complementarity. http://www.arxiv.org/abs/1902.03373
- A. Yurtsever, M. Udell, J. A. Tropp, and V. Cevher. Sketchy Decisions: Convex Optimization with Optimal Storage. AISTATS 2017.
- J. A Tropp, A. Yurtsever, M. Udell, and V. Cevher. Fixed-Rank Approximation of a Positive-Semidefinite Matrix from Streaming Data. NIPS 2017.

Outline

Motivation

Large scale SDP

Complementary slackness

SketchyCGM

SketchyCGAL

*

References

Absil, P.-A., Mahony, R., and Sepulchre, R. (2009). *Optimization algorithms on matrix manifolds*. Princeton University Press.

Alizadeh, F. (1991). Combinatorial optimization with interior point methods and semi-definite matrices. *Ph. D. thesis, University of Minnesota*.

Alizadeh, F. (1995). Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM journal on Optimization*, 5(1):13–51.

Alizadeh, F., Haeberly, J.-P. A., and Overton, M. L. (1997). Complementarity and nondegeneracy in semidefinite programming. *Mathematical programming*, 77(1):111–128.

Arora, S., Hazan, E., and Kale, S. (2005). Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 339–348. IEEE.

Auslender, A. and Teboulle, M. (2006). *Asymptotic cones and functions in optimization and variational inequalities*. Springer Science & Business Media.

Barvinok, A. I. (1995). Problems of distance geometry and convex properties of quadratic maps. *Discrete & Computational Geometry*, 13(2):189–202.

Becker, S. R., Candès, E. J., and Grant, M. C. (2011). Templates for convex cone problems with applications to sparse signal recovery. *Mathematical programming computation*, 3(3):165.

Boumal, N., Mishra, B., Absil, P.-A., and Sepulchre, R. (2014). Manopt, a matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15(1):1455–1459.

Boumal, N., Voroninski, V., and Bandeira, A. (2016). The non-convex burer-monteiro approach works on smooth semidefinite programs. In *Advances in Neural Information Processing Systems*, pages 2757–2765.

Bravo Ferreira, J. F. S., Khoo, Y., and Singer, A. (2018). Semidefinite programming approach for the quadratic assignment problem with a sparse graph. *Comput. Optim. Appl.*, 69(3):677–712.

Burer, S. and Monteiro, R. D. (2003). A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357.

Ding, L. and Udell, M. (2020). On the regularity and conditioning of semidefinite programs. *In preparation*.

Ding, L., Yurtsever, A., Cevher, V., Tropp, J. A., and Udell, M. (2019). An optimal-storage approach to semidefinite programming using approximate complementarity. *Submitted*.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Friedlander, M. P. and Macedo, I. (2016). Low-rank spectral optimization via gauge duality. *SIAM Journal on Scientific Computing*, 38(3):A1616–A1638.

Goemans, M. X. and Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145.

Halko, N., Martinsson, P. G., and Tropp, J. A. (2011). Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288.

Hazan, E. (2008). Sparse approximate solutions to semidefinite programs. In *Latin American Symposium on Theoretical Informatics*, pages 306–316. Springer.

Helmberg, C. and Rendl, F. (2000). A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3):673–696.

Jaggi, M. (2013). Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML (1)*, pages 427–435.

Kuczyński, J. and Woźniakowski, H. (1992). Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM J. Matrix Anal. Appl.*, 13(4):1094–1122.

Levy, K. (2017). Online to offline conversions, universality and adaptive minibatch sizes. In *Advances in Neural Information Processing Systems*, pages 1613–1622.

Levy, K. Y., Yurtsever, A., and Cevher, V. (2018). Online adaptive methods, universality and acceleration. *arXiv preprint arXiv:1809.02864*.

Nesterov, Y. and Nemirovski, A. (1989). Self-concordant functions and polynomial time methods in convex programming, ussr acad. *Sci., Central Economic&Mathematical Institute, Moscow*.

Nesterov, Y. and Nemirovskii, A. (1994). *Interior-point polynomial algorithms in convex programming*, volume 13. Siam.

O'Donoghue, B., Chu, E., Parikh, N., and Boyd, S. (2016). Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068.

Pataki, G. (1998). On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of operations research*, 23(2):339–358.

Rao, N., Shah, P., and Wright, S. (2013). Conditional gradient with enhancement and truncation for atomic-norm regularization. In *NIPS workshop on Greedy Algorithms*.

Rockafellar, R. T. (1970). *Convex Analysis*. Citeseer.

Srebro, N. and Shraibman, A. (2005). Rank, trace-norm and max-norm. In *International Conference on Computational Learning Theory*, pages 545–560. Springer.

Tropp, J. A., Yurtsever, A., Udell, M., and Cevher, V. (2017a). Fixed-rank approximation of a positive-semidefinite matrix from streaming data. In *Advances in Neural Information Processing Systems*.

Tropp, J. A., Yurtsever, A., Udell, M., and Cevher, V. (2017b). Practical sketching algorithms for low-rank matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1454–1485.

Tropp, J. A., Yurtsever, A., Udell, M., and Cevher, V. (2019). Streaming low-rank matrix approximation with an application to scientific simulation. *SIAM Scientific Computing (SISC)*.

Waldspurger, I. and Waters, A. (2018). Rank optimality for the burer-monteiro factorization. *arXiv preprint arXiv:1812.03046*.

Yurtsever, A., Fercoq, O., and Cevher, V. (2019a). A conditional gradient-based augmented lagrangian framework. *arXiv preprint arXiv:1901.04013*.

Yurtsever, A., Fercoq, O., Locatello, F., and Cevher, V. (2018). A conditional gradient framework for composite convex minimization with applications to semidefinite programming. *arXiv preprint arXiv:1804.08544*.

Yurtsever, A., Hsieh, Y.-P., and Cevher, V. (2015). Scalable convex methods for phase retrieval. In *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015 IEEE 6th International Workshop on*, pages 381–384. IEEE.

Yurtsever, A., Tropp, J. A., Fercoq, O., Udell, M., and Cevher, V. (2019b). Scalable semidefinite programming. *Submitted*.

Yurtsever, A., Udell, M., Tropp, J., and Cevher, V. (2017). Sketchy decisions: Convex low-rank matrix optimization with optimal storage. In *Artificial Intelligence and Statistics*, pages 1188–1196.

Zaslavskiy, M., Bach, F., and Vert, J. (2009). A path following algorithm for the graph matching problem. *IEEE Trans. Pattern Anal. Mach. Intel.*, 31(12):2227–2242.