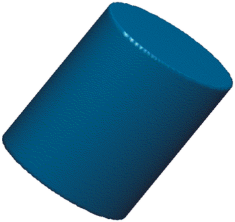
Computational Principles for High-dim Data Analysis

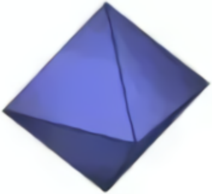
(Lecture Eleven)

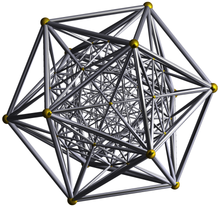
Yi Ma and Jiantao Jiao

EECS Department, UC Berkeley

October 5, 2021







Decomposing Low-Rank and Sparse Matrices
(Principal Component Pursuit: Algorithms)

1. Problem and Motivating Example

2. Principal Component Pursuit

3. Conditions for Correct Decomposition

"The whole is greater than the sum of the parts."
– Aristotle, *Metaphysics*

Problem Formulation: Mixture of Sparse and Low-Rank

Given a large data matrix $\boldsymbol{Y} \in \mathbb{R}^{n_1 \times n_2}$ which is a superposition of two unknown matrices:

$$\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{S}_o, \tag{1}$$

where

- $\boldsymbol{L}_o \in \mathbb{R}^{n_1 \times n_2}$ is a low-rank matrix;
- $\boldsymbol{S}_o \in \mathbb{R}^{n_1 \times n_2}$ is a sparse matrix.

Problem: Can we hope to efficiently recover both $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$?

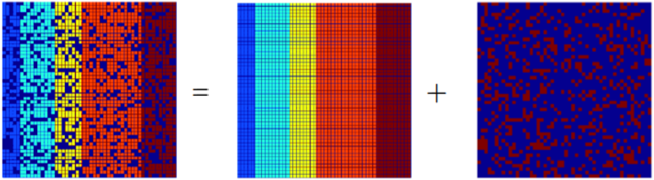
Compare this with the classic model for PCA:

$$\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{Z}_o, \tag{2}$$

where $\boldsymbol{Z}_o$ is dense but small, say Gaussian, noise?

PCA versus Robust PCA.

Complexity of Low-Rank Sparse Decomposition



Definition (Matrix Rigidity)

The *rigidity* of a matrix $\boldsymbol{M}$ (relative to rank r matrices) is defined to be:

$$R_{\boldsymbol{M}}(r) \doteq \min\{\|\boldsymbol{S}\|_0 : \text{rank}(\boldsymbol{M}+\boldsymbol{S}) \leq r\}, \tag{3}$$

the smallest # of entries modified in order to change $\boldsymbol{M}$ rank r.

Computing matrix rigidity is NP-Hard[1], so is decomposition.

[1] On the complexity of matrix rank and rigidity. Meena Mahajan and Jayalal Sarma M.N., 2007

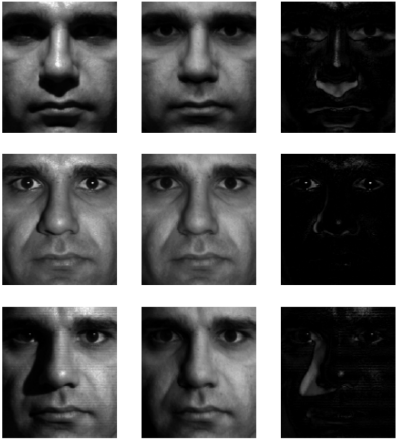
Examples of Low-Rank Sparse Decomposition

Example. A sequence of video frames can be modeled as a static background (low-rank) and moving foreground (sparse).

(a) Original frames (b) Low-rank $\hat{L}$ (c) Sparse $\hat{S}$

Examples of Low-Rank Sparse Decomposition

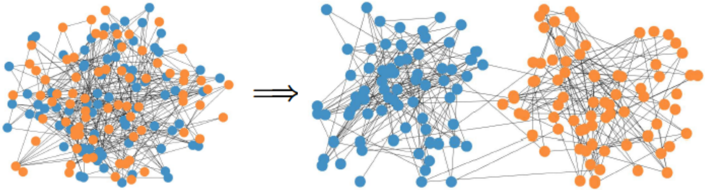
Example. A set of face images of the same person under different lightings can be modeled as a low-dimensional, $3 \sim 9$D (see Chapter 14), subspace and sparse occlusions and corruptions (specularities).



(a) $\boldsymbol{Y}$ (b) $\hat{\boldsymbol{L}}$ (c) $\hat{\boldsymbol{S}}$

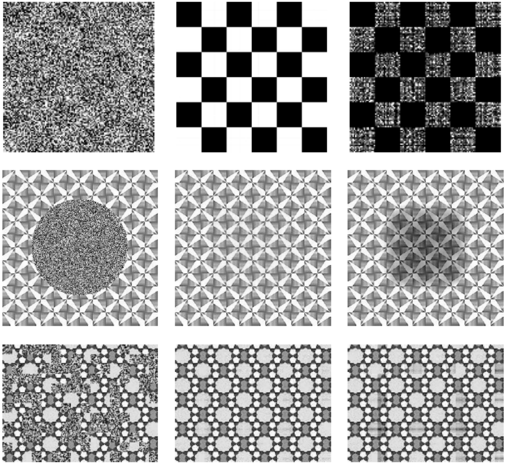
Examples of Low-Rank Sparse Decomposition

Example. Finding communities in a large social networks. Each community can be modeled as a clique of the social graph $\mathcal{G}$, hence a rank-1 block in the connectivity matrix $\boldsymbol{M}$. Hence $\boldsymbol{M}$ is a low-rank matrix and some sparse connections across communities.



Examples of Low-Rank Sparse Decomposition

Example. Structured regular texture recovery (Chapter 15).



(a) Input image (b) Sparse + Low-rank (c) Low-rank only

and many more...

Convex Relaxation: Principal Component Pursuit

Optimization problem:

$$\text{minimize} \quad \text{rank}(\boldsymbol{L}) + \lambda \|\boldsymbol{S}\|_0 \quad \text{subject to} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}, \tag{4}$$

which is intractable. Consider **convex relaxation**:

$$\|\boldsymbol{S}\|_0 = \#\{S_{ij} \neq 0\} \quad \rightarrow \quad \|\boldsymbol{S}\|_1 = \sum_{ij} |S_{ij}| \quad (\ell^1 \text{norm}). \tag{5}$$

$$\text{rank}(\boldsymbol{L}) = \#\{\sigma_i(\boldsymbol{L}) \neq 0\} \quad \rightarrow \quad \|\boldsymbol{L}\|_* = \sum_{i} \sigma_i(\boldsymbol{L}) \quad (\text{nuclear norm}) \tag{6}$$

Principal Component Pursuit (PCP):

$$\text{minimize} \quad \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}. \tag{7}$$

Alternating Directions Method of Multipliers (ADMM)

Augmented Lagrangian

$$\mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}) = \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 + \langle \boldsymbol{\Lambda}, \boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y} \rangle + \frac{\mu}{2}\|\boldsymbol{L} + \boldsymbol{S} - Y\|_F^2 \tag{8}$$

Instead of

$$(\boldsymbol{L}_{k+1}, \boldsymbol{S}_{k+1}) = \arg\min_{\boldsymbol{L},\boldsymbol{S}} \mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}_k), \tag{9}$$

we realize

$$\arg\min_{\boldsymbol{S}} \mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}) = \mathcal{S}_{\lambda/\mu}(\boldsymbol{Y} - \boldsymbol{L} - \mu^{-1}\boldsymbol{\Lambda}) \tag{10}$$

$$\arg\min_{\boldsymbol{L}} \mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}) = \mathcal{D}_{1/\mu}(\boldsymbol{Y} - \boldsymbol{S} - \mu^{-1}\boldsymbol{\Lambda}) \tag{11}$$

Soft-thresholding Operators

Recall

$$\mathcal{S}_\tau(x) = \mathsf{sgn}(x)\max(|x| - \tau, 0) \tag{12}$$

For matrix $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, we define *the singular value thresholding operator*:

$$\mathcal{D}_\tau(\boldsymbol{M}) = \boldsymbol{U}\mathcal{S}_\tau(\boldsymbol{\Sigma})\boldsymbol{V}^*. \tag{13}$$

Dominating computation is $\mathcal{D}_{1/\mu}$, can speed up using partial SVD.

Algorithm: Alternating Direction Minimization

initialize: $\boldsymbol{S}_0 = \boldsymbol{\Lambda}_0 = 0, \mu > 0.$
while not converged **do**
    compute $\boldsymbol{L}_{k+1} = \mathcal{D}_{1/\mu}(\boldsymbol{Y} - \boldsymbol{S}_k - \mu^{-1}\boldsymbol{\Lambda}_k)$
    compute $\boldsymbol{S}_{k+1} = \mathcal{S}_{\lambda/\mu}(\boldsymbol{Y} - \boldsymbol{L}_{k+1} - \mu^{-1}\boldsymbol{\Lambda}_k)$
    compute $\boldsymbol{\Lambda}_{k+1} = \boldsymbol{\Lambda}_k + \mu\big(\boldsymbol{L}_{k+1} + \boldsymbol{S}_{k+1} - \boldsymbol{Y}\big).$
end while

Algorithm: A Little Lesson from History

Comparison from chronological development of algorithms for solving the PCP problem: **the older the algorithm, the more efficient!**

GOOD NEWS: Scalable first-order gradient-descent algorithms:

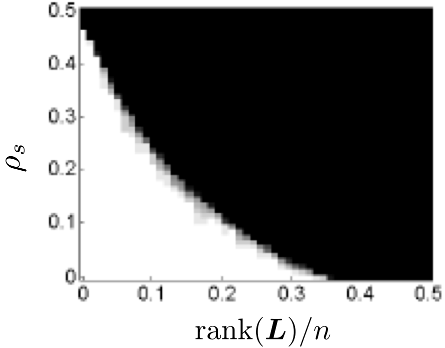
- Proximal Gradient [Osher, Mao, Dong, Yin '09,Wright et. al.'09, Cai et. al.'09].
- Accelerated Proximal Gradient [Nesterov '83, Beck and Teboulle '09]:
- Augmented Lagrange Multiplier [Hestenes '69, Powell '69]:
- Alternating Direction Method of Multipliers [Gabay and Mercier '76].

For a 1000x1000 matrix of rank 50, with 10% (100,000) entries randomly corrupted: $\min \ \|A\|_* + \lambda\|E\|_1 \quad \text{subj} \quad A + E = D.$

Algorithms	Accuracy	Rank	||E||_0	# iterations	time (sec)
IT	5.99e-006	50	101,268	8,550	119,370.3
DUAL	8.65e-006	50	100,024	822	1,855.4
APG	5.85e-006	50	100,347	134	1,468.9
APG_P	5.91e-006	50	100,347	134	82.7
$EALM_P$	2.07e-007	50	100,014	34	37.5
$IALM_P$	3.83e-007	50	99,996	23	11.8

10,000 times speedup!

Empirical Success Rate



$n = 400$

Fig. credit: Candès, Li, Ma, Wright '11

When Is Decomposition Possible?

Identifiability issue: a matrix might be simultaneously low-rank and sparse!

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{\text{sparse and low-rank}} \quad \text{vs.} \quad \underbrace{\begin{bmatrix} 1 & 0 & 1 & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & 1 \end{bmatrix}}_{\text{sparse but not low-rank}}$$

Nonzero entries of sparse component need to be spread out

— This lecture: assume locations of the nonzero entries are random

When Is Decomposition Possible?

Identifiability issue: a matrix might be simultaneously low-rank and sparse!

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & & \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}}_{\text{low-rank and dense}} \quad \text{vs.} \quad \underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{\text{low-rank but sparse}}.$$

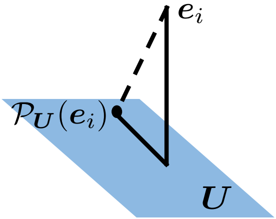
The low-rank component needs to be incoherent.

Low-rank Component: Incoherence

Definition (Incoherent Low-rank Matrix)

Incoherence parameter μ_1 of $\boldsymbol{L}_o = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ is the smallest quantity s.t.

$$\max_i \|\boldsymbol{e}_i^* \boldsymbol{U}\|_2^2 \leq \frac{\mu_1 r}{n} \quad \text{and} \quad \max_i \|\boldsymbol{e}_i^* \boldsymbol{V}\|_2^2 \leq \frac{\mu_1 r}{n}.$$



Low-rank Component: Joint Coherence

Definition (Joint Coherence)

Joint coherence parameter μ_2 of $\boldsymbol{L}_o = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ is the smallest quantity s.t.

$$\|\boldsymbol{U}\boldsymbol{V}^*\|_\infty \leq \sqrt{\frac{\mu_2 r}{n^2}}.$$

This prevents $\boldsymbol{U}\boldsymbol{V}^*$ from being too peaky

- $\mu_1 \leq \mu_2 \leq \mu_1^2 r$, since

$$|(\boldsymbol{U}\boldsymbol{V}^*)_{ij}| = |\boldsymbol{e}_i^\top \boldsymbol{U}\boldsymbol{V}_j^*| \leq \|\boldsymbol{e}_i^\top \boldsymbol{U}\|_2 \cdot \|\boldsymbol{V}_j^*\|_2 \leq \frac{\mu_1 r}{n}$$

$$\|\boldsymbol{U}\boldsymbol{V}^*\|_\infty^2 \geq \frac{\|\boldsymbol{U}\boldsymbol{V}^*\boldsymbol{e}_j\|_{\mathrm{F}}^2}{n} = \frac{\|\boldsymbol{V}_j^*\|_2^2}{n} = \frac{\mu_1 r}{n^2} \text{ (suppose } \|\boldsymbol{V}_j^*\|_2^2 = \frac{\mu_1 r}{n}\text{)}$$

In the book we have set $\mu_1 = \mu_2 = \nu$.

Theoretical Guarantee

Theorem (Candès, Li, Ma, Wright '11)

- rank$(\boldsymbol{L}) \lesssim \frac{n}{\max\{\mu_1,\mu_2\}\log^2 n}$;
- *Nonzero entries of $\boldsymbol{S}$ are randomly located, and $\|\boldsymbol{S}\|_0 \leq \rho_s n^2$ for some constant $\rho_s > 0$ (e.g. $\rho_s = 0.2$).*

Then PCP with $\lambda = 1/\sqrt{n}$ is exact with high prob.

- rank$(\boldsymbol{L})$ can be quite high (up to $n/$polylog(n))
- Parameter free: $\lambda = 1/\sqrt{n}$
- Ability to correct gross error: $\|\boldsymbol{S}\|_0 \asymp n^2$
- Sparse component $\boldsymbol{S}$ can have arbitrary magnitudes / signs!

Geometry

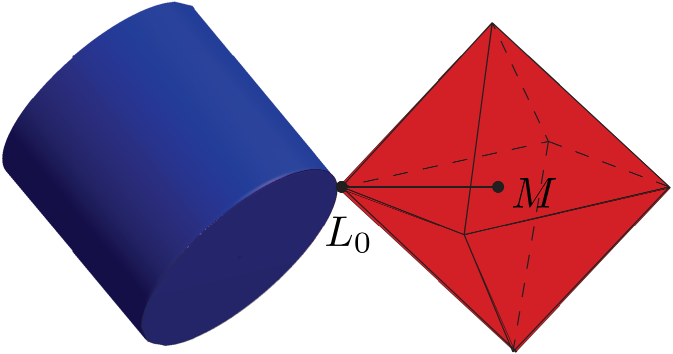


Fig. credit: Candès '14

Dense Error Correction

Theorem (Ganesh, Wright, Li, Candès, Ma '10, Chen, Jalali, Sanghavi, Caramanis '13)

- rank$(\boldsymbol{L}) \lesssim \frac{n}{\max\{\mu_1,\mu_2\}\log^2 n}$;
- *Nonzero entries of* $\boldsymbol{S}$ *are randomly located, have* random sign, *and* $\|\boldsymbol{S}\|_0 = \rho_s n^2$.

Then PCP with $\lambda \asymp \sqrt{\frac{1-\rho_s}{\rho_s n}}$ *succeeds with high prob., provided that*

$$\underbrace{1-\rho_s}_{\textit{non-corruption rate}} \quad \gtrsim \quad \sqrt{\frac{\max\{\mu_1,\mu_2\} r \,\mathrm{polylog}(n)}{n}}.$$

- When additive corruptions have random signs, PCP works even when *a dominant fraction* of the entries are corrupted!

Is Joint Coherence Needed?

- Matrix completion: does not need μ_2.
- Robust PCA: so far we need μ_2.

Question: Can we recover $\boldsymbol{L}$ with rank up to $\frac{n}{\mu_1 \text{polylog}(n)}$ (rather than $\frac{n}{\max\{\mu_1,\mu_2\}\text{polylog}(n)}$), with tractable solutions?

Answer: highly unlikely...

Planted Clique Problem

Setup: a graph $\mathcal{G}$ of n nodes generated as follows

1. connect each pair of nodes independently with prob. 0.5;
2. pick n_0 nodes and make them a clique (fully connected).

Goal: find the hidden clique from $\mathcal{G}$

Information theoretically, one can recover the clique if $n_0 > 2\log_2 n$.

Conjecture on Computational Barrier

Conjecture: $\forall$ constant $\epsilon > 0$, if $n_0 \leq n^{0.5-\epsilon}$, then no tractable algorithm can find the clique from $\mathcal{G}$ with prob. $1 - o(1)$.

— often used as a hardness assumption in theoretical computer science.

Lemma (Sharpness of the Conditions)

If there is an algorithm that allows recovery of any $\boldsymbol{L}$ from $\boldsymbol{Y}$ with $\text{rank}(\boldsymbol{L}) \leq \frac{n}{\mu_1 \text{polylog}(n)}$, *then the above conjecture is violated.*

Proof of Lemma

Suppose $\boldsymbol{L}$ is the true adjacency matrix,

$$L_{i,j} = \begin{cases} 1, & \text{if } i, j \text{ are both in the clique;} \\ 0, & \text{else.} \end{cases}$$

Let $\boldsymbol{A}$ be the adjacency matrix of $\mathcal{G}$, and generate $\boldsymbol{Y}$ s.t.

$$Y_{i,j} = \begin{cases} A_{i,j}, & \text{with prob. } 2/3; \\ 0, & \text{else.} \end{cases}$$

Therefore, one can write

$$\boldsymbol{Y} = \boldsymbol{L} + \underbrace{\boldsymbol{Y} - \boldsymbol{L}}_{\text{each entry is nonzero w.p. } 1/3}.$$

Proof of Lemma

Note that

$$\mu_1 = \frac{n}{n_0} \qquad \text{and} \qquad \mu_2 = \frac{n^2}{n_0^2}.$$

If there is an algorithm that can recover any $\boldsymbol{L}$ of rank $\frac{n}{\mu_1 \text{polylog}(n)}$ from $\boldsymbol{M}$, then

$$\text{rank}(\boldsymbol{L}) = 1 \leq \frac{n}{\mu_1 \text{polylog}(n)} \quad \Longleftrightarrow \quad n_0 \geq \text{polylog}(n).$$

But this contradicts the conjecture (which claims computational infeasibility to recover $\boldsymbol{L}$ unless $n_0 \geq n^{0.5-o(1)}$).

Assignments

- Reading: Section 5.1 - 5.3 of Chapter 5.
- Written Homework #3.