Sketchy Decisions

Joel A. Tropp

Steele Family Professor
of Applied & Computational Mathematics

Computing + Mathematical Sciences
California Institute of Technology
jtropp@cms.caltech.edu

Collaborators: Volkan Cevher (EPFL), Roarke Horstmeyer (Duke),
Quoc Tran-Dinh (UNC), Madeleine Udell (Cornell), Alp Yurtsever (EPFL)

Research supported by ONR, AFOSR, NSF, DARPA, ERC, SNF, Sloan, and Moore.

Optimization with Optimal Storage

Optimization with Optimal Storage

Can we develop algorithms
that reliably solve an optimization problem
using storage that does not exceed
the size of the problem data
or the size of the solution?

Convex Low-Rank Matrix Optimization

$$\underset{\boldsymbol{X} \in \mathbb{H}_n}{\text{minimize}} \quad f(\mathcal{A}\boldsymbol{X}) \quad \text{subject to} \quad \operatorname{trace}(\boldsymbol{X}) = \alpha; \quad \boldsymbol{X} \text{ psd}$$

Details:

- $\mathcal{A} : \mathbb{H}_n \to \mathbb{R}^d$ is a real-linear map on $n \times n$ Hermitian matrices
- $f : \mathbb{R}^d \to \mathbb{R}$ is convex and differentiable
- In many applications,
  - $\mathcal{A}$ extracts d linear measurements of $n \times n$ matrix
  - $f = \operatorname{loss}(\,\cdot\,; \boldsymbol{b})$ for data $\boldsymbol{b} \in \mathbb{R}^d$
  - $d \ll n^2$
  - α modulates rank of solution
- Models problems in signal processing, statistics, and machine learning

Optimal Storage

What kind of storage bounds can we hope for?

- **Assume** black-box implementation of operations with linear map:

$$\boldsymbol{u} \mapsto \mathcal{A}(\boldsymbol{u}\boldsymbol{u}^*) \qquad\qquad (\boldsymbol{u}, \boldsymbol{z}) \mapsto (\mathcal{A}^*\boldsymbol{z})\boldsymbol{u}$$

$$\mathbb{C}^n \to \mathbb{R}^d \qquad\qquad \mathbb{C}^n \times \mathbb{R}^d \to \mathbb{C}^n$$

- Need $\Theta(n+d)$ storage for output of black-box operations

- Need $\Theta(rn)$ storage for rank-r approximate solution of model problem

> `Definition.` An algorithm for the model problem has optimal storage if its working storage is $\Theta(d+rn)$ rather than $\Theta(n^2)$.

Source: Yurtsever et al. 2017; Cevher et al. 2017.

So Many Algorithms...

- 1990s: **Interior-point methods**
  - Storage cost $\Theta(n^4)$ for Hessian
- 2000s: **Convex first-order methods**
  - (Accelerated) proximal gradient, spectral bundle methods, and others
  - Store matrix variable $\Theta(n^2)$
- 2008–Present: **Storage-efficient convex first-order methods**
  - Conditional gradient method (CGM) and extensions
  - Store matrix in low-rank form $O(tn)$; no storage guarantees
- 2009–Present: **Nonconvex heuristics**
  - Burer–Monteiro factorization idea + various nonlinear programming methods
  - Store low-rank matrix factors $\Theta(rn)$
  - For guaranteed solution, need unrealistic + unverifiable statistical assumptions

Sources: Interior-point: Nemirovski & Nesterov 1994; ... **First-order:** Rockafellar 1976; Helmberg & Rendl 1997; Auslender & Teboulle 2006; ... **CGM:** Frank & Wolfe 1956; Levitin & Poljak 1967; Jaggi 2013; ... **Heuristics:** Burer & Monteiro 2003; Keshavan et al. 2009; Jain et al. 2012; Candès et al. 2014; Bhojanapalli et al. 2015; Boumal et al. 2016; ....

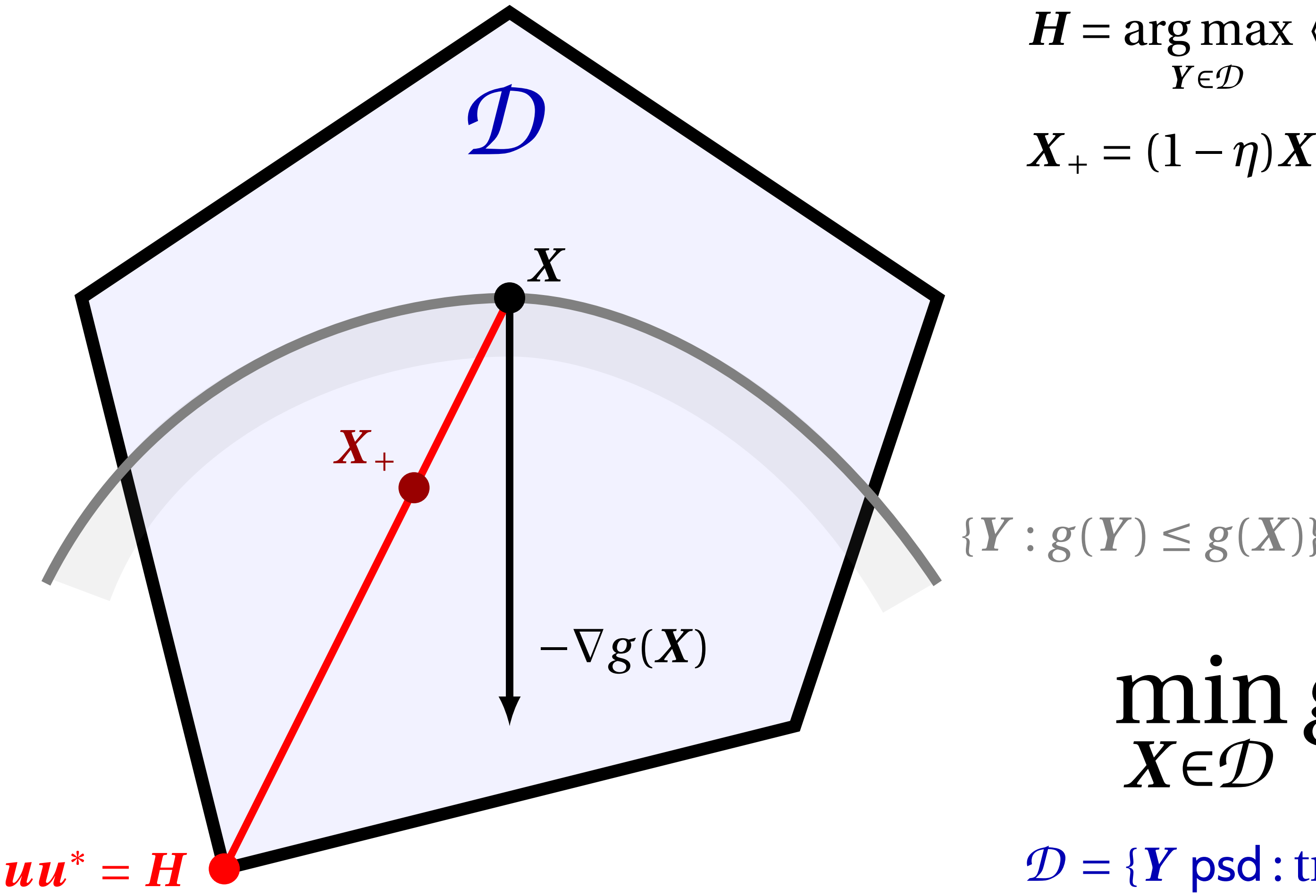
The Challenge

- Some algorithms provably solve the model problem...
- Some algorithms have optimal storage guarantees...

Is there an algorithm
that provably computes
a low-rank approximation
to a solution of the model problem
+ has optimal storage guarantees?

SketchyCGM

Geometry of CGM



$$H = \arg\max_{Y \in \mathcal{D}} \langle Y, -\nabla g(X) \rangle$$

$$X_+ = (1-\eta)X + \eta H$$

$$\min_{X \in \mathcal{D}} g(X)$$

$\mathcal{D} = \{Y \text{ psd} : \operatorname{trace}(Y) = 1\}$

CGM for the Model Problem

Input: Problem data; suboptimality ε

Output: Approximate solution $\boldsymbol{X}_{\mathrm{cgm}}$

```
function CGM
    X ← 0                                            ▷ Initialize variable
    for t ← 0, 1, 2, 3, ... do
        u ← MinEigVec(𝒜*(∇f(𝒜X)))                    ▷ Lanczos!
        H ← −α u u*                                  ▷ Form update direction
        if ⟨X − H, 𝒜*(∇f(𝒜X))⟩ ≤ ε
            then break for                           ▷ Stop when ε-suboptimal
        η ← 2/(t + 2)                                ▷ Update step size
        X ← (1 − η)X + ηH                            ▷ Update variable
    return X
```

Comment: In notation of last slide, $g = f \circ \mathcal{A}$. The gradient $\nabla g = \mathcal{A}^* \circ \nabla f \circ \mathcal{A}$.

Sources: Frank & Wolfe 1956; Levitin & Poljak 1967; Hazan 2008; Clarkson 2010; Jaggi 2013.

Crisis / Opportunity

Crisis:

- CGM needs many iterations to converge to a near-low-rank solution
- The ε-rank of the CGM iterates can increase without bound
- CGM requires high + unpredictable storage
- Typically involves dynamic memory allocation

Opportunity:

- Modify CGM to work with optimal storage!
- Drive the CGM iteration with small "dual" variable $\boldsymbol{z} = \mathcal{A}\boldsymbol{X}$
- Maintain small randomized sketch of primal matrix variable $\boldsymbol{X}$
- After iteration terminates, reconstruct matrix variable $\boldsymbol{X}$ from sketch

Source: Yurtsever et al. 2017.

SketchyCGM for the Model Problem

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

function SKETCHYCGM
  SKETCH.INIT(n, r) ▷ Initialize SKETCH to zero
  $\boldsymbol{z} \leftarrow \boldsymbol{0}$
  for $t \leftarrow 0, 1, 2, 3, \dots$ **do**
    $\boldsymbol{u} \leftarrow \texttt{MinEigVec}(\mathcal{A}^*(\nabla f(\boldsymbol{z})))$ ▷ Lanczos!
    $\boldsymbol{h} \leftarrow \mathcal{A}(-\alpha \boldsymbol{u}\boldsymbol{u}^*)$
    if $\langle \boldsymbol{z} - \boldsymbol{h},\ \nabla f(\boldsymbol{z}) \rangle \leq \varepsilon$ **then break for**
    $\eta \leftarrow 2/(t+2)$
    $\boldsymbol{z} \leftarrow (1-\eta)\boldsymbol{z} + \eta \boldsymbol{h}$
    SKETCH.CGMUPDATE$(-\sqrt{\alpha}\boldsymbol{u}, \eta)$ ▷ Update sketch of $\boldsymbol{X}$
  $(\boldsymbol{V}, \boldsymbol{\Lambda}) \leftarrow$ SKETCH.RECONSTRUCT$()$ ▷ Approx. eigendecomp of $\boldsymbol{X}$
  return $(\boldsymbol{V}, \boldsymbol{\Lambda})$

Source: Yurtsever et al. 2017.

Methods for Sketch Object

```
function SKETCH.INIT(n, r)                          ▷ Rank-r approx of n × n psd matrix
    k ← 2r                                          ▷ Increase k for better quality
    Ω ← randn(ℂ, n, k)
    Y ← zeros(n, k)

function SKETCH.CGMUPDATE(s, θ)
    Y ← (1 − θ)Y + θ s(s*Ω)                         ▷ Average ss* into sketch

function SKETCH.RECONSTRUCT( )
    C ← chol(Ω*Y)                                   ▷ Cholesky decomposition
    Z ← Y/C                                         ▷ Solve least-squares problems
    (U, Σ, ∼) ← svds(Z, r)                          ▷ Compute r-truncated SVD
    return (U, Σ²)                                  ▷ Return eigenvalue factorization
```

Sources: Williams & Seeger 2001; Drineas & Mahoney 2005; Gittens 2011, 2013; Pourkamali-Anaraki & Becker 2016; Wang, Gittens, & Mahoney 2017; Tropp et al. 2017.

Less Filling / Great Taste

Theorem 1 (YUTC 2016). *SketchyCGM has the following properties:*

- *SketchyCGM has optimal storage guarantee $\Theta(d + rn)$*
- *SketchyCGM produces an ε-suboptimal objective value after $O(\varepsilon^{-1})$ iterations*
- *Suppose CGM produces iterates $\boldsymbol{X}_t$ that converge to a rank-r matrix $\boldsymbol{X}_{\mathrm{cgm}}$. Then SketchyCGM produces rank-r iterates $\hat{\boldsymbol{X}}_t$ that satisfy*

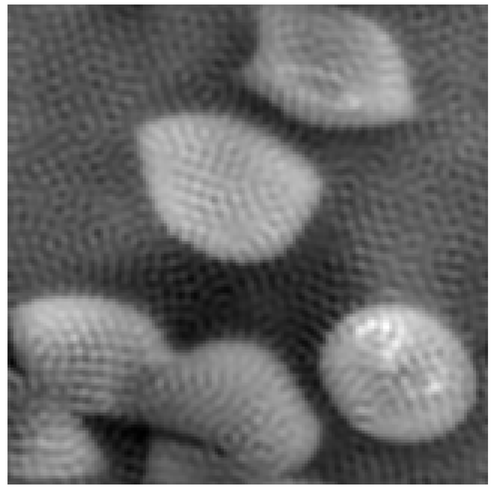
$$\mathbb{E}\left\| \hat{\boldsymbol{X}}_t - \boldsymbol{X}_{\mathrm{cgm}} \right\|_{S_1} \to 0.$$

Source: "Everything you always wanted in an algorithm. And less." https://www.youtube.com/watch?v=0agZEMEpiVI.

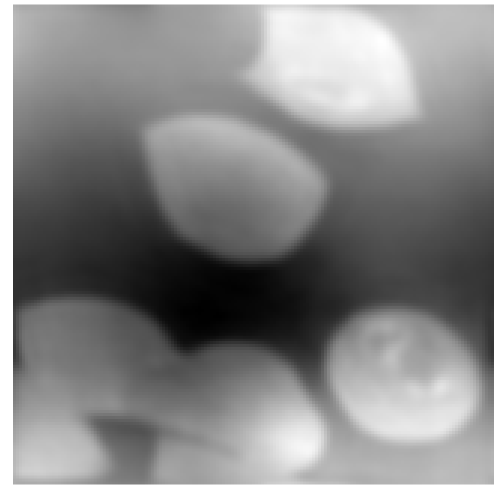
Performance of SketchyCGM

Fourier Ptychography

Wirtinger Flow



Burer–Monteiro



SketchyCGM

29 illuminations; 80×80 pixels each; $d = 1.86 \cdot 10^5$ measurements

image size $n = 160 \times 160$ pixels; matrix size $n^2 = 6.55 \cdot 10^8$

SketchyCGM storage (rank $r = 1$): $6.53 \cdot 10^5$

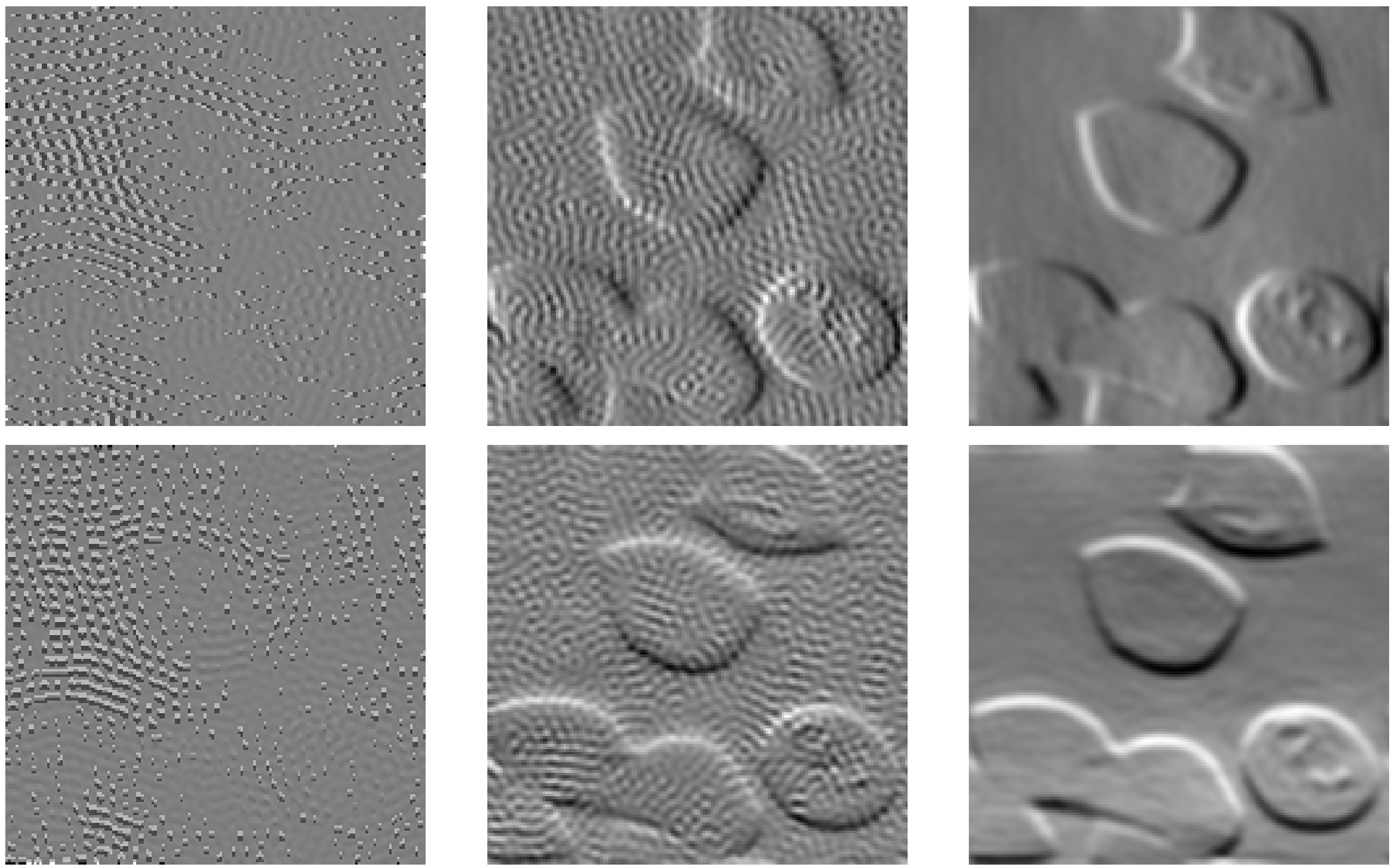
quadratic loss

Sources: Burer & Monteiro 2003; Balan et al. 2008; Chai et al. 2011; Zheng, Horstmeyer, & Yang 2013; Horstmeyer & Yang 2014; Candès et al. 2014; Horstmeyer et al. 2015; Yurtsever et al. 2017.

Fourier Ptychography: Malaria Phase Gradients

Δ_x

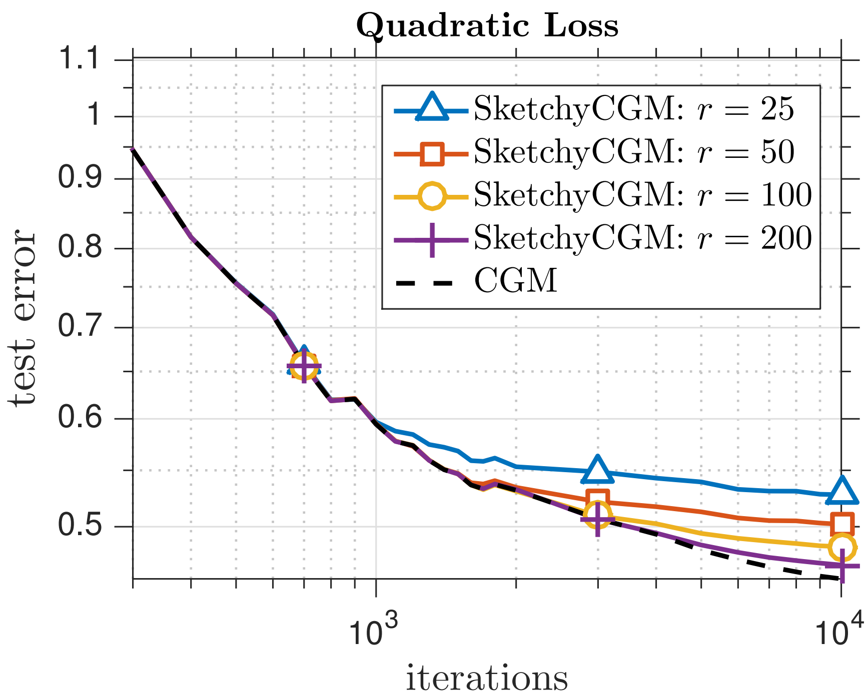
Δ_y



Wirtinger Flow   Burer–Monteiro   SKETCHYCGM

MovieLens 10M

❧ $m = 71,567$ users, $n = 10,681$ movies, $d = 10^7$ ratings, dim. $mn = 7.64 \cdot 10^8$



Approximate storage costs

Rank (r)	SketchyCGM
25	$3.28 \cdot 10^7$
50	$4.51 \cdot 10^7$
100	$6.98 \cdot 10^7$
200	$1.19 \cdot 10^8$

Source: Harper & Konstan 2015; Yurtsever et al. 2017.

To learn more...

E-mail: `jtropp@cms.caltech.edu`

Web: `http://users.cms.caltech.edu/~jtropp`

Papers:

- Halko, Martinsson, & Tropp, "Finding structure with randomness: Probabilistic algorithms for computing approximate matrix decompositions," *SIAM Review*, 2011
- Horstmeyer et al. "Solving ptychography with a convex relaxation," *New J. Physics*, 2015
- Tropp, Yurtsever, Udell, & Cevher, "Fixed-rank approximation of a positive-semidefinite matrix from streaming data," NIPS, 2017
- Tropp, Yurtsever, Udell, & Cevher, "Practical sketching algorithms for low-rank matrix approximation," *SIMAX*, 2017
- Tropp, Yurtsever, Udell, & Cevher, "More practical sketching algorithms for low-rank matrix approximation," soon!
- Yurtsever, Udell, Tropp, & Cevher, "Sketchy decisions: Convex low-rank matrix optimization with optimal storage," AISTATS, 2017
- Cevher, Tropp, & Yurtsever, "Storage-optimal algorithms for semidefinite programming," eventually!