Chapter 3
Robust Principal Component Analysis

. . . any statistical procedure . . . should be robust in the sense that small deviations from the model assumptions should impair its performance only slightly . . . Somewhat larger deviations from the model should not cause a catastrophe.

—Peter J. Huber

In the previous chapter, we considered the PCA problem under the assumption that all the sample points are drawn from the same statistical or geometric model: a low-dimensional subspace. In practical applications, it is often the case that some entries of the data points can be missing or incomplete. For example, the 2-dimensional trajectories of an object moving in a video may become incomplete when the object becomes occluded. Sometimes, it could be the case that some entries of the data points are corrupted by gross errors and we do not know a priori which entries are corrupted. For instance, the intensities of some pixels of the face image of a person can be corrupted when the person is wearing glasses. Sometimes it could also be the case that a small subset of the data points are outliers. For instance, if we are trying to distinguish face images from non-face images, then we can model all face images as samples from a low-dimensional subspace, but non-face images will not follow the same model. Such data points that do not follow the model of interest are often called *sample outliers* and should be distinguished from the case of samples with some corrupted entries, also referred to as *intrasample outliers*. The main distinction to be made is that in the latter case, we do not want to discard the entire data point, but only the atypical entries.

In this chapter, we will introduce several techniques for recovering a low-dimensional subspace from missing or corrupted data. We will first consider the PCA problem with missing entries, also known as *incomplete PCA* or *low-rank matrix completion* (for linear subspaces). In Section 3.1, we will describe several representative methods for solving this problem based on maximum likelihood estimation, convex optimization, and alternating minimization. Such methods are featured due to their simplicity, optimality, or scalability, respectively. In Section 3.2,

R. Vidal et al., *Generalized Principal Component Analysis*, Interdisciplinary Applied Mathematics 40, DOI 10.1007/978-0-387-87811-9_3

we will consider the PCA problem with corrupted entries, also known as the *robust PCA* (RPCA) problem. We will introduce classical alternating minimization methods for addressing this problem as well as convex optimization methods that offer theoretical guarantees of correctness. Finally, in Section 3.3, we will consider the PCA problem with sample outliers and describe methods for solving this problem based on classical robust statistical estimation techniques as well as techniques based on convex relaxations. Face images will be used as examples to demonstrate the effectiveness of these algorithms.

3.1 PCA with Robustness to Missing Entries

Recall from Section 2.1.2 that in the PCA problem, we are given N data points $\mathcal{X} \doteq \{\boldsymbol{x}_j \in \mathbb{R}^D\}_{j=1}^N$ drawn (approximately) from a d-dimensional affine subspace $S \doteq \{\boldsymbol{x} = \boldsymbol{\mu} + U\boldsymbol{y}\}$, where $\boldsymbol{\mu} \in \mathbb{R}^D$ is an arbitrary point in S, $U \in \mathbb{R}^{D\times d}$ is a basis for S, and $\mathcal{Y} = \{\boldsymbol{y}_j \in \mathbb{R}^d\}_{j=1}^N$ are the principal components.

In this section, we consider the PCA problem in the case that some of the given data points are *incomplete*. A data point $\boldsymbol{x} = [x_1, x_2, \ldots, x_D]^\top$ is said to be incomplete when some of its entries are missing or unspecified. For instance, if the ith entry x_i, of $\boldsymbol{x}$ is missing, then $\boldsymbol{x}$ is known only up to a line in $\mathbb{R}^D$, i.e.,

$$\begin{aligned} \boldsymbol{x} \in L &\doteq \left\{[x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_D]^\top, x_i \in \mathbb{R}\right\} \\ &= \left\{\boldsymbol{x}_{-i} + x_i \boldsymbol{e}_i, x_i \in \mathbb{R}\right\}, \end{aligned} \tag{3.1}$$

where $\boldsymbol{x}_{-i} = [x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_D]^\top \in \mathbb{R}^D$ is the vector $\boldsymbol{x}$ with its ith entry zeroed out and $\boldsymbol{e}_i = [0, \ldots, 0, 1, 0, \ldots, 0]^\top \in \mathbb{R}^D$ is the ith basis vector. More generally, if the point $\boldsymbol{x}$ has M missing entries, without loss of generality we can partition it as $\begin{bmatrix} \boldsymbol{x}_U \\ \boldsymbol{x}_O \end{bmatrix}$, where $\boldsymbol{x}_U \in \mathbb{R}^M$ denotes the unobserved entries and $\boldsymbol{x}_O \in \mathbb{R}^{D-M}$ denotes the observed entries. Thus, $\boldsymbol{x}$ is known only up to the following M-dimensional affine subspace:

$$\boldsymbol{x} \in L \doteq \left\{ \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{x}_O \end{bmatrix} + \begin{bmatrix} I_M \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{x}_U, \boldsymbol{x}_U \in \mathbb{R}^M \right\}. \tag{3.2}$$

Incomplete PCA When the Subspace Is Known

Let us first consider the simplest case, in which the subspace S is known. Then we know that the point $\boldsymbol{x}$ belongs to both L and S. Therefore, given the parameters $\boldsymbol{\mu}$ and U of the subspace S, we can compute the principal components $\boldsymbol{y}$ and the missing entries $\boldsymbol{x}_U$ by intersecting L and S. In the case of one missing entry (illustrated in Figure 3.1), the intersection point can be computed from

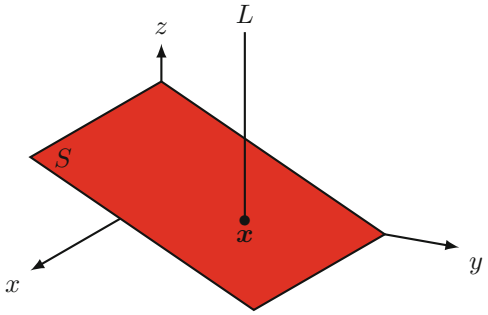


Fig. 3.1 Given a point $\boldsymbol{x} \in \mathbb{R}^D$ with one unknown entry x_i, the point $\boldsymbol{x}$ is known only up to a line L. However, if we also know that $\boldsymbol{x}$ belongs to a subspace S, we can find the unknown entry by intersecting L and S, provided that L is not parallel to S.

$$\boldsymbol{x} = \boldsymbol{x}_{-i} + x_i \boldsymbol{e}_i = \boldsymbol{\mu} + U\boldsymbol{y} \implies \begin{bmatrix} U & -\boldsymbol{e}_i \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ x_i \end{bmatrix} = \boldsymbol{x}_{-i} - \boldsymbol{\mu}. \tag{3.3}$$

Note that a necessary condition for this linear system to have a unique solution is that the line L is not parallel to the principal subspace, i.e., $\boldsymbol{e}_i \notin \operatorname{span}(U)$.

In the case of M missing entries, we can partition the point $\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_U \\ \boldsymbol{\mu}_O \end{bmatrix}$ and the subspace basis $U = \begin{bmatrix} U_U \\ U_O \end{bmatrix}$ according to $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_U \\ \boldsymbol{x}_O \end{bmatrix}$. Then, the intersection of L and S can be computed from

$$\begin{bmatrix} \boldsymbol{x}_U \\ \boldsymbol{x}_O \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_U \\ \boldsymbol{\mu}_O \end{bmatrix} + \begin{bmatrix} U_U \\ U_O \end{bmatrix} \boldsymbol{y} \implies \begin{bmatrix} U_U & -I_M \\ U_O & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x}_U \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\mu}_U \\ \boldsymbol{x}_O - \boldsymbol{\mu}_O \end{bmatrix}. \tag{3.4}$$

A necessary condition for the linear system in (3.4) to have a unique solution is that the matrix on the left-hand side be of full column rank $d + M \leq D$. This implies that $\boldsymbol{e}_i \notin \operatorname{span}(U)$ for each missing entry i. This also implies that $M \leq D - d$; hence we need to have at least d observed entries in order to complete a data point. When the data point $\boldsymbol{x}$ is not precise and has some noise, we can compute $\boldsymbol{y}$ and $\boldsymbol{x}_U$ as the solution to the following optimization problem:

$$\min_{\boldsymbol{y}, \boldsymbol{x}_U} \|\boldsymbol{x} - \boldsymbol{\mu} - U\boldsymbol{y}\|^2. \tag{3.5}$$

It is easy to derive that the closed-form solution to the unknowns $\boldsymbol{y}$ and $\boldsymbol{x}_U$ is given by

$$\begin{aligned} \boldsymbol{y} &= (I - U_U^\top U_U)^{-1} U_O^\top (\boldsymbol{x}_O - \boldsymbol{\mu}_O) = (U_O^\top U_O)^{-1} U_O^\top (\boldsymbol{x}_O - \boldsymbol{\mu}_O), \\ \boldsymbol{x}_U &= \boldsymbol{\mu}_U + U_U \boldsymbol{y} = \boldsymbol{\mu}_U + U_U (U_O^\top U_O)^{-1} U_O^\top (\boldsymbol{x}_O - \boldsymbol{\mu}_O). \end{aligned} \tag{3.6}$$

We leave the derivation to the reader as an exercise (see Exercise 3.1). Notice that this solution is simply the least squares solution to (3.4), and that in order for U_O to be of full rank (so that $U_O^\top U_O$ is invertible), we need to know at least d entries. Interestingly, the solution for $\boldsymbol{y}$ is obtained from the observed entries ($\boldsymbol{x}_O$) and the part of the model corresponding to the observed entries ($\boldsymbol{\mu}_O$ and U_O). Then the missing entries ($\boldsymbol{x}_U$) are obtained from the part of the model corresponding to the unobserved entries ($\boldsymbol{\mu}_U$ and U_U) and $\boldsymbol{y}$.

Incomplete PCA as a Well-Posed Problem
In practice, however, we do not know the subspace S (neither $\boldsymbol{\mu}$ nor U) a priori. Instead, we are given only N incomplete samples, which we can arrange as the columns of an incomplete data matrix $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D\times N}$. Let $W \in \mathbb{R}^{D\times N}$ be the matrix whose entries $\{w_{ij}\}$ encode the locations of the missing entries, i.e.,

$$w_{ij} = \begin{cases} 1 & \text{if } x_{ij} \text{ is known,} \\ 0 & \text{if } x_{ij} \text{ is missing,} \end{cases} \tag{3.7}$$

and let $W \odot X$ denote the Hadamard product of two matrices, which is defined as the entrywise product $(W \odot X)_{ij} = w_{ij}x_{ij}$. The goal of *PCA with missing data*, also known as *matrix completion*, is to find the missing entries $(\mathbf{1}\mathbf{1}^\top - W) \odot X$, the point $\boldsymbol{\mu}$, the basis U, and the matrix of low-dimensional coordinates $Y = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N] \in \mathbb{R}^{d\times N}$ from the known entries $W \odot X$.

Obviously, we cannot expect to always be able to find the correct solution to this problem. Whether the correct complete matrix X can be recovered depends on:

1. Which entries are missing or observed;
2. How many entries are missing or observed.

To see why the location of missing entries matters, suppose the first entry of all data points is missing. Then we cannot hope to be able to recover the first row of X at all. Likewise, suppose that all the entries of one data point are missing. While in this case we can hope to find the subspace from the other data points, we cannot recover the low-dimensional representation of the missing point. These two examples suggest that the location of missing entries should not have any conspicuous patterns.

Now suppose that the matrix X is

$$X = \boldsymbol{e}_1\boldsymbol{e}_1^\top = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{3.8}$$

which is a rank-one matrix. In this case, we cannot hope to recover X even if a relatively large percentage of its entries are given, because most entries are equal to zero, and we will not be able to distinguish X from the zero matrix from many

observed entries. This suggests that if we want to recover a low-rank data matrix from a small portion of its entries, the matrix itself should not be too sparse.

Thus, to avoid ambiguous solutions due to the above situations, we must require that the locations of the missing entries be random enough so that the chance that they form a conspicuous pattern is very low; and in addition, we must restrict our low-rank matrices to those that are not particularly sparse. The following definition gives a set of technical conditions to impose on a matrix so that its singular vectors are not too spiky, and hence the matrix itself is not too sparse.

Definition 3.1 (Matrix Incoherence with Respect to Sparse Matrices)**.** *A matrix $X \in \mathbb{R}^{D \times N}$ is said to be ν-*incoherent with respect to the set of sparse matrices *if*

$$\max_i \|\boldsymbol{u}_i\|_2 \leq \frac{\nu\sqrt{d}}{\sqrt{D}}, \quad \max_j \|\boldsymbol{v}_j\|_2 \leq \frac{\nu\sqrt{d}}{\sqrt{N}}, \quad \|UV^\top\|_\infty \leq \frac{\nu\sqrt{d}}{\sqrt{DN}}, \tag{3.9}$$

where d is the rank of X, $X = U\Sigma V^\top$ is the compact SVD of X, and $\boldsymbol{u}_i$, and $\boldsymbol{v}_j$ are the ith row of U and jth row V, respectively.

Notice that since U is orthonormal, the largest absolute value of the entries of $U \in \mathbb{R}^{D \times d}$ is equal to 1, which happens when a column of U is 1-sparse, i.e., when a column of U has only one nonzero entry. On the other hand, if each column of U is so dense that all its entries are equal to each other up to sign, then each entry is equal to $\pm 1/\sqrt{D}$, and the norm of each row is $\sqrt{d/D}$. Therefore, when $\nu < 1$, the first condition above controls the level of sparsity of U. Similarly, the other two conditions control the levels of sparsity of V and $UV^\top$, respectively. From a probabilistic perspective, these conditions are rather mild in the sense that they hold for almost all generic matrices—a random (say Gaussian) matrix satisfies these conditions with high probability when the dimension of the matrix is large enough. As we will see, incoherence is indeed a very useful technical condition to ensure that low-rank matrix completion is a meaningful problem.

Regarding the number of entries required, notice that in order to specify a d-dimensional subspace S in $\mathbb{R}^D$ together with N points on it, we need to specify $D + dD + dN - d^2$ independent entries in $\boldsymbol{\mu}$, U, and Y.[1] That is, it is necessary to observe at least this number of entries of X in order to have a unique solution for X. However, the sufficient conditions for ensuring a unique and correct solution highly depend on the approach and method one uses to recover X.

Incomplete PCA Algorithms

In what follows, we discuss a few approaches for solving the PCA problem with missing entries. The first approach (described in Section 3.1.1) is a simple extension

[1] If $U \in \mathbb{R}^{D \times d}$ and $V \in \mathbb{R}^{N \times d}$, then U and V have $dD + dN$ degrees of freedom in general. However, to specify the subspace, it suffices to specify $UV^\top$, which is equal to $UAA^{-1}V^\top$ for every invertible matrix $A \in \mathbb{R}^{d \times d}$; hence the matrix $UV^\top$ has $dD + dN - d^2$ degrees of freedom.

of geometric PCA (see Section 2.1) in which the sample mean and covariance are directly computed from the incomplete data matrix. However, this approach has a number of disadvantages, as we shall see. The second approach (described in Section 3.1.2) is a direct extension of probabilistic PCA (see Section 2.2) and uses the expectation maximization (EM) algorithm (see Appendix B.2.1) to complete the missing entries. While this approach is guaranteed to converge, the solution it finds is not always guaranteed to be the global optimum, and hence it is not necessarily the correct solution. The third approach (described in Section 3.1.3) uses convex relaxation and optimization techniques to find the missing entries of the low-rank data matrix X. Under the above incoherent conditions and with almost minimal observations, this approach is guaranteed to return a perfect completion of the low-rank matrix. However, this approach may not be scalable to large matrices, since it requires solving for as many variables as the number of entries in the data matrix. The fourth and final approach (described in Section 3.1.4) alternates between solving for $\boldsymbol{\mu}$, U, and Y given a completion of X, and solving for the missing entries of X given $\boldsymbol{\mu}$, U, and Y. Since this method uses a minimal parameterization of the unknowns, it is more scalable. While in general, this approach is not guaranteed to converge to the correct solution, we present a variant of this method that is guaranteed to recover the missing entries correctly under conditions similar to those for the convex relaxation method.

3.1.1 Incomplete PCA by Mean and Covariance Completion

Recall from Section 2.1.2 that the optimization problem associated with geometric PCA is

$$\min_{\boldsymbol{\mu}, U, \{\boldsymbol{y}_j\}} \sum_{j=1}^{N} \left\| \boldsymbol{x}_j - \boldsymbol{\mu} - U\boldsymbol{y}_j \right\|^2 \text{ s.t. } U^\top U = I_d \text{ and } \sum_{j=1}^{N} \boldsymbol{y}_j = \mathbf{0}. \tag{3.10}$$

We already know that the solution to this problem can be obtained from the mean and covariance of the data points,

$$\hat{\boldsymbol{\mu}}_N = \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{x}_j \quad \text{and} \quad \hat{\Sigma}_N = \frac{1}{N} \sum_{j=1}^{N} (\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N)(\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N)^\top, \tag{3.11}$$

respectively. Specifically, $\boldsymbol{\mu}$ is given by the sample mean $\hat{\boldsymbol{\mu}}_N$, U is given by the top d eigenvectors of the covariance matrix $\hat{\Sigma}_N$, and $\boldsymbol{y}_j = U^\top(\boldsymbol{x}_j - \boldsymbol{\mu})$. Alternatively, an optimal solution can be found from the rank-d SVD of the mean-subtracted data matrix $[\boldsymbol{x}_1 - \hat{\boldsymbol{\mu}}_N, \ldots, \boldsymbol{x}_N - \hat{\boldsymbol{\mu}}_N]$, as shown in Theorem 2.3.

When some entries of each $\boldsymbol{x}_j$ are missing, we cannot directly compute $\hat{\boldsymbol{\mu}}_N$ or $\hat{\Sigma}_N$ as in (3.11). A straightforward method for dealing with missing entries was

introduced in (Jolliffe 2002). It basically proposes to compute the sample mean and covariance from the known entries of X. Specifically, the entries of the incomplete mean and covariance can be computed as

$$\hat{\mu}_i = \frac{\sum_{j=1}^{N} w_{ij} x_{ij}}{\sum_{j=1}^{N} w_{ij}} \quad \text{and} \quad \hat{\sigma}_{ik} = \frac{\sum_{j=1}^{N} w_{ij} w_{kj} (x_{ij} - \hat{\mu}_i)(x_{kj} - \hat{\mu}_k)}{\sum_{j=1}^{N} w_{ij} w_{kj}}, \tag{3.12}$$

where $i, k = 1, \ldots, D$. However, as discussed in (Jolliffe 2002), this simple approach has several disadvantages. First, the estimated covariance matrix need not be positive semidefinite. Second, these estimates are not obtained by optimizing any statistically or geometrically meaningful objective function (least squares, maximum likelihood, etc.) Nonetheless, estimates $\hat{\boldsymbol{\mu}}_N$ and $\hat{\Sigma}_N$ obtained from the naive approach in (3.12) may be used to initialize the methods discussed in the next two sections, which are iterative in nature. For example, we may initialize the columns of U as the eigenvectors of $\hat{\Sigma}_N$ associated with its d largest eigenvalues. Then given $\hat{\boldsymbol{\mu}}_N$ and $\hat{U}$, we can complete each missing entry as described in (3.6).

3.1.2 Incomplete PPCA by Expectation Maximization

In this section, we derive an EM algorithm (see Appendix B.2.1) for solving the PPCA problem with missing data. Recall from Section 2.2 that in the PPCA model, each data point is drawn as $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}}, \Sigma_{\boldsymbol{x}})$, where $\boldsymbol{\mu}_{\boldsymbol{x}} = \boldsymbol{\mu}$ and $\Sigma_{\boldsymbol{x}} = BB^\top + \sigma^2 I_D$, where $\boldsymbol{\mu} \in \mathbb{R}^D$, $B \in \mathbb{R}^{D \times d}$, and $\sigma > 0$. Recall also from (2.56) that the log-likelihood of the PPCA model is given by

$$\mathscr{L} = -\frac{ND}{2} \log(2\pi) - \frac{N}{2} \log \det(\Sigma_{\boldsymbol{x}}) - \frac{1}{2} \sum_{j=1}^{N} \operatorname{trace}(\Sigma_{\boldsymbol{x}}^{-1} (\boldsymbol{x}_j - \boldsymbol{\mu})(\boldsymbol{x}_j - \boldsymbol{\mu})^\top), \tag{3.13}$$

where $\{\boldsymbol{x}_j\}_{j=1}^N$ are N i.i.d. samples of $\boldsymbol{x}$. Since the samples are incomplete, we can partition each point $\boldsymbol{x}$ and the parameters $\boldsymbol{\mu}_{\boldsymbol{x}}$ and $\Sigma_{\boldsymbol{x}}$ as

$$\begin{bmatrix} \boldsymbol{x}_U \\ \boldsymbol{x}_O \end{bmatrix} = P\boldsymbol{x}, \quad \begin{bmatrix} \boldsymbol{\mu}_U \\ \boldsymbol{\mu}_O \end{bmatrix} = P\boldsymbol{\mu}, \quad \text{and} \quad \begin{bmatrix} \Sigma_{UU} & \Sigma_{UO} \\ \Sigma_{OU} & \Sigma_{OO} \end{bmatrix} = P \Sigma_{\boldsymbol{x}} P^\top. \tag{3.14}$$

Here $\boldsymbol{x}_O$ is the observed part of $\boldsymbol{x}$, $\boldsymbol{x}_U$ is the unobserved part of $\boldsymbol{x}$, and P is any permutation matrix that reorders the entries of $\boldsymbol{x}$ so that the unobserved entries appear first. Notice that P is not unique, but we can use any such P. Notice also that the above partition of $\boldsymbol{x}$, $\boldsymbol{\mu}_{\boldsymbol{x}}$, and $\Sigma_{\boldsymbol{x}}$ could be different for each data point, because the missing entries could be different for different data points. When strictly necessary, we will use $\boldsymbol{x}_{jU}$ and $\boldsymbol{x}_{jO}$ to denote the unobserved and observed parts of

point $\boldsymbol{x}_j$, respectively, and P_j to denote the permutation matrix. Otherwise, we will avoid using the index j in referring to a generic point.

In what follows, we derive two variants of the EM algorithm for learning the parameters $\theta = (\boldsymbol{\mu}, B, \sigma)$ of the PPCA model from incomplete samples $\{\boldsymbol{x}_j\}_{j=1}^N$. The first variant, called Maximum a Posteriori Expectation Maximization (MAP-EM), is an approximate EM method whereby the unobserved variables are given by their MAP estimates (see Appendix B.2.2). The second variant is the exact EM algorithm (see Appendix B.2.1), where we take the conditional expectation of $\mathscr{L}$ over the incomplete entries. Interestingly, both variants lead to the same estimate for $\boldsymbol{\mu}_{\boldsymbol{x}}$, though the estimates for $\Sigma_{\boldsymbol{x}}$ are slightly different. In our derivations, we will use the fact that the conditional distribution of $\boldsymbol{x}_U$ given $\boldsymbol{x}_O$ is Gaussian. More specifically, $\boldsymbol{x}_U \mid \boldsymbol{x}_O \sim \mathcal{N}(\boldsymbol{\mu}_{U|O}, \Sigma_{U|O})$, where

$$\boldsymbol{\mu}_{U|O} = \boldsymbol{\mu}_U + \Sigma_{UO}\Sigma_{OO}^{-1}(\boldsymbol{x}_O - \boldsymbol{\mu}_O) \;\text{ and }\; \Sigma_{U|O} = \Sigma_{UU} - \Sigma_{UO}\Sigma_{OO}^{-1}\Sigma_{OU}.$$

We leave this fact as an exercise to the reader (see Exercise 3.2).

Maximum a Posteriori Expectation Maximization (MAP-EM)
The MAP-EM algorithm (see Appendix B.2.2) is a simplified version of the EM algorithm (see Appendix B.2.1) that alternates between the following two steps:

MAP-step: Complete each data point $\boldsymbol{x}$ by replacing the unobserved variables $\boldsymbol{x}_U$ with their MAP estimates, $\arg\max_{\boldsymbol{x}_U} p_{\theta^k}(\boldsymbol{x}_U \mid \boldsymbol{x}_O)$, where θ^k is an estimate for the model parameters at iteration k.

M-step: Maximize the complete log-likelihood with respect to θ, with $\boldsymbol{x}_U$ given as in the MAP-step.

During the MAP step, the MAP estimate of the unobserved variables can be computed in closed form as

$$\arg\max_{\boldsymbol{x}_U} p_{\theta^k}(\boldsymbol{x}_U \mid \boldsymbol{x}_O) = \boldsymbol{\mu}_{U|O}^k = \boldsymbol{\mu}_U^k + \Sigma_{UO}^k(\Sigma_{OO}^k)^{-1}(\boldsymbol{x}_O - \boldsymbol{\mu}_O^k). \tag{3.15}$$

Therefore, we can complete each data point as $\boldsymbol{x}^k = P^\top \begin{bmatrix} \boldsymbol{\mu}_{U|O}^k \\ \boldsymbol{x}_O \end{bmatrix}$. Letting $\boldsymbol{x}_j^k$ be the completion of $\boldsymbol{x}_j$ at iteration k, we obtain the complete log-likelihood as

$$\mathscr{L} = -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log\det(\Sigma_{\boldsymbol{x}}) - \frac{1}{2}\sum_{j=1}^{N}(\boldsymbol{x}_j^k - \boldsymbol{\mu})^\top \Sigma_{\boldsymbol{x}}^{-1}(\boldsymbol{x}_j^k - \boldsymbol{\mu}). \tag{3.16}$$

During the M-step, we need to maximize $\mathscr{L}$ with respect to θ. Since the data are already complete, we can update the model parameters as described in Theorem 2.9, i.e.,

$$\boldsymbol{\mu}^{k+1} = \frac{1}{N}\sum_{j=1}^{N}\boldsymbol{x}_j^k, \;\; B^{k+1} = U_1\big(\Lambda_1 - (\sigma^k)^2 I\big)^{1/2} R, \;\text{ and }\; (\sigma^k)^2 = \frac{\sum_{i=d+1}^{D}\lambda_i}{D-d},$$

where $U_1 \in \mathbb{R}^{D\times d}$ is the matrix whose columns are the top d eigenvectors of the *complete sample covariance* matrix

$$\hat{\Sigma}_N^{k+1} = \frac{1}{N}\sum_{j=1}^{N}(\boldsymbol{x}_j^k - \boldsymbol{\mu}^{k+1})(\boldsymbol{x}_j^k - \boldsymbol{\mu}^{k+1})^\top, \tag{3.17}$$

$\Lambda_1 \in \mathbb{R}^{d\times d}$ is a diagonal matrix with the top d eigenvalues of $\hat{\Sigma}_N^{k+1}$, $R \in \mathbb{R}^{d\times d}$ is an arbitrary orthogonal matrix, and λ_i is the ith-largest eigenvalue of $\hat{\Sigma}_N^{k+1}$. We can then update the covariance matrix as $\Sigma_{\boldsymbol{x}}^{k+1} = B^{k+1}(B^{k+1})^\top + (\sigma^k)^2 I$.

Expectation Maximization (EM)
The EM algorithm (see Appendix B.2.1) alternates between the following steps:

E-step: Compute the expectation $Q(\theta \mid \theta^k) \doteq \mathbb{E}_{\boldsymbol{x}_U}[\mathscr{L} \mid \boldsymbol{x}_O, \theta^k]$ of the complete log-likelihood $\mathscr{L}$ with respect to the missing entries $\boldsymbol{x}_U$ given the observed entries $\boldsymbol{x}_O$ and an estimate θ^k of the parameters at iteration k.

M-step: Maximize the expected completed log-likelihood $\mathbb{E}_{\boldsymbol{x}_U}[\mathscr{L} \mid \boldsymbol{x}_O, \theta^k]$ with respect to θ.

Observe from (3.13) that to compute the expectation of $\mathscr{L}$, it suffices to compute the following matrix for each incomplete data point $\boldsymbol{x}$:

$$S^k = \mathbb{E}_{\boldsymbol{x}_U}[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^\top \mid \boldsymbol{x}_O, \theta^k] = P^\top \begin{bmatrix} S_{UU}^k & S_{UO}^k \\ S_{OU}^k & S_{OO}^k \end{bmatrix} P. \tag{3.18}$$

Each block of this matrix can be computed as

$$\begin{aligned}
S_{OO}^k &= \mathbb{E}[(\boldsymbol{x}_O-\boldsymbol{\mu}_O)(\boldsymbol{x}_O-\boldsymbol{\mu}_O)^\top \mid \boldsymbol{x}_O, \theta^k] = (\boldsymbol{x}_O-\boldsymbol{\mu}_O^k)(\boldsymbol{x}_O-\boldsymbol{\mu}_O^k)^\top, \\
S_{UO}^k &= \mathbb{E}[(\boldsymbol{x}_U-\boldsymbol{\mu}_U)(\boldsymbol{x}_O-\boldsymbol{\mu}_O)^\top \mid \boldsymbol{x}_O, \theta^k] = (\boldsymbol{\mu}_{U|O}^k-\boldsymbol{\mu}_U^k)(\boldsymbol{x}_O-\boldsymbol{\mu}_O^k)^\top = (S_{OU}^k)^\top, \\
S_{UU}^k &= \mathbb{E}[(\boldsymbol{x}_U-\boldsymbol{\mu}_U)(\boldsymbol{x}_U-\boldsymbol{\mu}_U)^\top \mid \boldsymbol{x}_O, \theta^k] \\
&= \mathbb{E}[(\boldsymbol{x}_U-\boldsymbol{\mu}_{U|O}^k)(\boldsymbol{x}_U-\boldsymbol{\mu}_{U|O}^k)^\top \mid \boldsymbol{x}_O, \theta^k] + \\
&\quad 2\mathbb{E}[(\boldsymbol{\mu}_{U|O}^k - \boldsymbol{\mu}_U)(\boldsymbol{x}_U - \boldsymbol{\mu}_{U|O}^k)^\top \mid \boldsymbol{x}_O, \theta^k] + (\boldsymbol{\mu}_{U|O}^k-\boldsymbol{\mu}_U)(\boldsymbol{\mu}_{U|O}^k-\boldsymbol{\mu}_U)^\top \\
&= \Sigma_{U|O}^k + (\boldsymbol{\mu}_{U|O}^k-\boldsymbol{\mu}_U)(\boldsymbol{\mu}_{U|O}^k-\boldsymbol{\mu}_U)^\top.
\end{aligned}$$

Let S_j^k denote the matrix S^k associated with point $\boldsymbol{x}_j$ and let $\hat{\Sigma}_N^k = \frac{1}{N}\sum_{j=1}^N S_j^k$. Then the expected complete log-likelihood is given by

$$Q(\theta \mid \theta^k) = -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log\det(\Sigma_{\boldsymbol{x}}) - \frac{N}{2}\operatorname{trace}(\Sigma_{\boldsymbol{x}}^{-1}\hat{\Sigma}_N^k). \tag{3.19}$$

In the M-step, we need to maximize this quantity with respect to θ. Notice that this quantity is almost identical to that in (2.56), except that the sample covariance

matrix $\hat{\Sigma}_N$ is replaced by $\hat{\Sigma}_N^k$. Thus, if $\hat{\Sigma}_N^k$ did not depend on the unknown parameter $\boldsymbol{\mu}$, we could immediately compute B and σ from Theorem 2.9. Therefore, all we need to do is to show how to compute $\boldsymbol{\mu}$. To this end, notice that

$$\frac{\partial}{\partial \boldsymbol{\mu}} \operatorname{trace}(\Sigma_{\boldsymbol{x}}^{-1} S^k) = \frac{\partial}{\partial \boldsymbol{\mu}} \mathbb{E}[(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma_{\boldsymbol{x}}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \mid \boldsymbol{x}_O, \theta^k] \tag{3.20}$$

$$= -2\Sigma_{\boldsymbol{x}}^{-1}\mathbb{E}[\boldsymbol{x}-\boldsymbol{\mu} \mid \boldsymbol{x}_O, \theta^k] = -2\Sigma_{\boldsymbol{x}}^{-1}(\boldsymbol{x}^k - \boldsymbol{\mu}), \tag{3.21}$$

where $\boldsymbol{x}^k = P^\top \begin{bmatrix} \boldsymbol{\mu}_{U|O}^k \\ \boldsymbol{x}_O \end{bmatrix}$ is the complete data point. Therefore,

$$\frac{\partial}{\partial \boldsymbol{\mu}} Q(\theta \mid \theta^k) = -\frac{1}{2}\frac{\partial}{\partial \boldsymbol{\mu}} \sum_{j=1}^{N} \operatorname{trace}(\Sigma_{\boldsymbol{x}}^{-1} S_j^k) = \sum_{j=1}^{N} \Sigma_{\boldsymbol{x}}^{-1}(\boldsymbol{x}_j^k - \boldsymbol{\mu}) = \mathbf{0}, \tag{3.22}$$

and so the optimal $\boldsymbol{\mu}$ is

$$\boldsymbol{\mu}^{k+1} = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{x}_j^k. \tag{3.23}$$

Notice that this solution is the same as that of the MAP-EM algorithm. That is, the optimal solution for $\boldsymbol{\mu}$ is the average of the complete data. We can then form the matrix $\hat{\Sigma}_N^k$ and compute B^{k+1} and σ^{k+1} as before. Notice, however, that $\hat{\Sigma}_N^k$ is not the covariance of the complete data. The key difference is in the term S_{UU}^k, which contains an additional term $\Sigma_{U|O}^k$.

The EM algorithm for PPCA with missing data is summarized in Algorithm 3.1. In step 2 of the algorithm, the missing entries of X are filled in with zeros, and an initial estimate of $\boldsymbol{\mu}$ and $\Sigma_{\boldsymbol{x}}$ is obtained from the zero-filled X. Alternatively, one may use other initialization methods, such as the mean and covariance completion method described in Section 3.1.1. In step 7, the missing entries of each $\boldsymbol{x}_j$ are filled in according to the initial estimates of mean and covariance in step 2, while the observed entries are kept intact. This corresponds to the MAP step of the MAP-EM algorithm, and is an intermediate calculation for the E-step of the EM algorithm. Next, steps 9 and 10 update the mean and covariance of the PPCA model. Step 9 is common to both the MAP-EM and EM algorithms, while step 10 is slightly different: the MAP-EM algorithm uses only the first term on the right-hand side of step 10, while the EM algorithm uses both terms. Steps 11–14 update the parameters of the PPCA model and correspond to the M-step of both the MAP-EM and EM algorithms. Finally, step 16 computes the probabilistic principal components. Recall from Section 2.2, equation (2.78), that given the parameters of the PPCA model $(\boldsymbol{\mu}, B, \sigma)$, the probabilistic principal components of a vector $\boldsymbol{x}$ are given by $\boldsymbol{y} = (B^\top B + \sigma^2 I)^{-1} B^\top (\boldsymbol{x} - \boldsymbol{\mu})$.

Algorithm 3.1 (Incomplete PPCA by Expectation Maximization)

Input: Entries x_{ij} of a matrix $X \in \mathbb{R}^{D\times N}$ for $(i,j) \in \Omega$ and dimension d.

initialize
$x_{ij} \leftarrow 0$ for $(i,j) \notin \Omega$, $\boldsymbol{\mu} \leftarrow \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{x}_j$, and $\Sigma \leftarrow \frac{1}{N}\sum_{j=1}^{N} (\boldsymbol{x}_j - \boldsymbol{\mu})(\boldsymbol{x}_j - \boldsymbol{\mu})^\top$.
$P_j \leftarrow$ any permutation matrix that sorts the entries of the jth column of X, $\boldsymbol{x}_j$, so that its unobserved entries (as specified in Ω) appear first.
$\begin{bmatrix} \boldsymbol{x}_U^j \\ \boldsymbol{x}_O^j \end{bmatrix} \leftarrow P_j \boldsymbol{x}_j, \quad \begin{bmatrix} \boldsymbol{\mu}_U^j \\ \boldsymbol{\mu}_O^j \end{bmatrix} \leftarrow P_j \boldsymbol{\mu}, \quad \text{and} \quad \begin{bmatrix} \Sigma_{UU}^j & \Sigma_{UO}^j \\ \Sigma_{OU}^j & \Sigma_{OO}^j \end{bmatrix} \leftarrow P_j \Sigma P_j^\top$.
repeat
  for all $j = 1, \ldots, N$ **do**
    $\boldsymbol{x}_j \leftarrow P_j^\top \begin{bmatrix} \boldsymbol{\mu}_U^j + \Sigma_{UO}^j (\Sigma_{OO}^j)^{-1} (\boldsymbol{x}_O^j - \boldsymbol{\mu}_O^j) \\ \boldsymbol{x}_O^j \end{bmatrix}$.
  end for
  $\boldsymbol{\mu} \leftarrow \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{x}_j$ and $\Sigma \leftarrow \frac{1}{N}\sum_{j=1}^{N} (\boldsymbol{x}_j - \boldsymbol{\mu})(\boldsymbol{x}_j - \boldsymbol{\mu})^\top$.
  $S \leftarrow \Sigma + P_j^\top \begin{bmatrix} \Sigma_{UU}^j - \Sigma_{UO}^j (\Sigma_{OO}^j)^{-1} \Sigma_{OU}^j & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} P_j$.
  $U_1 \leftarrow$ top d eigenvectors of S.
  $\Lambda_1 \leftarrow$ top d eigenvalues of S.
  $\sigma^2 \leftarrow \frac{1}{D-d}\sum_{i=d+1}^{D} \lambda_i(S)$.
  $B \leftarrow U_1(\Lambda_1 - \sigma^2 I)^{1/2} R$, where $R \in \mathbb{R}^{d\times d}$ is an arbitrary orthogonal matrix.
until convergence of $\boldsymbol{\mu}$ and S.
$Y \leftarrow (B^\top B + \sigma^2 I)^{-1} B^\top (X - \boldsymbol{\mu}\mathbf{1}^\top)$.

Output: $\boldsymbol{\mu}$, B, and Y.

3.1.3 Matrix Completion by Convex Optimization

The EM-based approaches to incomplete PPCA discussed in the previous section rely on (a) explicit parameterizations of the low-rank factors and (b) minimization of a nonconvex cost function in an alternating minimization fashion. Specifically, such approaches alternate between completing the missing entries given the parameters of a PPCA model for the data and estimating the parameters of the model from complete data. While simple and intuitive, such approaches suffer from two important disadvantages. First, the desired rank of the matrix needs to be known in advance. Second, due to the greedy nature of the EM algorithm, it is difficult to ensure convergence to the globally optimal solution. Therefore, a good initialization of the EM-based algorithm is critical for converging to a good solution.

In this section, we introduce an alternative approach that solves the low-rank matrix completion problem via a convex relaxation. As we will see, this approach allows us to complete a low-rank matrix by minimizing a convex objective function, which is guaranteed to have a globally optimal minimizer. Moreover, under rather

benign conditions on the missing entries, the global minimizer is guaranteed to be the correct low-rank matrix, even without knowing the rank of the matrix in advance.

A rigorous justification for the correctness of the convex relaxation approach requires a deep knowledge of high-dimensional statistics and geometry that is beyond the scope of this book. However, this does not prevent us from introducing and summarizing here the main ideas and results, as well as the basic algorithms offered by this approach. Practitioners can apply the useful algorithm to their data and problems, whereas researchers who are more interested in the advanced theory behind the algorithm may find further details in (Cai et al. 2008; Candès and Recht 2009; Candès and Tao 2010; Gross 2011; Keshavan et al. 2010a; Zhou et al. 2010a).

Compressive Sensing of Low-Rank Matrices

The matrix completion problem can be considered a special case of the more general class of problems of recovering a high-dimensional low-rank matrix X from highly compressive linear measurements $B = \mathcal{P}(X)$, where $\mathcal{P}$ is a linear operator that returns a set of linear measurements B of the matrix X. It is known from high-dimensional statistics that if the linear operator $\mathcal{P}$ satisfies certain conditions, then the rank minimization problem

$$\min_{A} \quad \text{rank}(A) \quad \text{s.t.} \quad \mathcal{P}(A) = B \tag{3.24}$$

is well defined, and its solution is unique (Candès and Recht 2009). However, it is also known that under general conditions, the task of finding such a minimal-rank solution is in general an NP-hard problem.

To alleviate the computational difficulty, instead of directly minimizing the discontinuous rank function, we could try to relax the objective and minimize its convex surrogate instead. More precisely, we could try to solve the following relaxed convex optimization problem

$$\min_{A} \quad \|A\|_* \quad \text{s.t.} \quad \mathcal{P}(A) = B, \tag{3.25}$$

where $\|A\|_*$ is the nuclear norm of the matrix A (i.e., the sum of all singular values of A). The theory of high-dimensional statistics (Candès and Recht 2009; Gross 2011) shows that when X is high-dimensional and the measurement operator $\mathcal{P}(\cdot)$ satisfies certain benign conditions,[2] the solution to the convex optimization problem (3.25) coincides with that of the rank minimization problem in (3.24).

In what follows, we illustrate how to apply this general approach to the low-rank matrix completion problem, derive a simple algorithm, and give precise conditions under which the algorithm gives the correct solution.

Exact Low-Rank Matrix Completion with Minimum Number of Measurements

[2]Such conditions typically require that the linear measurements and the matrix X be in some sense *incoherent*.

Let $X \in \mathbb{R}^{D\times N}$ be a matrix whose columns are drawn from a low-dimensional subspace of $\mathbb{R}^D$ of dimension $d \ll D$. Assume that we observe only a subset of the entries of X indexed by a set Ω, i.e.,

$$\Omega = \{(i,j) : x_{ij} \text{ is observed}\}. \tag{3.26}$$

Let $\mathcal{P}_\Omega : \mathbb{R}^{D\times N} \to \mathbb{R}^{D\times N}$ be the orthogonal projector onto the span of all matrices vanishing outside of Ω so that the (i,j)th component of $\mathcal{P}_\Omega(X)$ is equal to x_{ij} if $(i,j) \in \Omega$ and zero otherwise. As proposed in (Candès and Recht 2009), we may complete the missing entries in X by searching for a complete matrix $A \in \mathbb{R}^{D\times N}$ that is of low rank and coincides with X in Ω. This leads to the following optimization problem:

$$\min_A \quad \text{rank}(A) \quad \text{s.t.} \quad \mathcal{P}_\Omega(A) = \mathcal{P}_\Omega(X). \tag{3.27}$$

As we have discussed before in Section 3.1, in order for this problem to have a unique solution, we must require that the matrix X be nonsparse, or incoherent according to Definition 3.1. In addition, the missing entries should be random enough and should not fall into any special pattern.

Regarding the minimal number of entries needed, let us assume $D = N$ for simplicity. An $N\times N$ matrix X of rank d has $2Nd - d^2$ degrees of freedom.[3] Therefore, one should not expect to complete or recover a rank-d matrix uniquely with fewer than $O(dN)$ entries, since in general, there will be infinitely many rank-d matrices that have the same given entries.

The question is how many more entries are needed in order for the above problem to have a unique solution and, even more importantly, for the solution to be found efficiently. Since the above rank-minimization problem is NP-hard (even if the solution exists and is unique), inspired by the compressive sensing story, we consider the following convex relaxation:

$$\min_A \quad \|A\|_* \quad \text{s.t.} \quad \mathcal{P}_\Omega(A) = \mathcal{P}_\Omega(X), \tag{3.28}$$

where $\|A\|_* = \sum \sigma_i(A)$ is the sum of the singular values of A, which is the convex envelope of the rank function $\text{rank}(A)$.

The seminal work of (Candès and Recht 2009; Candès and Tao 2010; Gross 2011) has established that when the low-rank matrix X is incoherent and the locations of the known entries are sampled uniformly at random, the minimizer to the problem (3.28) is unique and equal to the correct matrix X even if the number of given entries is barely above the minimum. More specifically, the minimum number of measurements that are needed in order for the convex optimization to give the

[3] X can be factorized as $X = UAA^{-1}V^\top$, where $U, V \in \mathbb{R}^{N\times d}$ have Nd entries each, and $A \in \mathbb{R}^{d\times d}$ is an invertible matrix.

correct solution with high probability is very close to the number of degrees of freedom of the unknowns. The following theorem summarizes the results.

Theorem 3.2 (Low-Rank Matrix Completion by Convex Optimization)**.** *Let X be a $D \times N$ matrix of rank d, with $N \geq D$. Assume that X is ν-incoherent with respect to the set of sparse matrices according to Definition 3.1. Let M be the expected number of observed entries, whose locations are sampled independently and uniformly at random.*[4] *Then there is a numerical constant c such that if*

$$M \geq c\, \nu^4 d\, N(\log(N))^2, \tag{3.29}$$

then X is the unique solution to the problem in (3.28) with probability at least $1 - N^{-3}$; that is, the program (3.28) recovers all the entries of X with no error.

Notice that for a general rank-d matrix, this bound is already very tight. To see this, recall from our previous discussion that the minimum number of required measurements is $O(d\, N)$. In essence, the theorem states that with only a polylog factor[5] of extra measurements, i.e., $O(d\, N\, \text{polylog}(N))$, we can obtain the unique correct solution via convex optimization. This bound can be strengthened under additional assumptions. For instance, if $d = O(1)$ (i.e., if X is a matrix whose rank does not increase with its dimension), then the minimum number of entries needed to guarantee the exact completion of X reduces to $M \geq N\, \log(N)$ (Keshavan et al. 2010a). It is worth mentioning that the above statement is not limited to matrix completion. As shown in (Gross 2011), the same bound and statement hold for the compressive sensing of low-rank matrices with general linear observations $\mathcal{P}(X)$, i.e., for the problem (3.25), as long as the linear operator $\mathcal{P}$ is "incoherent" with the matrix X.

Low-Rank Matrix Completion via Proximal Gradient

The work of (Cai et al. 2008) proposes to find the solution to the optimization problem in (3.28) by solving the following problem:

$$\min_{A} \quad \tau \|A\|_* + \frac{1}{2}\|A\|_F^2 \quad \text{s.t.} \quad \mathcal{P}_\Omega(A) = \mathcal{P}_\Omega(X), \tag{3.30}$$

[4] Previously, we have used M to denote the number of observed entries in a specific matrix X. Notice that here, M is the expected number of observed entries under a random model in which the locations are sampled independently and uniformly at random. Thus, if p is the probability that an entry is observed, then the expected number of observed entries is pDN. Therefore, one can state the result either in terms of p or in terms of the expected number of observed entries, as we have done. For ease of exposition, we will continue to refer to M as the number of observed entries in the main text, but the reader is reminded that all the theoretical results refer to the expected number of observed entries, because the model for the observed entries is random.

[5] A polylog factor means a polynomial in the log function, i.e., $O(\text{polylog}(N))$ means $O(\log(N)^k)$ for some integer k.

in which the nuclear norm is augmented with a quadratic penalty term on A. As we will see, the additional quadratic term leads to a very simple algorithm. Furthermore, one can show that as the weight $\tau > 0$ increases, the solution of this regularized program converges to that of (3.28) (Cai et al. 2008).

More specifically, using the method of Lagrange multipliers described in Appendix A, we can write the Lagrangian function of (3.30) as

$$\mathscr{L}(A, Z) = \tau \|A\|_* + \frac{1}{2}\|A\|_F^2 + \langle Z, \mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(A)\rangle, \tag{3.31}$$

where $Z \in \mathbb{R}^{D\times N}$ is a matrix of Lagrange multipliers. The optimal solution is given by the saddle point of the Lagrangian, i.e., the solution to the problem $\max_Z \min_A \mathscr{L}(A, Z)$, which can be found by iterating the following two steps:

$$\begin{cases} A_k &= \arg\min_A \mathscr{L}(A, Z_{k-1}), \\ Z_k &= Z_{k-1} + \beta \frac{\partial \mathscr{L}}{\partial Z}(A_k, Z_{k-1}), \end{cases} \tag{3.32}$$

where $\beta > 0$ is the step size. It is very easy to see that $\frac{\partial \mathscr{L}}{\partial Z}(A_k, Z_{k-1}) = \mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(A_k)$. To compute the optimal A given Z_{k-1}, notice that $\langle Z, \mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(A)\rangle = \langle \mathcal{P}_\Omega(Z), X - A\rangle$, and by completing squares, we have

$$\arg\min_A \mathscr{L}(A, Z) = \arg\min_A \tau\|A\|_* + \frac{1}{2}\|A - \mathcal{P}_\Omega(Z)\|_F^2. \tag{3.33}$$

The minimizer to this problem is given by the so-called *proximal operator* of the nuclear norm: $A^* = \mathcal{D}_\tau(\mathcal{P}_\Omega(Z))$, where $\mathcal{D}_\tau$ is the singular value thresholding operator defined in (2.95). We have left the derivation as Exercise 2.16.

Hence, starting from $Z_0 = 0$, the Lagrangian objective $\max_Z \min_A \mathscr{L}(A, Z)$ can be optimized via Algorithm 3.2. This is also known as the *proximal gradient descent* method. Even though the objective function (3.31) is not smooth, this method is known to converge as fast as the regular gradient descent method for smooth functions, with a rate of $O(1/k)$. If one wants to obtain the solution to the problem (3.28), one can repeat the algorithm with an increasing sequence of τ's and at each run, initialize A with the value previously obtained.

Algorithm 3.2 (Low-Rank Matrix Completion by Proximal Gradient)

Input: Entries x_{ij} of a matrix $X \in \mathbb{R}^{D\times N}$ for $(i, j) \in \Omega$ and parameter $\tau > 0$.

1: Initialize $Z \leftarrow \mathbf{0}$.
2: **repeat**
3: $\quad A \leftarrow \mathcal{D}_\tau(\mathcal{P}_\Omega(Z))$.
4: $\quad Z \leftarrow Z + \beta(\mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(A))$.
5: **until** convergence of Z.

Output: Matrix A.

Example 3.3 (Completing Face Images with Missing Pixels by Convex Optimization) As we have seen in Chapter 2, under certain idealized circumstances (such as Lambertian reflectance), images of the same object taken under different illumination conditions lie near an approximately nine-dimensional linear subspace known as the *harmonic plane* (Basri and Jacobs 2003). In this example, we exploit such a low-dimensional structure to recover face images from the extended Yale B data set that have been corrupted so that the intensity values of some pixels are missing. The data matrix is formed by taking frontal face images of subject 20 under all 64 different illumination conditions. Each image is down-sampled to size 96×84. To synthesize a matrix with missing entries, a fraction of pixels from each image is randomly selected as the missing entries. We apply the proximal gradient algorithm described in Algorithm 3.2 to complete such "missing" entries. Figure 3.2 shows the results of image completion for different parameters τ for varying levels of missing entries (from 30% missing entries to 90%). Notice that with a proper choice of the parameter τ (around $\tau = 4 \times 10^5$ in this case), the convex optimization method is able to recover up to 80% of missing entries.

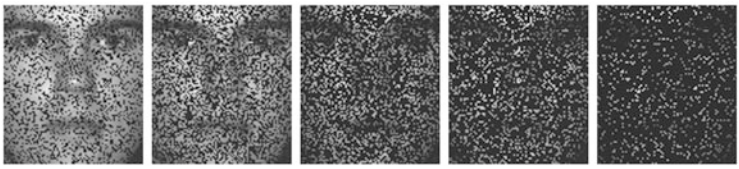
3.1.4 Incomplete PCA by Alternating Minimization

Although the convex-optimization-based approach can ensure correctness of the low-rank solution for the matrix completion problem, it requires solving a convex program of the same size as the matrix. When the data matrix X is very large, parameterizing the low-rank solution A and Lagrange multipliers Z with two matrices of the same size as X seems rather demanding, actually redundant. At least the low-rank solution A could be parameterized more economically with its low-rank factors. Hence, if scalability of the algorithm is a serious concern, it makes sense to look for the low-rank factors of the solution matrix directly.

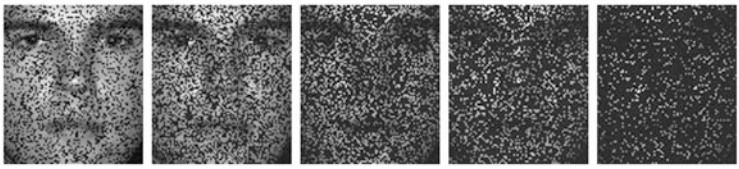
To this end, we introduce in this section an alternating minimization algorithm for solving the geometric PCA problem with missing data. The main idea behind this approach, which was probably first proposed in (Wiberg 1976), is to find $\boldsymbol{\mu}$, U, and Y that minimize the error $\|X - \boldsymbol{\mu}\mathbf{1}^\top - UY\|_F^2$ considering only the known entries of X in the set $\Omega = \{(i,j) : w_{ij} = 1\}$, i.e.,

$$\begin{aligned} \|\mathcal{P}_\Omega(X - \boldsymbol{\mu}\mathbf{1}^\top - UY)\|_F^2 &= \|W \odot (X - \boldsymbol{\mu}\mathbf{1}^\top - UY)\|_F^2 \\ &= \sum_{i=1}^{D}\sum_{j=1}^{N} w_{ij}(x_{ij} - \mu_i - \boldsymbol{u}_i^\top \boldsymbol{y}_j)^2, \end{aligned} \tag{3.34}$$

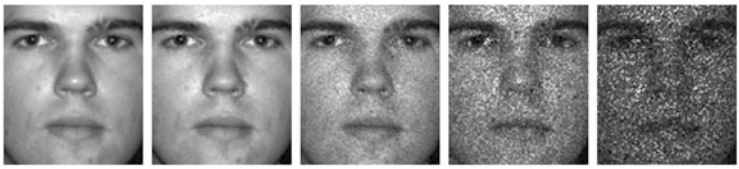
where x_{ij} is the (i,j)th entry of X, μ_i is the ith entry of $\boldsymbol{\mu}$, $\boldsymbol{u}_i^\top$ is the ith row of U, and $\boldsymbol{y}_j$ is the jth column of Y. Notice that this cost function is the same as that in (3.10), except that the errors $\varepsilon_{ij} = x_{ij} - \boldsymbol{u}_i^\top \boldsymbol{y}_j$ associated with the missing entries ($w_{ij} = 0$) are removed.



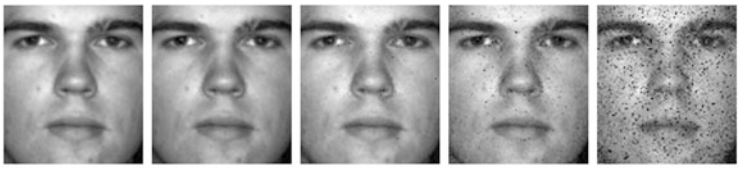
(a) Face images with (30, 50, 70, 80, 90)% percentage of missing entries



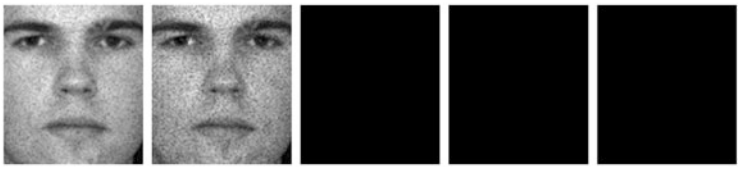
(b) Face images reconstructed by convex optimization with $\tau = 10^3$



(c) Face images reconstructed by convex optimization with $\tau = 2 \times 10^4$



(d) Face images reconstructed by convex optimization with $\tau = 4 \times 10^5$



(e) Face images reconstructed by convex optimization with $\tau = 8 \times 10^6$

Fig. 3.2 Matrix completion via convex optimization for face image completion. We take frontal face images (size 96×84) of subject 20 from the extended Yale B data set and randomly select a fraction of pixels as missing entries. Each column corresponds to input or result under a different percentage of missing entries. The first row is the input images, and other rows show the completion results by convex optimization with different values of τ for the algorithm. Each image shows one typical example of the recovered 64 images.

In what follows, we will derive an alternating minimization algorithm for minimizing the cost function in (3.34). For the sake of simplicity, we will first derive the algorithm in the case of zero-mean and complete data. In this case, the problem in (3.34) reduces to a low-rank matrix approximation problem, which can be solved using the SVD, as described in Theorem 2.3. The alternating minimization algorithm to be derived provides an alternative to the SVD solution, which, however, can be more easily extended to the case of incomplete data, as we will see. Moreover, the algorithm can also be extended to the more challenging PCA problem with missing entries, as we will see.

Matrix Factorization by Alternating Minimization
In the case of complete, zero-mean data, the optimization problem in (3.34) reduces to the low-rank matrix approximation problem based on explicit factorization $\min_{U,Y} \|X - UY\|_F^2$. As we have seen in Chapter 2, this problem can be solved from the SVD of X. Here, we consider an alternative method based on the *orthogonal power iteration* method (Golub and Loan 1996) for computing the top d eigenvectors of a square matrix.

Suppose that $A \in \mathbb{R}^{N\times N}$ is a symmetric positive semidefinite matrix with eigenvectors $\{\boldsymbol{u}_i\}_{i=1}^N$ and eigenvalues $\{\lambda_i\}_{i=1}^N$ sorted in decreasing order. Suppose that $\lambda_1 > \lambda_2$ and let $\boldsymbol{u}^0 \in \mathbb{R}^N$ be an arbitrary vector such that $\boldsymbol{u}_1^\top \boldsymbol{u}^0 \neq 0$. One can show (see Exercise 3.3) that the sequence of vectors

$$\boldsymbol{u}^{k+1} = \frac{A\boldsymbol{u}^k}{\|A\boldsymbol{u}^k\|} \tag{3.35}$$

converges to the top eigenvector of A up to sign, i.e., $\boldsymbol{u}^k \to \pm\boldsymbol{u}_1$, and that the rate of convergence is $\frac{\lambda_2}{\lambda_1}$. This method for computing the top eigenvector of a matrix is called the *power method*.

More generally, assume that $\lambda_d > \lambda_{d+1}$ and let $U^0 \in \mathbb{R}^{N\times d}$ be an arbitrary matrix whose column space is not orthogonal to the subspace $\{\boldsymbol{u}_i\}_{i=1}^d$ spanned by the top d eigenvectors. One can show (see Exercise 3.3) that the sequence of matrices

$$U^{k+1} = AU^k(R^k)^{-1}, \tag{3.36}$$

where $Q^kR^k = AU^k$ is the QR decomposition of AU^k, converges to a matrix U whose columns are the top d eigenvectors of A and that the rate of convergence is $\frac{\lambda_{d+1}}{\lambda_d}$. This method for computing the top d eigenvectors of a matrix is called the *orthogonal power iteration* method or Lanczos method (Lanczos 1950).

Power Factorization (PF) (Hartley and Schaffalitzky 2003) is a generalization of the orthogonal power iteration approach for computing the top d singular vectors of a (possibly) nonsquare matrix X. The main idea behind PF is that given $Y \in \mathbb{R}^{d\times N}$, an optimal solution for $U \in \mathbb{R}^{D\times d}$ that minimizes $\|X - UY\|_F^2$ is given by $XY^\top(YY^\top)^{-1}$. As before, such a matrix can be made orthogonal by

Algorithm 3.3 (Complete Matrix Factorization by Power Factorization)

Input: Matrices $X \in \mathbb{R}^{D\times N}$ and $Y^0 \in \mathbb{R}^{d\times N}$.

1: **initialize** $Y \leftarrow Y^0$.
2: **repeat**
3: Given Y, find $U \leftarrow Q$, where $QR = XY^\top(YY^\top)^{-1}$.
4: Given U, find $Y \leftarrow U^\top X$.
5: **until** convergence of the product UY.

Output: Matrices U and Y.

replacing U by the Q factor of the QR decomposition of $XY^\top(YY^\top)^{-1}$. Then, given an orthogonal U, the optimal Y that minimizes $\|X - UY\|_F^2$ is $U^\top X$. The PF algorithm (see Algorithm 3.3) then iterates between these two steps till convergence is achieved. The method is guaranteed to converge to the rank-d approximation of X, as stated in the following theorem, whose proof is left as an exercise to the reader (see Exercise 3.4).

Theorem 3.4 (Power Factorization)**.** *Let X_d be the best rank-d approximation of X according to the Frobenius norm. Let σ_i be the ith singular value of X. If $\sigma_d > \sigma_{d+1}$, then there exists a constant $c > 0$ such that for all $k \geq 0$,*

$$\|X_d - U^k Y^k\|_F^2 \leq c\Big(\frac{\sigma_{d+1}}{\sigma_d}\Big)^{2k}, \tag{3.37}$$

where U^k and Y^k are the values at iteration k of the matrices U and Y in Algorithm 3.3.

Matrix Completion by Alternating Minimization

Let us now consider the matrix factorization problem with incomplete, zero-mean data, i.e., the problem in (3.34) with $\boldsymbol{\mu} = \mathbf{0}$. Taking the derivatives of the cost function in (3.34) with respect to $\boldsymbol{u}_i$ and $\boldsymbol{y}_j$ and setting them to zero leads to

$$\Big(\sum_{j=1}^{N} w_{ij}\boldsymbol{y}_j\boldsymbol{y}_j^\top\Big)\boldsymbol{u}_i = \sum_{j=1}^{N} w_{ij}x_{ij}\boldsymbol{y}_j, \qquad i = 1,\ldots,D, \tag{3.38}$$

$$\Big(\sum_{i=1}^{D} w_{ij}\boldsymbol{u}_i\boldsymbol{u}_i^\top\Big)\boldsymbol{y}_j = \sum_{i=1}^{D} w_{ij}x_{ij}\boldsymbol{u}_i, \qquad j = 1,\ldots,N. \tag{3.39}$$

Therefore, given Y, the optimal U can be computed linearly from (3.38). As before, the constraint $U^\top U = I$ can be enforced by replacing U by the Q factor of the QR decomposition of $U = QR$. Then, given U, the optimal Y can be computed linearly from (3.39). This leads to the PF algorithm for matrix factorization with missing entries summarized in Algorithm 3.4.

Algorithm 3.4 (Matrix Completion by Power Factorization)

Input: Matrices $W \odot X \in \mathbb{R}^{D\times N}$ and $Y^0 \in \mathbb{R}^{d\times N}$.

1: **initialize** $Y \leftarrow Y^0$.
2: **repeat**
3: Given $Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]$, solve $\min_U \|W \odot (X - UY)\|_F^2$ as

$$U = \begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_D^\top \end{bmatrix}, \quad \boldsymbol{u}_i \leftarrow \Big(\sum_{j=1}^N w_{ij}\boldsymbol{y}_j\boldsymbol{y}_j^\top\Big)^{-1} \sum_{j=1}^N w_{ij}x_{ij}\boldsymbol{y}_j, \quad i = 1, \ldots, D.$$

4: Normalize $U \leftarrow UR^{-1}$, where $QR = U$.
5: Given $U = \begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_D^\top \end{bmatrix}$, solve $\min_Y \|W \odot (X - UY)\|_F^2$ as

$$Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N], \; \boldsymbol{y}_j \leftarrow \Big(\sum_{i=1}^D w_{ij}\boldsymbol{u}_i\boldsymbol{u}_i^\top\Big)^{-1} \sum_{i=1}^D w_{ij}x_{ij}\boldsymbol{u}_i, \; j = 1, \ldots, N.$$

6: **until** convergence of the sequence UY.

Output: U and Y.

Incomplete PCA by Alternating Minimization

Let us now consider the PCA problem in the case of incomplete data, i.e., the problem in (3.34), where we want to recover both the mean $\boldsymbol{\mu}$ and the subspace basis U. As in the case of complete data, the solution to this problem need not be unique, because if $(\boldsymbol{\mu}, U, Y)$ is an optimal solution, then so is $(\boldsymbol{\mu} - U\boldsymbol{b}, UA, A^{-1}Y)$ for all $\boldsymbol{b} \in \mathbb{R}^d$ and $A \in \mathbb{R}^{d\times d}$. To handle this issue, we usually enforce the constraints $U^\top U = I$ and $Y\mathbf{1} = \mathbf{0}$. For the sake of simplicity, we will forgo these constraints for a moment, derive an algorithm for solving the unconstrained problem, and then find a solution that satisfies the constraints.

To solve the unconstrained problem, let us take the derivatives of the cost function in (3.34) with respect to μ_i, $\boldsymbol{u}_i$, and $\boldsymbol{y}_j$ and set them to zero. This leads to

$$\Big(\sum_{j=1}^N w_{ij}\Big)\mu_i = \sum_{j=1}^N w_{ij}(x_{ij} - \boldsymbol{u}_i^\top \boldsymbol{y}_j), \qquad i = 1, \ldots, D, \tag{3.40}$$

$$\Big(\sum_{j=1}^N w_{ij}\boldsymbol{y}_j\boldsymbol{y}_j^\top\Big)\boldsymbol{u}_i = \sum_{j=1}^N w_{ij}(x_{ij} - \mu_i)\boldsymbol{y}_j, \qquad i = 1, \ldots, D, \tag{3.41}$$

$$\Big(\sum_{i=1}^D w_{ij}\boldsymbol{u}_i\boldsymbol{u}_i^\top\Big)\boldsymbol{y}_j = \sum_{i=1}^D w_{ij}(x_{ij} - \mu_i)\boldsymbol{u}_i, \qquad j = 1, \ldots, N. \tag{3.42}$$

Algorithm 3.5 (Incomplete PCA by Power Factorization)

Input: Matrix W, entries x_{ij} of (i,j) such that $w_{ij} = 1$, and dimension d.

1: **initialize** $\begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_D^\top \end{bmatrix} \leftarrow U^0 \in \mathbb{R}^{D\times d}$ and $\begin{bmatrix} \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N \end{bmatrix} \leftarrow Y^0 \in \mathbb{R}^{d\times N}$.

2: **repeat**

3: $\quad \mu_i \leftarrow \frac{\sum_{j=1}^N w_{ij}(x_{ij} - \boldsymbol{u}_i^\top \boldsymbol{y}_j)}{\sum_{j=1}^N w_{ij}}$.

4: $\quad \boldsymbol{u}_i \leftarrow \Big(\sum_{j=1}^N w_{ij} \boldsymbol{y}_j \boldsymbol{y}_j^\top \Big)^{-1} \sum_{j=1}^N w_{ij}(x_{ij} - \mu_i)\boldsymbol{y}_j$.

5: $\quad U = \begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_D^\top \end{bmatrix} \leftarrow UR^{-1}$, where $QR = \begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_D^\top \end{bmatrix}$.

6: $\quad Y = \begin{bmatrix} \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N \end{bmatrix}$ where $\boldsymbol{y}_j \leftarrow \Big(\sum_{i=1}^D w_{ij} \boldsymbol{u}_i \boldsymbol{u}_i^\top \Big)^{-1} \sum_{i=1}^D w_{ij}(x_{ij} - \mu_i)\boldsymbol{u}_i$.

7: **until** convergence of $\boldsymbol{\mu}\mathbf{1}^\top + UY$.

Output: $\boldsymbol{\mu} + \frac{1}{N} UY\mathbf{1}$, U and $Y(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top)$.

Therefore, given U and Y, the optimal $\boldsymbol{\mu}$ can be computed from (3.40). Likewise, given $\boldsymbol{\mu}$ and Y, the optimal U can be computed linearly from (3.41). Also, given $\boldsymbol{\mu}$ and U, the optimal Y can be computed linearly from (3.42).

As before, we can enforce the constraint $U^\top U = I$ by replacing U by the Q factor of the compact QR decomposition of $U = QR$. Also, we can enforce the constraint $Y\mathbf{1} = \mathbf{0}$ by replacing $\boldsymbol{\mu}$ by $\boldsymbol{\mu} + \frac{1}{N} UY\mathbf{1}$, and Y by $Y(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top)$. This leads to the alternating minimization approach for PCA with missing entries summarized in Algorithm 3.5.

A similar alternating minimization approach was proposed in (Shum et al. 1995), in which the steps in (3.40) and (3.41) are combined into a single step

$$\sum_{j=1}^N w_{ij} \begin{bmatrix} \boldsymbol{y}_j \\ 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_j \\ 1 \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{u}_i \\ \mu_i \end{bmatrix} = \sum_{j=1}^N w_{ij} x_{ij} \begin{bmatrix} \boldsymbol{y}_j \\ 1 \end{bmatrix}, \quad i = 1, \ldots, D. \tag{3.43}$$

This leads to an alternating minimization scheme whereby given Y, one solves for $\boldsymbol{\mu}$ and U from (3.43), and given $\boldsymbol{\mu}$ and U, one solves for Y from (3.42).

Ensuring Global Optimality of Alternating Minimization for Matrix Completion
According to Theorem 3.4, when the data matrix $X \in \mathbb{R}^{D\times N}$ is complete, the alternating minimization method in Algorithm 3.3 is guaranteed to converge exponentially to the optimal rank-d approximation of X as long as $\sigma_{d+1}/\sigma_d < 1$. In the case of incomplete data, the alternating procedure in Algorithm 3.4 is perhaps the simplest and most natural extension of Algorithm 3.3. However, since the objective function is nonconvex, there is no guarantee that the algorithm will

converge. Thus, a natural question is whether there are conditions on the rank of X and the number of observed entries M under which the alternating minimization approach is guaranteed to converge. Now, even if the algorithm were to converge, there is no guarantee that it would converge to the globally optimal low-rank factors, or that the product of the factors would give the optimal rank-d approximation of X. Thus, another natural question is whether there are conditions on the rank of X and the number of observed entries M under which the alternating minimization approach is guaranteed to converge to the globally optimal rank-d approximation of X, and hence perfectly complete X when it has rank d. According to Theorem 3.2, the nuclear norm minimization approach in (3.28) is able to complete most rank-d matrices from $M \geq O(d\, N \log(N)^2)$ entries. Thus, a natural conjecture is that the alternating minimization approach should be able to complete a rank-d matrix from a number of entries that depends on d, $N\text{polylog}(N)$, and some ratio of the singular values of X. However, while alternating minimization methods for matrix completion have been used for many years, theoretical guarantees for the convergence and optimality of such methods have remained elusive.

Nonetheless, recent progress in low-rank matrix factorization (Burer and Monteiro 2005; Bach 2013; Haeffele et al. 2014) has shown that under certain conditions, local minimizers for certain classes of matrix factorization problems are global minimizers. Moreover, recent progress in low-rank matrix completion (Jain et al. 2012; Keshavan 2012; Hardt 2014; Jain and Netrapalli 2014) has shown that under certain benign conditions, certain alternating minimization methods do converge to the globally optimal solution with high probability when the matrix is of sufficiently high dimension. While a detailed explanation of such results is far beyond the scope of this book, we provide here a brief introduction with two purposes in mind. First, the analytical conditions required for optimality provide good intuition as to when we should expect low-rank matrix completion to work well in general. Second, some of the proposed algorithms introduce some modifications to the above alternating minimization methods, which may inspire readers to develop even better algorithms in the future.

As before, we are interested in finding a rank-d factorization UY, with factors $U \in \mathbb{R}^{D\times d}$ and $Y \in \mathbb{R}^{d\times N}$, that best approximates the data matrix $X \in \mathbb{R}^{D\times N}$ given the observed entries $W \odot X$ specified by the matrix $W \in \{0,1\}^{D\times N}$, i.e.,

$$\min_{U,Y} \|W \odot (X - UY)\|_F^2. \tag{3.44}$$

The alternating minimization algorithm for solving this problem (Algorithm 3.4) uses all of the observed entries of X at each iteration in order to update the factors. In contrast, the work of (Jain et al. 2012) proposes a modified alternating minimization algorithm (see Algorithm 3.6) that uses only a *partition* of the observed entries at each iteration, whence the name *partition alternating minimization*. Specifically, the set of observed entries W is partitioned into $2K+1$ randomly chosen nonoverlapping and equally sized subsets, denoted by $W_0, W_1, \ldots, W_{2K}$. Then the updates of the original alternating minimization algorithm, Algorithm 3.4, are applied using

Algorithm 3.6 (Matrix Completion by Partition Alternating Minimization)

Input: Observed matrix $W \odot X$ and partition matrices $W_1, \ldots, W_{2K}$.
1: **initialization**
2: $U^0 \leftarrow$ top d left singular vectors of the matrix $\frac{1}{p} W_0 \odot X$.
3: $U^0 \leftarrow Q$, where $QR = U_0 - \mathcal{H}_{\frac{2\nu\sqrt{d}}{\sqrt{N}}}(U_0)$.
4: **end initialization**
5: **for** $k = 0, 1, \ldots, K-1$ **do**
6: $Y^{k+1} \leftarrow \arg\min_Y ||W_{k+1} \odot (U^k Y - X)\|_F^2$.
7: $U^{k+1} \leftarrow \arg\min_U ||W_{K+k+1} \odot (UY^{k+1} - X)\|_F^2$.
8: **end for**
Output: Matrix $U^K Y^K$.

the observed entries specified by W_{k+1} to update Y and the observed entries specified by W_{K+k+1} to update U, for each $k = 0, \ldots, K-1$, instead of those specified by W. The second main difference between Algorithm 3.6 and the original alternating minimization algorithm, Algorithm 3.4, is the way in which the factor U is initialized. While in Algorithm 3.4, U is typically initialized at random, in Algorithm 3.6, the factor U is initialized using the observed entries. Specifically, let p be the probability that an entry is observed, and let $M = pDN$ be the expected number of observed entries. Let U be the top d singular vectors of $\frac{1}{p} W_0 \odot X$, and $\nu > 0$ the incoherence parameter for X according to Definition 3.1. We clip entries of U that have magnitude greater than $\frac{2\nu\sqrt{d}}{\sqrt{N}}$ to be zero and let the initial U^0 be the orthonormalized version of such U obtained via QR decomposition.

In short, there are two major differences between Algorithm 3.6 and Algorithm 3.4: the initialization based on the singular vectors of $\frac{1}{p} W_0 \odot X$ and the update in each iteration using only a subset of the observations. It is surprising that these small modifications to the basic alternating minimization method can ensure that the new procedure approximates the globally optimal solution as described by the following theorem. A complete proof and explanation of this theorem is beyond the scope of this book. We refer interested readers to (Jain et al. 2012).

Theorem 3.5 (Partition Alternating Minimization for Matrix Completion)**.** *Let X be a $D \times N$ matrix of rank d, with $N \geq D$. Assume that X is ν-incoherent with respect to the set of sparse matrices according to Definition 3.1. Let M be the expected number of observed entries, whose locations are sampled independently and uniformly at random. If there exists a constant $c > 0$ such that*

$$M \geq c\, \nu^2 \left(\frac{\sigma_1}{\sigma_d}\right)^4 d^{4.5} N \log(N) \log\left(\frac{d\|X\|_F}{\varepsilon}\right), \tag{3.45}$$

then with high probability, for $K = C' \log(\|X\|_F/\varepsilon)$ with some constant $C' > 0$, the outputs of Algorithm 3.6 satisfy $\|X - U^K Y^K\|_F \leq \varepsilon$.

In words, the alternating minimization procedure guarantees to recover X up to precision ε in $O(\log(1/\varepsilon))$ steps given that the number of observations is of order $O(d^{4.5}N\log(N)\log(d))$. This result is in perfect agreement with our conjecture that the sample complexity of alternating minimization for matrix completion should depend on d, $N\text{polylog}(N)$, and some ratio of the singular values of X. However, by comparing this result with the one for the convex optimization approach, $M \geq O(\nu^2 dN\log(N)^2)$, we see that this comes at the cost of an increase of the sample complexity as a function of d from linear to polynomial. This has motivated the development of modified versions of Algorithm 3.6 that are guaranteed to recover X up to precision ε under either incomparable or weaker conditions. For example, the method proposed in (Keshavan 2012) requires the expected number of observed entries to satisfy (for some constant c)

$$M \geq c\,\nu\Big(\frac{\sigma_1}{\sigma_d}\Big)^8 dN\log\Big(\frac{N}{\varepsilon}\Big), \tag{3.46}$$

which is superior when the matrix has a small condition number, while the method in (Hardt 2014) requires the expected number of observed entries to satisfy (for some constant c)

$$M \geq c\,\nu\Big(\frac{\sigma_1}{\sigma_d}\Big)^2 d^2\Big(d+\log\big(\frac{N}{\varepsilon}\big)\Big)N, \tag{3.47}$$

which reduces the exponent of both the ratio of the singular values as well as the subspace dimension.

Observe also that the results of (Jain et al. 2012; Hardt 2014) are of a slightly different flavor from that of results for convex optimization-based methods, since the minimum number of observed entries depends not only on the dimension of the subspace d, but also on the condition number σ_1/σ_d, which could be arbitrarily large, and the desired accuracy ε. In particular, to achieve perfect completion ($\varepsilon = 0$), we would need to observe the whole matrix. To address this issue, the work of (Jain and Netrapalli 2014) proposes a factorized version of the singular value projection algorithm of (Jain et al. 2010), called stagewise singular value projection, which is guaranteed to complete a rank-d matrix X exactly, provided that the expected number of observed entries satisfies (for some constant c)

$$M \geq c\,\nu^4 d^5 N(\log(N))^3. \tag{3.48}$$

Evidently, this result is worse than that for the nuclear norm minimization approach, which has sample complexity $O(\nu^2 dN\log(N)^2)$. But this comes at the advantage of improving the computational complexity from $O(N^3\log(\frac{1}{\varepsilon}))$ for the nuclear norm minimization approach to $O(\nu^4 d^7 N\log^3(N)\log(\frac{1}{\varepsilon}))$ for the stagewise singular value projection.

In summary, there is currently great interest in trying to develop alternating minimization algorithms for matrix completion with theoretical guarantees of

convergence to the optimal rank-d matrix. Such algorithms are computationally less expensive that the nuclear minimization approach, but this comes at the cost of tolerating a smaller number of missing entries. However, as of the writing of this book, existing results do not directly apply to the basic alternating minimization procedure given in Algorithm 3.4. We conjecture that this procedure should be able to correctly complete a matrix under conditions similar to those presented in this section. Having such a result would be important, because in practice, it may be preferable to use Algorithm 3.4 because it is simpler and easier to implement.

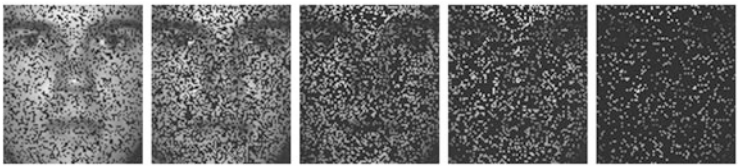
Example 3.6 (Completing Face Images with Missing Pixels by Power Factorization) In Example 3.3, we applied the convex optimization approach (Algorithm 3.2) to complete face images in the extended Yale B data set with missing pixels. In this example, we apply the PF method for incomplete PCA (Algorithm 3.5) to the same images. Figure 3.3 shows the results for different values of the subspace dimension d. We see that for a proper choice of d (in this case from 2 to 9), the PF method works rather well up to 70% of random missing entries. However, PF fails completely for higher percentages of missing entries. This is because PF can become numerically unstable when some of the matrices are not invertible. Specifically, since there are only $N = 64$ face images, it is likely that for some rows of the data matrix, the number of observed entries is less than d; thus the matrix $\sum_{j=1}^{N} w_{ij} \boldsymbol{y}_j \boldsymbol{y}_j^\top$ in line 4 of Algorithm 3.5 becomes rank-deficient. We also observed that as expected, PF is faster than the convex approach. Specifically, in this example, PF took 1.48 seconds in MATLAB, while the convex optimization approach took 10.15 seconds.

3.2 PCA with Robustness to Corrupted Entries

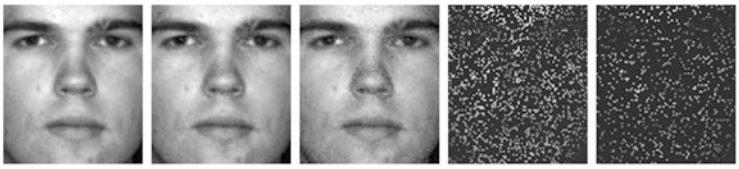
In the previous section, we considered the PCA problem in the case that some entries of the data points are missing. In this section, we consider the PCA problem in the case that some of the entries of the data points have been corrupted by gross errors, known as intrasample outliers. The additional challenge is that we do not know which entries have been corrupted. Thus, the problem is to simultaneously detect which entries have been corrupted and replace them by their uncorrupted values. In some literature, this problem is referred to as the *robust PCA* problem (De la Torre and Black 2004; Candès et al. 2011).

Let us first recall the PCA problem (see Section 2.1.2) in which we are given N data points $\mathcal{X} = \{\boldsymbol{x}_j \in \mathbb{R}^D\}_{j=1}^N$ drawn (approximately) from a d-dimensional affine subspace $S = \{\boldsymbol{x} = \boldsymbol{\mu} + U\boldsymbol{y}\}$, where $\boldsymbol{\mu} \in \mathbb{R}^D$ is an arbitrary point in S, $U \in \mathbb{R}^{D \times d}$ is a basis for S, and $\{\boldsymbol{y}_j \in \mathbb{R}^d\}_{j=1}^N$ are the principal components. In the robust PCA problem, we assume that the ith entry x_{ij} of a data point $\boldsymbol{x}_j$ is obtained by corrupting the ith entry ℓ_{ij} of a point $\boldsymbol{\ell}_j$ lying perfectly on the subspace S by an error e_{ij}, i.e.,

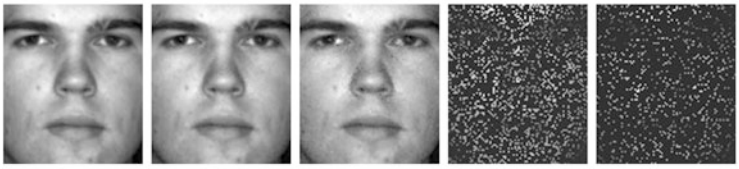
$$x_{ij} = \ell_{ij} + e_{ij}, \quad \text{or} \quad \boldsymbol{x}_j = \boldsymbol{\ell}_j + \boldsymbol{e}_j, \quad \text{or} \quad X = L + E, \tag{3.49}$$



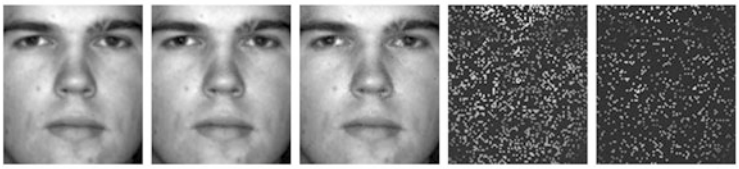
(a) Face images with (30, 50, 70, 80, 90)% percentage of missing entries



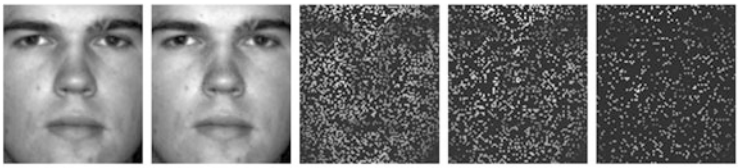
(b) Face images reconstructed by Power Factorization with $d = 2$



(c) Face images reconstructed by Power Factorization with $d = 4$



(d) Face images reconstructed by Power Factorization with $d = 6$



(e) Face images reconstructed by Power Factorization with $d = 9$

Fig. 3.3 Power factorization for recovering face images. We take frontal face images (size 96×84) of subject 20 from the extended Yale B data set and randomly select a fraction of pixels as missing entries. Each column corresponds to input or result under a different percentage of missing entries. The first row is the input images, and other rows are the results obtained by power factorization with different values of d used. Each image shows one typical example of the recovered 64 images.

where $X, L, E \in \mathbb{R}^{D \times N}$ are matrices with entries x_{ij}, ℓ_{ij}, and e_{ij}, respectively. Such errors can have a huge impact on the estimation of the subspace. Thus it is very important to be able to detect the locations of those errors,

$$\Omega = \{(i,j) : e_{ij} \neq 0\}, \tag{3.50}$$

as well as correct the erroneous entries before applying PCA to the given data.

As discussed before, a key difference between the robust PCA problem and the incomplete PCA problem is that we do not know the location of the corrupted entries. This makes the robust PCA problem harder, since we need to simultaneously detect and correct the errors. Nonetheless, when the number of corrupted entries is a small enough fraction of the total number of entries, i.e., when $|\Omega| < \rho \cdot DN$ for some $\rho < 1$, we may still hope to be able to detect and correct such errors. In the remainder of this section, we describe methods from robust statistics and convex optimization for addressing this problem.

3.2.1 Robust PCA by Iteratively Reweighted Least Squares

One of the simplest algorithms for dealing with corrupted entries is the iteratively reweighted least squares (IRLS) approach proposed in (De la Torre and Black 2004). In this approach, a subspace is fit to the corrupted data points using standard PCA. The corrupted entries are detected as those that have a large residual with respect to the identified subspace. A new subspace is estimated with the detected corruptions down-weighted. This process is then repeated until the estimated model stabilizes.

The first step is to apply standard PCA to the given data. Recall from Section 2.1.2 that when the data points $\{\boldsymbol{x}_j \in \mathbb{R}^D\}_{j=1}^N$ have no gross corruptions, an optimal solution to PCA can be obtained as

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{x}_j \quad \text{and} \quad \hat{\boldsymbol{y}}_j = \hat{U}^\top (\boldsymbol{x}_j - \boldsymbol{\mu}), \tag{3.51}$$

where $\hat{U}$ is a $D \times d$ matrix whose columns are the top d eigenvectors of

$$\hat{\Sigma}_N = \frac{1}{N}\sum_{j=1}^{N} (\boldsymbol{x}_j - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_j - \hat{\boldsymbol{\mu}})^\top. \tag{3.52}$$

When the data points are corrupted by gross errors, we may improve the estimation of the subspace by recomputing the model parameters after down-weighting samples that have large residuals. More specifically, let $w_{ij} \in [0,1]$ be a weight assigned to the ith entry of $\boldsymbol{x}_j$ such that $w_{ij} \approx 1$ if x_{ij} is not corrupted,

and $w_{ij} \approx 0$ otherwise. Then a new estimate of the subspace can be obtained by minimizing the weighted sum of the least-squares errors between a point $\boldsymbol{x}_j$ and its projection $\boldsymbol{\mu} + U\boldsymbol{y}_j$ onto the subspace S, i.e.,

$$\sum_{i=1}^{D}\sum_{j=1}^{N} w_{ij}(x_{ij} - \mu_i - \boldsymbol{u}_i^\top \boldsymbol{y}_j)^2, \tag{3.53}$$

where μ_i is the ith entry of $\boldsymbol{\mu}$, $\boldsymbol{u}_i^\top$ is the ith row of U, and $\boldsymbol{y}_j$ is the vector of coordinates of the point $\boldsymbol{x}_j$ in the subspace S.

Notice that the above objective function is identical to the objective function in (3.34), which we used for incomplete PCA. The only difference is that in incomplete PCA, $w_{ij} \in \{0, 1\}$ denotes whether x_{ij} is observed or unobserved, while here $w_{ij} \in [0, 1]$ denotes whether x_{ij} is corrupted or uncorrupted. Other than that, the iterative procedure for computing $\boldsymbol{\mu}$, U, and Y given W is the same as that outlined in Algorithm 3.5.

Given $\boldsymbol{\mu}$, U, and Y, the main question is how to update the weights. A simple approach is to set the weights depending on the residual $\varepsilon_{ij} = x_{ij} - \mu_i - \boldsymbol{u}_i^\top \boldsymbol{y}_j$. Our expectation is that when the residual is small, x_{ij} is not corrupted, and so we should set $w_{ij} \approx 1$. Conversely, when the residual is large, x_{ij} is corrupted, and so we should set $w_{ij} \approx 0$. *Maximum-likelihood-type estimators* (M-Estimators) define the weights to be

$$w_{ij} = \rho(\varepsilon_{ij})/\varepsilon_{ij}^2 \tag{3.54}$$

for some robust loss function $\rho(\cdot)$. The objective function then becomes

$$\sum_{i=1}^{D}\sum_{j=1}^{N} \rho(\varepsilon_{ij}). \tag{3.55}$$

Many loss functions $\rho(\cdot)$ have been proposed in the statistics literature (Huber 1981; Barnett and Lewis 1983). When $\rho(\varepsilon) = \varepsilon^2$, all weights are equal to 1, and we obtain the standard least-squares solution, which is not robust. Other robust loss functions include the following:

1. L_1 loss: $\rho(\varepsilon) = |\varepsilon|$;
2. Cauchy loss: $\rho(\varepsilon) = \varepsilon_0^2 \log(1 + \varepsilon^2/\varepsilon_0^2)$;
3. Huber loss (Huber 1981): $\rho(\varepsilon) = \begin{cases} \varepsilon^2 & \text{if} \quad |\varepsilon| < \varepsilon_0, \\ 2\varepsilon_0|\varepsilon| - \varepsilon_0^2 & \text{otherwise;} \end{cases}$
4. Geman–McClure loss (Geman and McClure 1987): $\rho(\varepsilon) = \frac{\varepsilon^2}{\varepsilon^2 + \varepsilon_0^2}$,

where $\varepsilon_0 > 0$ is a parameter. Following the work of (De la Torre and Black 2004), we use the Geman–McClure loss scaled by ε_0^2, which gives

Algorithm 3.7 (Robust PCA by Iteratively Reweighted Least Squares)

Input: Data matrix X, dimension d, and parameter $\varepsilon_0 > 0$.

initialize $[\boldsymbol{\mu}, U, Y] = \text{PCA}(X)$ using PCA from Chapter 2.
repeat
  $\varepsilon_{ij} \leftarrow x_{ij} - \mu_i - \boldsymbol{u}_i^\top \boldsymbol{y}_j$.
  $w_{ij} \leftarrow \frac{\varepsilon_0^2}{\varepsilon_{ij}^2 + \varepsilon_0^2}$.
  $\mu_i \leftarrow \frac{\sum_{j=1}^N w_{ij}(x_{ij} - \boldsymbol{u}_i^\top \boldsymbol{y}_j)}{\sum_{j=1}^N w_{ij}}$.
  $\boldsymbol{u}_i \leftarrow \Big(\sum_{j=1}^N w_{ij} \boldsymbol{y}_j \boldsymbol{y}_j^\top \Big)^{-1} \sum_{j=1}^N w_{ij}(x_{ij} - \mu_i)\boldsymbol{y}_j$.
  $U = \begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_D^\top \end{bmatrix} \leftarrow \begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_D^\top \end{bmatrix} R^{-1}$, where $QR = \begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_D^\top \end{bmatrix}$.
  $Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]$ where $\boldsymbol{y}_j \leftarrow \Big(\sum_{i=1}^D w_{ij} \boldsymbol{u}_i \boldsymbol{u}_i^\top \Big)^{-1} \sum_{i=1}^D w_{ij}(x_{ij} - \mu_i)\boldsymbol{u}_i$.
until convergence of $\boldsymbol{\mu}\mathbf{1}^\top + UY$.
$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \frac{1}{N} UY\mathbf{1}$, $Y \leftarrow Y(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top)$, $L \leftarrow UY$, and $E \leftarrow X - L$.
Output: $\boldsymbol{\mu}$, U, Y, L and E.

$$w_{ij} = \frac{\varepsilon_0^2}{\varepsilon_{ij}^2 + \varepsilon_0^2}. \tag{3.56}$$

The overall algorithm for PCA with corruptions is summarized in Algorithm 3.7. This algorithm initializes all the weights to $w_{ij} = 1$. This gives an initial estimate for the subspace, which is the same as that given by PCA. Given this initial estimate of the subspace, the weights w_{ij} are computed from the residuals as in (3.56). Given these weights, one can reestimate the subspace using the steps of Algorithm 3.5. One can then iterate between computing the weights given the subspace and computing the subspace given the weights.

Example 3.7 (Face Shadow Removal by Iteratively Reweighted Least Squares) As we have seen in Chapter 2, the set of images of a convex Lambertian object obtained under different lighting conditions lies close to a nine-dimensional linear subspace known as the *harmonic plane* (Basri and Jacobs 2003). However, since faces are neither perfectly convex nor Lambertian, face images taken under different illuminations often suffer from several nuances such as self-shadowing, specularities, and saturations in brightness. Under the assumption that the images of a person's face are aligned, the above robust PCA algorithm offers a principled way of removing the shadows and specularities, because such artifacts are concentrated on small portions of the face images, i.e., they are sparse in the image domain. In this example, we use the frontal face images of subject 20 under 64 different

(a) Input images of a face under different illuminations

(b) Low rank and sparse components by IRLS

Fig. 3.4 Removing shadows and specularities from face images using IRLS for PCA with corrupted data. We apply Algorithm 3.7 to 64 frontal face images of subject 20 from the extended Yale B data set. Each image is of size 96×84. (a) Four out of 64 representative input face images. (b) Recovered images from the low-rank component L (first row) and sparse errors E (second row).

illumination conditions. Each image is down-sampled to size 96×84. We then apply the IRLS method (Algorithm 3.7) with $\varepsilon_0 = 1$ and $d = 4$ to remove the shadows and specularities in the face images. The results in Figure 3.4 show that the IRLS method is able to do a reasonably good job of removing some of the shadows and specularities around the nose and eyes area. However, the error image in the third column shows that the recovered errors are not very sparse, and the method could confuse valid image signal due to darkness with true errors (caused by shadows, etc.)

3.2.2 Robust PCA by Convex Optimization

Although the IRLS scheme for robust PCA is very simple and efficient to implement, and widely used in practice, there is no immediate guarantee that the method converges. Moreover, even if the method were to converge, there is no guarantee that the solution to which it converges corresponds to the correct low-rank matrix. As we have seen in the low-rank matrix completion problem, we should not even expect the problem to have a meaningful solution unless proper conditions are imposed on the low-rank matrix and the matrix of errors.

In this section, we will derive conditions under which the robust PCA problem is well posed and admits an efficient solution. To this end, we will formulate the robust PCA problem as a (nonconvex and nonsmooth) rank minimization problem in which we seek to decompose the data matrix X as the sum of a low-rank matrix L and a matrix of errors E. Similar to the matrix completion case, we will study convex relaxations of the rank minimization problem and resort to advanced tools from high-dimensional statistics to show that under certain conditions, the convex relaxations can effectively and efficiently recover a low-rank matrix with intrasample outliers as long as the outliers are sparse enough. Although the mathematical theory that supports the correctness of these methods is far beyond the scope of this book, we will introduce the key ideas and results of this approach to PCA with intrasample outliers.

More specifically, we assume that the given data matrix X is generated as the sum of two matrices

$$X = L_0 + E_0. \tag{3.57}$$

The matrix L_0 represents the ideal low-rank data matrix, while the matrix E_0 represents the intrasample outliers. Since many entries of X are not corrupted (otherwise, the problem would not be well posed), many entries of E_0 should be zero. As a consequence, we can pose the robust PCA problem as one of decomposing a given matrix X as the sum of two matrices $L + E$, where L is of low rank and E is sparse. This problem can be formulated as

$$\min_{L,E} \quad \text{rank}(L) + \lambda \|E\|_0 \quad \text{s.t.} \quad X = L + E, \tag{3.58}$$

where $\|E\|_0$ is the number of nonzero entries in E, and $\lambda > 0$ is a tradeoff parameter.

Robust PCA as a Well-Posed Problem

At first sight, it may seem that solving the problem in (3.58) is impossible. First of all, we have an underdetermined system of DN linear equations in $2DN$ unknowns. Among the many possible solutions, we are searching for a solution (L, E) such that L is of low rank and E is sparse. However, such a solution may not be unique. For instance, if $x_{11} = 1$ and $x_{ij} = 0$ for all $(i, j) \neq (1, 1)$, then the matrix X is both of rank 1 and sparse. Thus, if $\lambda = 1$, we can choose $(L, E) = (X, \mathbf{0})$ or $(L, E) = (\mathbf{0}, X)$ as valid solutions. To avoid such an ambiguity, as suggested by the results for matrix completion, the low-rank matrix L_0 should be in some sense "incoherent" with the sparse corruption matrix E_0. That is, the low-rank matrix L_0 itself should not be sparse. To capture this, we will assume that L_0 is an incoherent matrix according to Definition 3.1. Second of all, as suggested also by results for matrix completion, if we want to recover the low-rank matrix L_0 correctly, the locations of the corrupted entries should not fall into any conspicuous pattern. Therefore, as in the matrix completion problem, we will assume that the locations of the corrupted entries are distributed uniformly at random so that the chance that they form any conspicuous pattern is very low.

As we will see, under the above condition of incoherence and random corruptions, the problem in (3.58) will become well posed for most matrices X. However, to be able to state the precise conditions under which the solution to (3.58) coincides with (L_0, E_0), we first need to study the question of how to efficiently solve the problem in (3.58).

Recovering a Low-Rank Matrix or a Sparse Vector by Convex Relaxation
Observe that even if the conditions above could guarantee that the problem in (3.58) has a unique globally optimal solution, another challenge is that the cost function to be minimized is nonconvex and nondifferentiable. In fact, it is well known that the problem of recovering either a low-rank matrix C or a sparse signal $\boldsymbol{c}$ from undersampled linear measurements B or $\boldsymbol{b}$, i.e.,

$$\min_{C} \ \operatorname{rank}(C) \ \text{s.t.} \ \mathcal{P}(C) = B, \quad \text{or} \quad \min_{\boldsymbol{c}} \ \|\boldsymbol{c}\|_0 \ \text{s.t.} \ A\boldsymbol{c} = \boldsymbol{b}, \tag{3.59}$$

is in general NP-hard (Amaldi and Kann 1998).

As we have seen for the low-rank matrix completion problem, the difficulty of solving the rank minimization on the left-hand side of (3.59) can be alleviated by minimizing the convex envelope of the rank function, which is given by the matrix nuclear norm $\|C\|_*$ and gives rise to the following optimization problem:

$$\min_{C} \ \|C\|_* \ \text{s.t.} \ \mathcal{P}(C) = B. \tag{3.60}$$

As it turns out, convex relaxation works equally well for finding the sparsest solution to a highly underdetermined system of linear equations $A\boldsymbol{c} = \boldsymbol{b}$, which is the problem on the right-hand side of (3.59). This class of problems is known in the literature as *compressed or compressive sensing* (Candès 2006). Since this linear system is underdetermined, in general there could be many solutions $\boldsymbol{c}$ to the equation $A\boldsymbol{c} = \boldsymbol{b}$. This mimics the matrix completion problem, where the number of given measurements is much less than the number of variables to be estimated or recovered (all the entries of the matrix). Hence we want to know under what conditions the sparsest solution to $A\boldsymbol{c} = \boldsymbol{b}$ is unique and can be found efficiently.

To this end, we briefly survey results from the compressive sensing literature (see (Candès and Tao 2005; Candès 2008) and others). Without loss of generality, let us assume that A is an $m \times n$ matrix with $m \ll n$ whose columns have unit norm. Let $\boldsymbol{b} = A\boldsymbol{c}_0$, where $\boldsymbol{c}_0$ is k-sparse, i.e., $\boldsymbol{c}_0$ has at most k nonzero entries. Our goal is to recover $\boldsymbol{c}_0$ by solving the optimization problem on the right-hand side of (3.59). Notice that if A has two identical columns, say columns 1 and 2, then $\tilde{\boldsymbol{c}}_0 = [1, -1, 0, \ldots, 0]^\top$ satisfies $A\tilde{\boldsymbol{c}}_0 = \boldsymbol{0}$. Thus, if $\boldsymbol{c}_0$ is a sparse solution to $A\boldsymbol{c} = \boldsymbol{b}$, then so is $\boldsymbol{c}_0 + \tilde{\boldsymbol{c}}_0$. More generally, if A has very sparse vectors in its (right) null space, then sparse solutions to $A\boldsymbol{c} = \boldsymbol{b}$ are less likely to be unique. Hence, to ensure the uniqueness of the sparsest solution, we typically need the measurement matrix A to be *mutually incoherent*, as defined next.

Definition 3.8 (Mutual Coherence)**.** *The* mutual coherence *of a matrix* $A \in \mathbb{R}^{m \times n}$ *is defined as*

$$\nu(A) = \max_{i \neq j = 1, \ldots, n} |\boldsymbol{a}_i^\top \boldsymbol{a}_j|. \tag{3.61}$$

A matrix is said to be mutually incoherent with parameter ν *if* $\nu(A) < \nu$.

This definition of incoherence is not to be confused with that in Definition 3.1. Definition 3.8 tries to capture whether each column of A is incoherent with other columns so that no sparse number of columns can be linearly independent, whence the name mutual coherence. This notion is useful for finding a sparse solution to a set of linear equations, as we will see in Theorem 3.10. On the other hand, Definition 3.1 tries to capture whether the matrix as a whole is incoherent with respect to sparse missing entries or sparse corruptions, whence the name incoherence with respect to sparse matrices. This notion of incoherence is useful for solving the matrix completion problem, as we saw in Theorem 3.2, and will be useful for solving the robust PCA problem, as we will see in Theorem 3.66.

Another property of a matrix that is typically used to characterize the conditions under which it is possible to solve a linear system is the notion of restricted isometry, as defined next.

Definition 3.9. *Given an integer* k*, the* restricted isometry constant *of a matrix* A *is the smallest number* $\delta_k(A)$ *such that for all* $\boldsymbol{c}$ *with* $\|\boldsymbol{c}\|_0 \leq k$*, we have*

$$(1 - \delta_k(A))\|\boldsymbol{c}\|_2^2 \leq \|A\boldsymbol{c}\|_2^2 \leq (1 + \delta_k(A))\|\boldsymbol{c}\|_2^2. \tag{3.62}$$

The remarkable results from compressive sensing have shown that if the measurement matrix A is sufficiently incoherent or the restricted isometry constant is small enough, then to find the correct sparsest solution to $A\boldsymbol{c} = \boldsymbol{b}$, we can replace the ℓ_0 norm in (3.59) by its convex envelope, the ℓ_1 norm, which gives rise to the following optimization problem:

$$\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_1 \quad \text{s.t.} \quad \boldsymbol{b} = A\boldsymbol{c}. \tag{3.63}$$

More precisely, we have the following result:

Theorem 3.10 (Sparse Recovery under Incoherence or Restricted Isometry)**.** *If the matrix* A *is incoherent, i.e., if* $\nu(A) < \frac{1}{2k-1}$*, or if it satisfies the restricted isometry property (RIP)* $\delta_{2k}(A) < \sqrt{2} - 1$*, then the optimal solution* $\boldsymbol{c}^*$ *to the* ℓ_1*-minimization problem in* (3.63) *is the correct sparsest solution, i.e.,* $\boldsymbol{c}^* = \boldsymbol{c}_0$.

In other words, when the matrix A is incoherent enough, the sparsest solution to the linear system $A\boldsymbol{c} = \boldsymbol{b}$ can be obtained by solving a convex ℓ_1-minimization problem as opposed to an NP-hard ℓ_0-minimization problem.

Robust PCA by Convex Relaxation
Inspired by the above convex relaxation techniques, for the robust PCA problem in (3.58) we would expect that under certain conditions on L_0 and E_0, we can decompose X as $L_0 + E_0$ by solving the following convex optimization problem:

$$\min_{L,E} \quad \|L\|_* + \lambda \|E\|_1 \quad \text{s.t.} \quad X = L + E, \tag{3.64}$$

where $\|L\|_* = \sum_i \sigma_i(L)$ is the nuclear norm of L, i.e., the sum of its singular values, and $\|E\|_1 = \sum_{i,j} |e_{ij}|$ is the ℓ_1 norm of E viewed as a vector. This convex program is known as *principal component pursuit* (PCP).

The following theorem gives precise conditions on the rank of the matrix and the percentage of outliers under which the optimal solution of the above convex program is exactly (L_0, E_0) with overwhelming probability.

Theorem 3.11 (Robust PCA by Principal Component Pursuit (Candès et al. 2011))**.** *Let $X = L_0 + E_0$. Assume that $L_0 = U\Sigma V^\top$ is ν-incoherent with respect to the set of sparse matrices according to Definition 3.1. Assume also that the support of E_0 is uniformly distributed among all the sets of cardinality $D \times N$. If*

$$\text{rank}(L_0) \leq \frac{\rho_d \min\{D, N\}}{\mu^2 \log^2\big(\max\{D, N\}\big)} \quad \text{and} \quad \|E_0\|_0 \leq \rho_s ND \tag{3.65}$$

for some constant $\rho_d, \rho_s > 0$, then there is a constant c such that with probability at least $1 - c\max\{N, D\}^{-10}$, the solution (L^*, E^*) to (3.64) with $\lambda = \frac{1}{\sqrt{\max\{N,D\}}}$ is exact, i.e.,

$$L^* = L_0 \quad \text{and} \quad E^* = E_0. \tag{3.66}$$

A complete proof and explanation for this theorem is beyond the scope of this book; interested readers are referred to (Candès et al. 2011). But this does not prevent us from understanding its implications and using it to develop practical solutions for real problems. The theorem essentially says that as long as the low-rank matrix is incoherent and its rank is bounded almost linearly from its dimension, the PCP program can correctly recover the low-rank matrix even if a constant fraction of its entries are corrupted. Other results show that under some additional benign conditions, say the signs of the entries of E_0 are random, the convex optimization can correct an arbitrarily high percentage of errors if the matrix is sufficiently large (Ganesh et al. 2010).

Alternating Direction Method of Multipliers for Principal Component Pursuit
Assuming that the conditions of Theorem 3.66 are satisfied, the next question is how to find the global minimum of the convex optimization problem in (3.64). Although in principle, many convex optimization solvers can be used, we introduce here an algorithm based on the augmented Lagrange multiplier (ALM) method suggested by (Candès et al. 2011; Lin et al. 2011).

The ALM method operates on the *augmented Lagrangian*

$$\mathscr{L}(L, E, \Lambda) = \|L\|_* + \lambda\|E\|_1 + \langle \Lambda, X - L - E\rangle + \frac{\beta}{2}\|X - L - E\|_F^2. \tag{3.67}$$

A generic Lagrange multiplier algorithm (Bertsekas 1999) would solve PCP by repeatedly setting $(L_k, E_k) = \arg\min_{L,E} \mathscr{L}(L, E, \Lambda_k)$ and then updating the Lagrange multiplier matrix by $\Lambda_{k+1} = \Lambda_k + \beta(X - L_k - E_k)$. This is also known as the *exact ALM method*.

For our low-rank and sparse decomposition problem, we can avoid having to solve a sequence of convex programs by recognizing that $\min_L \mathscr{L}(L, E, \Lambda)$ and $\min_E \mathscr{L}(L, E, \Lambda)$ both have very simple and efficient solutions. In particular, it is easy to show that

$$\arg\min_E \mathscr{L}(L, E, \Lambda) = \mathcal{S}_{\lambda\beta^{-1}}(X - L + \beta^{-1}\Lambda), \tag{3.68}$$

where $\mathcal{S}_\tau(X)$ is the soft-thresholding operator defined in (2.96) applied to each entry x of the matrix X as $\mathcal{S}_\tau(x) = \text{sign}(x)\max(|x| - \tau, 0)$. Similarly, it is not difficult to show that (see Exercise 2.16)

$$\arg\min_L \mathscr{L}(L, E, \Lambda) = \mathcal{D}_{\beta^{-1}}(X - E + \beta^{-1}\Lambda), \tag{3.69}$$

where $\mathcal{D}_\tau(X)$ is the singular value thresholding operator defined in (2.95) as $\mathcal{D}_\tau(X) = U\mathcal{S}_\tau(\Sigma)V^*$, where $U\Sigma V^*$ is any singular value decomposition of X.

Thus, a more practical strategy is first to minimize $\mathscr{L}$ with respect to L (fixing E), then minimize $\mathscr{L}$ with respect to E (fixing L), and then finally update the Lagrange multiplier matrix Λ based on the residual $X - L - E$, a strategy that is summarized as Algorithm 3.8 below.

Algorithm 3.8 is a special case of a general class of algorithms known as *alternating direction method of multipliers* (ADMM), described in Appendix A. The convergence of these algorithms has been well studied and established (see e.g., (Lions and Mercier 1979; Kontogiorgis and Meyer 1989) and the many references therein, as well as discussion in (Lin et al. 2011; Yuan and Yang 2009)). Algorithm 3.8 performs excellently on a wide range of problems: relatively small numbers of iterations suffice to achieve good relative accuracy. The dominant cost of each iteration is computing L_{k+1} by singular value thresholding. This requires us to compute the singular vectors of $X - E_k + \beta^{-1}\Lambda_k$ whose corresponding singular values exceed the threshold β^{-1}. Empirically, the number of such large singular values is often bounded by $\text{rank}(L_0)$, allowing the next iterate to be computed efficiently by a partial SVD.[6] The most important implementation details for this

[6]Further performance gains might be possible by replacing this partial SVD with an approximate SVD, as suggested in (Goldfarb and Ma 2009) for nuclear norm minimization.

Algorithm 3.8 (Principal Component Pursuit by ADMM (Lin et al. 2011))

initialize: $E_0 = \Lambda_0 = 0, \beta > 0$.
while not converged **do**
  compute $L_{k+1} = \mathcal{D}_{\beta^{-1}}(X - E_k + \beta^{-1}\Lambda_k)$.
  compute $E_{k+1} = \mathcal{S}_{\lambda\beta^{-1}}(X - L_{k+1} + \beta^{-1}\Lambda_k)$.
  compute $\Lambda_{k+1} = \Lambda_k + \beta(X - L_{k+1} - E_{k+1})$.
end while
output: L, E.

algorithm are the choice of β and the stopping criterion. In this work, we simply choose $\beta = ND/4\|X\|_1$, as suggested in (Yuan and Yang 2009).

Some Extensions to PCP

In most practical applications, there is also small dense noise in the data. So a more realistic model for robust PCA can be $X = L + E + Z$, where Z is a Gaussian matrix that models small Gaussian noise in the given data. In this case, we can no longer expect to recover the exact solution to the low-rank matrix (which is impossible even if there are no outliers). Nevertheless, one can show that the natural convex extension

$$\min_{L,E} \quad \|L\|_* + \lambda\|E\|_1 \quad \text{s.t.} \quad \|X - L - E\|_2^2 \le \varepsilon^2, \tag{3.70}$$

where ε is the known noise variance, gives a stable estimate to the low-rank and sparse components L and E, subject to a small residual proportional to the noise variance (Zhou et al. 2010b).

Another extension is to recover a low-rank matrix from both corrupted and compressive measurements. In other words, we try to recover the low-rank and sparse components (L, E) of $X = L + E$ from only some of its linear measurements: $\mathcal{P}_Q(X)$, where $\mathcal{P}_Q(\cdot)$ could be a general linear operator. The special case in which the operator represents a subset of the entries has been covered in the original work of principal component pursuit (Candès et al. 2011). It has been shown that under similar conditions as in Theorem 3.66, one can correctly recover the low-rank and sparse components by the following optimization:

$$\min_{L,E} \quad \|L\|_* + \lambda\|E\|_1 \quad \text{s.t.} \quad \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(L + E), \tag{3.71}$$

where as in matrix completion, $\mathcal{P}_\Omega(\cdot)$ represents projection onto the observed entries.

The case of a more general linear operator $\mathcal{P}_Q(\cdot)$ for projecting onto an arbitrary subspace Q has also been studied in (Wright et al. 2013) and is known as *compressive principal component pursuit* (CPCP). It has been shown that under fairly broad conditions (so that Q is in some sense "incoherent" to L and E), the

low-rank and sparse components can be correctly recovered by the following convex program:

$$\min_{L,E} \quad \|L\|_* + \lambda \|E\|_1 \quad \text{s.t.} \quad \mathcal{P}_Q(X) = \mathcal{P}_Q(L + E). \tag{3.72}$$

We leave as an exercise for the reader (see Exercise 3.8) to derive an algorithm for solving the above problems using ideas from Lagrangian methods and alternating direction minimization methods (please refer to Appendix A).

Example 3.12 (Face Shadow Removal by PCP) As we have seen in Example 3.7, robust PCA can be used to remove shadows and specularities in face images that are typically sparse in the image domain. In this example, we apply the PCP method to the same face images in Example 3.7, which correspond to frontal face images of subject 20 under 64 different illuminations (see Figure 3.5). As before, each image is down-sampled to size 96×84. We solve the PCP problem using both the exact ALM method and the inexact method via ADMM (Algorithm 3.8). We set the parameter λ according to Theorem 3.66. The exact and the inexact ALM methods give almost identical results, but the latter is much faster than the former: 2.68 seconds for inexact ALM versus 42.0 seconds for exact ALM in MATLAB on a typical desktop computer. As a comparison, the IRLS method in Example 3.7 takes 2.68 seconds on average. Comparing with the results in Figure 3.4 obtained by the IRLS method, the results given by PCP are qualitatively better in the sense that the recovered errors are indeed sparse and correspond better to true corruptions in the face images due to shadows and specularities. In particular, we can appreciate a significant improvement in the third image. This technique is potentially useful for preprocessing training images in face recognition systems to remove such deviations from the linear model. We leave the implementation of the algorithms as a programming exercise to the reader (see Exercise 3.10).

3.3 PCA with Robustness to Outliers

Another issue that we often encounter in practice is that a small portion of the data points does not fit the subspace as well as the rest of the data. Such points are called *outliers* or *outlying samples*, and their presence can lead to a completely wrong estimate of the underlying subspace. Therefore, it is very important to develop methods for detecting and eliminating outliers from the given data.

The true nature of outliers can be very elusive. In fact, there is really no unanimous definition for what an outlier is.[7] Outliers could be atypical samples that have an unusually *large influence* on the estimated model parameters. Outliers could

[7]For a more thorough exposition of outliers in statistics, we recommend the books of (Barnett and Lewis 1983; Huber 1981).

(a) Input images of a face under different illuminations

(b) Low rank and sparse components by PCP via exact ALM

(c) Low rank and sparse components by PCP via ADMM

Fig. 3.5 Removing shadows and specularities from face images by principal component pursuit. We apply Algorithm 3.7 to 64 frontal face images of subject 20 from the extended Yale B database. Each image is of size 96×84. (a) Four out of 64 representative input face images. (b)-(c) Recovered images from the low-rank component L (first row) and sparse errors E (second row).

also be perfectly valid samples from the same distribution as the rest of the data that happen to be *small-probability* instances. Alternatively, outliers could be samples drawn from a different model, and therefore they will likely *not be consistent* with the model derived from the rest of the data. In principle, however, there is no way to tell which is the case for a particular "outlying" sample point.

In this section, we will discuss two families of methods for dealing with outliers in the context of PCA. The first family will include classical methods based on the

robust statistics literature described in Appendix B. The second family will include modern convex optimization techniques similar to those we have described in the previous two sections for incomplete PCA and robust PCA.

3.3.1 Outlier Detection by Robust Statistics

We begin by discussing three classical approaches from robust statistics for dealing with outliers in the context of PCA. The first method, called an influence-based method, detects outliers as points that have a large influence in the estimated subspace. The second method detects outliers as points whose probability of belonging to the subspace is very low or whose distance to the subspace is very high. Interestingly, this latter method leads to an IRLS approach to detecting outliers. The third method detects outliers by random sample consensus techniques.

Influence-Based Outlier Detection
This approach relies on the assumption that an outlier is an *atypical* sample that has an unusually large influence on the estimated subspace. This leads to an outlier detection scheme whereby the influence of a sample is determined by comparing the subspace $\hat{S} = (\hat{\boldsymbol{\mu}}, \hat{U})$ estimated with all the samples, and the subspace $\hat{S}_{(-j)} = (\hat{\boldsymbol{\mu}}_{(-j)}, \hat{U}_{(-j)})$ estimated without the jth sample. For instance, one may use a *sample influence function* based on some distance between $\hat{S}$ and $\hat{S}_{(-j)}$ such as

$$\text{dist}(\hat{S}, \hat{S}_{(-j)}) = \angle\big(\text{span}(\hat{U}), \text{span}(\hat{U}_{(-j)})\big) \quad \text{or} \tag{3.73}$$

$$\text{dist}(\hat{S}, \hat{S}_{(-j)}) = \|(I - \hat{U}\hat{U}^\top)\boldsymbol{\mu}_{(-j)}\| + \|(I - \hat{U}_{(-j)}\hat{U}_{(-j)}^\top)\boldsymbol{\mu}\|. \tag{3.74}$$

The first quantity is the largest subspace angle (see Exercise 2.8) between the linear subspace spanned by $\hat{U}$ and the linear subspace spanned by $\hat{U}_{(-j)}$. Such a distance measures the influence based on comparing only the linear part of the subspaces, which is appropriate only when the subspaces are linear but may fail otherwise. On the other hand, the second quantity is based on the orthogonal distance from point $\boldsymbol{\mu}$ in $\hat{S}$ to the subspace $\hat{S}_{(-j)}$, plus the orthogonal distance from point $\boldsymbol{\mu}_{(-j)}$ in $\hat{S}_{(-j)}$ to the subspace $\hat{S}$. This distance is more appropriate for comparing the affine part of the subspaces and can be combined with the distance between the linear parts to form a distance between affine subspaces. Given any such distance, the larger the value of the distance, the larger the influence of $\boldsymbol{x}_j$ on the estimate, and the more likely it is that $\boldsymbol{x}_j$ is an outlier. Thus, we may detect sample $\boldsymbol{x}_j$ as an outlier if its influence is above some threshold $\tau > 0$, i.e.,

$$\text{dist}(\hat{S}, \hat{S}_{(-j)}) \geq \tau. \tag{3.75}$$

However, this method does not come without extra cost. We need to compute the principal components (and hence perform SVD) $N + 1$ times: once with all the samples together and another N times with one sample eliminated. There have been many studies that aim to give a formula that can accurately approximate the sample influence without performing SVD $N + 1$ times. Such a formula is called a *theoretical influence* function (see Appendix B). For a more detailed discussion about influence-based outlier rejection for PCA, we refer the interested reader to (Jolliffe 2002).

Probability-Based Outlier Detection: Multivariate Trimming, M-Estimators, and Iteratively Weighted Recursive Least Squares
In this approach, a subspace is fit to *all* sample points, including potential outliers. Outliers are then detected as the points that correspond to small-probability events or that have large fitting errors with respect to the identified subspace. A new subspace is then estimated with the detected outliers removed or down-weighted. This process is then repeated until the estimated subspace stabilizes.

More specifically, recall that in PCA, the goal is to find a low-dimensional subspace that best fits a given set of data points $\mathcal{X} \doteq \{\boldsymbol{x}_j \in \mathbb{R}^D\}_{j=1}^N$ by minimizing the least-squares error

$$\sum_{j=1}^{N} \|\boldsymbol{x}_j - \boldsymbol{\mu} - U\boldsymbol{y}_j\|^2, \tag{3.76}$$

between each point $\boldsymbol{x}_j$ and its projection onto the subspace $\boldsymbol{\mu} + U\boldsymbol{y}_j$, where $\boldsymbol{\mu} \in \mathbb{R}^D$ is any point in the subspace, $U \in \mathbb{R}^{D\times d}$ is a basis for the subspace, and $\boldsymbol{y}_j \in \mathbb{R}^d$ are the coordinates of the point in the subspace. If there are no outliers, an optimal solution to PCA can be obtained as described in Section 2.1.2, i.e.,

$$\hat{\boldsymbol{\mu}}_N = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{x}_j \quad \text{and} \quad \hat{\boldsymbol{y}}_j = \hat{U}^\top(\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N), \tag{3.77}$$

where $\hat{U}$ is a $D \times d$ matrix whose columns are the top d eigenvectors of

$$\hat{\Sigma}_N = \frac{1}{N}\sum_{j=1}^{N} (\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N)(\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N)^\top. \tag{3.78}$$

If we adopt the guideline that outliers are samples that do not fit the model well or have a small probability with respect to the estimated model, then the outliers are exactly those samples that have a relatively large residual

$$\|\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N - \hat{U}\hat{\boldsymbol{y}}_j\|^2 \quad \text{or} \quad \varepsilon_j^2 = (\boldsymbol{x}_j^\top - \boldsymbol{\mu}_N)^\top \Sigma_N^{-1}(\boldsymbol{x}_j^\top - \boldsymbol{\mu}_N), \quad j = 1, 2, \ldots, N. \tag{3.79}$$

The first error is simply the distance to the subspace, while the second error is the *Mahalanobis distance*,[8] which is obtained when we approximate the probability that a sample $\boldsymbol{x}_j$ comes from this model by a multivariate Gaussian

$$p(\boldsymbol{x}_j; \boldsymbol{\mu}_N, \hat{\Sigma}_N) = \frac{1}{(2\pi)^{D/2} \det(\hat{\Sigma}_N)^{1/2}} \exp\big(-\frac{1}{2}(\boldsymbol{x}_j^\top - \boldsymbol{\mu}_N)^\top \Sigma_N^{-1}(\boldsymbol{x}_j^\top - \boldsymbol{\mu}_N)\big). \tag{3.80}$$

In principle, we could use $p(\boldsymbol{x}_j, \boldsymbol{\mu}_N, \hat{\Sigma}_N)$ or either residual ε_j to determine whether $\boldsymbol{x}_j$ is an outlier. However, the above estimate of the subspace is obtained using all the samples, including the outliers themselves. Therefore, the estimated subspace could be completely wrong, and hence the outliers could be incorrectly detected. In order to improve the estimate of the subspace, one can recompute the model parameters after discarding or down-weighting samples that have large residuals. More specifically, let $w_j \in [0, 1]$ be a weight assigned to the jth point such that $w_j \approx 1$ if $\boldsymbol{x}_j$ is an inlier and $w_j \approx 0$ if $\boldsymbol{x}_j$ is an outlier. Then, similarly to (2.23), a new estimate of the subspace can be obtained by minimizing a reweighted least-squares error:

$$\min_{\boldsymbol{\mu}, U, Y} \sum_{j=1}^{N} w_j \|\boldsymbol{x}_j - \boldsymbol{\mu} - U\boldsymbol{y}_j\|^2 \;\text{ s.t. }\; U^\top U = I_d \;\text{ and }\; \sum_{j=1}^{N} w_j \boldsymbol{y}_j = \boldsymbol{0}. \tag{3.81}$$

It can be shown (see Exercise 3.12) that the optimal solution to this problem is of the form

$$\hat{\boldsymbol{\mu}}_N = \frac{\sum_{j=1}^{N} w_j \boldsymbol{x}_j}{\sum_{j=1}^{N} w_j} \quad \text{and} \quad \hat{\boldsymbol{y}}_j = \hat{U}^\top(\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N) \;\; \forall j \;\text{ s.t. }\; w_j > 0, \tag{3.82}$$

where $\hat{U}$ is a $D \times d$ matrix whose columns are the top d eigenvectors of

$$\hat{\Sigma}_N = \frac{\sum_{j=1}^{N} w_j (\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N)(\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N)^\top}{\sum_{j=1}^{N} w_j}. \tag{3.83}$$

As a consequence, under the reweighted least-squares criterion, finding a robust solution to PCA reduces to finding a robust estimate of the sample mean and the sample covariance of the data by properly setting the weights.

In what follows, we discuss two main approaches for estimating the weights.

[8] In fact, it can be shown that (Ferguson 1961), if the outliers have a Gaussian distribution of a different covariance matrix $a\Sigma$, then ε_i is a sufficient statistic for the test that maximizes the probability of correct decision about the outlier (in the class of tests that are invariant under linear transformations). Interested readers may want to find out how this distance is equivalent (or related) to the sample influence $\hat{\Sigma}_N^{(i)} - \hat{\Sigma}_N$ or the approximate sample influence given in (B.91).

1. *Multivariate Trimming* (MVT) is a popular robust method for estimating the sample mean and covariance of a set of points. This method assumes discrete weights

$$w_j = \begin{cases} 1 & \text{if } \boldsymbol{x}_j \text{ is an inlier,} \\ 0 & \text{if } \boldsymbol{x}_j \text{ is an outlier,} \end{cases} \tag{3.84}$$

and chooses the outliers as a certain percentage of the samples (say 10%) that have relatively large residual. This can be done by simply sorting the residuals $\{\varepsilon_j\}$ from the lowest to the highest and then choosing as outliers the desired percentage of samples with the highest residuals. Once the outliers are trimmed out, one can use the remaining samples to reestimate the subspace as in (3.82)–(3.83). Each time we have a new estimate of the subspace, we can recalculate the residual of every sample and reselect samples that need to be trimmed. We can repeat the above process until a stable estimate of the subspace is obtained. When the percentage of outliers is somewhat known, it usually takes only a few iterations for MTV to converge, and the resulting estimate is in general more robust. However, if the percentage is wrongfully specified, MVT may not converge, or it may converge to a wrong estimate of the subspace. In general, the "breakdown point" of MTV, i.e., the proportion of outliers that it can tolerate before giving a completely wrong estimate, depends only on the chosen trimming percentage.

2. *Maximum-Likelihood-Type Estimators* (M-Estimators) is another popular robust method for estimating the sample mean and covariance of a set of points. As we saw in the case of PCA with corrupted entries, this method assumes continuous weights

$$w_j = \rho(\varepsilon_j)/\varepsilon_j^2 \tag{3.85}$$

for some robust loss function $\rho(\cdot)$. The objective function then becomes

$$\sum_{j=1}^{N} \rho(\varepsilon_j). \tag{3.86}$$

Many loss functions $\rho(\cdot)$ have been proposed in the statistics literature (Huber 1981; Barnett and Lewis 1983). When $\rho(\varepsilon) = \varepsilon^2$, all weights are equal to 1, and we obtain the standard least-squares solution, which is not robust. Other robust loss functions include

(a) L_1 loss: $\rho(\varepsilon) = |\varepsilon|$;
(b) Cauchy loss: $\rho(\varepsilon) = \varepsilon_0^2 \log(1 + \varepsilon^2/\varepsilon_0^2)$;
(c) Huber loss (Huber 1981): $\rho(\varepsilon) = \begin{cases} \varepsilon^2 & \text{if } \ |\varepsilon| < \varepsilon_0, \\ 2\varepsilon_0|\varepsilon| - \varepsilon_0^2 & \text{otherwise;} \end{cases}$
(d) Geman–McClure loss (Geman and McClure 1987): $\rho(\varepsilon) = \frac{\varepsilon^2}{\varepsilon^2 + \varepsilon_0^2}$,

Algorithm 3.9 (Iteratively Reweighted Least Squares for PCA with Outliers)

Input: Data matrix X, dimension d, and parameter $\varepsilon_0 > 0$.

initialize $[\boldsymbol{\mu}, U, Y] = \text{PCA}(X)$ using PCA from Chapter 2.
repeat
  $\varepsilon_j \leftarrow \|\boldsymbol{x}_j - \boldsymbol{\mu} - U\boldsymbol{y}_j\|_2, \quad w_j \leftarrow \frac{\varepsilon_0^2}{\varepsilon_j^2 + \varepsilon_0^2}.$
  $\boldsymbol{\mu} \leftarrow \frac{\sum_{j=1}^N w_j(\boldsymbol{x}_j - U\boldsymbol{y}_j)}{\sum_{j=1}^N w_j}, \quad \Sigma \leftarrow \frac{\sum_{j=1}^N w_j(\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N)(\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N)^\top}{\sum_{j=1}^N w_j}.$
  $U \leftarrow$ top d eigenvectors of Σ.
  $Y \leftarrow U^\top(X - \boldsymbol{\mu}\mathbf{1}^\top)$.
until convergence of $\boldsymbol{\mu}\mathbf{1}^\top + UY$.
$L \leftarrow UY$ and $E \leftarrow X - L - \boldsymbol{\mu}\mathbf{1}^\top$.

Output: $\boldsymbol{\mu}$, U, Y, L and E.

where $\varepsilon_0 > 0$ is a parameter. Given any choice for the weights, one way of minimizing (3.86) with respect to the subspace parameters is to initialize all the weights to $w_j = 1, j = 1, \ldots, N$. This will give an initial estimate for the subspace that is the same as that given by PCA. Given this initial estimate of the subspace, one may compute the weights as $w_j = \rho(\varepsilon_j)/\varepsilon_j^2$ using any of the aforementioned robust cost functions. Given these weights, one can reestimate the subspace from (3.82)–(3.83). One can then iterate between computing the weights given the subspace and computing the subspace given the weights. This iterative process is called iteratively reweighted least squares (IRLS), as in the case of PCA with corrupted entries, and is summarized in Algorithm 3.9 for the Geman-McClure loss function. An alternative method for minimizing (3.86) is simply to do gradient descent. This method may be preferable for loss functions ρ that are differentiable, e.g., the Geman–McClure loss function. One drawback of M-estimators is that their breakdown point is inversely proportional to the dimension of the space. Thus, M-estimators become much less robust when the dimension is high.

Consensus-Based Outlier Detection

This approach assumes that the outliers are not drawn from the same subspace as the rest of the data. Hence it makes sense to try to avoid the outliers when we infer the subspace in the first place. However, without knowing which points are outliers beforehand, how can we avoid them?

One idea is to fit a subspace to a *subset* of the data instead of to all the data points. This is possible when the number of data points required to fit a subspace ($k = d$ for linear subspace or $k = d + 1$ for affine subspaces) is *much* smaller than the size N of the given data set. Of course, we should *not* expect that a randomly chosen subset will have no outliers and always lead to a good estimate of the subspace. Thus, we should try *many different subsets*:

$$\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_m \subset \mathcal{X}, \tag{3.87}$$

where each subset $\mathcal{X}_i$ is independently drawn and contains $k \ll N$ samples.

If the number of subsets is large enough, one of the trials should contain few or no outliers and hence give a "good" estimate of the subspace. Indeed, if p is the fraction of valid samples (the "inliers"), one can show that (see Exercise B.8) with probability $q = 1 - (1 - p^k)^m$, one of the above subsets will contain only valid samples. In other words, if q is the probability that one of the selected subsets contains only valid samples, we need to randomly sample at least

$$m \geq \frac{\log(1-q)}{\log(1-p^k)} \tag{3.88}$$

subsets of k samples.

Now, given multiple subspaces estimated from multiple subsets, the next question is how to select a "good" subspace among them. Let $\hat{S}_i$ be the subspace fit to the set of points in $\mathcal{X}_i$. If the set $\mathcal{X}_i$ is contaminated by outliers, then $\hat{S}_i$ should be a "bad" estimate of the true subspace S, and hence few points in $\mathcal{X}$ should be well fit by $\hat{S}_i$. Conversely, if the set $\mathcal{X}_i$ contains only inliers, then $\hat{S}_i$ should be a "good" estimate of the true subspace S, and many points should be well fit by $\hat{S}_i$. Thus, to determine whether $\hat{S}_i$ is a good estimate of S, we need some criterion to determine when a point is well fit by $\hat{S}_i$ and another criterion to determine when the number of points that are well fit by $\hat{S}_i$ is sufficiently large. We declare that the subset $\mathcal{X}_i$ gives a "good" estimate $\hat{S}_i$ of the subspace S if

$$\#\big\{\boldsymbol{x} \in \mathcal{X} : \operatorname{dist}(\boldsymbol{x}, \hat{S}_i) \leq \tau\big\} \geq N_{\min}, \tag{3.89}$$

where # is the cardinality of the set, $\tau > 0$ is the threshold on the distance from any point $\boldsymbol{x} \in \mathcal{X}$ to the estimated subspace $\hat{S}$ used to determine whether a point is an inlier to $\hat{S}$, and $N_{\min}$ is a threshold on the minimum number of inliers needed to declare that the estimated subspace is "good." If the number of inliers to the subspace estimated from a given subset of the data points is too small, then the process is repeated for another sample of points until a good subspace is found or the maximum number of iterations has been exhausted. Upon termination, PCA is reapplied to all inliers in order to improve the robustness of the estimated subspace to noise. This approach to PCA with outliers is called *random sample consensus* (RANSAC) (Fischler and Bolles 1981) and is summarized in Algorithm 3.10.

One of the main advantages of RANSAC is that in theory, it can tolerate more than 50% outliers; hence it is extremely popular for practitioners who handle grossly contaminated data sets. Nevertheless, the computational cost of this scheme is proportional to the number of candidate subsets needed to ensure that the probability of choosing an outlier-free subset is large enough. This number typically grows exponentially with the subspace dimension and the number of samples. Hence, RANSAC is used mostly in situations in which the subspace dimension is low; in most of the cases we have seen, the subspace dimension does not exceed 10. Another challenge is that in order to design a successful RANSAC algorithm, one

Algorithm 3.10 (Random Sample Consensus for PCA with Outliers)

Input: Data points $\mathcal{X}$, subspace dimension d, maximum number of iterations k, threshold on fitting error τ, threshold on minimum number of inliers $N_{\min}$.

initialization $i = 0$.
while $i < k$ **do**
  $\mathcal{X}_i \leftarrow d+1$ randomly chosen data points from $\mathcal{X}$.
  $\hat{S}_i \leftarrow \text{PCA}(\mathcal{X}_i)$.
  $\mathcal{X}_{inliers} \leftarrow \{\boldsymbol{x} \in \mathcal{X} : \text{dist}(\boldsymbol{x}, \hat{S}_i) \leq \tau\}$.
  if $|\mathcal{X}_{inliers}| \geq N_{\min}$ **then**
    $i \leftarrow k$.
  else
    $i \leftarrow i+1$.
  end if
end while

Output: Estimated subspace $\hat{S} \leftarrow \text{PCA}(\mathcal{X}_{inliers})$ and set of inliers $\mathcal{X}_{inliers}$.

needs to choose a few key parameters carefully, such as the size of every subset (or the subspace dimension), the distance dist and the parameter τ to determine whether a point is an inlier or outlier, and the threshold $N_{\min}$ on the minimum number of inliers to the estimated subspace.

There is a vast amount of literature on RANSAC-type algorithms, especially in computer vision (Steward 1999). For more details on RANSAC and other related random sampling techniques, the reader is referred to Appendix B.

3.3.2 Outlier Detection by Convex Optimization

So far, we have presented classical techniques from the robust statistics literature and shown how they can be used for dealing with outliers in the context of PCA. The techniques presented so far are generally simple and intuitive. However, they do not provide clear conditions under which they can guarantee the correctness or global optimality of their solutions. To address this issue, in what follows we will present alternative approaches based on convex optimization for dealing with outliers in the context of PCA. As we will see, when the dimension of the subspace is small enough and the percentage of outliers is small enough, it is possible to perfectly recover which data points are inliers and which ones are outliers.

Outlier Detection by ℓ_1 Minimization
Let $\mathcal{X} = \{\boldsymbol{x}_j\}_{j=1}^N$ be a collection of points in $\mathbb{R}^D$. Assume that $N_{in} \leq N$ points are drawn from a linear subspace $S \subset \mathbb{R}^D$ of dimension $d \ll D$ and that the remaining $N_{out} = N - N_{in}$ data points do not belong to S. We thus have $N = N_{in} + N_{out}$ data points, where N_{in} points are inliers and N_{out} points are outliers. Assume also that there are d linearly independent data points among the inliers. Then every point

$\boldsymbol{x} \in S$ can be written as a linear combination of the inliers. In fact, every point $\boldsymbol{x} \in S$ can be written as a linear combination of at most d inliers. More generally, we can write $\boldsymbol{x} \in S$ as a linear combination of all N data points as

$$\boldsymbol{x} = \sum_{j=1}^{N} \boldsymbol{x}_j c_j = X\boldsymbol{c} \quad \text{where} \quad X = \left[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\right] \in \mathbb{R}^{D \times N}, \tag{3.90}$$

and set $c_j = 0$ whenever $\boldsymbol{x}_j$ is an outlier. Hence, there exists a solution $\boldsymbol{c}$ of $\boldsymbol{x} = X\boldsymbol{c}$ with at most d nonzero entries, which correspond to any d inliers that span S. Therefore, an optimal solution $\boldsymbol{c}^*$ to the following optimization problem

$$\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_0 \quad \text{s.t.} \quad \boldsymbol{x} = X\boldsymbol{c} \tag{3.91}$$

should be d-sparse, i.e., it should have at most d nonzero entries, i.e., $\|\boldsymbol{c}^*\|_0 \leq d$.

Assume now that there are D linearly independent data points among both the inliers and outliers. Assume also that $\boldsymbol{x}$ does not belong to the subspace S. Then we can still express $\boldsymbol{x}$ as a linear combination of all data points as $\boldsymbol{x} = X\boldsymbol{c}$. However, when $\boldsymbol{x}$ is an arbitrary point in $\mathbb{R}^D$, we no longer expect $\boldsymbol{c}$ to be d-sparse. In fact, in general, we expect at least D entries of $\boldsymbol{c}$ to be nonzero, i.e., $\|\boldsymbol{c}\|_0 \geq D$. Of course, in some rare circumstances it could be the case that $\boldsymbol{x}$ is a linear combination of two outliers in the data, in which case we can choose $\boldsymbol{c}$ such that $\|\boldsymbol{c}\|_0 = 2$. However, such cases occur with extremely low probability.

The above discussion suggests a simple procedure to determine whether a point $\boldsymbol{x}$ is an inlier: we try to express $\boldsymbol{x}$ as a linear combination of the data points in $\mathcal{X}$ with the sparsest possible coefficients $\boldsymbol{c}$, as in (3.91). If the optimal solution $\boldsymbol{c}^*$ is d-sparse, then $\boldsymbol{x}$ is an inlier; otherwise, $\boldsymbol{x}$ is an outlier. In practice, however, we face a couple of challenges that prevent us from implementing this simple strategy.

1. The optimization problem in (3.91) is NP-hard (Amaldi and Kann 1998). Intuitively this is because there are numerous choices of d out of N nonzero entries in $\boldsymbol{c}$, and for each such choice, we need to check whether a linear system has a solution or not.
2. While in general we expect that $\|\boldsymbol{c}\|_0 \gg d$ when $\boldsymbol{x}$ is an outlier, this may not always be the case. Thus, we may be interested in characterizing whether for some distribution of the outliers we can guarantee that $\|\boldsymbol{c}\|_0 \gg d$ with high probability. Moreover, since the subspace dimension d may not be known a priori, we may want to declare $\boldsymbol{x}$ an outlier if $\|\boldsymbol{c}\|_0 > \lambda D$ for some $\lambda < 1$. This may require some mechanism for determining λ.
3. In practice, we are not trying to determine whether a generic data point $\boldsymbol{x}$ is an inlier or an outlier, but rather whether one of the given data points, say $\boldsymbol{x}_j$, is an inlier or an outlier. Trivially, $\boldsymbol{x}_j$ has a 1-sparse representation with respect to X, i.e., $\boldsymbol{x}_j = \boldsymbol{x}_j$. Thus, we need a mechanism to prevent this trivial solution.

To address the first issue, as we have learned from the brief survey of compressive sensing in Section 3.2.2, an effective technique to obtain a sparse solution is to replace the ℓ_0-minimization problem in (3.91) by the ℓ_1-minimization problem

$$\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_1 \quad \text{s.t.} \quad \boldsymbol{x} = X\boldsymbol{c}. \tag{3.92}$$

In particular, it follows from Theorem 3.10 that if $X = \left[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\right] \in \mathbb{R}^{D \times N}$ is an arbitrary matrix whose columns are of unit norm, i.e., $\|\boldsymbol{x}_j\|_2 = 1$ for all $j = 1, \ldots, N$ and $\boldsymbol{c}_0 \in \mathbb{R}^N$ is a d-sparse vector, then given $\boldsymbol{x} = X\boldsymbol{c}_0$, we can recover $\boldsymbol{c}_0$ by solving the optimization problem in (3.92) when the matrix X is incoherent or satisfies the RIP. In other words, the sparsest solution to the linear system $X\boldsymbol{c} = \boldsymbol{x}$ can be obtained by solving the convex ℓ_1-minimization problem in (3.92) as opposed to the NP-hard ℓ_0-minimization problem in (3.91).

The fundamental question is whether the conditions under which the solution to (3.92) coincides with that of (3.91) are satisfied by a data matrix X with N_{in} data points in a linear subspace of dimension d and N_{out} points not in the subspace. Unfortunately, this is not the case: the matrix of inliers X_{in} cannot be incoherent according to Definition 3.8, because it is not of full column rank. For instance, if $\text{rank}(X_{in}) = 1$, then X has maximum coherence $\nu(X_{in}) = 1$.

Does this mean that we cannot use ℓ_1-minimization? As it turns out, we can still use ℓ_1 minimization to recover a sparse representation of a point $\boldsymbol{x} \in S$. The reason is that the conditions in Theorem 3.10 aim to guarantee that we can recover a *unique* sparse solution, while here the solution for $\boldsymbol{c}$ is not always unique, and thus we cannot hope for the ℓ_1-minimization problem to give us a unique sparse solution to begin with. Indeed, if $\boldsymbol{x} \in S$ and $N_{in} > d$, then there may be many ways in which we may express a point in S as a linear combination of d inliers. Therefore, our goal is not to find a unique representation of $\boldsymbol{x}$ in terms of d inliers, but rather to find any representation of $\boldsymbol{x}$ in terms of any d inliers. As a consequence, we do not need the matrix of inliers to be incoherent. All we need is for the set of inliers to be incoherent with the set of outliers. More precisely, if $\mathcal{I}_{in}$ is the set of inliers and $\mathcal{I}_{out}$ is the set of outliers, all we need is that

$$\max_{j \in \mathcal{I}_{in}} \max_{k \in \mathcal{I}_{out}} |\boldsymbol{x}_j^\top \boldsymbol{x}_k| < \frac{1}{2d-1}. \tag{3.93}$$

This is in contrast to the classical condition on the mutual coherence of X in Definition 3.8, which is given by $\max_{j \neq k} |\boldsymbol{x}_j^\top \boldsymbol{x}_k| < \frac{1}{2d-1}$.

To address the second issue, we assume from now on that all N points are of unit norm, i.e., they lie in the $D-1$ dimensional sphere $\mathbb{S}^{D-1}$. We assume also that the outliers are drawn uniformly at random from $\mathbb{S}^{D-1}$. Moreover, since we will be solving an ℓ_1 minimization problem, we may want to use the ℓ_1 norm of $\boldsymbol{c}$ to determine whether $\boldsymbol{x}$ is an inlier or outlier. More specifically, when $\boldsymbol{x}$ is an inlier, we expect $\|\boldsymbol{c}\|_0 = d$; hence we expect $\|\boldsymbol{c}\|_1 = \sqrt{d}$. Likewise, when $\boldsymbol{x}$ is an outlier,

we expect $\|\boldsymbol{c}\|_0 = D$, and then we expect $\|\boldsymbol{c}\|_1 = \sqrt{D}$. Therefore, we may want to declare $\boldsymbol{x}$ an outlier if $\|\boldsymbol{c}\|_1 > \lambda\sqrt{D}$ for some λ.

The third issue is relatively easy to address. When we find the sparse solution for the point $\boldsymbol{x}_j$ with respect to X, we need to enforce only that the jth entry of $\boldsymbol{c}$ is zero, so that $\boldsymbol{x}_j$ is not represented by itself. This leads to the following convex optimization problem:

$$\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_1 \quad \text{s.t.} \quad \boldsymbol{x}_j = X\boldsymbol{c} \quad \text{and} \quad c_j = 0, \tag{3.94}$$

which can be solved easily using existing ℓ_1-minimization techniques.

The following result, which follows as a direct corollary of (Soltanolkotabi and Candès 2013, Theorem 1.3) (see also Theorem 8.27), shows how the optimal solution to (3.94) can be used to distinguish inliers from outliers.

Theorem 3.13. *Let S be a randomly chosen subspace of $\mathbb{R}^D$ of dimension d. Suppose there are $N_{in} = \rho d + 1$ inlier points chosen independently and uniformly at random in $S \cap \mathbb{S}^{D-1}$, where $\rho > 1$. Suppose there are N_{out} points chosen independently and uniformly at random in $\mathbb{S}^{D-1}$. Let $\boldsymbol{x}_j \in \mathbb{S}^{D-1}$ be the jth data point and let $\boldsymbol{c} \in \mathbb{R}^N$ be the solution to the ℓ_1-minimization problem in (3.94). Declare $\boldsymbol{x}_j$ to be an outlier if $\|\boldsymbol{c}\|_1 > \lambda(\gamma)\sqrt{D}$, where $\gamma = \frac{N-1}{D}$, $N = N_{in} + N_{out}$, and*

$$\lambda(\gamma) = \begin{cases} \sqrt{\frac{2}{\pi}}\frac{1}{\sqrt{\gamma}}, & 1 \le \gamma \le e \\ \sqrt{\frac{2}{\pi e}}\frac{1}{\sqrt{\log \gamma}}, & \gamma \ge e. \end{cases} \tag{3.95}$$

If the number of outliers is such that

$$N_{out} < \frac{1}{D}\exp(c_1\sqrt{D}) - N_{in} \tag{3.96}$$

for some constant $c_1 > 0$, then the method above detects all the outliers with probability at least $1 - N_{out}\exp(-c_2 D/\log(N_{in} + N_{out}))$ for some constant $c_2 > 0$. Moreover, if the number of outliers is such that

$$N_{out} < D\rho^{c_3\frac{D}{d}} - N_{in} \tag{3.97}$$

for some constant $c_3 > 0$, then the method above does not detect any point in S as an outlier with probability at least $1 - N_{out}\exp(-c_4 D/\log(N_{in} + N_{out})) - N_{in}\exp(-\sqrt{\rho}d)$ for some constant $c_4 > 0$.

Outlier Detection by $\ell_{2,1}$ Minimization

An alternative approach to outlier detection in PCA is based on the observation that the data matrix X can be seen as a low-rank matrix with sparsely corrupted columns that correspond to the outliers. More specifically, the matrix X can be decomposed as

$$X = L_0 + E_0. \tag{3.98}$$

The jth column of L_0 is equal to $\boldsymbol{x}_j$ if it is an inlier to the subspace and is equal to $\mathbf{0}$ otherwise. Therefore, L_0 is of rank d and spans the same subspace as the inliers. Conversely, the jth column of E_0 is equal to $\boldsymbol{x}_j$ if it is an outlier to the subspace and is equal to $\mathbf{0}$ otherwise. Therefore, the nonzero columns of E_0 contain the outliers. If we assume that the fraction γ of outliers is small, then the matrix E_0 is *column sparse*.

Obviously, such a decomposition is ill posed (at least ambiguous) if the matrix X or L_0 is also column sparse. Therefore, in order for the decomposition to be unique, the matrix L_0 cannot be column sparse on the $(1-\gamma)N$ columns on which it can be nonzero. To ensure that this is the case, we need to introduce a *column incoherence* condition:

Definition 3.14 (Matrix Incoherence with Respect to Column Sparse Matrices)**.** *A rank-d matrix $L \in \mathbb{R}^{D\times N}$ with compact SVD $L = U\Sigma V^\top$ and $(1-\gamma)N$ nonzero columns is said to be* ν-incoherent with respect to the set of column sparse matrices *if*

$$\max_j \|\boldsymbol{v}_j\|^2 \leq \frac{\nu d}{(1-\gamma)N}, \tag{3.99}$$

where $\boldsymbol{v}_j$ is the jth row of V.

Following the discussion after Definition 3.1, notice that since $V \in \mathbb{R}^{N\times d}$ is orthonormal, the largest absolute value of the entries of V is equal to 1, which happens when a column of V is 1-sparse. On the other hand, if all columns of V are so dense that all their $(1-\gamma)N$ nonzero entries are equal to each other up to sign, then each entry is equal to $\pm 1/\sqrt{(1-\gamma)N}$, and the norm of each row is $\sqrt{d/(1-\gamma)N}$. Therefore, when $\nu < 1$, the condition above controls the level of sparsity of V. As argued before, from a probabilistic perspective, this condition is rather mild in the sense that it holds for almost all generic matrices: a random (say Gaussian) matrix satisfies this condition with high probability when the dimension of the matrix is large enough. As we will see, incoherence with respect to column sparse matrices is a very useful technical condition to ensure that outlier detection is a meaningful problem.

Now, even though the incoherence condition may ensure that the above low-rank plus column-sparse decomposition problem is well posed, there is no guarantee that one can find the correct decomposition efficiently. As before, we may formulate the problem of recovering L_0 and E_0 as a rank minimization problem:

$$\min_{L,E} \quad \operatorname{rank}(L) + \lambda \|E\|_{2,0} \quad \text{s.t.} \quad X = L + E, \tag{3.100}$$

where $\|E\|_{2,0} = \sum_{j=1}^N 1(\|\boldsymbol{e}_j\|_2 \neq 0)$ is the number of nonzero columns in the matrix of outliers $E = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N]$. However, since this problem is NP-hard, we need to resort to a proper relaxation. For this purpose, we can use a norm that promotes columnwise sparsity, such as the $\ell_{2,1}$ norm of E:

$$\|E\|_{2,1} = \sum_{j=1}^{N} \|\boldsymbol{e}_j\|_2, \tag{3.101}$$

which is the sum of the ℓ_2 norms of all the columns of E. Notice that if we collect all the ℓ_2 norms of the columns of E as a vector $\boldsymbol{e} = [\|\boldsymbol{e}_1\|_2, \ldots, \|\boldsymbol{e}_N\|_2]^\top$, then the above norm is essentially the ℓ_1 norm of the vector, $\|\boldsymbol{e}\|_1$; hence it measures how sparse the columns are. Notice also that $\|E\|_{2,0} = \|\boldsymbol{e}\|_0$.

Similar to the PCP optimization problem in (3.64) for PCA with robustness to intrasample outliers, we can use the convex optimization

$$\min_{L,E} \quad \|L\|_* + \lambda \|E\|_{2,1} \quad \text{s.t.} \quad X = L + E \tag{3.102}$$

to decompose sparse column outliers in the data matrix X from the low-rank component. This convex program is called *outlier pursuit*.

One can rigorously show that under certain benign conditions, the outlier pursuit program can correctly identify the set of sparse (column) outliers.

Theorem 3.15 (Robust PCA by Outlier Pursuit (Xu et al. 2010))**.** *Let $X = L_0 + E_0$ be a given $D \times N$ matrix. Assume that L_0 is ν-incoherent with respect to the set of column-sparse matrices according to Definition 3.14. Assume also that E_0 is supported on at most γN columns. If*

$$\operatorname{rank}(L_0) \leq \frac{c_1(1-\gamma)}{\gamma \nu}, \tag{3.103}$$

where $c_1 = \frac{9}{121}$, then the solution (L^*, E^*) to the outlier pursuit program (3.102) with λ set to be $\frac{3}{7\sqrt{\gamma N}}$ recovers the low-dimensional column space of L_0 exactly and identifies exactly the indices of columns corresponding to outliers not lying in the column space.

If the data also contain small noise $X = L_0 + E_0 + Z$, where Z is a random Gaussian matrix that models small noise in the data, then we can modify the outlier pursuit program as

$$\min_{L,E} \quad \|L\|_* + \lambda \|E\|_{2,1} \quad \text{s.t.} \quad \|X - L - E\|_2^2 \leq \varepsilon^2, \tag{3.104}$$

where ε is the noise variance. It can be shown that under conditions similar to those in the above theorem, this program gives a stable estimate of the correct solution. For more details, we refer the reader to (Xu et al. 2010).

Using optimization techniques introduced in Appendix A, one can easily develop ALM- or ADMM-based algorithms to solve the above convex optimization problems. We leave that to the reader as an exercise (see Exercise 3.8).

Fig. 3.6 Example images taken from the Caltech 101 data set. These images are then resized to 96×84 and used as outliers for the experiments below.

Example 3.16 (Outlier Detection among Face Images) Sometimes a face image data set can be contaminated by images of irrelevant objects, like many imperfectly sorted data sets in the Internet. In this case, it would be desirable to detect and remove such irrelevant outliers from the data set. In this example, we illustrate how to do this with the outlier detection methods introduced in this section.

As in previous experiments, we take as inliers the frontal face images of subject 20 under 64 different illumination conditions in the extended Yale B data set. For outlier images, we randomly select some pictures from the Caltech 101 data set (Fei-Fei et al. 2004) and merge them into the face image data set. Some typical examples of such pictures are shown in Figure 3.6. All the inlier and outlier images are normalized to size 96×84.

We use the outlier pursuit method, which is based on solving (3.102), to decompose the data matrix into a low-rank part L and a sparse-column term E. In this experiment, we set the parameter of the method according to Theorem 3.15 with a multiplication factor of 3, i.e., we set $\lambda = 3 \times \lambda_0$ where $\lambda_0 = \frac{3}{7\sqrt{\gamma N}}$.

Ideally, columns of E with large magnitude correspond to outliers. To show how the method performs, we apply it to data sets with increasing percentages of outliers. We compute for each column of E its ℓ_2 norm to measure whether it is an outlier. True outliers are marked in red. The results for varying percentages of outliers are shown in Figure 3.7. As we can see from the results, up to nearly 50% outliers, the outliers have significantly larger norm than the inliers.

3.4 Bibliographic Notes

PCA with Robustness to Missing Entries

The problem of completing a low-rank matrix with missing entries has a very long and rich history. Starting with the original work of (Wiberg 1976), one can refer to (Johnson 1990) for a survey on some of the early developments on this topic.

Since then, this problem has drawn tremendous interest, particularly in computer vision and pattern recognition, where researchers needed to complete data with missing entries due to occlusions. For instance, many algorithms were proposed

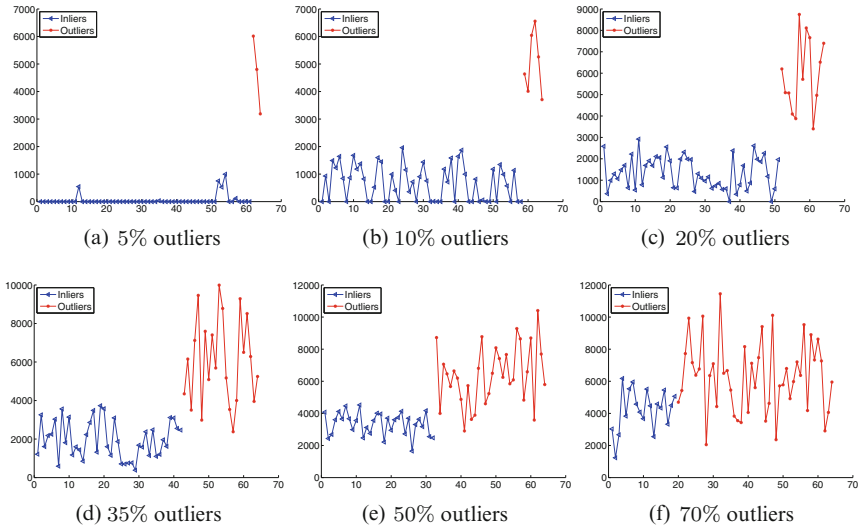


Fig. 3.7 Outlier detection among face images. In each experiment, we use 64 images, in which a certain percentage of images are selected randomly from the Caltech 101 data set as outliers, and the rest are taken randomly from the 64 illuminations of frontal face images of subject 20 in extended Yale B. We plot the column ℓ_2 norm of the matrix E given by the convex optimization method, with ground truth outliers marked as red.

to solve matrix completion problems in the late 1990s and early 2000s, including (Shum et al. 1995; Jacobs 2001; H.Aanaes et al. 2002; Brandt 2002) for the purpose of reconstructing a 3D scene from a collection of images. The power factorization method featured in this chapter was proposed in (Hartley and Schaffalitzky 2003) for the same purpose, while a variant of the EM algorithm we described appeared in (Gruber and Weiss 2004). Also, the work of (Ke and Kanade 2005) proposed the use of the ℓ_1 norm for matrix completion and recovery, which extends the original Wiberg method(Wiberg 1976) from the ℓ_2 to the ℓ_1 norm. A survey and evaluation of state-of-the-art methods for solving the matrix completion problem can be found in (Buchanan and Fitzgibbon 2005).

However, all of the work described so far has focused primarily on developing algorithms for completing a matrix, without any guarantees of correctly recovering the original low-rank matrix. The seminal work of (Recht et al. 2010; Candès and Recht 2009) has shown that under broad conditions, one can correctly recover a low-rank matrix with a significant percentage of missing entries using convex optimization (i.e., minimizing the nuclear norm of the matrix). This has inspired a host of work on developing ever stronger conditions and more efficient algorithms for low-rank matrix completion (Cai et al. 2008; Candès and Tao 2010; Keshavan et al. 2010b; Gross 2011; Keshavan et al. 2010a; Zhou et al. 2010a), including work that extends to the case of noisy data (Candès and Plan 2010).

PCA with Robustness to Corrupted Entries and Outliers
Regarding the robust recovery of a low-rank matrix, it was first proposed by (Wright et al. 2009a; Chandrasekaran et al. 2009) to use the convex relaxation (3.64) to solve the robust PCA problem. This formulation was soon followed by a rather strong theoretical justification (Candès et al. 2011) and efficient algorithms (Lin et al. 2011). This has made convex relaxation a very successful and popular technique for robust low-rank matrix recovery or outlier rejection, leading to extensions to many different settings and more scalable convex optimization algorithms.

Revival of the Factorization Approach
Due to the advent of large data sets and large-scale problems, there has been a revival of factorization (alternating minimization) approaches with theoretical guarantees of correctness for low-rank matrix completion and recovery, including the very interesting work of (Jain et al. 2012; Keshavan 2012; Hardt 2014; Jain and Netrapalli 2014). The more recent work of (Udell et al. 2015) further generalizes the factorization framework to situations in which the factors are allowed to have additional structures; and (Haeffele and Vidal 2015) combines factorization with certain nonlinear mappings typically used in a deep learning framework.

3.5 Exercises

Exercise 3.1 (Data Completion with the Subspace Known). Show that the solution to the problem (3.5) is given by the formula in (3.6).

Exercise 3.2. For the PPCA model with missing data discussed in Section 3.1.2, show that the conditional distribution of $\boldsymbol{x}_U$ given $\boldsymbol{x}_O$ is Gaussian with the following mean vector and covariance matrix:

$$\boldsymbol{\mu}_{U|O} = \boldsymbol{\mu}_U + \Sigma_{UO}\Sigma_{OO}^{-1}(\boldsymbol{x}_O - \boldsymbol{\mu}_O) \text{ and } \Sigma_{U|O} = \Sigma_{UU} - \Sigma_{UO}\Sigma_{OO}^{-1}\Sigma_{OU}.$$

Exercise 3.3 (Orthogonal Power Iteration Method). Let $A \in \mathbb{R}^{N\times N}$ be a symmetric positive semidefinite matrix with eigenvectors $\{\boldsymbol{u}_i\}_{i=1}^N$ and eigenvalues $\{\lambda_i\}_{i=1}^N$ sorted in descending order. Assume that $\lambda_1 > \lambda_2$ and let $\boldsymbol{u}^0$ be an arbitrary vector not orthogonal to $\boldsymbol{u}_1$, i.e., $\boldsymbol{u}_1^\top \boldsymbol{u}^0 \neq 0$. Consider the sequence of vectors

$$\boldsymbol{u}_{k+1} = \frac{A\boldsymbol{u}_k}{\|A\boldsymbol{u}_k\|}. \tag{3.105}$$

1. Show that there exist $\{\alpha_i\}_{i=1}^N$ with $\alpha_1 \neq 0$ such that

$$\boldsymbol{u}^k = A^k\boldsymbol{u}^0 = \sum_{i=1}^N \alpha_i\lambda_i^k\boldsymbol{u}_i. \tag{3.106}$$

2. Use this expression to show that $\boldsymbol{u}^k$ converges to $\frac{\alpha_1}{|\alpha_1|}\boldsymbol{u}_1$ with rate $\frac{\lambda_2}{\lambda_1}$. That is, show that there exists a constant $C > 0$ such that for all $k \geq 0$,

$$\left\| \boldsymbol{u}^k - \frac{\alpha_1}{|\alpha_1|}\boldsymbol{u}_1 \right\| \leq C\left(\frac{\lambda_2}{\lambda_1}\right)^k. \tag{3.107}$$

3. Assume that $\lambda_d > \lambda_{d+1}$ and let $U^0 \in \mathbb{R}^{N\times d}$ be an arbitrary matrix whose column space is not orthogonal to the subspace $\{\boldsymbol{u}_i\}_{i=1}^d$ spanned by the top d eigenvectors of A. Consider the sequence of matrices

$$U^{k+1} = AU^k(R^k)^{-1}, \tag{3.108}$$

where $Q^kR^k = AU^k$ is the QR decomposition of AU^k. Show that U^k converges to a matrix U whose columns are the top d eigenvectors of A. Moreover, show that the rate of convergence is $\frac{\lambda_{d+1}}{\lambda_d}$.

Exercise 3.4 (Convergence of Orthogonal Power Iteration). Prove Theorem 3.4.

Exercise 3.5 (Properties of the ℓ_1 Norm). Let X be a matrix.

1. Show that the ℓ_1 norm $f(X) = \|X\|_1 = \sum_{ij} |X_{ij}|$ of X is a convex function of X.
2. Show that the subgradient of the ℓ_1 norm is given by

$$\partial\|X\|_1 = \operatorname{sign}(X) + W, \tag{3.109}$$

where W is a matrix such that $\max_{ij} |W_{ij}| \leq 1$.

3. Show that the optimal solution of

$$\min_A \quad \frac{1}{2}\|X - A\|_F^2 + \tau\|A\|_1 \tag{3.110}$$

is given by $A = \mathcal{S}_\tau(X)$, where $\mathcal{S}_\tau(x) = \operatorname{sign}(x)\max(|x| - \tau, 0)$ is the soft-thresholding operator applied entrywise to X.

Exercise 3.6 (Properties of the Weighted Nuclear Norm). Consider the following optimization problem:

$$\min_A \quad \frac{1}{2}\|X - A\|_F^2 + \tau\phi(A), \tag{3.111}$$

where $\phi(A) = \sum_{i=1}^r w_i\sigma_i(A)$ is the weighted sum of singular values of A with $w_i \geq 0$. Show that

1. $\phi(A)$ is convex when w_i is monotonically decreasing. Please derive the optimal solution under this condition.
2. Is $\phi(A)$ still convex if w_i is an increasing sequence of weights? Why?

Exercise 3.7 (Properties of the $\ell_{2,1}$ Norm).

1. Let $\boldsymbol{x}$ be a vector. Show that the subgradient of the ℓ_2 norm is given by

$$\partial \|\boldsymbol{x}\|_2 = \begin{cases} \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2} & \text{if } \boldsymbol{x} \neq \boldsymbol{0}, \\ \boldsymbol{w} : \|\boldsymbol{w}\|_2 \leq 1\} & \text{if } \boldsymbol{x} = \boldsymbol{0}. \end{cases} \tag{3.112}$$

2. Let X be a matrix. Show that the $\ell_{2,1}$ norm $f(X) = \|X\|_{2,1} = \sum_j \|X_{\cdot j}\|_2 = \sum_j \sqrt{\sum_i X_{ij}^2}$ of X is a convex function of X.
3. Show that the subgradient of the $\ell_{2,1}$ norm is given by

$$(\partial \|X\|_{2,1})_{ij} = \begin{cases} \frac{X_{ij}}{\|X_{\cdot j}\|_2} & X_{\cdot j} \neq \boldsymbol{0} \\ W_{ij} : \|W_{\cdot j}\|_2 \leq 1 & X_{\cdot j} = \boldsymbol{0}. \end{cases} \tag{3.113}$$

4. Show that the optimal solution of

$$\min_A \quad \frac{1}{2}\|X - A\|_F^2 + \tau \|A\|_{2,1} \tag{3.114}$$

is given by $A = X\mathcal{S}_\tau(\text{diag}(\boldsymbol{x}))\text{diag}(\boldsymbol{x})^{-1}$, where $\boldsymbol{x}$ is a vector whose jth entry is given by $x_j = \|X_{\cdot j}\|_2$, and $\text{diag}(\boldsymbol{x})$ is a diagonal matrix with the entries of $\boldsymbol{x}$ along its diagonal. By convention, if $x_j = 0$, then the jth entry of $\text{diag}(\boldsymbol{x})^{-1}$ is also zero.

Exercise 3.8. Let $X = L_0 + E_0$ be a matrix formed as the sum of a low-rank matrix L_0 and a matrix of corruptions E_0, where the corruptions can be either outlying entries (gross errors) or outlying data points (outliers).

1. **(PCA with robustness to outliers).** Assuming that the matrix X is fully observed and that the matrix E_0 is a matrix of outliers, propose an algorithm for solving the outlier pursuit problem (3.102):

$$\min_{L,E} \quad \|L\|_* + \lambda \|E\|_{2,1} \quad \text{s.t.} \quad X = L + E. \tag{3.115}$$

2. **(PCA with robustness to missing entries and gross errors).** Assuming that you observe only a fraction of the entries of X as indicated by a set Ω and that the matrix E_0 is a matrix of gross errors, propose an algorithm for solving the following optimization problem:

$$\min_{L,E} \quad \|L\|_* + \lambda \|E\|_1 \quad \text{s.t.} \quad \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(L + E). \tag{3.116}$$

Exercise 3.9 (Implementation of Power Factorization (PF), Expectation Maximization (EM), and Low-Rank Matrix Completion (LRMC)). Implement the functions below using as few lines of MATLAB code as possible. Compare the

performance of these methods: which method works better and which regime is best (e.g., depending on the percentage of missing entries, subspace dimension d/D)?

Function `[mu,U,Y]=pf(X,d,W)`

Parameters

`X` $D \times N$ data matrix.

`d` Number of principal components.

`W` $D \times N$ binary matrix denoting known (1) or missing (0) entries

Returned values

`mu` Mean of the data.

`U` Orthonormal basis for the subspace.

`Y` Low-dimensional representation (or principal components).

Description

Finds the d principal components of a set of points from the data X with incomplete entries as specified in W using the power factorization algorithm.

Function `[mu,U,sigma]=emppca(X,d,W)`

Parameters

`X` $D \times N$ data matrix.

`d` Number of principal components.

`W` $D \times N$ binary matrix denoting known (1) or missing (0) entries

Returned values

`mu` Mean of the data.

`U` Basis for the subspace (does not need to be orthonormal).

`sigma` Standard deviation of the noise.

Description

Finds the parameters of the PPCA model $\boldsymbol{\mu}$ and $\Sigma = UU^\top + \sigma^2 I$ from the data X with incomplete entries as specified in W using the expectation maximization algorithm.

Function `A=lrmc(X,tau,W)`

Parameters

`X` $D \times N$ data matrix.

τ Parameter of the augmented Lagrangian.

`W` $D \times N$ binary matrix denoting known (1) or missing (0) entries

Returned values

`A` Low-rank completion of the matrix X.

Description

Finds the low-rank approximation of a matrix X with incomplete entries as specified in W using the low-rank matrix completion algorithm based on the augmented Lagrangian method.

Exercise 3.10 (Implementation of IRLS and ADMM Methods for Robust PCA). Implement Algorithms 3.7 and 3.8 for the functions below using as few lines of MATLAB code as possible. Compare the performance of these methods: which method works better and which regime is best (e.g., depending on percentage of corrupted entries (or corrupted data points), subspace dimension d/D)?

Function `[mu,U,Y]=rpca_irls(X,d,sigma)`

Parameters

`X` $D \times N$ data matrix.

`d` Number of principal components.

Returned values

`mu` Mean of the data.

`U` Basis for the subspace.

Description

Finds the parameters of the PCA model $\boldsymbol{\mu}$ and U and the low-dimensional representation using reweighted least squares with weights $w(e) = \frac{\sigma^2}{e^2+\sigma^2}$.

Function `[L,E]=rpca_admm(X,tau,'method')`

Parameters

`X` $D \times N$ data matrix.

τ Parameter of the augmented Lagrangian.

`method` 'L1' for gross errors or 'L21' for outliers

Returned values

`L` Low-rank completion of the matrix X.

`E` Matrix of errors.

Description

Solves the optimization problem $\min_{L,E} \|L\|_* + \lambda \|E\|_\ell$ subject to $X = L + E$ where $\ell = \ell_1$ or $\ell = \ell_{2,1}$ using the ADMM algorithm.

Exercise 3.11 (Robust Face Recognition with Varying Illumination). In this exercise, you will use a small subset of the Yale B data set[9] that contains photos of ten individuals under various illumination conditions. Specifically, you will use only images from the first three individuals under ten different illumination conditions. Divide these images into two sets: *Training Set* (images 1–5 from individuals 1 to 3) and *Test Set* (images 6–10 from individuals 1–3). Notice also that there are five nonface images (accessible as images 1–5 from individual 4). We will refer to these as the *Outlier Set*. Download the file YaleB-Dataset.zip. This file contains the images along with the MATLAB function `loadimage.m`. Decompress the file and type `help loadimage` at the MATLAB prompt to see how to use this function. The function operates as follows.

[9] http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html.

Function `img=loadimage(individual,condition)`

Parameters

`individual`	Number of the individual.
`condition`	Number of the image for that individual.

Returned values

`img`	The pixel image loaded from the database.

Description

Read and resize an image from the data set. The database (directory `images`) must be in the same directory as this file.

1. **Face completion.** Remove uniformly at random 0%, 10%, 20%, 30%, and 40% of the entries of all images of individual 1. Apply the low-rank matrix completion (LRMC) algorithm in Exercise 3.9 to these images to compute the mean face and the eigenfaces as well as to fill in the missing entries. Note that LRMC does not compute the mean face, so you will need to modify the algorithm slightly. Plot the mean face and the top three eigenfaces and compare them to what you obtained with PCA in Chapter 2. Plot also the completed faces and comment on the quality of completion as a function of the percentage of missing entries by visually comparing the original images (before removing the missing entries) to the completed ones. Plot also the error (Frobenius norm) between the original images and the completed ones as a function of the percentage of missing entries and comment on your results. Repeat for individuals 2 and 3.
2. **Face recognition with missing entries.** Remove uniformly at random 0%, 10%, 20%, 30%, and 40% of the entries of all images in the *Training Set* and *Test Set*. Apply the low-rank matrix completion (LRMC) algorithm that you implemented in part (a) to the images in the *Training Set*. Plot the projected training images $\boldsymbol{y} \in \mathbb{R}^d$ for $d = 2$ and $d = 3$ using different colors for the different classes. Do faces of different individuals naturally cluster in different regions of the low-dimensional space? Classify the faces in the *Test Set* using 1-nearest-neighbor. That is, label an image $\boldsymbol{x}$ as corresponding to individual i if its projected image $\boldsymbol{y}$ is closest to a projected image $\boldsymbol{y}_j$ of individual i. Notice that you will need to develop new code to project an image with missing entries $\boldsymbol{x}$ onto the face subspace you already estimated from the *Training Set*, which you can do as described in Section 3.1 of this book. Report the percentage of correctly classified face images for $d = 1, \ldots, 10$ and the percentage of missing entries $\{0, 10, 20, 30, 40\}\%$.
3. **Face correction.** Remove uniformly at random 0%, 10%, 20%, 30%, and 40% of the entries of all images of individual 1 and replace them by arbitrary values chosen uniformly at random from $[0, 255]$. Apply the PCP algorithm, Algorithm 3.8, for corrupted entries that you implemented in Exercise 3.10 to these images to compute the mean face and the eigenfaces as well as correct the corrupted entries. Note that RPCA does not compute the mean face, so you will

need to modify the algorithm accordingly. Plot the mean face and the top three eigenfaces and compare them to what you obtained with PCA from Chapter 2. Plot also the corrected faces and comment on the quality of correction as a function of the percentage of corrupted entries by visually comparing the original images (before removing the missing entries) to the completed ones. Plot also the error (Frobenius norm) between the original images and the corrected ones as a function of the percentage of corrupted entries and comment on your results. Repeat for individuals 2 and 3.

4. **Face recognition with corrupted entries.** Remove uniformly at random 0%, 10%, 20%, 30%, and 40% of the entries of all images of individual 1 and replace them by arbitrary values chosen uniformly at random from $[0, 255]$. Apply the RPCA algorithm for corrupted entries that you implemented in part (a) to the images in the *Training Set*. Plot the projected training images $\boldsymbol{y} \in \mathbb{R}^d$ for $d = 2$ or $d = 3$ using different colors for the different classes. Do faces of different individuals naturally cluster in different regions of the low-dimensional space? Classify the faces in the *Test Set* using 1-nearest-neighbor. That is, label an image $\boldsymbol{x}$ as corresponding to individual i if its projected image $\boldsymbol{y}$ is closest to a projected image $\boldsymbol{y}_j$ of individual i. Notice that you will need to develop new code to project an image with corrupted entries $\boldsymbol{x}$ onto the face subspace you already estimated from the *Training Set*. Report the percentage of correctly classified face images for $d = 1, \ldots, 10$ and the percentage of missing entries $\{0, 10, 20, 30, 40\}\%$.
5. **Outlier detection.** Augment the images of individual 1 with those from an *Outlier Set*. Apply the RPCA algorithm for data corrupted by outliers that you implemented in Exercise 3.10 to these images to compute the mean face and the eigenfaces as well as detect the outliers. Note that RPCA does not compute the mean face, so you will need to modify your code accordingly. Plot the mean face and the top three eigenfaces and compare them to what you obtained with PCA. Report the percentage of correctly detected outliers.
6. **Face recognition with corrupted entries.** Apply the RPCA algorithm for data corrupted by outliers that you implemented in part (e) to the images in *Training Set* $\cup$ *Outlier Set*. Plot the projected training images $\boldsymbol{y} \in \mathbb{R}^d$ for $d = 2$ or $d = 3$ using different colors for the different classes. Do faces of different individuals naturally cluster in different regions of the low-dimensional space? Classify the faces in the *Test Set* using 1-nearest-neighbor. That is, label an image $\boldsymbol{x}$ as corresponding to individual i if its projected image $\boldsymbol{y}$ is closest to a projected image $\boldsymbol{y}_j$ of individual i. Report the percentage of correctly detected outliers and the percentage of correctly classified face images for $d = 1, \ldots, 10$ and compare your results to those using PCA in Chapter 2.

Exercise 3.12 Show that the optimal solution to the PCA problem with robustness to outliers

$$\min_{\boldsymbol{\mu}, U, Y} \sum_{j=1}^{N} w_j \|\boldsymbol{x}_j - \boldsymbol{\mu} - U\boldsymbol{y}_j\|^2 \quad \text{s.t.} \quad U^\top U = I_d \quad \text{and} \quad \sum_{j=1}^{N} w_j \boldsymbol{y}_j = \boldsymbol{0}, \tag{3.117}$$

where $w_j \in [0, 1]$ is large when point $\boldsymbol{x}_j$ is an inlier and small otherwise, is given by

$$\hat{\boldsymbol{\mu}}_N = \frac{\sum_{j=1}^N w_j \boldsymbol{x}_j}{\sum_{j=1}^N w_j} \quad \text{and} \quad \hat{\boldsymbol{y}}_j = \hat{U}^\top (\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N) \;\; \forall j \;\text{ s.t. }\; w_j > 0, \tag{3.118}$$

where $\hat{U}$ is a $D \times d$ matrix whose columns are the top d eigenvectors of

$$\hat{\Sigma}_N = \frac{\sum_{j=1}^N w_j (\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N)(\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}_N)^\top}{\sum_{j=1}^N w_j}. \tag{3.119}$$