EE227 Project Report
Robust Principal Component Analysis

Maximilian Balandat Walid Krichene Chi Pang Lam Ka Kit Lam

May 10, 2012

Introduction

Over the past decade there has been an explosion in terms of the massive amounts of high-dimensional data in almost all fields of science and engineering. This situation presents a challenge as well as an opportunity to many areas such as web data analysis, search, biomedical imaging, bioinformatics, (social) network analysis, image, video and multimedia processing and many others. In such applications, researches today routinely deal with data that lie in thousands or even billions of dimensions, with a number of samples sometimes of the same order of magnitude.

Often, in order to be able to even make sense of data in such high dimensionality and scale, one has to leverage on the fact that such data often have low intrinsic dimensionality. One reasonable (but not the only) approach that is well justified in many practical situations is to assume that the data all lie near some low-dimensional subspace. In mathematical terms, this amounts to saying that if all the data points are stacked as column vectors of a matrix M, the matrix should (approximately) have low rank. Specifically, a common model is to assume that

$$M = L_0 + N_0 \tag{1}$$

where L_0 is of low rank and N_0 is a small perturbation matrix (i.e. the noise). A classical approach to the problem is to seek the best (in an ℓ_2-sense) rank-k estimate of L_0 by solving the optimization problem

$$\min_L \|M - L\|_2 \quad : \quad \mathrm{rk}(L) \leq k \tag{2}$$

This approach, known as Principal Component Analysis (PCA) [12, 13], has been studied for almost a century and is used extensively for data analysis and dimensionality reduction in many areas. It is easy to show that this problem can be efficiently solved using the singular value decomposition (SVD) of the data matrix M. It also enjoys a number of optimality properties when the noise N_0 is small and i.i.d. Gaussian. However, PCA in general may fail spectacularly when the assumption on the noise matrix is not satisfied: Even a single grossly corrupted entry in M could render the estimated $\hat{L}$ arbitrarily far from the true L_0. Unfortunately, these kinds of gross errors are very common in many applications, arising for example from corrupted data, sensor failures or corrupted samples in repetitive measurement tasks in biology applications.

We would therefore like a method that is able to extract the principal components (the low rank structure) of measurement data even in the presence of such gross but sparse corruptions. The recently proposed Robust PCA framework [10] is a very promising candidate for this task. Robust PCA combines the two popular heuristics of nuclear norm minimization (used to encourage low-rankness) and ℓ_1-norm minimization (used to encourage sparsity) and casts the problem of separating a low-rank "data" component from a sparse "noise" component into a convex optimization problem. The surprising fact is that one can show that, under certain reasonable conditions, the recovered solution is exact.

The objective of this project is threefold: First, we survey the recent results on Robust PCA and related extensions and use the theory of convex optimization to develop an understanding of how the main theoretical results are proven. Second, we discuss some efficient algorithms for Robust PCA that allow the application of the framework to large-scale problems. Finally, we apply the Robust PCA framework to a number of different problems in order to illustrate its potential practical relevance.

Contents

Chapter 1

Theory

1.1 Introduction

Given an observed matrix $M \in \mathbb{R}^{n_1 \times n_2}$ that is formed as a superposition of a low-rank matrix L_0 and a sparse matrix S_0,

$$M = L_0 + S_0$$

Robust Principal Component Analysis [10] is the problem of recovering the low-rank and sparse components. Under suitable assumptions on the rank and incoherence of L_0, and the distribution of the support of S_0, the components can be recovered exactly with high probability, by solving the Principal Component Pursuit (PCP) problem given by

$$\begin{aligned} &\text{minimize} && \|L\|_* + \lambda \|S\|_1 \\ &\text{subject to} && L + S = M \end{aligned} \tag{1.1}$$

Principal component pursuit minimizes a linear combination of the nuclear norm of a matrix L and the ℓ_1 norm of $M - L$. Minimizing the ℓ_1 norm is known to favor sparsity, while minimizing the nuclear norm $\|L\|_* = \sum_{\sigma \in \sigma(L)} \sigma$ is known to favor low-rank matrices (intuitively, favors sparsity of the vector of singular values).

The low-rank component S_0 is viewed as a noise matrix, that can represent measurement noise, failure in some sensors that will result in completely corrupting a fraction of the observed entries, or missing data (which translates to having a fraction of the entries equal to zero). In this setting, one would like to be able to recover the original data L_0, without making assumptions on the magnitude $\|S_0\|_\infty$ of the sparse component, where $\|S\|_\infty = \max_{i,j} |S_{ij}|$. PCP achieves recovery with high probability in this setting, under alternate assumptions on the structure of L_0 and sparsity pattern of S_0.

One cannot expect to recover the components exactly in the most general case. Assume for example that L_0 is such that $(L_0)_{ij} = \delta_i^1 \delta_j^1$, and $S_0 = -L_0$. Both matrices are sparse and low-rank, and clearly one cannot expect to recover the components in this case, since the observed matrix is $M = 0$. Therefore assumptions need to be made on the incoherence of L_0 and the support of S_0.

1.1.1 Incoherence of the low rank component L_0

The Incoherence conditions describe how much the singular vectors of a given matrix are aligned with the vectors of the canonical basis.

Let the (slim) SVD of L_0 be given by

$$L_0 = U\Sigma V^* = \sum_{i=1}^{r} \sigma_i u_i v_i^* \tag{1.2}$$

where $U \in \mathbb{R}^{n_1 \times r}$ and $V \in \mathbb{R}^{n_2 \times r}$ are the matrices of left and right singular vectors respectivley, $U = [u_1, \ldots, u_r]$, $V = [v_1, \ldots, v_r]$. Then the incoherence conditions are given by

$$\max_i \|U^* e_i\|_2^2 \leq \frac{\mu r}{n_1}, \quad \max_i \|V^* e_i\|_2^2 \leq \frac{\mu r}{n_2} \tag{1.3}$$

and

$$\|UV^*\|_\infty \leq \sqrt{\frac{\mu r}{n_1 n_2}} \tag{1.4}$$

Note that the condition $\|U^* e_i\|_2^2 \leq \frac{\mu r}{n_1}$ translates to $\sum_{k=1}^{r} (u_k)_i^2 \leq \frac{\mu r}{n_1}$. Also note that the orthogonal projection P_U on $\text{Span}(u_1, \ldots, u_r)$ is given by

$$UU^* = [u_1, \ldots, u_r] \begin{bmatrix} u_1^* \\ \vdots \\ u_r^* \end{bmatrix} = \sum_{k=1}^{r} u_k u_k^*$$

and the condition is equivalent to $\|P_U e_i\|_2^2 \leq \frac{\mu r}{n_1}$ since $\|U^* e_i\|_2^2 = e_i^*(UU^*)e_i = e_i^* P_U e_i = (e_i - P_U e_i + P_U e_i)^* P_U e_i = \|P_U e_i\|_2^2$ ($P_U e_i$ and $e_i - P_U e_i$ are orthogonal). Or simply $\|P_U e_i\|_2^2 = e_i^* UU^* UU^* e_i = e_i^* UU^* e_i = \|U^* e_i\|_2^2$ since $U^* U = I_r$.

These conditions require the singular vectors to be "spread" enough with respect to the canonical basis. Intuitively, if the singular vectors of the low-rank matrix L_0 are aligned with a few canonical basis vectors, then L_0 will be sparse and hard to distinguish from the sparse corruption matrix S_0.

1.1.2 Support of the sparse component S_0

The cardinality of the support of S_0 is denoted m. Guaranteeing exact recovery requires m to be small enough, in a sense that will be defined in the next section. Proving exact recovery will rely on a probabilistic argument which assumes the sparse matrix S_0 is drawn from a uniform distribution on the set of matrices with support of a fixed cardinality m, i.e. $\{S \in \mathbb{R}^{n_1 \times n_2} \mid \textbf{card}(\textbf{supp}(S)) = m\}$. Here **card** denotes the cardinality, and **supp** denotes the set of non zero elements, or support. The proof of the main result will use a different sampling model, and prove equivalence with the uniform model.

1.2 Main Result

Theorem 1. *Suppose $L_0 \in \mathbb{R}^{n\times n}$ satisfies incoherence conditions (1.3) and (1.4) and that the support of S_0 is uniformly distributed among all sets of cardinality m. Then $\exists\, c$ such that with high probability over the choice of support of S_0 (at least $1 - cn^{-10}$), Principal Component Pursuit with $\lambda = 1/\sqrt{n}$ is exact, i.e. $\hat{L} = L_0$ and $\hat{S} = S_0$ provided that*

$$rank(L_0) \leq \frac{\rho_r}{\mu}\frac{n}{(\log n)^2} \quad \text{and} \quad m \leq \rho_s n^2 \tag{1.5}$$

Above, ρ_r and ρ_s are positive numerical constants. Note in particular that no assumptions are made on the magnitudes of the nonzero entries of S_0.

The first condition in the theorem bounds the rank of L_0, but also how spread the singular vectors have to be, since we need to have $\forall i$ (from the incoherence condition)

$$\|U^* e_i\|_2^2 \leq \frac{\mu \operatorname{rank}(L_0)}{n} \leq \frac{\rho_r}{(\log n)^2}$$

The second condition bounds the size m of the support of S_0.

1.3 Proof of the main result

The main arguments of the proof are the following:

First, change the model of the sparse matrix S_0 from the uniform sampling model, to the Bernoulli sampling model with fixed signs, then to the Bernoulli sampling model with random signs. To show equivalence of the results under the different sampling models, use an elimination theorem.

Then using the random sign Bernoulli sampling model, it is shown that a dual certificate can be constructed with high probability, proving that (L_0, S_0) is the unique optimizer, by constructing a subgradient that shows that any non-zero perturbation H will result in a strict increase in the objective value $\|L_0 + H\|_* + \lambda\|S_0 - H\|_1$.

1.3.1 Preliminaries

- The subgradient of of the ℓ_1 norm at S_0 supported on Ω is of the form $\operatorname{sgn}(S_0) + F$ where $P_\Omega F = 0$ and $\|F\|_\infty \leq 1$.
- The subgradient of the nuclear norm (for details see Section 1.3.9) at $L_0 = U\Sigma V^*$ where $U, V \in \mathbb{R}^{n\times r}$ and $\Sigma \in \mathbb{R}^{r\times r}$, is of the form $UV^* + W$, where

$$\begin{aligned} U^*W &= 0 \\ WV &= 0 \\ \|W\| &\leq 1 \end{aligned} \tag{1.6}$$

  where $\|W\|$ denotes the operator norm of matrix W, i.e. $\|W\| = \max_{\|u\|_2=1} \|Wu\|_2 = \sigma_{\max}(W)$. Conditions (1.6) are equivalent to

$$\begin{aligned} P_T W &= 0 \\ \|W\| &\leq 1 \end{aligned} \tag{1.7}$$

where T is the linear space of matrices defined by

$$T = \{UX^* + YV^*,\ X, Y \in \mathbb{R}^{n\times r}\} \tag{1.8}$$

Indeed, we have

$$\begin{aligned} P_T W = 0 &\Leftrightarrow W \in T^\perp \\ &\Leftrightarrow \forall M \in T,\ Tr(W^* M) = 0 \\ &\Leftrightarrow \forall X, Y \in \mathbb{R}^{n\times r},\ Tr(W^*(UX^* + YV^*)) = 0 \\ &\Leftrightarrow \forall X, Y \in \mathbb{R}^{n\times r},\ Tr((U^*W)^* X^*) + Tr((WV)^* Y) = 0 \\ &\Leftrightarrow U^*W = WV^* = 0 \end{aligned}$$

Note that the projection on the orthogonal of T is given by

$$P_{T^\perp} M = (I - UU^*)M(I - VV^*) \tag{1.9}$$

Proof. Note that UU^* is the orthogonal projection on the subspace spanned by the columns of U, and similarly for VV^*. Let $P_U = UU^*$, $P_{U^\perp} = I - UU^*$, and similarly for V.

Let $M_1 = (I - UU^*)M(I - VV^*) = P_{U^\perp} M P_{V^\perp}$. We have

$$P_{T^\perp} M = M_1 \Leftrightarrow (M - M_1 \in T \text{ and } M_1 \perp M - M_1)$$

and we have $M - M_1 = UU^*M + MVV^* - UU^*MVV^* = U(U^*M) + (MV - UU^*MV)V^* \in T$, and

$$\begin{aligned} Tr(M_1^*(M - M_1)) &= Tr((I - VV^*)M^*(I - UU^*)(UU^*M + MVV^* - UU^*MVV^*)) \\ &= Tr(P_{V^\perp} M^* P_{U^\perp}(P_U M + (M - UU^*M)P_V)) \\ &= Tr(P_{V^\perp} M^* P_{U^\perp} P_U M) + Tr((M - UU^*M)P_V P_{V^\perp} M^* P_{U^\perp}) \\ &= 0 \end{aligned}$$

using the fact that $P_{U^\perp} P_U = P_V P_{V^\perp} = 0$ (projecting consecutively on a subspace and its orthogonal yields 0, or simply expanding, $(I - UU^*)UU^* = UU^* - UU^*UU^* = UU^* - UI_r U^* = 0$). This completes the proof. □

Note that since $P_{T^\perp}$ is an orthogonal projection, we have

$$\|P_{T^\perp} M\| \leq \|M\| \tag{1.10}$$

and for any dyad $e_i e_j^*$, we have

$$\begin{aligned} \|P_{T^\perp} e_i e_j^*\|_F^2 &= Tr\left((I - UU^*)e_i e_j^*(I - VV^*)(I - VV^*)^* e_j e_i^*(I - UU^*)^*\right) \\ &= Tr\left(e_i^*(I - UU^*)^*(I - UU^*)e_i e_j^*(I - VV^*)(I - VV^*)^* e_j\right) \\ &= Tr\left(e_i^*(I - UU^*)^*(I - UU^*)e_i\right) Tr\left(e_j^*(I - VV^*)(I - VV^*)^* e_j\right) \\ &= \|(I - UU^*)e_i\|_2^2 \|(I - VV^*)e_j\|_2^2 \end{aligned}$$

and since UU^* is an orthogonal projection, we have

$$\begin{aligned}\|(I - UU^*)e_i\|_2^2 &= \|e_i\|_2^2 - \|UU^*e_i\|_2^2 \\ &\geq 1 - \mu r/n\end{aligned}$$

where the last inequality results form the incoherence condition (1.4), $\|U^*e_i\|_2^2 \leq \frac{\mu r}{n}$. Therefore

$$\|P_{T^\perp} e_i e_j^*\|_F^2 \geq (1 - \mu r/n)^2 \tag{1.11}$$

Equivalently, using the fact that $\|P_{T^\perp} e_i e_j^*\|_F^2 + \|P_T e_i e_j^*\|_F^2 = \|e_i e_j^*\|_F^2 = 1$, we have

$$\begin{aligned}\|P_T e_i e_j^*\|_F^2 &\leq 1 - \left(1 - \frac{\mu r}{n}\right)^2 \\ &= \frac{2\mu r}{n} - \left(\frac{\mu r}{n}\right)^2 \\ &\leq \frac{2\mu r}{n}\end{aligned}$$

Thus

$$\|P_T e_i e_j^*\|_F^2 \leq \frac{2\mu r}{n} \tag{1.12}$$

1.3.2 Elimination Theorem

The following elimination theorem states the intuitive fact that if PCP exactly recovers the components of $M = L + S$, then it also exactly recovers the components of $M = L + S'$ where S' is a trimmed version of S ($\text{supp}(S') \subset \text{supp}(S)$ and S and S' coincide on $\text{supp}(S')$).

Theorem 2. *Suppose the solution to the PCP problem (1.1) with input data $M_0 = L_0 + S_0$ is unique and exact, and consider $M_0' = L_0 + S_0'$ where S_0' is a trimmed version of S_0. Then the solution to (1.1) with input M_0' is exact as well.*

Proof. Let $S_0' = P_{\Omega_0} S_0$ and let $(\hat{L}, \hat{S})$ be the solution to (1.1) with input $L_0 + S_0'$. Then since (L_0, S_0') is a feasible point for (1.1), it provides un upper bound on the optimal value

$$\|\hat{L}\|_* + \lambda \|\hat{S}\|_1 \leq \|L_0\|_* + \lambda \|S_0'\|_1$$

then decomposing S_0 into the orthogonal components $S_0 = P_{\Omega_0} S_0 + P_{\Omega_0^\perp} S_0 = S_0' + P_{\Omega_0^\perp} S_0$, we have $\|S_0'\|_1 = \|S_0\|_1 - \|P_{\Omega_0^\perp} S_0\|_1$, thus we have

$$\|\hat{L}\|_* + \lambda \|\hat{S}\|_1 + \lambda \|P_{\Omega_0^\perp} S_0\|_1 \leq \|L_0\|_* + \lambda \|S_0\|_1$$

and using the triangle inequality

$$\|\hat{L}\|_* + \lambda \|\hat{S} + P_{\Omega_0^\perp} S_0\|_1 \leq \|L_0\|_* + \lambda \|S_0\|_1$$

we observe that $(\hat{L}, \hat{S} + P_{\Omega_0^\perp} S_0)$ is feasible for the problem with input $M = L_0 + S_0$, for which the optimal value is precisely $\|L_0\|_* + \lambda \|S_0\|_1$. Therefore by uniqueness of the solution, we have

$$\begin{aligned}\hat{L} &= L_0 \\ \hat{S} + P_{\Omega_0^\perp} S_0 &= S_0\end{aligned}$$

the second equality is equivalent to $\hat{S} = S_0 - P_{\Omega_0^\perp} S_0 = P_{\Omega_0} S_0 = S_0'$. This completes the proof. $\square$

1.3.3 Derandomization

Derandomization is used to show equivalence between the problem where the signs of the entries of S_0 are random, and the problem where the entries of S_0 have fixed signs.

In the setting of Theorem 1, the non-zero entries of the sparse component S_0 are fixed, but the proof will use a stronger assumption: the signs of the non-zero entries are independent Bernoulli variables. The following theorem shows equivalence of the two settings. We remark that we take $\rho_s = \frac{m}{n^2}$ in the robust PCA setting.

Theorem 3. *Suppose L_0 satisfies conditions of Theorem 1, and that the support of S_0 is sampled from a Bernoulli model with parameter $2\rho_s$, and the signs of S_0 are i.i.d. Bernoulli ± 1 with parameter $\frac{1}{2}$, and independent from the support. Then:*

If the PCP solution is exact with high probability, then it is exact with at least the same probability for the model in which signs of S_0 are fixed and the support is sampled from a Bernoulli distribution with parameter ρ_s.

Proof. Consider the fixed values model, and let $S_0 = P_\Omega S$ for some matrix S, and the support Ω is sampled from a Bernoulli distribution. Thus the components of S_0 are independent and

$$(S_0)_{ij} = \begin{cases} S_{ij} & \text{w.p. } \rho_s \\ 0 & \text{w.p. } 1 - \rho_s \end{cases}$$

the idea of the proof is to craft a new model, and show that it is equivalent (in terms of probability distribution) to the above model.

Let E be a random sign matrix, with i.i.d. entries

$$E_{ij} = \begin{cases} 1 & \text{w.p. } \rho_s \\ 0 & \text{w.p. } 1 - 2\rho_s \\ -1 & \text{w.p. } \rho_s \end{cases}$$

and $\Delta(E)$ an elimination matrix, function of E, defined as

$$\Delta_{ij} = \begin{cases} 0 & \text{if } E_{ij} S_{ij} < 0 \text{ (E_{ij} and S_{ij} have different signs)} \\ 1 & \text{otherwise} \end{cases}$$

the entries of Δ are functions of independent variables, and are therefore independent.

Now consider the following variable

$$S_0' = \Delta \circ |S| \circ E$$

where $\circ$ is the component wise product. Then S_0 and S_0' have the same distribution. Indeed, it suffices by independence to check that they have the same marginals:

$$\begin{aligned} P((S_0')_{ij} = S_{ij}) &= P(\Delta_{ij} = 1 \text{ and } E_{ij} = \operatorname{sgn}(S_{ij})) \\ &= P(E_{ij}S_{ij} \geq 0 \text{ and } E_{ij} = \operatorname{sgn}(S_{ij})) \\ &= P(E_{ij} = \operatorname{sgn} S_{ij}) \\ &= \rho_s \end{aligned}$$

and

$$P(S_0 = S_{ij}) = \rho_s$$

Finally, since, by assumption, PCP recovers $|S| \circ E$ with high probability, then by the elimination theorem, it also recovers $\Delta \circ |S| \circ E$ with at least the same probability. The result follows since S_0' and S_0 have the same distribution. □

We remark that the uniform sampling and the iid Bernoulli sampling model are indeed equivalent and the justification is given in Section 1.3.8.

1.3.4 Dual certificate

The following lemma gives a simple sufficient condition for the pair (L_0, S_0) to be the unique optimal solution to PCP.

Lemma 1. *Assume that $\|P_\Omega P_T\| < 1$. Then (L_0, S_0) is the unique solution to PCP if $\exists (W, F)$ such that*

$$\begin{aligned} &UV^* + W = \lambda(sgn(S_0) + F) \\ &P_T W = 0 \\ &\|W\| < 1 \\ &P_\Omega F = 0 \\ &\|F\|_\infty < 1 \end{aligned} \tag{1.13}$$

Proof. We first prove that the condition $\|P_\Omega P_T\| < 1$ is equivalent to $\Omega \cap T = \{0\}$.

First, if $\Omega \cap T \neq \{0\}$, then let $M_0 \in \Omega \cap T$, $M_0 \neq 0$. We have $\|P_\Omega P_T M_0\| = \|M_0\|$, thus $\|P_\Omega P_T\| = \max_{M \neq 0} \frac{\|P_\Omega P_T M\|}{\|M\|} \geq 1$.

Conversely, if $\|P_\Omega P_T\| \geq 1$, then $\exists M_0 \neq 0$ such that $\|M_0\| \leq \|P_\Omega P_T M_0\|$. But since P_Ω and P_T are orthogonal projections, we have $\|M_0\| \leq \|P_\Omega P_T\| \leq \|P_T M_0\| \leq \|M_0\|$, where inequalities must hold with equality. In particular, we have $\|P_T M_0\| = \|M_0\|$, which implies $P_T M_0 = M_0$ (to prove this, decompose $\|M_0\|$ into the orthogonal components $\|M_0\|^2 = \|M_0 - P_T M_0\|^2 + \|P_T M_0\|^2$, thus $\|P_T M_0\| = \|M_0\| \Rightarrow \|M_0 - P_T M_0\| = 0 \Rightarrow M_0 = P_T M_0$), then similarly, $\|P_\Omega M_0\| = \|M_0\|$, which implies $P_\Omega M_0 = M_0$. Therefore $M_0 \in \Omega \cap T$. This proves the equivalence $\|P_\Omega P_T\| < 1 \Leftrightarrow \Omega \cap T = \{0\}$.

To prove that (L_0, S_0) is the unique optimizer, we show that for any feasible perturbation $(L_0 + H, S_0 - H)$ where $H \neq 0$ strictly increases the objective. Let

- $UV^* + W_0$ be an arbitrary subgradient of the nuclear norm at L_0, where $\|W_0\| \leq 1$ and $P_T W_0 = 0$
- $\text{sgn}(S_0) + F_0$ be an arbitrary subgradient of the ℓ_1-norm at S_0, where $\|F_0\|_\infty \leq 1$ and $P_\Omega F_0 = 0$

Then we can lower bound the value of the objective

$$\|L_0 + H\|_* + \lambda\|S_0 - H\|_1 \geq \|L_0\|_* + \lambda\|S_0\|_1 + \langle UV^* + W_0, H\rangle - \lambda\langle \text{sgn}(S_0) + F_0, H\rangle$$

Now we pick a particular pair (W_0, F_0) such that

- $\langle W_0, H\rangle = \|P_{T^\perp} H\|_*$, for example $W_0 = P_{T^\perp} W$ where W is a normed matrix such that $\langle W, P_{T^\perp} H\rangle = \|P_{T^\perp} H\|_*$ (by duality of $\|.\|$ and $\|.\|_*$)
- $\langle F_0, H\rangle = -\|P_{\Omega^\perp} H\|_1$, for example $F_0 = -\text{sgn}(P_{\Omega^\perp} H)$

then we have

$$\|L_0 + H\|_* + \lambda\|S_0 - H\|_1 \geq \|L_0\|_* + \lambda\|S_0\|_1 + \|P_{T^\perp} H\|_* + \|P_{\Omega^\perp} H\|_1 + \langle UV^* - \lambda\text{sgn}(S_0), H\rangle$$

we can bound the inner product using the definition of W and F,

$$\begin{aligned} |\langle UV^* - \lambda\text{sgn}(S_0), H\rangle| &= |\langle \lambda F - W, H\rangle| && \text{since } UV^* + W = \lambda(\text{sgn}(S_0) + F) \\ &\leq |\langle W, H\rangle| + \lambda|\langle F, H\rangle| && \text{by the triangular inequality} \\ &\leq \beta(\|P_{T^\perp} H\|_* + \lambda\|P_{\Omega^\perp} H\|_1) \end{aligned}$$

where $\beta = \max(\|W\|, \|F\|_\infty) < 1$, and the last inequality follows from the fact that

$$\begin{aligned} \|P_{T^\perp} H\|_* &\geq \langle P_{T^\perp} H, W/\|W\|\rangle \geq \langle H, W/\|W\|\rangle \\ \|P_{\Omega^\perp} H\|_1 &\geq \langle P_{\Omega^\perp} H, F/\|F\|_\infty\rangle \geq \langle H, F/\|F\|_\infty\rangle \end{aligned}$$

Thus

$$\begin{aligned} \|L_0 + H\|_* + \lambda\|S_0 - H\|_1 - \|L_0\|_* - \lambda\|S_0\|_1 &\geq (1 - \beta)\left(\|P_{T^\perp} H\|_* + \lambda\|P_{\Omega^\perp} H\|_1\right) \\ &> 0 \end{aligned}$$

since $\|P_{T^\perp} H\|_* = \|P_{\Omega^\perp} H\|_1 = 0$ only if $P_{T^\perp} H = P_{\Omega^\perp} H = 0$, i.e. $H \in \Omega \cap T$, and, by assumption, $\Omega \cap T = 0$ and $H \neq 0$. Therefore the objective strictly increases with a non-zero perturbation. This completes the proof. □

The proof of the main theorem will use a slightly different result, given by the following Lemma:

Lemma 2. *Assume that* $\|P_\Omega P_T\| \leq 1/2$. *Then* (L_0, S_0) *is the unique solution to PCP if* $\exists(W, F)$ *such that*

$$\begin{aligned} &UV^* + W = \lambda(sign(S_0) + F + P_\Omega D) \\ &P_T W = 0 \\ &\|W\| \leq 1/2 \\ &P_\Omega F = 0 \\ &\|F\|_\infty \leq 1/2 \\ &\|P_\Omega D\|_F \leq 1/4 \end{aligned} \tag{1.14}$$

Proof. Using $\beta = \max(\|W\|, \|F\|_\infty) \leq \frac{1}{2}$ in the previous proof, we have for a non-zero perturbation H

$$\begin{aligned}\|L_0 + H\|_* + \lambda\|S_0 - H\|_1 - \|L_0\|_* - \lambda\|S_0\|_1 &\geq \frac{1}{2}\left(\|P_{T^\perp}H\|_* + \lambda\|P_{\Omega^\perp}H\|_1\right) - \lambda\langle P_\Omega D, H\rangle \\ &\geq \frac{1}{2}\left(\|P_{T^\perp}H\|_* + \lambda\|P_{\Omega^\perp}H\|_1\right) - \frac{\lambda}{4}\|P_\Omega H\|_F\end{aligned}$$

the last term can be further bounded

$$\begin{aligned}\|P_\Omega H\|_F &\leq \|P_\Omega P_T H\|_F + \|P_\Omega P_{T^\perp} H\|_F \\ &\leq \frac{1}{2}\|H\|_F + \|P_{T^\perp}H\|_F \qquad \text{using } \|P_\Omega P_T\| \leq \frac{1}{2} \text{ and } \|P_\Omega\| \leq 1 \\ &\leq \frac{1}{2}\|P_\Omega H\|_F + \frac{1}{2}\|P_{\Omega^\perp}H\|_F + \|P_{T^\perp}H\|_F\end{aligned}$$

therefore

$$\|P_\Omega H\|_F \leq \|P_{\Omega^\perp}H\|_F + 2\|P_{T^\perp}H\|_F$$

and we conclude by lower bounding the increase in the objective

$$\begin{aligned}\|L_0 + H\|_* + \lambda\|S_0 - H\|_1 - \|L_0\|_* - \lambda\|S_0\|_1 &\geq \frac{1}{2}\left((1-\lambda)\|P_{T^\perp}H\|_* + \frac{\lambda}{2}\|P_{\Omega^\perp}H\|_1\right) \\ &> 0\end{aligned}$$

since $\|P_{T^\perp}H\|_* = \|P_{\Omega^\perp}H\|_1 = 0$ only if $P_{\Omega^\perp}H = P_{T^\perp}H = 0$, i.e. $H \in \Omega \cap T$, and, by assumption, $\Omega \cap T = \{0\}$ ($\|P_\Omega P_T\| \leq \frac{1}{2} < 1$). This completes the proof. □

Bounding $\|P_\Omega P_T\|$

Under suitable conditions on the size of the support Ω_0 of the sparse component, a bound can be derived on $\|P_\Omega P_T\|$ [9].

Theorem 4. *Suppose Ω_0 is sampled from the Bernoulli model with parameter ρ_0. Then with high probability,*

$$\|P_T - \rho_0^{-1} P_T P_{\Omega_0} P_T\| \leq \epsilon$$

provided that $\rho_0 \geq C_0 \epsilon^{-2} \frac{\mu r \log n}{n}$ where μ is the incoherence parameter and C_0 is a numerical constant.

As a consequence, $\|P_\Omega P_T\|$ can be bounded, and if $|\Omega|$ is not too large, then the desired bound $\|P_\Omega P_T\| \leq 1/2$ holds.

1.3.5 Probabilistic Guarantee via Dual Certification

We now present the proof that, under the assumptions of Robust PCA, then with high probability, we can find a dual certificate (W, F, D) that satisfies the conditions (1.14) of Lemma 2. This will achieve the proof of the probabilistic guarantee of recovery of (L_0, S_0).

In order to construct a dual certificate (W, F, D), we define $W = W^L + W^S$ where W^L and W^S are constructed to satisfy the following properties. First, $P_\Omega(W^S) = \lambda \operatorname{sign}(S_0)$. Then it also need to satisfies the following. With $\rho = \frac{m}{n^2}$,

Lemma 3. *Let $S_0 \sim Bern(\rho)$ iid for each entry with Ω as its support set. Set $j_0 = 2 \log n$. With the assumptions in the main theorem of RPCA, W^L satisfies the following with high probability.*

1. $||W^L|| < \frac{1}{4}$
2. $||P_\Omega(UV^* + W^L)||_F < \frac{\lambda}{4}$
3. $||P_{\Omega^\perp}(UV^* + W^L)||_\infty < \frac{\lambda}{4}$

Lemma 4. *Let $S_0 \sim Bern(\rho)$ iid for each entry with Ω as its support set. With the assumptions in the main theorem of RPCA, W^S satisfies the following with high probability.*

1. $||W^S|| < \frac{1}{4}$
2. $||P_{\Omega^\perp}(W^S)||_\infty < \frac{\lambda}{4}$

With these two lemmas, we are ready to show that they help to justify the probabilistic guarantee. We note that

$$\begin{aligned} UV^* + W &= W^L + W^S + UV^* \\ &= \lambda \left[P_\Omega \left(\frac{UV^* + W^L}{\lambda} \right) + sign(S_0) + P_{\Omega^\perp} \left(\frac{W^L + W^S + UV^*}{\lambda} \right) \right] \end{aligned}$$

Thus we take

$$\begin{aligned} D &= \frac{UV^* + W^L}{\lambda} \\ W &= W^L + W^S \\ F &= P_\Omega \left(\frac{W^L + W^S + UV^*}{\lambda} \right) \end{aligned}$$

From Lemma (3) and Lemma (4), we can check that (W, F, D) satisfies the conditions of Lemma (2), thus establishing the probabilistic guarantee.

The following section will discuss in depth how to construct W^L and W^S that satisfy conditions of Lemma (3) and Lemma (4).

1.3.6 Proof of the Lemma about golfing scheme and dual certificate

Golfing scheme:

The golfing scheme involves creating a W^L according to the following method.

1. Fix $j_0 \geq 1$, define $\Omega_j \sim Bern(q)$ iid with $1 \leq j \leq j_0$ and $\rho = (1-q)^{j_0}$. Define the complement of support of Ω by $\Omega = \cup_{1 \leq j \leq j_0} \Omega_j^C$.
2. Define a sequence of matrix which finally ends at W^L
   (a) $Y_0 = 0$
   (b) $Y_j = Y_{j-1} + \frac{1}{q} P_{\Omega_j} P_T (UV^* - Y_{j-1})$ for $1 \leq j \leq j_0$
   (c) $W^L = P_{T^\perp}(Y_{j_0})$

We first list a number of facts that will be used in the proof of Lemma (3).

Fact 1 ([10]). *If we fix $Z \in T$, $\Omega_0 \sim Bern(\rho_0)$, and $\rho_0 \geq C_0 \epsilon^{-2} \frac{\mu r \log n}{n}$, then with high probability, we will have,*

$$\|Z - \rho_0^{-1} P_T P_{\Omega_0}(Z)\|_\infty \leq \epsilon \|Z\|_\infty$$

Fact 2 ([10]). *If we fix Z, $\Omega_0 \sim Bern(\rho_0)$, and $\rho_0 \geq C_0 \frac{\mu \log n}{n}$, then with high probability, we will have,*

$$\|(I - \rho_0^{-1} P_{\Omega_0}) Z\| \quad \leq \quad C_0' \sqrt{\frac{n \log n}{\rho_0}} \|Z\|_\infty$$

Fact 3 ([10]). *If $\Omega_0 \sim Bern(\rho_0)$,$\rho_0 \geq C_0 \epsilon^{-2} \frac{\mu r \log n}{n}$, then with high probability, we will have,*

$$\|P_T - \rho_0^{-1} P_T P_{\Omega_0} P_T\| \leq \epsilon$$

Fact 4 ([10]). *If $\Omega \sim Bern(\rho)$ and $1 - \rho \geq C_0 \epsilon^{-2} \frac{\mu r \log n}{n}$, then with high probability $\|P_\Omega P_T\|^2 \leq \rho + \epsilon$*

Now we present the proof of Lemma (3)

Proof. We define another sequence of matrix $Z_j = UV^* - P_T(Y_j)$. There are some properties about Z_j which allows us to establish the proof. We survey them here and provides the proof of them.

i) Note that

$$Z_j = \left(P_T - \frac{1}{q} P_T P_{\Omega_j} P_T \right) Z_{j-1}. \tag{1.15}$$

The reason is as follows.

$$\begin{aligned}
Z_j &= UV^* - P_T\left(Y_{j-1} + \frac{1}{q}P_{\Omega_j}P_T(UV^* - Y_{j-1})\right) && \text{by construction of } Y_j \\
&= UV^* - P_T(Y_{j-1}) - \frac{1}{q}P_TP_{\Omega_j}P_T(UV^* - Y_{j-1}) && \text{by of linearity of } P_T \\
&= Z_{j-1} - q^{-1}(P_TP_{\Omega_j}(UV^* - P_T(Y_{j-1}))) && \text{since } P_T(UV^*) = UV^* \\
&= P_T(Z_{j-1}) - q^{-1}(P_TP_{\Omega_j}P_TZ_{j-1}) && \text{since } Z_{j-1} \in T \\
&= (P_T - q^{-1}P_TP_{\Omega_j}P_T)Z_{j-1} && \text{by linearity}
\end{aligned}$$

ii) If $q \geq C_0\epsilon^{-2}\frac{\mu r \log n}{n}$, then with high probability,

$$\|Z_j\|_\infty \leq \epsilon^j \|UV^*\|_\infty \tag{1.16}$$

The reason is as follows. By Fact (1), we have,

$$\begin{aligned}
\|Z_{j-1} - q^{-1}P_TP_{\Omega_j}Z_{j-1}\|_\infty &\leq \epsilon\|Z_{j-1}\|_\infty \\
\|Z_j\|_\infty &\leq \epsilon\|Z_{j-1}\|_\infty \text{by (1.15)}
\end{aligned}$$

Inductively, we get the desired. iii) If $q \geq C_0\epsilon^{-2}\frac{\mu r \log n}{n}$, then

$$\|Z_j\|_F \leq \epsilon^j\sqrt{r} \tag{1.17}$$

The reason is as follows. By Fact (3), we have,

$$\begin{aligned}
&\left\|(P_T - q^{-1}P_TP_{\Omega_0}P_T)\left(\frac{Z_{j-1}}{\|Z_{j-1}\|_F}\right)\right\|_F \leq \epsilon \\
&\|(P_T - q^{-1}P_TP_{\Omega_0}P_T)Z_{j-1}\|_F \leq \epsilon\|Z_{j-1}\|_F && \text{by rearranging terms} \\
&\|Z_j\|_F \leq \epsilon\|Z_{j-1}\|_F && \text{by (1.15)}
\end{aligned}$$

Inductively, we get the desired result.

After establishing these properties, we are ready to prove that golfing scheme yields W^L that satisfies the desired properties.

1) Proof of condition (1):

$$\begin{aligned}
\|W^L\| &= \|P_{T^\perp}(Y_{j_0})\| && \text{by definition}\\
&\leq \sum_{j=1}^{j_0} \frac{1}{q}\|P_{T^\perp}P_{\Omega_j}Z_{j-1}\| && \text{since } Y_j = Y_{j-1} + q^{-1}P_{\Omega_j}(Z_{j-1})\\
&= \sum_{j=1}^{j_0} \|P_{T^\perp}(\frac{1}{q}P_{\Omega_j}Z_{j-1} - Z_{j-1})\| && \text{since } Z_j \in T\\
&\leq \sum_{j=1}^{j_0} \|(\frac{1}{q}P_{\Omega_j}Z_{j-1} - Z_{j-1})\| && \text{since } \|P_{T^\perp}(M)\| \leq \|M\|\\
&\leq C_0'\sqrt{\frac{n\log n}{q}}\sum_{j=1}^{j_0}\|Z_{j-1}\|_\infty && \text{by Fact(2)}\\
&\leq C_0'\sqrt{\frac{n\log n}{q}}\sum_{j=1}^{j_0}\epsilon^j\|UV^*\|_\infty && \text{by (1.16)}\\
&\leq C_0'\sqrt{\frac{n\log n}{q}}\frac{1}{1-\epsilon}\|UV^*\|_\infty && \text{by bound on geometric series}\\
&\leq C_0'\sqrt{\frac{n\log n}{q}}\frac{1}{1-\epsilon}\frac{\sqrt{\mu r}}{n} && \text{by RPCA assumptions}\\
&\leq C''\epsilon < \frac{1}{4} && \text{for some constant } C''
\end{aligned}$$

2) Proof of condition (2) : First, we expand,

$$\|P_\Omega(UV^* + W^L)\|_F \quad = \quad \|P_\Omega(UV^* + P_{T^\perp}Y_{j_0})\|_F$$

Then, because $P_\Omega(Y_{j_0}) = P_\Omega(\sum_j P_{\Omega_j}Z_{j-1}) = 0$ and $P_\Omega(P_T(Y_{j_0}) + P_{T^\perp}(Y_{j_0})) = 0$, we have,

$$\|P_\Omega(UV^* + W^L)\|_F \quad = \quad \|P_\Omega(UV^* - P_TY_{j_0})\|_F$$

Continuing,

$$\begin{aligned}
\|P_\Omega(UV^* + W^L)\|_F &= \|P_\Omega(Z_{j_0})\|_F && \text{by definition}\\
&\leq \|Z_{j_0}\|_F && \text{because of summing over larger set}\\
&\leq \epsilon^{j_0}\sqrt{r} && \text{by (1.17)}\\
&\leq \sqrt{r}\frac{1}{n^2} \leq \frac{\lambda}{4} && \text{by the choice of } \lambda \text{ and } \epsilon
\end{aligned}$$

3) Proof of condition (3) :

$$\begin{aligned}
\|P_{\Omega^\perp}(UV^* + W^L)\|_\infty &= \|P_{\Omega^\perp}(Z_{j_0} + Y_{j_0})\|_\infty && \text{by definition}\\
&\leq \|Z_{j_0}\|_\infty + \|Y_{j_0}\|_\infty && \text{by triangle inequality and summing over larger set}\\
&\leq \|Z_{j_0}\|_F + \|Y_{j_0}\|_\infty && \text{by the properties of Frobenius and infinite norms}\\
&\leq \frac{\lambda}{8} + \|Y_{j_0}\|_\infty && \text{similar argument as in Proof of condition (2)}
\end{aligned}$$

Moreover, we have

$$\begin{aligned}
\|Y_{j_0}\|_\infty &\le q^{-1}\sum_j \|P_{\Omega_j} Z_{j-1}\|_\infty && \text{by triangle inequality}\\
&\le q^{-1}\sum_j \|Z_{j-1}\|_\infty && \text{by summing over a larger set}\\
&\le q^{-1}\sum_j \epsilon^j \frac{\sqrt{\mu r}}{n} && \text{by (1.16)}\\
&\le \frac{\lambda}{8} && \text{if } \epsilon \text{ is sufficiently small}
\end{aligned}$$

□

1.3.7 Proof of the Lemma about least square construction and dual certificate

Construction of W^S:

$$W^S = \lambda P_{T^\perp}((P_\Omega - P_\Omega P_T P_\Omega)^{-1} sign(S_0))$$

Now we present the proof of Lemma (4).

Proof. We consider the sign of S_0 to be distrbuted as follows

$$sign(S_0)_{i,j} = \begin{cases} 1 & \text{wp } \frac{\rho}{2} \\ 0 & \text{wp } 1-\rho \\ -1 & \text{wp } \frac{\rho}{2} \end{cases}$$

1) Proof of condition (1) :

I) We note the we can separate W^Sinto two parts and then bound them separately.

$$W^S = \lambda P_{T^\perp}(sign(S_0)) + \lambda P_{T^\perp}(\sum_{k\ge 1}(P_\Omega P_T P_\Omega)^k (sign(S_0)))$$

II) Then, we have

$$\begin{aligned}
\lambda\|P_{T^\perp}(sign(S_0))\| &\le \lambda\|\operatorname{sign}(S_0)\| && \text{by (1.10)}\\
&= \frac{1}{\sqrt{n}}\|sign(S_0)\| && \text{by the choice of } \lambda\\
&\le 4\sqrt{\rho} && \text{with high probability}
\end{aligned}$$

where the last inequality uses the fact that for the entry-wise distribution of $sign(S_0)$, we can have $\|sign(S_0)\| \le 4\sqrt{n\rho}$ holds with high probability.

III) Now, for the other part, $\lambda P_{T^\perp}(\sum_{k\geq 1}(P_\Omega P_T P_\Omega)^k(sign(S_0)))$, we bound it by first expressing it in the form of $< X, sign(S_0)\rangle$ and then claim that with high probability, this term is bounded above as desired. Let $R = \sum_{k\geq 1}(P_\Omega P_T P_\Omega)^k$, then we have,

$$\begin{aligned}\|P_{T^\perp}(R(sign(S_0)))\| &\leq \|R(sign(S_0))\| \\ &\leq 4 \sup_{x,y\in N} < y, R(sign(S_0)x)\rangle\end{aligned}$$

where the last inequality uses the fact that there exists a $\frac{1}{2} - net$ of the Eucledean ball and it has at most 6^n elements. Continuing, we have

$$\begin{aligned}\|P_{T^\perp}(R(sign(S_0)))\| &\leq 4 \sup_{x,y\in N} \langle y, R(sign(S_0)x)\rangle \\ &= 4 \sup_{x,y\in N} \langle yx^*, R(sign(S_0))\rangle \\ &= 4 \sup_{x,y\in N} \langle R(yx^*), sign(S_0)\rangle \qquad (1.18)\end{aligned}$$

and that we denote $X(x,y) = \langle R(yx^*), sign(S_0)\rangle$ afterwards.

Note that, by Hoeffding's inequality, we have,

$$Pr(|X(x,y)| > t \mid \Omega) \leq 2exp(-\frac{t^2}{2\|R(xy^*)\|_F^2})$$

This gives,

$$\begin{aligned}Pr(\|P_{T^\perp}(R(sign(S_0)))\| > 4t \mid \Omega) &\leq Pr(\|R(sign(S_0))\| > 4t \mid \Omega) \\ &\leq Pr(\sup_{x,y} |X(x,y)| > t \mid \Omega) \qquad \text{by (1.18)} \\ &\leq 2N^2 \exp\left(-\frac{t^2}{2\|R\|_F^2}\right) \qquad \text{since } \|yx^*\|_F \leq 1\end{aligned}$$

Now, we proceed to bound the probability without the condition on Ω.

First, note that the event of $\|P_\Omega P_T\| \leq \sigma = \rho + \epsilon$, implies that $\|R\| \leq (\frac{\sigma^2}{1-\sigma^2})^2$. Thus, unconditionally, we have

$$\begin{aligned}Pr(\|R(sign(S_0))\| > 4t) &\leq 2|N|^2 \exp\left(\frac{-t^2}{2(\frac{\sigma^2}{1-\sigma^2})^2}\right) + Pr(\|P_\Omega P_T\| > \sigma) \\ &\leq 2 \cdot 6^{2n} \exp\left(\frac{-t^2}{2(\frac{\sigma^2}{1-\sigma^2})^2}\right) + Pr(\|P_\Omega P_T\| > \sigma)\end{aligned}$$

Thus, where we finally put $t = \frac{1}{16}$

$$Pr(\lambda\|R(sign(S_0))\| > 4t) \leq 2 \cdot 6^{2n} \exp\left(\frac{-\frac{t^2}{\lambda^2}}{2(\frac{\sigma^2}{1-\sigma^2})^2}\right) + Pr(\|P_\Omega P_T\| > \sigma)$$

With $\lambda = \sqrt{\frac{1}{n}}$,we have this probability$\to 0$ as $n \to \infty$. Thus with high probability $\|W^S\| \leq \frac{1}{4}$

2) Proof of condition (2) :

The idea is that we first express $P_{\Omega^\perp}(W^S)$ in the form of $< X, sign(S_0) >$and we can derive upper bound on it if highly probably event of $\{\|P_\Omega P_T\| \leq \sigma\}$ for some small $\sigma = \rho + \epsilon$ holds .

I) First,

$$P_{\Omega^\perp}(W^S) = P_{\Omega^\perp}\left(\lambda(I - P_T)\left(\sum_{k\geq 0}(P_\Omega P_T P_\Omega)^k\right)\operatorname{sign}(S_0)\right) \quad \text{since } P_{T^\perp} = I - P_T \tag{1.19}$$

$$= -\lambda P_{\Omega^\perp} P_T (P_\Omega - P_\Omega P_T P_\Omega)^{-1}\operatorname{sign}(S_0) \quad \text{by summing over terms and canceling} \tag{1.20}$$

For $(i,j) \in \Omega^C$, we have

$$\begin{aligned}
e_i^* W^S e_j &= \langle e_i e_j^*, W_S\rangle && \text{by property of trace}\\
&= \langle e_i e_j^*, -\lambda P_{\Omega^\perp} P_T (P_\Omega - P_\Omega P_T P_\Omega)^{-1}\operatorname{sign}(S_0)\rangle && \text{by (1.20)}\\
&= -\lambda\langle e_i e_j^*, P_T (P_\Omega - P_\Omega P_T P_\Omega)^{-1}\operatorname{sign}(S_0)\rangle && \text{by rearranging terms}\\
&= -\lambda\langle e_i e_j^*, P_T P_\Omega (P_\Omega - P_\Omega P_T P_\Omega)^{-1}\operatorname{sign}(S_0)\rangle && \text{by the property of the inverse}\\
&= -\lambda\langle e_i e_j^*, P_T \sum_{k\geq 0}(P_\Omega P_T P_\Omega)^k \operatorname{sign}(S_0)\rangle && \text{by infinite sum representation}
\end{aligned}$$

Noting that P_Ω, P_T are self-adjoint, thus, we have

$$e_i^* W^S e_j \;=\; \lambda\langle -(P_\Omega - P_\Omega P_T P_\Omega)^{-1} P_\Omega P_T (e_i e_j^*), \operatorname{sign}(S_0)\rangle \tag{1.21}$$

where we now denote $X(i,j) = -(P_\Omega - P_\Omega P_T P_\Omega)^{-1} P_\Omega P_T (e_i e_j^*)$

II)We now consider, where we put $t = \frac{1}{4}$,

$$\begin{aligned}
Pr(\|P_{\Omega^\perp}(W^S)\|_\infty > t\lambda \mid \Omega) &\leq \sum_{(i,j)\in\Omega^C} Pr(|e_i^* W^S e_j| > t\lambda|\Omega) && \text{by union bound}\\
&\leq n^2 Pr(|e_i^* W^S e_j| > t\lambda|\Omega) \text{ for some (i,j)} && \text{by taking the maximum}\\
&= n^2 Pr(|\langle X(i,j), \operatorname{sign}(S_0)\rangle| > t|\Omega) && \text{by (1.21)}\\
&\leq 2n^2 \exp\left(-\frac{2t^2}{4\|X(i,j)\|_F}\right) && \text{by Hoeffding's inequality}
\end{aligned}$$

III) We then proceed to bound the $\|X(i,j)\|$. On the event of $\{\|P_\Omega P_T\| \leq \sigma\}$, we have

$$\begin{aligned}
\|P_\Omega P_T (e_i e_j^*)\|_F &\leq \|P_\Omega P_T\| \cdot \|P_T(e_i e_j^*)\|_F && \text{by property of spectral norm}\\
&\leq \sigma\sqrt{\frac{2\mu r}{n}} && \text{by (1.12) and the bound on } \|P_\Omega P_T\|
\end{aligned}$$

Moreover, we have

$$\begin{aligned}\|(P_\Omega - P_\Omega P_T P_\Omega)^{-1}\| &\leq \sum_{k\geq 0} \|(P_\Omega P_T P_\Omega)^k\| \quad \text{by triangle inequality} \\ &\leq \frac{1}{1-\sigma} \quad \text{by the bound on } \|P_\Omega P_T\|\end{aligned}$$

Finally, we have

$$\|X(i,j)\|_F \leq 2\sigma^2 \frac{\frac{\mu r}{n}}{(1-\sigma)^2}$$

Combining, we have

$$\begin{aligned}Pr(\|P_{\Omega^\perp} W^S\| > t\lambda) &\leq 2n^2 \exp\left(\frac{-t^2 n(1-\sigma)^2}{4\sigma^2(\mu r)}\right) + Pr(\|P_\Omega P_T\| \geq \sigma) \\ &\leq \epsilon \text{ if } \mu r < \rho_r^{'} \frac{n}{\log n}\end{aligned}$$

□

1.3.8 Proof of the equivalence of the Bernoulli sampling and uniform sampling model

To complete the story about the equivalence of sampling model, we present theorem.
Theorem 5. *Let E be the event that the recovery of (L_0, S_0) is exact through the RPCA. Then, $\forall \epsilon > 0$,*

- With $\rho = \frac{m}{n^2} + \epsilon$, E holds with high probability when the sparse matrix $S_{i,j} \sim Bern(\rho)$ iid $\Longrightarrow E$ holds with high probability when the sparse matrix $S \sim Uniform(m)$.
- With $\rho = \frac{m}{n^2} - \epsilon$, E holds with high probability when the sparse matrix $S \sim Uniform(m)$ $\Longrightarrow$ E holds with high probability when the sparse matrix $S_{i,j} \sim Bern(\rho)$ iid

Proof. Let us use the notation of subscrpt to denote the underlying sampling process, e.g. $,P_{B(\rho)}(E)$ and $P_{U(m)}(E)$ be the probability of success recovery using Bernoulli sampling and uniform sampling respectively. We then upper and lower bound the difference of $P_{B(\rho)}(E) - P_{U(m)}(E)$ and show that the difference goes to zero as the dimension of the matrix $n \to \infty$.

$$
\begin{aligned}
P_{B(\rho)}(E) &= \sum_{i=0}^{n^2} P_{B(\rho)}(|\Omega| = i) P_{B(\rho)}(E \mid |\Omega| = i) \\
&= \sum_{i=0}^{n^2} P_{B(\rho)}(|\Omega| = i) P_{U(i)}(E) \\
&\leq \sum_{i=0}^{m-1} P_{B(\rho)}(|\Omega| = i) + \sum_{i=m}^{n^2} P_{U(i)}(E) P_{B(\rho)}(|\Omega| = i) \\
&\leq \sum_{i=0}^{m-1} P_{B(\rho)}(|\Omega| = i) + \sum_{i=m}^{n^2} P_{U(i)}(E) P_{B(\rho)}(|\Omega| = i) \\
&\leq \sum_{i=0}^{m-1} P_{B(\rho)}(|\Omega| = i) + \sum_{i=m}^{n^2} P_{U(m)}(E) P_{B(\rho)}(|\Omega| = i) \\
&\leq P_{B(\rho)}(|\Omega| < m) + P_{U(m)}(E)
\end{aligned}
$$

This gives, $P_{B(\rho)}(E) - P_{U(m)}(E) \leq P_{B(\rho)}(|\Omega| < m)$. With $\rho = \frac{m}{n^2} + \epsilon$, by law of large number, when $n \to \infty$ we get, $P_{B(\rho)}(E) \leq P_{U(m)}(E)$.

On the other hand,

$$
\begin{aligned}
P_{B(\rho)}(E) &\geq \sum_{i=0}^{m} P_{B(\rho)}(|\Omega| = i) P_{B(\rho)}(E \mid |\Omega| = i) \\
&\geq P_{U(m)}(E) \sum_{i=0}^{m} P_{B(\rho)}(|\Omega| = i) \\
&= P_{U(m)}(E)(1 - P_{B(\rho)}(|\Omega| > m)) \\
&\geq P_{U(m)}(E) - P_{B(\rho)}(|\Omega| > m)
\end{aligned}
$$

This gives, $P_{B(\rho)}(E) - P_{U(m)} \geq -P_{B(\rho)}(|\Omega| > m)$. With $\rho = \frac{m}{n^2} - \epsilon$, by law of large number, when $n \to \infty$ we get, $P_{B(\rho)}(E) \geq P_{U(m)}(E)$. □

1.3.9 Proof of the form of sub-differential of nuclear norm

To complete the story on the structure of subdifferential of nuclear norm, we present the following justifications.

Definition 1. For matrix norms $\|\cdot\|$ which satisfy $\|UAV\| = \|A\|$ $\forall U, V$ being orthonormal, then they are called orthogonally invariant norm.

Definition 2. For orthogonally invariant norm $\|\cdot\|$ which is defined by its singular values $\|A\| = \phi(\vec{\sigma})$ where $\vec{\sigma}$ are the singular values of A, we call the function ϕ as a symmetric gauge function if it is a norm and it satisfies $\phi(\vec{\sigma}) = \phi(\epsilon_1 \sigma_{i_1}, ..., \epsilon_n \sigma_{i_n})$ for any permulation of $(i_1, ..., i_n)$ of $(1, ..., n)$ and $\epsilon_i = \pm 1$.

Fact 5. *For orthogonally invariant norm $\|\cdot\|$ with symmetric gauge function ϕ, the sub-differential is given by*

$$\partial\|A\| = \{U diag(\vec{d}) V \mid A = U\Sigma V^T, \vec{d} \in \partial\phi(\vec{d}), U \in R^m, V \in R^n\}$$

Theorem 6. *Let $A = U^{(1)}\Sigma {V^{(1)}}^T$. Then*

$$\partial\|A\|_* = \{U^{(1)}{V^{(1)}}^T + W : \|W\| \leq 1, {U^{(1)}}^T W = 0, W V^{(1)} = 0\}$$

Proof. We take the symmetric gauge function as $\|\cdot\|_1$ and then apply the Fact (5) and will obtain the desired result. □

1.4 Related Problems and Extensions

In this section we discuss a number of problems and extensions related to the Robust PCA framework. We will be much briefer in our discussion than for the classic case and mainly provide an overview of where the field going and what the most important developments are.

1.4.1 Exact Matrix completion

Robust PCA is an extension of the exact matrix completion problem that was introduced in [9], in which one seeks to recover a low-rank matrix L_0 from a small fraction of its entries. More precisely, assume one is given $\{(L_0)_{ij}, (i,j) \in \Omega\}$ where Ω is a subset of $[n] \times [n]$. The observed matrix in this case is

$$M = P_\Omega L_0$$

where P_Ω denotes the sampling operator, i.e. the orthogonal projection on the subspace of matrices supported on Ω. One seeks to solve the problem

$$\begin{array}{ll} \text{minimize} & \text{rank}(L) \\ \text{subject to} & P_\Omega L = P_\Omega L_0 \end{array} \tag{1.22}$$

A popular heuristic is to minimize the nuclear norm of L, $\|L\|_* = \|\sigma(L)\|_1$ which encourages sparsity of the vector of singular components of L, and can thus be interpreted as an approximation of the rank operator, similarly to the ℓ_1-norm that can be considered an approximation of the ℓ_0 count operator. The approximate problem then reads

$$\begin{array}{ll} \text{minimize} & \|L\|_* \\ \text{subject to} & P_\Omega L = P_\Omega L_0 \end{array} \tag{1.23}$$

Incoherence

In order to guarantee recovery with high probability, an incoherence condition is introduced. This condition is similar to the one that appears in the Robust PCA framework, though slightly different.

First consider an orthogonal matrix $U = [u_1, \ldots, u_n]$, and define its coherence of $\mu(U)$ with respect to the canonical (euclidean) basis to be

$$\mu(U) = \frac{n}{r} \max_i \|P_U e_i\|_2^2 = \frac{n}{r} \max_i \left[\sum_{k=1}^{r} u_{ki}^2 \right] \tag{1.24}$$

The coherence $\mu(U)$ is a measure of spread of the vectors $u_1, \ldots, u_n$ with respect to the canonical basis. One seeks matrices with low coherence, since intuitively those matrices will have low probability to be in the null space of the sampling operator P_Ω.

Main result

Theorem 6. *Let the (slim) SVD of the original matrix L_0 be given by $L_0 = U\Sigma V^T$, and assume that the following conditions hold:*

- $\max\{\mu(U), \mu(V)\} \leq \mu_0$
- $\left(\sum_k u_k v_k^T\right)_{ij} \leq \mu_1 \sqrt{\frac{r}{n_1 n_2}}$ *(true for $\mu_1 = \mu_0 \sqrt{r}$)*
- $m \geq c \max\left\{\mu_1^2, \sqrt{\mu_0}\mu_1, \mu_0 n^{1/4}\right\} nr\beta \log n$

Then recovery is exact with high probability (at least $1 - \frac{c}{n^\beta}$).

The authors of [9] also give a list of models that can be used to generate incoherent matrices. Let the SVD of L_0 be given by $L_0 = \sum_{k=1}^{r} \sigma_k u_k v_k^*$. Then L_0 is incoherent with high probability if it is sampled from:

- The incoherent basis model: U and V satisfy the size property

  $$\|U\|_\infty \leq \sqrt{\mu_B/n} \quad \|V\|_\infty \leq \sqrt{\mu_B/n}$$

  for some numerical constant μ_B. Observe that under these conditions, one can bound the coherence $\max(\mu(U), \mu(V)) \leq \mu_B$, and it can be shown that the second condition of Theorem 6 holds for $\mu_1 = O(\sqrt{\log n})$.
- The random orthogonal model: if , then $\{u_1, \ldots, u_r\}$ and $\{v_1, \ldots, v_r\}$ are assumed to be selected at random.

Relation to Robust PCA

Robust PCA can be thought of as an extension of the matrix completion problem, where instead of being given a known subset of the entries $\{(L_0)_{ij}, (i,j) \in \Omega\}$ with the rest of the entries missing, we have exact information about an unknown subset of the entries and the rest of the entries is corrupted. In this sense, Robust PCA is a harder problem than matrix completion.

Note that the matrix L_0 can be recovered by Principal Component Pursuit, solving a different problem:

$$\begin{aligned} &\text{minimize} && \|L\|_* + \lambda \|S\|_1 \\ &\text{subject to} && P_\Omega(L + S) = M \end{aligned} \tag{1.25}$$

where now the observed matrix M is assumed to be given by

$$M = P_\Omega(L_0 + S_0) = P_\Omega(L_0) + S_0'$$

Here the original data matrix L_0 is assumed to be corrupted with the noise matrix S_0 in addition to being under-sampled. The exact matrix completion problem however, assumes that the observed data is perfect $S_0 = 0$. Under the assumptions of Theorem 1, recovery is exact with high probability, in particular for $S_0 = 0$ (support of the sparse matrix has cardinality 0).

1.4.2 Stable Principal Component Pursuit

One issue with Robust PCA that limits its practical applicability is the assumption that, while part of the data may be arbitrarily corrupted, the rest of the data is exact. In most applications, however, there will also be some small but non-sparse noise component present, caused for example by basic measurement inaccuracies, quantization or compression effects and so on. In [34] the authors therefore study the problem of recovering a low-rank matrix (the principal components) from a high-dimensional data matrix despite both small entry-wise noise and gross sparse errors. It proves that the solution to a convex program (a relaxation of classic Robust PCA) gives an estimate of the low-rank matrix that is simultaneously stable to small entry- wise noise and robust to gross sparse errors. The result shows that the proposed convex program recovers the low-rank matrix even though a positive fraction of its entries are arbitrarily corrupted, with an error bound proportional to the noise level.

Main result

The paper [34] consider a matrix $M \in \mathbb{R}^{n_1 \times n_2}$ of the from $M = L_0 + S_0 + Z_0$, where L_0 is (non-sparse) low rank, S_0 is sparse (modeling gross errors) and Z_0 is "small" (modeling a small noisy perturbation). The assumption on Z_0 is simply that $\|Z_0\|_F \leq \delta$ for some small known δ. Hence at least for the theory part of the paper the authors do not assume anything about the distribution of the noise other than it is bounded (however they will gloss over this in their algorithm).

The convex program to be solved is a slight modification of the standard Robust PCA problem and given by

$$\begin{aligned} &\min_{L,S} \|L\|_* + \lambda\|S\|_1 \\ &\text{s.t.} \quad \|M - L - S\|_F \leq \delta \end{aligned} \tag{1.26}$$

where $\lambda = 1/\sqrt{n_1}$. Under a standard incoherence assumption on L_0 (which essentially means that L_0 should not be sparse) and a uniformity assumption on the sparsity pattern of S_0 (which means that the support of S_0 should not be too concentrated) the main result states that, with high probability in the support of S_0, for any Z_0 with $\|Z_0\|_F \leq \delta$, the solution $(\hat{L}, \hat{S})$ to (1.26) satisfies

$$\|\hat{L} - L_0\|_F^2 + \|\hat{S} - S_0\|_F^2 \leq C n_1 n_2 \delta^2$$

where C is a numerical constant. The above claim essentially states that the recovered low-rank matrix $\hat{L}$ is stable with respect to non-sparse but small noise acting on all entries of the matrix.

In order to experimentally verify the predicted performance to their formulation, the authors provide a comparison with an oracle. This oracle is assumed to provide information about the support of S_0 and the row and column spaces of L_0, which allows the computation of the MMSE estimator which otherwise would be computationally intractable (strictly speaking it of course is not really the MMSE, since it uses additional information from the oracle). Simulation results that show that the RMS error of the solution obtained through (1.26) in the non-breakdown regime (that is, for the support of S_0 sufficiently small) is only about twice as large as that of the oracle-based MMSE. This suggests that the proposed algorithm works quite well in practice. But since efficient algorithms that are practical also for large-scale problems have been proposed only recently, there does not seem to have been much work on applications yet.

Relations to existing work

The result of the paper can be seen from two different view points. On the one hand, it can be interpreted from the point of view of classic PCA. In this case, the result states that classic PCA, which can in fact be shown to be statistically optimal w.r.t. i.i.d Gaussian perturbations, can also be made robust with respect to sparse gross corruptions. On the other hand, the result can be interpreted from the point of view of Robust PCA. In this case, it essentially states that the classic Robust PCA solution can itself be made robust with respect to some small but non-sparse noise acting on all entries of the matrix.

Conceptually, the work presented in the paper is similar to the development of results for "imperfect" scenarios in compressive sensing where the measurements are noisy and the signal is not exact sparse. In this body of literature, l_1-norm minimization techniques are adapted to recover a vector $x_0 \in \mathbb{R}^n$ from contaminated observations $y = Ax_0 + z$, where $A \in \mathbb{R}^{m\times n}$ with $m \ll n$ and z is the noise term.

Algorithm

For the case of a noise matrix Z_0 whose entries are i.i.d. $\mathcal{N}(0, \sigma^2)$, the paper suggests to use an Accelerated Proximal Gradient (APG) algorithm (see section 2.2.3 for details) for solving (1.26). Note that for $\delta = 0$ the problem reduces to the standard Robust PCA problem with an equality constraint on the matrices. For this case the APG algorithm proposed in [21] solves an approximation of the form

$$\min_{L,S} \|L\|_* + \lambda\|S\|_1 + \frac{1}{2\mu}\|M - L - S\|_F$$

For the Stable PCP problem where $\delta > 0$ the authors advocate using the same algorithm with fixed but carefully chosen parameter μ (similar to [8]). In particular, they point out[1] that for $Z_0 \in \mathbf{R}^{n\times n}$ with $(Z_0)_{ij} \sim \mathcal{N}(0, \sigma^2)$ i.i.d. it holds that $n^{-1/2}\|Z_0\|_2 \to \sqrt{2}\sigma$ almost surely as $n \to \infty$. They then choose the parameter μ such that if $M = Z_0$, i.e. if $L_0 = S_0 = 0$, the minimizer of the above problem is likely to be $\hat{L} = \hat{S} = 0$. The claim is that this is the case for $\mu = \sqrt{2n}\sigma$.

It is worth noting that the assumption of a Gaussian noise matrix Z_0 is reasonable but not always satisfied. If it is not, then it is not clear if using the APG algorithm to solve the associated

[1] this based on the strong Bai Yin Theorem [2], which implies that for an $n\times n$ real matrix with entries $\xi_{ij} \sim \mathcal{N}(0, 1)$ the it holds that $\limsup_{n\to\infty} \|Z_0\|_2/\sqrt{n} = 2$ almost surely

approximate problem is a good idea and different algorithms may be needed. The problem (1.26) can be expressed as an SDP and can therefore in principle be solved using general purpose interior point solvers. However, the same scalability issues as in the standard Robust PCA problem will limit prohibit to use these methods for high-dimensional data. The paper [1] focuses on efficient first-order algorithms for solving (1.26).

Conclusion

The Stable PCP problem is one of potentially very high practical relevance. While it is reasonable to assume that in many applications the low-rank component L_0 will only be corrupted by a comparatively small number of gross errors (caused by rare and isolated events), the assumption of perfect measurements for the rest of the data outside the support of S_0 that is made in classic Robust PCA will generally not hold for example due to sensor noise. This paper asserts that if the non-sparse noise component Z_0 is sparse, then with high probability the recovered components are "close" to the actual ones.

For simplicity, the paper [34] models the non-sparse noise simply as an additive perturbation that is bounded in the Frobenius norm. In cases where one has additional information available about this noise, for example its distribution or some bounds on the absolute value of each entry, it might be possible to derive better bounds on the resulting errors. One possible extension could therefore be to look at exploiting structure in the noise.

One thing the paper claims is that "at a cost not so much higher than the classical PCA, [the] result is expected to have significant impact on many practical problems". As mentioned above, one can indeed expect that the result has a significant impact on many practical problems. However, the claim concerning the computational complexity is very optimistic. The fastest solver for the special case $\delta = 0$ (classic Robust PCA) currently seems to be a alternating directions augmented Lagrangian method (see Chapter 2). This method requires an SVD at each iteration, and for problems involving large-scale data the number of iterations can be very large. The standard PCP algorithm on the other hand is based on a single SVD, hence it can be computed much faster.

1.4.3 Robust Alignment by Sparse and Low-rank Decomposition

The convex optimization framework for low-rank matrix recovery has been employed successfully as seen in the previous discussion of the Robust PCA problem. However, in many cases in practice the data can be viewed as low-rank only after some transformation is applied. The associated recovery problem has been dubbed Robust Alignment by Sparse and Low-rank Decomposition (RASL) and was investigated in [26]. The formulation of this problem is the following:

$$\min_{L,S,\tau} \|L\|_* + \lambda \|S\|_1 \quad \text{s.t. } M \circ \tau = L + S \tag{1.27}$$

Here $L \in \mathbb{R}^{m \times n}$ is the low-rank matrix, $S \in \mathbb{R}^{m \times n}$ is a sparse corruption matrix and $M \in \mathbb{R}^{m \times n}$ contains the measurements, which are the result of the transformation τ^{-1} applied to the matrix $(L + S)$. The authors in [26] assume that the transformation is invertible. Define $M \circ \tau$ as: $M \circ \tau = [\, M_1 \circ \tau_1 \mid M_2 \circ \tau_2 \mid \ldots \mid M_n \circ \tau_n \,]$, which are the measurements $M = [\, M_1 \mid M_2 \mid \ldots \mid M_n \,]$ subject to the set of transformations $\tau = [\, \tau_1 \mid \tau_2 \mid \ldots \mid \tau_n \,] \in \mathbb{G}^n$, where $\mathbb{G}$ is a group of certain type of invertible transformations, which could be affine transform, rotation transform, etc.

The main difficulty in solving (1.27) is the nonlinearity of the equality constraint $M \circ \tau = L + S$. When the change induced by τ is small, one can approximate this constraint by linearizing about the current estimate of the transformation τ. Suppose now that $\mathbb{G}$ is some p-parameter group and identify $\tau = [\ \tau_1 \mid \tau_2 \mid \ldots \mid \tau_n\ ] \in \mathbb{R}^{p \times n}$ with the parameterizations of all of the transformations. For $\Delta\tau = [\ \Delta\tau_1 \mid \Delta\tau_2 \mid \ldots \mid \Delta\tau_n\ ]$, write $M \circ (\tau + \Delta\tau) \approx M \circ \tau + \sum_{i=1}^{n} J_i \Delta\tau_i e_i$, where $J_i \doteq \frac{\partial}{\partial \zeta}(M_i \circ \zeta)|_{\zeta=\tau_i}$ is the Jacobian of the i-th measurement with respect to the parameters τ_i and $\{e_i\}$ denotes the standard basis for $\mathbb{R}^n$. This leads to a convex optimization problem involving the variables $L, S, \Delta\tau$:

$$\min_{L,S,\Delta\tau} \|L\|_* + \lambda\|S\|_1 \quad \text{s.t. } M \circ \tau + \sum_{i=1}^{n} J_i \Delta\tau e_i e_i^T = L + S \tag{1.28}$$

Using a successive approximation of the changes $\Delta\tau$ leads to Algorithm 1.

Algorithm 1: RASL (Robust Alignment by Sparse and Low-rank Decomposition

Input: $M = [\ M_1 \mid M_2 \mid \ldots \mid M_n]$, initial transformation $\tau_1, \tau_2, \ldots, \tau_n$ in a certain parametric group $\mathbb{G}$, weight $\lambda > 0$.

while *not converged* **do**

Step 1: compute Jacobian matrices w.r.t. transformation:

$$J_i \leftarrow \frac{\partial}{\partial \zeta}(M_i \circ \zeta)|_{\zeta=\tau_i}$$

Step 2 (inner loop): solve the linearized convex optimization:

$$(L^*, S^*, \Delta\tau^*) \leftarrow \arg\min_{L,S,\Delta\tau} \|L\|_* + \lambda\|S\|_1 \quad \text{s.t. } M \circ \tau + \sum_{i=1}^{n} J_i \Delta\tau e_i e_i^T = L + S$$

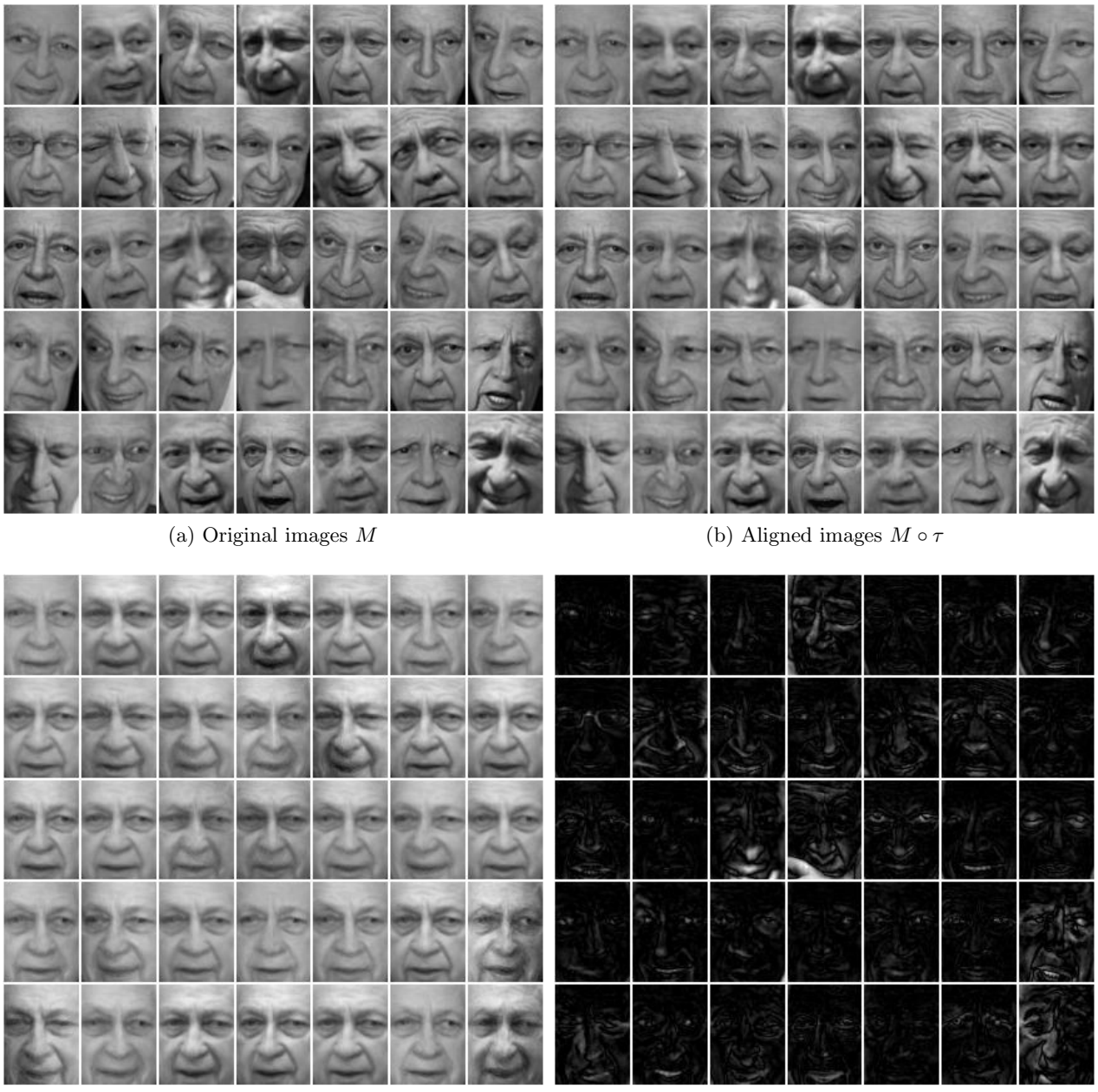
Step 3: update the transformation: $\tau \leftarrow \tau + \Delta\tau^*$

Output: L^*, S^*, τ^*

The authors in [26] have shown that the RASL algorithm can in fact be viewed as a Gauss-Newton method for minimizing the composition of a nonsmooth covex function with a smooth, nonlinear mapping. The convergence behavior of such algorithms was extensively studied in the late 1970's and early 1980's [16]. Although the authors claim that the result of [16] implies that RASL converges quadratically in the neighborhood of any strongly unique local minimum, a complete convergence analysis for RASL has yet to be developed.

Another remark is that the inner loop (Step 2) in the RASL algorithm is another convex optimization problem, this problem can be solved by Augmented Lagrange Multiplier (ALM) algorithm [21, 10]. We do not derive this algorithm here, but its derivation is similar to that of the ALM algorithm for Robust PCA that we will discuss in the Algorithms chapter of this report. Similarly, although the algorithm for Step 2 always converges to the optimal solution in numerical experiments, a rigorous proof of convergence so far has not been given. As in every iteration in RASL, a standard PCP problem needs to be solved in the inner loop, the computational complexity of RASL will be higher than that of standard Robust PCA. The authors observed that in their experiments only a few iterations, typically less than 20, were required for the algorithm to converge due to the outer algorithm's quadratic convergence property.

RASL can be used for example for long-standing batch image alignment problem: given many images of an object or objects of interest, the task is to align them to a fixed canonical template. The images may be subject to different illumination variation, partial occlusion, as well as poor or even no alignment. Figure 1.1 shows an example of how RASL performs on an image alignment problem:



(a) Original images M

(b) Aligned images $M \circ \tau$

(c) Low-rank component L

(d) Sparse component S

Figure 1.1: Aligning face images.

1.4.4 Robust Matrix Decomposition With Sparse Corruptions

Introduction

In the problem of solving Robust PCA, exact recovery via convex optimization of the ℓ_1 and nuclear norm heuristic is possible with high probability if the sparse noise is evenly spread out. However, in some applications, the sparse noise may not be evenly spread out and we are satisfied as long as the recovered matrix is sufficiently close with the original matrix. Therefore, the authors [14] analyze the Robust PCA problem in a deterministic setting without assuming a particular sparsity pattern of the noise. They give sufficient conditions on the level of sparsity that is required for a tolerable recovery of the sparse and low-rank pairs.

Main ideas and contributions

Similar to the prior study of Robust PCA, the authors of [14] study a minimization of a weighted combination of ℓ_1 and nuclear norm . However, they also introduce some tolerance on the perturbation in the constraints. In particular, given the observed matrix M (a perturbed observation of the original (L_0, S_0) pairs), they analyze the following two optimization problems:

$$\begin{aligned} \min_{(L,S)} \quad & ||L||_* + \lambda||S||_1 \\ s.t. \quad & ||L + S - M||_1 \leq \epsilon_1 \\ & ||L + S - M||_* \leq \epsilon_* \end{aligned} \tag{1.29}$$

and the regularized version

$$\min_{(L,S)} \quad ||L||_* + \lambda||S||_1 + \tfrac{1}{2\mu}||L + S - M||_F \tag{1.30}$$

The authors provide sufficient conditions on the pair (L_0, S_0) that allow accurate recovery in the sense that $||L_0 - \hat{L}||_\infty, ||S_0 - \hat{S}||_\infty$ is small. Moreover, they show that if the observed matrix M is perturbed from $L_0 + S_0$ by a small amount (i.e. ϵ), the optimizer $(\hat{L}, \hat{S})$ will still be ϵ-close to the original (L_0, S_0) pair. As in Robust PCA, the key ideas of the analysis of the performance guarantee is based on the properties of the constructed dual certificate.

Discussion and applications

The work [14] parallels the work of [10] but gives a weaker guarantee in a deterministic setting. We believe that the performance guarantee in this [14] and related work is also useful in many applications. For example, in computer vision, the noise is normally clustered at some portion of the image and it is not natural to assume the noise to be uniformly distributed. Moreover, in many real world applications, we are often satisfied with approximate recovery. For example, in many analyses of large data sets (say for example consumer ranking in Netflix), we are interested in the general idea of the pattern but not necessarily the fine details. At this point it is still unclear whether the sufficient conditions provided for the perturbed problem are also necessary.

Application to image processing

1.5 Robust PCA with known rank: a block coordinate descent approach

1.5.1 Motivation

In some applications we may have some prior information about the low rank component of an observed matrix. For example, in computer vision, if we were to extract the background from different frames of a video, then it is natural to consider each video frame as a long vector (by stacking its columns). The background that we are recovering is then a single vector, which means that the associated component in the matrix collecting the frames is of rank 1. Therefore, it is a very natural question to ask whether we can utilize this additional rank information to derive faster or more scalable algorithms while retaining the performance guarantees of Robust PCA.

1.5.2 Equivalent formulation of Robust PCA with rank information

We first show the intuitive fact that within the robust PCA framework, the same probability guarantee will still hold when additional information on the problem is incorporated as constraints to the optimization problem. Then we derive a block-coordinate descent algorithm for the case when rank information is known.

Proposition 7. *Let $M = L_0 + S_0$, $rank(L_0) \leq r$ and (L_0, S_0) satisfy the Robust PCA assumptions. Then with high probability, the following problems are equivalent:*

$$\begin{aligned} J_1 \quad &= \min_{L,S} \quad \|L\|_* + \lambda \|S\|_1 \\ &\quad s.t. \qquad M = L + S \end{aligned} \tag{1.31}$$

$$\begin{aligned} J_2 \quad &= \min_{L,S} \quad \|L\|_* + \lambda \|S\|_1 \\ &\quad s.t. \qquad M = L + S \\ &\qquad\qquad \textit{and } \ T(L,S) \textit{ holds} \end{aligned} \tag{1.32}$$

where $T(L, S)$ are conditions on (L, S) that (L_0, S_0) also satisfies.

Proof. We use subscript to denote the optimizer for J_1 and J_2 respectively. With high probability, $(L_1, S_1) = (L_0, S_0)$. Now, over the event of $(L_1, S_1) = (L_0, S_0)$, (L_1, S_1) is also a feasible solution for (1.32). Since $J_1 \leq J_2$ always, now (L_1, S_1) achieve the bound for (1.32). Note that it should also be unique because otherwise it would contradict with the recovery of (1.31). □

Now we apply Proposition (7) to the case when the rank information is known and derive a block coordinate descent algorithm.

Consider the problem

$$\begin{aligned} J_3 = &\min_{L,S} && \|L\|_* + \lambda\|S\|_1 \\ &\text{s.t.} && M = L + S,\ \text{rank}(L) \le r \\ = &\min_{L,S,p_i,q_i,\mu_i} && \|L\|_* + \lambda\|M - L\|_1 \\ &\text{s.t.} && L = \sum_{i=1}^{r} \mu_i p_i q_i^T,\ p_i^T p_j = \delta_i^j,\ q_i^T q_j = \delta_i^j,\ \mu_i \ge 0 \\ = &\min_{\mu_i,p_i,q_i} && \sum_{i=1}^{r} \mu_i + \lambda\|M - \sum_{i=1}^{r} \mu_i p_i q_i^T\|_1 \\ &\text{s.t.} && p_i^T p_j = \delta_i^j,\ q_i^T q_j = \delta_i^j, \mu_i \ge 0 \\ = &\min_{\mu_i,p_i,q_i} && \sum_{i=1}^{r} |\mu_i| + \lambda\|M - \sum_{i=1}^{r} \mu_i p_i q_i^T\|_1 \\ &\text{s.t.} && p_i^T p_j = \delta_i^j,\ q_i^T q_j = \delta_i^j,\ \mu_i \ge 0 \end{aligned} \tag{1.33}$$

Here $L = \sum_{i=1}^{r} \mu_i p_i q_i^T$ is the SVD of L, hence the condition of orthonormality $p_i^T p_j = \delta_i^j,\ q_i^T q_j = \delta_i^j$. Note that this formulation allows us to optimize over μ_i, p_i and q_i sequentially. By Proposition 7, we know that the formulation of (1.33) can recover the original pair (L_0, S_0) with high probability.

1.5.3 Simplification using ℓ_1 heuristic

Introduction

Recall that the nuclear norm is used in the PCP scheme as a heuristic to recover a low rank component corrupted by gross random noise. The nuclear norm is used because it penalizes high rank matrices and encourages sparsity of the vector of singular values. Now, we consider the case when we have additional information about the rank of the low-rank matrix L. Assume the rank of the matrix is known. Since information about the rank of the matrix is available, one can replace the nuclear norm heuristic by writing L in the form $L = \sum_{j=1}^{r} p_j q_j^T$. This guarantees $\mathbf{rank}(L) \le r$. Note that p_j's and q_j's do not necessarily give the directions of the singular vectors of L, since they are not assumed to be orthogonal. This results in the following heuristic

$$E^* = \min_{\{p_j\}\{q_j\},1\le j\le r} \|M - \sum_{j=1}^{r} p_j q_j^T\|_1 \tag{1.34}$$

Performance guarantee for the ℓ_1 heuristic

For this new heuristic, we provide some performance guarantee for the case when the noise is bounded. One is a result for deterministic case and the other is for the random case. They are as follows.

Proposition 8. *Let* $M = S + \sum_{i=1}^{r} p_i q_i^T$ *and* $\frac{2}{\epsilon}\|S\|_1 \leq \|\sum_{i=1}^{r} p_i q_i^T\|_1$. *Then, the estimate* $\hat{L}$ *recovered from* (1.34) *satisfies*

$$\frac{\|\sum_{i=1}^{r} p_i q_i^T - \hat{L}\|_1}{\|\sum_{i=1}^{r} p_i q_i^T\|_1} \leq \epsilon$$

Proof. Suppose not, then

$$\begin{aligned}
\|S\|_1 &\geq \|\sum_{i=1}^{r} p_i q_i^T + S - \hat{L}\|_1 \\
&\geq \|\sum_{i=1}^{r} p_i q_i^T - \hat{L}\|_1 - \|S\|_1 \\
&> \epsilon\|\sum_{i=1}^{r} p_i q_i^T\|_1 - \|S\|_1
\end{aligned}$$

which contradicts the assumption that

$$\frac{2}{\epsilon}\|S\|_1 > \|\sum_{i=1}^{r} p_i q_i^T\|$$

□

Proposition 9. *Let* $M = \sum_{i=1}^{r} p_i q_i^T + S$, *where* $S_{i,j} \sim Uniform(-x_s, x_s)$, $(p_i)_j \sim Uniform(-x_p, x_p)$, $(q_i)_j \sim Uniform(-x_q, x_q)$, *where all random variables are independent. With* $|S| = k$ *such that* $\lim_{n\to\infty} \frac{k^2}{n}$, *then we have,*

$$\lim_{n\to\infty} P\left(\frac{\|\sum_{i=1}^{r} p_i q_i^T - \hat{L}\|_1}{\|\sum_{i=1}^{r} p_i q_i^T\|_1} > \epsilon\right) = 0$$

Proof. Let E be the error event that $\frac{\|\sum_{i=1}^{r} p_i q_i^T - \hat{L}\|_1}{\|\sum_{i=1}^{r} p_i q_i^T\|_1} > \epsilon$. If error occurs,

$$\begin{aligned}
kx_s &\geq \|\sum_{i=1}^{r} p_i q_i^T + S - \hat{L}\|_1 \\
&\geq \|\sum_{i=1}^{r} p_i q_i^T - \hat{L}\|_1 - \|S\|_1 \\
&\geq \epsilon\|\sum_{i=1}^{r} p_i q_i^T\|_1 - kx_s \\
&\geq \epsilon\sqrt{\sum_{l_1 l_2}(\sum_{i=1}^{r}(p_i)_{l_1}(q_i)_{l_2})^2} - k_s
\end{aligned}$$

Thus,

$$\begin{aligned} Pr(E) &\leq Pr\left(\left(\frac{2kx_s}{\epsilon}\right)^2 \geq \sum_{l_1 l_2}\left(\sum_{i=1}^{r}(p_i)_{l_1}(q_i)_{l_2}\right)^2\right) \\ &\leq Pr\left(\left(\frac{2kx_s}{\epsilon}\right)^2 \geq \sum_{l_1=1}^{n}\left(\sum_{i=1}^{r}(p_i)_{l_1}(q_i)_{l_1}\right)^2\right) \\ &= Pr\left(\frac{1}{n}\left(\frac{2kx_s}{\epsilon}\right)^2 \geq \frac{1}{n}\sum_{l_1=1}^{n}\left(\sum_{i=1}^{r}(p_i)_{l_1}(q_i)_{l_1}\right)^2\right) \end{aligned}$$

Moreover, as $E(\sum_{i=1}^{r}(p_i)_{l_1}(q_i)_{l_1})^2) = \frac{r}{3}x_p^2x_q^2$, by the law of large numbers, $\frac{1}{n}\sum_{l_1=1}^{n}(\sum_{i=1}^{r}(p_i)_{l_1}(q_i)_{l_1})^2) \to \frac{r}{3}x_p^2x_q^2$. Thus, since $\frac{1}{n}(\frac{2kx_s}{\epsilon})^2 \to 0$. This gives $Pr(E) \to 0$ as $n \to \infty$. □

However, we know that the ℓ_1 heuristic cannot work well in the case of unbounded noise. This can be seen from the following example. Say, for $n = 100$,

$$L_0 = \begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix}\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}, \qquad S_0 = \begin{bmatrix}10^9 & 0 & \dots & 0\\0 & 0 & \dots & 0\\\vdots & \vdots & & \vdots\\0 & 0 & \dots & 0\end{bmatrix}$$

then by the ℓ_1 heuristic, we would get

$$\hat{L}_0 = \begin{bmatrix}10^9\\1\\\vdots\\1\end{bmatrix}\begin{bmatrix}1 & 10^{-9} & \dots & 10^{-9}\end{bmatrix}, \qquad \hat{S}_0 = \begin{bmatrix}1 & 0 & \dots & 0\\0 & 1 & \dots & 1\\\vdots & \vdots & & \vdots\\0 & 1 & \dots & 1\end{bmatrix}$$

which deviates grossly from the original pair. However, Robust PCA can achieve exact recovery even in this case because it penalize the nuclear norm of L.

1.5.4 A block coordinate descent algorithm

For simplicity we restrict our discussion to $r = 1$ in (1.34). Let $M = (M_{i,j}) \in R^{n\times m}$. The problem is then

$$\min_{p\in\mathbb{R}^n, q\in\mathbb{R}^m} \|M - pq^T\|_1 \quad \text{subject to } \|p\|_\infty = 1$$

One can use a block coordinate descent algorithm, similar to power iteration, by iteratively fixing one vector and optimizing on the other.

q-step

For a fixed $p \in \mathbb{R}^n$, the problem is

$$\min_{q\in\mathbb{R}^m}\sum_{j=1}^{m}\|M_j - q_j p\|_1 = \sum_{j=1}^{m}\min_{q_j}\|M_j - q_j p\|_1$$

where $M_j \in \mathbb{R}^n$ is the j-th column of M. Each subproblem is an ℓ_1-norm projection (projection of M_j onto span(p))

$$\min_t \|z - tp\|_1$$

This is a weighted median problem $\min_t \|z - tp\|_1 \Leftrightarrow \min_t \sum_{i \in \text{supp(p)}} |p_i| \left|\frac{z_i}{p_i} - t\right|$.

Median problem

The median problem is

$$\min_t \sum_{i=1}^{n} |z_i - t|$$

and an optimizer is $t = z_{[k]}$ where $z_{[i]}$ is the i-th element in the ordered sequence, k is the maximum index such that $k \leq n - k$ i.e. $\lfloor \frac{n}{2} \rfloor \leq k \leq \lceil \frac{n}{2} \rceil$. This can be solved by simply sorting the elements of z, which can be done in $O(n \log n)$.

Note that computing the median can be done in $O(n)$ if the elements are not sorted entirely: using the quick-select algorithm, a modified version of the quicksort algorithm. At each step, assume we have a list of k elements. We choose a random pivot, and in one pass compute the elements that are less than, respectively greater than, the pivot. From the length of each list, we know where to look for the median next. If the sizes of the sublists are sufficiently balanced, this will result in $O(n)$ algorithm. In the exactly balanced case, the complexity is proportional to $n + \frac{n}{2} + \frac{n}{4} + \cdots + \frac{n}{\log_2 n} = n\frac{1-(1/2)^{1+\log_2 n}}{1-1/2} \leq 2n$. In the case the size of each sublist is no less than α times the size of the parent list, the complexity is also linear, since it is proportional to $n + \alpha n + \alpha^2 n + \cdots + \alpha^{\log_{1/\alpha} n} n \leq \frac{1}{1-\alpha} n$. Although the worst-case complexity is $O(n^2)$, the algorithm is linear in practice. Assuming a uniform distribution of the permutation that will result in the sorted list, the expected complexity is $O(n)$.

Weighted median problem

The solution to the weighted median problem is slightly different

$$\min_t \sum_{i=1}^{n} \alpha_j |z_i - t|$$

where $\alpha_i > 0$. An optimizer is $t = z_{[k]}$ where k is the maximum index such that $\sum_{i=1}^{k} \alpha_i \leq \sum_{i=k+1}^{n} \alpha_i$. This is shown by using the property of sub-differential of $\|.\|_1$ and by noting that $0 \in \partial(\sum_{i=1}^{n} \alpha_j |z_i - t|)$ for $t = z_{[k]}$. Finding this optimal t can also be achieved by sorting.

If we denote this subproblem by

$$\begin{aligned}
\mathbf{wmed}(M, p) &= \arg\min_t \|M - pt^T\|_1 \\
&= \arg\min_t \sum_j \|M_j - t_j p\|_1 \\
&= \arg\min_t \sum_j \sum_{i \in \mathbf{supp}(p)} |p_i| \left|\frac{M_{ij}}{p_i} - t_j\right|
\end{aligned}$$

where M_j is the j-th column of M, then the q-step is simply given by

$$q \leftarrow \mathbf{wmed}(M, p)$$

and the complexity of the q-step is $O(mn \log n)$. The **wmed** algorithm is summarized in Algorithm [**?**].

Algorithm 2: q = $\mathbf{wmed}(M, p)$

Input: Data matrix M, vector p

for $j \in \{1, \ldots, m\}$ **do**

1. Define vectors z and α: $\forall i \in \mathbf{supp}(p)$, $\alpha_i = |p_i|$ and $z_i = \frac{M_{ij}}{p_j}$.
2. Sort z s.t. $z_{i_1} \leq z_{i_2} \leq \cdots \leq z_{i_n}$
3. Find the maximum k s.t. $\sum_{l=1}^{k} \alpha_{i_l} \leq \sum_{l=k+1}^{n} \alpha_{i_l}$
4. Set $q_j = z_{i_k}$

Output: vector q

p-step

For a fixed $q \in \mathbb{R}^m$, the problem is given by

$$\min_{\|p\|_\infty = 1} \sum_{i=1}^{n} \|M_i - p_i q\|_1$$

where M_i is the i-th row of M. We can denote this problem by

$$p \leftarrow \mathbf{cwmed}(M^T, q)$$

where $\mathbf{cwmed}(M^T, q)$ is the solution of the constrained weighted median problem

$$\min_{\|p\|_\infty = 1} \sum_{i} \|M_i - p_i q\|_1 = \sum_{i=1}^{n} \min_{|p_i| \leq 1} \|M_i - p_i q\|_1 \text{ subject to } \max_i |p_i| = 1$$

We can start by solving n constrained weighted median problems $\min_{|p_i| \leq 1} \|M_i - p_i q\|_1$. This is solved by computing the solution to the unconstrained weighted median

$$p_i = \arg\min_{t_i} \sum_{j \in \mathrm{supp}(q)} |q_j| \left| \frac{M_{ij}}{q_j} - t_i \right|$$

then projecting p_i on $[-1, 1]$.

After solving the n subproblems, there are 2 cases. Either one of the p_i's satisfies $|p_i| = 1$, in which case the $\|p\|_\infty = 1$ constraint is satisfied and the problem is solved, or all the p_i's are such that $|p_i| < 1$. In this case, we choose which p_i to project on $[-1, 1]$ by solving the problem

$$\begin{array}{ll} \text{minimize}_{i,\epsilon} & -\|M_i - p_i q\|_1 + \|M_i - \epsilon q\|_1 \\ \text{subject to} & i \in \{1, \ldots, n\} \\ & \epsilon \in \{-1, 1\} \end{array} \tag{1.35}$$

this problem finds (i, ϵ) that minimize the increase of the objective value that is due to setting $p_i = \epsilon$. Then if (i_0, ϵ_0) is a minimizer, we set $p_{i_0} = \epsilon_0$. This additional projection step is $O(nm)$ since there are $2n$ feasible points to evaluate in problem (1.35). The **cwmed** algorithm can be summarized by Algorithm 3

Algorithm 3: $\mathbf{p} = \mathbf{cwmed}(M^T, q)$

Input: Data matrix M, vector q

for $i \in \{1, \ldots, n\}$ **do**

1. Define vectors z and α: $\forall j \in \mathbf{supp}(q)$, $\alpha_j = |q_j|$ and $z_j = \frac{M_{ij}}{p_i}$.
2. Sort z s.t. $z_{j_1} \leq z_{j_2} \leq \cdots \leq z_{j_n}$
3. Find the maximum k s.t. $\sum_{l=1}^{k} \alpha_{j_l} \leq \sum_{l=k+1}^{n} \alpha_{j_l}$
4. Set $p_i = z_{j_k}$
5. Project p_i on $[-1, 1]$

if $\|p\|_\infty < 1$ **then**

1. Let $(i_0, \epsilon_0) = \arg\min_{i,\epsilon} -\|M_i - p_i q\|_1 + \|M_i - \epsilon q\|_1$
2. Set $p_{i_0} = \epsilon_0$

Output: vector p

Algorithm for the constrained problem

The Constrained rank one ℓ_1 heuristic is given in Algorithm 4

Algorithm 4: Constrained rank one ℓ_1 heuristic

while *not converged* **do**

$$q \leftarrow \mathbf{wmed}(M, p)$$
$$p \leftarrow \mathbf{cwmed}(M^T, q)$$

The complexity of each iteration is $O(nm \log(nm))$ if the weighted median is computed by sorting the vectors, or $O(nm)$ if the quick select algorithm is used instead. It is important to point out that while the ℓ_1 heuristic enjoys recovery guarantees, we have no guarantee that this iterative algorithm will converge to an optimizer of the problem. However, we observe that this heuristic performs well in practice when the low rank component L is rank one. We also observe that solving an unconstrained version of the problem, where vectors p and q are normalized at each iteration, works better in practice. An example simulation is shown in Figure **??**, where a simulation was run using randomly generated rank-one 10×10 matrices, corrupted with random noise matrices with increasing support. For each support size, 20 simulations are run. The plots show the average results.

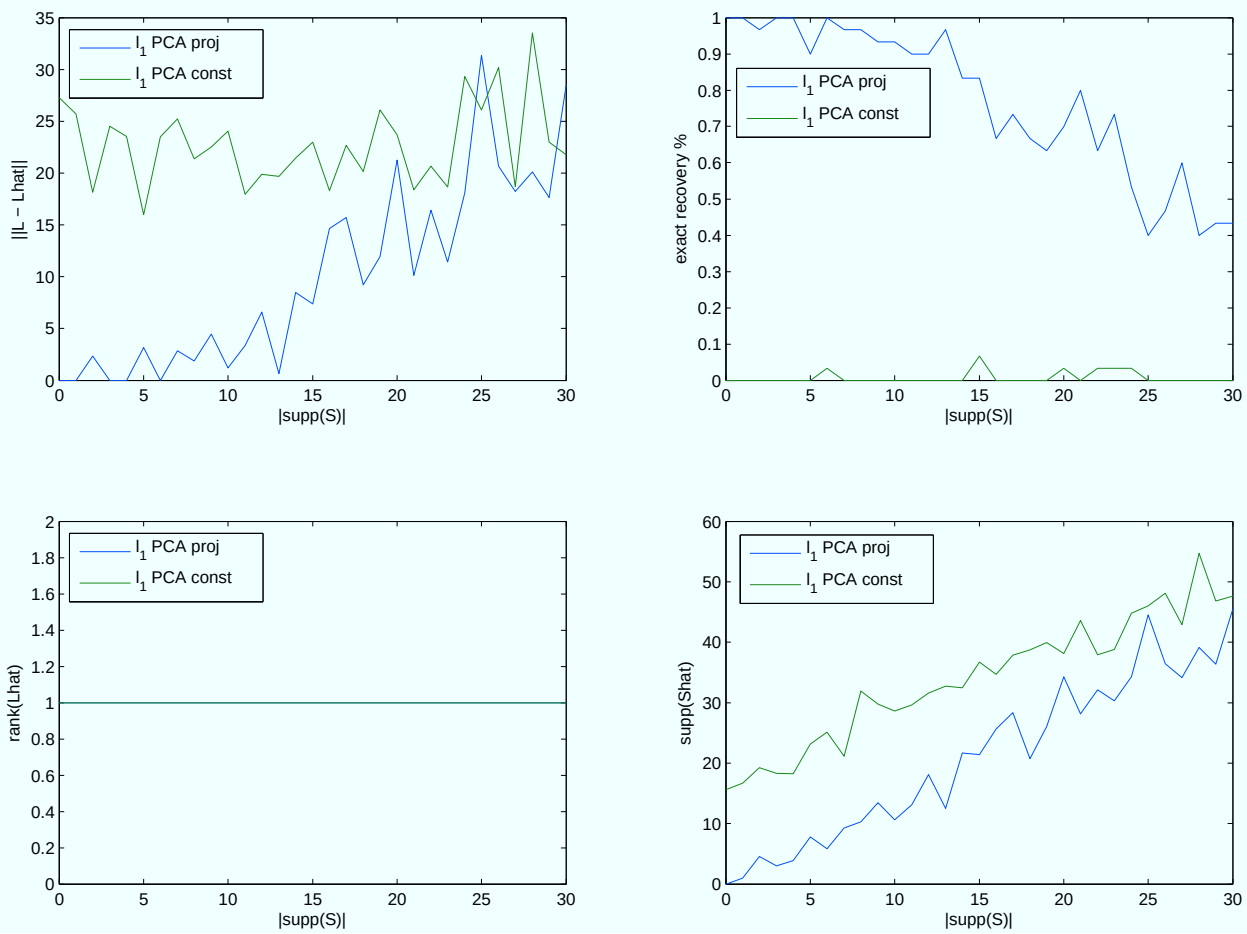


Figure 1.2: Performance of the constrained ℓ_1 PCA heuristic (green) Vs. the projected ℓ_1 PCA heuristic (blue), for increasing size of the support of the corruption matrix S. The projected method performs considerably better (higher exact recovery rate, lower average ℓ_1 error, support size of the recovered $\hat{S}$ closer to the original support).

The unconstrained version is detailed next.

Algorithm for the unconstrained projected problem

Another approach is to apply block coordinate decent to the unconstrained problem

$$\min_{p\in\mathbb{R}^n, q\in\mathbb{R}^m} \|M - pq^T\|_1$$

but project the vectors at each iteration. This results in Algorithm 5

The complexity of each iteration is $O(nm\log(nm))$ if the weighted median is computed by sorting the vectors, or $O(nm)$ if the quick select algorithm is used instead.

Algorithm 5: Projected rank one ℓ_1 heuristic

while *not converged* **do**

$$q \leftarrow \mathbf{wmed}(M, p)$$
$$q \leftarrow q/\|q\|_\infty$$
$$p \leftarrow \mathbf{wmed}(M^T, q)$$
$$q \leftarrow p/\|p\|_\infty$$

convergence in the rank one case

Note on convergence when there is no noise. Assume A is rank one (no observation noise), and let $A = \sigma u v^T$ where $\sigma > 0$, u and v are unit vectors.

Then the p-step is given by

$$\begin{aligned} p_i &= \arg\min_{p_i} \|\sigma u_i v - p_i q\|_1 \\ &= \arg\min_{p_i} \sum_{j=1}^{n} |\sigma u_i v_j - p_i q_j| \\ &= \arg\min_{p_i} \sum_{j=1}^{n} |q_j| \left| \frac{\sigma u_i v_j}{q_j} - p_i \right| \\ &= \sigma u_i \tilde{v}_{[k]} \end{aligned}$$

where $\tilde{v}_i = \frac{v_i}{q_i}$, $\tilde{v}_{[k]}$ is the k-th element in the ordered sequence of $\tilde{v}$. Note that the sign of u_i does not affect the solution, even though it may seem to affect the order of the sequence: if $u_i < 0$, simply write the optimization problem as $\min_{p_i} \|\sigma u_i v - p_i q\|_1 = \min_{p_i} \|\sigma(-u_i)v - (-p_i)q\|_1$ where now $-u_i > 0$ and the variable is $-p_i$. The solution in the positive case yields $-p_i = \sigma(-u_i)\tilde{v}_{[k]}$, which is the same as $p_i = \sigma u_i \tilde{v}_{[k]}$.

Therefore we have that after the first iteration,

$$p = \tilde{v}_{[k]} u$$

since $\tilde{v}_{[k]}$ does not depend on i (the order of the sequence only depends on q), thus we immediately recover the direction of the left singular vector u. Similarly, q recovers the direction of the right singular vector v, and the algorithm converges after one iteration. This is observed in practice (however since the stopping needs to detect that the vectors are not changing, this requires two iterations in practice).

1.5.5 Sensitivity of the Robust PCA solution to λ

Note that in the Robust PCA framework, the parameter λ is specifically chosen to be $\lambda = \frac{1}{\sqrt{n}}$, and when λ is too large or too small, it would significantly affect the recovery. However, in the case where rank information is given, the effect of λ may be different. One may ask whether recovery can be be guaranteed with very small values of λ. In particular, we specialize to the rank 1 case, and it turns out that it cannot be done, as demonstrated in the following.

Recall that if we directly apply Robust PCA we will get

$$\begin{aligned}
\min_{M=L+S,rank(L)\leq 1} \|L\|_* + \lambda\|S\|_1 &= \min_{S=M-pq^T,L=pq^T} \|L\|_* + \lambda\|S\|_1 \\
&= \min_{S=M-pq^T,L=pq^T} \|pq^T\|_* + \lambda\|M-pq^T\|_1 \\
&= \min_{p,q:\|p\|_2=1} \|pq^T\|_* + \lambda\|M-pq^T\|_1 \\
&= \min_{p,q:\|p\|_2=1} \|q\|_2 + \lambda\|M-pq^T\|_1 \\
&= \min_{p:\|p\|_2=1} \min_q \|q\|_2 + \lambda\|M-pq^T\|_1
\end{aligned}$$

Now, for every fixed p, consider the subproblem of directly applying Robust PCA with $\lambda \leq \frac{1}{n}$,

$$\begin{aligned}
\min_q \|q\|_2 + \lambda\|M-pq^T\|_1 &= \min_q \max_{\|u\|_2\leq 1,\|V\|_\infty\leq 1} u^Tq + \lambda Tr(V^T(M-pq^T)) \\
&= \max_{\|u\|_2\leq 1,\|V\|_\infty\leq 1} \min_q u^Tq + \lambda Tr(V^T(M-pq^T)) \\
&= \max_{\|u\|_2\leq 1,\|V\|_\infty\leq 1,u=\lambda V^Tp} \lambda Tr(V^TM) \\
&= \max_{\|\lambda V^Tp\|_2\leq 1,\|V\|_\infty\leq 1} \lambda Tr(V^TM) \\
&= \max_{\|V^Tp\|_2\leq n,\|V\|_\infty\leq 1} \lambda Tr(V^TM)
\end{aligned}$$

Now note that, since $\|p\|_2 \leq 1$, we have $\|V^Tp\|_2 \leq \sqrt{\sum_{i=1}^n \sigma_i(V^TV)} = \sqrt{Tr(V^TV)} \leq \sqrt{n^2\|V\|_\infty}$. Thus, the optimal value is

$$\begin{aligned}
\min_{p,q:\|p\|_2=1} \|pq^T\|_* + \lambda\|M-pq^T\|_1 &= \min_{p:\|p\|_2=1} \max_{\|V\|_\infty\leq 1} \lambda Tr(V^TM) \\
&= \lambda \min_{p:\|p\|_2=1} \|M\|_1 \\
&= \lambda\|M\|_1
\end{aligned}$$

And it is achieved by $pq^T = 0$ matrix, which deviates from what we expect to recover.

1.5.6 Recovery performance

Comparison between Robust PCA and ℓ_1 PCA heuristics

We perform a numerical comparison of the performance of Robust PCA and the ℓ_1 heuristic based on the weighted median iteration method. We test recovery of corrupted rank one matrices. Note that in using the power iteration method for Robust PCA, we would not update μ if the value of that iteration is 0 because this will make the algorithm to converge to the wrong value (as observed from simulation, this happens quite frequently so this conditioning is needed).

Note that in the rank one case, the formulation (1.33) of Robust PCA becomes

$$\begin{aligned} J_3 = &\min_{L,S} \|L\|_* + \lambda \|S\|_1 \\ &\text{s.t. } M = L + S, \ \text{rank}(L) = 1 \\ = &\min_{\mu,p,q} |\mu| + \lambda \|M - \mu p q^T\|_1 \\ &\text{s.t. } \|p\|_2 = 1, \ \|q\| = 1, \ \mu \geq 0 \end{aligned} \tag{1.36}$$

Therefore the rank one Robust PCA problem can be solved using a modified weighted median problem.

In the simulation, we randomly generated the entries of p and q as $N(0,1)$ iid. We randomly generate sparse matrix with a uniformly distributed sparse support. Each sparse entry is randomly distributed $N(0,1)$. We then plot the graph of different degree of sparsity and the corresponding effectiveness of the optimization heuristic in extracting the original pq^T. We run the experiment 20 times and show the average. A recovery is considered to be correct if the relative error of the low rank matrix and the recovered low rank matrix is less than 10^{-3}. We remark that we only use 20 number of iteration in the power iteration method where one iteration means updating all the entries of p and q once (i.e. outer iteration). The results is shown in FIgure **??**.

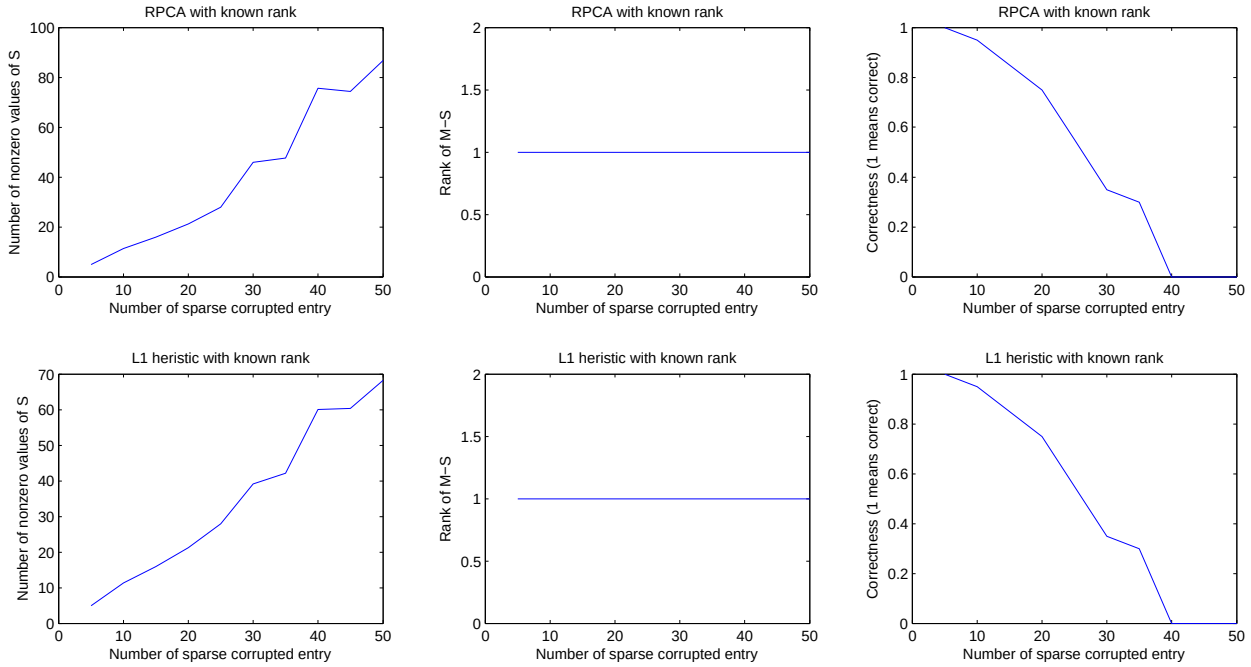


Figure 1.3: Comparison of rank one Robust PCA and the ℓ_1 PCA heuristic in solving rank-one matrix recovery. Both algorithms have similar performance, but ℓ_1 PCA shows slightly better recovery rate, and its estimated corruption matrix has size of support closer to reality for large supports.

If we do not use the rank information in the Robust PCA formulation, the exact recovery performance degrades compared to ℓ_1 PCA. This is shown in Figure **??**

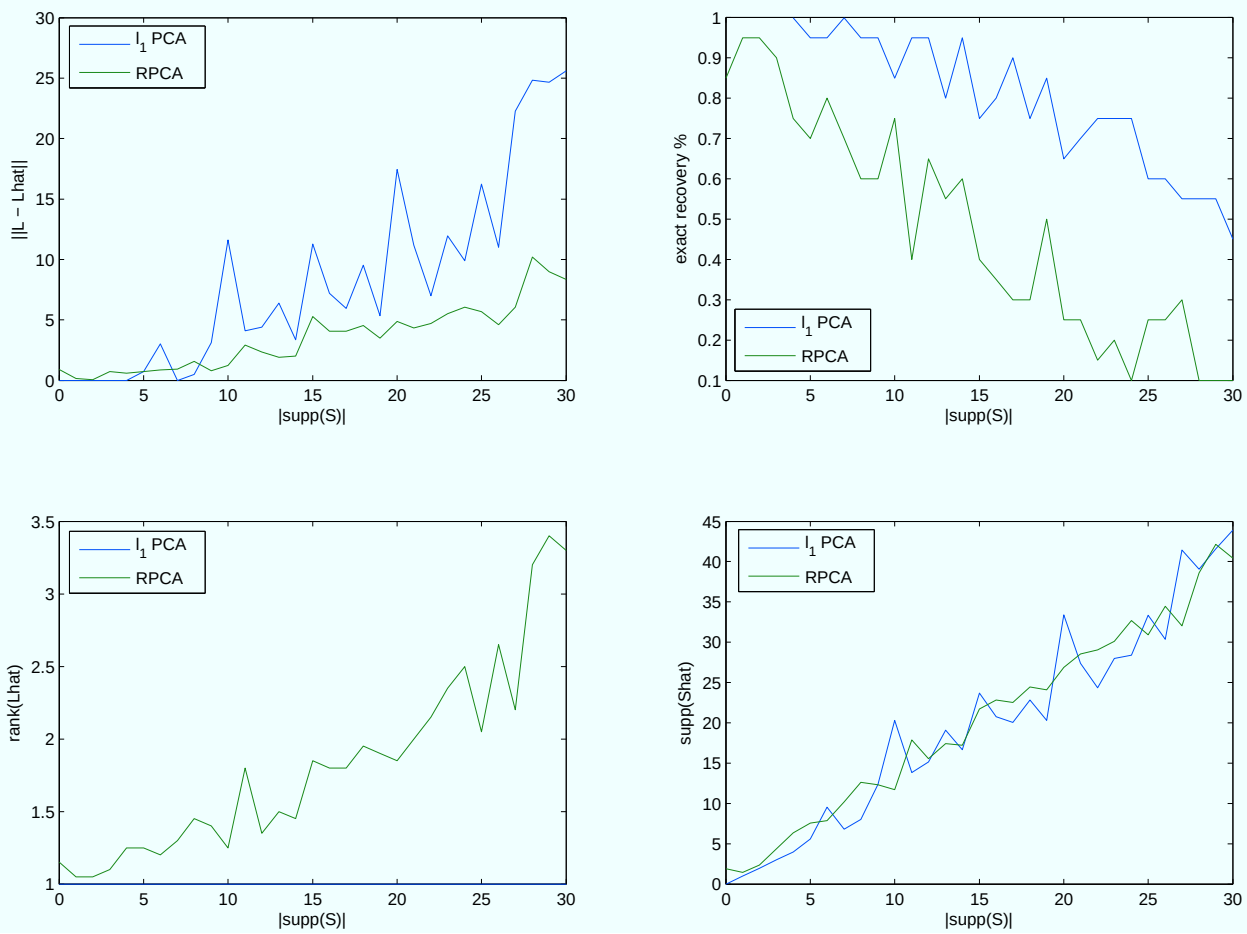


Figure 1.4: Comparison of pure Robust PCA (no rank information) and ℓ_1 PCA for solving rank one recovery, for increasing sizes of the corruption matrix. ℓ_1 PCA shows higher exact recovery rates, and lower average error. Observe that the rank of the estimated matrix is increasing in the case of RPCA, but is constant (by construction) in the case of ℓ_1 PCA.

1.5.7 ℓ_1 PCA heuristics for higher ranks

One can generalize the ℓ_1 PCA block coordinate descent algorithms to cases where L is of known rank $r \geq 1$, in two different ways depending on what we want to achieve, and on the performance requirements.

Analysis ℓ_1 PCA

In the analysis view of PCA, one seeks to *sequentially* find directions that best explain the data. In the ℓ_2 case, at each step, we look for a direction p that maximizes the variance along p, i.e. $\max_{\|p\|_2=1} p^T M M^T p = \max_{\|p\|_2=1} \|M^T p\|_2^2$. Equivalently, we look for p that minimizes the distances of the data points M_i to $\mathbf{span}(p)$ (M_i is the i-th column of M). Indeed, assuming $\|p\|_2 = 1$, the projection of data point M_i on p is given by $(M_i^T p)p$, and the distance of M_i to $\mathbf{span}(p)$ is simply $\|M_i - (M_i^T p)p\|_2^2 = \|M_i\|_2^2 - (M_i^T p)^2$. Thus minimizing the sum of the squared distances is equivalent

to

$$\min_{\|p\|_2=1} -\sum_i (M_i^T p)^2 = \max_{\|p\|_2=1} \|M^T p\|_2^2$$

For a general norm $\|.\|$, minimizing the distance to $\mathbf{span}(p)$ can be written

$$\min_{q,\|p\|_*=1} \|M^T - pq^T\|$$

where p is normalized in the dual sense, and $q_i p$ is the projection of M_i on p.

The analysis view of ℓ_1 PCA leads to a sequential formulation where at each step, a one-dimensional projection of the data is computed

$$\min_{q_k,\|p_k\|_\infty=1} \|M^{(k)} - p_k q_k^T\|_1$$

then the data matrix is updated (the covariance matrix is deflated)

$$M^{(k+1)} = M^{(k)} - p_k q_k^T$$

One way to solve this problem is to use block coordinate descent *separately for each pair* (p_k, q_k) (i.e. one block coordinate descent for each step k).

Synthesis ℓ_1 PCA

In the synthesis view of PCA, one seeks to find, *simultaneously*, a small set of directions (of size r), or dictionary elements, such that each data point M_i admits a decomposition on that set with low error. This leads to the batch problem

$$\min_{\|p_k\|_\infty=1,q_k} \|M - \sum_{k=1}^{r} p_k q_k^T\|_1$$

that can be solved using block coordinate descent for all vectors p_k, q_k Note that in the case of ℓ_2 PCA, both views are equivalent and lead to the same solution (p_k is the left singular vector corresponding to the k-th largest singular value σ_k, and $q_k/sigma_k$ is the corresponding right singular vector). However in the general case, these formulations are not equivalent. The synthesis view has slower convergence, since the optimization is done simultaneously on all vectors. The analysis view has faster convergence, but performs poorly compared to the analysis view if the criterion is the reconstruction error $\|M - \sum_{k=1}^{r} p_k q_k^T\|_1$. This is illustrated in the following section.

Numerical results

We first test the performance of the synthesis ℓ_1 PCA, for recovering rank-2 10×10 matrices. The results show that we do have a good performance when the noise is sparse enough.

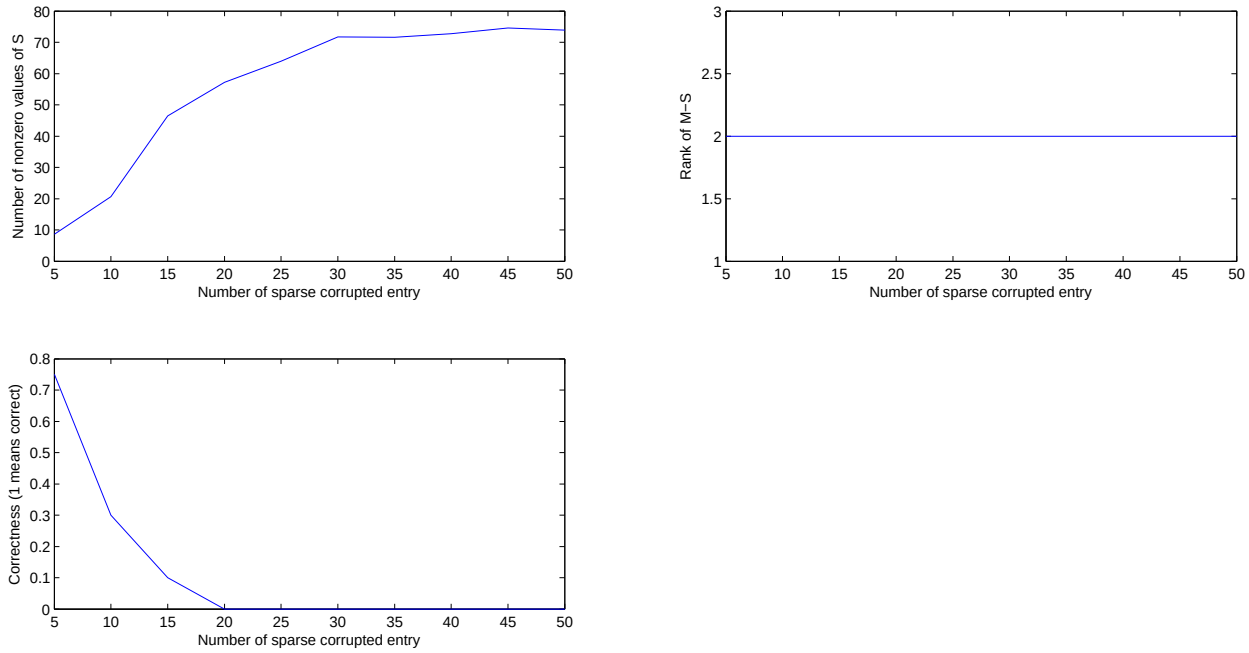


Figure 1.5: Performance of synthesis ℓ_1 PCA in recovering a rank 2 matrix using (batch) block coordinate descent.

We next compare the performance of the synthesis ℓ_1 PCA (green), analysis ℓ_1 PCA (blue), and Robust PCA with no rank information (red). We generate random rank-r 10×10 matrices, and add random uniformly distributed noise. In a first simulation (Figure 1.6), we fix the rank to be $r = 2$, and vary the size of the support of the corruption matrix S. In a second simulation (Figure 1.7), we fix the size of the support to be 20, and vary the rank of L. The results show that

- Synthesis ℓ_1 PCA performs better than analysis ℓ_1 PCA, as expected: it achieves higher exact recovery rate, although has comparable average error. However, convergence is much slower. Synthesis ℓ_1 PCA becomes impractical for large matrices (size of the order of thousand by thousand), but analysis ℓ_1 PCA still converges fast and yields relatively good recovery rates.
- The exact recovery rates of RPCA and ℓ_1 PCA are comparable, although RPCA achieves lower average error (and slightly better exact recovery, for example, when the rank is 3). The ℓ_1 PCA heuristics have an advantage in terms of rank and support of the estimated components: for higher ranks, the $\mathbf{rank}(\hat{L})$ is closer to the original $\mathbf{rank}(L)$, and the size of the support $|\mathbf{supp}(\hat{S})|$ is closer to the original $|\mathbf{supp}(S)|$ (see bottom sub-figures in Figure 1.7, original noise support size is $|\mathbf{supp}(S)| = 8$.)
- The ℓ_1 PCA heuristics therefore offer some advantages: they allow us to incorporate rank information (we do not see an easy way to add rank information to the RPCA formulation when the rank is strictly greater than one), and offer more robustness when the rank is high.

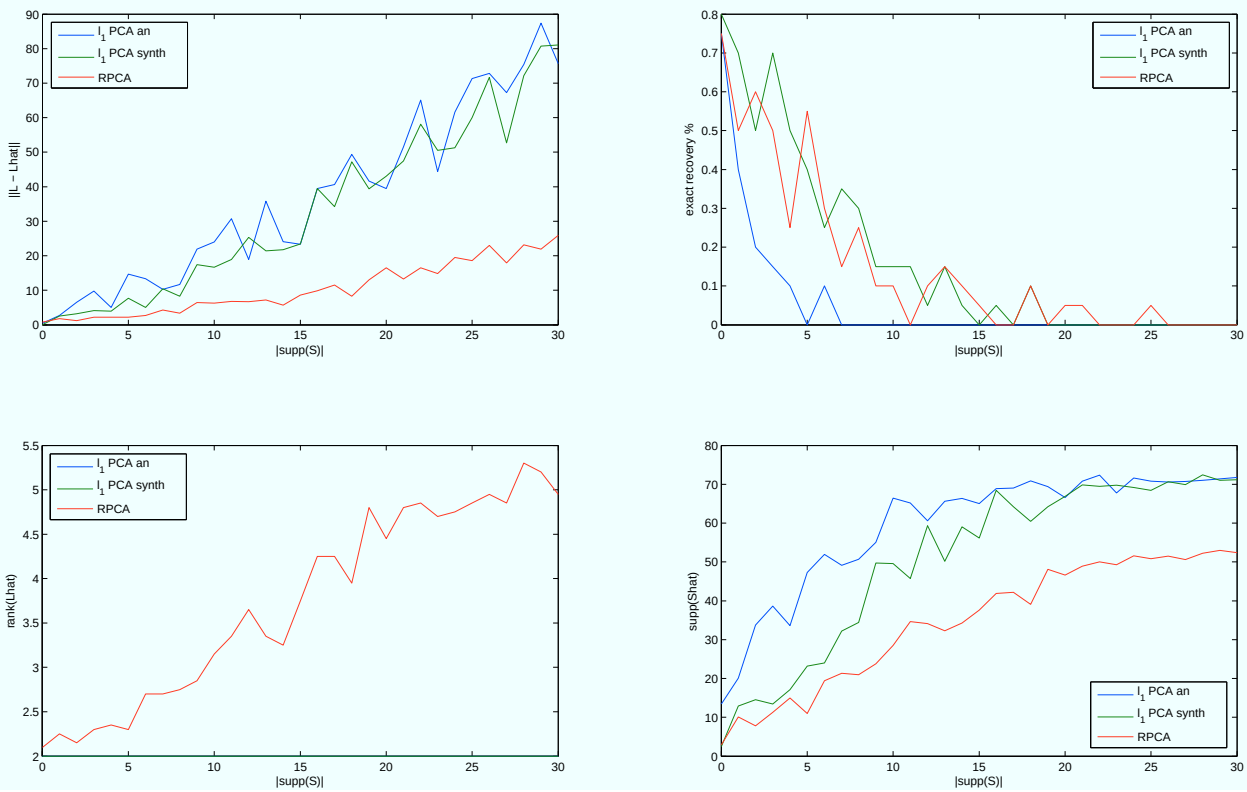


Figure 1.6: Analysis ℓ_1 PCA (blue), synthesis ℓ_1 PCA (green), and Robust PCA (red) for rank-2 recovery, by increasing noise support.

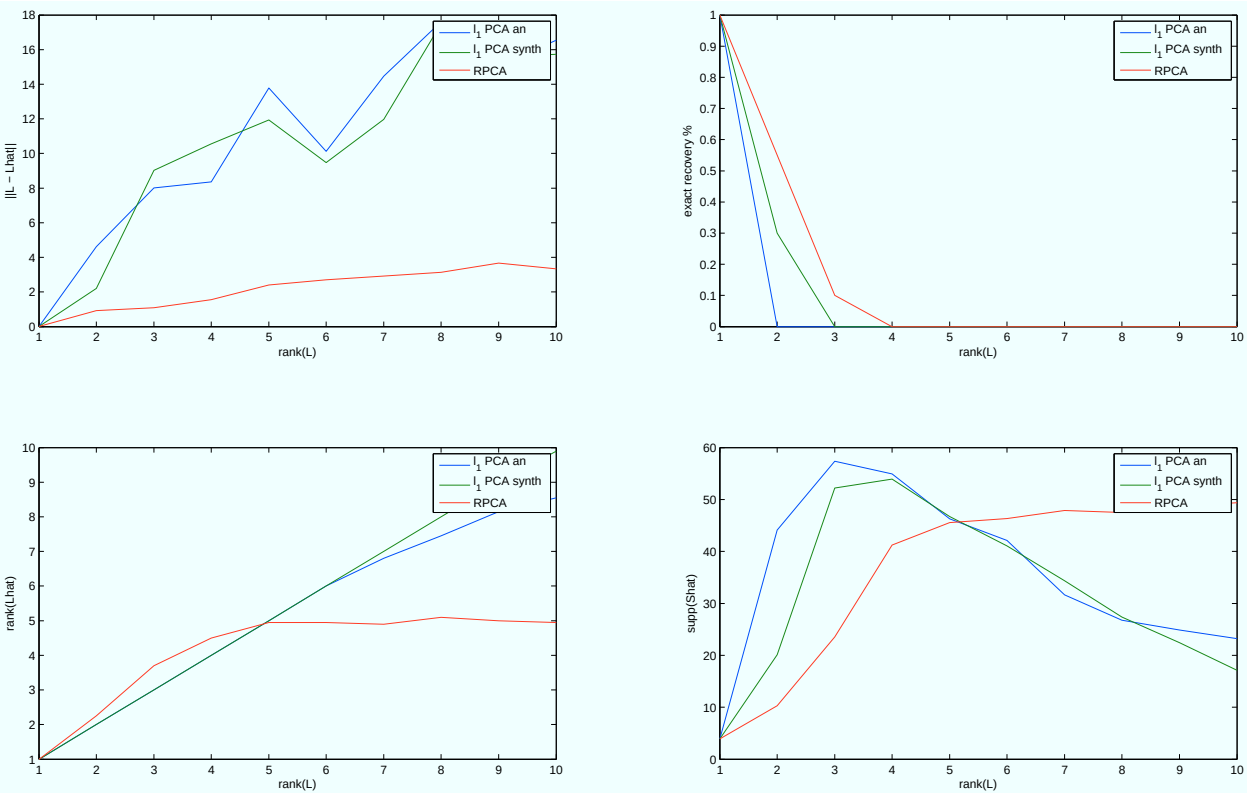


Figure 1.7: Analysis ℓ_1 PCA (blue), synthesis ℓ_1 PCA (green), and Robust PCA (red) for fixed noise support (20), by increasing rank.

Chapter 2

Algorithms

2.1 Overview

This section discusses some of the algorithms that have recently been proposed in the literature to solve the Robust PCA problem

$$\begin{aligned} p^* = \min_{L,S} \; & \|L\|_* + \lambda \|S\|_1 \\ \text{s.t.} \quad & M = L + S \end{aligned} \tag{2.1}$$

There are various methods that can be used to solve (2.1). A straightforward way described in section 2.2.1 is to use general purpose interior point solvers [28, 30] to solve an SDP formulation of the dual of (2.1), an approach that works well for low-dimensional data M but unfortunately does not scale well. Another approach is to use iterative thresholding techniques as described in section 2.2.2, which result in a very simple algorithm that can be applied also to high-dimensional data. Unfortunately the convergence of this algorithm is extremely slow. More sophisticated approaches include an accelerated proximal gradient algorithm (section 2.2.3) and a gradient ascent algorithm applied to the dual problem of (2.1) (section 2.2.4). The current state of the art seems to be a adaptation of the Augmented Lagrangian Method to the non-smooth problem (2.1), an approach that is discussed in section 2.2.5.

2.2 Main algorithms for Robust PCA

2.2.1 Interior Point Methods

As will be shown in the following section, the dual problem of (2.1) can be cast as a Semidefinite Programming (SDP) problem for which a number of efficient polynomial-time interior-point solvers have been developed. In principle, one can then just apply these general purpose off-the-shelf solvers to the dual SDP. This approach will work well for small and medium sized problems, but unfortunately it does not scale with the size of the data matrix M. The following section first develops the dual of the Robust PCA problem, section 2.2.1 then discusses the limitations of using interior point methods on the resulting SDP.

Formulation of the Dual problem of Robust PCA as an SDP

Let us use the equality constraint to eliminate the variable L from (2.1). Using the facts that $\|X\|_* = \max_{\|Y\|_2 \leq 1} \mathbf{Tr}\, Y^T X$ and $\|X\|_1 = \max_{\|Z\|_\infty \leq 1} \mathbf{Tr}\, Z^T X$ the problem can be written as

$$\begin{aligned} p^* = \min_{S} \max_{Y,Z} \quad & \mathbf{Tr}\, Y^T(M-S) + \lambda\, \mathbf{Tr}\, Z^T S \\ \text{s.t.} \quad & \|Y\|_2 \leq 1 \\ & \|Z\|_\infty \leq 1 \end{aligned} \tag{2.2}$$

The dual function is

$$\begin{aligned} g(Y,Z) &= \min_{S} \quad \mathbf{Tr}\, Y^T M + \mathbf{Tr}(\lambda Z - Y)^T S \\ &= \begin{cases} \mathbf{Tr}\, Y^T M & \text{if } \lambda Z = Y \\ -\infty & \text{otherwise} \end{cases} \end{aligned} \tag{2.3}$$

We obtain the dual problem

$$\begin{aligned} p^* \geq d^* = \max_{Y,Z} \quad & \mathbf{Tr}\, Y^T M \\ \text{s.t.} \quad & \lambda Z = Y \\ & \|Y\|_2 \leq 1 \\ & \|Z\|_\infty \leq 1 \end{aligned} \tag{2.4}$$

which after eliminating the variable Z and using homogeneity of the norm becomes

$$\begin{aligned} d^* = \max_{Y} \quad & \mathbf{Tr}\, Y^T M \\ \text{s.t.} \quad & \|Y\|_2 \leq 1 \\ & \|Y\|_\infty \leq \lambda \end{aligned} \tag{2.5}$$

Noting that $\|Y\|_2 \leq 1 \iff I - Y^T (I)^{-1} Y \succeq 0$ and using a Schur Complement lemma, the dual problem can be transformed into the following SDP standard form:

$$\begin{aligned} d^* = \max_{Y} \quad & \mathbf{Tr}\, M^T Y \\ \text{s.t.} \quad & \begin{bmatrix} I & Y^T \\ Y & I \end{bmatrix} \succeq 0 \\ & \mathbf{Tr}\, \Delta_{ij}^T Y \leq \lambda, \quad i = 1, \ldots, m,\ j = 1, \ldots, n \\ & \mathbf{Tr}\, \Delta_{ij}^T Y \geq -\lambda, \quad i = 1, \ldots, m,\ j = 1, \ldots, n \end{aligned} \tag{2.6}$$

Here $\Delta_{ij} \in \mathbf{R}^{m \times n}$ is such that $\Delta_{ij}(k,l) = \delta_{ik}\delta_{jl}$. Note that both primal and dual problem are (trivially) strictly feasible. In fact, the primal is unconstrained and an obvious strictly feasible dual variable is $Y = 0$. Hence strong duality holds ($p^* = d^*$) and both primal and dual problem are attained.

Recovering the primal solution

To recover the primal solution we can distinguish the two cases $\lambda \leq 1$ and $\lambda > 1$. For the latter one, first note that $\|Y\|_\infty \leq \|Y\|_2$ for all Y. To show this first recall that $c := \|Y\|_2 = \max_{\|x\|_2=1} \|Yx\|_2$. This means that $c\|x\|_2^2 - x^T Y^T Y x \geq 0, \forall\, x$ or, equivalently, $cI - Y^T Y \succeq 0$. Using a Schur Complement this can be written as

$$\begin{bmatrix} I & Y \\ Y^T & cI \end{bmatrix} \succeq 0$$

It is easy to see that this implies that

$$\begin{bmatrix} 1 & Y_{ij} \\ Y_{ij} & c \end{bmatrix} \succeq 0$$

for all i, j, which is equivalent to $|Y_{ij}| \leq c,\ \forall\, i, j$. This shows that $\|Y\|_\infty \leq \|Y\|_2$.

Hence we observe that if $\lambda > 1$ the constraint $\|Y\|_2 \leq \lambda$ will be inactive, that is $|Y_{ij}| < \lambda,\ \forall\, i, j$. Therefore, for $\lambda > 1$, removing this constraint from (2.5) does not change the problem. We have $d^* = \max_{\|Y\|_2 \leq 1} \mathbf{Tr}\, Y^T M = \|M\|_*$, and this lower bound is achieved by the primal (feasible) solution $L = M$ and $S = 0$.

If $\lambda \leq 1$, then we can look at the dual of the dual (2.5). One way to form a Lagrangian of this dual problem is to introduce Lagrange multipliers $S_- \geq 0$ and $S_+ \geq 0$ for the constraints $Y_{ij} \leq \lambda$ and $-Y_{ij} \leq \lambda$, respectively:

$$\begin{aligned} \mathcal{L}(Y, S_+, S_-) &= -\mathbf{Tr}\, Y^T M - \mathbf{Tr}\, S_-^T (Y - \lambda 1) - \mathbf{Tr}\, S_+^T (-Y - \lambda 1) \\ &= \mathbf{Tr}(S_+ - S_- - M)^T Y + \lambda\, \mathbf{Tr}(S_+ + S_-)^T 1 \end{aligned}$$

Here 1 denotes the $n \times n$ matrix whose entries are all ones. We have

$$\begin{aligned} d^* &= \max_{\substack{\|Y\|_2 \leq 1 \\ \|Y\|_\infty \leq \lambda}} -\mathbf{Tr}\, M^T Y \\ &= \max_{\|Y\|_2 \leq 1} \min_{\substack{S_+ \geq 0 \\ S_- \geq 0}} \mathcal{L}(Y, S_+, S_-) \\ &= \min_{\substack{S_+ \geq 0 \\ S_- \geq 0}} \max_{\|Y\|_2 \leq 1} \mathcal{L}(Y, S_+, S_-) \\ &= \min_{\substack{S_+ \geq 0 \\ S_- \geq 0}} g(S_+, S_-) \end{aligned}$$

where

$$\begin{aligned} g(S_+, S_-) &= \max_{\|Y\|_2 \leq 1} \mathbf{Tr}(S_+ - S_- - M)^T Y + \lambda\, \mathbf{Tr}(S_+ + S_-)^T 1 \\ &= \|S_+ - S_- - M\|_* + \lambda\, \mathbf{Tr}(S_+ + S_-)^T 1 \end{aligned}$$

A dual problem of the dual (2.5) is then simply

$$\min_{S_+ \geq 0, S_- \geq 0} \|S_+ - S_- - M\|_* + \lambda\, \mathbf{Tr}(S_+ + S_-)^T 1 \tag{2.7}$$

We have $d^* = \min_{S_+ \geq 0, S_- \geq 0} g(S_+, S_-)$. Note that at the optimum, the variables S_+ and S_- satisfy $\min(S_{+ij}, S_{-ij}) = 0\ \forall i, j$. Indeed, suppose there existed (i, j) such that $\min(S_{+ij}, S_{-ij}) = a > 0$,

then $g(S_+ - ae_ie_j^T, S_- - ae_ie_j^T) = g(S_+, S_-) - 2\lambda a < g(S_+, S_-)$, and (S_+, S_-) is not optimal. Letting $S = S_+ - S_-$, the problem is equivalent to the primal

$$p^* = \min_S \|M - S\|_* + \lambda \|S\|_1$$

Therefore it is possible to recover the optimizer S^* from the dual problem: Indeed, we have that $S^* = S_+^* - S_-^*$ where S_- and S_+ are respectively the Lagrange multipliers associated with the constraints $Y_{ij} \leq \lambda$ and $-Y_{ij} \leq \lambda$

Using Interior Point Methods to Solve the Dual

Problem (2.5) is an SDP and can therefore in principle be solved using off-the-shelf solver packages such as `sedum` [28] and `sdpt3` [30]. These solvers are based on interior point methods which offer superior convergence rates [6] but unfortunately do not scale well at all with the size of the matrix M. Figure 2.1 shows the average running time over 10 instances of the dual problem for the solvers `sedumi` and `sdpt3` for randomly generated data of very low dimensions. There is a

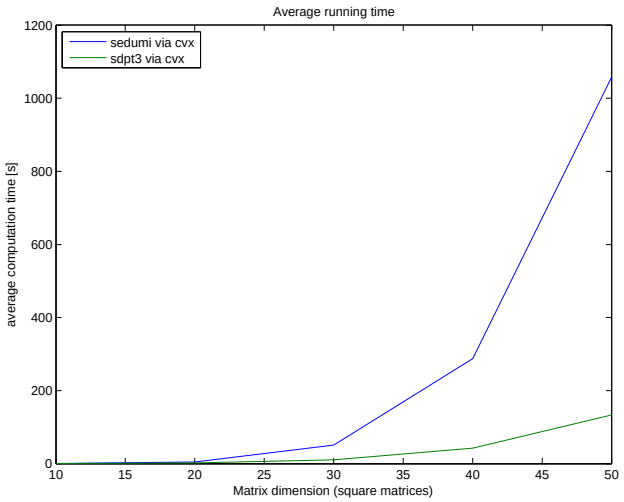


Figure 2.1: Average running time of interior point methods for the dual problem

surprisingly big difference in terms of the running times: `sdpt3` is consistently faster than `sedumi`, with the spread growing significantly with the problem size. For the largest dimension `sdpt3` on average is 8 times faster than `sedumi` for this particular problem type. To be able to give a realistic assessment of why this is the case would however require detailed knowledge of the solver implementation.

Regardless of which solver is used, it is obvious that, with an average running time of the faster of the two solvers of more than 2 minutes even on toy-sized problems involving matrices with only 2500 entries, interior point solvers are unsuitable for most problems of interest. The reason for this extremely bad scaling behavior with the matrix dimension is that the computation of the Newton step direction relies on second-order information of the (barrier-augmented) objective function, the computation of which is too expensive infeasible when the number of variables is very high. For applications that involve large-scale data matrices (with mn in the millions or even billions), using interior point methods therefore is essentially impossible. In order to overcome this scalability issue,

a variety of first-order methods exploiting the particular structure of the Robust PCA problem have been proposed. Some of these methods will be described in the following.

2.2.2 Iterative Thresholding Method

Among the first techniques used to solve (2.1) for high-dimensional data was an adaptation [31][1] of an iterative thresholding method originally proposed for the matrix completion problem in [7]. For this method the authors consider a relaxation of the original problem (2.1) of the form

$$\begin{aligned} \min_{L,S} \ & \|L\|_* + \lambda\|S\|_1 + \frac{1}{2\tau}\|L\|_F^2 + \frac{1}{2\tau}\|S\|_F^2 \\ \text{s.t.} \ & M = L + S \end{aligned} \tag{2.8}$$

where $\tau \gg 1$ is a scalar parameter. Generalizing the result from [7] (which considers the matrix completion problem, i.e. $S = 0$), the authors of [31] argue that for large values of τ the solution of (2.8) will be very close to that of (2.1). One can now employ iterative thresholding techniques to solve (2.8). Although the resulting algorithm has little relevance in practice because of its extremely slow convergence speed, we will see in the following sections that its main ideas are the basis for a number of other similar algorithms. Therefore we briefly review the iterative thresholding algorithm here.

The Lagrangian of the problem (2.8) is given by

$$\mathcal{L}(L, S, Y) = \|L\|_* + \lambda\|S\|_1 + \frac{1}{2\tau}\|L\|_F^2 + \frac{1}{2\tau}\|S\|_F^2 + \frac{1}{\tau}\langle Y, M - L - S\rangle \tag{2.9}$$

The idea of the iterative thresholding algorithm is, as the name suggests, to update the variables L, S and Y iteratively. More specifically, the Lagrangian $\mathcal{L}(L, S, Y)$ is minimized w.r.t L and S for some fixed dual variable Y, and the violation of the constraint is then used to update Y using the gradient step $Y^+ = Y + t(M - L - S)$, where $0 < t < 1$ is the step size.

Note that for fixed Y we have

$$\begin{aligned} (\hat{L}, \hat{S}) :&= \arg\min_{L,S} \ \mathcal{L}(L, S, Y) \\ &= \arg\min_{L,S} \ \|L\|_* + \lambda\|S\|_1 + \frac{1}{2\tau}\|L\|_F^2 + \frac{1}{2\tau}\|S\|_F^2 + \frac{1}{\tau}\langle Y, M - L - S\rangle \\ &= \arg\min_{L,S} \ \|L\|_* + \lambda\|S\|_1 + \frac{1}{2\tau}\|L - Y\|_F^2 + \frac{1}{2\tau}\|S - Y\|_F^2 \end{aligned} \tag{2.10}$$

We note that the above problem is completely separable hence the minimizers $\hat{L}$ and $\hat{S}$ can be determined independently. Using optimality conditions based on the subgradient it can be shown [7, 31] that the minimizers $\hat{L}$ and $\hat{S}$ are both of a simple form. To this end, consider the following extension to the matrix case of the well known soft-thresholding operator (e.g. from the proximal

[1] note that the iterative thresholding algorithm seems to never have appeared in the published version of [31], which instead contains the accelerated proximal gradient method described in section 2.2.3

mapping of the l_1-norm in the vector case):

$$(\mathcal{S}_\varepsilon[X])_{ij} := \begin{cases} x_{ij} - \varepsilon & \text{if } x_{ij} > \varepsilon \\ x_{ij} + \varepsilon & \text{if } x_{ij} < -\varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

Theorem 7 (Singular Value Thresholding [7])**.** *For each $\tau \geq 0$ and $Y \in \mathbf{R}^{m\times n}$, we have that*

$$\mathcal{D}_\tau(Y) := U\mathcal{S}_\tau[\Sigma]V^T = \arg\min_X \ \|X\|_* + \frac{1}{2\tau}\|X - Y\|_F^2 \tag{2.12}$$

where $Y = U\Sigma V^*$ is the SVD of Y.

Since Theorem 7 is of fundamental importance to all algorithms discussed in this chapter, we give its proof here.

Proof of Theorem 7. Since the objective function $f_0(X) := \arg\min_X \ \|X\|_* + \frac{1}{2\tau}\|X - Y\|_F^2$ is strictly convex its minimizer is unique. Recall that Z is a subgradient of a convex function $f : \mathbf{R}^{m\times n} \mapsto \mathbf{R}$ at X_0 if $f(X) \geq f(X_0) + \langle Z, X - X_0\rangle$ for all X. The set of all sugradients Z at X_0 is called the subdifferential and denoted $\partial f(X_0)$. By definition it follows that $\hat{X}$ minimizes f_0 if and only if 0 is a subgradient of f_0 at $\hat{X}$. Using standard subgradient calculus this translates to $0 \in \hat{X} - Y + \tau\partial||\hat{X}||_*$, where $\partial||\hat{X}||_*$ is the subdifferential of the nuclear norm at $\hat{X}$. To show that $\mathcal{D}_\tau(Y)$ satisfies (2.12) we therefore need to show that $0 \in \mathcal{D}_\tau(Y) - Y + \tau\partial||\mathcal{D}_\tau(Y)||_*$.

Let $X \in \mathbf{R}^{m\times n}$ be arbitrary. Let $U\Sigma V^*$ be its SVD. It can be shown that

$$\partial||X||_* = \left\{UV^* + W \ \middle|\ W \in \mathbf{R}^{m\times n},\ U^*W = 0,\ WV = 0,\ \|W\|_2 \leq 1\right\}$$

To use the above result decompose the SVD of Y as $Y = U_0\Sigma_0V_0^* + U_1\Sigma_1V_1^*$, where U_0, V_0 (U_1, V_1) are the singular vectors associated with singular values greater (less or equal) than τ. With this notation we have $\mathcal{D}_\tau(Y) = U_0(\Sigma_0 - \tau I)V_0^*$, so $Y - \mathcal{D}_\tau(Y) = \tau(U_0V_0^* + W)$ with $W = \tau^{-1}U_1\Sigma_1V_1^*$. By definition of the SVD $U_0^*W = 0$ and $WV_0 = 0$. Furthermore, since $\max(\Sigma_1) \leq \tau$ it also holds that $\|W\|_2 \leq 1$. Together this shows that $Y - \mathcal{D}_\tau(Y) \in \tau\partial||\mathcal{D}_\tau(Y)||_*$. □

It is also not hard to find the minimizer $\hat{S}$ of (2.9), as the following proposition shows:

Proposition 1 (Matrix Value Thresholding)**.** *For each $\tau, \lambda \geq 0$ and $Y \in \mathbf{R}^{m\times n}$, we have that*

$$\mathcal{S}_{\tau\lambda}[Y] = \arg\min_X \ \lambda\|X\|_1 + \frac{1}{2\tau}\|X - Y\|_F^2 \tag{2.13}$$

Proof of Proposition 1. Note that the objective function $h_0(X) := \lambda\|X\|_1 + \frac{1}{2\tau}\|X-Y\|_F^2$ in (2.13) is completely decomposable and can be written in the form $h_0(X) = \sum_{i,j} \lambda|X_{ij}| + \frac{1}{2\tau}(X_{ij}-Y_{ij})^2$, so the minimization over X can be carried out element-wise. Using the same reasoning as in the proof of Theorem 7, we have that the element X_{ij} is a minimizer if and only if $0 \in \partial h_0(X)$. This condition can be written as $Y_{ij} - X_{ij} \in \tau\lambda\,\partial|X_{ij}|$. We consider the cases $X_{ij} = 0$ and $X_{ij} \neq 0$. If $X_{ij} = 0$ then the condition reads $Y_{ij} \in \tau\gamma[-1, 1]$. In other words, when $|Y_{ij}| \leq \tau\gamma$ then $X_{ij} = 0$. On the other hand, if $X_{ij} \neq 0$, then $\partial\,|X_{ij}| = \operatorname{sign}(X_{ij})$ and we have $X_{ij} = Y_{ij} - \tau\lambda\operatorname{sign}(Y_{ij})$, which is in fact the soft-thresholding operator $\mathcal{S}_{\tau\lambda}[X_{ij}]$. □

Applying Theorem 7 and Proposition 1 to (2.10) we find that the minimizers $\hat{L}$ and $\hat{S}$ of (2.9) are given, respectively, by

$$\hat{L} = U\mathcal{S}_\tau[\Sigma]V^T \tag{2.14}$$

$$\hat{S} = \mathcal{S}_{\tau\lambda}[Y] \tag{2.15}$$

Algorithm 6 summarizes the overall iterative thresholding algorithm for Robust PCA.

Algorithm 6: Iterative Thresholding Algorithm

Input: Observation matrix M, parameters λ, τ
initialization: $k = 0$, $Y_0 = 0$;
while *not converged* **do**
  $k = k + 1$;
  $(U, \Sigma, V) = \text{svd}(Y_{k-1})$;
  $L_k = U\mathcal{S}_\tau[\Sigma]V^T$;
  $S_k = \mathcal{S}_{\lambda\tau}[Y_{k-1}]$;
  $Y_k = Y_{k-1} + t_k(M - L_k - S_k)$;
Output: $L = L_k$, $S = S_k$

Algorithm 6 is extremely simple to implement, as each iteration only requires the computation of an SVD of the current dual variable Y_k and a few elementary matrix operations. Note that, in fact, since the shrinking operator sets all singular values less than the threshold parameter to zero one really only needs to compute the singular values that lie above this threshold. In section 2.3.1 we will discuss how this can be exploited to achieve potential speedups in all of the algorithms considered in this chapter. The convergence criterion used in practice is usually of the form $\|M - L_k - S_k\|_F < \varepsilon\|M\|_F$, where ε is the desired relative accuracy.

While the iterative thresholding scheme in Algorithm 6 is very simple and has been proved to converge, its convergence speed unfortunately is extremely slow. The authors of [31] found that it typically requires about 10^4 iterations to converge for problems of reasonable size, an observation that is confirmed by our numerical experiments. Furthermore it is hard to find good general schemes to optimize the choice of the step size t_k [20]. As it turns out the other algorithms discussed in the following sections have a comparable complexity but converge much faster, both in theory and practice. Therefore the practical applicability of this approach is limited.

2.2.3 Accelerated Proximal Gradient Method

An accelerated proximal gradient (APG) method for solving (2.1) was proposed in [21]. The method is essentially an application of the FISTA algorithm [3] to a relaxation of the original RPCA problem in combination with a continuation technique. This FISTA algorithm algorithm is reminiscent of Nesterov's "optimal" first-oder method for smooth objective functions [24], whose $O(1/\varepsilon)$ convergence result was extended to non-smooth objectives in [25]. Besides being theoretically appealing, the APG method is also in practice much faster than the iterative thresholding algorithm discussed in section 2.2.2.

The following sections describe the APG algorithm. As in [21] we first give a general formulation and then show how it can be applied to the Robust PCA problem, using ideas from section 2.2.2.

A General Form of the Accelerated Proximal Gradient Method

Proximal gradient algorithms in general can be used to solve unconstrained problems of the form

$$\min_x \ f(x) := g(x) + h(x) \tag{2.16}$$

where g is convex and differentiable and h is closed[2], convex but need not be differentiable. The proximal mapping of a convex function h is defined as

$$\text{prox}_h(x) := \arg\min_u \left(h(u) + \frac{1}{2}\|u - x\|^2 \right) \tag{2.17}$$

Here $\|\cdot\|$ denotes the inner product norm[3]. Note that since h is convex and the norm is strictly convex, the optimizer in (2.17) is unique. The step of each iteration of the classic (non-accelerated) proximal gradient algorithm is

$$x^+ = \text{prox}_{\delta h}(x - \delta \nabla g(x)) \tag{2.18}$$

where x^+ denotes the next iterate, based on the current iterate x, and δ is the step size (which can be fixed or determined via backtracking line search). Since the proximal mapping (2.18) needs to be computed at each iteration (possibly multiple times because of the line search), the algorithm is most effective when this can be done cheaply. Depending on the function h, the proximal gradient algorithm can be seen as a generalization of different gradient-based algorithms. In particular, for $h(x) \equiv 0$ the standard gradient algorithm is recovered and for $h(x) = I_C(x)$, where I_C denotes the indicator function for the convex set C, the projected gradient algorithm is recovered.

Note that (2.18) can be written as

$$\begin{aligned} x^+ &= \arg\min_u \left(h(u) + \frac{1}{2\delta}\|u - x + \delta \nabla g(x)\|^2 \right) = \arg\min_u \left(h(u) + \frac{1}{2\delta}\|u - G(x)\|^2 \right) \\ &= \arg\min_u \left(h(u) + g(x) + \langle \nabla g(x), u - x \rangle + \frac{1}{2\delta}\|u - x\|^2 \right) \end{aligned} \tag{2.19}$$

where $G(x) := x - \delta \nabla g(x)$. Therefore, each step (2.18) can be interpreted as minimizing the function $h(u)$ plus a quadratic local model of $g(u)$ around the current iterate x. This classic form of the proximal gradient algorithm is well known in the literature and, under Lipschitz continuity of the gradient of g and some technical conditions on the step size t, has been shown to have a convergence rate no worse than $O(1/k)$ [25]. In particular, a popular choice for δ is the fixed step size $\delta = 1/L_g$, where L_g is a Lipschitz constant of the gradient of g, that is we have $\|\nabla g(x) - \nabla g(y)\| \leq L_g \|x - y\|$ for all x, y. Note that in practice for a general problem the value of L_g might be unknown. However, this is not a big concern for the type of problem arising in Robust PCA, hence we will assume this choice of $\delta = 1/L_g$ in the following.

It turns out that the choice for the point around which the quadratic local model of $g(u)$ is constructed has an important effect on the convergence rate, and that less obvious choices than just the previous iterate x are actually better (in the sense that they yield higher convergence rates).

[2] that is, its epigraph is closed

[3] the derivation holds for real inner product spaces, not just $\mathbf{R}^n$. We are particularly interested in the case $\mathbf{R}^{m \times n}$

Specifically, consider a generalized version of (2.19), where the approximation of g is constructed around some y that may depend on all prior iterates $x_0, \ldots, x_k$:

$$x^+ = \arg\min_u \left(h(u) + \frac{L_g}{2} \|u - G(y)\|^2 \right) \tag{2.20}$$

Nesterov in [24] showed that in the smooth case the standard gradient algorithm (i.e. $h(x) \equiv 0$) can be accelerated by choosing $y_k = x_k + \frac{t_{k-1}-1}{t_k}(x_k - x_{k-1})$, leading to a theoretical convergence rate of $O(1/k^2)$. This algorithm is optimal in the sense that its convergence rate achieves (in terms of order) the theoretical complexity bound on all first order algorithms. The seminal work [24] has been extended to the non-smooth case in [3, 25], yielding the generic accelerated proximal gradient method given in Algorithm 7.

Algorithm 7: Accelerated Proximal Gradient Algorithm

initialization: $k = 0$, $t_0 = t_{-1} = 1$;
while *not converged* **do**
  $y_k = x_k + \frac{t_{k-1}-1}{t_k}(x_k - x_{k-1})$;
  $G_k = y_k - \frac{1}{L_g} \nabla g(y_k)$;
  $x_{k+1} = \arg\min_x \left(h(x) + \frac{L_g}{2} \|x - G(y_k)\|^2 \right)$;
  $t_{k+1} = \frac{1+\sqrt{4t_k^2+1}}{2}$;
  $k = k + 1$;

Accelerated Proximal Gradient Algorithm Applied to Robust PCA

Using some the ideas from the iterative thresholding method in section 2.2.2, it is possible to construct an accelerated proximal gradient algorithm for a relaxation of the original problem (2.1). In particular, it is easy to see that in the common case $h(x) = \|x\|_1$ the update for x in Algorithm 7 is simply given by $x_{k+1} = \mathcal{S}_{1/L_g}[G(y_k)]$, where $\mathcal{S}_\varepsilon$ is the soft-thresholding operator defined in (2.11). Similar ideas can be used to develop an APG algorithm for the Robust PCA problem.

The authors of [21] consider a relaxation of (2.1) of the form

$$\min_{L,S} \ \mu\|L\|_* + \mu\lambda\|S\|_1 + \frac{1}{2}\|L + S - M\|_F^2 \tag{2.21}$$

It can be shown that for $\mu \to 0$, any solution of (2.21) approaches the solution set of (2.1). In practice, rather than fixing μ to some small value, one can achieve superior convergence of the algorithm by using a continuation technique on μ, that is, by solving (2.21) by repeatedly decreasing the value of μ in the steps of the accelerated proximal gradient algorithm. This can be interpreted as a particular homotopy method.

With $X = (L, S)$ we identify $h(X) = \mu\|L\|_* + \mu\lambda\|S\|_1$ and $g(X) = \frac{1}{2}\|L + S - M\|_F^2$ in (2.21). The gradient of g is $\nabla g(X) = (\nabla_L g(X), \nabla_S g(X))$, where $\nabla_L g(X) = \nabla_S g(X) = L + S - M$. Thus

$$\begin{aligned}
\|\nabla g(X_1) - \nabla g(X_2)\| &= \left\| \begin{bmatrix} L_1 + S_1 - M \\ L_1 + S_1 - M \end{bmatrix} - \begin{bmatrix} L_2 + S_2 - M \\ L_2 + S_2 - M \end{bmatrix} \right\| = \left\| \begin{bmatrix} L_1 - L_2 + S_1 - S_2 \\ L_1 - L_2 + S_1 - S_2 \end{bmatrix} \right\| \\
&\leq \left\| \begin{bmatrix} L_1 - L_2 \\ S_1 - S_2 \end{bmatrix} \right\| + \left\| \begin{bmatrix} S_1 - S_2 \\ L_1 - L_2 \end{bmatrix} \right\| = 2\,\|X_1 - X_2\|
\end{aligned}$$

which means that a Lipschitz constant is given by $L_g = 2$. It turns out that because of the separability of both the function h and the Frobenius norm the update can be be decomposed as follows:

$$\begin{aligned}(L_{k+1}, S_{k+1}) = X_{k+1} &= \arg\min_X \left(h(X) + \frac{L_g}{2}\|X - G(Y_k)\|^2 \right) \\ &= \arg\min_{L,S} \left(\mu\|L\|_* + \mu\lambda\|S\|_1 + \left\| \begin{bmatrix} L \\ S \end{bmatrix} - \begin{bmatrix} G_k^L \\ G_k^S \end{bmatrix} \right\|^2 \right) \\ &= \arg\min_{L,S} \left(\|L\|_* + \frac{1}{2\frac{\mu}{2}}\left\|L - G_k^L\right\|^2 + \lambda\|S\|_1 + \frac{1}{2\frac{\mu}{2}}\left\|S - G_k^S\right\|^2 \right)\end{aligned}$$

The problem is therefore completely separable and clearly the sum of two problems of the form (2.16). In particular, if $G_k^L = U\Sigma V^T$ is the SVD of G_k^L then, using the results from section 2.2.2, the iterates L_{k+1} and S_{k+1} are given by

$$L_{k+1} = U\mathcal{S}_{\frac{\mu}{2}}[\Sigma]V^T \tag{2.22}$$

$$S_{k+1} = \mathcal{S}_{\frac{\lambda\mu}{2}}[G_k^S] \tag{2.23}$$

Therefore it is rather straightforward to formulate a version of Algorithm 7 for the Robust PCA problem. One extension to the basic form of Algorithm 7 is the use of a continuation technique (or homotopy method) for the parameter μ. In [21], the authors propose to start from some large value μ_0 and then choose $\mu_{k+1} = \max(\eta\mu_k, \bar{\mu})$, where $0 < \eta < 1$ and $\bar{\mu}$ is a lower limit on μ_k. This continuation technique has been observed to improve convergence, and is used much for the same reasons as the one in modern interior point methods. The Accelerated Proximal Gradient Algorithm applied to the Robust PCA problem is summarized in Algorithm 8.

Algorithm 8: Accelerated Proximal Gradient Algorithm for Robust PCA

Input: Observation matrix M, parameter λ

initialization: $k = 0$, $L_0 = L_{-1} = 0$, $S_0 = S_{-1} = 0$, $t_0 = t_{-1} = 1$, $\bar{\mu} = \delta\mu_0$;

while *not converged* **do**

$Y_k^L = L_k + \frac{t_{k-1}-1}{t_k}(L_k - L_{k-1})$, $Y_k^S = S_k + \frac{t_{k-1}-1}{t_k}(S_k - S_{k-1})$;

$G_k^L = Y_k^L - \frac{1}{2}(Y_k^L + Y_k^S - M)$;

$(U, \Sigma, V) = \text{svd}(G_k^L)$, $L_{k+1} = U\mathcal{S}_{\mu_k/2}[\Sigma]V^T$;

$G_k^S = Y_k^S - \frac{1}{2}(Y_k^L + Y_k^S - M)$;

$S_{k+1} = \mathcal{S}_{\lambda\mu_k/2}[G_k^S]$;

$t_{k+1} = \frac{1+\sqrt{4t_k^2+1}}{2}$;

$\mu_{k+1} = \max(\eta\mu_k, \bar{\mu})$;

$k = k + 1$;

Output: $L = L_k$, $S = S_k$

We will restate the main convergence theorem for Algorithm 8 without proof.

Theorem 8 ([20])**.** *Let* $F(X) = F(L, S) := \bar{\mu}\|L\|_* + \bar{\mu}\lambda\|S\|_1 + \frac{1}{2}\|L + S - M\|_F^2$. *Then for all* $k > k_0 := -C_1 \log(\eta)$ *it holds that*

$$F(X_k) - F(X^*) \leq \frac{4\|C_{k_0} - X^*\|_F^2}{(k - k_0 + 1)^2}$$

where $C_1 = \log(\mu_0/\bar{\mu})$ *and* X^* *is any solution to* (2.21).

Remark 2.2.1 (Accuracy of the APG method). Note that for any finite $\bar{\mu}$ Algorithm 8 provides only an approximate solution of the Robust PCA problem due to the replacement of the equality constraint with the penalty term $\frac{1}{2\bar{\mu}}\|L + S - M\|_F^2$. In practice one therefore has to tradeoff accuracy ($\bar{\mu}$ extremely small) and computational efficiency.

2.2.4 Gradient Ascent on the Dual

In section 2.2.1 we discussed why solving the Robust PCA dual problem (2.5) via interior point techniques quickly becomes infeasible with growing size of the data matrix M. Another way of solving the dual given in [21] is based on a steepest ascent algorithm. Note that the dual problem (2.5) derived in section 2.2.1 can be written as

$$d^* = \max_Y \ \mathbf{Tr}\, M^T Y \quad \text{subject to} \quad J(Y) \leq 1 \tag{2.24}$$

where $J(Y) := \max\left(\|Y\|_2, \lambda^{-1}\|Y\|_\infty\right)$. Denote by $F := \{Y \mid J(Y) \leq 1\}$ the feasible set. The maximum operator in the function J implies that $F = \{Y \mid \|Y\|_2 \leq 1\} \cap \{Y \mid \lambda^{-1}\|Y\|_\infty \leq 1\}$, hence F is clearly convex, being the intersection of two convex sets. Further, the function $J(Y)$ is positive and homogenous in Y, hence the maximum over a linear of the objective function is achieved on the boundary ∂F of the feasible set, that is, the optimum is achieved when $J(Y) = 1$.

To solve (2.24), the authors in [21] propose to use a projected gradient algorithm. The gradient of the objective function $\mathbf{Tr}\, M^T Y$ is simply M. The projection onto the constraint set is more involved but can be treated using standard methods from convex optimization. In particular, the algorithm at each iteration k involves projecting the gradient M onto the tangent cone $T_F(Y_k)$ of the feasible set F at the point Y_k. If W_k is the steepest ascent direction obtained from this projection, the iterate can be updated using the (projected) gradient step

$$Y_{k+1} = \frac{Y_k + t_k W_k}{J(Y_k + t_k W_k)} \tag{2.25}$$

where t_k is the step size that can be determined by a simple line search of the form

$$t_k = \arg\max_{t \geq 0} \left\langle M, \frac{Y_k + t W_k}{J(Y_k + t W_k)} \right\rangle \tag{2.26}$$

Note that the division by $J(Y_k + t_k W_k)$ ensures that the next iterate Y_{k+1} lies on ∂F. It is shown in [21] that if the maximizing step size t_k is equal to zero at some point Y_k, then Y_k is the optimal solution to the dual problem (2.24).

In order to perform the projection on the tangent cone T_F of F at the point Y_k, the authors of [21] make use the following two facts:

1. For any convex subset $K \subseteq V$ of a real vector space V the normal cone $N_K(x)$ at the point $x \in K$ is the polar cone to the tangent cone $T_K(x)$.
2. If two cones C_1 and C_2 in V are are polar cones to each other and $\mathcal{P}_{C_1}$ and $\mathcal{P}_{C_2}$ are the projection operators onto C_1 and C_2, respectively, then $\mathcal{P}_{C_1}(X) + \mathcal{P}_{C_1}(X) = X$ for any point $X \in V$.

From these facts, it follows immediately that $\mathcal{P}_{T_F(Y_k)}(M)$, the projection of M onto the dual cone of F at Y_k, can be computed as $\mathcal{P}_{T_F(Y_k)}(M) = M - M_k$, where $M_k := \mathcal{P}_{N_F(Y_k)}(M)$. It can be shown that the normal cone is characterized by the subgradient of the function J via $N(Y_k) = \{\alpha X \mid \alpha \geq 0, X \in \partial J(Y_k)\}$. Note that J is in fact the pointwise maximum over two functions, hence from strong subgradient calculus it follows that

$$\partial J(Y_k) = \begin{cases} \partial \|Y_k\|_2 & \text{if} \quad \|Y_k\|_2 > \lambda^{-1}\|Y_k\|_\infty \\ \lambda^{-1}\partial \|Y_k\|_\infty & \text{if} \quad \|Y_k\|_2 < \lambda^{-1}\|Y_k\|_\infty \\ \text{Conv}\{\partial \|Y_k\|_2 \,,\, \lambda^{-1}\partial \|Y_k\|_\infty\} & \text{if} \quad \|Y_k\|_2 = \lambda^{-1}\|Y_k\|_\infty \end{cases} \tag{2.27}$$

where $\|\cdot\|_\infty$ is the maximum absolute value of the matrix entries (not the induced ∞-norm).

The first two cases are rather simple and the associated projections $\mathcal{P}_2(\cdot)$ and $\mathcal{P}_\infty(\cdot)$ onto the normal cones generated by the sub gradients of $\|\cdot\|_2$ and $\|\cdot\|_\infty$ at Y_k, respectively, are efficiently computable [21]. In particular, at each step the projection $\mathcal{P}_2(\cdot)$ requires the computation of the largest singular value[4] (and associated singular vectors) of the matrix Y_k, and the projection $\mathcal{P}_\infty(\cdot)$ requires only a simple element-wise comparison of the matrices M and Y_k. The projection in the third case, i.e. when $N(Y_k) = N_2(Y_k) + N_\infty(Y_k)$, is more involved and can be performed using an alternating projections algorithm. It can be shown [21] that by initializing $S_0 = 0$ and $j = 0$, and then repeatedly setting $L_{j+1} = \mathcal{P}_2(M - S_j)$, $S_{j+1} = \mathcal{P}_\infty(M - L_{j+1})$ will yield the projection $M_k = \mathcal{P}_{N(Y_k)}(M)$ in the sense that $\lim_{j\to\infty} L_j + S_j = M_k$. The projected gradient ascent algorithm for the dual problem is given in Algortihm (9).

Algorithm 9: Projected Gradient Ascent for the Dual Problem

Input: Observation matrix M, parameter λ
initialization: $k = 0$, $Y_0 = \text{sign}(M)/J(M)$;
while *not converged* **do**
  Compute the projection M_k of M onto $N_F(Y_k)$:
  if $\|Y_k\|_2 > \lambda^{-1}\|Y_k\|_\infty$ **then**
    $M_k = \mathcal{P}_2(M)$, $L = M$, $S = 0$
  else if $\|Y_k\|_2 < \lambda^{-1}\|Y_k\|_\infty$ **then**
    $M_k = \mathcal{P}_\infty(M)$, $L = 0$, $S = M$
  else
    $L = 0$, $S = 0$;
    while *not converged* **do**
      $L = \mathcal{P}_2(M - S)$, $S = \mathcal{P}_\infty(M - L)$
    $M_k = L + S$
  Perform a line search as in (2.26) to determine step size t_k;
  $Y_{k+1} = \frac{Y_k + t_k(M - M_k)}{J(Y_k + t_k(M - M_k))}$;
  $k = k + 1$;
Output: L, S

Using the properties of dual norms, it is possible to show [21] that the primal solution $(\hat{L}, \hat{S})$ can easily be recovered from the dual optimal $\hat{Y}$. Specifically, it can be shown that if $\|Y_k\|_2 < 1$ or $\lambda^{-1}\|Y_k\|_\infty < 1$ the solution is degenerate and given by $\hat{L} = 0, \hat{S} = M$ (in case $\|Y_k\|_2 < 1$) or

[4]Note that the largest singular value of Y_k needs to be computed anyway at each step in order to distinguish the three cases in (2.27)

$\hat{L} = M, \hat{S} = 0$ (in case $\lambda^{-1}\|Y_k\|_\infty < 1$). If $\|Y_k\|_2 = \lambda^{-1}\|Y_k\|_\infty = 1$, then $(\hat{L}, \hat{S})$ is any pair of points that achieves convergence of the alternating projections method described above.

Algortihm (9) at heart is really just a simple projected gradient ascent algorithm, the convergence rate of which is known to be $O(1/k)$ (in terms of outer iterations, not counting the alternating projection sub-algorithm). The most costly operations are the projections $\mathcal{P}_2(\cdot)$ that require the computation of the largest singular value of matrices of the same size as M. As our numerical simulations show, while faster than naive application of interior point methods or the iterative thresholding algorithm, Algortihm (9) in general performs much worse than the Augmented Lagrangian methods discussed in the following sections.

2.2.5 Augmented Lagrangian Method

The General Case

The classic Augmented Lagrangian Method (ALM), described in [5], is a method for solving equality-constrained convex optimization problems. Consider a generic equality-constrained optimization problem of the form

$$p^* = \min_{x \in \mathcal{H}} f_o(x) \quad \text{s.t.} \quad f_c(x) = 0 \tag{2.28}$$

where $\mathcal{H}$ and $\mathcal{H}'$ are Hilbert spaces and $f_o : \mathcal{H} \mapsto \mathbf{R}$ and $f_c : \mathcal{H} \mapsto \mathcal{H}'$ are both convex functions. The conventional Lagrangian of the problem is $\mathcal{L}(x, \lambda) = f_o(x) + \langle \lambda, f_c(x) \rangle$, with $\lambda \in \mathcal{H}'$ being the dual variable. The augmented Lagrangian method, as its name suggests, uses an augmented Lagrangian of the form

$$\mathcal{L}(x, \lambda, \mu) = f_o(x) + \langle \lambda, f_c(x) \rangle + \frac{\mu}{2}\|f_c(x)\|^2 \tag{2.29}$$

where $\|\cdot\|$ is the inner product norm. Here λ can now be interpreted as an estimate of a dual variable. The general algorithm is quite simple and given in Algorithm 10.

Algorithm 10: Generic Augmented Lagrangian Method

while *not converged* **do**
  $x_{k+1} = \arg\min_x \mathcal{L}(x, \lambda_k, \mu_k)$;
  $\lambda_{k+1} = \lambda_k + \mu_k f_c(x_{k+1})$;
  update μ_k to μ_{k+1};
  $k = k + 1$;
Output: $X = X_k$

Alternating Directions ALM for Robust PCA

For the Robust PCA problem (2.1) the augmented Lagrangian is (associating $x = (L, S)$)

$$\mathcal{L}(L, S, Y, \mu) = \|L\|_* + \lambda\|S\|_1 + \langle Y, M - L - S \rangle + \frac{\mu}{2}\|M - L - S\|_F^2 \tag{2.30}$$

Note that one of the usual assumptions made when using an augmented Lagrangian method is that the objective function f_o is differentiable. This is clearly not the case for the Robust PCA problem.

However, in [20] the authors show that the same convergence properties as for the differentiable case can be retained also for the Robust PCA problem. The main step in Algorithm 10 is solving the problem $x_{k+1} = \arg\min_x \mathcal{L}(x, \lambda_k, \mu_k)$, which in the case of the Robust PCA problem reads

$$(L_{k+1}, S_{k+1}) = \arg\min_{L,S} \ \|L\|_* + \lambda\|S\|_1 + \langle Y_k, M - L - S\rangle + \frac{\mu_k}{2}\|M - L - S\|_F^2 \tag{2.31}$$

and can be solved using an alternating directions approach based on the ideas in section 2.2.2. Specifically, consider (2.31) for S fixed. The corresponding "directional" subproblem in the variable L can be formulated as

$$\begin{aligned} L_{k+1} &= \arg\min_L \ \|L\|_* + \lambda\|S\|_1 + \langle Y_k, M - L - S\rangle + \frac{\mu_k}{2}\|M - L - S\|_F^2 \\ &= \arg\min_L \ \|L\|_* + \langle Y_k, M - L - S\rangle + \frac{1}{2\mu_k^{-1}}\|M - L - S\|_F^2 \\ &= \arg\min_L \ \|L\|_* + \frac{1}{2\mu_k^{-1}}\big\|L - (M - S + \mu_k^{-1}Y_k)\big\|_F^2 \end{aligned} \tag{2.32}$$

Suppose $U\Sigma V^T = M - S + \mu_k^{-1}Y_k$ is the SVD of $M - S + \mu_k^{-1}Y_k$. Using the same arguments as in section 2.2.2, the minimizer in (2.32) is given by $L_{k+1} = U\mathcal{S}_{\mu_k^{-1}}[\Sigma]V^T$, where $\mathcal{S}_\varepsilon$ is the soft-thresholding operator defined in (2.11).

Conversely, suppose that L in (2.31) is fixed. The "directional" subproblem in the variable S is

$$\begin{aligned} S_{k+1} &= \arg\min_S \ \|L\|_* + \lambda\|S\|_1 + \langle Y_k, M - L - S\rangle + \frac{\mu_k}{2}\|M - L - S\|_F^2 \\ &= \arg\min_S \ \lambda\|S\|_1 + \langle Y_k, M - L - S\rangle + \frac{1}{2\mu_k^{-1}}\|M - L - S\|_F^2 \\ &= \arg\min_S \ \lambda\|S\|_1 + \frac{1}{2\mu_k^{-1}}\big\|S - (M - L + \mu_k^{-1}Y_k)\big\|_F^2 \end{aligned} \tag{2.33}$$

Again using the results from section 2.2.2 we find that $S_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}[M - L + \mu_k^{-1}Y_k]$.

The alternating directions method for solving (2.31) is based on solving (2.32) and (2.33) iteratively, alternating between using a fixed S to update L and using the newly computed L to update S until convergence. Note that one can use prior iterates L_k and S_k for hot-starting the alternative directions based solution of (2.31) in order to significantly reduce the necessary number of directional steps within each iteration. These are all the ingredients needed for the alternating directions ALM method, which is given is Algorithm 11.

The following theorem gives the main convergence result for Algorithm 11.

Theorem 9 ([20])**.** *For Algorithm 11, any accumulation point $(\hat{L}, \hat{S})$ of $(\hat{L}_k, \hat{S}_k)$ is an optimal solution to the Robust PCA problem* (2.1) *with convergence rate of at least $O(\mu_{k-1}^{-1})$ in the sense that*

$$\left|||\hat{L}_k||_* + \lambda||\hat{S}_k||_1 - f^*\right| = O(\mu_{k-1}^{-1})$$

where f^* is the optimal value of the Robust PCA problem.

Algorithm 11: Alternating Directions Augmented Lagrangian Method

Input: Observation matrix M, parameter λ
initialization: $k = 0$, $\mu_0 > 0$, $\hat{Y}_0 = \text{sign}(M)/J(\text{sign}(M))$;
while *not converged* **do**
  $j = 0$, $L^0_{k+1} = \hat{L}_k$, $S^0_{k+1} = \hat{S}_k$;
  while *not converged* **do**
    $(U, \Sigma, V) = \text{svd}(M - S^j_{k+1} + \mu_k^{-1}\hat{Y}_k)$;
    $L^{j+1}_{k+1} = U\mathcal{S}_{\mu_k^{-1}}[\Sigma]V^T$;
    $S^{j+1}_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}[M - L^{j+1}_{k+1} + \mu_k^{-1}\hat{Y}_k]$;
    $j = j + 1$;
  $\hat{Y}_{k+1} = \hat{Y}_k + \mu_k(M - \hat{L}_{k+1} - \hat{S}_{k+1})$;
  Update μ_k to μ_{k+1};
  $k = k + 1$;
Output: $L = \hat{L}_k$, $S = \hat{S}_k$

Inexact ALM for Robust PCA

The authors in [20] point out that it is not really necessary to solve (2.31) exactly at each iteration k by running the alternating directions subroutine until convergence. In fact, at the next iteration the computation will be done again for the updated matrix $\hat{Y}_{k+1}$, so the solution of the subproblem need only provide an intermediate result that is adequate to perform a sufficiently good update of $\hat{Y}$. It turns out that, under some technical conditions on the sequence $\{\mu_k\}$, one can also prove convergence of the overall algorithm when performing only a single step in either direction at each iteration k. This results in Algorithm 12 which in [20] is referred to as the "Inexact Augmented Lagrangian Method" (IALM).

Algorithm 12: Inexact Augmented Lagrangian Method

Input: Observation matrix M, parameter λ
initialization: $k = 0$, $\mu_0 > 0$, $Y_0 = M/J(M)$;
while *not converged* **do**
  $(U, \Sigma, V) = \text{svd}(M - S_k + \mu_k^{-1}Y_k)$;
  $L_{k+1} = U\mathcal{S}_{\mu_k^{-1}}[\Sigma]V^T$;
  $S_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}[M - L_{k+1} + \mu_k^{-1}Y_k]$;
  $\hat{Y}_{k+1} = Y_k + \mu_k(M - L_{k+1} - S_{k+1})$;
  Update μ_k to μ_{k+1};
  $k = k + 1$;
Output: $L = \hat{L}_k$, $S = \hat{S}_k$

While it is possible to show convergence of Algorithm 12, the authors do not provide a bound on the rate of convergence as they do for the "exact ALM" (Algorithm 11). However, empirical results, both in the original papers and in the following section, suggest that the IALM algorithm is generally significantly faster that the EALM algorithm with only slightly lower accuracy.

2.3 Discussion of the Algorithms

In the previous section we discussed a number of algorithms for Robust PCA. The naive application of general-purpose interior point solvers to the dual problem only works for small problems but does not scale with the size of the data matrix M. The simple iterative thresholding algorithm can handle very large problem sizes but its convergence is extremely slow. Both the dual gradient ascent algorithm and in particular the APG algorithm are much faster and better suited for large problems. An extension of the APG algorithm that uses a prediction strategy for the dimension of the principal singular space whose singular values are larger than the threshold allows to only compute a partial SVD in each step of the algorithm. With this extension the the APG algorithm turns out to be faster than the dual gradient ascent algorithm.

The current state of the art in terms of complexity, accuracy and convergence seem to be the ALM methods discussed in section 2.2.5. Though the authors of [20] do not give a complexity analysis for the inexact ALM algorithm, empirical results suggest that for practical problems the inexact ALM is considerably faster than its exact counterpart.

2.3.1 The Importance of the SVD

Most of the presented algorithms (in fact, all of them except for the direct use of interior point solvers on the dual) involve repeated computations of the Singular Value Decomposition (or at least a partial SVD) of matrices of considerable size. This is not very surprising, as the nuclear norm in the objective function is the sum of the singular values of the matrix argument. This repeated computation of the SVD is in fact the bottleneck of most current algorithms for Robust PCA. Hence it seems that, at least for the algorithms discussed above, being able to perform SVD on large matrices are the key to developing fast algorithms that can be used in applications with large-scale data.

Comparison of different SVD algorithms

Figure 2.2 shows a comparison of the average computation time of the SVD of matrices of different sizes using Matlab's internal SVD routine (based on the algorithm in [11]) and PROPACK [18], which is based on a Lanczos bidiagonalization algorithm with partial reorthogonalization. We have compared the average running time for square matrices of different sizes, up to dimension 3000. For each size we generated 20 random matrices, and determined the average running time over this set. From Figure 2.2 we see that for smaller matrices Matlab's internal SVD routine is significantly faster than PROPACK. On the other hand, for large matrices (of dimension greater than 500×500) PROPACK is much faster than the Matlab routine. In fact, for the largest matrices we tested it is about one order of magnitude faster. This shows that the SVD-based algorithms discussed above can all benefit strongly from fast SVD algorithms.

When the matrix under consideration is sparse, the SVD can be carried out much faster using specialized methods [4]. Unfortunately, while the iterates of the matrix S_k (in case of primal algorithms) are indeed sparse, the matrices for which the SVD actually needs to be computed are not. Therefore it does not seem clear how sparse SVD methods could help in improving performance of SVD-based Robust PCA algorithms.

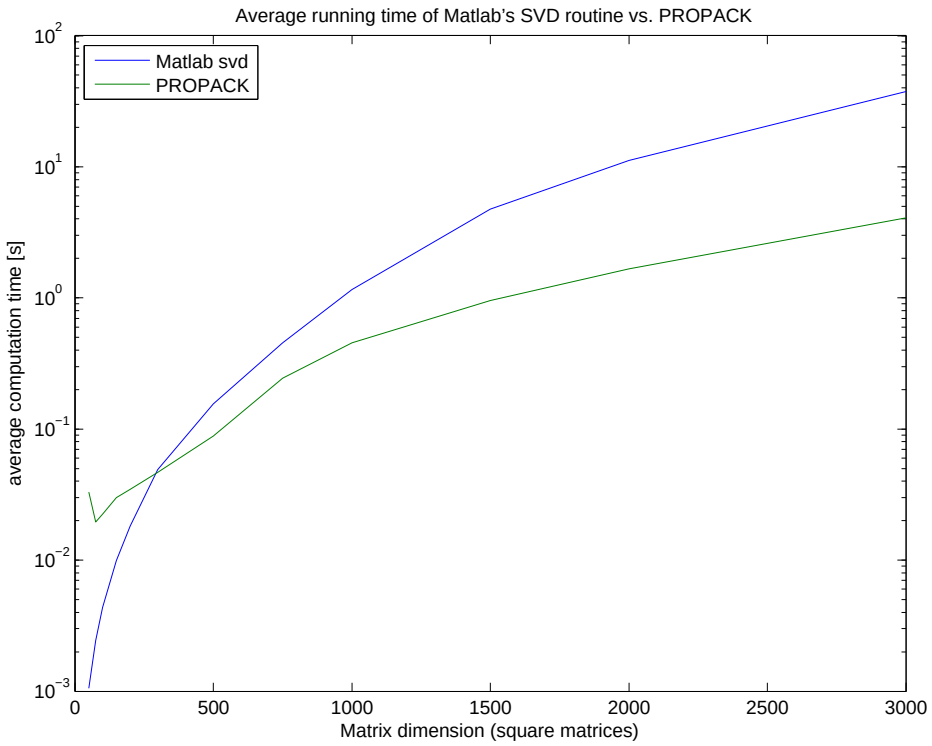


Figure 2.2: Numerical comparison of Matlab's and PROPACK's SVD routines

Partial SVD methods

If we look at the a Singular Value Thresholding operation it is obvious that we really only needs to compute those singular values that lie above the specified threshold (which is known a priori in each step and does not depend on the singular values), since the other singular values will be set to zero anyway. One possibility to speed up the algorithms that involve thresholding of singular values is therefore to use a partial SVD to compute only those singular values of interest. For the APG algorithm the authors of [20] use the software PROPACK that allows the computation of such a partial SVD. However, PROPACK by default requires the dimension of the principal singular space whose singular values are larger than the threshold, which is of course unknown a priori. Since it turns out that the rank of the the iterates L_k in the APG algorithm is monotonically increasing, a reasonable prediction of this dimension is not too hard [20]. Doing so then allows to a partial SVD at each step rather than a full SVD, which can potentially speed up the algorithm (if the partial SVD can be computed efficiently).

The PROPACK package has later also been modified to allow the computation of those singular values above a given threshold [19]. Figure 2.3 shows a comparison of the computation times of PROPACK's standard (full) SVD routine and the algorithm [19] for different thresholds. Contrary to what one would expect, the full SVD in all cases is significantly faster than the partial SVD. At this point it is not clear what the reason for this is. Either the algorithm for performing the "thresholded SVD" itself is very slow, or the implementation that is provided by [19] is extremely inefficient (or both). Either way, it obviously makes no sense to use to the provided implementation to compute partial SVDs. This is not to say that partial SVD is in generally slow. In fact, in [20] it is found that computing the partial SVD for a fixed dimension of the principal singular space

using PROPACK indeed results in a speedup of the APG algorithm.

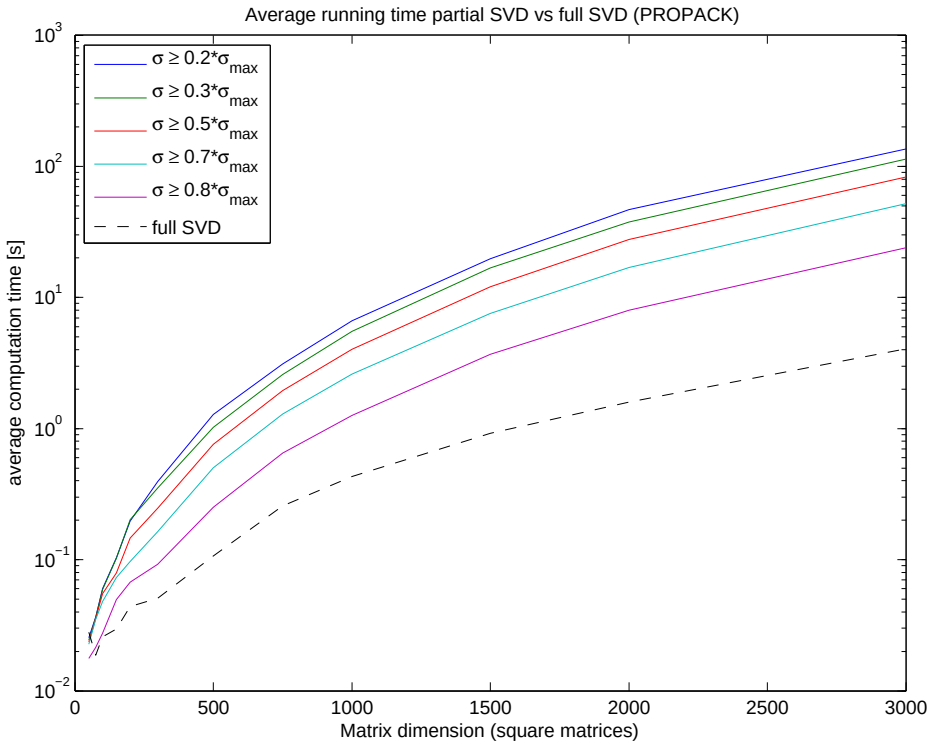


Figure 2.3: Numerical comparison of partial and full SVD via PROPACK

If we look at the discussed algorithms, we notice another problem with the partial SVD methods: the value of the threshold for the singular value thresholding operation in all cases decreases with the iteration number. Hence the number of singular values that need be computed actually grows with the number of iterations, which means that computing a partial SVD yields the highest benefit only in the early stages of the algorithm.

Warm-start methods for SVD

Another potential way of speeding up many SVD-based algorithms is to exploit the fact that the matrix of which the SVD has to be computed does generally not change much between iterations, in particular after a few iterations. To utilize this fact, the authors of [22] propose a warm start technique for the block Lanczos method for SVD computation. In the following section we will also provide simulation results of an inexact ALM algorithm based on this warm-start SVD method.

2.3.2 Numerical Comparison of Robust PCA algorithms

In this section we provide some simulation results to illustrate the performance of the different Robust PCA algorithms. As seen in section 2.2.1, using interior point methods to solve the dual problem is computationally infeasible for all but the smallest problems, hence we will not include interior point methods into the comparison. Furthermore, the iterative thresholding method, while

applicable also to very large problem sizes, exhibits extremely slow convergence. For a comparable relative error the required number of iterations (and therefore the computation time) for problems of reasonable size is so large that a more detailed comparison with the faster APG and ALM algorithms seems unnecessary.

The implementation of the algorithms used for the simulation are slightly modified versions of the matlab code that is freely available from `http://perception.csl.uiuc.edu/matrix-rank/sample_code.html`. The files are all provided with this report.

All of the numerical results that will be presented in this section have been obtained from simulations on randomly generated data. The (square) test matrices were generated as $M = L + S$, where the low rank component L was formed as $L = L_1 L_2^T$, with $L_k \in \mathbb{R}^{n \times r}$ where $r = 0.1n$ is the rank of L, and each entry $(L_k)_{i,j}$ i.i.d. Gaussian with variance $\sigma^2 = n^{-1}$. The cardinality of the support of the sparse error component S was chosen as $0.1n$, with each non-zero entry i.i.d uniform in $[-500, 500]$. For each of the selected matrix dimensions we simulated 10 different scenarios and computed the average overall running time, the average number of iterations and the average relative errors $\|L - \hat{L}\|_F / \|L\|_F$ and $\|S - \hat{S}\|_F / \|S\|_F$.

As can be seen from the first comparison in Figure 2.4, the Dual Projected Gradient Ascent algorithm (`dpga`) is much slower than both the Accelerated Proximal Gradient method (`apg`) and the Exact and Augmented Lagrangrian Method (`ealm`). Therefore we will exclude the Dual Projected Gradient Ascent algorithm from the following comparisons.

This leaves us with the Accelerated Proximal Gradient method (`apg`), the Accelerated Proximal Gradient method with partial SVD (`papg`), the Exact and Inexact Augmented Lagrangrian Methods (`ealm` and `ialm`, respectively) and the Inexact Augmented Lagrangrian Method with warm-start Block Lanczos SVD computation (`BLWSialm`). Figure 2.5 shows simulation results for these algorithms on matrices of sizes between $n = 50$ and $n = 1500$.

From Figure 2.5 we can see that there is a clear qualitative and quantitative difference between the APG-based algorithms and the ALM-based algorithms in terms of the relative errors, with the ALM-based algorithms achieving much higher accuracies, in particular problems of large size. This relates back to Remark 2.2.1, in which we pointed out that the APG algorithm provides only an approximate solution to the problem (to be fair, we also chose the tolerance of the APG-based algorithms a slightly higher so as to achieve comparable average running times). Moreover, we notice that the number of iterations of the APG-based algorithms is much higher than the one of the ALM-based algorithms, in particular the one of the exact ALM algorithm[5]. Interestingly, the number of required iterations for all of the simulated algorithms is more or less independent of the problem size. Figure 2.5 also shows that the APG algorithm using partial SVD computation (`papg`) is significantly faster than the one computing a full SVD at each step (`apg`). As one would expect, the average number of iterations in both cases is the same.

Numerical Comparison of ALM-based algorithms

The previous simulation results show that the Augmented Lagrangian Methods are superior to all other algorithms discussed so far, both in terms of average running time and in terms of accuracy (the memory requirements do not differ much between the first-order algorithms). At this point we

[5] note that for this algorithm we only count the outer number of iterations, not the inner iterations that are needed for solving the subproblem using the alternating directions method

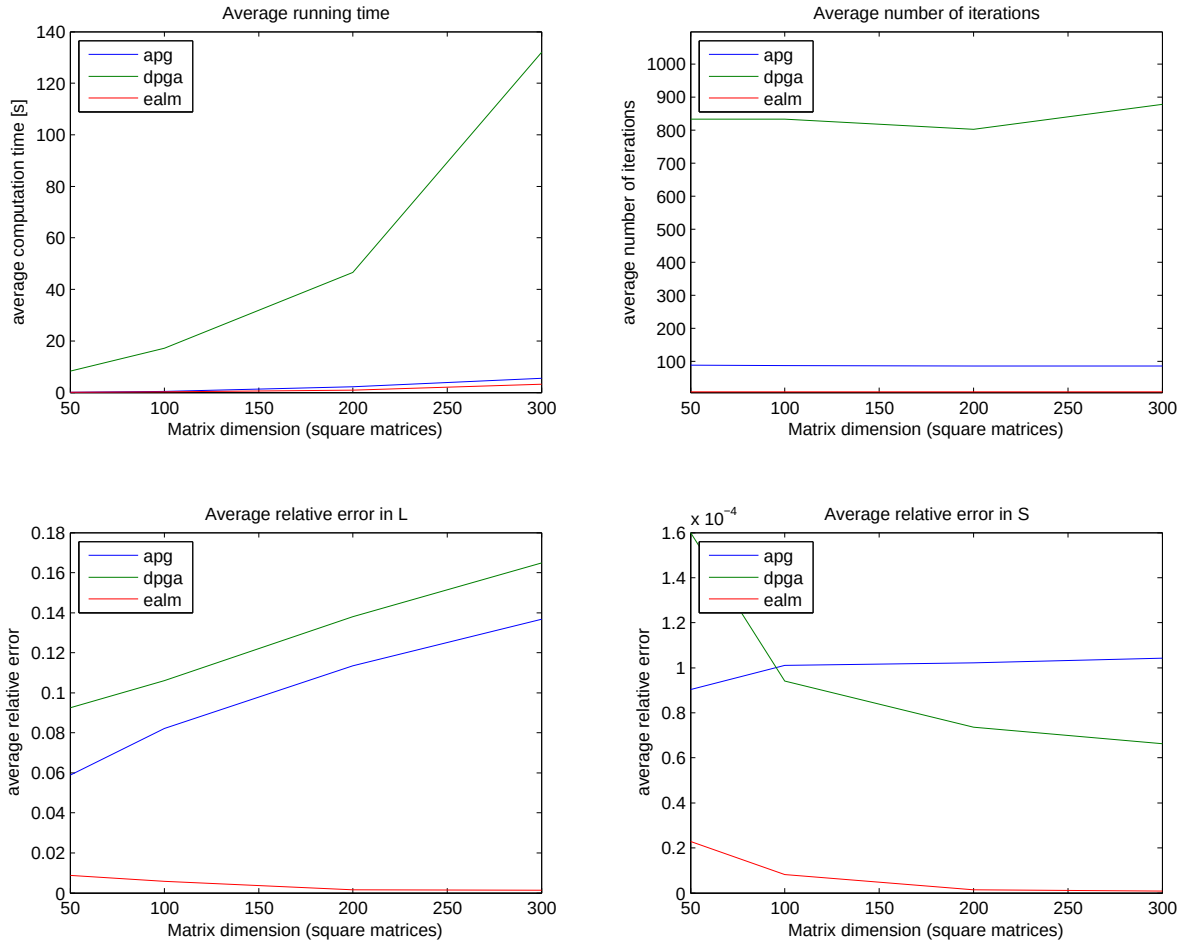


Figure 2.4: Simulation result for different algorithms for Robust PCA

do not see any reason to use a different algorithm, at least for the type of problem we study here. In our final simulation study, summarized in Figure 2.6, we compare the different ALM-based algorithms, namely the Exact and Inexact Augmented Lagrangrian Methods (`ealm` and `ialm`, respectively) and the Inexact Augmented Lagrangrian Method with warm-start Block Lanczos SVD computation (`BLWSialm`). We consider random (square) data matrices up to dimension $n = 3000$.

Here the verdict is also quite clear, with both inexact ALM algorithms consistently faster than the exact ALM algorithm, while achieving a comparable accuracy. For all but the smallest matrix dimensions, the inexact ALM algorithm based on the warm-start Block Lanczos SVD computation is faster (about 1.5x) than the standard inexact ALM algorithm. However, its accuracy is generally lower (this can be more easily seen in Figure 2.7, which does not include the exact ALM algorithm). It should be pointed out that this is also the case when the tolerance of the stopping criterion is reduced. This seemingly counterintuitive behavior is due to the fact that the SVD computation of the warm-start Block Lanczos SVD method is inherently approximate (which is not the case for the standard inexact ALM algorithm). What is remarkable in Figure 2.7 is that the number of iterations is essentially independent of the matrix size. Note however that this strong consistency very likely also has to do with the fact that all the considered randomly generated matrices are

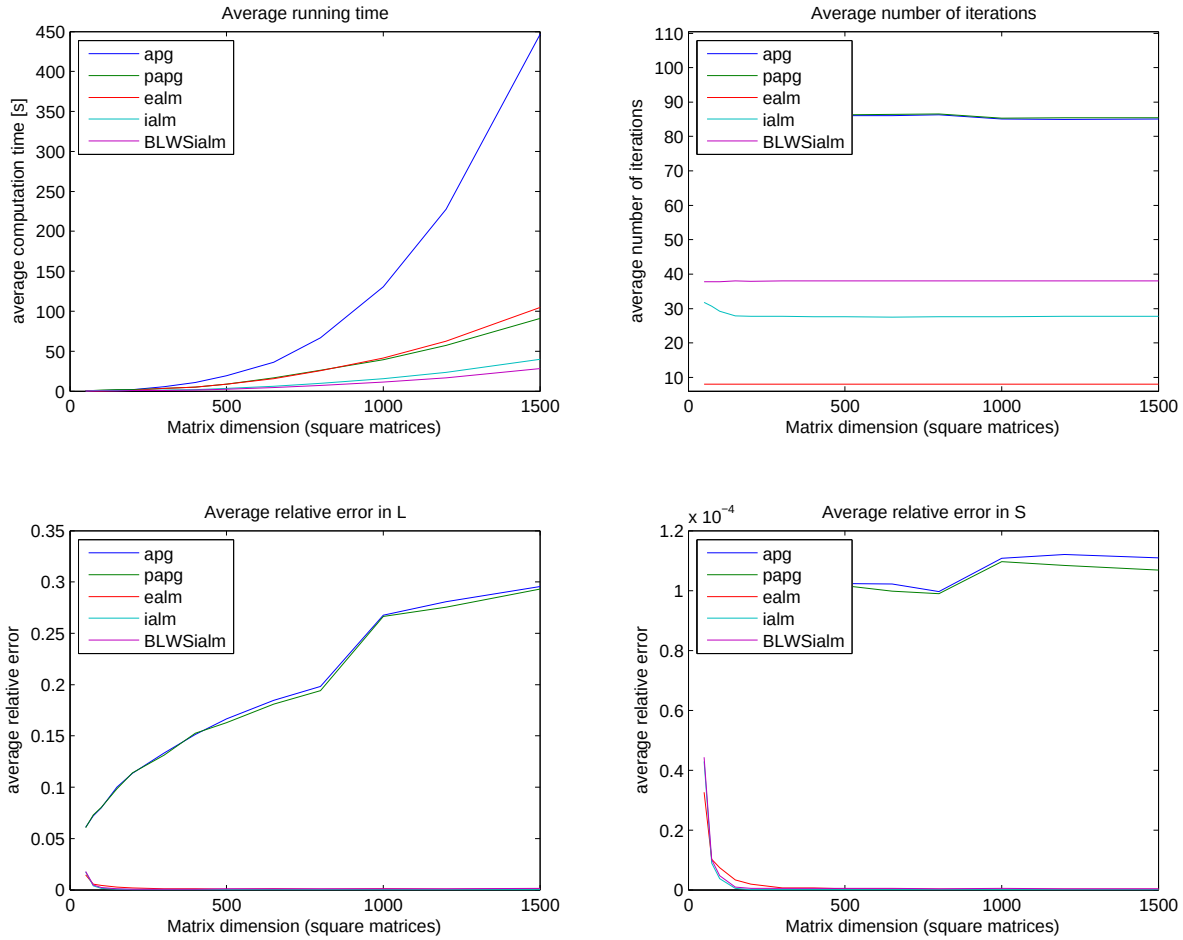


Figure 2.5: Numerical comparison of first-order algorithms for Robust PCA

structurally identical.

We want to emphasize here that the implementation of the warm-start block Lanczos method, as provided by [19], is completely Matlab-based and not fully optimized for performance. A careful implementation of the complete warm-start block Lanczos based inexact ALM algorithm in a performance-oriented language such as C can be expected to yield further speedups. Nevertheless, even with the current implementation it is possible to solve rather large problems quite fast, with problems involving matrices with tens of million entries being solved in just a few minutes.

2.3.3 Possible Directions for Parallelization

With the advent of highly parallelized computing architectures in modern CPUs and GPUs, a number of projects have been devoted to the development and implementation of algorithms that exploit this massive computing power. Examples for these are MAGMA [27] and CULA [15], which are adapting classic high-performance linear algebra packages such as LAPACK and BLAS to the highly parallelized architecture of modern GPUs.

Some of the steps in the discussed SVD-based algorithms are inherently parallelizable, for example the entry-wise soft-thresholding of a matrix, the elementary matrix operations or the computation

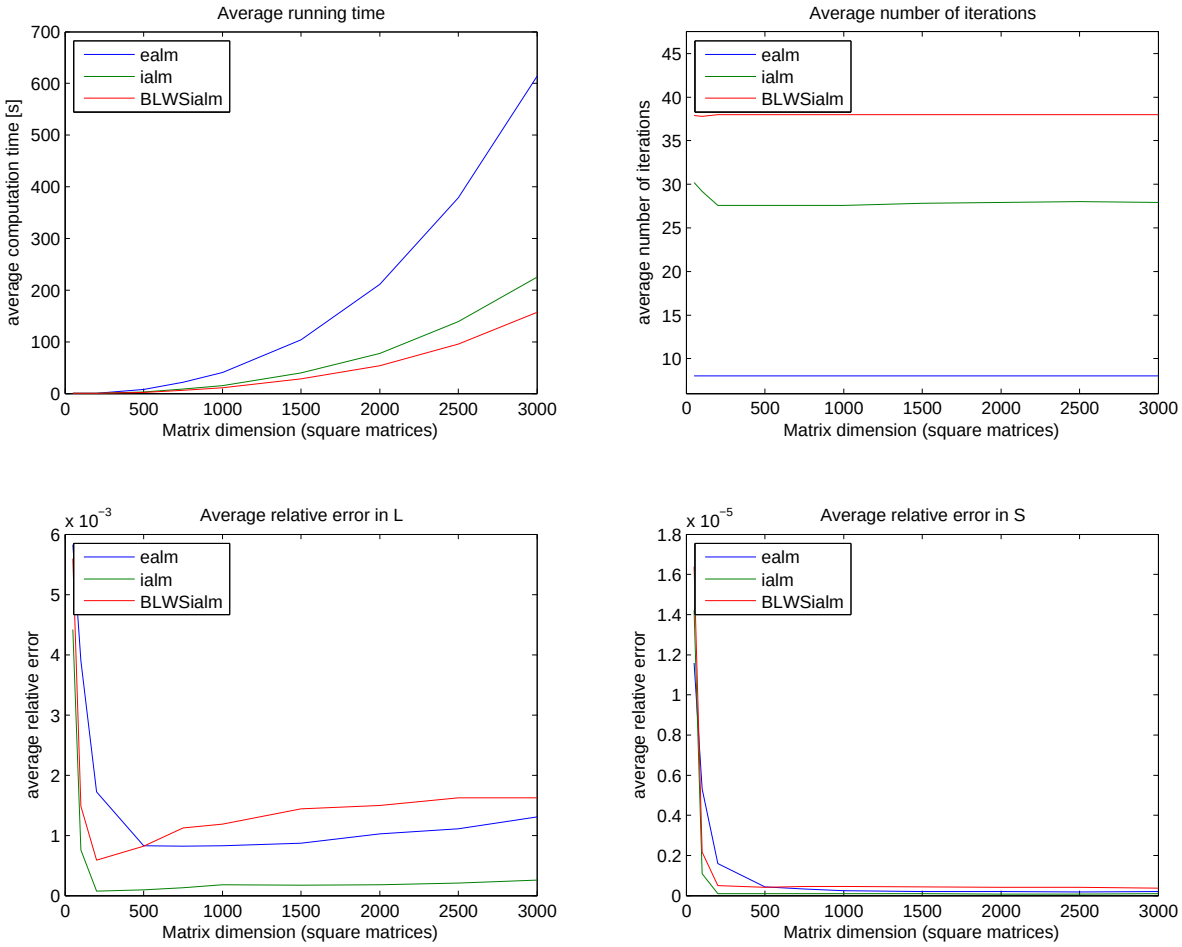


Figure 2.6: Numerical comparison of different ALM methods for Robust PCA

of the Frobenius norm for the stopping criterion. The parallelization of the SVD is less obvious, but research in this area is quite active and some methods have been proposed [4]. At this point we do not want to go into the details of how to parallelize the surveyed algorithms (in particular the fast augmented Lagrangian methods), we merely want to point out that we indeed see potential for further speedups and hence the use of the discussed algorithms also on large-scale data.

2.4 Outlook: Algorithms for Stable Principal Component Pursuit

As was discussed in section 1.4.2, the results obtained for the Robust PCA problem have been extended to the case when in addition to the sparse noise, the data is corrupted also by a small non-sparse noise component [34]. This problem is usually referred to as Stable Principal Component Pursuit (Stable PCP). The associated optimization problem is

$$\begin{aligned} p^* = \min_{L,S} \; & \|L\|_* + \lambda \|S\|_1 \\ \text{s.t.} \quad & \|M - L - S\|_F \leq \delta \end{aligned} \tag{2.34}$$

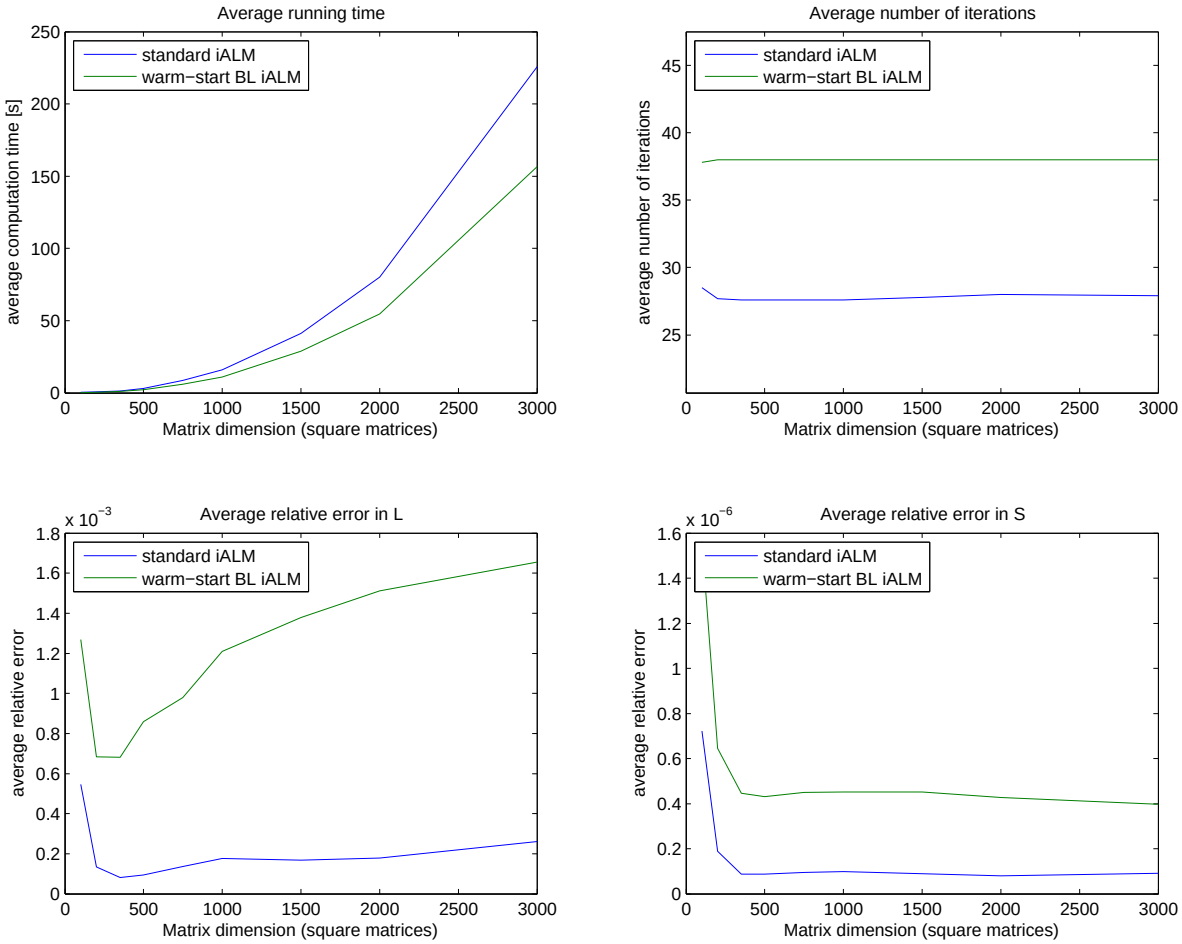


Figure 2.7: Numerical comparison of warm-start vs. standard Block-Lanczcos method

where $\delta > 0$ is given. In this section we want to give a brief outlook on what algorithms have been proposed for (2.34). While to date the literature on algorithms for this problem is less extensive than for the classic Robust PCA problem, a number of efficient ALM algorithms have already been proposed.

First of all, it is not hard to reformulate problem (2.34) as an SDP, which can then in principle be solved using general purpose interior point solvers. However, the same scalability issues as in the standard Robust PCA problem will prohibit the use of these methods for most problems of practical interest.

In the original Stable PCP paper [34], the authors directly use the Accelerated Proximal Gradient (APG) algorithm from section 2.2.3 for solving an approximate version of (2.34). Using a duality argument it is easy to see that (2.34) is equivalent to

$$\min_{L,S} ||L||_* + \lambda||S||_1 + \frac{1}{2\mu}||M - L - S||_F$$

for some value $\mu(\delta)$. Note that the above problem is just (2.21). The authors make the argument that it makes sense to choose μ to be the smallest value such that the minimizer of is likely to be $\hat{L} = \hat{S} = 0$ if $L = S = 0$ and $M = Z$. In this way, μ is large enough to threshold away the noise,

but not too large to over-shrink the original matrices. Assuming an iid Gaussian distribution of the dense noise component, i.e. $(Z_0)_i, j) \sim \mathcal{N}(0, \sigma^2)$, it turns out[6] that $n^{-1/2}\|Z_0\| \to \sqrt{2}\sigma$ a.s. as $n \to \infty$. As a result of this, the choice for μ becomes $\mu = \sqrt{2n}\sigma$. We note that the assumption of a Gaussian noise matrix Z_0 is often reasonable but not always satisfied. If it is not, then it is not clear whether using the APG algorithm to solve the associated approximate problem is a good idea and different algorithms may be needed.

In [29] an algorithm for (2.34) based on partial variable splitting is proposed[7]. In fact, the problem an simply be written as

$$\begin{aligned} p^* = \min_{L,S,Z} \ & \|L\|_* + \lambda\|S\|_1 \\ \text{s.t.} \quad & M = L + S + Z \\ & \|Z\|_F \leq \delta \end{aligned} \tag{2.35}$$

At heart, the proposed ASALM algorithm is the extension of the ALM method from section 2.2.5 to the case with partial variable splitting. It turns out that the additional subproblem appearing at each iteration can be solved explicitly at a cost similar to the matrix value thresholding (the cost of which is $O(mn)$). The dominant computation is still the singular value thresholding operation that is based on the computation of the SVD. Hence the proposed method is comparable in computational complexity to the ALM method from section 2.2.5.

In [1] a number of different algorithms for Stable PCP are discussed, most of which are based on a smoothed or partially smoothed objective function. Two of these algorithms essentially apply Nesterov's optimal algorithms [23] to the partially smoothed problem, which yields a theoretical complexity of $O(1/\varepsilon)$. It is further shown that the subproblems appearing in these algorithms either have a closed-form solution or can be solved very efficiently. Still using a partially smoothed objective function, the authors also apply a partial variable splitting technique and propose to use an alternating minimization algorithm for a linearized version of the associated augmented Lagrangian function. This also yields an algorithm with a theoretical complexity of $O(1/\varepsilon)$.

The authors of [1] also propose a first-order algorithm that works directly with the fully non-smooth objective, which they call NSA. Very similar in nature to the ASALM method proposed in [29], this algorithm is also an extension of the ALM method from section 2.2.5 to the setting with partial variable splitting. While the authors were not able to derive theoretical complexity results, empirical evidence suggests that this algorithm is quite efficient in solving the Stable PCP problem. In particular, numerical simulations indicate that NSA consistently outperforms ASALM. The main reason for this seems to be that while ASALM performs an alternating minimization over three directions (L, S and Z), NSA uses the fact that the joint minimization over (S, Z) also has an explicit solution and therefore only alternates over two directions.

[6] this based on the strong Bai Yin Theorem [2], which implies that for an $n \times n$ real matrix with entries $\xi_{ij} \sim \mathcal{N}(0, 1)$ the it holds that $\limsup_{n\to\infty} \|Z_0\|_2/\sqrt{n} = 2$ almost surely

[7] The authors in fact consider the more general problem where the constraint reads $\|\mathcal{P}_\Omega(M - L - S)\|_F \leq \delta$, where $\mathcal{P}_\Omega$ is the projection on the set Ω of observed data

Chapter 3

Applications

3.1 Overview

This section will review some applications using Robust PCA. Currently, most of the applications are related to computer vision. As we will show later, many images involve natural characteristic of low-rank structure, which makes Robust PCA a perfect fit to them. We will also explore some other applications that is theoretically with low-rank and sparse structure and show what we get from them.

3.2 Robust PCA Applications

3.2.1 Background modeling from surveillance video

Video data is a natural candidate for low-rank modeling, due to the correlation between frames. One of the most basic algorithmic tasks in video surveillance is to estimate a good model for the background variations in a scene. This task is complicated by the presence of foreground objects: in busy scenes, every frame may contain some anomaly. The background model needs to be flexible enough to accommodate changes in the scene, for example due to varying illumination In such situations, it is natural to model the background variations as approximately low rank. Foreground objects, such as cars or pedestrians, generally occupy only a fraction of the image pixels and hence can be treated as sparse errors.

Figure 3.1: Original video frames

We consider five frames from an original video, as shown in fig. 3.1, which is a scenario that one man passes by. The resolution of each frame is 176×144. We first separate them into three channels (RGB). For each channel, we stack each frame as a column of our matrix $M \in \mathbb{R}^{25344 \times 5}$. We

Figure 3.2: Low-rank components

Figure 3.3: Sparse components

decompose M into low-rank components L and sparse components S by applying the Robust PCA framework. Then we combine the three channels again to form images with low-rank component and sparse components respectively. We are using a 2GHz quad core laptop on which it takes 0.92s to finish decomposition for three channels. We can find that the L, as shown in fig. 3.2, correctly recovers the background, while S, as shown in fig. 3.3, correctly identifies the moving person.

The same setting is used for our own video that we took at UC Berkeley. The resolution in our video is 480×640, which is consistent with the resolution of many cell phone cameras. Figure 3.4 shows the result on 5 images each taken one second apart. In this case the computation takes 15 seconds. We can see that, even though there is more than one moving person present, as long as the noise (which in this case is the foreground) is sparse, Robust PCA can still separate background and foreground. Since it does not consume too much time, this method could potentially be used to removing wandering people when someone takes a picture.

Figure 3.4: Perfect decomposition from campus video frames

Finally, we used all the frames from the 30 seconds video to run Robust PCA. We first down-sample them into 160×214 resolution in order to reduce the running time. We capture some frames from the resulting video. Most frames are good, but some are imperfect in terms of separating the

Figure 3.5: Imperfect decomposition from campus video frames

foreground and background. We can find that in Figure 3.5, when the foreground objects are dense, there are some "ghost" foreground people appearing in background video frames (known to us as the "Ghosts of Berkeley"). To explain this behavior, consider the following matrix:

$$M = \begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \\ 0 & 0 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \end{bmatrix}$$

We can seperate M into low-rank component and sparse component by Robust PCA. However, Robust PCA will favor a decomposition as:

$$M = \begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \\ 0 & 0 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \end{bmatrix} = \begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \\ 34.0 & 34.0 & 36.8 & 36.8 & 36.8 \\ 100 & 100 & 100 & 100 & 100 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -34.0 & -34.0 & 63.2 & 63.2 & 63.2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

such that $||L||_* + \lambda||S||_1 = 513.64$ rather than

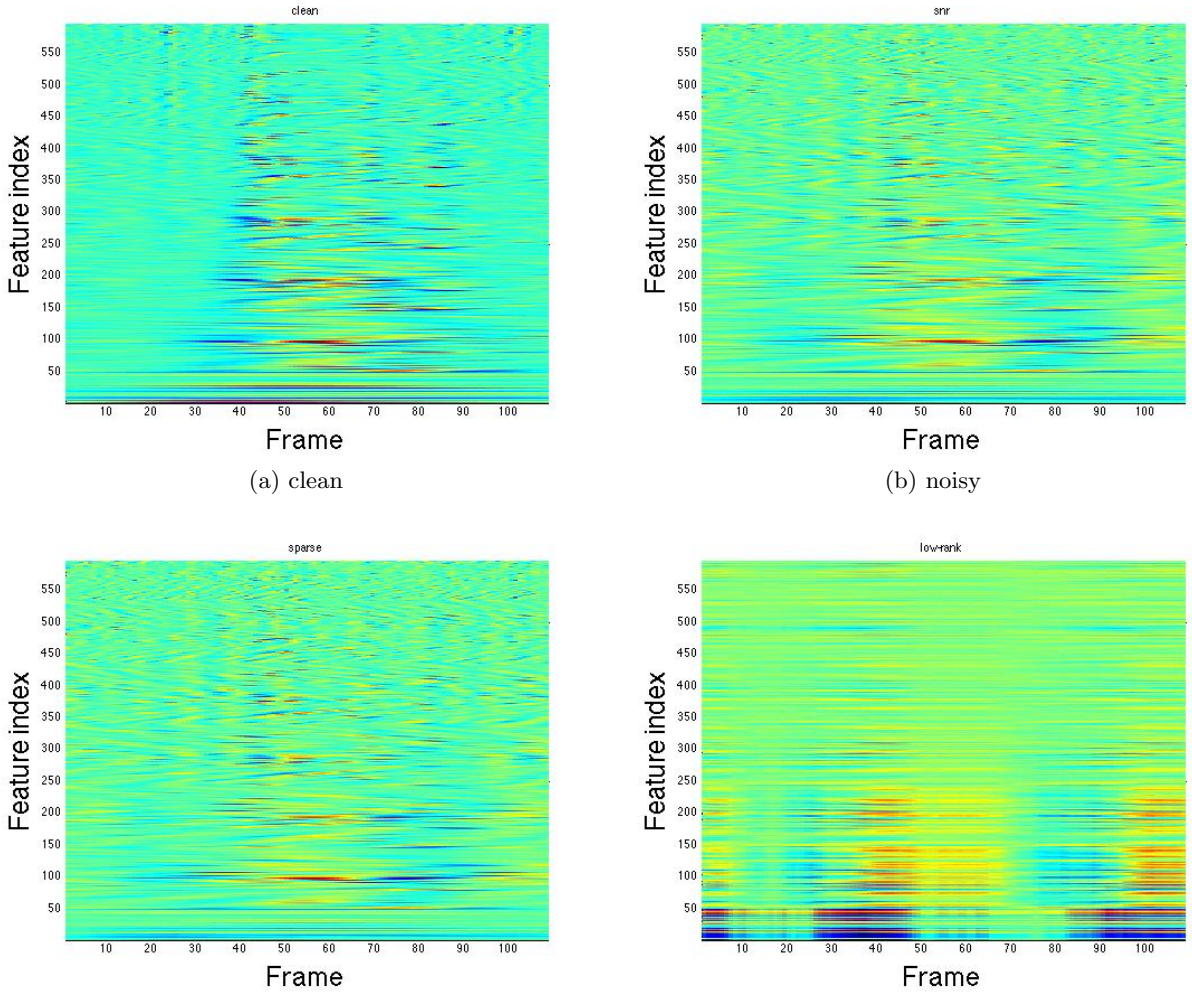
$$M = \begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \\ 0 & 0 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \end{bmatrix} = \begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -100 & -100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

such that $||L||_* + \lambda||S||_1 = 536.66$

From the above example, we can see that if most of the frames are corrupted in given pixels, Robust PCA will generally not be able to separate the background and foreground at those pixels perfectly. In fact, in this case the sparsity assumption on the noise is not satisfied. One way to fix this issue in a given frame is to adjust the value of λ, but it is generally hard to choose a λ that fits the entire video. Despite that, Robust PCA is still performing well in separating background and sparse foreground moving objects.

3.2.2 Using Robust PCA in speech recognition

Intuitively, a consistent sound would have a low-rank structure. If someone is speaking with background noise which is consistent, we would believe that we can separate a clearer speech (the sparse component) from background noise (low-rank component) using the Robust PCA framework. If we get a clearer speech signal, the accuracy of speech recognition with current standard technique will increase. According to this thought, we did an experiment and tried to see whether Robust PCA works in such cases.



(a) clean

(b) noisy

(c) sparse

(d) low-rank

Figure 3.6: Speech features

Figure 3.6a shows the features of a clean speech signal, where the x-axis denotes indices of small time frames and the y-axis denote the features vectors at each frame. These features are computed by a standard method [17]. Figure 3.6b shows the features domain of the same speech signal subjecting to noise with $SNR = 0$ in this case. We apply Robust PCA to decouple these features into low-rank component, as shown in Figure 3.6d, and sparse component, as shown in Figure 3.6c. Technically, we would believe that sparse component is corresponding to our speech signal whereas low-rank component is corresponding to noise. Then we use the sparse component as new features

and train a classifier via method in [32]. Unfortunately, comparing to the original method without Robust PCA, we only got improvement in data set with subway noise, which we consider as the most consistent noise among the data set. In this data set, Robust PCA improve the accuracy rate with 3% comparing to original method. For other noise, such as street noise and car noise, we both got decrease about 7% in accuracy. One possible reason is that in the real world, most of the noise is not low-rank and they are not consistent. However, speech signal could be low-rank sometimes because there are only a limited vowels and consonants in a language, which seems to makes Robust PCA not very suitable for de-noising.

3.2.3 Senate voting data analysis

We have used Robust PCA to analyze voting data of the US senators. The data involves 100 senators voting at 542 bills around 2005 to 2008. The original data matrix is a 542×100 matrix with elements {-1, 0, 1}, representing voting for and against the bill and abstention. We first form a 100×100 covariant matrix and then run Robust PCA to this covariant matrix. We choose $\lambda = 1/\sqrt{10}$ as general. The rank of the low-rank component is 58 and the number of non-zero entries in sparse component is 6222. The results is shown in Figure 3.7:

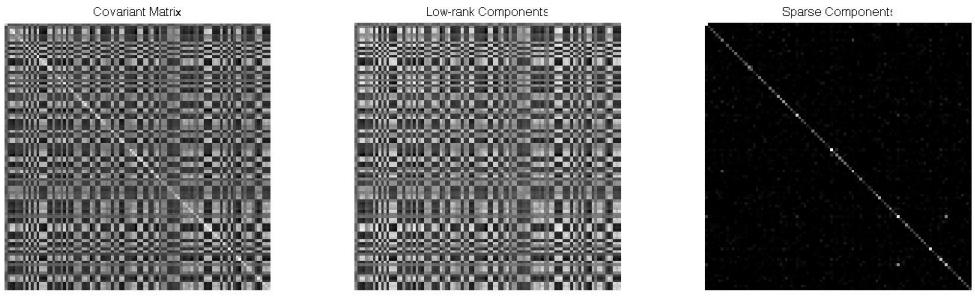


Figure 3.7: Covariant matrix and its low-rank component and sparse component

By analyzing the sparse component, we found that the sparse matrix seems to tell us some elements (i.e. senators) with strong connection. In general, the values in diagonal terms are large in sparse component because they are highly correlated. On the other hand, we could find that there are some other highly correlated elements expect for the diagonal terms from the sparse matrix. We filter the values larger than 0.8 in the sparse matrix expect for the diagonal terms. We found the following senators who are highly correlated in the sparse matrix: 1. (Specter, Snowe); 2. (Specter, Collins); 3. (Snowe, Collins) 4. (Jeffords, Collins) 5. (Dorgan, Conrad) 6. (Inouye, Akaka) 7. (McCain, Kyl). Interestingly, the facts seem to match these results. First, the three Republican senators Specter, Snowe and Collins, who appear in pairs 1, 2 and 3, are always doing the same decisions, which has been noted by several reports[1] [2]. Dorgan and Conrad in pair 5 are both from North Dakota. Inouye and Akaka in pair 6 are both from Hawaii. McCain and Kyl are both from Arizona.

As shown in Fig 3.8 and Fig 3.9, we also compared the first two principal components in the

[1]"The Gang of Three: Specter, Collins and Snowe" `http://usconservatives.about.com/b/2009/02/11/the-gang-of-three-specter-collins-and-snowe.htm`

[2]"NYT: Obama thanked Collins, Snowe, Specter for their patriotism `http://hotair.com/archives/2009/02/07/nyt-obama-thanked-collins-snowe-specter-for-their-patriotism/`

original covariant matrix and the low-rank component. Their patterns remain similar with each other.

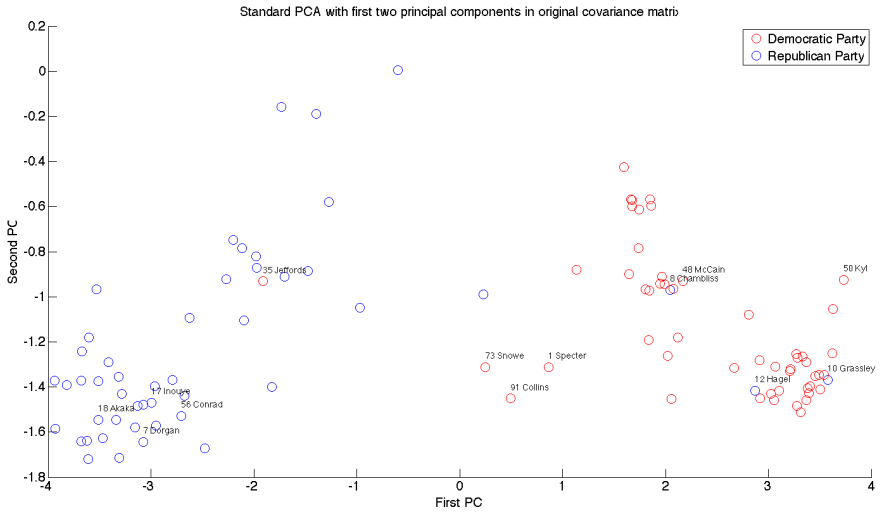


Figure 3.8: Standard PCA with first two principal components in original covariant matrix

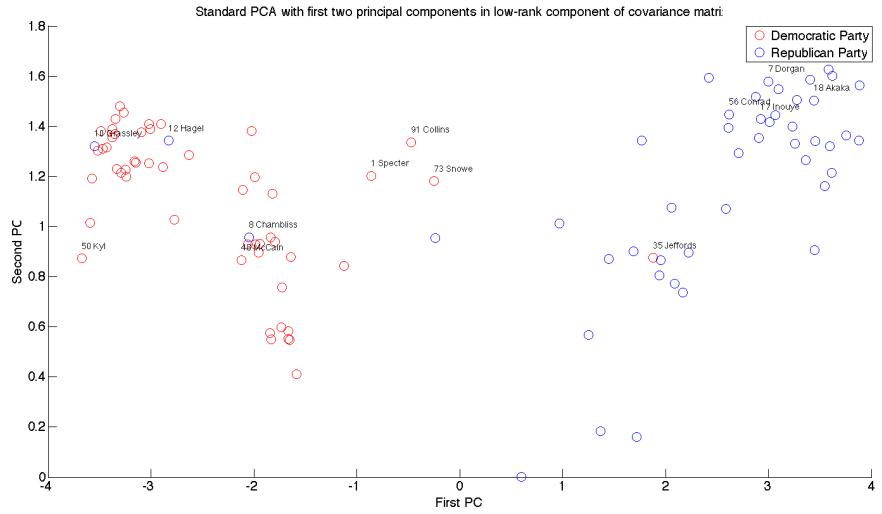


Figure 3.9: Standard PCA with first two principal components in low-rank component of covariant matrix

3.2.4 Pre-processing of Brain-Machine Interface neural spike data

We have used Robust PCA on some neural spike data that was collected from a monkey's brain. The Brain-Machine Interface project investigates how neural spike data can be used to interface with machines, with the ultimate goal of establishing some sort of bi-directional link (that is, to be able to control a machine through brain activity and, conversely, to "feel" feedback from measurements obtained by the machine).

The particular problem we have looked at in this project was to use Robust PCA to analyze the neural spike data obtained from the motor cortex of a monkey, who was moving his arms to control a computer cursor in different directions. The assumption was that the measurements would consist of a somewhat "low-rank" component associated to the movement required by the particular task, and some superimposed sparse noise associated to neuron firings in the background. The data available was time-series neural spike data from a particular experiment, where the monkey had to move the cursor in 8 different directions, according to 45 degree increments. Each of the task was repeated a certain number of times. A total of 127 parallel measurement channels were available, each corresponding to a neuron (or a small number of neurons).

As a pre-processing step, we normalized the time over the different trials for each component. The reasoning here was that the same task will correspond to the same neural firing pattern, independent of whether it was performed faster or slower[3]. The next step was to group each of the trials for each of the tasks into a number of time bins, i.e. if the number of bins is N then the i-th bin associated to neuron k counts the number of firings of neuron k within the interval $[(i-1)/N, i/N]$ (recall that trial time is normalized to 1).

Figure 3.10 shows the raw data after binning with 3 bins. here each of the 25 columns in each task represent the binned data from a single trial, where the 3 bins are stacked vertically. That is, within the large column associated to the 0 deg task, the first three entries of the first column correspond to the three time bins of the first neuron. The data is for 127 neurons, hence the overall matrix has $3 \cdot 127 = 381$ rows. From the raw data in Figure 3.10 we can already see some interesting features in the data, namely that each of the tasks seems to have its characteristic pattern.

For each of the tasks we apply Robust PCA to the data matrix to extract low-rank and sparse components. Figure 3.11 shows the low rank component extracted from the raw data of Figure 3.10. We notice that, as expected, the result is a "filtered" version of the raw data, in which the principal characteristics are retained while a sparse component has been removed. We think that the low-rank component in each of the tasks corresponds to the neuron firing pattern that can be directly related to the task, while the sparse component may be mis-detections, firings due to other movements, or simply sparse "background" firing that is always present.

For the 90 deg task Figure 3.12 shows a direct comparison of the raw data and the low-rank and sparse components obtained via Robust PCA. While for this comparably small data set the basic characteristics can be extracted visually just by looking at the matrix, this in general is not possible for high-dimensional data. One issue that we see with the available dataset is that the low rank component itself is quite sparse (since we are considering time-series data), hence some technical assumptions such as the incoherence condition will not necessarily hold. However, since not even neuroscientists have a clear understanding of what the data characteristics are and what the different patterns mean, we will not attempt to resolve this issue here. After the end of the semester we plan to learn more about the Brain-Machine interface project and, together with domain experts, to identify possible problems where Robust PCA could be used.

Based on the difference in neurons firing patterns between different tasks one can use machine learning techniques to predict from neural measurements which task the monkey is performing. Here Robust PCA could be used as a pre-processing step to filter out the sparse noise component. However, due to our lack of domain knowledge it is at this point unclear whether the Robust PCA framework has advantages over other and possibly simpler methods, for example a simple low-pass

[3] very little is known at this point about the structure of the dataset, so it was up to us to make reasonable assumptions about the data

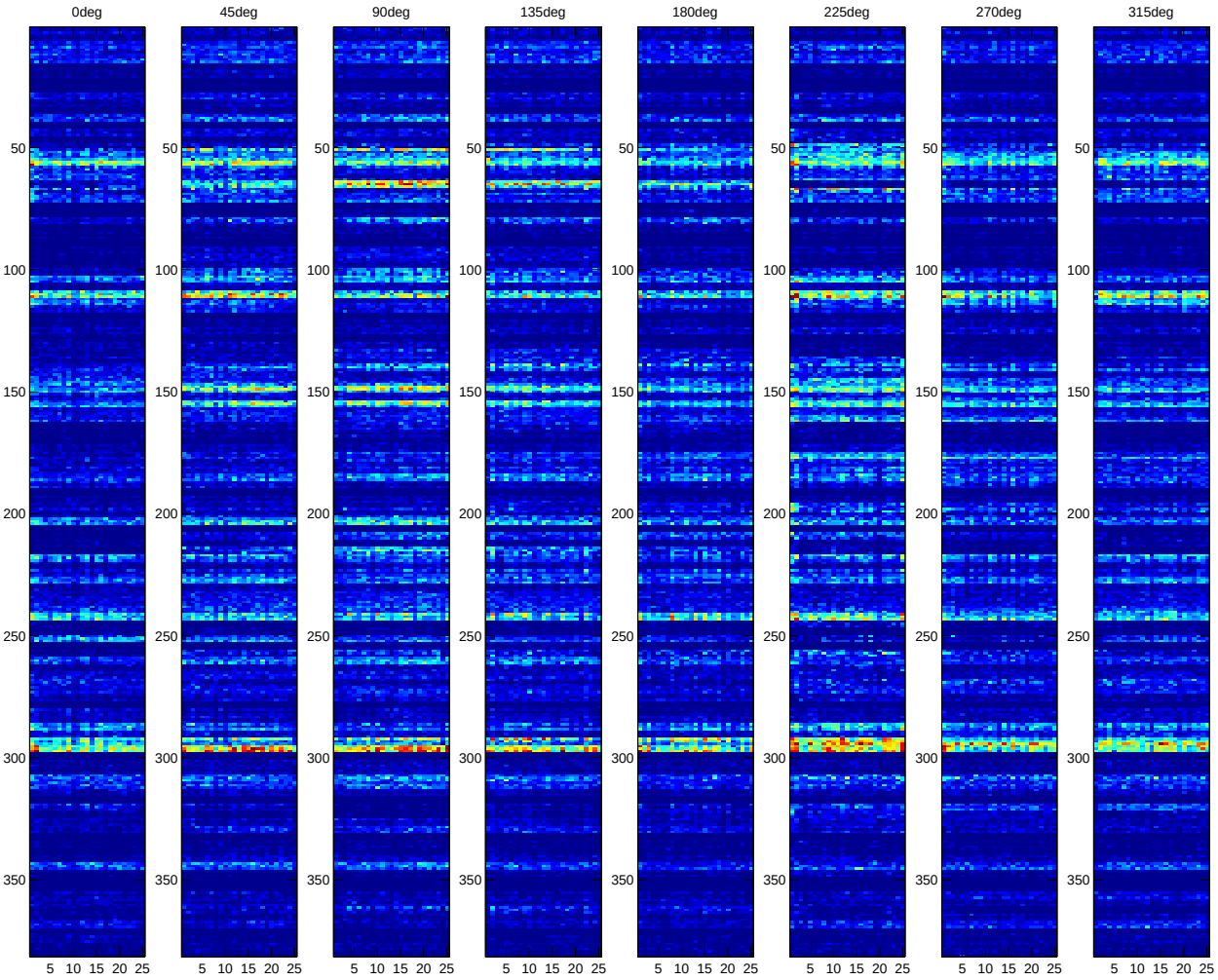


Figure 3.10: Raw (normalized and binned) neural spike data, $n_{bin} = 3$

filters along the rows of the data matrix. Our main reason for presenting this application example here was to illustrate that Robust PCA can indeed be used as a tool to analyze interesting real data of reasonable size.

3.3 Discussion

Robust PCA framework is a powerful tool to separate low-rank components and sparse components if we are given a combination of these two structures. However, generally most of the real world data is not directly a combination of these two structures unless the data is subjected to some kinds of transformation. One example is the data set of human faces with different expressions and illuminations. In this case, some researches have proposed some methods, RASL [26] and TILT [33], based on the original convex optimization problem in Robust PCA. They introduce transformation parameters to the original framework and by linearizing with respect to the transformation parameters, we got a new but not difficult convex problem. As long as the data is within a certain degree of transformation, the algorithms can still recover the low-rank structure and sparse structure.

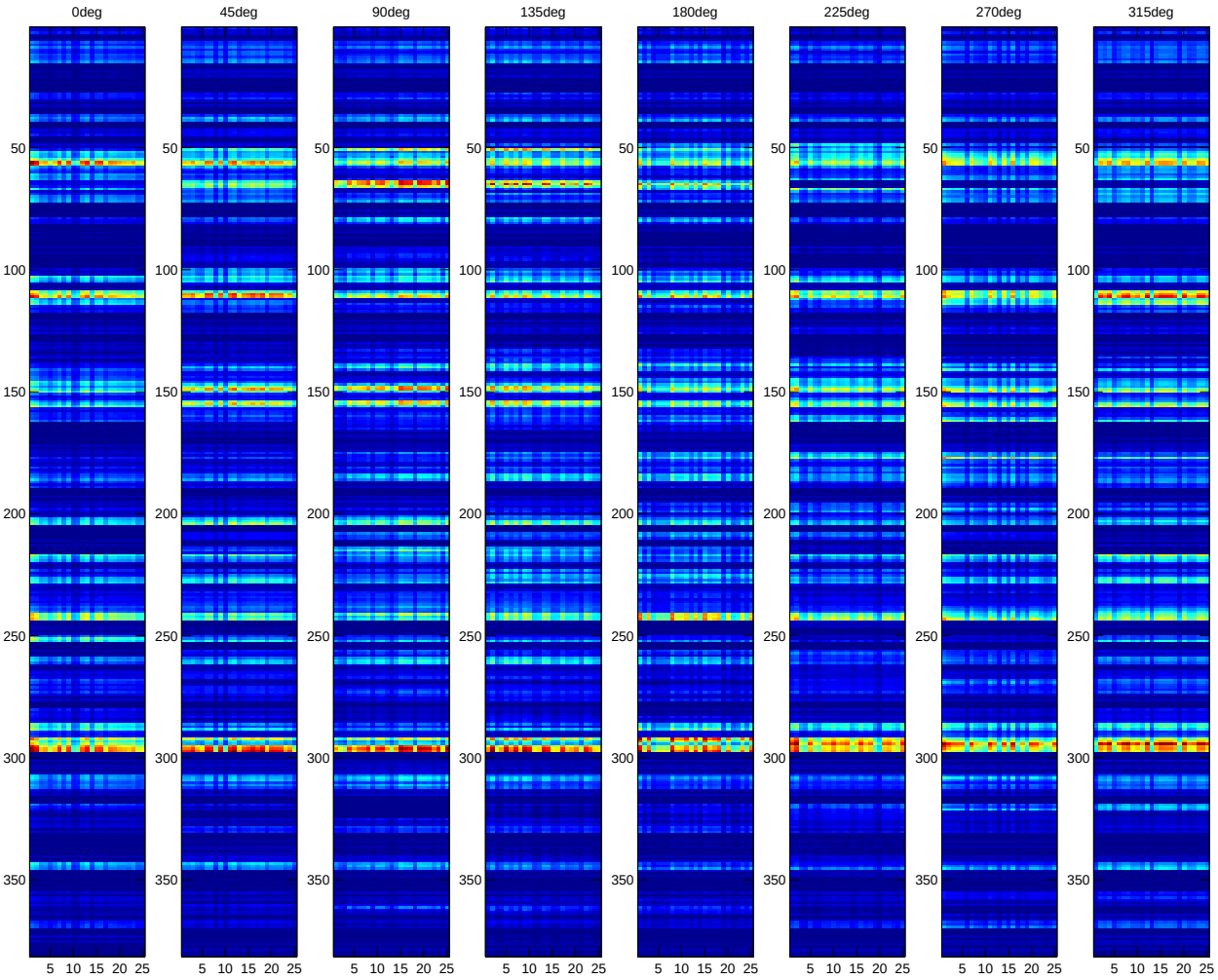


Figure 3.11: Processed neural spike data, $n_{bin} = 3$

We can also observe that in image data, the sparse noise is always presented as blocks. For example, the person in fig. 3.3 appears in connected pixels, so the noise is not distributed randomly in pixels. A possible direction that can improve the separation in image data is that we can introduce a penalty term in the original optimization problem:

$$\begin{aligned} p^* = \min_{L,S} \; & ||L||_* + \lambda |||S||_1 + \beta \sum_{(i,j),(k,l)\in\mathcal{N}(i,j)} (S_{ij} - S_{kl})^2 \\ \text{s.t.} \quad & M = L + S \end{aligned} \tag{3.1}$$

where $\mathcal{N}(i,j)$ means all the entries neighbor to (i,j) in matrix S and β is an constant. This problem is still convex so we can solve it via existing algorithms for convex optimization problems theoretically. However, we are not able to directly applying ALM which we discussed in Robust PCA problem because S_{ij} and S_{kl} are now coupled with each other that $\arg\min_S \; \mathcal{L}(L,S,Y,\mu)$ cannot be solved efficiently. If the scale of the problem becomes larger, an efficient algorithm to solve the problem is still needed to be developed.

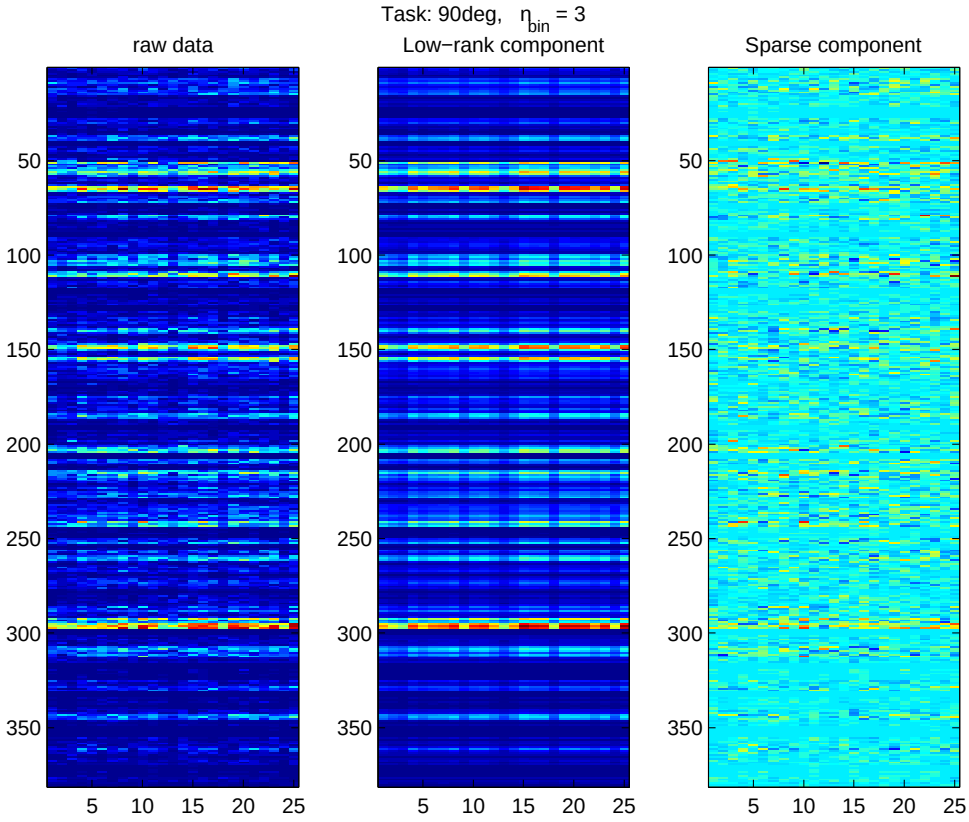


Figure 3.12: Raw data, low-rank component L and sparse component L for 90 deg task, $n_{bin} = 3$

Summary

In this report, we studied the recently proposed Robust Principal Component Analysis framework in terms of its underlying theory, algorithms and applications.

Regarding the theory, we focused our discussion on the proof of the probabilistic guarantees for recovery of a low rank matrix corrupted by sparse noise via a convex optimization techniques based on the a heuristic of minimizing ℓ_1-norm and nuclear norm. We recalled the results from several papers containing the initial contributions and discussed and derived proofs for details that were left out in the original discussion. In addition, we surveyed different existing variants and generalization of Robust PCA. Finally, we extended the Robust PCA framework to problems in which a bound on the rank of the low rank component is known. In this setting, we discussed an ℓ_1 heuristic for this problem which is applicable also to very large scale problems.

In the algorithms part, we surveyed different algorithms for the Robust PCA problem. We compared their theoretical complexity and convergence rates and performed numerical experiments on synthetic data. Our main finding was that the bottleneck of essentially all efficient algorithms is the computation of the Singular Value Decomposition of large but unfortunately non-sparse matrices. We have also discussed potential methods that could alleviate this bottleneck. Despite these shortcomings, we have found Robust PCA to work well on standard computers, even on large scale data involving matrices with tens of millions of entries.

We applied Robust PCA to several different problem domains. We used it to extract moving objects

from a sequence video frames and to extract speech from recordings corrupted by background noise. In addition, we analyzed the correlation of votes by candidates in senate voting data and used the framework to extract the low-rank component corresponding to particular movements from neural spike data.

To summarize, we have found Robust PCA to be a powerful tool to recover a low-rank matrix that is grossly corrupted by sparse noise. The framework provides strong performance guarantees, it can be implemented easily using efficient algorithms and has to potential to be applied to many practical problems.

Bibliography

[1] N. S. Aybat, D. Goldfarb, and G. Iyengar. Fast First-Order Methods for Stable Principal Component Pursuit. *arXiv preprint*, 1105.2126S, May 2011. 26, 69

[2] Z. D. Bai and Y. Q. Yin. Necessary and sufficient conditions for almost sure convergence of the largest eigenvalue of a wigner matrix. *The Annals of Probability*, 16(4):pp. 1729–1741, 1988. 25, 69

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. 52, 54

[4] M. Berry, D. Mezher, B. Philippe, and A. Sameh. *Handbook of parallel computing and statistics*, chapter Parallel Algorithms for the Singular Value Decomposition. Statistics, textbooks and monographs. Chapman & Hall/CRC, 2005. 61, 67

[5] D. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Optimization and neural computation series. Athena Scientific, 1996. 58

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. 49

[7] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM J. on Optimization*, 20(4):1956–1982, Mar. 2010. 50, 51

[8] E. Candes and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925 –936, june 2010. 25

[9] E. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9:717–772, 2009. 12, 22, 23

[10] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *J. ACM*, 58:11:1–11:37, June 2011. 1, 4, 14, 27, 29

[11] J. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM Journal on Scientific and Statistical Computing*, 11(5):873–912, 1990. 61

[12] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, Sept. 1936. 1

[13] H. Hotelling. Analysis of a complex of statistical variables into principal components. *J. Educ. Psych.*, 24, 1933. 1

[14] D. Hsu, S. Kakade, and T. Zhang. Robust matrix decomposition with sparse corruptions. *Information Theory, IEEE Transactions on*, 57(11):7221 –7234, nov. 2011. 29

[15] J. R. Humphrey, D. K. Price, K. E. Spagnoli, A. L. Paolini, and E. J. Kelmelis. Cula: hybrid gpu accelerated linear algebra routines. volume 7705, page 770502. SPIE, 2010. 66

[16] K. Jittorntrum and M. R. Osborne. Strong uniqueness and second order convergence in non-linear discrete approximation. *Numerische Mathematik*, 34(4):439–455, 1980. 27

[17] C. Kim and R. M. Stem. Feature extraction for robust speech recognition using a power-law nonlinearity and power-bias subtraction. *10th Annual Conference of the International Speech Communication Association*, pages 28–31, 2009. 73

[18] R. M. Larsen. Lanczos bidiagonalization with partial reorthogonalization. Technical Report DAIMI PB-357, Dept. of Computer Science, Aarhus University, Sep 1998. 61

[19] Z. Lin. Some Software Packages for Partial SVD Computation. *Arxiv preprint arXiv:1108.1548*, Aug. 2011. 62, 66

[20] Z. Lin, M. Chen, and Y. Ma. The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices. *arXiv preprint*, Sept. 2010. 52, 55, 59, 60, 61, 62

[21] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. M. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. Technical Report UILU-ENG-09-2214, UIUC, Jul 2009. 25, 27, 52, 54, 55, 56, 57

[22] Z. Lin and S. Wei. A Block Lanczos with Warm Start Technique for Accelerating Nuclear Norm Minimization Algorithms. *Arxiv preprint arXiv:1012.0365*, Dec. 2010. 63

[23] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103:127–152, 2005. 69

[24] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Soviet Math. Dokl.*, 27(2):372–376, 1983. 52, 54

[25] Y. E. Nesterov. Gradient methods for minimizing composite objective function. Technical report, CORE, 2007. 52, 53, 54

[26] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma. Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 763 –770, june 2010. 26, 27, 77

[27] B. J. Smith. R package magma: Matrix algebra on gpu and multicore architectures, version 0.2.2, August 2010. http://icl.cs.utk.edu/magma/index.html. 66

[28] J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4):625–653, 1999. 46, 49

[29] M. Tao and X. Yuan. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization*, 21(1):57–81, 2011. 69

[30] K. C. Toh, M. J. Todd, and R. H. Tütüncü. Sdpt3 — a matlab software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11(1-4):545–581, 1999. 46, 49

[31] J. Wright, A. Ganesh, S. Rao, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. *Submitted to Journal of the ACM*, 2009. 50, 52

[32] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. *The HTK Book, version 3.4*. Cambridge University Engineering Department, 2006. 74

[33] Z. Zhang, X. Liang, A. Ganesh, and Y. Ma. Tilt: Transform invariant low-rank textures, 2011. 77

[34] Z. Zhou, X. Li, J. Wright, E. Candès, and Y. Ma. Stable principal component pursuit. In *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, pages 1518 –1522, june 2010. 24, 26, 67, 68