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 2D trajectories of an object moving in a video may become incomplete when the object becomes occluded. Sometimes it is also the case that some entries of the data points are corrupted by gross errors and we do not know which entries are corrupted in advance. For instance, the intensities of some pixels of the face image of a person can be corrupted when the person is wearing a glasses. Sometimes it is also the case that all the entries of a subset of the data points are corrupted. For instance, if we are trying to detect 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 will be distinguished from the case of corrupted entries, some times also referred to as *intra-sample 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 introduce some effective techniques based on statistical estimation or convex relaxation methods that can still recover the low-dimensional subspace under such adversarial conditions. We first consider the PCA problem with missing data and describe methods for solving this problem based on alternating minimization, maximum likelihood and convex optimization. We then consider the PCA problem with corrupted entries and describe methods for solving this problem based on alternating minimization and convex optimization. We then consider the PCA problem with outliers and describe methods for solving this problem based on robust estimation and convex optimization.

3.1 PCA with Missing Entries

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

In this section, we consider the PCA problem in the case where 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 x_i-entry 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 i-th entry zeroed out and $\boldsymbol{e}_i = [0, \ldots, 0, 1, 0, \ldots, 0]^\top \in \mathbb{R}^D$ is the i-th 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:

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

In the PCA with missing data problem, the point $\boldsymbol{x}$ belongs to both L and S. Therefore, given $\boldsymbol{\mu}$ and U_d, 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

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

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., $\boldsymbol{e}_i \notin \text{span}(U_d)$.

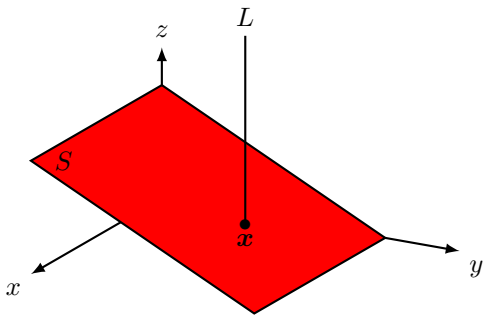

Figure 3.1. Given a point $\boldsymbol{x} \in \mathbb{R}^D$ with one unknown entry, x_i, the point $\boldsymbol{x}$ is known only up to a line L. However, if we also know that $\boldsymbol{x}$ belongs to a subspace S, we can find the unknown entry by intersecting L and S provided that L is not parallel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

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 $U_d = \begin{bmatrix} U_U \\ U_O \end{bmatrix}$ 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

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

A necessary condition for the linear system in (3.4) to have a unique solution is that the matrix in the left hand side be of full column rank $d+m \leq D$. This implies that $\boldsymbol{e}_i \notin \text{span}(U_d)$ 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. Finally, in the case where 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

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

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 **??**). Notice that this solution is simply the least square solution to (3.4) and that in order for U_O to be 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 $\boldsymbol{y}$.

In practice, however, we do not know $\boldsymbol{\mu}$ or U_d a priori. Instead, we are given only N incomplete samples, which we can arrange as the columns of an incom-

plete 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 Haddamart product of two matrices, which is defined as the entry-wise 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_d, 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 a solution to this problem. For instance, suppose the first entry is missing from everyone of the data points. 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. Moreover, suppose that the matrix X is given by

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

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 so we will not be able to distinguish X from the zero matrix.

In spite of the existence of obvious cases where the PCA problem with missing entries is not well-posed, we expect that if the matrix X is generic and the missing entries do not follow a specific pattern, we should be able to recover both $\boldsymbol{\mu}$, U_d and Y as long as the number of measurements (known entries of X) is sufficiently large relative to the number of unknowns ($D + dD + dN - d^2$ independent entries in $\boldsymbol{\mu}$, U_d and Y). Intuitively, the smaller the dimension of the subspace, d, the larger the amount of missing entries we should be able to deal with.

In what follows, we discuss a few approaches for solving the PCA problem with missing data. The first approach (described in Section 3.1.1) is a simple extension of geometric PCA (see Section 2.1) in which the sample mean and covariance are directly computed from the incomplete data. 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 algorithm (EM) described in Appendix B to complete the missing entries. While this approach is guaranteed to converge, the solution it finds is not always guaranteed to coincide with the global optimum. The third approach (described in Section 3.1.3) alternates between solving for $\boldsymbol{\mu}$, U_d and Y given a completion of X, and solving for the missing entries of X given $\boldsymbol{\mu}$, U_d and Y. In general, this approach is not guaranteed to converge to

the global optimum. However, we present a variant of this method which, under certain conditions, is guaranteed to recover the missing entries. The fourth and final approach (described in Section 3.1.4) uses convex optimization to find the missing entries of X. Under certain conditions on the subspace and the missing entries, this approach is guaranteed to return a perfect completion of the low-rank matrix.

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:

$$\min_{\boldsymbol{\mu}, U_d, \{\boldsymbol{y}_j\}} \sum_{j=1}^{N} \left\| \boldsymbol{x}_j - \boldsymbol{\mu} - U_d \boldsymbol{y}_j \right\|^2 \text{ s.t. } U_d^\top U_d = I_d \text{ and } \sum_{j=1}^{N} \boldsymbol{y}_j = \mathbf{0}. \quad (3.9)$$

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

$$\widehat{\boldsymbol{\mu}}_N = \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{x}_j \quad \text{and} \quad \widehat{\Sigma}_N = \frac{1}{N} \sum_{j=1}^{N} (\boldsymbol{x}_j - \widehat{\boldsymbol{\mu}}_N)(\boldsymbol{x}_j - \widehat{\boldsymbol{\mu}}_N)^\top, \quad (3.10)$$

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

When some entries of each $\boldsymbol{x}_j$ are missing, we cannot directly compute $\widehat{\boldsymbol{\mu}}_N$ nor $\widehat{\Sigma}_N$ as in (3.9). 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

$$\hat{\mu}_i = \frac{\sum_{j=1}^{N} w_{ij} x_{ij}}{\sum_{j=1}^{N} w_{ij}} \quad \text{and} \quad \hat{\sigma}_{ik} = \frac{\sum_{j=1}^{N} w_{ij} w_{kj} (x_{ij} - \hat{\mu}_i)(x_{kj} - \hat{\mu}_k)}{\sum_{j=1}^{N} w_{ij} w_{kj}}, \quad (3.11)$$

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 semi-definite. Second, these estimates are not obtained by optimizing any statistically or geometrically meaningful objective function (least squares, maximum likelihood, etc.) Nonetheless, estimates obtained from this naive approach 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_d as the eigenvectors of $\widehat{\Sigma}_N$ associated with its d largest eigenvalues. Then, given $\widehat{\boldsymbol{\mu}}_N$ and $\widehat{U}_d$, we can complete each missing entry as described in (3.6).

3.1.2 Incomplete PCA by Expectation Maximization

In this section, we derive an Expectation Maximization (EM) algorithm 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 $\Sigma_{\boldsymbol{x}} = U_d U_d^\top + \sigma^2 I_D$. Recall also from (2.56) that the log-likelihood of the PPCA model is given by

$$\mathcal{L} = -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log\det(\Sigma_{\boldsymbol{x}}) - \frac{1}{2}\sum_{j=1}^{N}\operatorname{trace}(\Sigma_{\boldsymbol{x}}^{-1}(\boldsymbol{x}_j - \boldsymbol{\mu})(\boldsymbol{x}_j - \boldsymbol{\mu})^\top). \tag{3.12}$$

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

$$\begin{bmatrix}\boldsymbol{x}_U \\ \boldsymbol{x}_O\end{bmatrix} = P\boldsymbol{x}, \quad \begin{bmatrix}\boldsymbol{\mu}_U \\ \boldsymbol{\mu}_O\end{bmatrix} = P\boldsymbol{\mu} \quad \text{and} \quad \begin{bmatrix}\Sigma_{UU} & \Sigma_{UO} \\ \Sigma_{OU} & \Sigma_{OO}\end{bmatrix} = P\Sigma_{\boldsymbol{x}}P^\top. \tag{3.13}$$

Here $\boldsymbol{x}_O$ is the observed part of $\boldsymbol{x}$, $\boldsymbol{x}_U$ is the unobserved part of $\boldsymbol{x}$, and P is a permutation matrix. Notice that conditional distribution of $\boldsymbol{x}_U$ given $\boldsymbol{x}_O$ is also Gaussian. More specifically, $\boldsymbol{x}_U \mid \boldsymbol{x}_O \sim \mathcal{N}(\boldsymbol{\mu}_{U|O}, \Sigma_{U|O})$, where

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

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 needed, we will use $\boldsymbol{x}_{jU}$ and $\boldsymbol{x}_{jO}$ to denote the unobserved and observed parts of point $\boldsymbol{x}_j$, respectively, and and P_j to denote the permutation matrix.

In what follows, we derive two variants of the EM algorithm for learning the parameters $\theta = (\boldsymbol{\mu}, U_d, \sigma)$ of the PPCA model from incomplete samples $\{\boldsymbol{x}_j\}_{j=1}^N$. The first variant (MAP-EM) is an approximate EM method where the unobserved variables are imputed by their MAP estimates. The second variant is the exact EM algorithm, where we take the conditional expectation of $\mathcal{L}$ 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.

Maximum a Posteriori Expectation Maximization (MAP-EM)

The MAP-EM algorithm (see Appendix B.2) is a simplified version of the EM algorithm that alternates between the following two steps:

- E: Complete each data point $\boldsymbol{x}$ by imputing 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: Maximize the complete log-likelihood, with $\boldsymbol{x}_U$ is imputed as in the E-step.

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

$$\arg\max_{\boldsymbol{x}_U} p_{\theta^k}(\boldsymbol{x}_U \mid \boldsymbol{x}_O) = \boldsymbol{\mu}_{U|O}^k = \boldsymbol{\mu}_U^k + \Sigma_{UO}^k (\Sigma_{OO}^k)^{-1}(\boldsymbol{x}_O - \boldsymbol{\mu}_O^k). \quad (3.14)$$

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

$$\mathcal{L} = -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log\det(\Sigma_{\boldsymbol{x}}) - \frac{1}{2}\sum_{j=1}^{N}(\boldsymbol{x}_j^k - \boldsymbol{\mu}_{\boldsymbol{x}})^\top \Sigma_{\boldsymbol{x}}^{-1}(\boldsymbol{x}_j^k - \boldsymbol{\mu}_{\boldsymbol{x}}). \quad (3.15)$$

During the M-step, we need to maximize $\mathcal{L}$ w.r.t. θ. Since the data are already complete, we can update the model parameters as described in Theorem 2.8, i.e.,

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

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

$$\widehat{\Sigma}_N^{k+1} = \frac{1}{N}\sum_{j=1}^{N}(\boldsymbol{x}_j^k - \boldsymbol{\mu}^{k+1})(\boldsymbol{x}_j^k - \boldsymbol{\mu}^{k+1})^\top, \quad (3.16)$$

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

Expectation Maximization (EM)

The EM algorithm (see Appendix B.2) alternates between the following steps

E: Compute the expectation of $\mathcal{L}$ given $\boldsymbol{x}_O$ and θ^k, $\mathbb{E}[\mathcal{L} \mid \boldsymbol{x}_O, \theta^k]$.

M: Maximize the expected completed log-likelihood $\mathbb{E}[\mathcal{L} \mid \boldsymbol{x}_O, \theta^k]$.

Observe from (3.15) that to compute the expectation of $\mathcal{L}$ it suffices to compute the following matrix for each incomplete data point $\boldsymbol{x}$:

$$S^k = \mathbb{E}[(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}})(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}})^\top \mid \boldsymbol{x}_O, \theta^k] = P^\top \begin{bmatrix} S_{UU}^k & S_{UO}^k \\ S_{OU}^k & S_{OO}^k \end{bmatrix} P. \quad (3.17)$$

Each block of this matrix can be computed as

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

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

$$Q(\theta \mid \theta^k) = -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log\det(\Sigma_{\boldsymbol{x}}) - \frac{N}{2}\operatorname{trace}(\Sigma_{\boldsymbol{x}}^{-1}\widehat{\Sigma}^k_N). \quad (3.18)$$

In the M-step, we need to maximize this quantity w.r.t. θ. Notice that this quantity is almost identical to that in (2.56), except that the sample covariance matrix $\widehat{\Sigma}_N$ is replaced by $\widehat{\Sigma}^k_N$. Thus, if $\widehat{\Sigma}^k_N$ didn't depend on the unknown parameter $\boldsymbol{\mu}$, we could immediately compute U_d and σ from Theorem 2.8. Therefore, all we need to do is to show how to compute $\boldsymbol{\mu}$. To that end, notice that

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\mu}}\operatorname{trace}(\Sigma_{\boldsymbol{x}}^{-1}S^k) &= \frac{\partial}{\partial \boldsymbol{\mu}}\mathbb{E}[(\boldsymbol{x}-\boldsymbol{\mu})^\top\Sigma_{\boldsymbol{x}}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \mid \boldsymbol{x}_O, \theta^k] && (3.19)\\
&= -\Sigma_{\boldsymbol{x}}^{-1}\mathbb{E}[\boldsymbol{x}-\boldsymbol{\mu} \mid \boldsymbol{x}_O, \theta^k] = -\Sigma_{\boldsymbol{x}}^{-1}(\boldsymbol{x}^k - \boldsymbol{\mu}), && (3.20)
\end{aligned}$$

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

$$\frac{\partial}{\partial \boldsymbol{\mu}}Q(\theta \mid \theta^k) = -\frac{1}{2}\frac{\partial}{\partial \boldsymbol{\mu}}\sum_{j=1}^N \operatorname{trace}(\Sigma_{\boldsymbol{x}}^{-1}S^k_j) = \sum_{j=1}^N \Sigma_{\boldsymbol{x}}^{-1}(\boldsymbol{x}^k_j - \boldsymbol{\mu}) = \mathbf{0}, \quad (3.21)$$

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

$$\boldsymbol{\mu}^{k+1} = \frac{1}{N}\sum_{j=1}^N \boldsymbol{x}^k_j. \quad (3.22)$$

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 $\widehat{\Sigma}^k_N$ and compute U^{k+1}_d and σ^{k+1} as before. Notice, however, that $\widehat{\Sigma}^k_N$ is not the covariance of the complete data. The key difference is in the term S^k_{UU}, which contains an additional term $\Sigma^k_{U|O}$. The EM algorithm for PPCA with missing data is summarized in Algorithm 3.1.

Algorithm 3.1 (Expectation Maximization for PPCA with Incomplete Data)

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

initialize

$x_{ij} = 0$ for $(i,j) \notin \Omega$, $\boldsymbol{\mu} = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{x}_j$, and $\Sigma = \frac{1}{N}\sum_{j=1}^{N}(\boldsymbol{x}_j - \boldsymbol{\mu})(\boldsymbol{x}_j - \boldsymbol{\mu})^\top$

repeat

$\boldsymbol{x}_j \leftarrow P_j^\top \begin{bmatrix} \boldsymbol{\mu}_U + \Sigma_{UO}\Sigma_{OO}^{-1}(\boldsymbol{x}_O - \boldsymbol{\mu}_O) \\ \boldsymbol{x}_O \end{bmatrix}$

$\boldsymbol{\mu} \leftarrow \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{x}_j$

$S \leftarrow \frac{1}{N}\sum_{j=1}^{N}(\boldsymbol{x}_j - \boldsymbol{\mu})(\boldsymbol{x}_j - \boldsymbol{\mu})^\top + P_j^\top \begin{bmatrix} \Sigma_{UU} - \Sigma_{UO}\Sigma_{OO}^{-1}\Sigma_{OU} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} P_j$

$U_1 \leftarrow$ top d eigenvectors of S

$\Lambda_1 \leftarrow$ top d eigenvalues of S

$\sigma^2 \leftarrow \frac{1}{D-d}\sum_{i=d+1}^{D} \lambda_i(S)$

$U_d \leftarrow U_1(\Lambda_1 - \sigma^2 I)^{1/2}R$, where $R \in \mathbb{R}^{d\times d}$ is an arbitrary orthogonal matrix

$\Sigma \leftarrow U_d U_d^\top + \sigma^2 I = \mathcal{D}_{\sigma^2}(S) + \sigma^2 I$

until convergence of $\boldsymbol{\mu}$ and S

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

3.1.3 Incomplete PCA by Matrix Factorization

In this section, we describe an alternating minimization algorithm for solving the geometric PCA problem with missing data. The main idea behind this approach, which was proposed in [Wiberg, 1976], is to find $\boldsymbol{\mu}$, U_d and Y that minimize the error $\|X - \boldsymbol{\mu}\mathbf{1}^\top - U_d Y\|_F^2$ considering only the known entries of X, i.e.,

$$\|W \odot (X - \boldsymbol{\mu}\mathbf{1}^\top - U_d Y)\|_F^2 = \sum_{i=1}^{D}\sum_{j=1}^{N} w_{ij}(x_{ij} - \mu_i - \boldsymbol{u}_i^\top \boldsymbol{y}_j)^2, \qquad (3.23)$$

where x_{ij} is the ij-th entry of X, μ_i is the i-th entry of $\boldsymbol{\mu}$, $\boldsymbol{u}_i^\top$ is the i-th row of U_d and $\boldsymbol{y}_j$ is the j-th column of Y. Notice that this cost function is the same as that in (3.9), 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.

In what follows, we will derive an alternating minimization algorithm for minimizing the cost function in (3.23). 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 (3.23) 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 soon

see. In fact, as we will show in a later section, the algorithm can also be extended to the more challenging PCA problem with missing entries.

Power Factorization for Complete Matrix Factorization

In the case of complete, zero-mean data, the optimization problem in (3.23) reduces to the low-rank matrix approximation problem based on explicit factorization $\min_{U_d,Y} \|X - U_d Y\|_F^2$. 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 3.1) that the sequence of vectors

$$\boldsymbol{u}^{k+1} = \frac{A\boldsymbol{u}^k}{\|A\boldsymbol{u}^k\|} \tag{3.24}$$

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 spanned by the top d eigenvectors, $\{\boldsymbol{u}_i\}_{i=1}^d$. One can show (see Exercise 3.2) that the sequence of matrices

$$U^{k+1} = AU^k (R^k)^{-1}, \tag{3.25}$$

where $Q^k R^k = AU^k$ is the QR decomposition of AU^k, converges to a matrix U_d 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 singular vectors of a (possibly) non-square matrix X. The main idea behind PF is that, given Y, an optimal solution for U_d is given by $XY^\top (YY^\top)^{-1}$. As before, such a matrix can be made orthogonal by replacing U_d by the Q factor of the QR decomposition of $XY^\top (YY^\top)^{-1}$. Then, given an orthogonal U_d, the optimal Y is $U_d^\top X$. The PF algorithm (see Algorithm 3.2) then iterates between these two steps till convergence. The method is guaranteed to converge to the rank d approximation of X as stated in the following theorem (see Exercise 3.3 for the proof).

Theorem 3.1. *Let $X_d = U_d \Sigma_d V_d^\top$ be the rank-d approximation of X obtained from the SVD of X. Let σ_i be the i-th singular value of X. If $\sigma_d > \sigma_{d+1}$, then there exists a constant $c > 0$ such that for all $k \geq 0$*

$$\|X_d - U_d^k Y^k\|_F^2 \leq c \left(\frac{\sigma_{d+1}}{\sigma_d}\right)^{2k}. \tag{3.26}$$

Algorithm 3.2 (Power Factorization for Complete Matrix Factorization)

Input: Matrices X and Y^0

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

Output: Matrices U_d and Y

Power Factorization for Incomplete Matrix Factorization

Let us now consider the matrix factorization problem with incomplete, zero-mean data, i.e., the problem in (3.23) with $\boldsymbol{\mu} = \mathbf{0}$. Taking the derivatives of the cost function in (3.23) with respect to $\boldsymbol{u}_i$ and $\boldsymbol{y}_j$ and setting them to zero leads to

$$\Big(\sum_{j=1}^{N} w_{ij} \boldsymbol{y}_j \boldsymbol{y}_j^\top\Big) \boldsymbol{u}_i = \sum_{j=1}^{N} w_{ij} x_{ij} \boldsymbol{y}_j, \qquad i = 1, \ldots, D, \tag{3.27}$$

$$\Big(\sum_{i=1}^{D} w_{ij} \boldsymbol{u}_i \boldsymbol{u}_i^\top\Big) \boldsymbol{y}_j = \sum_{i=1}^{D} w_{ij} x_{ij} \boldsymbol{u}_i, \qquad j = 1, \ldots, N. \tag{3.28}$$

Therefore, given Y, the optimal U_d can be computed linearly from (3.27). As before, the constraint $U_d^\top U_d = I$ can be enforced by replacing U_d by the Q factor of the QR decomposition of $U_d = QR$. Then, given U_d, the optimal Y can be computed linearly from (3.28). This leads to the PF algorithm for matrix factorization with missing entries summarized in Algorithm 3.3.

Notice that when the matrix is complete, according to Theorem 3.1, the alternating matrix factorization method is guaranteed to converge exponentially as long as $\sigma_d > \sigma_{d+1}$. In the case of incomplete data, the above alternating procedure appears to be the most natural extension to the complete case. It is arguably the simplest and most popular low-rank matrix completion algorithm and works fairly well for many practical problems. However, since the objective function is non-convex, to the best of our knowledge, there are no results in the literature that guarantee the convergence of the algorithm to the globally optimal solution.

Nevertheless, recent progress in high-dimensional statistical analysis has started to indicate that when the matrix is high-dimensional, under certain benign conditions one should expect the above process to converge exponentially to the optimal solution with high-probability [Jain et al., 2012]. Although a detailed explanation of such results is far beyond the scope of this book, we here provide a brief introduction to the results 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, the proposed

Algorithm 3.3 (Power Factorization for Incomplete Matrix Factorization)

Input: Matrices $W \odot X$ and Y^0

1: Initialize $Y \leftarrow Y^0$
2: **repeat**
3: Given $Y = \big[\boldsymbol{y}_1, \cdots, \boldsymbol{y}_d\big]$, solve $\min_{U_d} \|W \odot (X - U_d Y)\|_F^2$ as

$$U_d = \begin{bmatrix} u_1^\top \\ \vdots \\ u_D^\top \end{bmatrix}, \quad u_i \leftarrow \Big(\sum_{j=1}^{N} w_{ij} \boldsymbol{y}_j \boldsymbol{y}_j^\top \Big)^{-1} \sum_{j=1}^{N} w_{ij} x_{ij} \boldsymbol{y}_j, \quad i = 1, \ldots, D.$$

4: Normalize $U_d \leftarrow U_d R^{-1}$, where $QR = U_d$
5: Given $U_d = \begin{bmatrix} u_1^\top \\ \vdots \\ u_D^\top \end{bmatrix}$, solve $\min_{Y} \|W \odot (X - U_d Y)\|_F^2$ as

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

6: **until** convergence of the sequence $U_d Y$

Output: U_d and Y

algorithm introduces some modification to the above straightforward extensions, which may inspire the readers to develop better algorithms in the future.

Consider a rank-d matrix X that we wish to complete. Assume that the locations of the given entries of X are drawn uniformly at random. Intuitively, one should expect that it would be very difficult or even impossible to recover a matrix X from such a subset of observations if the matrix X is itself very sparse, i.e., if it has a very small portion of nonzero entries. In the extreme, if X only has one nonzero entry (hence it is a rank-1 matrix), with high probability, any subset of observations of X will all be zeros. Hence, there is little chance that one can succeed in recovering the true X as all observations indicate that X is a matrix full of zeros. To avoid such ambiguity, we must restrict our low-rank matrices to those that are not so particularly sparse.

Definition 3.2 (Incoherent Matrix)**.** *A matrix $X \in \mathbb{R}^{D \times N}$ is incoherent with a constant $\mu > 0$ if*

$$\max_i \|\boldsymbol{u}_i\|_2 \leq \frac{\mu\sqrt{d}}{\sqrt{D}}, \quad \max_j \|\boldsymbol{v}_j\|_2 \leq \frac{\mu\sqrt{d}}{\sqrt{N}}, \quad \|UV^\top\|_\infty \leq \frac{\mu\sqrt{d}}{\sqrt{DN}} \tag{3.29}$$

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

This is a popular and convenient technical condition that people typically use to impose that a given matrix is not so sparse or its singular vectors are not so spiky. This condition is not so strict either – when the matrix is large, it holds with high probability for randomly generated (say Gaussian) matrices.

As before, we are interested in finding a rank-d factorization $UV^\top$ that best approximates the matrix X given the observed entries:

$$\min_{U,V} \|W \odot (X - UV^\top)\|_F^2. \tag{3.30}$$

In the alternating minimization for Algorithm 3.3 , in each iteration, we use all the observed entries to update the factors. As it turns out that at least technically at this point, it is easier to prove convergence and global optimality if one incorporates information from observations in an incremental fashion. Let us randomly and equally partition the set of observed entries W into $2K+1$ non-overlapping subsets, denoted as $W_0, W_1, \ldots, W_{2K}$. Let $p = \|W\|_0/DN$ be the probability that an entry is given. Let U be the top d singular vectors of $\frac{1}{p}W_0 \odot X$. We clip entries of U that have magnitude higher than $\frac{2\mu\sqrt{d}}{\sqrt{N}}$ to be zero and let the initial U^0 be the orthonormalized version of such U. We then can use the alternating minimization procedure described in Algorithm 3.4 to obtain the rank-d factors U and V for X from the incomplete observations.

Algorithm 3.4 (Alternating Minimization for Matrix Completion)

Input: Observed matrix $W \odot X$ and observation partition matrices $W_1, \ldots, W_{2K}$.

1: Initialize $\hat{U}^0 \leftarrow U^0$
2: **For** $k = 0, 1, \ldots, 2K$ **do**
3: $\hat{V}^{k+1} \leftarrow \arg\min_V ||W_{k+1} \odot (\hat{U}^k V^T - X)\|_F^2$.
4: $\hat{U}^{k+1} \leftarrow \arg\min_U ||W_{K+k+1} \odot (U(\hat{V}^{k+1})^T - X)\|_F^2$.
5: **End do**

Output: Return matrix $X = \hat{U}^K(\hat{V}^K)^T$.

Notice that there are two major differences between Algorithm 3.4 and Algorithm 3.3: the initialization to left singular vectors U and the update in each iteration uses only an independent subset of the observations. It is surprising that these small modifications to the basic power factorization method can ensure the convergence of the new procedure to the globally optimal solution as described by the following theorem. It is beyond the scope of this book to provide a complete proof and explanation to this theorem. We refer interested readers to [Jain et al., 2012].

Theorem 3.3. *Let $X = U\Sigma V^\top \in \mathbb{R}^{D\times N}$, $(N \geq D)$, be a rank-d matrix that is incoherent. Let each entry of X be observed independently and uniformly at random. If there exists a constant $c > 0$ such that the number of observed entries*

satisfies

$$\|W\|_0 \geq c\big(\frac{\sigma_1}{\sigma_d}\big)^4 d^{4.5} N \log N \log(d\|X\|_F/\varepsilon), \tag{3.31}$$

then with high probability, for $K = C' \log(\|X\|_F/\varepsilon)$ with some constant $C' > 0$, the outputs of Algorithm 3.4 satisfy $\|X - \hat{U}^K(\hat{V}^K)^T\|_F \leq \varepsilon$.

In words, the alternating minimization procedure guarantees to recover X up to the precision ε in $O(\log(1/\varepsilon))$ steps given that the number of observations are in the order of $O(d^{4.5} N \log N \log d)$. Although the theorem does not directly apply to the basic alternating minimization procedure given in Algorithm 3.3, it suggests that the procedure should work under similar conditions. Since it is very simple and easier to implement in practice, we would prefer Algorithm 3.3 to Algorithm 3.4 in real use.

Power Factorization for PCA with Incomplete Data

Let us now consider the PCA problem in the case of incomplete data, i.e., the problem in (3.23) where we want to recover both the mean $\boldsymbol{\mu}$ and the subspace basis U_d. As in the case of complete data, the solution to this problem need not be unique, because if $(\boldsymbol{\mu}, U_d, Y)$ is an optimal solution, then so is $(\boldsymbol{\mu} - U_d\boldsymbol{b}, U_d A, A^{-1}Y)$ for all $\boldsymbol{b} \in \mathbb{R}^d$ and $A \in \mathbb{R}^{d\times d}$. Therefore, we will forgo the constraints $U_d^\top U_d = I$ and $Y\mathbf{1} = \mathbf{0}$ 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 (3.23) with respect to μ_i, $\boldsymbol{u}_i$ and $\boldsymbol{y}_j$ and set them to zero. This leads to

$$\Big(\sum_{j=1}^N w_{ij}\Big)\mu_i = \sum_{j=1}^N w_{ij}(x_{ij} - \boldsymbol{u}_i^\top \boldsymbol{y}_j), \qquad i = 1, \ldots, D, \tag{3.32}$$

$$\Big(\sum_{j=1}^N w_{ij}\boldsymbol{y}_j\boldsymbol{y}_j^\top\Big)u_i = \sum_{j=1}^N w_{ij}(x_{ij} - \mu_i)\boldsymbol{y}_j, \qquad i = 1, \ldots, D, \tag{3.33}$$

$$\Big(\sum_{i=1}^D w_{ij}\boldsymbol{u}_i\boldsymbol{u}_i^\top\Big)\boldsymbol{y}_j = \sum_{i=1}^D w_{ij}(x_{ij} - \mu_i)\boldsymbol{u}_i, \qquad j = 1, \ldots, N. \tag{3.34}$$

Therefore, given U_d and Y, the optimal $\boldsymbol{\mu}$ and can be computed from (3.32). Likewise, given $\boldsymbol{\mu}$ and Y, the optimal U_d can be computed linearly from (3.33). Also, given $\boldsymbol{\mu}$ and U_d, the optimal Y can be computed linearly from (3.34).

As before, we can enforce the constraint $U_d^\top U_d = I$ by replacing U_d by the Q factor of the compact QR decomposition of $U_d = QR$. Also, we can enforce the constraint $Y\mathbf{1} = \mathbf{0}$ by replacing $\boldsymbol{\mu}$ by $\boldsymbol{\mu} - U_d Y\mathbf{1}$, and Y by $(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top)Y$. 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 this approach, the steps in (3.32) and (3.33) are combined into a single

Algorithm 3.5 (Power Factorization for PCA with Incomplete Data)

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

1: **initialize** $U_d = \begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_D^\top \end{bmatrix} \leftarrow U_d^0$ and $Y = \begin{bmatrix} \boldsymbol{y}_1, \cdots, \boldsymbol{y}_d \end{bmatrix} \leftarrow Y^0$.

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

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

6: $\quad Y = \begin{bmatrix} \boldsymbol{y}_1, \cdots, \boldsymbol{y}_d \end{bmatrix}$ 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 $\boldsymbol{\mu}\mathbf{1}^\top + U_d Y$

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

step

$$\sum_{j=1}^N w_{ij} \begin{bmatrix} \boldsymbol{y}_j \\ 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_j \\ 1 \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{u}_i \\ \mu_i \end{bmatrix} = \sum_{j=1}^N w_{ij} x_{ij} \begin{bmatrix} \boldsymbol{y}_j \\ 1 \end{bmatrix}, \quad j = 1, \ldots, N. \tag{3.35}$$

This leads to an alternating minimization scheme where, given Y one solves for $\boldsymbol{\mu}$ and U_d from (3.35), and given $\boldsymbol{\mu}$ and U_d, one solves for Y from (3.34).

As before, neither the three-term alternating minimization or the two-term alternating minimization is guaranteed to converge to the global optimal solution of the PCA problem. But it should not be so difficult to adjust the procedures based on the general alternating minimization Algorithm 3.4 to ensure global optimality under certain conditions. We leave such technical developments to the readers as exercises. Nevertheless, the above simple procedures remain as viable choices for practitioners to solve their problems in practice.

3.1.4 Incomplete PCA by Convex Optimization

The previous two approaches, namely expectation maximization and matrix factorization with missing data, are based on (a) explicit parameterizations of the low-rank factors and (b) minimization of a non-convex cost function in an alternating minimization fashion. These approaches suffer from two important disadvantages. First, the desired rank of the matrix needs to be known in advance. Second, it is difficult to ensure convergence and global optimality.

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 global minimizer. Moreover, under rather benign conditions on the missing entries, this approach is guaranteed to perfectly recover the desired low-rank matrix, even without knowing the rank of the matrix in advance.

Similar to the convergence result for the matrix factorization algorithm, a rigorous justification of the optimality of the convex relaxation approach requires a deep knowledge in 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 results and basic algorithm offered by this approach so that practitioners can apply them to their data and problems. The interested reader 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 Sparse Vector or Low-rank Matrix

Let us first motivate this approach with a simpler but related problem of finding a solution to an underdetermined system of linear equations $A\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{x} \in \mathbb{R}^N$, $\boldsymbol{b} \in \mathbb{R}^D$ and $A \in \mathbb{R}^{D\times N}$, with $D < N$. Since this linear system is underdetermined, in general there could be many solutions $\boldsymbol{x}$. 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). In other words, we want to recover an unknown vector or matrix from highly compressive or incomplete observations. This class of problems is known in the literature as *compressed or compressive sensing* [Candès, 2006].

To find a unique solution, we often need to specify what type of solutions we prefer among all possible ones. This requires us to *regularize* the solution $\boldsymbol{x}$ based some measure of *goodness*. A classical approach to finding a unique solution (when a solution exists) is to look for a vector $\boldsymbol{x}$ of minimum ℓ_2 norm, i.e.,

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2 \quad \text{s.t.} \quad A\boldsymbol{x} = \boldsymbol{b}. \tag{3.36}$$

In physics, least ℓ_2 norm solutions often corresponds to minimum energy.

An alternative approach is to look for a solution $\boldsymbol{x}$ that is the sparsest, whose physical meaning, unlike the ℓ_2 norm solution, would now corresponds to minimum entropy. Specifically, assume that the vector $\boldsymbol{b}$ is generated as $A\boldsymbol{\mu} = \boldsymbol{b}$, where $\boldsymbol{\mu}$ is a d-sparse vector, i.e., $\|\boldsymbol{\mu}\|_0 = d \ll N$. When the matrix A is such that $\delta_{2d}(A) < 1$, where $\delta_d(A)$ is the smallest number such that for all $\boldsymbol{x}$ with $\|\boldsymbol{x}\|_0 \leq d$, we have

$$(1 - \delta_d(A))\|\boldsymbol{x}\|_2^2 \leq \|A\boldsymbol{x}\|_2^2 \leq (1 + \delta_d(A))\|\boldsymbol{x}\|_2^2, \tag{3.37}$$

then $\boldsymbol{\mu}$ is the only d-sparse vector such that $A\boldsymbol{x} = \boldsymbol{b}$. Such a matrix A satisfying $\delta_k(A) < \tau$ for some k and τ is said to have the *restricted isometry property* (RIP).

In order to find $\boldsymbol{\mu}$, we seek a solution to the problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_0 \quad \text{s.t.} \quad A\boldsymbol{x} = \boldsymbol{b}. \tag{3.38}$$

In general, this problem is NP hard [Amaldi and Kann, 1998]. However, as recently shown in the compressive sensing literature (see [Candès and Tao, 2005, Candes, 2008] and others), under fairly broad conditions, say when the matrix A is large and satisfies the RIP $\delta_{2d}(A) < \sqrt{2} - 1$, then the optimal solution to (3.38) can be correctly found with high-probability by solving the following convex optimization problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \quad \text{s.t.} \quad A\boldsymbol{x} = \boldsymbol{b}. \tag{3.39}$$

Inspired by the success of compressive sensing of high-dimensional sparse vectors, one could naturally expect that a similar convex relaxation also applies to the recovery of a high-dimensional low-rank matrix M from compressive linear measurements $b = \mathcal{A}(M)$. If the linear operator $\mathcal{A}$ satisfies certain conditions (say a generalized RIP condition), then the solution to the rank minimization problem (which is also NP-hard in general)

$$\min_{X} \operatorname{rank}(X) \quad \text{s.t.} \quad \mathcal{A}(X) = B, \tag{3.40}$$

could be correctly found with high probability by solving the following convex optimization problem instead

$$\min_{X} \|X\|_* \quad \text{s.t.} \quad \mathcal{A}(X) = B, \tag{3.41}$$

where $\|X\|_*$ is the nuclear norm of the matrix X (i.e., the sum of all singular values of X). Below we summarize the exact conditions and results related to this approach, and give a simple algorithm to solve the above optimization problem.

Exact Matrix Completion with Minimum Number of Measurements

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 dimensions $d \ll D$. Assume that we observe only a subset of the entries of X indexed by a set Ω, i.e.,

$$\Omega = \{(i,j) : x_{ij} \text{ is observed}\}. \tag{3.42}$$

Let $\mathcal{P}_\Omega : \mathbb{R}^{D\times N} \to \mathbb{R}^{D\times N}$ be the orthogonal projector onto the span of 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:

$$\min_{A} \quad \operatorname{rank}(A) \quad \text{s.t.} \quad \mathcal{P}_\Omega(A) = \mathcal{P}_\Omega(X). \tag{3.43}$$

For simplicity, let us assume $D = N$ for now. An $N \times N$ matrix X of rank d has $2Nd - d^2$ degrees of freedom. Therefore, one should not expect to complete or

recover a rank-d matrix uniquely with less than $2Nd - d^2$ entries as, 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 have a unique solution and the solution to be found efficiently.

Since the above rank-minimization problem (even if the solution exists and is unique) is NP hard, inspired the compressive sensing story, we consider the following convex relaxation

$$\min_{A} \quad \|A\|_* \quad \text{s.t.} \quad \mathcal{P}_\Omega(A) = \mathcal{P}_\Omega(X), \tag{3.44}$$

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

It has been established in a series of seminal work [Candès and Recht, 2009, Candès and Tao, 2010, Gross, 2011] that when the matrix is incoherent, the locations of the known entries are sampled uniformly at random, and the number of known entries is sufficiently large, the minimizer to the problem (3.44) is unique and equal to the matrix X for most matrices X. More specifically, they together have developed very tight bounds for the minimum number of measurements that are needed in order for the convex optimization to give the correct solution with high probability. The following theorem summarizes their results.

Theorem 3.4. *Let X be a $D \times N$ matrix of rank d and let $K = \max(D, N)$. Assume X is incoherent with parameter μ according to Definition 3.2. Suppose we observe M entries of X with locations sampled uniformly at random. Then there is a numerical constant c (depending on μ) such that if*

$$M \geq cdK(\log K)^2, \tag{3.45}$$

the minimizer to the problem (3.44) is unique and equal to X with probability at least $1 - K^{-3}$; that is to say, the program (3.44) 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 $O(dK)$. In essence, the theorem states that with only a polylog factor of extra measurements, i.e., $O(dK\text{polylog}(K))$, 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 of constant) the minimum number of entries needed to guarantee the exact completion of X reduces to $M \geq K \log(K)$ [Keshavan et al., 2010a]. It is worth mentioning that the above statement is not only 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 $\mathcal{A}(X)$, i.e., for the problem (3.41), as long as the linear operator $\mathcal{A}(\cdot)$ is incoherent with the matrix X.

Efficient Algorithm via Proximal Gradient

In order to compute the solution to the problem in (3.44), we can use the method of augmented Lagrange multipliers (ALM) described in Appendix A. According

to this method, the Lagrangian of (3.44), $\|A\|_* + \langle Z, P_\Omega(X) - P_\Omega(A)\rangle$, where $Z \in \mathbb{R}^{D\times N}$ is a matrix of Lagrange multipliers, is augmented by the sum of the squares of the violation of the constraints, i.e.,

$$\mathcal{L}(A, Z) = \|A\|_* + \langle Z, P_\Omega(X) - P_\Omega(A)\rangle + \frac{\beta}{2}\|P_\Omega(X) - P_\Omega(A)\|_F^2, \quad (3.46)$$

where $\beta > 0$ is a parameter. Then, the optimal solution to the problem $\max_Z \min_A \mathcal{L}(A, Z)$ is found by iterating the following two steps

$$\begin{cases} A_k &= \arg\min_A \mathcal{L}(A, Z_{k-1}), \\ Z_k &= Z_{k-1} + \beta \frac{\partial \mathcal{L}}{\partial Z}(A_k, Z_{k-1}). \end{cases} \quad (3.47)$$

To compute the optimal A, notice that the sub-gradient of $\mathcal{L}$ w.r.t. A is given by $\partial\|A\|_* - P_\Omega(Z) - \beta(P_\Omega(X) - P_\Omega(A))$, where $\partial\|A\|_*$ is the sub-gradient of the nuclear norm of A (see Exercise 3.7). Thus, the optimal solution for A given Z is $A = \mathcal{D}_{\frac{1}{\beta}}(P_\Omega(X) + \frac{1}{\beta}P_\Omega(Z))$ (see Exercise 3.8), where $\mathcal{D}_\varepsilon$ is the singular value thresholding operator defined in (2.90). This is also known as the *proximal gradient descent* method. Even though the ALM objective function (3.46) is not smooth, this method is known to converge as fast as the regular gradient descent method for smooth functions, with a rate of $O(1/k)$. Therefore, starting from $Z_0 = 0$, the augmented Lagrangian multiplier objective $\max_Z \min_A \mathcal{L}(A, Z)$ can be optimized via the Algorithm 3.6 summarized below.

Algorithm 3.6 (Low-rank Matrix Completion by ALM via Proximal Gradient)

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

1: Initialize $Z_0 \leftarrow \mathbf{0}$.
2: **repeat**
3: $\quad A_k \leftarrow \mathcal{D}_{\frac{1}{\beta}}(P_\Omega(X) + \frac{1}{\beta}Z_{k-1})$.
4: $\quad Z_k \leftarrow Z_{k-1} + \beta(P_\Omega(X) - P_\Omega(A_k))$.
5: **until** convergence of Z.

Output: Matrix A.

3.2 PCA with Corrupted Data

Recall from Section 2.1.2 that in the PCA problem we are given N data points $\{\boldsymbol{x}_j \in \mathbb{R}^D\}_{j=1}^N$ drawn (approximately) from a d-dimensional affine subspace $S = \{\boldsymbol{x} = \boldsymbol{\mu} + U_d\boldsymbol{y}\}$, where $\boldsymbol{\mu} \in \mathbb{R}^D$ is an arbitrary point in S, $U_d \in \mathbb{R}^{D\times d}$ is a basis for S, and $\{\boldsymbol{y}_j \in \mathbb{R}^d\}_{j=1}^N$ are the principal components. In this section, we consider the PCA problem in the case where some of the entries of the data points have been corrupted by gross errors. That is, the i-th entry x_{ij} of a data point $\boldsymbol{x}_j$

is obtained by corrupting the i-th entry x_{ij}^0 of a point $\boldsymbol{x}_j^0$ lying perfectly on the subspace S by an error e_{ij}, i.e.,

$$x_{ij} = x_{ij}^0 + e_{ij}, \quad \text{or} \quad \boldsymbol{x}_j = \boldsymbol{x}_j^0 + \boldsymbol{e}_j, \quad \text{or} \quad X = X^0 + E, \tag{3.48}$$

where $X, X^0, E \in \mathbb{R}^{D \times N}$ are matrices with entries x_{ij}, x_{ij}^0 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,

$$\Omega = \{(i,j) : e_{ij} \neq 0\}, \tag{3.49}$$

as well as correct the erroneous entries before applying PCA to the given data. Indeed, notice that a key difference with the incomplete PCA problem, where we know the location of the missing entries, 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 $|\Omega| \ll DN$, we expect to be able detect and correct such gross 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 Reweighed Least Squares

One of the simplest algorithms for dealing with corrupted entries is the reweighted least squares 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

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{x}_j \quad \text{and} \quad \hat{\boldsymbol{y}}_j = \hat{U}_d^\top (\boldsymbol{x}_j - \boldsymbol{\mu}), \tag{3.50}$$

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

$$\widehat{\Sigma}_N = \frac{1}{N}\sum_{j=1}^{N} (\boldsymbol{x}_j - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_j - \hat{\boldsymbol{\mu}})^\top. \tag{3.51}$$

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 i-th entry of $\boldsymbol{x}_j$ such that $w_{ij} \approx 1$ if x_{ij} is not corrupted and $w_{ij} \approx 0$ otherwise. Then, similarly to (2.23), 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 $\boldsymbol{\mu} + U_d \boldsymbol{y}_j$ onto the subspace S, i.e.,

$$\sum_{i=1}^{D} \sum_{j=1}^{D} w_{ij} (x_{ij} - \mu_i - \boldsymbol{u}_i^\top \boldsymbol{y}_j)^2, \tag{3.52}$$

where μ_i is the i-th entry of $\boldsymbol{\mu}$, $\boldsymbol{u}_i^\top$ is the i-th row of U_d, and y_{ij} is the i-th entry of the vector of coordinates $\boldsymbol{y}_j$ of the point $\boldsymbol{x}_j$ in the subspace S.

Notice that the above objective function is identical to the objective function in (3.23), 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 not, while here $w_{ij} \in [0, 1]$ denotes whether x_{ij} is corrupted or not. Other than that, the iterative procedure for computing $\boldsymbol{\mu}$, U_d and Y given W is the same as that outlined in Algorithm 3.5.

Gven $\boldsymbol{\mu}$, U_d 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 $w_{ij} \approx 0$. One possible choice is

$$w_{ij} = \frac{\varepsilon_0^2}{\varepsilon_{ij}^2 + \varepsilon_0^2}, \tag{3.53}$$

where $\varepsilon_0 > 0$ is a parameter. This is the choice in [De la Torre and Black, 2004].

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 per (3.53). Given these weights, one can reestimate the subspace using the steps of Algorithm 3.5. One can then iterate in between computing the weights given the subspace and computing the subspace given the weights. This iterative process, called iterative re-weighted least squares, converges to a local minima of the cost function (3.52).

3.2.2 Robust PCA by Convex Optimization

Although the above reweighting scheme is very simple and efficient to implement and use in practice, there is no immediate guarantee that the solution that the iteration converges to is the correct low-rank matrix. In this section, we would resort to some advanced tools from high-dimensional sparse signal and low-rank matrix recovery, which gives provably correct solution for a low-rank matrix with intra-sample outliers as long as they 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 their key ideas and results from this approach for PCA with intra-sample outliers. In later Section 3.3.2, we will see how the same ideas apply to the case with sample outliers.

Algorithm 3.7 (Reweighted Least Squares for PCA with Corrupted Data)

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

1: **initialize** $[\boldsymbol{\mu}, U_d, Y] = \text{PCA}(X)$ using Algorithm **??**.
2: **repeat**
3: $\quad \varepsilon_{ij} \leftarrow x_{ij} - \mu_i - \boldsymbol{u}_i^\top \boldsymbol{x}_j$
4: $\quad w_{ij} \leftarrow \frac{\varepsilon_0^2}{\varepsilon_{ij}^2 + \varepsilon_0^2}$,
5: $\quad \mu_i \leftarrow \frac{\sum_{j=1}^N w_{ij}(x_{ij} - \boldsymbol{u}_i^\top \boldsymbol{y}_j)}{\sum_{j=1}^N w_{ij}}$,
6: $\quad \boldsymbol{u}_i \leftarrow \Big(\sum_{j=1}^N w_{ij} \boldsymbol{y}_j \boldsymbol{y}_j^\top \Big)^{-1} \sum_{j=1}^N w_{ij}(x_{ij} - \mu_i)\boldsymbol{y}_j$
7: $\quad U_d = \begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_D^\top \end{bmatrix} \leftarrow \begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_D^\top \end{bmatrix} R^{-1}$, where $QR = \begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_D^\top \end{bmatrix}$
8: $\quad Y = \big[\boldsymbol{y}_1, \cdots, \boldsymbol{y}_d\big]$ where $\boldsymbol{y}_j \leftarrow \Big(\sum_{i=1}^D w_{ij} \boldsymbol{u}_i \boldsymbol{u}_i^\top \Big)^{-1} \sum_{i=1}^D w_{ij}(x_{ij} - \mu_i)\boldsymbol{u}_i$
9: **until** convergence of $\boldsymbol{\mu}\mathbf{1}^\top + U_d Y$

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

Let us assume that the given data matrix X is generated as the sum of two matrices

$$X = L_0 + E_0. \tag{3.54}$$

The matrix L_0 represents the ideal low-rank matrix (the principal components), while the matrix E_0 represents the intra-sample outliers. Since many entries of X are not corrupted (otherwise the problem is not 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

$$\min_{L,E} \quad \text{rank}(L) + \lambda \|E\|_0 \quad \text{s.t.} \quad X = L + E, \tag{3.55}$$

where $\|E\|_0$ is the number of non-zero entries in E and $\lambda > 0$ is a weight parameter.

At a first sight, one may think that solving the problem in (3.55) is really impossible. First of all, we have $D \times N$ equations and $2D \times N$ unknowns. Second, it is not clear that we can always decompose a matrix as the sum of a low-rank matrix and a sparse matrix. For instance, if $x_{11} = 1$ and $x_{ij} = 0$ for all $(i, j) \neq (1, 1)$, then the matrix X is both rank 1 and sparse. Thus, if $\lambda = 1$, we can choose $L = X$ and $E = 0$ or $L = 0$ and $E = X$ as valid solutions. Last, but not least, even if there is a unique global minimum, the cost function to be minimized is non-convex and non-differentiable. In fact, it is well known that problems of re-

covering a sparse signal or a low-rank matrix are in general NP-hard [Amaldi and Kann, 1998].

In what follows, we will show that, under certain conditions on L_0 and E_0, the correct solution to decompose $X \rightarrow (L_0, E_0)$ can be found by solving the following convex optimization problem, which is known as *Principal Component Pursuit* (PCP):

$$\min_{L,E} \quad \|L\|_* + \lambda \|E\|_1 \quad \text{s.t.} \quad X = L + E, \tag{3.56}$$

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 considered as a vector. As we have discussed in Section 3.1.4, the inspiration for using convex relaxations of ℓ_0 norm and matrix rank came from some related problems from compressive sensing, which aim at finding a sparse solution to a linear system $A\boldsymbol{x} = b$ or a low-rank matrix from its linear measurements $\mathcal{A}(X) = b$.

The following theorem gives conditions on the rank of the matrix and the percentage of outliers under which the optimal solution is exactly L_0 and E_0 with overwhelming probability.

Theorem 3.5 ([Candès et al., 2011]). *Let $X = L_0 + E_0$. Assume that there exists a $\mu > 0$ such that the compact SVD of $L_0 = U\Sigma V^\top$ is incoherent with parameter μ according to Definition 3.2. Assume also that the support of E_0 is uniformly distributed among all the sets of cardinality $D \times N$. If*

$$\operatorname{rank}(L_0) \leq \frac{\rho_d \min\{D, N\}}{\mu^2 \log^2\big(\max\{D, N\}\big)} \quad \textit{and} \quad \|E_0\|_0 \leq \rho_s ND. \tag{3.57}$$

Then there is a constant c such that with probability at least $1 - c\max\{N, D\}^{-10}$, the solution (L^*, E^*) to (3.56) with $\lambda = \frac{1}{\sqrt{\max\{N,D\}}}$ is exact, i.e.,

$$L^* = L_0 \quad \textit{and} \quad E^* = E_0. \tag{3.58}$$

A complete proof and explanation for this theorem is beyond the scope of this book – for interested readers please refer to [Candès et al., 2011]. But that does not prevent us from understand its implications and use it to develop practical solutions for real problems. The theorem essentially says, as long as the matrix is incoherent and its rank is bounded almost linearly from its dimension, the convex optimization can correctly recover the low-rank matrix despite a constant fraction of entries are completely corrupted. More recent results show that under benign conditions, the percentage of errors can be surprisingly high (not even have to be sparse [Ganesh et al., 2010]).

Alternating Direction Algorithm for Principal Component Pursuit

Assuming that the conditions of the theorem are satisfied, the next question is how we actually optimize the cost function in order to find the global minimum. Although many convex optimization solvers can be used to solve the convex

optimization (3.56), we here introduce 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*

$$l(L, E, Y) = \|L\|_* + \lambda\|E\|_1 + \langle Y, X - L - E\rangle + \frac{\beta}{2}\|X - L - E\|_F^2. \quad (3.59)$$

A generic Lagrange multiplier algorithm [Bertsekas, 1999] would solve PCP by repeatedly setting $(L_k, E_k) = \arg\min_{L,E} l(L, E, Y_k)$, and then updating the Lagrange multiplier matrix via $Y_{k+1} = Y_k + \beta(X - L_k - E_k)$.

For our low-rank and sparse decomposition problem, we can avoid having to solve a sequence of convex programs by recognizing that $\min_L l(L, E, Y)$ and $\min_E l(L, E, Y)$ both have very simple and efficient solutions. Let $\mathcal{S}_\tau : \mathbb{R} \to \mathbb{R}$ denote the shrinkage operator $\mathcal{S}_\tau[x] = \text{sign}(x)\max(|x| - \tau, 0)$, and extend it to matrices by applying it to each element. It is easy to show that

$$\arg\min_S l(L, E, Y) = \mathcal{S}_{\lambda\beta}(X - L + \beta^{-1}Y). \quad (3.60)$$

Similarly, for matrices X, let $\mathcal{D}_\tau(X)$ denote the singular value thresholding operator given by $\mathcal{D}_\tau(X) = U\mathcal{S}_\tau(\Sigma)V^*$, where $X = U\Sigma V^*$ is any singular value decomposition. It is not difficult to show that

$$\arg\min_L l(L, E, Y) = \mathcal{D}_\beta(X - E + \beta^{-1}Y). \quad (3.61)$$

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

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

1: **initialize:** $E_0 = Y_0 = 0, \beta > 0$.
2: **while** not converged **do**
3: compute $L_{k+1} = \mathcal{D}_\beta(X - E_k + \beta^{-1}Y_k)$;
4: compute $E_{k+1} = \mathcal{S}_{\lambda\beta}(X - L_{k+1} + \beta^{-1}Y_k)$;
5: compute $Y_{k+1} = Y_k + \beta(X - L_{k+1} - E_{k+1})$;
6: **end while**
7: **output:** L, E.

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} via singular value thresholding. This requires us to compute those singular vectors of $X - E_k + \beta^{-1}Y_k$ whose corresponding singular values exceed the threshold β. Empirically, the number of such large singular values is often bounded by $\text{rank}(L_0)$, allowing the next iterate to be computed efficiently via a partial SVD.[1] The most important implementation details for this 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 are also small dense noise in the data. So a more realistic model for robust PCA can be $X = L + E + Z$ where Z are 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 there is no outlier). Nevertheless, one can show that the natural convex extension:

$$\min_{L,E} \quad \|L\|_* + \lambda\|E\|_1 \quad \text{s.t.} \quad \|X - L - E\|_2^2 \leq \varepsilon^2, \tag{3.62}$$

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: $P_Q(X)$ where $P_Q(\cdot)$ could be a general linear operator. The special case of when 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 Theorem 3.5, one can correctly recover the low-rank and sparse components via the following optimization:

$$\min_{L,E} \quad \|L\|_* + \lambda\|E\|_1 \quad \text{s.t.} \quad P_\Omega(X) = P_\Omega(L + E), \tag{3.63}$$

where as in matrix completion, $P_\Omega(\cdot)$ represents projection onto the observable entries. The case of a more general linear operator $P_Q(\cdot)$ for projecting onto an arbitrary subspace Q has also been studied in [Wright et al., 2013], known as *Compressive Principal Component Pursuit* (CPCP). It has been shown that under fairly broad conditions, the low-rank and sparse components can be correctly recovered via the following optimization:

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

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

[1] 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.

3.2.3 Example: Face Images under Different Illuminations

Face recognition is another domain in computer vision where low-dimensional linear models have received a great deal of attention due to the work of [**?**]. The key observation is that under certain idealized circumstances, images of the same face under varying illumination lie near an approximately nine-dimensional linear subspace known as the *harmonic plane*. However, since faces are neither perfectly convex nor Lambertian, face images taken under directional illumination often suffer from self-shadowing, specularities, or saturations in brightness.

Suppose that we have a matrix D whose columns represent perfectly aligned training images of a person's face under various illumination conditions. The above robust PCA algorithm offers a principled way of removing the shadows and specularities in the images since these artifacts are concentrated on small portions of the face images *i.e.,* sparse in the image domain. Figure 3.2 illustrates the results of our algorithm on images from the Extended Yale B database [**?**]. We observe that our algorithm removes the specularities in the eyes and the shadows around the nose region. This technique is potentially useful for pre-processing training images in face recognition systems to remove such deviations from the linear model.

3.3 PCA with Outliers

Another issue that we often encounter in practice is that a small portion of the data points does not fit well the same model as the rest of the data. Such points are called *outliers* or *outlying samples*. 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.[2] Outliers could be atypical samples that have an unusually *large influence* on the estimated model parameters. Outliers could 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.

3.3.1 Outlier Detection by Robust Statistics

In this section, we discuss classical approaches from robust statistics for dealing with outliers in the context of PCA.

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

(a) (b) (c)

Figure 3.2. **Removing shadows and specularities from face images.** (a) Cropped and aligned images of a person's face under different illuminations from the Extended Yale B database. The size of each image is 96×84 pixels, and a total of 29 different illuminations were used for each person. (b) Images recovered by our algorithm. (c) The sparse errors returned by our algorithm correspond to specularities in the eyes, shadows around the nose region, or brightness saturations on the face.

Influence-Based Outlier Detection

This approach relies on the assumption that an outlier is an *atypical* sample which has an unusually large influence on the estimated model parameters. This leads to an outlier detection scheme where the influence of a sample is determined by comparing the difference between the model estimated with and without this sample. For instance, for PCA one may use a *sample influence function* to measure the difference:

$$I(\boldsymbol{x}_i, U_d) \doteq \langle \hat{U}_d, \hat{U}_{d(i)} \rangle, \tag{3.65}$$

where $\langle \cdot, \cdot \rangle$ is the largest subspace angle (see Exercise 2.5) between the subspace span$(\hat{U}_d)$ estimated with the ith sample and the subspace span$(\hat{U}_{d(i)})$ without the ith sample. The larger the difference, the larger the influence of $\boldsymbol{x}_i$ on the estimate, and the more likely that $\boldsymbol{x}_i$ is an outlier. Thus, we may eliminate a sample $\boldsymbol{x}_i$ as an outlier if

$$I(\boldsymbol{x}_i, U_d) \geq \tau \tag{3.66}$$

for some threshold $\tau > 0$ or if $I(\boldsymbol{x}_i, U_d)$ is relatively large among all the samples.

However, this method does not come without an extra cost. We need to compute the principal components (and hence perform SVD) N times: one time with all the samples together and another $N-1$ times with one sample eliminated from each time. There have been many studies that aim to give a formula that can accurately approximate the sample influence without performing SVD N times. Such a formula is called a *theoretical influence* function. For more detailed discussion of the sample influence for PCA, we refer the interested readers to [Jolliffe, 2002].

Probability-Based Outlier Detection

In this approach a model is fit to *all* the 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 model. A new model is then estimated with the detected outliers removed or down-weighted. This process is then repeated until the estimated model stabilizes.

In the case of PCA, the goal is to find a low-dimensional subspace that best fits a given set of data points $\{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^N$ by minimizing the least-squares errors

$$\sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{\mu} - U_d \boldsymbol{y}_i\|^2, \tag{3.67}$$

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

$$\hat{\boldsymbol{x}}_0 = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i \quad \text{and} \quad \hat{\boldsymbol{y}}_i = \hat{U}_d^\top (\boldsymbol{x} - \hat{\boldsymbol{x}}_0), \tag{3.68}$$

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

$$\widehat{\Sigma}_N = \frac{1}{N-1} \sum_{i=1}^{N} (\boldsymbol{x}_i - \hat{\boldsymbol{x}}_0)(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_0)^\top. \tag{3.69}$$

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

$$\varepsilon_i^2 = \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_0 - \hat{U}_d \hat{\boldsymbol{y}}_i\|^2 \quad \text{or} \quad \varepsilon_i^2 = \boldsymbol{x}_i^\top \widehat{\Sigma}_N^{-1} \boldsymbol{x}_i, \quad i = 1, 2, \ldots, N. \tag{3.70}$$

The first error is simply the distance to the subspace, while the second error is the *Mahalanobis distance*,[3] which is obtained when we approximate the probability that a sample $\boldsymbol{x}_i$ comes from this model by a multivariate Gaussian

$$p(\boldsymbol{x}_i; \widehat{\Sigma}_N) = \frac{1}{(2\pi)^{D/2} \det(\widehat{\Sigma}_N)^{1/2}} \exp\big(-\frac{1}{2}\boldsymbol{x}_i^\top \widehat{\Sigma}_N^{-1} \boldsymbol{x}_i\big). \tag{3.71}$$

In principle, we could use $p(\boldsymbol{x}_i, \widehat{\Sigma}_N)$ or either residual ε_i to determine if $\boldsymbol{x}_i$ 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 $w_i \in [0, 1]$ be a weight assigned to the ith point such that $w_i \approx 1$ if $\boldsymbol{x}_i$ is an inlier and $w_i \approx 0$ if $\boldsymbol{x}_i$ is an outlier. Then, similarly to (2.23), a new estimate of the subspace can be obtained by minimizing a weighted least-squares error:

$$\sum_{i=1}^{N} w_i \|\boldsymbol{x}_i - \boldsymbol{\mu} - U_d \boldsymbol{y}_i\|^2 \quad \text{s.t.} \quad U_d^\top U_d = I_d \quad \text{and} \quad \sum_{i=1}^{N} w_i \boldsymbol{y}_i = \boldsymbol{0}. \tag{3.72}$$

If we follow the same steps as in Section 2.1.2, we can find that an optimal solution to this problem is of the form:

$$\hat{\boldsymbol{x}}_0 = \frac{\sum_{i=1}^N w_i \boldsymbol{x}_i}{\sum_{i=1}^N w_i} \quad \text{and} \quad \hat{\boldsymbol{y}}_i = \hat{U}_d^\top (\boldsymbol{x} - \hat{\boldsymbol{x}}_0), \tag{3.73}$$

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

$$\widehat{\Sigma}_N = \frac{\sum_{i=1}^N w_i (\boldsymbol{x}_i - \hat{\boldsymbol{x}}_0)(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_0)^\top}{\sum_{i=1}^N w_i - 1}. \tag{3.74}$$

As a consequence, under the 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 approaches for estimating the weights.

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

$$w_i = \begin{cases} 1 & \text{if } \boldsymbol{x}_i \text{ is an inlier,} \\ 0 & \text{if } \boldsymbol{x}_i \text{ is an outlier,} \end{cases} \tag{3.75}$$

[3] 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 reader may want to find out how this distance is equivalent (or related) to the sample influence $\widehat{\Sigma}_N^{(i)} - \widehat{\Sigma}_N$ or the approximate sample influence given in (B.72).

and chooses the outliers as a certain percentage of the samples (say 10 percent) that have relatively large residual. This can be done by simply sorting the residuals $\{\varepsilon_i\}$ 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 re-estimate the subspace as in (3.73)-(3.74). 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.

Maximum Likelihood Type Estimators (M-Estimators) uses continuous weights $w_i = \rho(\varepsilon_i)/\varepsilon_i^2$ for some robust loss function $\rho(\cdot)$. The objective function then becomes

$$\sum_{i=1}^{N} \rho(\varepsilon_i). \tag{3.76}$$

Many loss functions $\rho(\cdot)$ have been proposed in the statistics literature [Huber, 1981, Barnett and Lewis, 1983]. When $\rho(\varepsilon) = \varepsilon^2$, it corresponds to the standard least-squares solution, which is not robust. Other robust loss functions include

1. L_1 or total variation 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]: $\rho(\varepsilon) = \frac{\varepsilon^2}{\varepsilon^2 + b^2}$,

where $\varepsilon > 0$ is a parameter.

One way of minimizing (3.76) with respect to the subspace parameters is to initialize all the weights to $w_i = 1$, $i = 1, \ldots, N$. This will give an initial estimate for the subspace which is the same as that given by PCA. Given this initial estimate of the subspace, one may compute the weights as $w_i = \rho(\varepsilon)/\varepsilon^2$ using any the aforementioned robust cost functions. Given these weights, one can reestimate the subspace from (3.73)-(3.74). One can then iterate in between computing the weights given the subspace and computing the subspace given the weights. This iterative process, called iterative re-weighted least squares, converges to a local minima of the cost function (3.76). An alternative method for minimizing (3.76) is to simply 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 the M-estimators is that its breakdown point is inversely proportional to the dimension of the space. Thus, the M-estimators become much less robust when the dimension is high. This makes M-estimators of limited use in the context of GPCA since the dimension of the space is typically very high.

Consensus-Based Outlier Detection

This approach assumes that the outliers are not drawn from the same model as the rest of the data. Hence it makes sense to try to avoid the outliers when we infer the model in the first place. However, without knowing which points are outliers beforehand, how can we avoid them? One idea is to fit a model, instead of to all the data points at the same time, only to a *subset* of the data. This is possible when the number of data points required for a unique solution for the estimate is *much* smaller than that of the given data set. Of course, one should *not* expect that a randomly chosen subset will have no outliers and always lead to a good estimate of the model. Thus, one should try on *many different subsets*:

$$X_1, X_2, \ldots, X_n \subset X. \tag{3.77}$$

The rationale is that if the number of subsets are large enough,[4] one of the trial subsets, say X_i, likely contains few or no outliers and hence the resulting model would be the most consistent with the rest of the data points.

In the case of PCA, the minimum number of data points needed to define the model is d for linear subspaces and $d+1$ for affine subspaces. Therefore, each subset X_i is formed by randomly sampling d (or $d+1$) data points and fitting a subspace with basis $\widehat{U}_d(X_i)$ to the subset. The subset X_i gives a consistent estimate $\widehat{U}_d(X_i)$ of the subspace if the number of data points that fit the subspace well is large enough. For instance, we may claim that the subset X_i gives a consistent estimate $\widehat{U}_d(X_i)$ if the following criterion is maximized (among all the chosen subsets):

$$\max_i \#\big\{\boldsymbol{x} \in X : \big\|\boldsymbol{x} - \hat{U}_d(X_i)\big\| \leq \tau\big\}, \tag{3.78}$$

where $\#$ is the cardinality of the set and $\tau > 0$ is a chosen error threshold. This scheme is typically called *Random Sample Consensus* (RANSAC) [Fischler and Bolles, 1981], and it normally improves the robustness of the estimate. As a word of caution, in practice, in order to design a successful RANSAC algorithm, one needs to carefully choose a few key parameters: the size of every subset, the number of subsets, and the consensus criterion.[5]

In theory, the RANSAC scheme can tolerate beyond 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 a decent probability of a good model being

[4] See Appendix B.5 for details on how large this number needs to be.

[5] That is, the criterion that verifies whether each sample is consistent with the model derived from the subset.

estimated. That typically grows exponentially with the dimension of the model and the number of samples. Hence, RANSAC is mostly used in situations when the model dimension is low – in most of the cases we have seen, the number of model parameters does not exceed 10. 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

When we deal with high-dimensional data, the above random sampling techniques could become computationally prohibitive. The other robust statistics based techniques normally do not provide clear conditions under which they could guarantee correctness or global optimality of the solution found. As we have seen in Section 3.2.2 for the problem with inter-sample outliers, in the high-dimensional regime, we may resort to results from high-dimensional statistics and convex optimization and obtain solutions to the outlier detection problem that is provably efficient and correct.

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

In Section 3.2.2 we have seen that we can recover a data matrix X with sparsely corrupted entries by decomposing it into the sum of a low-rank component and a sparse component: $X = L_0 + S_0$, which in turn can be effectively solved via convex relaxation.

We here could attempt to extend the same ideas and techniques to the case of a data matrix X with sparsely corrupted columns. These columns correspond to outlying samples in the dataset. We may assume the fraction of outliers, γ, is small. We denote their column support in X as $\mathcal{I}_0$. Hence, the data matrix naturally can be written as the sum of two components:

$$X = L_0 + C_0,$$

where L_0 is a low-rank matrix which has zero columns on $\mathcal{I}_0$ and C_0 is a matrix which only has nonzero columns on $\mathcal{I}_0$. As γ is small, the columns of C_0 are sparse in X.

Obviously such a decomposition is ill-posed (at least ambiguous) if the matrix X or L_0 are 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 non-zero. We need to introduce a *column incoherence* condition:

Definition 3.6. *A rank r matrix $L \in \mathbb{R}^{D\times N}$ with SVD: $L = U\Sigma V^T$, and nonzero on $(1-\gamma)N$ columns is said to be* column incoherent *with parameter μ if:*

$$\max_i \|V^T \boldsymbol{e}_i\|^2 \leq \frac{\mu r}{(1-\gamma)N}, \tag{3.79}$$

where $\boldsymbol{e}_i$ are the standard basis vectors for $\mathbb{R}^N$.

It is easy to verify that the smaller μ is, the entries of each column of V are more spread in value. For randomly generated low-rank matrices, μ is typically of order $O(1)$.

Even though the incoherence condition may ensure the above low-rank column-sparse decomposition meaningful, there is no guarantee one can find the correct decomposition efficiently. For that, we need to resort to proper relaxation. To this end, we need to use a norm that promotes column-wise sparsity. Let $\{c_1, c_2, \ldots, c_N\}$ be the N columns of the matrix C. The so-called $\ell_{2,1}$ norm of C is the sum of ℓ_2 norm of all the columns:

$$\|C\|_{2,1} = \sum_{i}^{n} \|c_i\|_2. \tag{3.80}$$

If we collect all the ℓ_2 norms of the columns as a vector, the above norm is essentially the ℓ_1 norm of the vector, hence measuring how sparse the columns are.

Similar to PCP for the intra-sample outliers, we can use the following convex optimization:

$$\min_{L,C} \quad \|L\|_* + \lambda \|C\|_{2,1} \quad \text{s.t.} \quad X = L + C, \tag{3.81}$$

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 similarly benign conditions as for Theorem 3.5, the outlier pursuit program can correctly identify the set of sparse (column) outliers.

Theorem 3.7 ([Xu et al., 2010]). *Given $X = L_0 + C_0 \in \mathbb{R}^{D \times N}$. Assume that C_0 is supported on at most γN columns and that there exists a $\mu > 0$ such that L_0 is column incoherent with parameter μ according to Definition 3.6. If*

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

where $c_1 = \frac{9}{121}$, then, with λ set to be $\frac{3}{7\sqrt{\gamma n}}$, the solution (L^*, C^*) to the outlier pursuit program (3.81) 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 $X = L_0 + C_0 + Z$, where Z is a random Gaussian matrix that models small noise in the data, then we can modify the outlier pursuit program as

$$\min_{L,C} \quad \|L\|_* + \lambda \|C\|_{2,1} \quad \text{s.t.} \quad \|X - L - C\|_2^2 \leq \varepsilon^2, \tag{3.82}$$

where ε is the noise variance. It can be shown that under similar conditions as 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 similar optimization techniques as in PCP and those introduced in Appendix A, one can easily develop ALM or ADM based algorithms to solve the above convex optimization problems. We leave that to the reader as an exercise (see Exercise 3.10).

Outlier Detection by ℓ_1 Minimization

3.4 Bibliographic Notes

Completing a low-rank matrix with missing entries has been a problem with a very long and rich history. Since the original work by Wiberg [Wiberg, 1976], one can refer to [Johnson, 1990] for a survey on some of the early developments on this topic.

This problem has drawn tremendous interest in recent years particularly in the area of computer vision and pattern recognition. This is because people are starting to deal with higher dimensional data that are acquired under less controlled conditions. Many algorithms have been proposed to solve this problem in late 1990s and early 2000s, including [Shum et al., 1995, Jacobs, 2001, H.Aanaes et al., 2002, Brandt, 2002]. There has been work [Ke and Kanade, 2005] that proposes the use of ℓ^1 norm for matrix completion and recovery, which extends the original Wiberg method [Wiberg, 1976] from ℓ^2 to ℓ^1 norm.

The seminal work of [Recht et al., 2010, Candès and Recht, 2008] has shown that under broad conditions, one can correctly recover a low-rank matrix with significant amount of missing entries via 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, 2009, Keshavan et al., 2010b], including work that extends to the case with noise [Candès and Plan, 2010].

3.5 Exercises

Exercise 3.1 (Power Method). Let $A \in \mathbb{R}^{N \times N}$ be a symmetric positive semidefinite matrix with eigenvectors $\{u_i\}_{i=1}^N$ and eigenvalues $\{\lambda_i\}_{i=1}^N$ sorted in descending order. Assume that that $\lambda_1 > \lambda_2$ and let u^0 be an arbitrary vector not orthogonal to u_1, i.e., $u_1^\top u^0 \neq 0$. Consider the sequence of vectors

$$u_{k+1} = \frac{Au_k}{\|Au_k\|}. \tag{3.83}$$

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

$$u^k = A^k u^0 = \sum_{i=1}^N \alpha_i \lambda_i^k u_i. \tag{3.84}$$

2. Use this expression to show that u^k converges to $\frac{\alpha_1}{|\alpha_1|}u_1$ with rate $\frac{\lambda_2}{\lambda_1}$. That is, show that there exists a constant $C > 0$ such that for all $k \geq 0$

$$\left\| u^k - \frac{\alpha_1}{|\alpha_1|}u_1 \right\| \leq C\left(\frac{\lambda_2}{\lambda_2}\right)^k. \tag{3.85}$$

Exercise 3.2 (Orthogonal Power Iteration). Let $A \in \mathbb{R}^{N\times N}$ be a symmetric positive semidefinite matrix with eigenvectors $\{u_i\}_{i=1}^N$ and eigenvalues $\{\lambda_i\}_{i=1}^N$ sorted in descending order. 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 spanned by the top d eigenvectors of A, $\{u_i\}_{i=1}^d$. Consider the sequence of matrices

$$U^{k+1} = AU^k(R^k)^{-1}, \tag{3.86}$$

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 3.3 (Convergence of Orthogonal Power Iteration). Prove Theorem 3.1

Exercise 3.4 (Convexity of the Nuclear Norm). Show that the nuclear norm of a matrix, $f(A) = \|A\|_* = \sum_i \sigma_i(A)$, is a convex function of A.

Exercise 3.5 (Sub-gradient of the ℓ_1 Norm). Let X be a matrix. Show that the sub-gradient of the ℓ_1 norm is given by

$$\partial\|X\|_1 = \text{sign}(X) + W \tag{3.87}$$

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

Exercise 3.6 (ℓ_1 Norm Approximation of a Given Matrix). Show that the optimal solution of

$$\min_A \quad \frac{1}{2}\|X - A\|_F^2 + \tau\|A\|_1 \tag{3.88}$$

is given by $A = \mathcal{S}_\tau(X)$.

Exercise 3.7 (Sub-gradient of the Nuclear Norm). Let X be a matrix of rank r. Show that the sub-gradient of the nuclear norm is given by

$$\partial\|X\|_* = UV^\top + W \tag{3.89}$$

where $X = U\Sigma V^\top$ is the compact (rank r) SVD of X and W is a matrix such that $U^\top W = 0$, $WV = 0$ and $\|W\|_2 \leq 1$.

Exercise 3.8 (Nuclear Norm Approximation of a given matrix). Show that the optimal solution of

$$\min_A \quad \frac{1}{2}\|X - A\|_F^2 + \tau\|A\|_* \tag{3.90}$$

is given by $A = \mathcal{D}_\tau(X)$.

Exercise 3.9 (Probability of Selecting a Subset of Inliers). Imagine we have 80 samples from a four-dimensional subspace in $\mathbb{R}^5$. However, the samples are contaminated with another 20 samples that are far from the subspace. We want to estimate the subspace from randomly drawn subsets of four samples. In order to draw a subset that only contains inliers with probability 0.95, what is the smallest number of subsets that we need to draw?

Exercise 3.10 (Algorithm for Outlier Pursuit). Similar to the Algorithm 3.8 for solving the principal component pursuit problem, design an algorithm that solves the outlier pursuit problem (3.81). Discuss what is the difference you have to make and why. Implement the algorithm in Matlab, and verify its correctness with simulated matrices.