Reexamining Low Rank Matrix Factorization for Trace Norm Regularization

Carlo Ciliberto[1], Dimitris Stamos[1], Massimilano Pontil[1,2]

massimiliano.pontil@iit.it

(1) CSML, Istituto Italiano di Tecnologia, Italy
(2) Dept of Computer Science, University College London, UK

Optimization and Big Data Summer School, Veroli, July 3-7, 2017

Matrix completion

The problem

$$\min_W \underbrace{\sum_t EmpError(w_t; D_t)}_{\ell(W)} + \lambda \|W\|_*$$

is of the form[1]:

$$\min_{W \in \mathbb{R}^{n \times m}} f_\lambda(W), \qquad f_\lambda(W) = \ell(W) + \lambda \|W\|_*$$

- $\ell : \mathbb{R}^{n \times m} \to \mathbb{R}$ **data fitting** term. Example: matrix completion

  $$\ell(W) = \|M \odot (Y - W)\|_F^2$$

  $Y \in \mathbb{R}^{n \times m}$ true matrix, $M \in \mathbb{R}^{n \times m}$ binary matrix of "observations".

[1] for the next few slides the matrix W is $m \times n$ rather than $d \times T$

Proximal gradient method

If ℓ is **convex and differentiable**, with Lipschitz continuous gradient

$$\min_{W \in \mathbb{R}^{n \times m}} \quad \ell(W) + \lambda \|W\|_*$$

can be solved by **forward-backward splitting** (PFB) [**?**]

Starting from any $W_0 \in \mathbb{R}^{n \times m}$, compute

$$W_{k+1} = \mathrm{Prox}_{\gamma\lambda\|\cdot\|_*}\left(W_k - \gamma \nabla \ell(W_k)\right)$$

with $\gamma > 0$ a suitable descent step.

Proximity operator of the trace norm

For any $W \in \mathbb{R}^{n \times m}$ of rank r and Singular Value Decomposition (SVD)

$$W = U \Sigma V^\top$$

- $U \in \mathbb{R}^{n \times r}$, $V \in \mathbb{R}^{m \times r}$ with orthonormal columns, and
- $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$, $\sigma_i > 0$ for $i = 1, \ldots, r$.

we have

$$\mathrm{Prox}_{\gamma\lambda}(W) = U \ \mathrm{diag}\big(s_{\gamma\lambda}(\sigma_1), \ldots, s_{\gamma\lambda}(\sigma_r)\big) \ V^\top$$

where $s_{\gamma\lambda} : \mathbb{R}_+ \to \mathbb{R}$ is the **soft thresholding** operator such that

$$s_{\gamma\lambda}(\sigma) = \begin{cases} \sigma - \lambda\gamma & \text{if } \sigma > \lambda\gamma \\ 0 & \text{otherwise} \end{cases}$$

Matrix learning with PFB

PROs.

- Convex Problem $\Rightarrow$ converges to global minimizer
- Convergence rate $O(1/k)$ (faster with accelerated methods)

CONs.

- One SVD per iteration! $O(nm\min(n,m))$ time complexity.
- $O(nm)$ memory requirement even when the solution is low rank.

Factorization based methods

Variational form for the nuclear norm

$$\|W\|_* = \frac{1}{2} \min \left\{ \|A\|_F^2 + \|B\|_F^2 \right\}$$
$$\text{s.t.} \quad r \in \mathbb{N}, \quad A \in \mathbb{R}^{n \times r}, \quad B \in \mathbb{R}^{m \times r}, \quad W = AB^\top$$

$\Rightarrow$ alternative problem in A and B

$$\min_{\substack{A \in \mathbb{R}^{n \times r} \\ B \in \mathbb{R}^{m \times r}}} g_{\lambda,r}(A, B), \qquad g_{\lambda,r}(A, B) = \ell(AB^\top) + \frac{\lambda}{2}(\|A\|_F^2 + \|B\|_F^2)$$

where r is now a *hyperparameter*.

Theorem. *The problems of minimizing f_λ and $g_{\lambda,r}$ are equivalent for sufficently large r.*

Matrix learning with factorization based

PROs.

- Smooth functional $\Rightarrow$ we can use any first or second order optimization method, e.g. Gradient Descent (GD):

$$A_{k+1} = A_k - \gamma\big(\nabla(A_k B_k^\top)B_k + \lambda A_k\big)$$
$$B_{k+1} = B_k - \gamma\big(\nabla(A_k B_k^\top)^\top A_k + \lambda B_k\big)$$

- $O(nmr)$ time complexity per iteration
- $O((m+n)r)$ space complexity

CONs.

- Not convex: Guarantees for global optimality?
- In practice r is unknown. How to find it?

A Meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

A Meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

A Meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

2. If the largest singular value of $\nabla \ell(A_r B_r^\top)$ is less than λ: **Stop**.

A Meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

2. If the largest singular value of $\nabla\ell(A_r B_r^\top)$ is less than λ: **Stop**.

3. Perform a step in a **descent direction*** for $g_{\lambda,r+1}$ from $([A_r\ 0], [B_r\ 0])$ to a point (A'_r, B'_r), $A'_r \in \mathbb{R}^{n\times(r+1)}$, $B'_r \in \mathbb{R}^{m\times(r+1)}$

A Meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

2. If the largest singular value of $\nabla\ell(A_r B_r^\top)$ is less than λ: **Stop**.

3. Perform a step in a **descent direction*** for $g_{\lambda,r+1}$ from $([A_r\ 0], [B_r\ 0])$ to a point (A'_r, B'_r), $A'_r \in \mathbb{R}^{n\times(r+1)}, B'_r \in \mathbb{R}^{m\times(r+1)}$

4. Increase r to $r+1$ and go back to Step 1.

A Meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

2. If the largest singular value of $\nabla\ell(A_r B_r^\top)$ is less than λ: **Stop**.

3. Perform a step in a **descent direction*** for $g_{\lambda,r+1}$ from $([A_r\ 0], [B_r\ 0])$ to a point (A'_r, B'_r), $A'_r \in \mathbb{R}^{n\times(r+1)}, B'_r \in \mathbb{R}^{m\times(r+1)}$

4. Increase r to $r+1$ and go back to Step 1.

This procedure is bound to stop **at most** for $r = \min(n, m)$.
(in practice it stops much earlier)

A criterion for global optimality

Theorem. *Let* $(\bar{A}, \bar{B})$ *be a critical point for* $g_{\lambda,r}$. *Then* $\bar{W} = \bar{A}\bar{B}^\top$ *is a global minimizer for* f_λ *if and only if*

$$\|\ell(\bar{W})\| \leq \lambda$$

where $\|\cdot\|$ *denotes the operator norm (i.e. the largest singular value).*

Note. In general it is NP-hard to determine whether a critical point of a *non-convex* function is a global minimizer.

Note. The operator norm requires only the largest singular value $\Rightarrow$ significantly faster than the full SVD.

A Meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

2. If the largest singular value of $\nabla\ell(A_r B_r^\top)$ is less than λ: **Stop**.

3. Perform a step in a **descent direction*** for $g_{\lambda,r+1}$ from $([A_r\ 0], [B_r\ 0])$ to a point (A'_r, B'_r), $A'_r \in \mathbb{R}^{n\times(r+1)}, B'_r \in \mathbb{R}^{m\times(r+1)}$

4. Increase r to $r+1$ and go back to Step 1.

This procedure is bound to stop **at most** for $r = \min(n, m)$.
(in practice it stops much earlier)

Escaping from critical points

Theorem. *Let* (A, B) *be a critical point of* $g_{\lambda,r}$ *with* $\|\nabla \ell(AB^\top)\| > \lambda$ *(i.e. not a global minimizer). If* $\operatorname{rank}(A) < r$ *(equivalently* $\operatorname{rank}(B)$*) then* (A, B) *is a* **strict saddle** point *(i.e. the corresponding Hessian has at least one negative eigenvalue)*.

- For $r > \min(n, m)$ it is always true that $\operatorname{rank}(A) < r \Rightarrow$ every critical point is either a global minimizer or a strict saddle! But...
- *Gradient descent* **does not** *converge to strict saddle points.* (Lee et al. 2016).

Corollary. *For* $r > \min(n, m)$*, gradient descent applied to* $g_{\lambda,r}$ **converges only to global minimizers**.

A Meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).
2. If the largest singular value of $\nabla\ell(A_r B_r^\top)$ is less than λ: **Stop**.
3. Perform a step in a **descent direction*** for $g_{\lambda,r+1}$ from $([A_r\ 0], [B_r\ 0])$ to a point (A'_r, B'_r), $A'_r \in \mathbb{R}^{n\times(r+1)}, B'_r \in \mathbb{R}^{m\times(r+1)}$
4. Increase r to $r+1$ and go back to Step 1.

This procedure is bound to stop **at most** for $r = \min(n, m)$.
(in practice it stops much earlier)

Escaping from critical points

Corollary. *Let u and v be any two left and right singular vectors of $\nabla \ell(AB^\top)$ associated to a singular values* larger than λ. *Then, for any any $q \in \mathbb{R}^r$ for which $Aq = 0$, we have $Bq = 0$ (and vice-versa). Moreover, if $\operatorname{rank}(A) < r$, then $(uq^\top, vq^\top)$ is a* descent direction *for $g_{\lambda,r}$ at (A, B).*

- If A and B are not full rank, we can explicitly find escape direction!
- If A and B are full rank, then $[A,\ 0], [B,\ 0]$:
  - is a critical point for $g_{\lambda,r}$
  - is not full rank
- We can "inflate" the problem by one column, from r to $r+1$, and find an escape direction!

A Meta-algorithm for fast large-scale matrix learning

Let $r = 1$, $A'_0 \in \mathbb{R}^n$, $B'_0 \in \mathbb{R}^m$. Then,

1. **Apply GD** (or other descent methods) from (A'_{r-1}, B'_{r-1}) to minimize $g_{\lambda,r}$ until convergence to a critical point (A_r, B_r).

2. If the largest singular value of $\nabla\ell(A_r B_r^\top)$ is less than λ: **Stop**.

3. Perform a step in a **descent direction*** for $g_{\lambda,r+1}$ from $([A_r\ 0], [B_r\ 0])$ to a point (A'_r, B'_r), $A'_r \in \mathbb{R}^{n\times(r+1)}, B'_r \in \mathbb{R}^{m\times(r+1)}$

4. Increase r to $r+1$ and go back to Step 1.

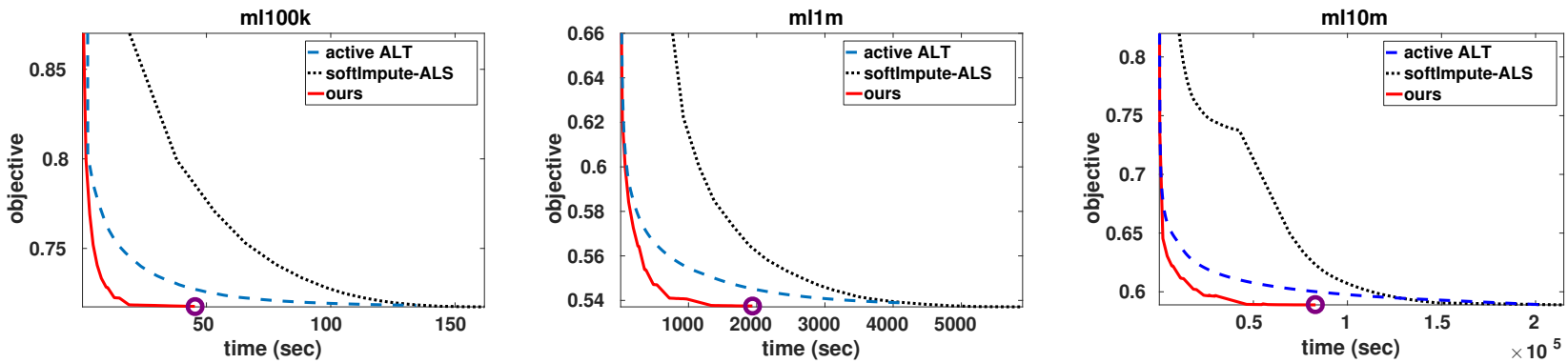
This procedure is bound to stop **at most** for $r = \min(n, m)$.
(in practice it stops much earlier)

Experiments - Movielens

- Movielens 100k (ml100k), 1M (ml1m) and 10M (ml10m).
- Movies rated (1 to 5) by users. Large scale matrix factorization:

$$\ell(W) = \|M \odot (Y - AB^\top)\|_F^2$$

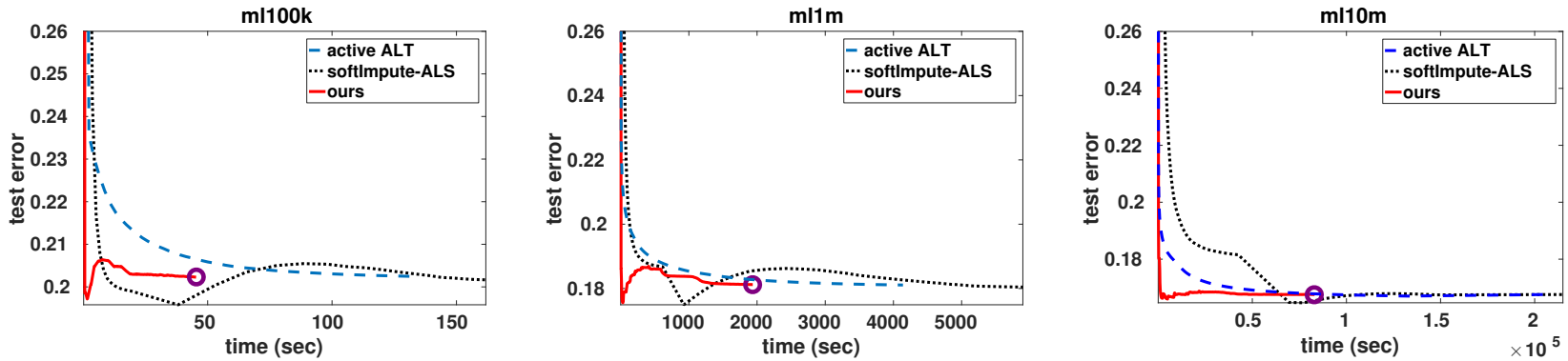
- 50% user ratings for training, 25% validation, 25% test.



Competitors: Active ALT (Hsieh et al. 2014), ALS Soft-impute (Hastie et al. 2015)

Experiments - Movielens (test error)

Normalized Mean Absolute Error (NMAE): mean of the entry-wise absolute errors normalized by the maximum discrepancy $\max_{i,j}(Y_{ij}) - \min_{i,j}(Y_{ij})$.



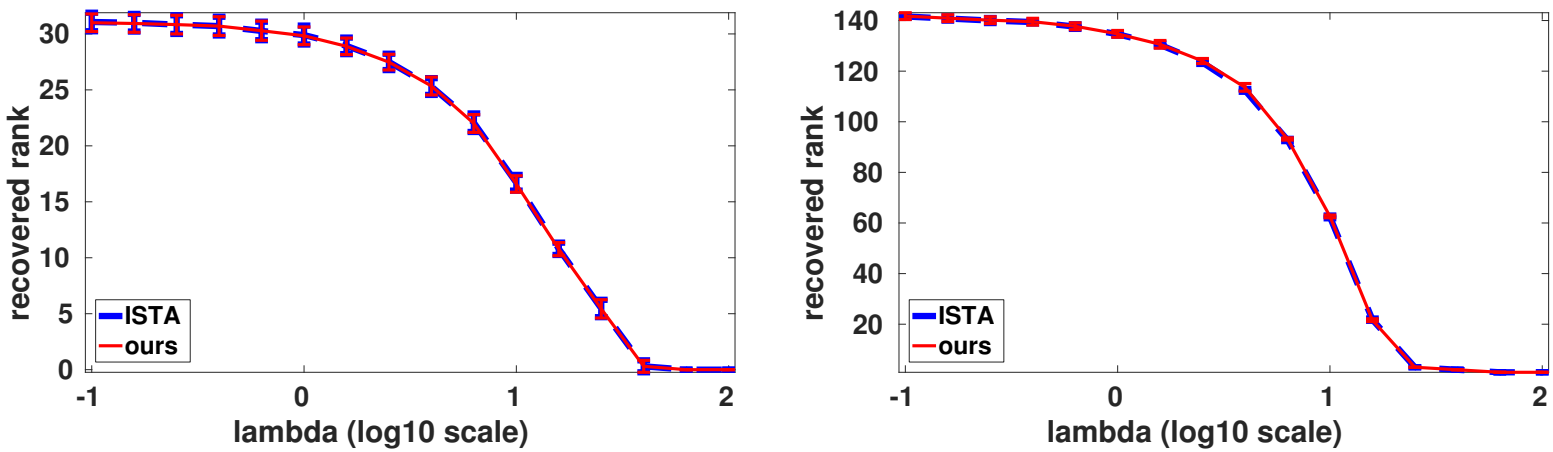
	ml100k			ml1m			ml10m		
	NMAE	**time(s)**	**rank**	**NMAE**	**time(s)**	**rank**	**NMAE**	**time(s)**	**rank**
ALT	0.2165	97	93	0.1806	4133	179	0.1670	205023	225
ALS-SI	0.1956	40	16	0.1749	832	31	0.1648	51205	36
Ours	0.1959	2	11	0.1751	39	25	0.1659	3150	41

Massimiliano Pontil, Veroli 2017

Experiments - recovered rank

Goal. Verify when the condition for global optimality activates: at the "true" rank or much later?

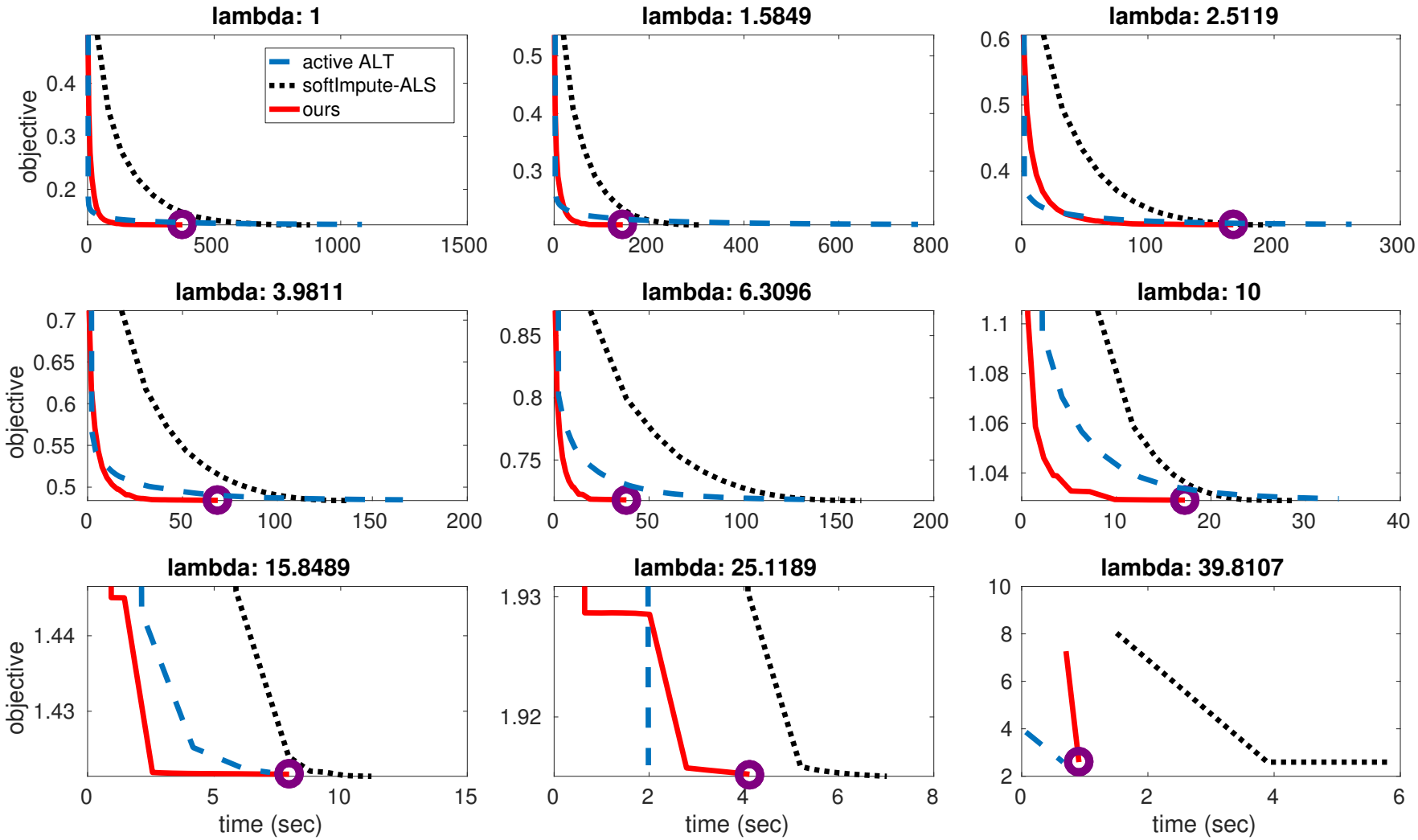
- **Synthetic (Left)**: random 100×100 matrix $Y = AB^\top + E$, product of two 100×10 matrices plus Gaussian noise E on the entries.
- **Movielens 100k (Right)**: 943×1682 matrix with 100k available ratings.



Comparison with rank recovered by Proximal method (ISTA).

Massimiliano Pontil, Veroli 2017

Bonus - more lambdas! (objective)



Massimiliano Pontil, Veroli 2017

Reference for this material

C Ciliberto, D Stamos, M Pontil. Reexamining Low Rank Matrix Factorization for Trace Norm Regularization. arXiv preprint arXiv:1706.08934.