
Fall in love
with your next book
Let the booksellers of
Gower Street help you
find your perfect match.

Low-rank matrix completion for recommender systems: optimization on manifolds at work

Nicolas Boumal

Joint work with Pierre-Antoine Absil

Université catholique de Louvain

March 2011

Recommender systems tell you which items you might like

based on a huge database of ratings

$$\leftarrow n \text{ cols} \rightarrow$$

$$X = \begin{pmatrix} ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \end{pmatrix} \begin{matrix} \uparrow \\ m \text{ rows} \\ \downarrow \end{matrix}$$

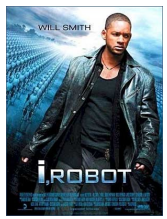
One row per item, one column per user

Ratings of items by the users are recorded

$$X = \begin{pmatrix} ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \end{pmatrix}$$

user j

item i



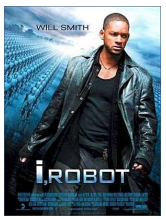
One row per item, one column per user

Ratings of items by the users are recorded

$$X = \begin{pmatrix} ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & 4 & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \end{pmatrix}$$

user j

item i



Most ratings are unknown

Our job is to complete X

$$X = \begin{pmatrix} 1 & ? & 2 & ? & ? & 5 & ? & ? & ? & ? & 5 & ? & ? & ? & 2 & ? \\ 2 & ? & 2 & ? & ? & 4 & ? & ? & ? & 3 & 4 & ? & 5 & ? & ? & ? \\ 1 & ? & 5 & 2 & ? & 4 & ? & 4 & ? & ? & ? & 2 & ? & ? & ? & ? \\ ? & 1 & ? & 3 & ? & ? & ? & 3 & ? & ? & 3 & ? & 2 & ? & 5 & 5 \\ 4 & 4 & ? & ? & ? & ? & 5 & ? & ? & ? & 1 & ? & ? & 1 & ? & 4 \end{pmatrix}$$

Exploit similarities between users and items

to complete the matrix

$$X = \begin{pmatrix} 1 & ? & 2 & ? & ? & 5 & ? & ? & ? & ? & 5 & ? & ? & ? & 2 & ? \\ 2 & ? & 2 & ? & ? & 4 & ? & ? & ? & 3 & 4 & ? & 5 & ? & ? & ? \\ 1 & ? & 5 & 2 & ? & 4 & ? & 4 & ? & ? & ? & 2 & ? & ? & ? & ? \\ ? & 1 & ? & 3 & ? & ? & ? & 3 & ? & ? & 3 & ? & 2 & ? & 5 & 5 \\ 4 & 4 & ? & ? & ? & ? & 5 & ? & ? & ? & 1 & ? & ? & 1 & ? & 4 \end{pmatrix}$$

In a global, automated, scalable way?

Scalability will guide the algorithm design

for both time and memory complexity

Netflix 1M$ prize:

- 17,700 movies;
- 480,000 users;
- 100,000,000 ratings (1%).

$\Rightarrow$ The whole matrix won't fit into memory,

$\Rightarrow$ but the known ratings will.

We assume that X has low-rank r

Hence, that ratings are inner products in a small space $\mathbb{R}^r$

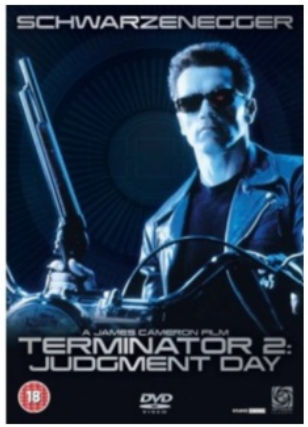
$$X = \begin{pmatrix} 1 & ? & 2 & ? & ? & 5 & ? & ? & ? & ? & 5 & ? & ? & ? & 2 & ? \\ 2 & ? & 2 & ? & ? & 4 & ? & ? & ? & 3 & 4 & ? & 5 & ? & ? & ? \\ 1 & ? & 5 & 2 & ? & 4 & ? & 4 & ? & ? & ? & 2 & ? & ? & ? & ? \\ ? & 1 & ? & 3 & ? & ? & ? & 3 & ? & ? & 3 & ? & 2 & ? & 5 & 5 \\ 4 & 4 & ? & ? & ? & ? & 5 & ? & ? & ? & 1 & ? & ? & 1 & ? & 4 \end{pmatrix}$$

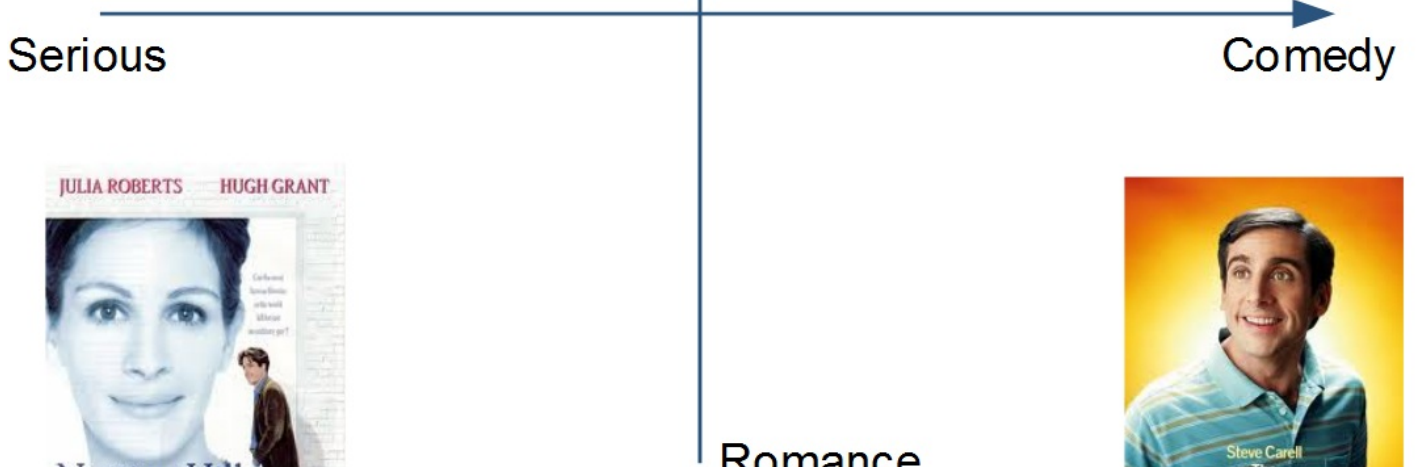
$$= \begin{pmatrix} ? & ? \\ ? & ? \\ ? & ? \\ ? & ? \\ ? & ? \end{pmatrix} \cdot \begin{pmatrix} ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \end{pmatrix}$$

Rationale: only a few factors influence our preferences

We map items and users to this small dimensional space

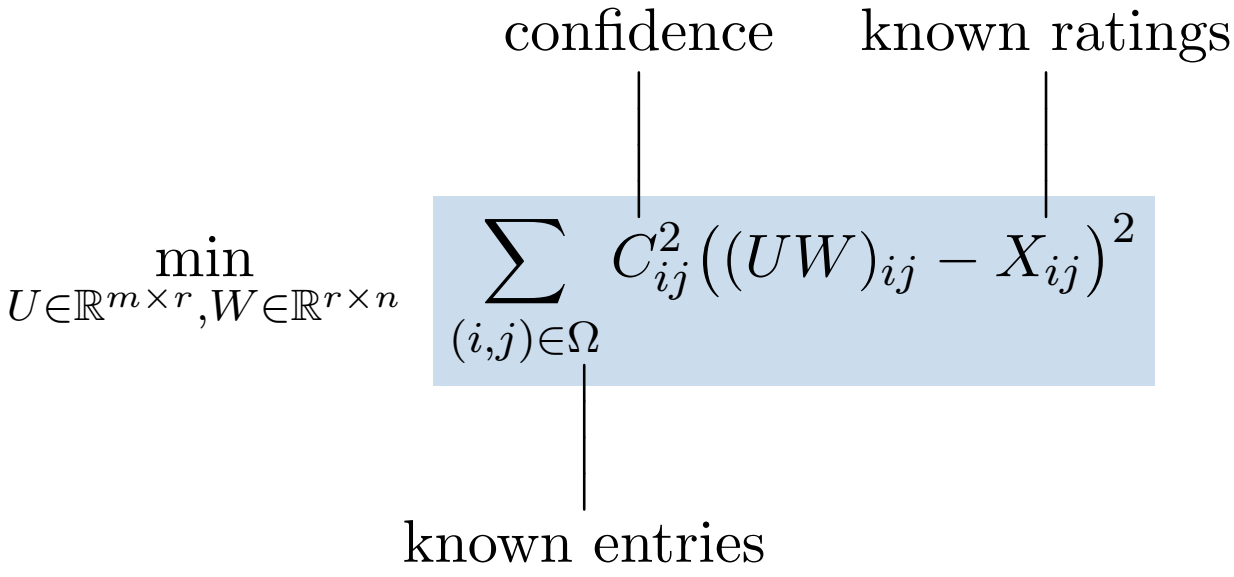
without any human intervention





Toward a reasonable objective function

The optimal choice UW is an m-by-n matrix of rank r in best agreement with the k known entries of X



A more comfortable notation

$$C_{ij} = 0 \ \forall (i,j) \notin \Omega$$

$$\|C \odot (UW - X)\|_{\mathrm{F}}^2 = \sum_{(i,j)\in\Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij}\big)^2$$

entry-wise product

$$\|A\|_{\mathrm{F}}^2 = \textstyle\sum_{ij} A_{ij}^2$$

This is reasonable

$$\min_{U \in \mathbb{R}^{m \times r}, W \in \mathbb{R}^{r \times n}} \|C \odot (UW - X)\|_{\mathrm{F}}^2$$

- Very natural;
- Search space of dimension $r(m+n)$;
- Objective computable in $\mathcal{O}(rk)$ time;
- Ideal for Gaussian noise on ratings X_{ij}.

This is reasonable, *but*

$$\min_{U \in \mathbb{R}^{m \times r}, W \in \mathbb{R}^{r \times n}} \quad \|C \odot (UW - X)\|_{\mathrm{F}}^2$$

Minimizers are not isolated.

This is reasonable, *but*

$$\min_{U \in \mathbb{R}^{m \times r}, W \in \mathbb{R}^{r \times n}} \quad \|C \odot (UW - X)\|_{\mathrm{F}}^2$$

Minimizers are not isolated.

If (U, W) is a minimizer, $(UM, M^{-1}W)$ is too,
for any r-by-r invertible matrix M.

The objective is invariant under invertible transformations

We don't want that. Why?

- The search space $\mathbb{R}^{m \times r} \times \mathbb{R}^{r \times n}$ is bigger than it ought to be;
- There are no theoretical guarantees of convergence when critical points are not isolated;
- And it may prevent superlinear convergence rates.

Partial solution: force U to be orthonormal

It's a kind of normalization

$$\min_{U \in \mathrm{St}(m,r), W \in \mathbb{R}^{r \times n}} \|C \odot (UW - X)\|_{\mathrm{F}}^2$$

$$\mathrm{St}(m,r) = \{U \in \mathbb{R}^{m \times r} : U^\top U = I_r\}$$

Partial solution: force U to be orthonormal

It's a kind of normalization

$$\min_{U \in \mathrm{St}(m,r), W \in \mathbb{R}^{r \times n}} \|C \odot (UW - X)\|_{\mathrm{F}}^2$$

$$\mathrm{St}(m,r) = \{U \in \mathbb{R}^{m \times r} : U^\top U = I_r\}$$

Minimizers are still not isolated.

Partial solution: force U to be orthonormal

It's a kind of normalization

$$\min_{U \in \mathrm{St}(m,r), W \in \mathbb{R}^{r \times n}} \|C \odot (UW - X)\|_{\mathrm{F}}^2$$

$$\mathrm{St}(m,r) = \{U \in \mathbb{R}^{m \times r} : U^\top U = I_r\}$$

Minimizers are still not isolated.

If (U, W) is a minimizer, $(UQ, Q^\top W)$ is too,
for any r-by-r orthogonal matrix Q.

The set of r-dimensional subspaces of $\mathbb{R}^m$ is the Grassmann manifold $\mathrm{Gr}(m, r)$

- Picture Grassmann as a smooth, curved surface in space (e.g., a sphere);
- We represent a point $\mathscr{U}$ on Grassmann with any orthonormal matrix U such that $\mathrm{col}(U) = \mathscr{U}$;
- Manifolds have geodesics, distances, tangent vectors...

The objective is a function of the column space $\mathrm{col}(U)$

$$\min_{\mathscr{U} \in \mathrm{Gr}(m,r), W \in \mathbb{R}^{r \times n}} \quad \|C \odot (UW - X)\|_{\mathrm{F}}^2$$

U is any m-by-r orthonormal matrix such that $\mathrm{col}(U) = \mathscr{U}$.

The objective is a function of the column space $\mathrm{col}(U)$

$$\min_{\mathscr{U} \in \mathrm{Gr}(m,r)} \quad \min_{W \in \mathbb{R}^{r \times n}} \quad \|C \odot (UW - X)\|_{\mathrm{F}}^2$$

U is any m-by-r orthonormal matrix such that $\mathrm{col}(U) = \mathscr{U}$.

W is the solution of a simple least squares problem.

The objective is a function of the column space $\mathrm{col}(U)$

$$\min_{\mathscr{U} \in \mathrm{Gr}(m,r)} \underbrace{\min_{W \in \mathbb{R}^{r \times n}} \|C \odot (UW - X)\|_{\mathrm{F}}^2}_{f(\mathscr{U})}$$

U is any m-by-r orthonormal matrix such that $\mathrm{col}(U) = \mathscr{U}$.

W is the solution of a simple least squares problem.

f is not continuous :(

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \|C \odot (UW - X)\|_{\mathrm{F}}^2$$

This stems from unattended entries.

Regularization makes f smooth

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij} \big)^2$$

Regularization makes f smooth

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij}\big)^2 + \lambda \sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

- This is an OptSpace-like regularization;
- Computation of the inner objective looks like it costs $\mathcal{O}(mnr)$ time!

A Matrix 101 trick to reduce computational costs

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij} \big)^2 + \lambda \sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

A Matrix 101 trick to reduce computational costs

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij} \big)^2 + \lambda \sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

$$\sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

A Matrix 101 trick to reduce computational costs

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij} \big)^2 + \lambda \sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

$$\sum_{(i,j) \notin \Omega} (UW)_{ij}^2 + \sum_{(i,j) \in \Omega} (UW)_{ij}^2$$

A Matrix 101 trick to reduce computational costs

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij} \big)^2 + \lambda \sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

$$\sum_{(i,j) \notin \Omega} (UW)_{ij}^2 + \sum_{(i,j) \in \Omega} (UW)_{ij}^2 = \sum_{(i,j)} (UW)_{ij}^2$$

A Matrix 101 trick to reduce computational costs

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij} \big)^2 + \lambda \sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

$$\sum_{(i,j) \notin \Omega} (UW)_{ij}^2 + \sum_{(i,j) \in \Omega} (UW)_{ij}^2 = \sum_{(i,j)} (UW)_{ij}^2$$

$$= \|UW\|_{\mathrm{F}}^2$$

A Matrix 101 trick to reduce computational costs

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij}\big)^2 + \lambda \sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

$$\sum_{(i,j) \notin \Omega} (UW)_{ij}^2 + \sum_{(i,j) \in \Omega} (UW)_{ij}^2 = \sum_{(i,j)} (UW)_{ij}^2$$

$$= \|UW\|_{\mathrm{F}}^2$$

$$= \|W\|_{\mathrm{F}}^2$$

$$\|UW\|_{\mathrm{F}}^2 = \operatorname{trace}\left((UW)^\top UW\right) = \operatorname{trace}\left(W^\top U^\top UW\right) = \operatorname{trace}\left(W^\top W\right) = \|W\|_{\mathrm{F}}^2$$

Complexity: $\mathcal{O}(r(k+n))$.

This (final) objective has many good properties

for the low-rank matrix completion problem

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij} \big)^2 + \lambda \sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

This (final) objective has many good properties

for the low-rank matrix completion problem

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij} \big)^2 + \lambda \sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

- It is natural and defined over the "right" space;

This (final) objective has many good properties

for the low-rank matrix completion problem

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij} \big)^2 + \lambda \sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

- It is natural and defined over the "right" space;
- It has isolated minimizers and it is smooth;

This (final) objective has many good properties

for the low-rank matrix completion problem

$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij}\big)^2 + \lambda \sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

- It is natural and defined over the "right" space;
- It has isolated minimizers and it is smooth;
- It is efficiently computable, and so are $\operatorname{grad} f$ and $\operatorname{Hess} f$;

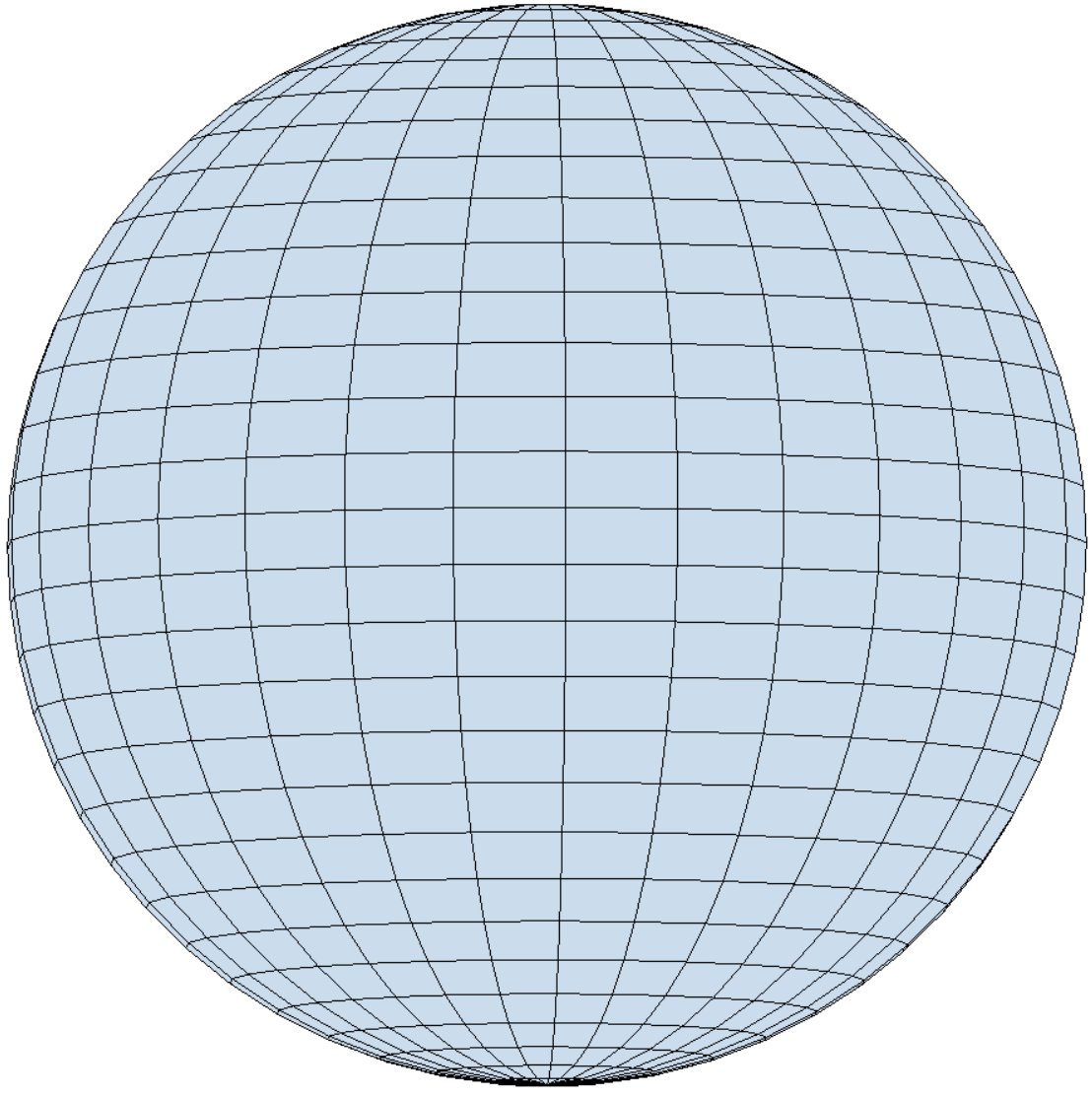
This (final) objective has many good properties

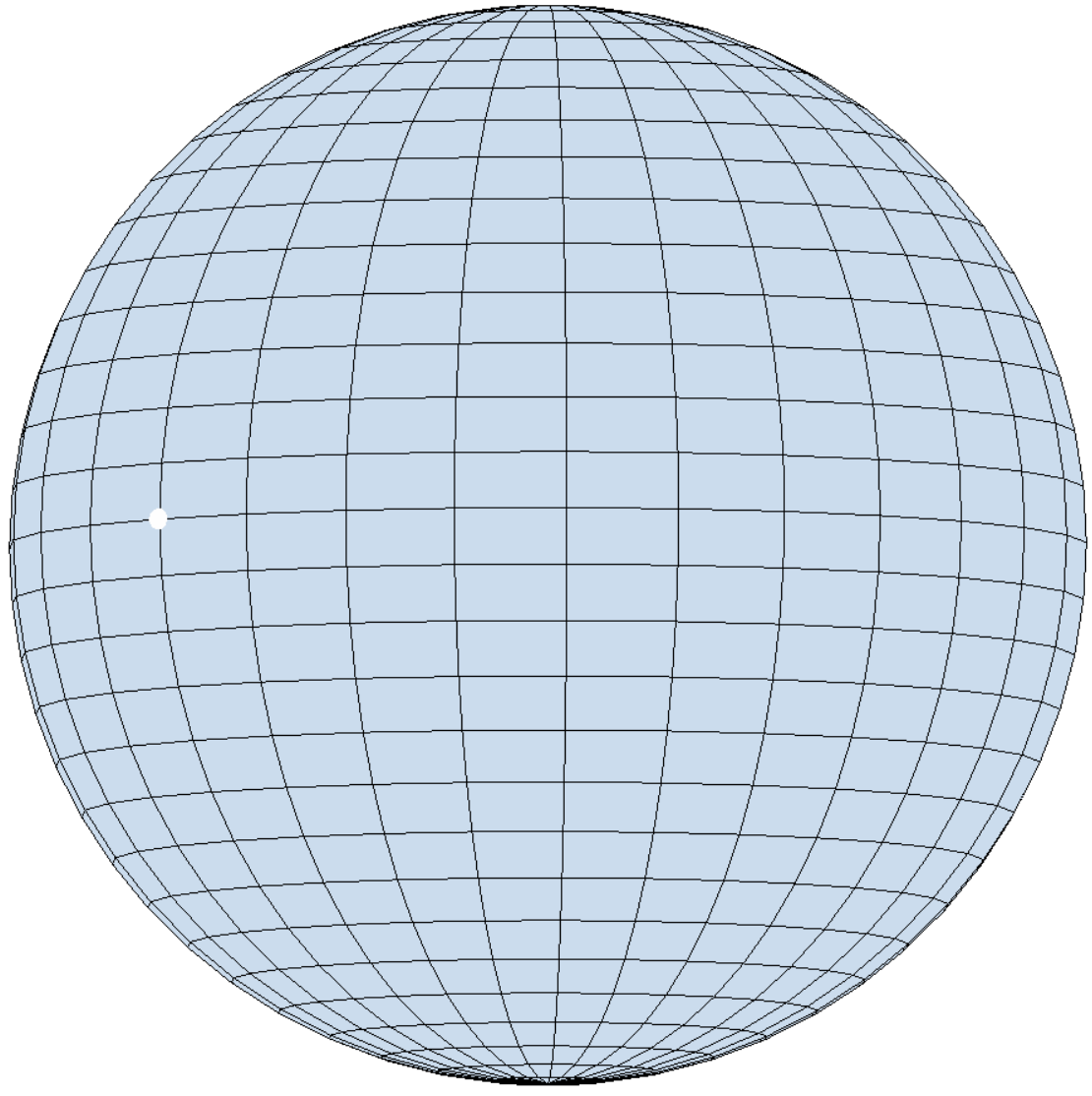
for the low-rank matrix completion problem

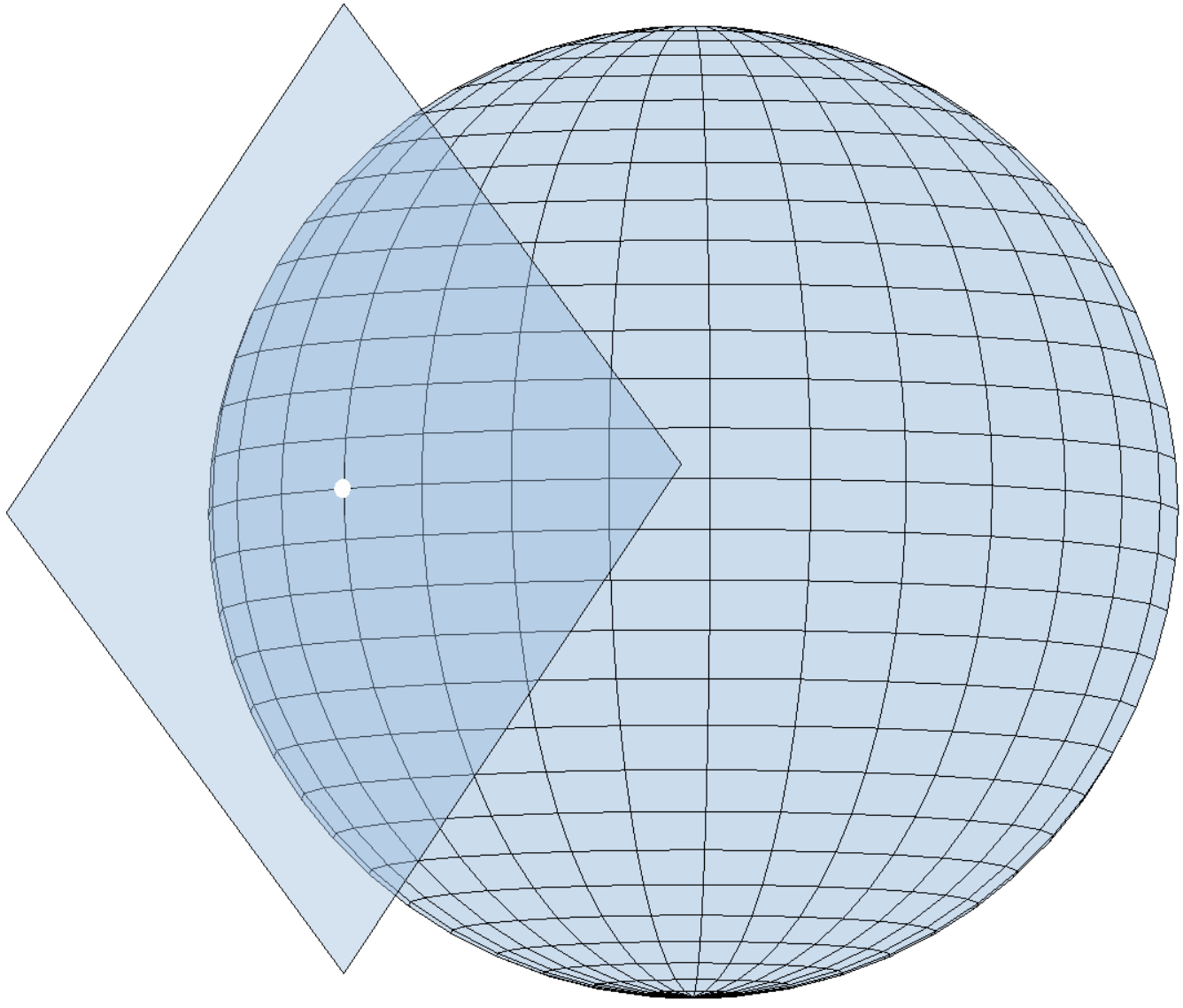
$$f(\mathscr{U}) = \min_{W \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} C_{ij}^2 \big((UW)_{ij} - X_{ij}\big)^2 + \lambda \sum_{(i,j) \notin \Omega} (UW)_{ij}^2$$

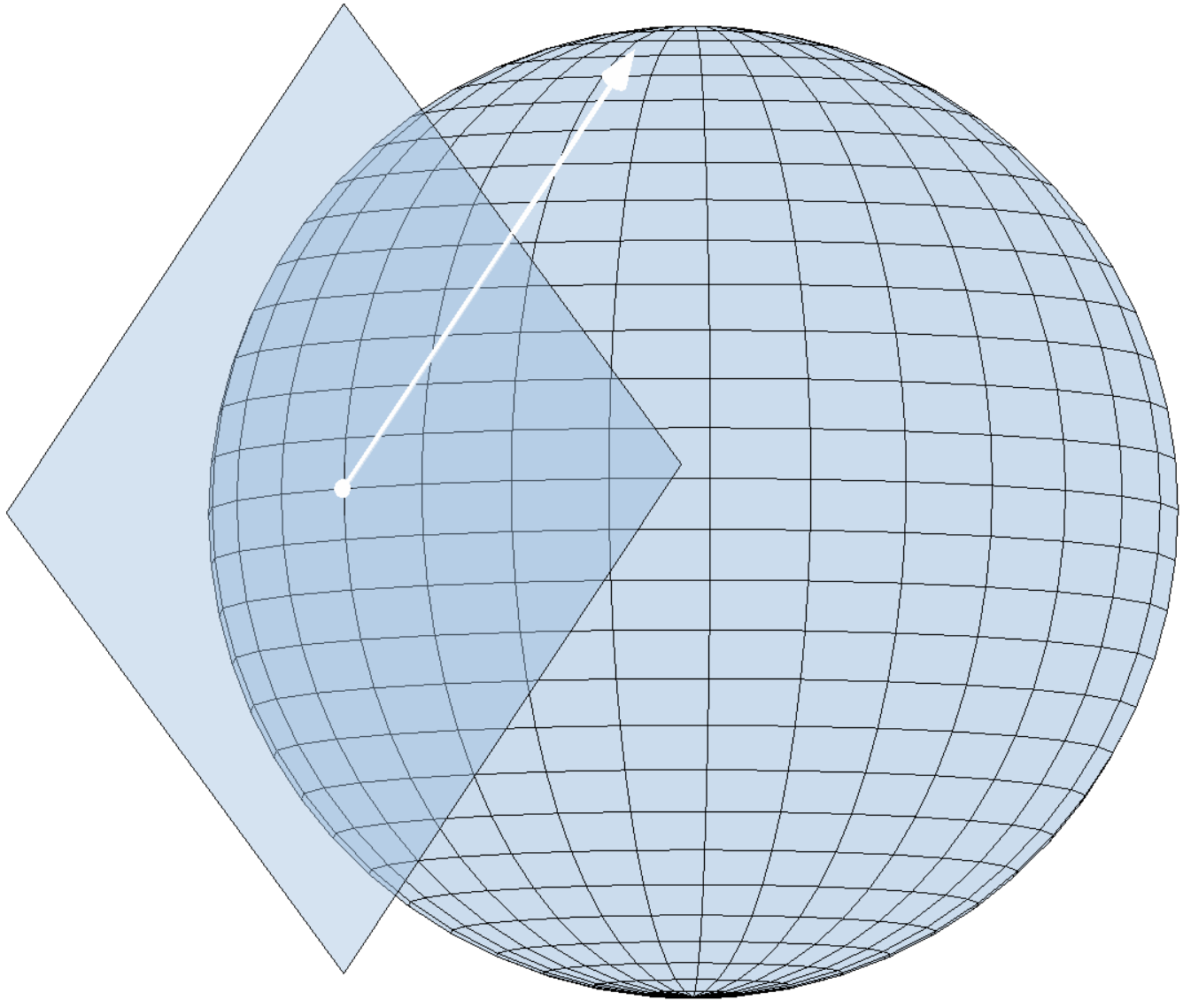
- It is natural and defined over the "right" space;
- It has isolated minimizers and it is smooth;
- It is efficiently computable, and so are $\operatorname{grad} f$ and $\operatorname{Hess} f$;
- It should be able to deal with noise.

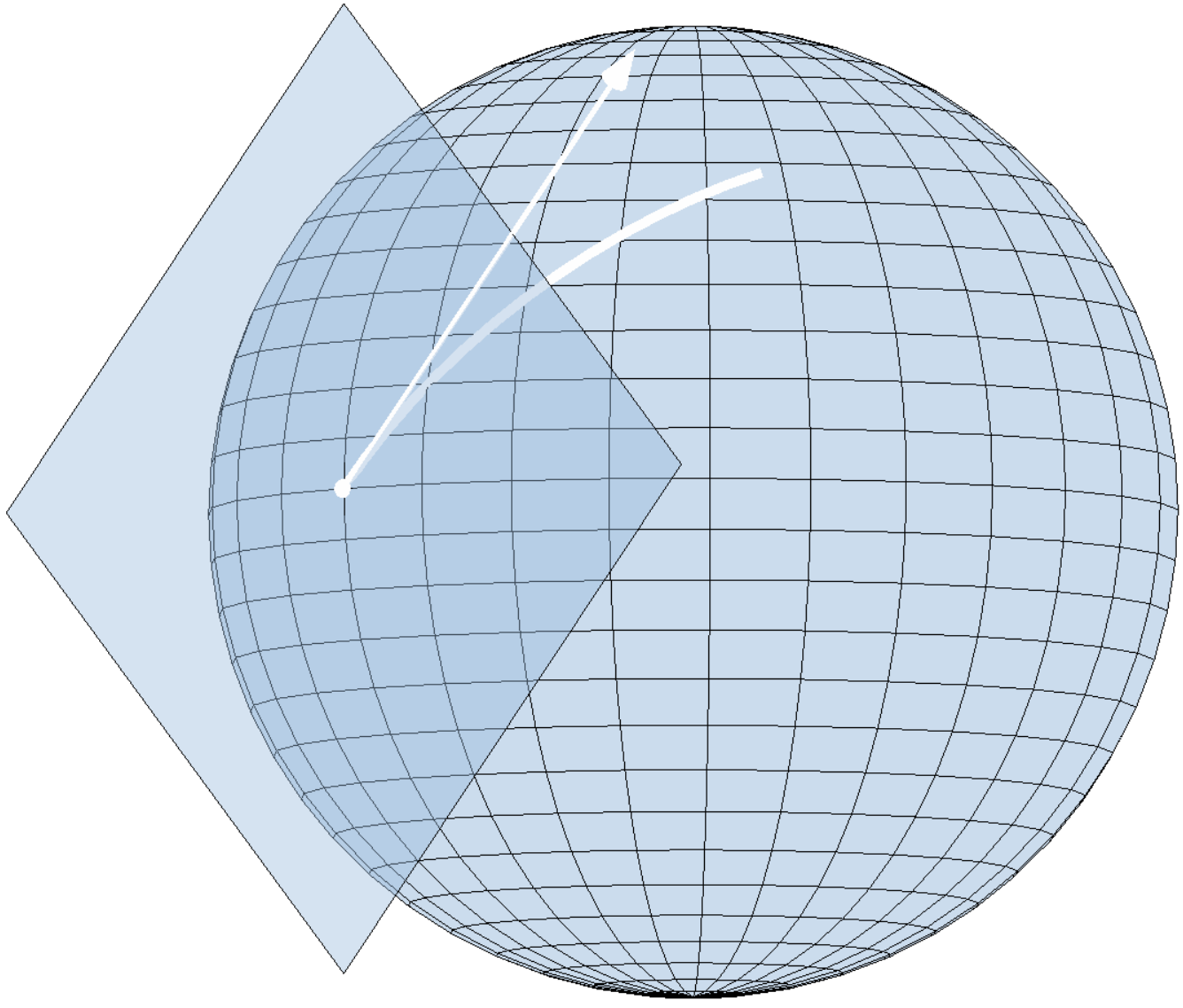
How do you minimize $f(\mathscr{U})$ over Grassmann?











We use a Riemannian trust-region method

called GenRTR

In $\mathbb{R}^n$, we would:

We use a Riemannian trust-region method

called GenRTR

In $\mathbb{R}^n$, we would:

- build a quadratic model of f around the current iterate x,

We use a Riemannian trust-region method

called GenRTR

In $\mathbb{R}^n$, we would:

- build a quadratic model of f around the current iterate x,
- minimize the model in a *trust*-region around x,

We use a Riemannian trust-region method

called GenRTR

In $\mathbb{R}^n$, we would:

- build a quadratic model of f around the current iterate x,
- minimize the model in a *trust*-region around x,
- make the step if it decreases f and rescale the region accordingly.

We use a Riemannian trust-region method

called GenRTR

In $\mathbb{R}^n$, we would:

- build a quadratic model of f around the current iterate x,
- minimize the model in a *trust*-region around x,
- make the step if it decreases f and rescale the region accordingly.

Baker, Gallivan and Pierre-Antoine generalized this to manifolds.

GenRTR comes with proofs

- Our method is guaranteed to converge toward critical points,
- with second-order speed once near the limit point.
- Usually, the limit point is a local minimizer.

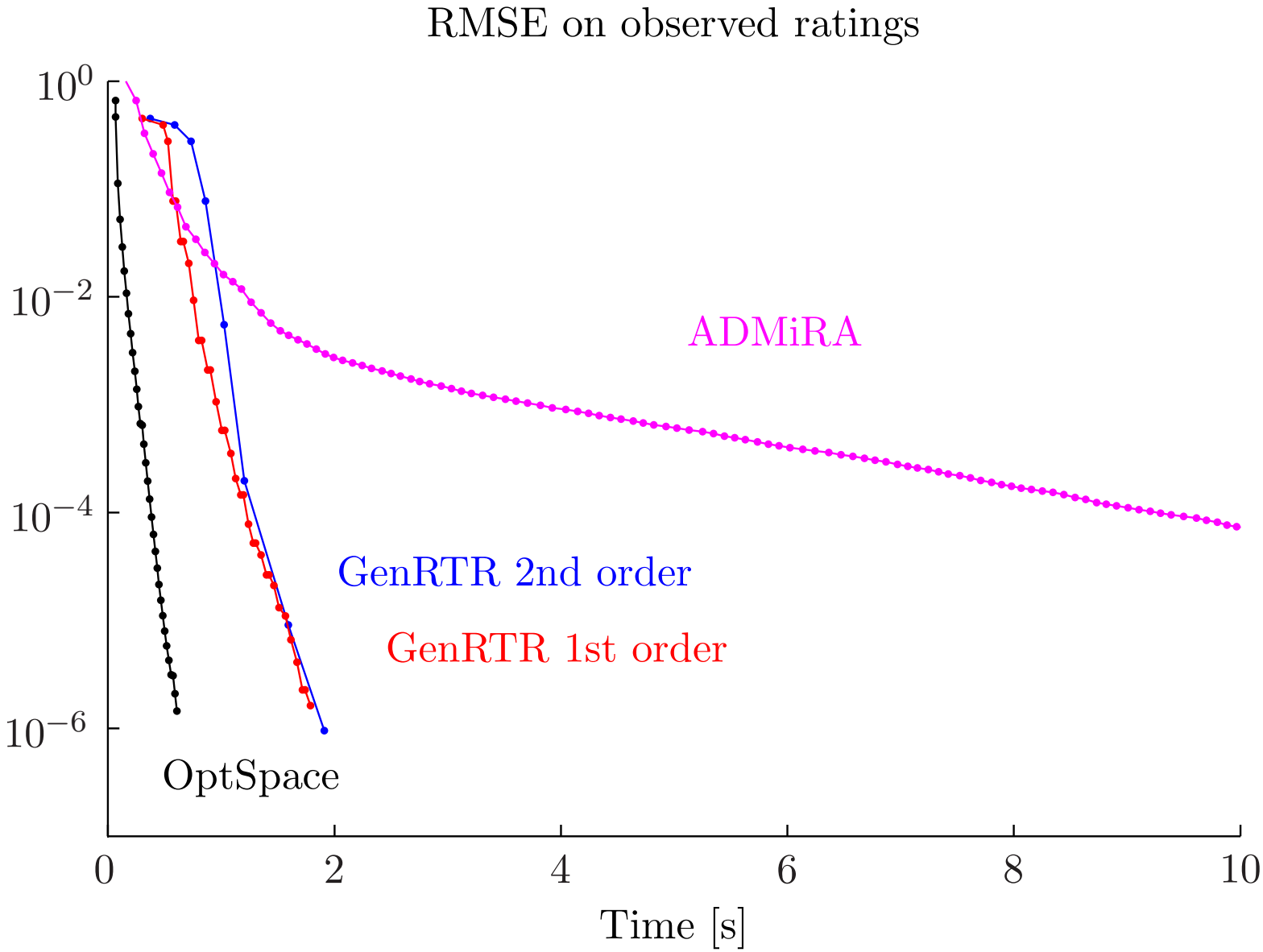
GenRTR is much better than a Newton method

- GenRTR has guaranteed objective decrease;
- can deal with approximate Hessians;
- and with approximate solutions of the Newton equations.

A few tests on synthetic data

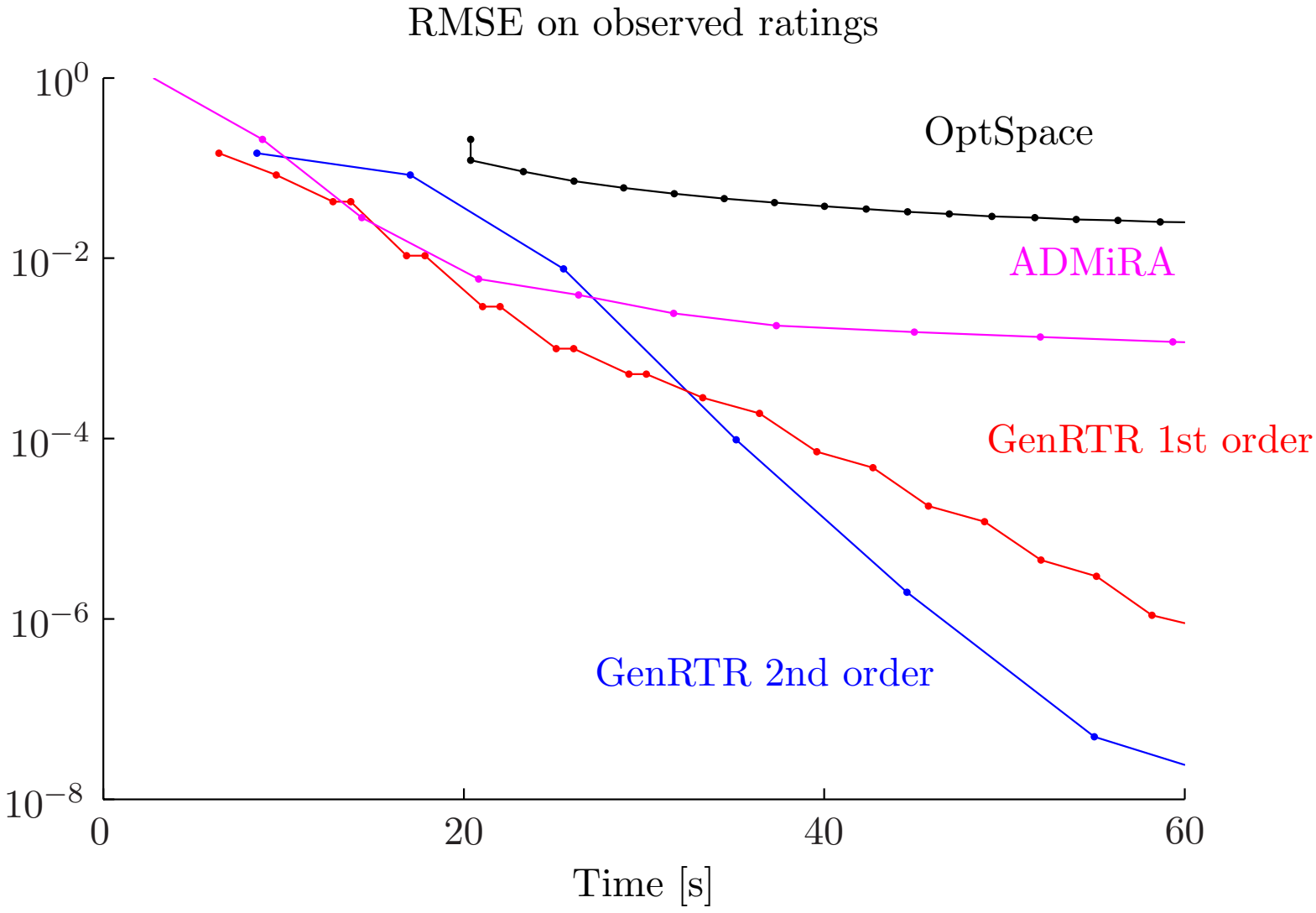
OptSpace is a serious contestant, ADMiRA less so.

Noiseless, $m = n = 1\,000$, $r = 4$, knowl. $= 10\%$



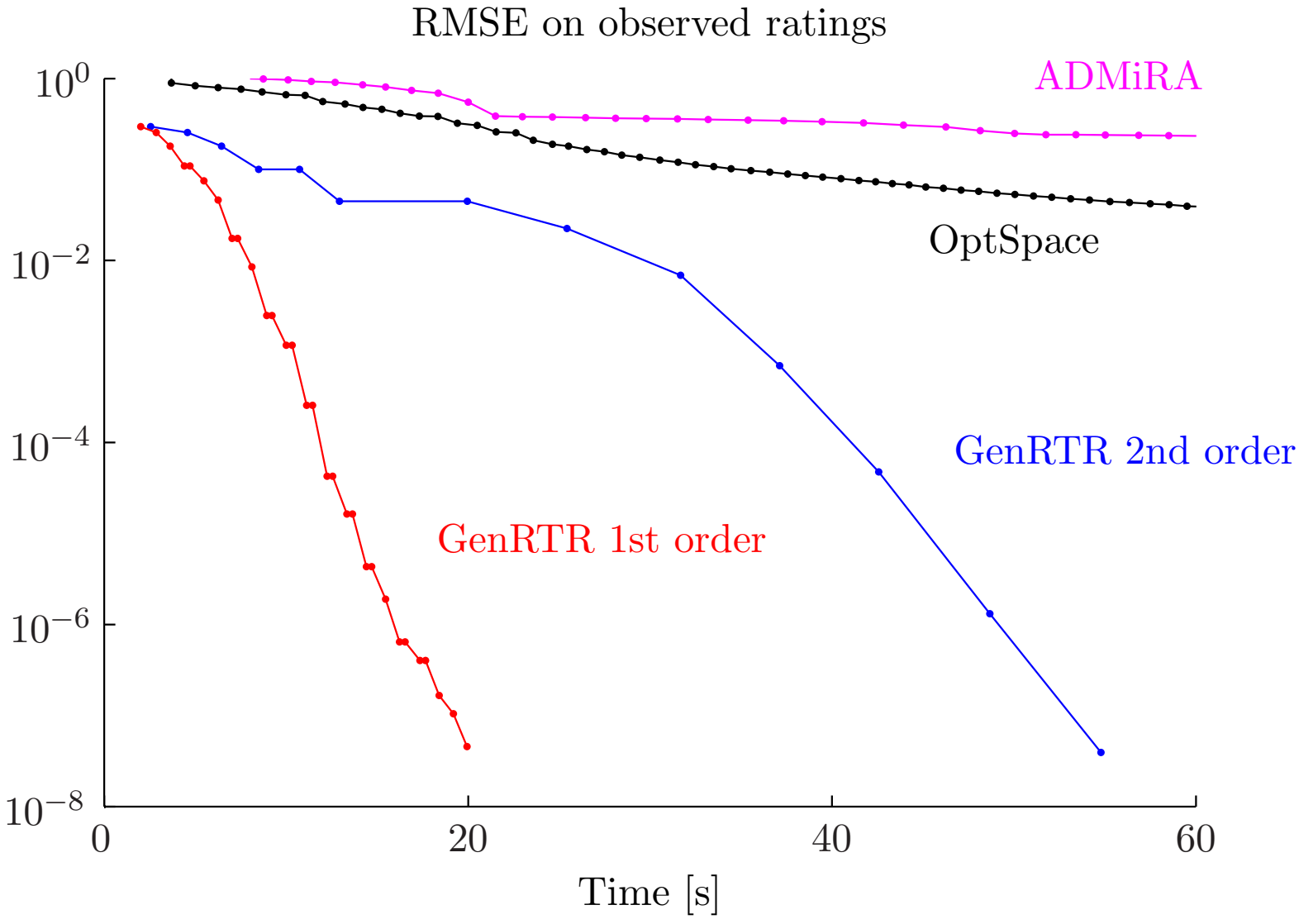
Our method seems to scale better.

Noiseless, $m = n = 10\,000$, $r = 4$, knowl. $= 10\%$



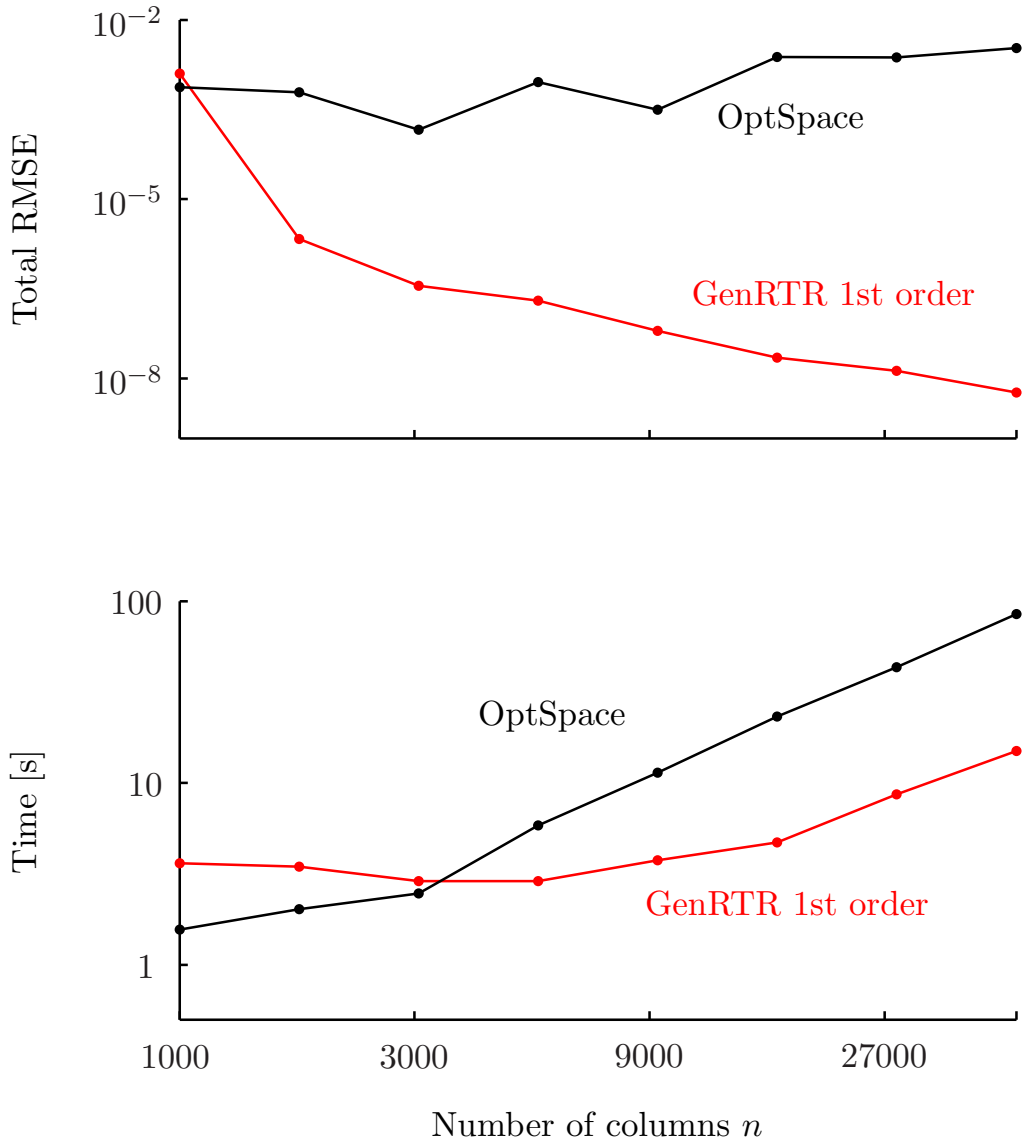
For rectangular matrices, we improve over OptSpace.

Noiseless, $m = 1\,000$, $n = 80\,000$, $r = 4$, knowl. $= 2\%$



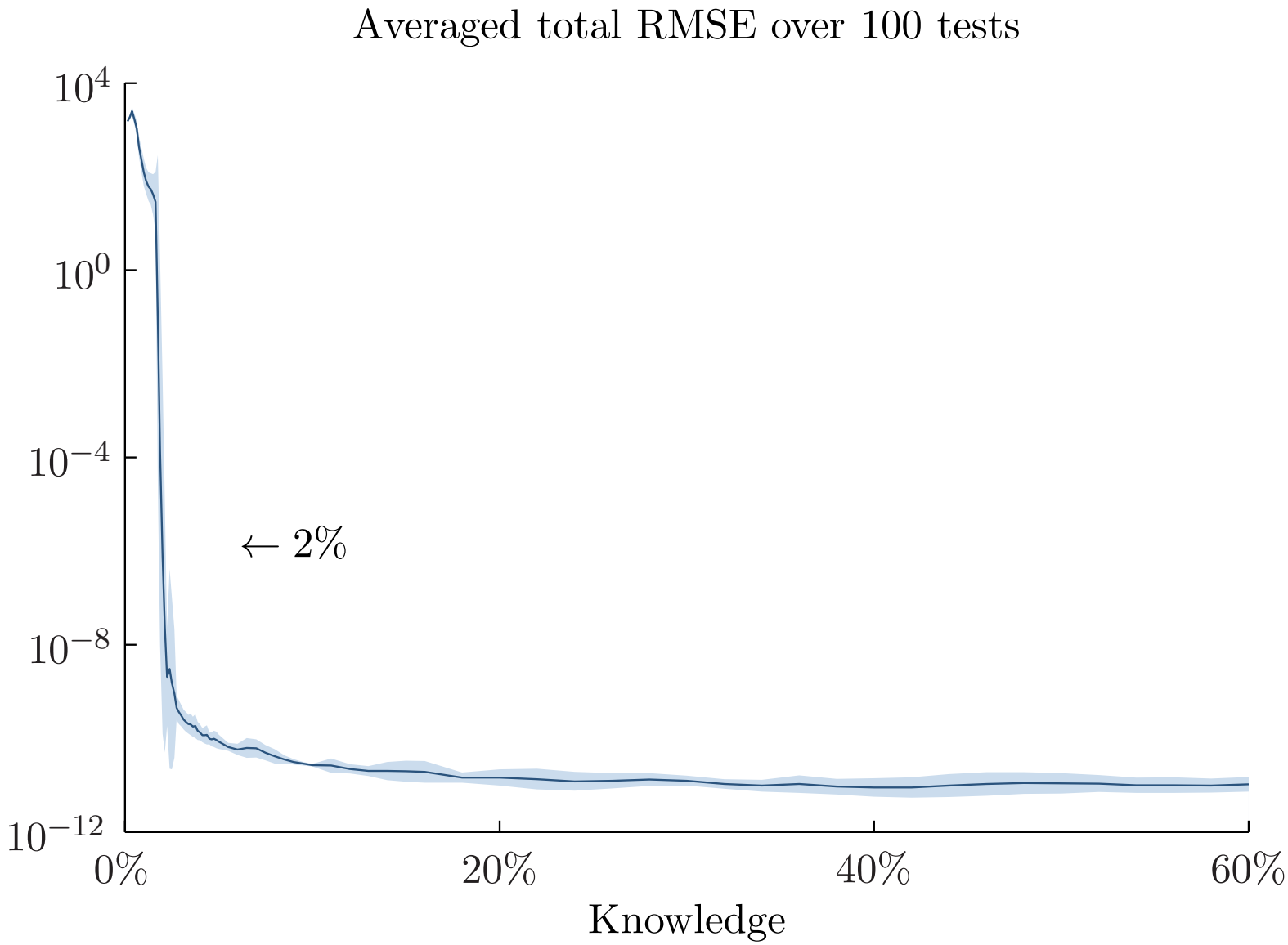
Rectangular matrices are important in applications.

Noiseless, $m = 1\,000$, $r = 3$, knowl. $= 2\%$



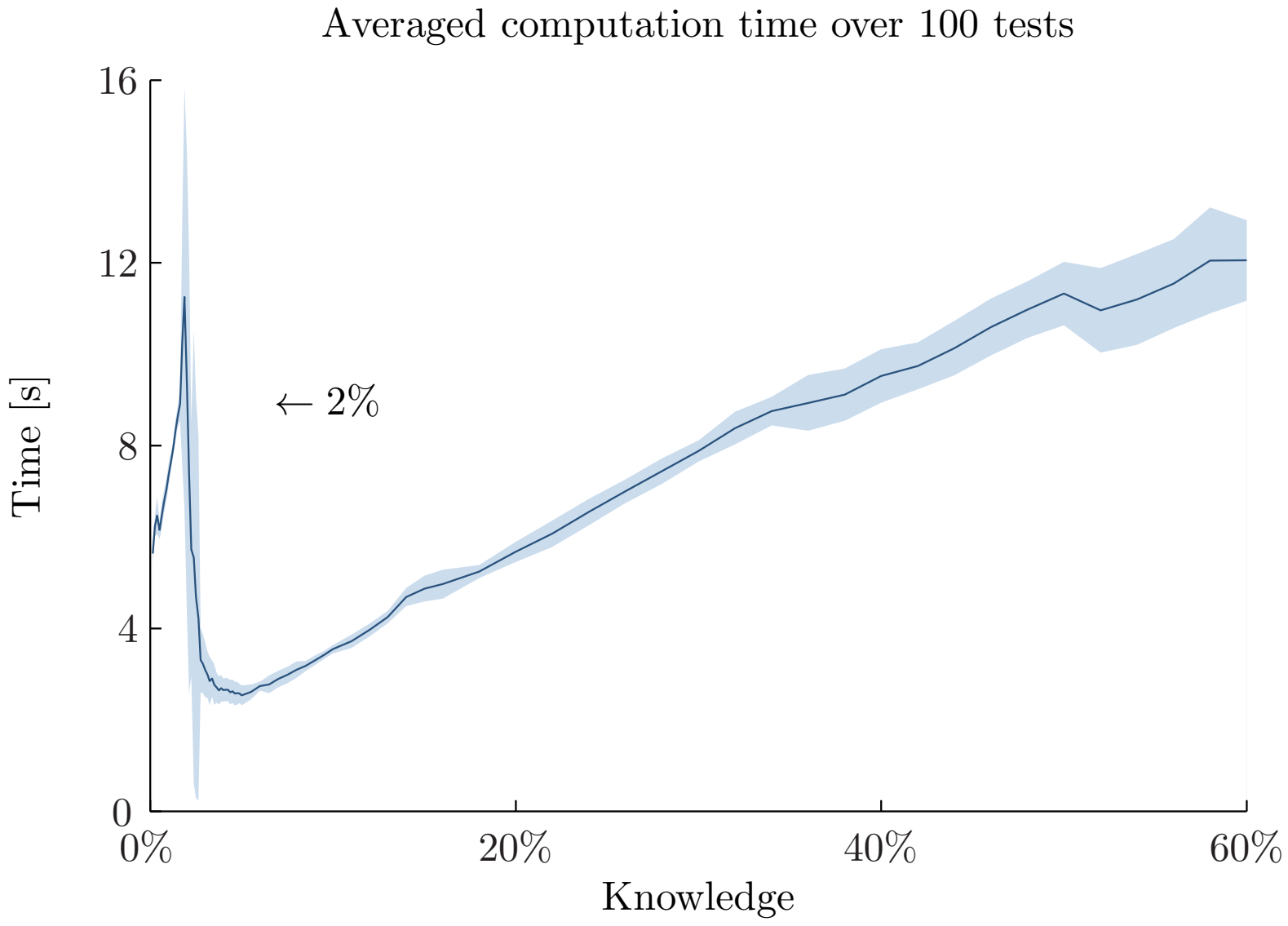
Given enough information, we consistently recover X.

Noiseless, $m = n = 1\,000$, $r = 5$, $\lambda = 10^{-12}$



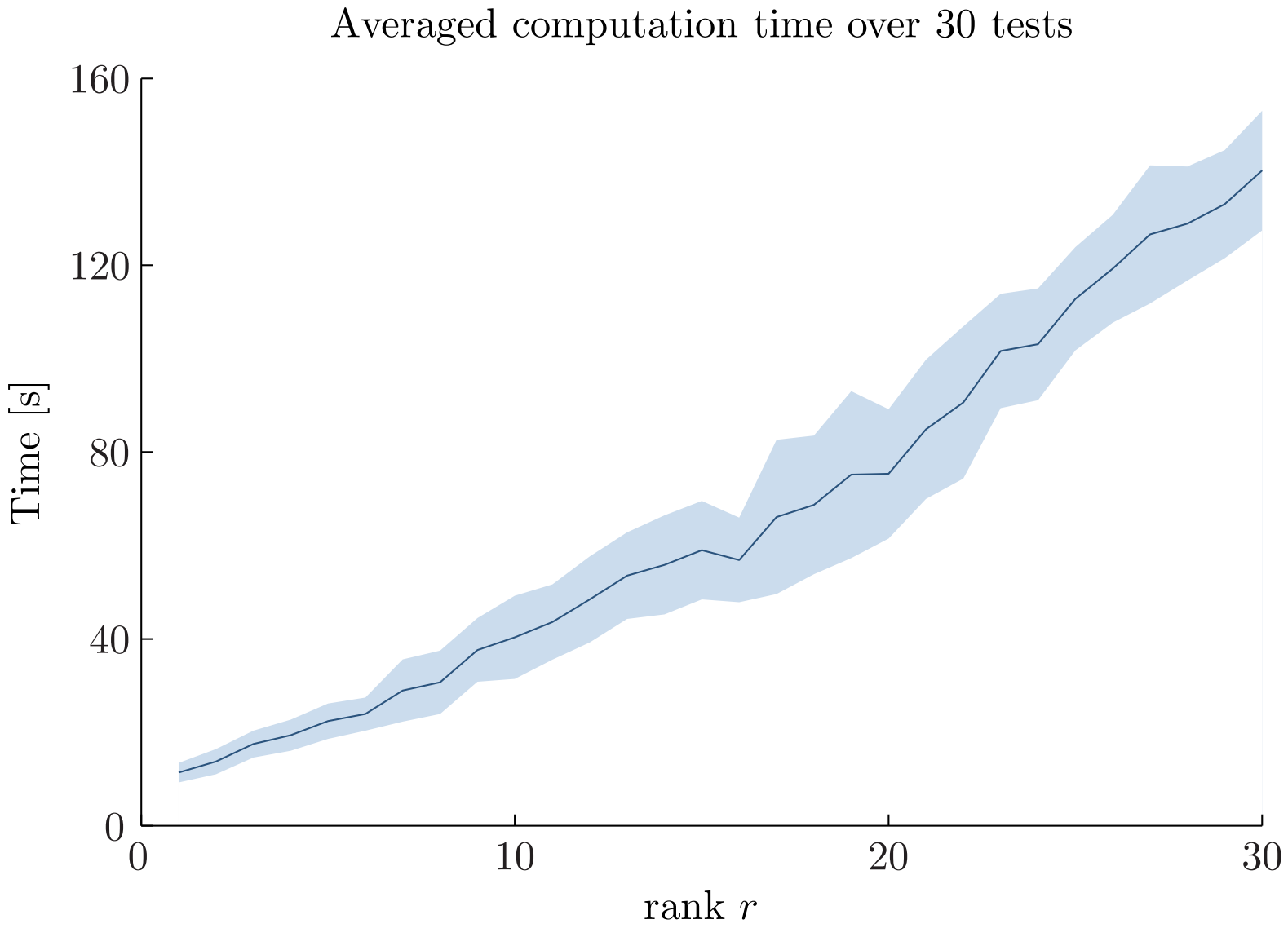
Computation time is proportional to ♯ known entries.

Noiseless, $m = n = 1\,000$, $r = 5$, $\lambda = 10^{-12}$



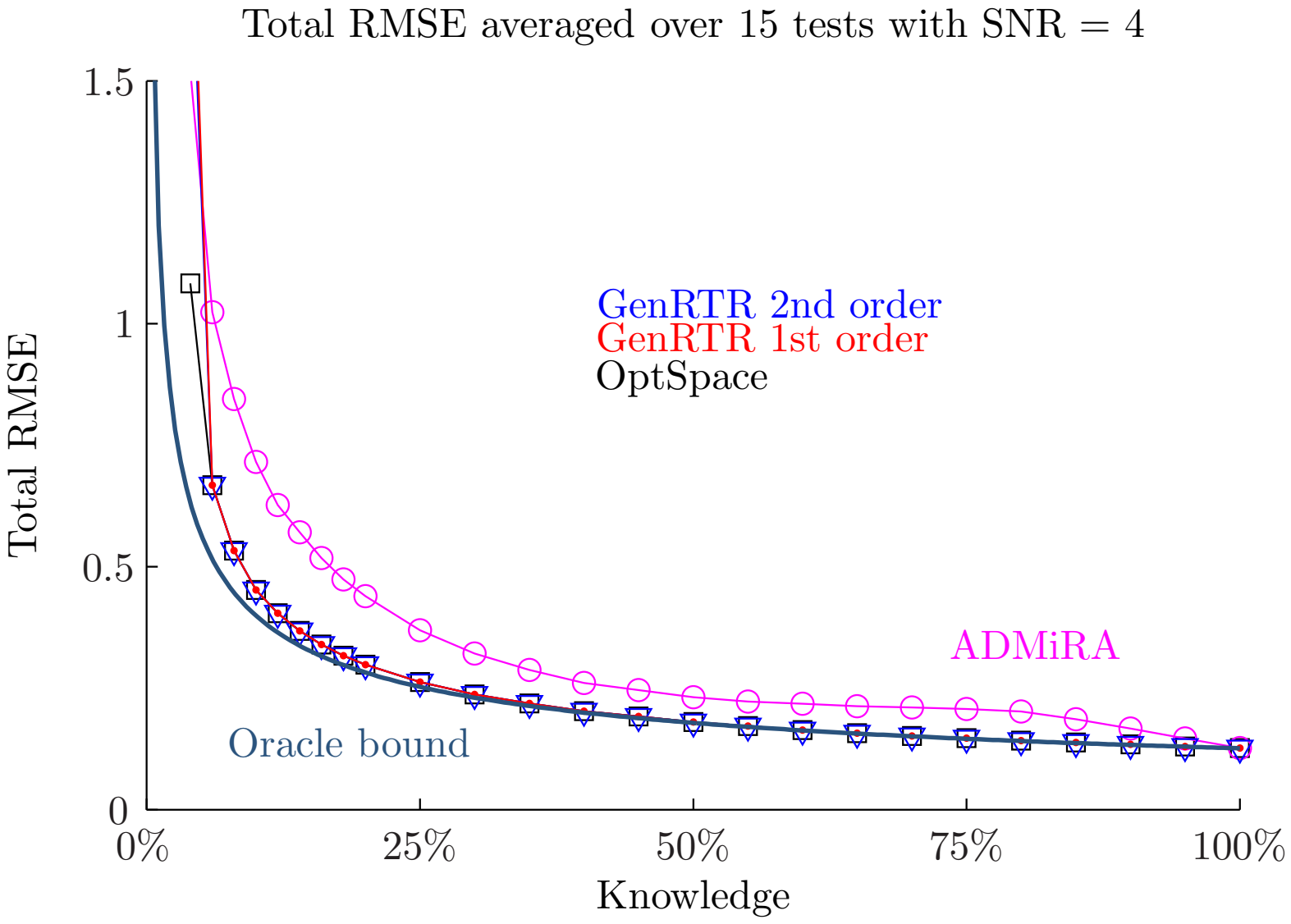
Computation time scales reasonably with the rank.

Noiseless, $m = n = 1\,000$, knowl. $= 25\%$



In the presence of noise, we are close to optimal.

Noisy, $m = n = 500$, $r = 4$



A few tests on real data

The Jester data set

100 jokes rated continuously between -10 and 10 by 4000 users

OptSpace	GenRTR	Trimmed SVD	ADMiRA	Nearest neighbors
0.1604	0.1605	0.1644	0.1660	0.1870

Normalized Mean Absolute Error (NMAE) scores averaged over 10 splits.

In each split, we retain two ratings for each user as validation data.

The small MovieLens data set

1682 movies rated between 1 and 5 by 943 users

OptSpace	GenRTR 1	GenRTR 2	ADMiRA	Medium guess
0.1809	0.1821	0.1822	0.2040	0.2504

Normalized Mean Absolute Error scores averaged over a 5-fold crossval.

In each split, we retain 20% of the ratings as validation data.

Final thoughts

What I overlooked

How do we accommodate new users / items?

What about on-line methods rather than a batch approach?

Gilles Meyer and Rodolphe Sepulchre have good results.

And the Netflix and bigger MovieLens data sets?

Are second-order methods really useful?

This is work in progress.

Conclusions

Our contribution is a sensible objective function with the right complexity fed to a state-of-the-art theory-backed solver

The method works smoothly for the low-rank matrix completion problem

The method is competitive on real data (but lacks validation for now)

It's a hot topic