Local Search Algorithms for Rank-Constrained Convex Optimization

Kyriakos Axiotis

MIT

Maxim Sviridenko

YAHOO! RESEARCH

The Problem

Matrix Completion

$\boldsymbol{M}$: Data matrix

Ω: Set of observable entries

Find $\mathbf{A}$ that minimizes

$$\sum_{ij \in \Omega} \left(M_{ij} - A_{ij}\right)^2$$

subject to

$$\operatorname{rank}(\boldsymbol{A}) \leq r$$

Rank-Constrained Convex Optimization

R: Convex function

Find $\boldsymbol{A}$ that minimizes

$$R(\boldsymbol{A})$$

subject to

$$\operatorname{rank}(\boldsymbol{A}) \leq r$$

Rank-Constrained Convex Optimization

R: Convex function

Find $\boldsymbol{A}$ that minimizes

$$R(\boldsymbol{A})$$

subject to

$$\boxed{\operatorname{rank}(\boldsymbol{A}) \le r}$$

NP-hard problem

Usual: Replace $\operatorname{rank}(\boldsymbol{A})$ by $\|\boldsymbol{A}\|_*$
(trace norm)

Here: Relax to $\operatorname{rank}(\boldsymbol{A}) \le c \cdot r$

Algorithms & Theoretical results

$$\min_{\mathrm{rank}(\boldsymbol{A}) \leq c \cdot r} R(\boldsymbol{A})$$

A Greedy Algorithm (Shalev-Schwarz et al. (2011))

$\boldsymbol{U} = \boldsymbol{V} = ()$

For $i = 1 \ldots c \cdot r$

$\boldsymbol{u}, \boldsymbol{v} =$ top singular vectors of $\nabla R(\boldsymbol{A})$

$$\min_{\operatorname{rank}(\boldsymbol{A}) \leq c \cdot r} R(\boldsymbol{A})$$

A Greedy Algorithm (Shalev-Schwarz et al. (2011))

$\boldsymbol{U} = \boldsymbol{V} = ()$

For $i = 1 \ldots c \cdot r$

$\quad \boldsymbol{u}, \boldsymbol{v} =$ top singular vectors of $\nabla R(\boldsymbol{A})$

$\quad \boldsymbol{U} = (\boldsymbol{U} \quad \boldsymbol{u})$

$\quad \boldsymbol{V}^\top = (\boldsymbol{V}^\top \quad \boldsymbol{v})$

$$\min_{\mathrm{rank}(\boldsymbol{A}) \leq c \cdot r} R(\boldsymbol{A})$$

A Greedy Algorithm (Shalev-Schwarz et al. (2011))

$\boldsymbol{U} = \boldsymbol{V} = ()$

For $i = 1 \ldots c \cdot r$

$\quad \boldsymbol{u}, \boldsymbol{v} =$ top singular vectors of $\nabla R(\boldsymbol{A})$

$\quad \boldsymbol{U} = (\boldsymbol{U} \quad \boldsymbol{u})$

$\quad \boldsymbol{V}^\top = (\boldsymbol{V}^\top \quad \boldsymbol{v})$

$\quad \boldsymbol{X} = \mathrm{argmin}_{\boldsymbol{X}} R(\boldsymbol{U}\boldsymbol{X}\boldsymbol{V})$

$\quad \boldsymbol{U} = \boldsymbol{U}\boldsymbol{X}$

$$\min_{\mathrm{rank}(\boldsymbol{A})\le c\cdot r} R(\boldsymbol{A})$$

A Greedy Algorithm (Shalev-Schwarz et al. (2011))

$\boldsymbol{U} = \boldsymbol{V} = ()$

For $i = 1 \ldots c \cdot r$

$\quad \boldsymbol{u}, \boldsymbol{v} =$ top singular vectors of $\nabla R(\boldsymbol{A})$

$\quad \boldsymbol{U} = (\boldsymbol{U} \quad \boldsymbol{u})$

$\quad \boldsymbol{V}^\top = (\boldsymbol{V}^\top \quad \boldsymbol{v})$

$\quad \boldsymbol{X} = \mathrm{argmin}_{\boldsymbol{X}}\, R(\boldsymbol{U}\boldsymbol{X}\boldsymbol{V})$

$\quad \boldsymbol{U} = \boldsymbol{U}\boldsymbol{X}$

Return $\boldsymbol{U}\boldsymbol{V}$

(κ: condition number of R)

Known bound: $c \le O\left(\kappa \frac{R(\mathbf{0})}{\epsilon}\right)$

Our bound: $c \le O\left(\kappa \log \frac{R(\mathbf{0})}{\epsilon}\right)$

$$\min_{\text{rank}(\boldsymbol{A}) \le c \cdot r} R(\boldsymbol{A})$$

A Faster Greedy Algorithm

$\boldsymbol{U} = \boldsymbol{V} = ()$

For $i = 1 \ldots c \cdot r$

$\boldsymbol{u}, \boldsymbol{v} =$ top singular vectors of $\nabla R(\boldsymbol{A})$

$\boldsymbol{U} = (\boldsymbol{U} \quad \boldsymbol{u})$

$\boldsymbol{V}^\top = (\boldsymbol{V}^\top \quad \boldsymbol{v})$

$\boldsymbol{X} = \text{argmin}_{\boldsymbol{X}} R(\boldsymbol{UXV})$

$\boldsymbol{U} = \boldsymbol{UX}$

Return $\boldsymbol{UV}$

$$\min_{\mathrm{rank}(\boldsymbol{A})\leq c\cdot r} R(\boldsymbol{A})$$

A Faster Greedy Algorithm

$\boldsymbol{U} = \boldsymbol{V} = ()$

For $i = 1 \ldots c \cdot r$

$\quad \boldsymbol{u}, \boldsymbol{v} =$ top singular vectors of $\nabla R(\boldsymbol{A})$

$\quad \boldsymbol{U} = (\boldsymbol{U} \quad \boldsymbol{u})$

$\quad \boldsymbol{V}^\top = (\boldsymbol{V}^\top \quad \boldsymbol{v})$

$\quad$ **If** i is even: $\boldsymbol{V} = \mathrm{argmin}_{\boldsymbol{V}} R(\boldsymbol{U}\boldsymbol{V}^\top)$

$\quad$ **Else**: $\boldsymbol{U} = \mathrm{argmin}_{\boldsymbol{U}} R(\boldsymbol{U}\boldsymbol{V}^\top)$

$\leftarrow \Theta(c \cdot r)$ speedup in many cases

Return $\boldsymbol{U}\boldsymbol{V}$

$$\min_{\text{rank}(\boldsymbol{A}) \le c \cdot r} R(\boldsymbol{A})$$

A Local Search Algorithm

After **inserting** new vectors, **remove** one set of vectors

$$c \le O(\kappa^2)$$

Experiments

FG-BG Separation using Robust PCA

$$\boldsymbol{M} = \boldsymbol{L} + \boldsymbol{S}$$

input video | BG (low-rank) | FG (sparse)

Huber loss

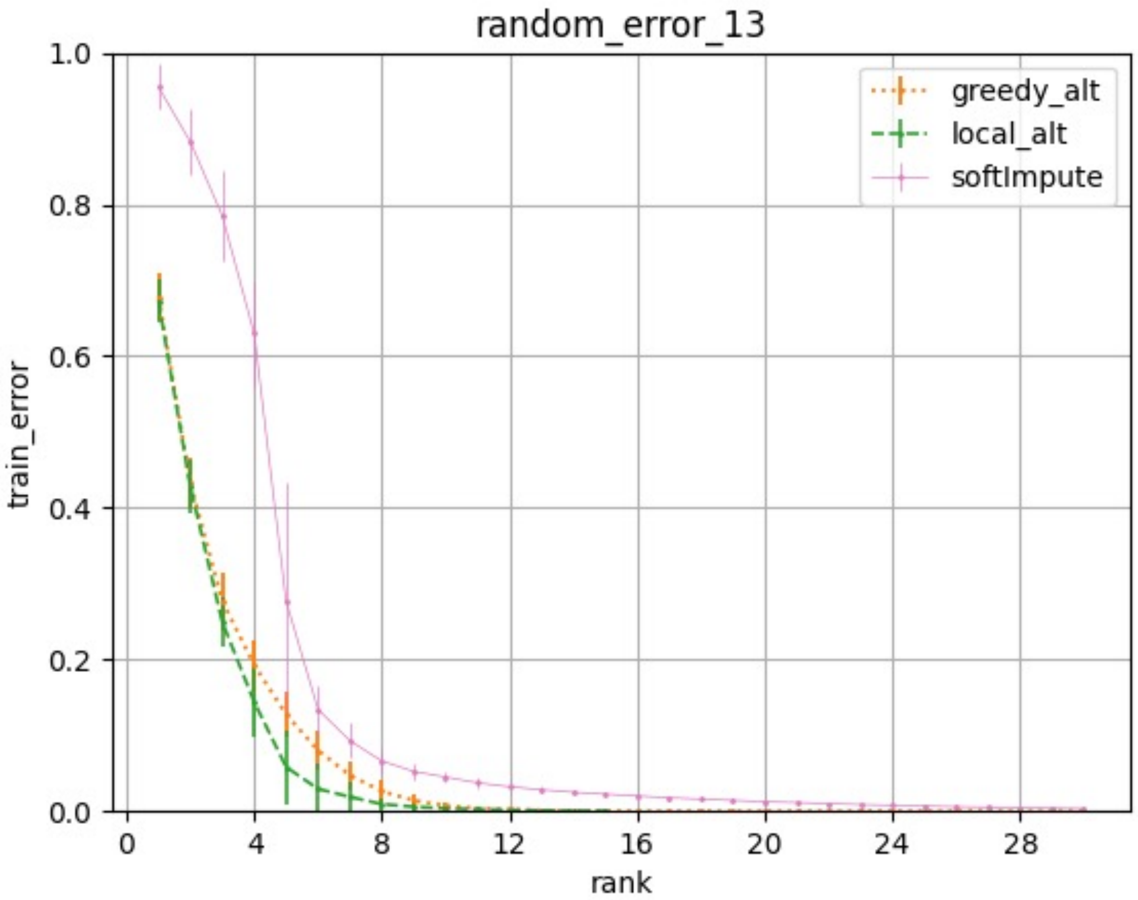
$$\min_{rank(\boldsymbol{L}) \leq r} \boxed{H_\delta(\boldsymbol{M} - \boldsymbol{L})}$$

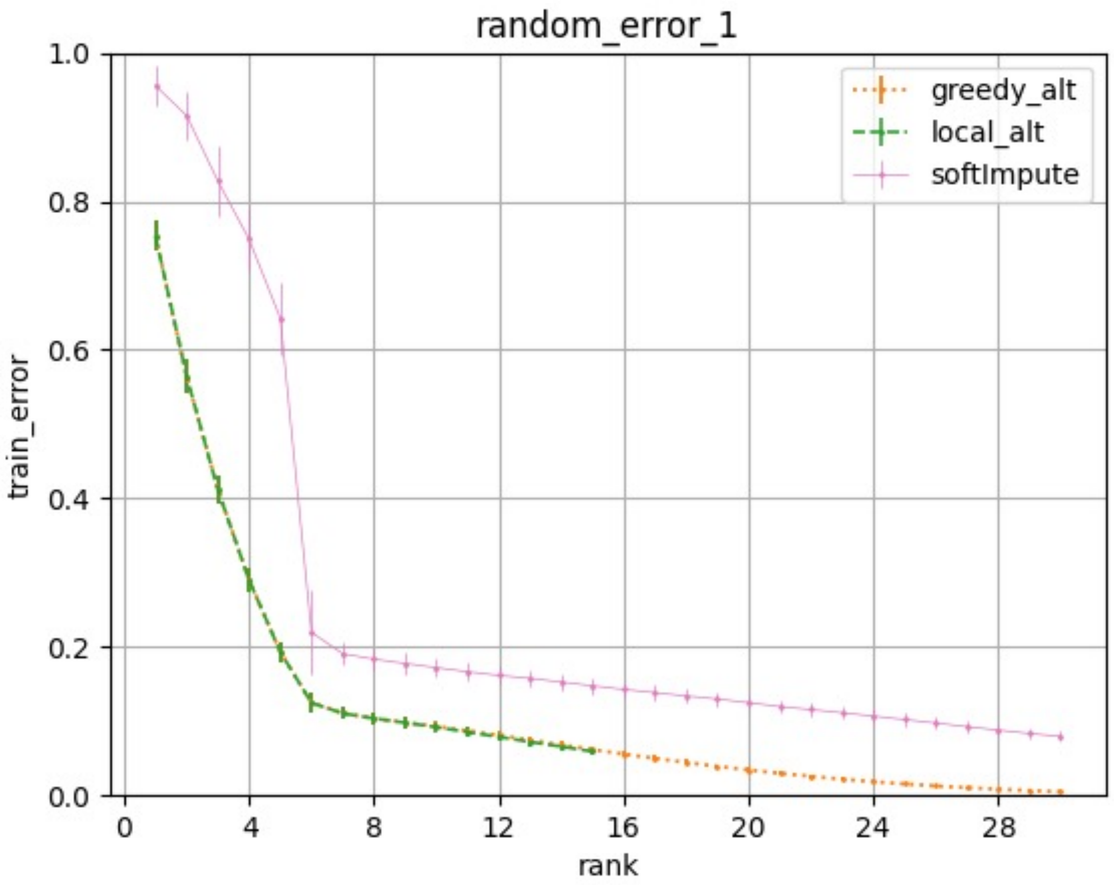
Fast Greedy on Huber loss →

original video
↓

Principal Component Pursuit (PCP) →

Matrix Completion (Optimization Task)





Recommender Systems

(metric: relative ℓ_2^2 error)

Algorithm	MovieLens 100K	MovieLens 1M	MovieLens 10M
NMF (Lee & Seung (2001))	0.9659	0.9166	0.8960
SoftImpute	1.0106	0.9599	0.957
Alternating Minimization	**0.9355**	0.8732	0.8410
SVD (Koren et al. (2009))	0.9533	0.8743	**0.8315**
Fast Greedy (Algorithm 2.3)	0.9451	**0.8714**	0.8330

Conclusions

- New **efficient greedy** & **local search** algorithms for rank-constrained convex optimization
- Even though very **general**, on par with algorithms for specialized applications (e.g. robust PCA, recommender systems)

Thank you!