A Minimization Algorithm

- Consider the minimization problem:

$$M^* = \min_M \|M\|_*$$

subject to

$$\sum_{(i,j)\in\Omega} (M(i,j) - \Gamma(i,j))^2 \le \delta$$

- There are many techniques to solve this problem (http://perception.csl.illinois.edu/matrix-rank/sample_code.html)
- Out of these, we will study one method called "singular value thresholding".

Ref: Cai et al, *A singular value thresholding algorithm for matrix completion*, SIAM Journal on Optimization, 2010.

Singular Value Thresholding (SVT)

$$\Phi^* = SVT(\Gamma, \tau > 0)$$
$$\{$$
$$Y^{(0)} = 0 \in R^{n_1 \times n_2}$$
$$\text{k} = 1$$
$$\text{while(convergence criterion not met)}$$
$$\{$$
$$\Phi^{(k)} = soft - threshold(Y^{(k-1)}; \tau)$$
$$Y^{(k)} = Y^{(k-1)} + \delta_k P_\Omega(\Gamma - \Phi^{(k)}); k = k+1;$$
$$\}$$
$$\Phi^* = \Phi^{(k)};$$
$$\}$$

$$\hat{Y} = soft - threshold(Y \in R^{n_1 \times n_2}; \tau)$$
$$\{$$
$$Y = USV^T \text{(using svd)}$$
$$\text{for } (k = 1 : rank(Y))$$
$$\{$$
$$S(k,k) = \max(0, S(k,k) - \tau);$$
$$\}$$
$$\hat{Y} = \sum_{i=1}^{rank(Y)} S(k,k) u_k v_k^t$$
$$\}$$

The soft-thresholding procedure obeys the following property (which we state w/o proof).

$$soft - threshold(Y; \tau) = \arg\min_X \frac{1}{2}\|X - Y\|_F^2 + \tau \|X\|_*$$

Properties of SVT (stated w/o proof)

- The sequence $\{\Phi_k\}$ converges to the true solution of the problem below provided the step-sizes $\{\delta_k\}$ all lie between 0 and 2.

$$M^* = \min_M \tau \|M\|_* + 0.5\|M\|_F^2$$

subject to

$$\forall (i,j) \in \Omega, M(i,j) = \Gamma(i,j)$$

- For large values of τ, this converges to the solution of the original problem (i.e. without the Frobenius norm term).

Properties of SVT (stated w/o proof)

- The matrices $\{\mathbf{\Phi_k}\}$ turn out to have low rank (empirical observation – proof not established).
- The matrices $\{\mathbf{Y_k}\}$ also turn out to be sparse (empirical observation – rigorous proof not established).
- The SVT step does not require computation of full SVD – we need only those singular vectors whose singular values exceed τ. There are special iterative methods for that.

Results

- The SVT algorithm works very efficiently and is easily implementable in MATLAB.

- The authors report reconstruction of a 30,000 by 30,000 matrix in just 17 minutes on a 1.86 GHz dual-core desktop with 3 GB RAM and with MATLAB's multithreading option enabled.

Results (Data without noise)

Unknown M				Computational results		
size ($n \times n$)	rank (r)	m/d_r	m/n^2	time(s)	# iters	relative error
	10	6	0.12	23	117	1.64×10^{-4}
$1,000 \times 1,000$	50	4	0.39	196	114	1.59×10^{-4}
	100	3	0.57	501	129	1.68×10^{-4}
	10	6	0.024	147	123	1.73×10^{-4}
$5,000 \times 5,000$	50	5	0.10	950	108	1.61×10^{-4}
	100	4	0.158	3,339	123	1.72×10^{-4}
	10	6	0.012	281	123	1.73×10^{-4}
$10,000 \times 10,000$	50	5	0.050	2,096	110	1.65×10^{-4}
	100	4	0.080	7,059	127	1.79×10^{-4}
$20,000 \times 20,000$	10	6	0.006	588	124	1.73×10^{-4}
	50	5	0.025	4,581	111	1.66×10^{-4}
$30,000 \times 30,000$	10	6	0.004	1,030	125	1.73×10^{-4}

TABLE 5.1

Experimental results for matrix completion. The rank r is the rank of the unknown matrix $\boldsymbol{M}$*,* m/d_r *is the ratio between the number of sampled entries and the number of degrees of freedom in an* $n \times n$ *matrix of rank r (oversampling ratio), and* m/n^2 *is the fraction of observed entries. All the computational results on the right are averaged over five runs.*

https://arxiv.org/abs/0810.3286

Results (Noisy Data)

noise ratio	Unknown matrix M				Computational results		
	size ($n \times n$)	rank (r)	m/d_r	m/n^2	time(s)	# iters	relative error
10^{-2}	$1,000 \times 1,000$	10	6	0.12	10.8	51	0.78×10^{-2}
		50	4	0.39	87.7	48	0.95×10^{-2}
		100	3	0.57	216	50	1.13×10^{-2}
10^{-1}	$1,000 \times 1,000$	10	6	0.12	4.0	19	0.72×10^{-1}
		50	4	0.39	33.2	17	0.89×10^{-1}
		100	3	0.57	85.2	17	1.01×10^{-1}
1	$1,000 \times 1,000$	10	6	0.12	0.9	3	0.52
		50	4	0.39	7.8	3	0.63
		100	3	0.57	34.8	3	0.69

TABLE 5.3

Simulation results for noisy data. The computational results are averaged over five runs. For each test, the table shows the results of Algorithm 1 applied with an early stopping criterion

https://arxiv.org/abs/0810.3286

Results on real data

- Dataset consists of a matrix **M** of geodesic distances between 312 cities in the USA/Canada.
- This matrix is of approximately low-rank (in fact, the relative Frobenius error between **M** and its rank-3 approximation is 0.1159).
- 70% of the entries of this matrix (chosen uniformly at random) were blanked out.

Results on real data

Algorithm	rank	k_i	time	$\|\boldsymbol{M}-\boldsymbol{M}_i\|_F/\|\boldsymbol{M}\|_F$	$\|\boldsymbol{M}-\boldsymbol{X}^{k_i}\|_F/\|\boldsymbol{M}\|_F$
SVT	1	58	1.4	0.4091	0.4170
	2	190	4.8	0.1895	0.1980
	3	343	8.9	0.1159	0.1252
(3.6)	1	47	2.6	0.4091	0.4234
	2	166	7.2	0.1895	0.1998
	3	310	13.3	0.1159	0.1270

TABLE 5.5

Speed and accuracy of the completion of the city-to-city distance matrix. Here, $\|\boldsymbol{M}-\boldsymbol{M}_i\|_F/\|\boldsymbol{M}\|_F$ *is the best possible relative error achieved by a matrix of rank i.*

https://arxiv.org/abs/0810.3286

Algorithm for Robust PCA

- The algorithm uses the augmented Lagrangian technique.
- See https://en.wikipedia.org/wiki/Augmented_Lagrangian_method and https://www.him.uni-bonn.de/fileadmin/him/Section6_HIM_v1.pdf
- Suppose you want to solve:

$$\min \; f(x) \text{ w.r.t. } x$$
$$\text{s.t. } \forall i \in I, c_i(x) = 0$$

Algorithm for Robust PCA

- Suppose you want to solve:

$$\min \; f(x) \text{ w.r.t. } x$$
$$\text{s.t. } \forall i \in I, c_i(x) = 0$$

- The augmented Lagrangian method (ALM) adopts the following iterative updates:

$$x_k = \arg\min_x \; f(x) + \mu_k \sum_{i \in I} c_i^2(x) - \sum_{i \in I} \lambda_i c_i(x)$$
$$\lambda_i = \lambda_i - \mu_k c_i(x_k)$$

Augmentation term Lagrangian term

ALM: Some intuition

- What is the intuition behind the update of the Lagrange parameters $\{\lambda_i\}$?

- The problem is:

$$\min f(x) \quad \text{s.t.} \ \forall i \in I, c_i(x) = 0$$

$$=$$

$$\min_x \max_\lambda f(x) + \boldsymbol{\lambda}^t \boldsymbol{c}(x)$$

$$\boldsymbol{c}(x) = (c_1(x), c_2(x), \ldots, c_{|I|}(x))$$

The maximum w.r.t. λ will be ∞ unless the constraint is satisfied. Hence these problems are equivalent.

ALM: Some intuition

- The problem is:

$$\min \; f(x) \quad \text{s.t.}\ \forall i \in I, c_i(x) = 0$$

$$=$$

$$\min_x \max_\lambda f(x) + \lambda^t \boldsymbol{c}(x)$$

$$\boldsymbol{c}(x) = (c_1(x), c_2(x), \ldots, c_{|I|}(x))$$

Due to non-smoothness of the max function, the equivalence has little computational benefit. We smooth it by adding another term that penalizes deviations from a prior estimate of the **λ** parameters.

$$\min_x \max_\lambda f(x) + \lambda^t \boldsymbol{c}(x) + \frac{\left\| \lambda - \bar{\lambda} \right\|^2}{2\mu}$$

$$\Rightarrow \quad \lambda = \bar{\lambda} - \mu \boldsymbol{c}(x)$$

Maximization w.r.t. **λ** is now easy

ALM: Some inutuion – inequality constraints

$$\min \; f(x) \quad \text{s.t.}\ \forall \mathrm{i} \in \mathrm{I}, c_i(x) \geq 0$$

=

$$\min_x \max_{\lambda \geq 0} f(x) - \boldsymbol{\lambda}^t \boldsymbol{c}(x)$$

$$\boldsymbol{c}(x) = (c_1(x), c_2(x), \ldots, c_{|I|}(x))$$

$$\min_x \max_\lambda f(x) + \boldsymbol{\lambda}^t \boldsymbol{c}(x) + \frac{\left\| \boldsymbol{\lambda} - \bar{\boldsymbol{\lambda}} \right\|^2}{2\mu}$$

→

$$\boldsymbol{\lambda} = \max(\bar{\boldsymbol{\lambda}} - \mu \boldsymbol{c}(x), 0)$$

Maximization w.r.t. **λ** is now easy

Theorem 1 (Informal Statement)

- Consider a matrix **M** of size n_1 by n_2 which is the sum of a "sufficiently low-rank" component **L** and a "sufficiently sparse" component **S** whose support is uniformly randomly distributed in the entries of **M**.
- Then the solution of the following optimization problem (known as **principal component pursuit**) yields **exact estimates** of **L** and **S** with "very high" probability:

$$E(L',S') = \min_{(L,S)} \|L\|_* + \frac{1}{\sqrt{\max(n_1,n_2)}} \|S\|_1$$

subject to $L + S = M$.

$$Note: \|S\|_1 = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} |S_{ij}|$$

This is a convex optimization problem.

Algorithm for Robust PCA

- In our case, we seek to optimize:

$$l(L,S,Y) = \|L\|_* + \lambda\|S\|_1 + \langle Y, M - L - S\rangle + \frac{\mu}{2}\|M - L - S\|_F^2.$$

Lagrange matrix

- Basic algorithm:

$$(L_k, S_k) = \arg\min_{(L,S)} l(L,S,Y_k), Y_{k+1} = Y_k + \mu(M - L_k - S_k)$$

$$\arg\min_S l(L,S,Y) = \mathcal{S}_{\lambda\mu^{-1}}(M - L + \mu^{-1}Y).$$

$$\mathcal{S}_\tau[x] = \operatorname{sgn}(x)\max(|x| - \tau, 0)$$

Update of **S** using soft-thresholding

$$\arg\min_L l(L,S,Y) = \mathcal{D}_{\mu^{-1}}(M - S + \mu^{-1}Y).$$

$$\mathcal{D}_\tau(X) = U\mathcal{S}_\tau(\Sigma)V^* \qquad X = U\Sigma V^*$$

Update of **L** using singular-value soft-thresholding

Alternating Minimization Algorithm for Robust PCA

initialize: $S_0 = Y_0 = 0, \mu > 0.$

while not converged do

  compute $L_{k+1} = \mathcal{D}_{\mu^{-1}}(M - S_k + \mu^{-1} Y_k);$

  compute $S_{k+1} = \mathcal{S}_{\lambda\mu^{-1}}(M - L_{k+1} + \mu^{-1} Y_k);$

  compute $Y_{k+1} = Y_k + \mu(M - L_{k+1} - S_{k+1});$

end while

output: $L, S.$

https://statweb.stanford.edu/~candes/papers/RobustPCA.pdf

Results

Dimension n	$\text{rank}(L_0)$	$\|S_0\|_0$	$\text{rank}(\hat{L})$	$\|\hat{S}\|_0$	$\frac{\|\hat{L}-L_0\|_F}{\|L_0\|_F}$	# SVD	Time(s)
500	25	12,500	25	12,500	1.1×10^{-6}	16	2.9
1,000	50	50,000	50	50,000	1.2×10^{-6}	16	12.4
2,000	100	200,000	100	200,000	1.2×10^{-6}	16	61.8
3,000	250	450,000	250	450,000	2.3×10^{-6}	15	185.2

$\text{rank}(L_0) = 0.05 \times n$, $\|S_0\|_0 = 0.05 \times n^2$.

Dimension n	$\text{rank}(L_0)$	$\|S_0\|_0$	$\text{rank}(\hat{L})$	$\|\hat{S}\|_0$	$\frac{\|\hat{L}-L_0\|_F}{\|L_0\|_F}$	# SVD	Time(s)
500	25	25,000	25	25,000	1.2×10^{-6}	17	4.0
1,000	50	100,000	50	100,000	2.4×10^{-6}	16	13.7
2,000	100	400,000	100	400,000	2.4×10^{-6}	16	64.5
3,000	150	900,000	150	900,000	2.5×10^{-6}	16	191.0

$\text{rank}(L_0) = 0.05 \times n$, $\|S_0\|_0 = 0.10 \times n^2$.

Table 1: Correct recovery for random problems of varying size. Here, $L_0 = XY^* \in \mathbb{R}^{n\times n}$ with $X, Y \in \mathbb{R}^{n\times r}$; X, Y have entries i.i.d. $\mathcal{N}(0, 1/n)$. $S_0 \in \{-1, 0, 1\}^{n\times n}$ has support chosen uniformly at random and independent random signs; $\|S_0\|_0$ is the number of nonzero entries in S_0. Top: recovering matrices of rank $0.05 \times n$ from 5% gross errors. Bottom: recovering matrices of rank $0.05 \times n$ from 10% gross errors. In all cases, the rank of L_0 and ℓ_0-norm of S_0 are correctly estimated. Moreover, the number of partial singular value decompositions (# SVD) required to solve PCP is almost constant.

(Compressive) Low Rank Matrix Recovery

Compressive RPCA: Algorithm and an Application

Primarily based on the paper:

Waters et al, "*SpaRCS: Recovering Low-Rank and Sparse Matrices from Compressive Measurements*", NIPS 2011

Problem statement

- Let **M** be a matrix which is the sum of low rank matrix **L** and sparse matrix **S**.
- We observed compressive measurements of **M** in the following form:

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{L} + \boldsymbol{S}), \boldsymbol{L} \in R^{n_1 \times n_2}, \boldsymbol{S} \in R^{n_1 \times n_2}, \boldsymbol{y} \in R^m, m \leq n_1 n_2$$

$$\mathcal{A} = \text{linear operator acting/map on } \boldsymbol{M}$$

$$\text{Retrieve } \boldsymbol{L}, \boldsymbol{S} \text{ given } \mathcal{A}, \boldsymbol{y}$$

Scenarios

- **M** could be a matrix representing a video – each column of **M** is a vectorized frame from the video.
- **M** could also be a matrix representing a hyperspectral image – each column is the vectorized form of a slice at a given wavelength.
- Robust Matrix completion – a special form of a compressive **L+S** recovery problem.

Objective function: SpaRCS

$$(\text{P1}) \quad \min \; \|\mathbf{y} - \mathcal{A}(\mathbf{L} + \mathbf{S})\|_2 \quad \text{subject to} \quad \text{rank}(\mathbf{L}) \leq r, \; \|\text{vec}(\mathbf{S})\|_0 \leq K.$$

Free parameters

SpaRCS = sparse and low rank decomposition via compressive sampling

SparCS Algorithm

Algorithm 1: $(\widehat{\mathbf{L}}, \widehat{\mathbf{S}}) = \text{SpaRCS}\,(\mathbf{y}, \mathcal{A}, \mathcal{A}^*, K, r, \epsilon)$

Initialization: $k \leftarrow 1, \widehat{\mathbf{L}}_0 \leftarrow \mathbf{0}, \widehat{\mathbf{S}}_0 \leftarrow \mathbf{0}, \boldsymbol{\Psi}_{\mathbf{L}} \leftarrow \emptyset, \boldsymbol{\Psi}_{\mathbf{S}} \leftarrow \emptyset, \mathbf{w}_0 \leftarrow \mathbf{y}$

while $\|\mathbf{w}_{k-1}\|_2 \geq \epsilon$ **do**

Compute signal proxy:

$\mathbf{P} \leftarrow \mathcal{A}^*(\mathbf{w}_{k-1})$

Support identification:

$\widehat{\boldsymbol{\Psi}}_{\mathbf{L}} \leftarrow \text{svd}(\mathbf{P}; 2r);\ \widehat{\boldsymbol{\Psi}}_{\mathbf{S}} \leftarrow \text{supp}(\mathbf{P}; 2K)$

Support merger:

$\widetilde{\boldsymbol{\Psi}}_{\mathbf{L}} \leftarrow \widehat{\boldsymbol{\Psi}}_{\mathbf{L}} \bigcup \boldsymbol{\Psi}_{\mathbf{L}};\ \widetilde{\boldsymbol{\Psi}}_{\mathbf{S}} \leftarrow \widehat{\boldsymbol{\Psi}}_{\mathbf{S}} \bigcup \boldsymbol{\Psi}_{\mathbf{S}}$

Least squares estimation:

$\mathbf{B}^{\mathbf{L}} \leftarrow \widetilde{\boldsymbol{\Psi}}_{\mathbf{L}}^{\dagger}(\mathbf{y} - \mathcal{A}(\widehat{\mathbf{S}}_{k-1}));\ \mathbf{B}^{\mathbf{S}} \leftarrow \widetilde{\boldsymbol{\Psi}}_{\mathbf{S}}^{\dagger}(\mathbf{y} - \mathcal{A}(\widehat{\mathbf{L}}_{k-1}))$

Support pruning:

$(\widehat{\mathbf{L}}_k\,,\ \boldsymbol{\Psi}_{\mathbf{L}}) \leftarrow \text{svd}(\mathbf{B}^{\mathbf{L}}; r);\ (\widehat{\mathbf{S}}_k\,,\ \boldsymbol{\Psi}_{\mathbf{S}}) \leftarrow \text{supp}(\mathbf{B}^{\mathbf{S}}; K)$

Update residue:

$\mathbf{w}_k \leftarrow \mathbf{y} - \mathcal{A}(\widehat{\mathbf{L}}_k + \widehat{\mathbf{S}}_k)$

$k \leftarrow k + 1$

end

$\widehat{\mathbf{L}} = \widehat{\mathbf{L}}_{k-1};\ \widehat{\mathbf{S}} = \widehat{\mathbf{S}}_{k-1}$

https://papers.nips.cc/paper/4438-sparcs-recovering-low-rank-and-sparse-matrices-from-compressive-measurements.pdf

Very simple to implement; but requires tuning of *K*, *r* parameters; convergence guarantees not established.

Results: Phase transition

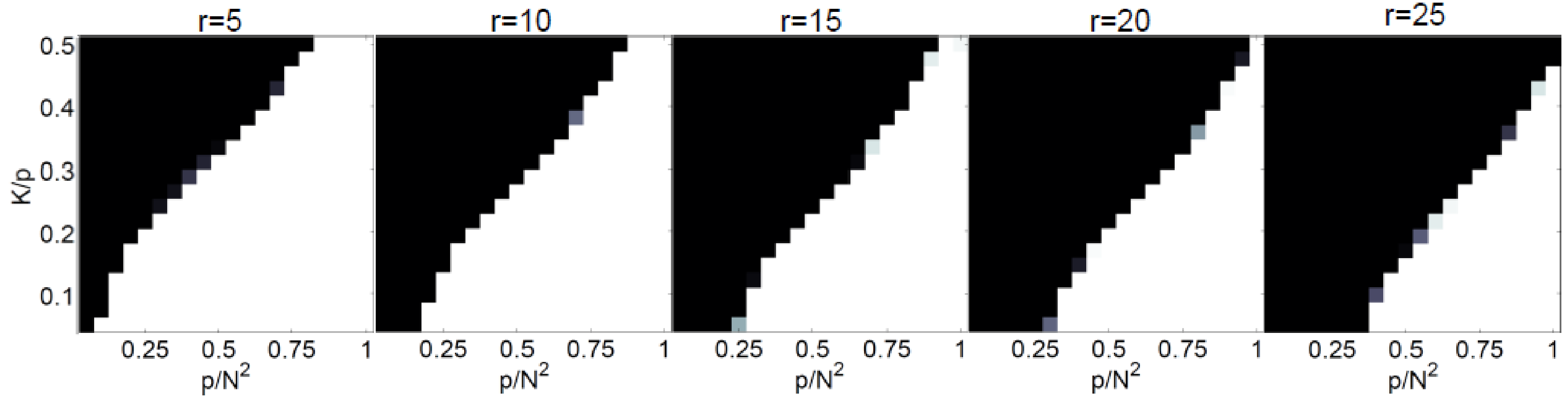


Figure 1: Phase transitions for a recovery problem of size $N_1 = N_2 = N = 512$. Shown are aggregate results over 20 Monte-Carlo runs at each specification of r, K, and p. Black indicates recovery failure, while white indicates recovery success.

https://papers.nips.cc/paper/4438-sparcs-recovering-low-rank-and-sparse-matrices-from-compressive-measurements.pdf

Code:
https://www.ece.rice.edu/~aew2/sparcs.html

Results: Video CS

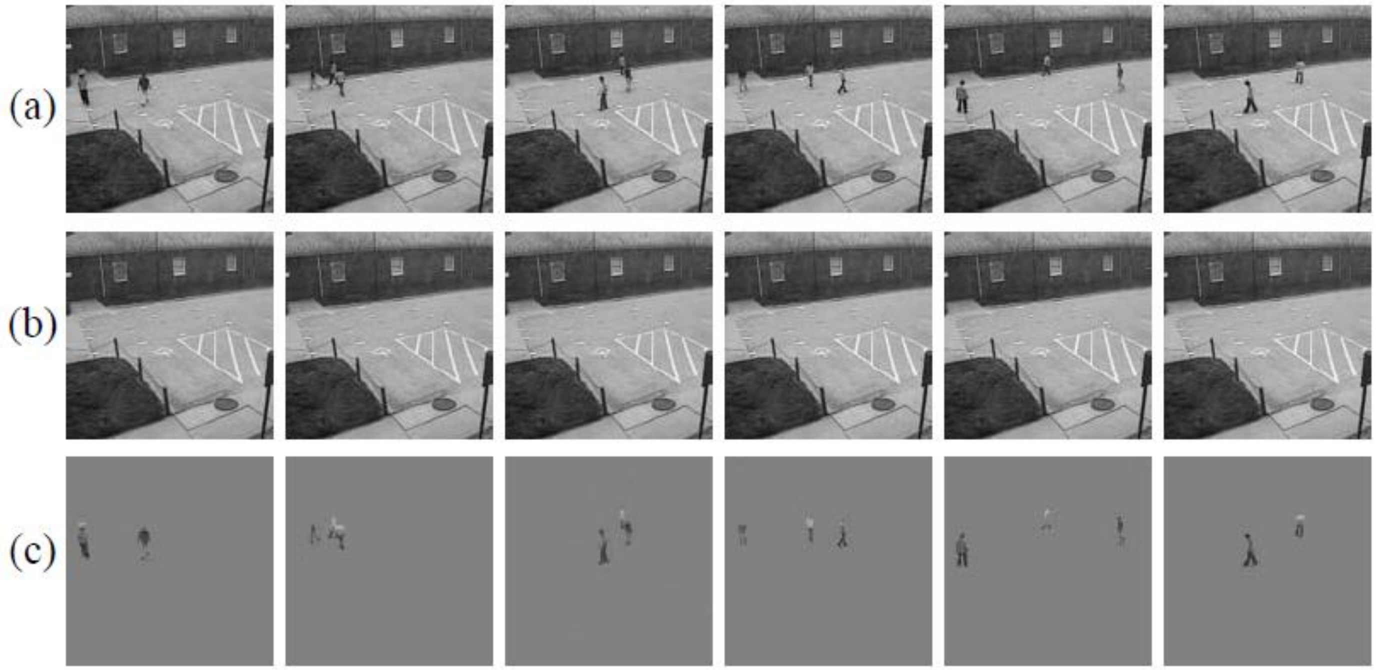


Figure 3: SpaRCS recovery results on a $128 \times 128 \times 201$ video sequence. The video sequence is reshaped into an $N_1 \times N_2$ matrix with $N_1 = 128^2$ and $N_2 = 201$. (a) Ground truth for several frames. (b) Estimated low-rank component $\mathbf{L}$. (c) Estimated sparse component $\mathbf{S}$. The recovery SNR is 31.2 dB at the measurement ratio $p/(N_1 N_2) = 0.15$. The recovery is accurate in spite of the measurement operator $\mathcal{A}$ working independently on each frame.

Follows Rice SPC model, independent compressive measurements on each frame of the matrix **M** representing the video.

https://papers.nips.cc/paper/4438-sparcs-recovering-low-rank-and-sparse-matrices-from-compressive-measurements.pdf

Results: Video CS

Figure 4: SpaRCS recovery results on a $64 \times 64 \times 234$ video sequence. The video sequence is reshaped into an $N_1 \times N_2$ matrix with $N_1 = 64^2$ and $N_2 = 234$. (a) Ground truth for several frames. (b) Recovered frames. The recovery SNR is 23.9 dB at the measurement ratio of $p/(N_1 N_2) = 0.33$. The recovery is accurate in spite of the changing illumination conditions.

Follows Rice SPC model, independent compressive measurements on each frame of the matrix **M** representing the video.

https://papers.nips.cc/paper/4438-sparcs-recovering-low-rank-and-sparse-matrices-from-compressive-measurements.pdf

Results: Hyperspectral CS

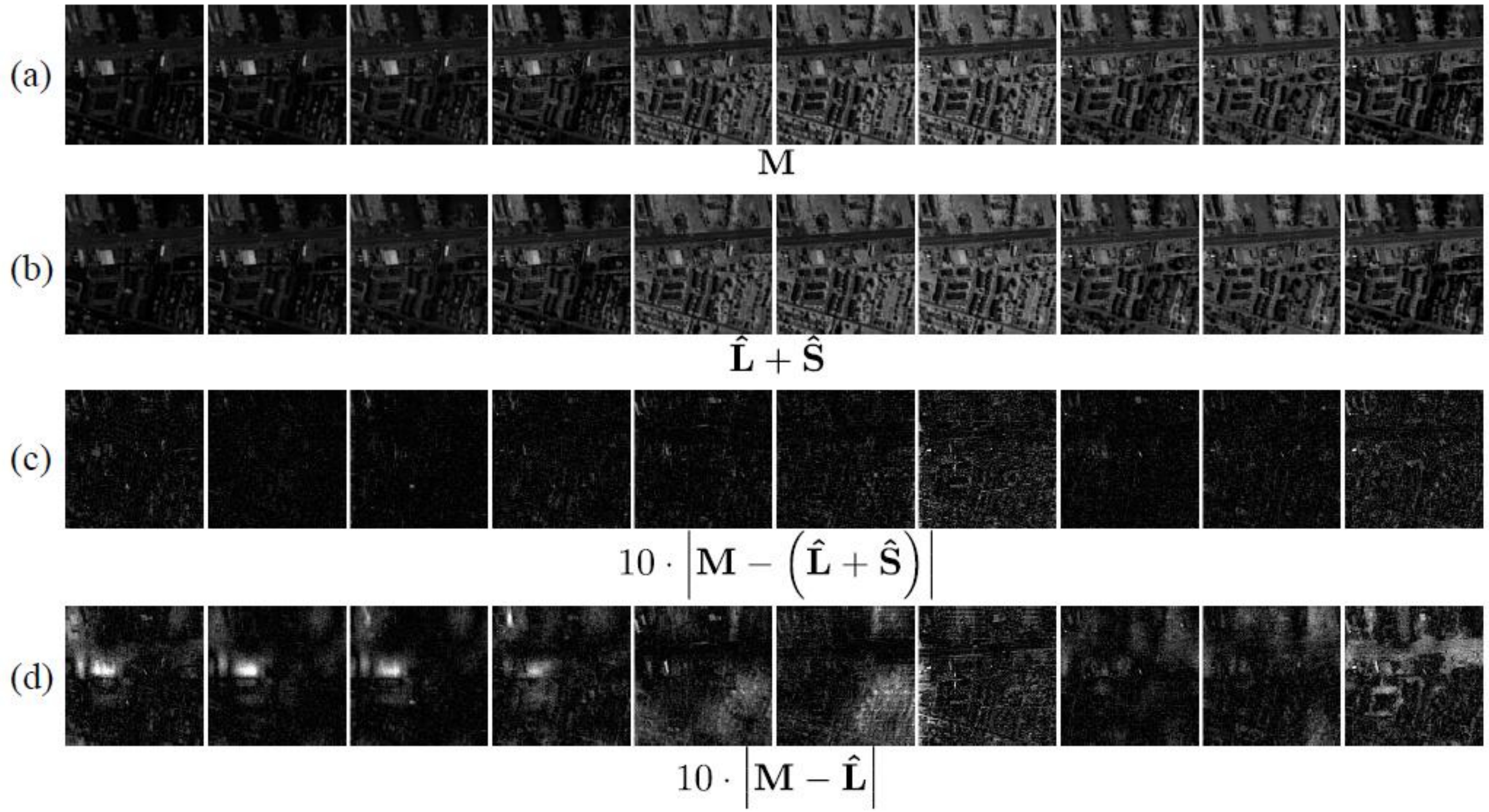


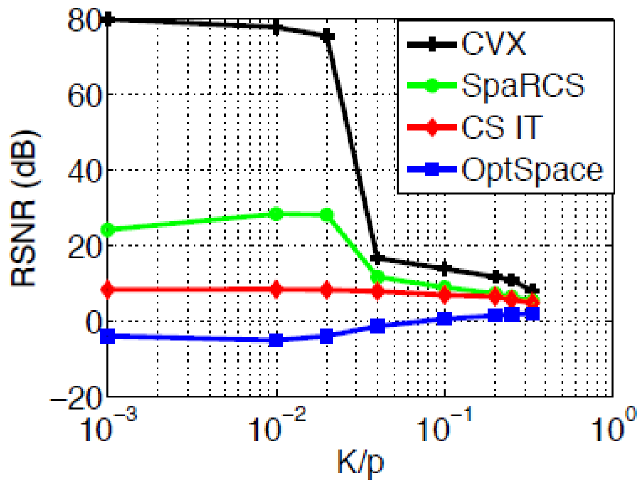
Figure 5: SpaRCS recovery results on a $128 \times 128 \times 128$ hyperspectral data cube. The hyperspectral data is reshaped into an $N_1 \times N_2$ matrix with $N_1 = 128^2$ and $N_2 = 128$. Each image pane corresponds to a different spectral band. (a) Ground truth. (b) Recovered images. (c) Residual error using both the low-rank and sparse component. (d) Residual error using only the low-rank component. The measurement ratio is $p/(N_1 N_2) = 0.15$.

https://papers.nips.cc/paper/4438-sparcs-recovering-low-rank-and-sparse-matrices-from-compressive-measurements.pdf

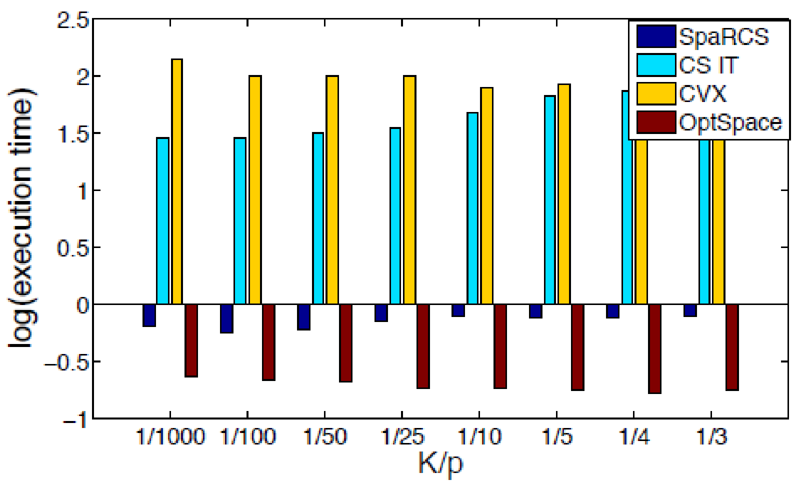
Rice SPC model of CS measurements on every spectral band

Results: Robust matrix completion

https://papers.nips.cc/paper/4438-sparcs-recovering-low-rank-and-sparse-matrices-from-compressive-measurements.pdf



(a) Performance



(b) Timing plot

Figure 7: Comparison of several algorithms for the robust matrix completion problem. (a) RSNR averaged over 10 Monte-Carlo runs for an $N \times N$ matrix completion problem with $N = 128$, $r = 1$, and $p/N^2 = 0.2$. Non-robust formulations, such OptSpace, fail. SpaRCS acheives performance close to that of the convex solver (CVX). (b) Comparison of convergence times for the various algorithms. SpaRCS converges in only a fraction of the time required by the other algorithms.

$$\min \|\mathbf{L}\|_* + \lambda\|\mathbf{s}\|_1 \quad \text{subject to } \mathbf{L}_\Omega + \mathbf{s} = \mathbf{y}$$

Theorem for Compressive PCP

Theorem 2.1 (Compressive PCP Recovery). *Let $\boldsymbol{L}_0, \boldsymbol{S}_0 \in \mathbb{R}^{m \times n}$, with $m \geq n$, and suppose that $\boldsymbol{L}_0 \neq \boldsymbol{0}$ is a rank-r, μ-incoherent matrix with*

$$r \leq \frac{c_r n}{\mu \log^2 m}, \tag{2.4}$$

and $\operatorname{sign}(\boldsymbol{S}_0)$ *is iid Bernoulli-Rademacher with nonzero probability $\rho < c_\rho$. Let $Q \subset \mathbb{R}^{m \times n}$ be a random subspace of dimension*

$$\dim(Q) \geq C_Q \cdot (\rho m n + m r) \cdot \log^2 m \tag{2.5}$$

distributed according to the Haar measure, probabilistically independent of $\operatorname{sign}(\boldsymbol{S}_0)$*. Then with probability at least $1 - C m^{-9}$ in $(\operatorname{sign}(\boldsymbol{S}_0), Q)$, the solution to*

$$\text{minimize} \quad \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \mathcal{P}_Q[\boldsymbol{L} + \boldsymbol{S}] = \mathcal{P}_Q[\boldsymbol{L}_0 + \boldsymbol{S}_0] \tag{2.6}$$

with $\lambda = 1/\sqrt{m}$ is unique, and equal to $(\boldsymbol{L}_0, \boldsymbol{S}_0)$. Above, c_r, c_ρ, C_Q, C are positive numerical constants.

Q is obtained from the linear span of different independent N(0,1) matrices with iid entries

Wright et al, “Compressive Principal Component Pursuit”
http://yima.csl.illinois.edu/psfile/CPCP.pdf

Summary

- Low rank matrix completion: motivation, key theorems, numerical results
- Algorithm for low rank matrix completion
- Robust PCA
- (Compressive) low rank matrix recovery
- Compressive RPCA
- Several papers linked on moodle