A Gauss-Newton Algorithm for Symmetric Low-Rank Product Optimization

Xin Liu

State Key Laboratory of Scientific and Engineering Computing
Institute of Computational Mathematics and Scientific/Engineering Computing
Academy of Mathematics and Systems Science
Chinese Academy of Sciences, China

(Joint work with **Zaiwen Wen**[1], **Yin Zhang**[2])

Southern California Optimization Day (SCOD14), UCSD
May 23, 2014

[1] Peking University, China
[2] Rice University, U.S.

Original Motivation

Given $B \in \mathbb{R}^{n\times m}$ and $k < \min(n, m)$,

$$\min_{X,Z} \left\{ \|XZ^{\mathsf{T}} - B\|_{\mathrm{F}} : X \in \mathbb{R}^{n\times k}, Z \in \mathbb{R}^{m\times k} \right\}$$

Closely related to SVD

- k-Dominant (k-D) SVD of $n \times m$ $\Rightarrow$ solution
- Solution + QR and SVD of $n \times k$ $\Rightarrow$ k-D SVD of $n \times m$

How about the symmetric case? for $A = A^{\mathsf{T}} \in \mathbb{R}^{n\times n}$ (e.g., $A = BB^{\mathsf{T}}$),

$$\min_{X} \left\{ \|XX^{\mathsf{T}} - A\|_{\mathrm{F}} : X \in \mathbb{R}^{n\times k} \right\}$$

Symmetric Low-Rank Product Optimization

A nonlinear, nonconvex least squares problem

$$\min_{X \in \mathbb{R}^{n \times k}} \|XX^{\mathsf{T}} - A\|_{\mathrm{F}}^2$$

Fundamental in low-rank matrix approximations

- Principal subspace of A:

$$\mathrm{span}(X) = \mathrm{span}\{q_1, q_2, \ldots, q_k\}$$

  where $\{q_j\}_{j=1}^k$ are dominant eigenvectors of A.
- For $A = BB^{\mathsf{T}}$, columns of X are "principal components" of B.

Notation

- Let eigenvalues of A be in a descending order

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

- Eigenvalue Decomposition:

$$A = Q_n \Lambda_n Q_n^{\mathsf{T}}, \quad Q_n^{\mathsf{T}} Q_n = I, \quad \Lambda_n \text{ diagonal}$$

- k-D principal eigenspace:

$$\text{span}(Q_k) \triangleq \text{span}\{q_1, q_2, \ldots, q_k\}$$

- k-D principal eigenpair: (Q_k, Λ_k)

Optimal Solutions of SLRP

Equivalence

Assume that $A = A^{\mathsf{T}} \in \mathbb{R}^{n\times n}$ such that $\lambda_k \geq 0$. Then $X \in \mathbb{R}^{n\times k}$ is a solution to $\min \|XX^{\mathsf{T}} - A\|_{\mathrm{F}}^2$ if and only if it has SVD:

$$X = Q_k \Lambda_k^{\frac{1}{2}} V^{\mathsf{T}},$$

where (Q_k, Λ_k) is a k-D principal eigenpair of A, $V \in \mathbb{R}^{k\times k}$ is orthogonal but otherwise arbitrary.

1st-order condition for SLRP:

$$AX = X(X^{\mathsf{T}}X)$$

Stationary points span invariant subspaces.

Most Established Eigensolvers

Why not just call `eigs` (or `svds`) in MATLAB? (ARPACK)

Why not use one of the existing eigensolvers?

Emerging applications demand new capacities.

- high efficiency at moderate accuracy
- high eigenspace dimensions
- high parallel scalability
- warm-start capacity

Established eigensolvers often lack in one or more aspects.

Advanced scientific computing and evolving computer archtectures call for new algorithms (either of general or special purpose).

Block Methods

Block vs. Sequential (Lanczos-type Methods)

- Block SpMV: $AV = [Av_1 \ Av_2 \ \cdots Av_k]$
- Sequential SpMv's: $Av \rightarrow A^2v \cdots \rightarrow A^k v$
  (+ inner products for orthogonalization)

As k increases, block methods are gaining advantages.

Block methods can be warm-started in an iterative setting.

Classic Block Method SSI: (power method)

$$X^{i+1} = orth(AX^i)$$

Other block algorithms:

- Block Jacobian-Davidson: Feast
- Trace minimization: LOBPCG, LMSVD

Research on block methods seems still largely unsettled.

Trace Minimization

Trace Minimization

$$\min_{X \in \mathbb{R}^{n \times k}} \mathbf{tr}(X^T A X) \quad \text{s.t.} \quad X^T X = I.$$

LMSVD, L.-Wen-Zhang, 2013, SISC

Two main types of operations: AX & $\mathrm{RR/orth}$

As k increases, $AX \ll \mathrm{RR/orth} \longrightarrow$ bottleneck

Parallel Scalability

- $AX \longrightarrow Ax_1 \cup Ax_2 \cup \ldots \cup Ax_k$. Higher.
- $\mathrm{RR/orth}$ inherits sequentiality. Lower.

Avoid bottleneck?

- Do less RR/orth

No free lunch?

- Do more BLAS3 (higher scalability than AX)

Orthogonal Free Models

Unconstrained Model: $\min_{X\in\mathbb{R}^{m\times k}} \frac{1}{4}\|X^TX\|_F^2 + \frac{1}{2}\mathbf{tr}(X^TAX)$, *Dai-Jiang-Cui, 2013*

Trace-penalty Minimization

$$\min_{X\in\mathbb{R}^{m\times k}} f(X) := \frac{1}{2}\mathbf{tr}(X^TAX) + \frac{\mu}{4}\|X^TX - I\|_F^2.$$

EIGPEN, Wen-Yang-L.-Zhang, 2012, available at "optimization online"

Good properties:

- Penalty parameter μ does not need to be infinity
- Equivalent to Trace Minimization for eigenspace computation
- No non-global local minimizer, less undesired saddle point

Algorithm

- Gradient method
- Barzilai Borwein stepsize
- Rayleigh-Ritz restart

Why SLRP?

Questions to EIGPEN

- Gradient method + BB (How about high-order methods?)
- Condition number: $k = 1$: $\kappa(\nabla^2 f_\mu(\hat{X})) = \frac{\lambda_n - \lambda_1}{\lambda_2 - \lambda_1}$; $k > 1$:

$$\begin{aligned}\kappa\left(\nabla^2 f_\mu(\hat{X})\big|_{Q_k^\perp}\right) &\triangleq \frac{\max\limits_{S\in\mathbb{R}^{n\times k}}\left\{\mathrm{tr}(S^{\mathsf{T}}\nabla^2 f_\mu(\hat{X})(S)) : \mathrm{tr}(S^{\mathsf{T}}S) = 1, S^{\mathsf{T}}Q_k = 0\right\}}{\min\limits_{S\in\mathbb{R}^{n\times k}}\left\{\mathrm{tr}(S^{\mathsf{T}}\nabla^2 f_\mu(\hat{X})(S)) : \mathrm{tr}(S^{\mathsf{T}}S) = 1, S^{\mathsf{T}}Q_k = 0\right\}} \\ &= \frac{\lambda_n - \lambda_1}{\lambda_{k+1} - \lambda_k}.\end{aligned}$$

(How about linear convergence rate?)

- μ should be tuned in properly. (How to avoid μ?)

Gauss-Newton Review

Nonlinear Least Squares Model:

$$\min_{x\in\mathbb{R}^n} f(x) \triangleq \frac{1}{2} r(x)^T r(x). \qquad r(x) : \mathbb{R}^n \to \mathbb{R}^m.$$

Linearize: $r(x+s) \approx r(x) + J(x)s$, where $J(x)$ is the Jacobian.

Normal equations + Line Search:
(minimize the lienar approximation)

$$J(x)^{\mathsf{T}} J(x)s = -J(x)^{\mathsf{T}} r(x). \qquad x = x + \alpha s.$$

Some properties:

- Fast for small residual. Slow for large residual.
- Local convergence may require $\alpha < 1$ all the time.

SLRP: Nonlinear Least Squares Model

$$\min_{X \in \mathbb{R}^{n \times k}} f(X) \triangleq \frac{1}{2}\|R(X)\|_F^2. \qquad R(X) \triangleq XX^T - A.$$

Let $J(X) : \mathbb{R}^{n \times k} \to \mathbb{R}^{n \times n}$ be the Jacobian operator of $R(X)$ at X.

Normal equations: (size $nk \times nk$)

$$J(X)^T J(X)(S) = -J(X)^T(R(X)).$$

Infinitely many solutions since $J(X)$ is rank deficient.

GN Directions

Special structure of normal equations allows low-cost solution:

$$SX^{\mathsf{T}}X + XS^{\mathsf{T}}X = AX - X(X^{\mathsf{T}}X)$$

GN Direction

Let $X \in \mathbb{R}^{n\times k}$ be full rank, and $\mathcal{P}_X = X(X^{\mathsf{T}}X)^{-1}X^{\mathsf{T}}$. Then

$$S_C = \left(I - \frac{1}{2}\mathcal{P}_X\right)\left(AX(X^{\mathsf{T}}X)^{-1} - X\right) + XC,$$

where $C^T = -C$, satisfies the normal equations. In particular, for $C = 0$,

$$S_0 = \left(I - \frac{1}{2}\mathcal{P}_X\right)\left(AX(X^{\mathsf{T}}X)^{-1} - X\right)$$

is a minimum weighted-norm Gauss-Newton direction at X.

Gauss-Newton Algorithm (Theoretical Version)

Gauss-Newton (GN):

- While not "converged", do
  1. If $\sigma_{\min}(X) < \delta$, set $X = X + P$; —- **Correction Step**
  2. Select $\alpha = \min\left(1, \sigma^3_{\min}(X)/\|\nabla f(X)\|_F\right)$, set $X = X + \alpha S_0$.

Calculation *GN* step:

- $Y = X(X^TX)^{-1}$, $G = AY - X$
- $S_0 = G - X(Y^TG)/2$

Computational cost:

- 1 block SpMV: AY
- 3 dense matrix multiplications
- 1 $k \times k$ linear system with n rhs

Practical Implementation

So far, in practice

- $\alpha = 1$ appears always to work well;
- Correction step can hardly be invoked.

$[X, Y] = GN(A, X)$

- While not "converged", do
  1. $Y = X(X^{\mathsf{T}}X)^{-1}$
  2. $X = AY - X(Y^{T}AY - I)/2$

Simple Algorithm

- Two-liner with no parameters
- No orthogonalization
- No Rayleigh-Ritz (unless eigenpairs are required)

Step Size and Correction Step

Full Rankness: $\sigma_{\min}(X^{i+1}) \geq 0.75\,\sigma_{\min}(X^i)$

Correction Step:

- $\delta \leq \left(\frac{\lambda_n/\lambda_1}{4+\sqrt{20}}\right)\sqrt{\frac{\lambda_n}{k}}$
- $X_c = X + P$ $(:= \sqrt{\frac{\lambda_n}{p}}\,UV_p^{\mathsf{T}}$, where $U^{\mathsf{T}}X = 0$ and $U^{\mathsf{T}}U = I)$

Key properties:

- $\sigma_{\min}(X_c) \geq \delta$
- $f(X_c) < f(X) - \frac{1}{4}\lambda_n^2$

Convergence Results

Theorem (Global Convergence)

Suppose that $A \succ 0$. Let $\{X^i\}$ be generated by SLRPGN(TH) starting from a full-rank initial point. Then after finite number of iterations, step-size $\alpha = 1$ will always be taken, no more correction step, and $\nabla f(X_i) \to 0$.

$f(X)$ does not have any local (non-global) minimum. It is unlikely that the iterates get trapped at a saddle point. Better local convergence result holds if we further assume $\lambda_k > \lambda_{k+1}$.

Theorem (Q-Linear Rate)

Suppose $A \succ 0$ and $\lambda_k > \lambda_{k+1}$. Then $\{X^i\}$, a sequence generated by SLRPGN(PR) starting from a full-rank initial point $X^0 \in \mathcal{L}(\underline{\gamma})$, globally converges to $\mathcal{L}(f^*)$, where $\mathcal{L}(\gamma) := \{X \mid f(X) \leq \gamma\}$ denotes the level set, f^* denotes the global minimum of SLRP and $\underline{\gamma} > f^*$ is a constant. Moreover, the gradient sequence $\{\nabla f(X^i)\}$ converges to zero at a Q-linear rate $\frac{\lambda_{k+1}}{\lambda_k}$.

Experiment Environment

Platform

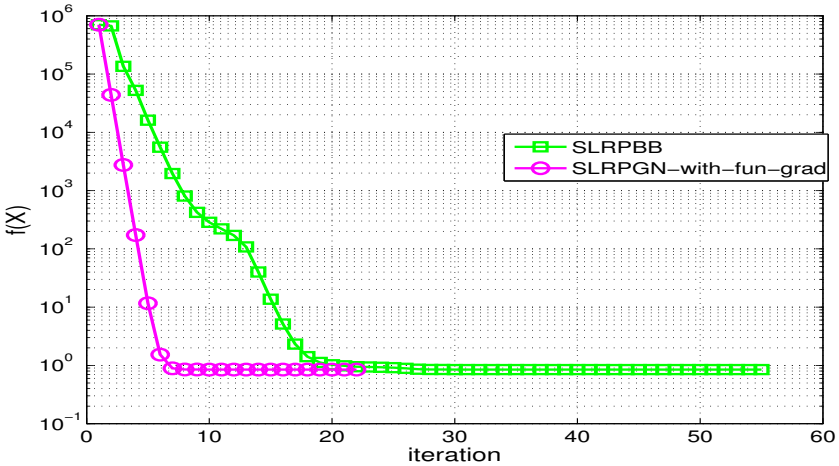
All the experiments were preformed on a linux workstation with 2 Intel Xeon E5-2697 CPUs (2.70GHz, 12 cores) and 128GB of memory running Ubuntu 12.04 and MATLAB 2013b.

Tested Methods

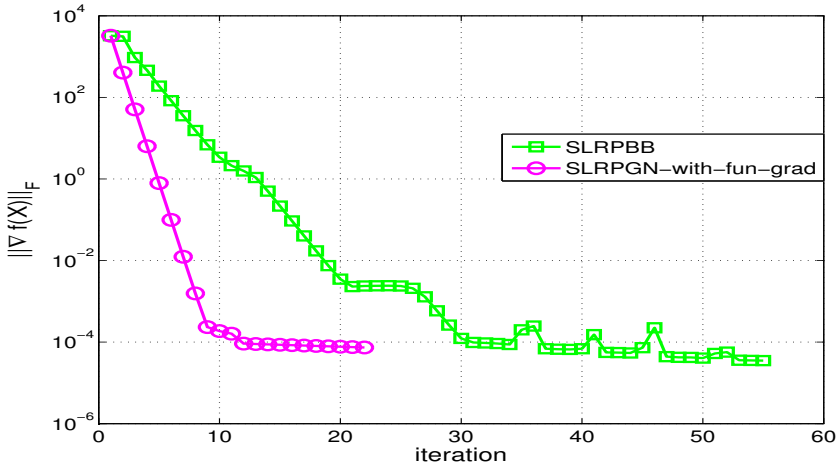
- MATLAB EIGS — Lanczos-based (ARPACK, Sorensen et.al.)
- LANSVD — Lanczos-based (PROPACK, R. M. Larsen)
- LMSVD — block subspace method **(L.-Wen-Zhang, SISC, 2013)**
- SLRPBB – BB + gradient (EIGPEN) **(Wen-Yang-L.-Zhang)**
- **SLRPGN** – proposed GN algorithm

Required Accuracy: moderate

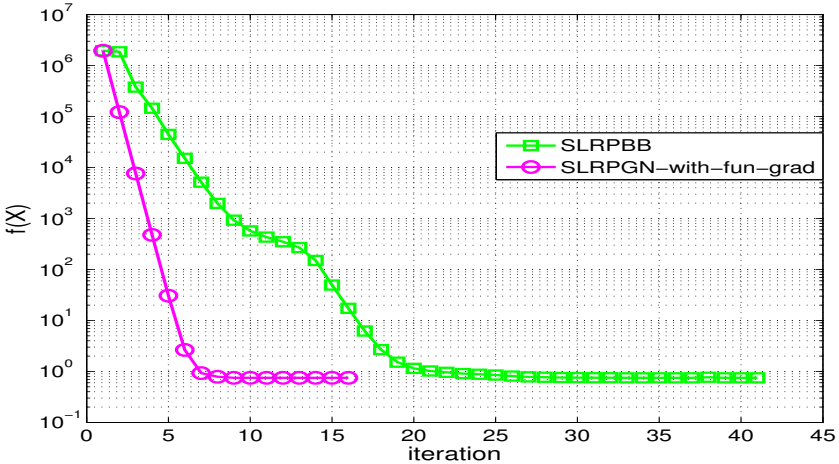
Comparison with the Gradient Method



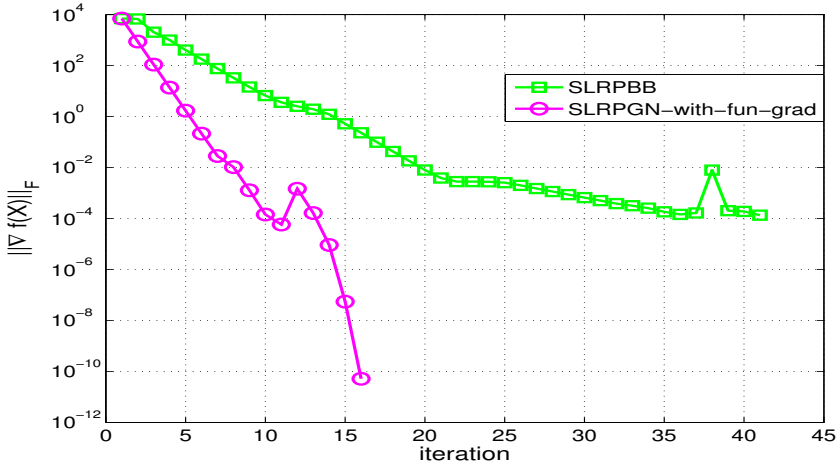
(a) $k = 300$, objective function values



(b) $k = 300$, gradient norms $\|\nabla f(X)\|_F$



(c) $k = 500$, objective function values



(d) $k = 500$, gradient norms $\|\nabla f(X)\|_F$

Robust Principal Component Analysis

Data matrix is

$$M = L_0 + S_0 + \omega \in \mathbb{R}^{m \times n},$$

where L_0 is low-rank, S_0 is sparse and ω is small noise.

Given M, find L_0 and S_0 approximately by solving:

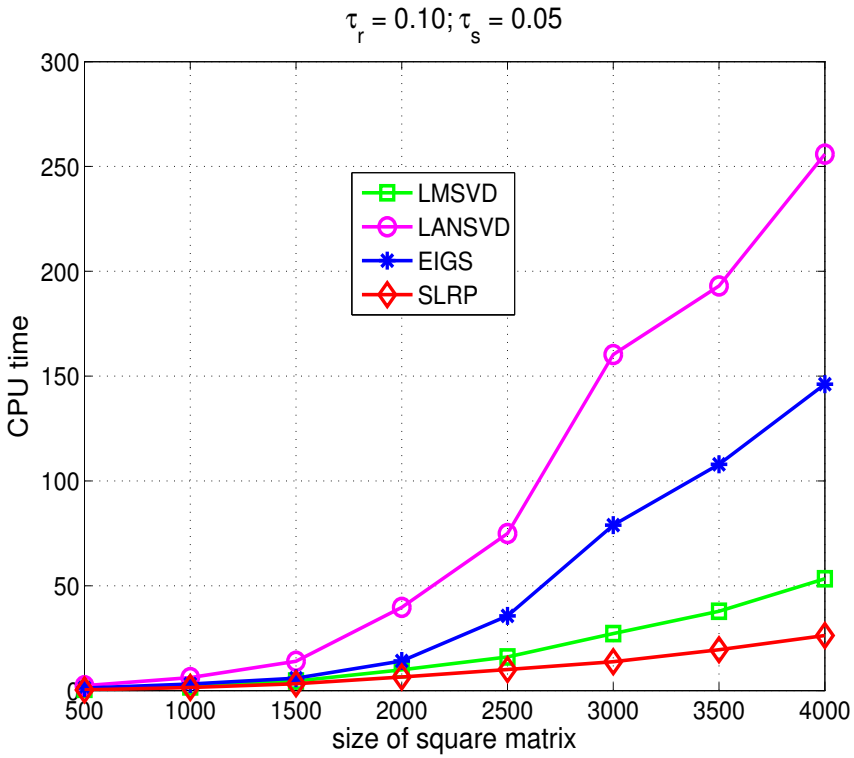
$$\min_{L,S} \|L\|_* + \mu \|S\|_1, \quad \text{s.t.} \quad L + S = M.$$

MATLAB Code: IALM (Lin *et al*)

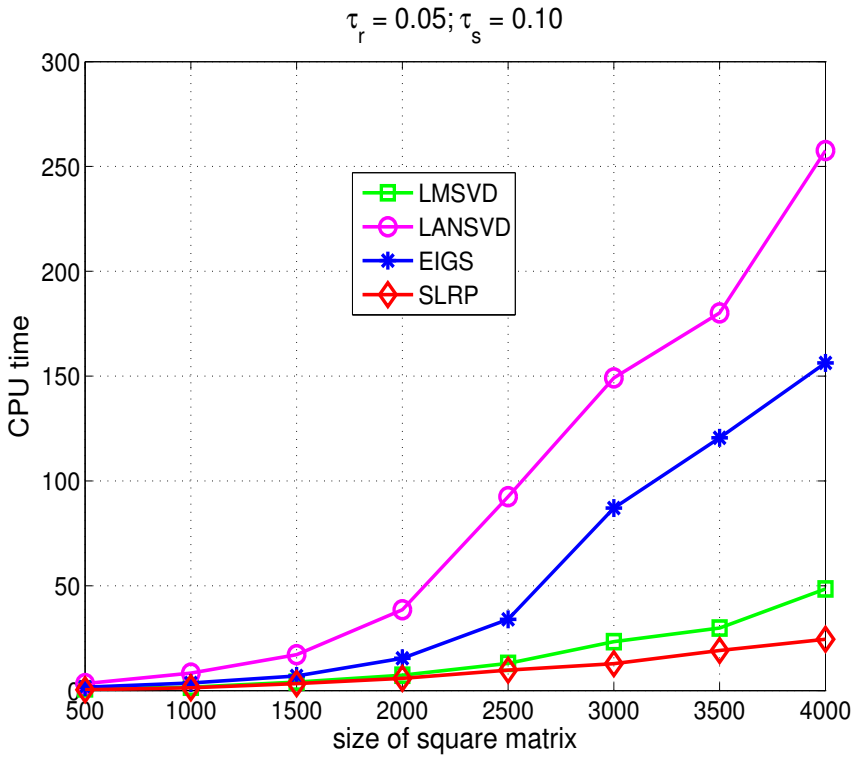
- Alternating Direction Multiplier Method (ADMM)
- Calls SVD at every iteration (warm-start desired)
- Test cases: random instances

RPCA Results

CPU Time in Seconds



(a) rank = $0.10n$ Sample 5%



(b) rank = $0.05n$ Sample 10%

(All achieved similar accuracy)

Matrix Completion

Find a low-rank matrix from a sampled set of its entries

Given the entries of $M + \omega$ in Ω, find $X \approx M$ by solving:

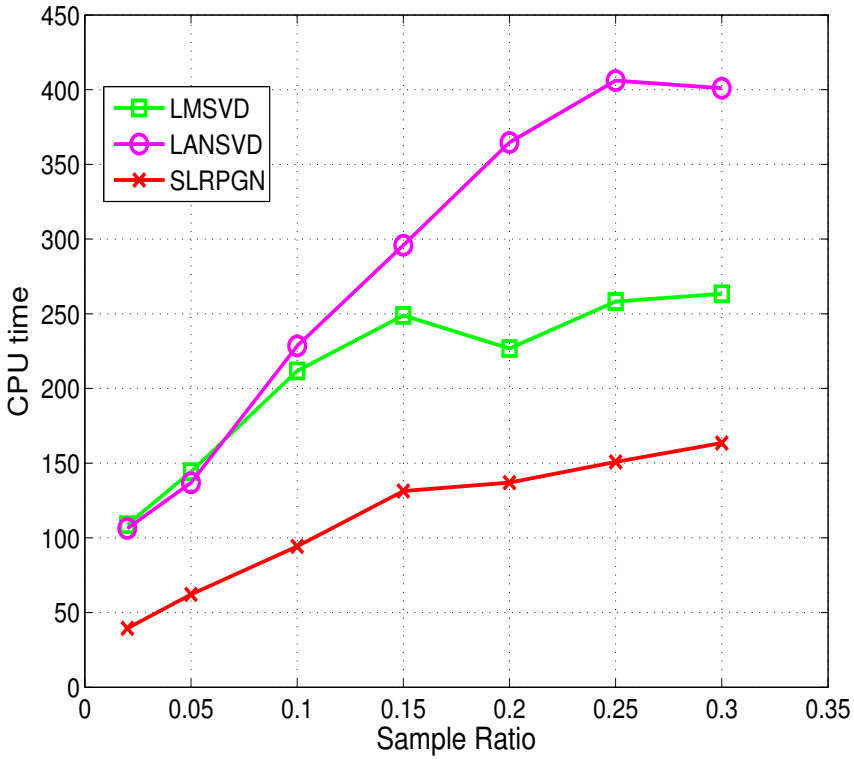
$$\min_X \|X\|_*, \quad \text{s.t.} \quad X_{ij} = M_{ij}, \ \forall (i,j) \in \Omega.$$

MATLAB Code: SVT and NNLS

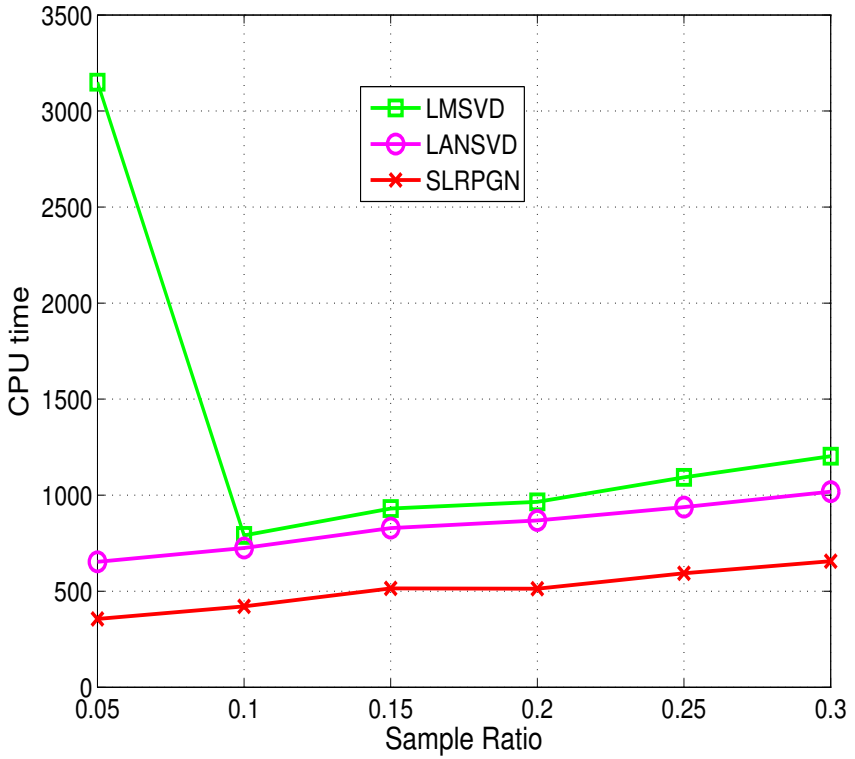
- Singular Value Thresholding
- Calls SVD at every iteration (warm-start desired)
- Test cases: random instances

SVT Results

CPU Time in Seconds



(a) rank = 10



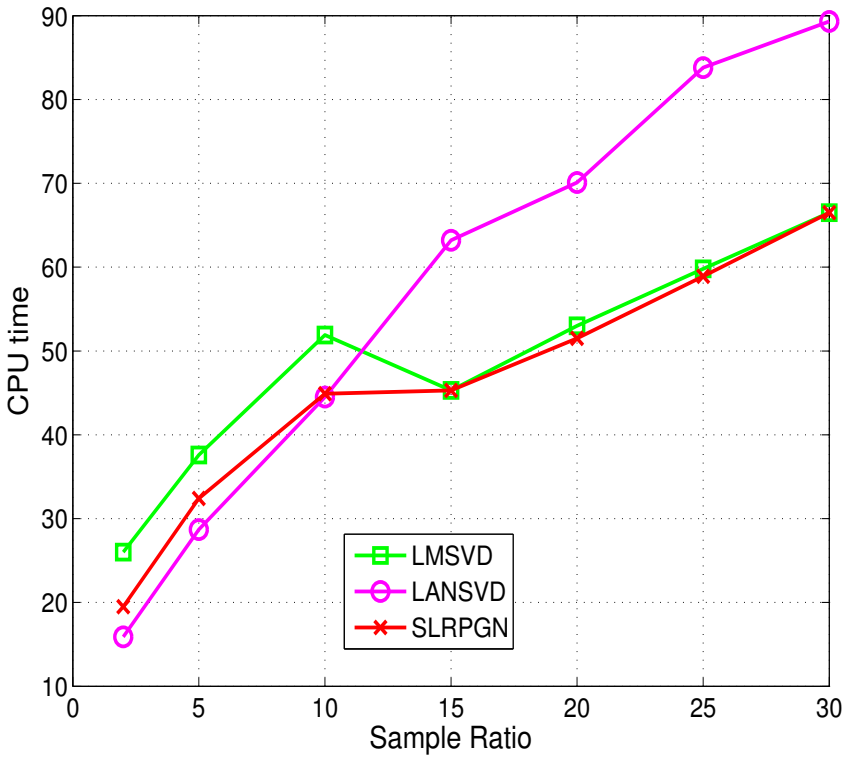
(b) rank = 50

(All achieved similar accuracy)

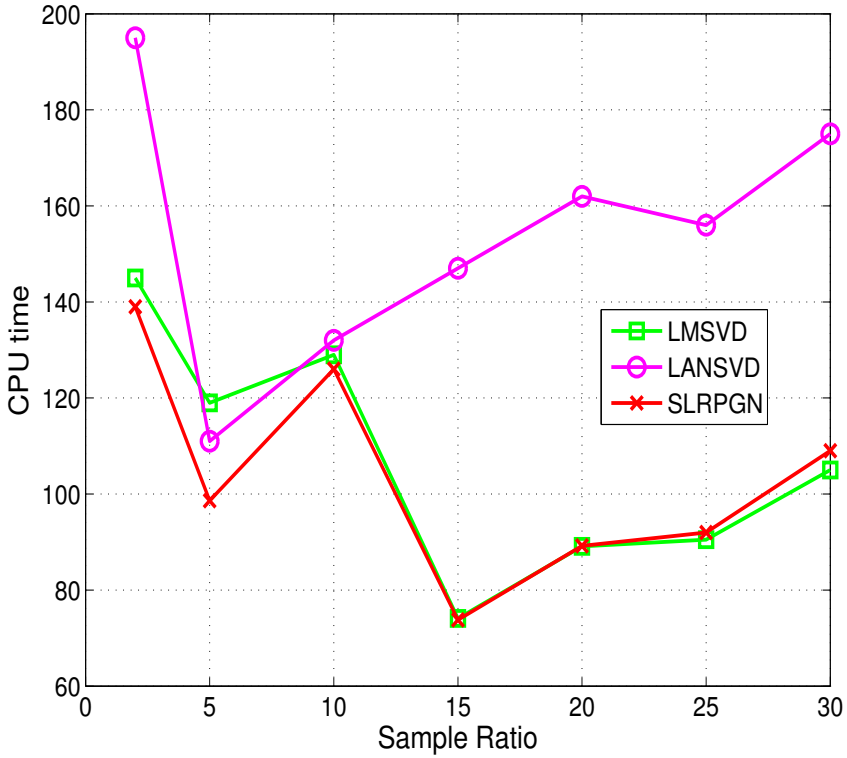
NNLS Results

The sparse-dense matrix multiplication uses Matlab's own version

CPU Time in Seconds



(a) rank = 10



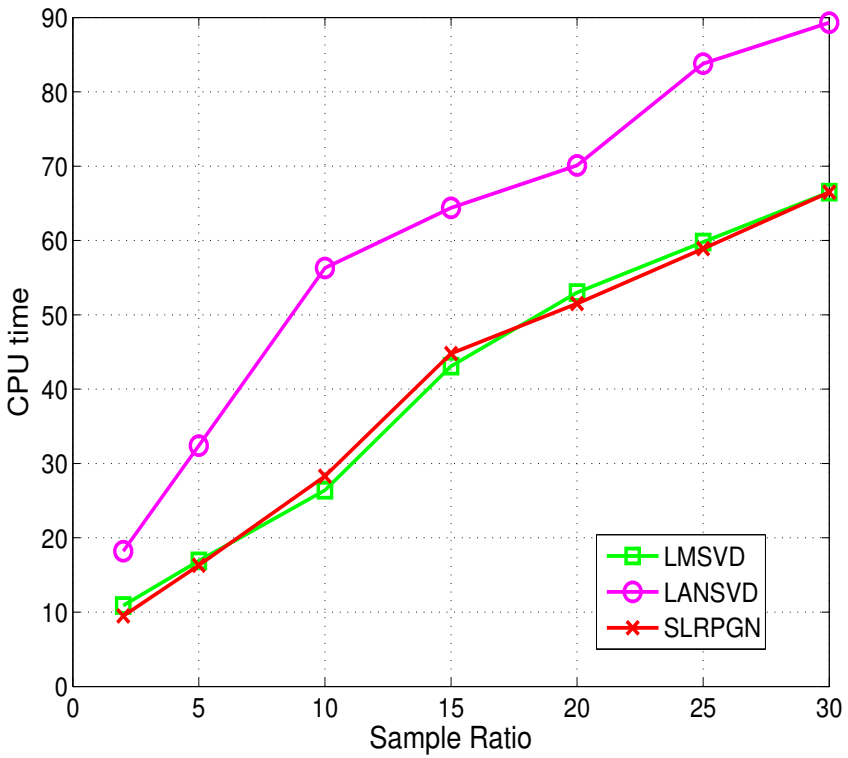
(b) rank = 50

(All achieved similar accuracy)

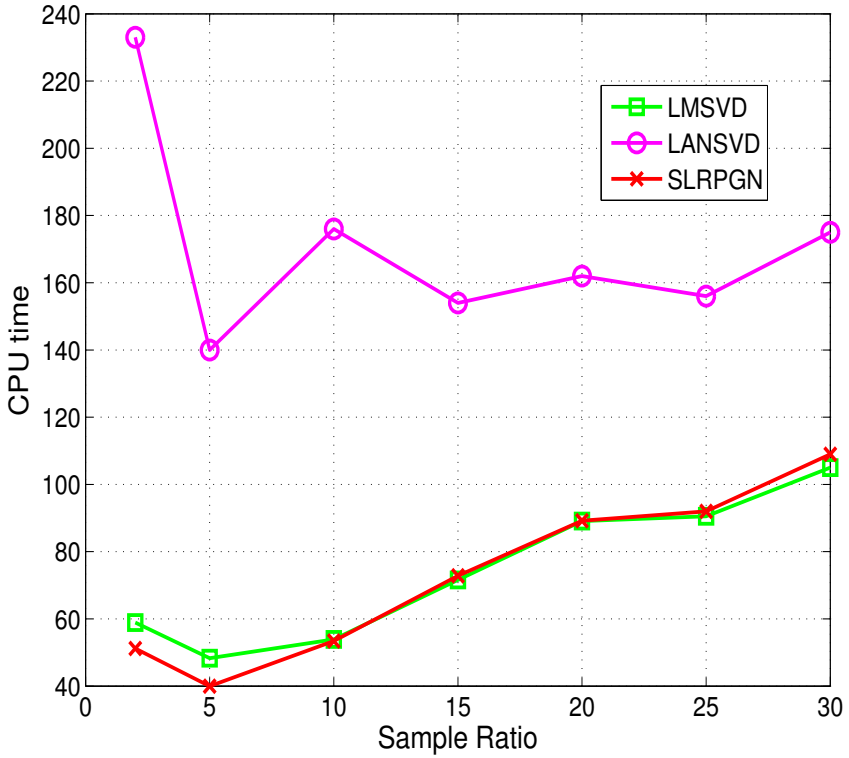
NNLS Results

The sparse-dense matrix multiplication uses MKL

CPU Time in Seconds



(a) rank = 10



(b) rank = 50

(All achieved similar accuracy)

Summery and Remarks

SLRP: $\min \|XX^T - A\|_F^2$. Output (X, Y)

GN: $Y = X(X^TX)^{-1}$; $X = AY - X(Y^TAY - I)/2$

- GN: simple and parameter-free
- Principal subspace without SVD, nor Rayleigh-Ritz
- Benefit of concurrency already seen in plain MATLAB
- Global convergence and local Q-linear convergence rate
- Effective for small residuals and low-moderate accuracy (so far)

Further Works

- Strategically placed Rayleigh-Ritz will improve accuracy
- Other eigen-techniques (poly-filtering, deflation, ...) help too
- GN + (a few RR): Potential as eigensolver worth investigating
- Parallel scalability to be exploited

Thank you for your attention!