CHAPTER

19 DIMENSIONALITY REDUCTION AND LATENT VARIABLES MODELING

CHAPTER OUTLINE

Machine Learning. http://dx.doi.org/10.1016/B978-0-12-801522-3.00019-7

19.1 INTRODUCTION

In many practical applications, although the data reside in a high-dimensional space, the true dimensionality, known as *intrinsic dimensionality*, can be of a much lower value. We have met such cases in the context of sparse modeling in Chapter 9. There, although the data lay in a high-dimensional space, a number of the components were known to be zero. The task was to learn the locations of the zeros; this is equivalent with learning the specific subspace, which is determined by the locations of the nonzero components. In this chapter, the goal is to treat the task in a more general setting and assume that the data can live in any possible subspace (not only the ones formed by the removal of coordinate axes) or manifold. For example, in a three-dimensional space, the data may cluster around a straight line, or around the circumference of a circle or the graph of a parabola, arbitrarily placed in $\mathbb{R}^3$. In all previous cases, the intrinsic dimensionality of the data is equal to one, as any of these curves can equivalently described in terms of a single parameter. Figure 19.1 illustrates the three cases. Learning the lower dimensional structure associated with a given set of data is gaining in importance in the context of *big data* processing and analysis. Some typical examples are the disciplines of computer vision, robotics, medical imaging, and computational neuroscience.

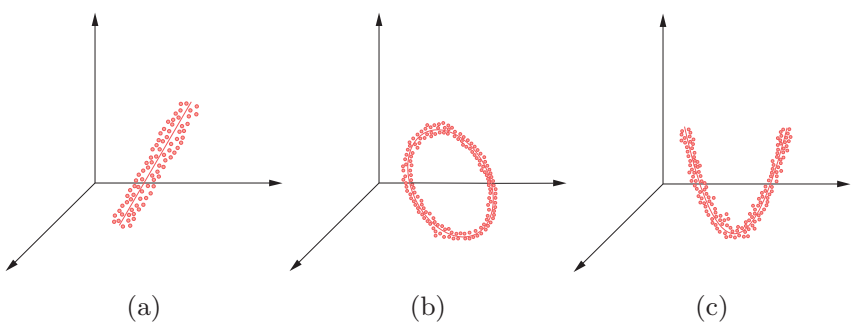

FIGURE 19.1

The data reside close to: (a) a straight line, (b) the circumference of a circle, and (c) the graph of a parabola in the three-dimensional space. In all three cases, the intrinsic dimensionality of the data is equal to one. In (a) the data are clustered around a (translated/affine) linear subspace and in (b) and (c) around one-dimensional manifolds.

The goal of this chapter is to introduce the reader to the main directions, which are followed in this topic, starting from more classical techniques such as the principle component analysis (PCA) and the factor analysis, both in their standard as well as in their more recent probabilistic formulations. Canonical correlation analysis (CCA), independent component analysis (ICA), nonnegative matrix factorization (NMF), and dictionary learning techniques are also discussed; in the latter case, data are represented via an expansion in terms of overcomplete dictionaries, and sparsity-related arguments are mobilized to detect the most relevant atoms in the dictionary. Finally, nonlinear techniques for learning (nonlinear) manifolds are presented such as the kernel PCA, the local linear embedding (LLE), and the isometric mapping (ISOMAP) techniques. At the end of the chapter, a case study in the context of fMRI data analysis is presented.

19.2 INTRINSIC DIMENSIONALITY

A data set, $\mathcal{X} \subset \mathbb{R}^l$, is said to have *intrinsic dimensionality* $m \leq l$, if $\mathcal{X}$ can be (approximately) described in terms of m free parameters. Take as an example the case where the vectors in $\mathcal{X}$ are generated as functions in terms of m random variables, that is, $\mathbf{x} = \mathbf{g}(\mathrm{u}_1, \ldots, \mathrm{u}_m)$, $\mathrm{u}_i \in \mathbb{R},\ i = 1, \ldots, m$. The corresponding geometric interpretation is that the respective observation vectors will lie along a manifold, whose form depends on the vector valued function $\boldsymbol{g} : \mathbb{R}^m \longmapsto \mathbb{R}^l$. Let us consider the case where

$$\mathbf{x} = [r\cos\theta, r\sin\theta]^T,$$

where r is a constant and the random variable $\theta \in [0, 2\pi]$. The data lie along the circumference of a circle of radius r and a single free parameter suffices to describe the data. If now a small amount of noise is added, then the data will be clustered close to the circumference, as for example in Figure 19.1b, and the intrinsic dimensionality is equal to one. From a statistical point of view, it means that the components of the random vectors are highly correlated. Sometimes, we say that, the "effective" dimensionality is lower than the apparent one of the "ambient" space, in which the lower dimensional manifold lies.

In a more general setting, the data may lie in groups of manifolds or even in groups of clusters or they may follow a special spatial or temporal structure. For example, in the wavelet domain most of the coefficients of an image are close to zero and can be neglected, yet the larger (nonzero) ones have a particular structure that is characteristic of natural images. Such a structured sparsity has been exploited in the JPEG2000 coding scheme. Structured sparsity representations are often met in many big data applications and are currently a hot topic of research; see, for example, [41]. In this chapter, we will only focus on identifying manifold structures, linear (subspaces/affine subspaces) in the beginning, and nonlinear ones later on.

Learning the manifold in which a data set resides can be used to provide a compact low-dimensional encoding of a high-dimensional data set, which can subsequently be exploited for performing processing and learning tasks in a much more efficient way. Also, dimensionality reduction can be used for data visualization.

19.3 PRINCIPLE COMPONENT ANALYSIS

Principle component analysis (PCA) or *Karhunen-Loève transform* is among the oldest and most widely used methods for dimensionality reduction, [98]. The assumption underlying PCA, as well as any

dimensionality reduction technique, is that the observed data are generated by a system or process that is driven by a (relatively) small number of *latent* (not directly observed) variables. The goal is to learn this latent structure.

Given a set of observation vectors, $\boldsymbol{x}_n \in \mathbb{R}^l$, $n = 1, 2, \ldots, N$, of a random vector $\mathbf{x}$, which will be assumed to be of zero-mean (otherwise the mean/sample mean is subtracted), PCA determines a *subspace* of dimension $m \leq l$, such that after projection on this subspace, the statistical variation of the data is optimally retained. This subspace is defined in terms of m *mutually orthogonal axes*, known as *principle axes* or *principle directions*, which are computed so that the variance of the data, after projection on the subspace, is maximized, [88].

We will derive the principle axes in a step-wise fashion. First, assume that $m = 1$ and the goal is to find a single direction in $\mathbb{R}^l$ so that the variance of the corresponding projections of the data points is maximized. Let $\boldsymbol{u}_1$ denote the principle axis. The variance of the projections (and having assumed centered data) is given by

$$\begin{aligned} J(\boldsymbol{u}_1) &= \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{u}_1^T\boldsymbol{x}_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{u}_1^T\boldsymbol{x}_n)(\boldsymbol{x}_n^T\boldsymbol{u}_1) \\ &= \boldsymbol{u}_1^T\hat{\Sigma}\boldsymbol{u}_1, \end{aligned}$$

where

$$\hat{\Sigma} := \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^T, \tag{19.1}$$

is the sample covariance matrix of the data. For large values of N or if the statistics can be computed, the covariance (instead of the sample covariance) matrix can be used. The task now becomes that of maximizing the variance. However, because we are only interested in directions, the principle axis will be represented by the respective unit norm vector. Thus, the optimization task is cast as

$$\boldsymbol{u}_1 = \arg\max_{\boldsymbol{u}} \boldsymbol{u}^T\hat{\Sigma}\boldsymbol{u}, \tag{19.2}$$

$$\text{s.t.} \quad \boldsymbol{u}^T\boldsymbol{u} = 1. \tag{19.3}$$

This is a constrained optimization problem and the corresponding Lagrangian is given by

$$L(\boldsymbol{u}, \lambda) = \boldsymbol{u}^T\hat{\Sigma}\boldsymbol{u} - \lambda(\boldsymbol{u}^T\boldsymbol{u} - 1). \tag{19.4}$$

Taking the gradient and setting it equal to zero we get

$$\hat{\Sigma}\boldsymbol{u} = \lambda\boldsymbol{u}. \tag{19.5}$$

In other words, the principle direction is an eigenvector of the sample covariance matrix. Plugging Eq. (19.5) into Eq. (19.2) and taking into account (19.3), we obtain that

$$\boldsymbol{u}^T\hat{\Sigma}\boldsymbol{u} = \lambda. \tag{19.6}$$

Hence, the variance is maximized if $\boldsymbol{u}_1$ is the eigenvector that corresponds to the maximum eigenvalue, λ_1. Recall that, because the (sample) covariance matrix is symmetric and positive semidefinite all the

eigenvalues are real and nonnegative. Assuming $\hat{\Sigma}$ to be invertible (hence, necessarily, $N > l$), the eigenvalues are all positive, that is, $\lambda_1 > \lambda_2 > \ldots \lambda_l > 0$, and we also assume they are distinct, in order to simplify the discussion.

The second principle component is selected so that: (a) is orthogonal to $\boldsymbol{u}_1$ and (b) maximizes the variance after projecting the data onto this direction. Following similar arguments as before, a similar optimization task results with an extra constraint, $\boldsymbol{u}^T\boldsymbol{u}_1 = 0$. It can easily be shown (Problem 19.1) that the second principle axis is the eigenvector corresponding to the second largest eigenvalue, λ_2. The process continues until m principle axes have been obtained; they are the eigenvectors corresponding to the m largest eigenvalues.

PCA, SVD, and low-Rank matrix factorization

The SVD decomposition of a matrix was discussed in Section 6.4. Given a matrix $X \in \mathbb{R}^{l\times N}$, we can write

$$X = UDV^T. \tag{19.7}$$

For a rank r matrix X, U is the $l \times r$ matrix having as columns the eigenvectors corresponding to the r nonzero eigenvalues of XX^T, and V is the $N \times r$ matrix with columns the respective eigenvectors of X^TX. D is a square $r \times r$ diagonal matrix comprising the singular values[1] $\sigma_i := \sqrt{\lambda_i},\ i = 1, 2, \ldots, r$. If we construct X to have as columns the data vectors $\boldsymbol{x}_n,\ n = 1, 2, \ldots, N$, then XX^T is a scaled version of the corresponding sample covariance matrix, $\hat{\Sigma}$; hence, the respective eigenvectors coincide and the corresponding eigenvalues are equal to within a scaling factor (N). Without harming generality, we can assume XX^T to be full rank ($r = l < N$), and Eq. (19.7) becomes

$$X = \underbrace{[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l]}_{l\times l} \underbrace{\begin{bmatrix} \sqrt{\lambda_1}\boldsymbol{v}_1^T \\ \vdots \\ \sqrt{\lambda_l}\boldsymbol{v}_l^T \end{bmatrix}}_{l\times N}. \tag{19.8}$$

Thus, the columns of X can be written in terms of the following expansion[2]

$$\boldsymbol{x}_n = \sum_{i=1}^{l} z_{ni}\boldsymbol{u}_i = \sum_{i=1}^{m} z_{ni}\boldsymbol{u}_i + \sum_{i=m+1}^{l} z_{ni}\boldsymbol{u}_i, \tag{19.9}$$

where $\boldsymbol{z}_n^T := [z_{n1}, \ldots, z_{nl}]$ is the nth column of the $l \times N$ factor on the right-hand side in Eq. (19.8) and the sum has been split into two terms where m can be any value $1 \le m \le l$. Note that, due to the orthonormality of the $\boldsymbol{u}_i$'s,

$$z_{ni} = \boldsymbol{u}_i^T\boldsymbol{x}_n,\ i = 1, 2, \ldots, l,\ n = 1, 2, \ldots, N.$$

[1] Because in some places we are going to involve the variance σ^2, we will carry on working with the square root of the eigenvalues, to avoid possible confusion.

[2] Note that what we have defined in previous chapters as the data matrix is the transpose of X. This is because, for dimensionality reduction tasks, it is more common to work with the current notational convention. If the transpose of X is used, the expansion of the data vectors is in terms of the columns of V and the analysis carries on in a similar way.

From Section 6.4, we know that the best, in the Frobenius sense, m-rank matrix approximation of X is given by

$$\hat{X} = \underbrace{[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m]}_{l \times m} \underbrace{\begin{bmatrix} \sqrt{\lambda_1}\boldsymbol{v}_1^T \\ \vdots \\ \sqrt{\lambda_m}\boldsymbol{v}_m^T \end{bmatrix}}_{m \times N}, \tag{19.10}$$

$$= \sum_{i=1}^{m} \sqrt{\lambda_i}\boldsymbol{u}_i\boldsymbol{v}_i^T. \tag{19.11}$$

Recalling the previous definition of z_{ni}, the nth column vector of $\hat{X}$ can now be written as

$$\hat{\boldsymbol{x}}_n = \sum_{i=1}^{m} z_{ni}\boldsymbol{u}_i. \tag{19.12}$$

Comparing Eqs. (19.9) and (19.12) and taking into account the orthonormality of $\boldsymbol{u}_i$, $i = 1, 2 \ldots, l$, we readily see that $\hat{\boldsymbol{x}}_n$ is the projection of the original observation vectors, $\boldsymbol{x}_n$, $n = 1, 2, \ldots, N$, onto the subspace span$\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m\}$ generated by the m principle axes of XX^T ($\hat{\Sigma}$) (Figure 19.2).

The previous arguments establish a bridge between PCA and SVD. In other words, the principle axes can be obtained via the SVD decomposition of X. Moreover, the columns of the best m-rank matrix approximation, $\hat{X}$, of X are the projections of the observation vectors $\boldsymbol{x}_n$ on the (optimally) reduced in dimension subspace, spanned by the principle axes.

Looking at Eq. (19.10), PCA can also be seen as a *low-rank matrix factorization* method. Matrix factorization will be a recurrent theme in this chapter. Given a matrix, X, there is not a unique way to factorize it, in terms of two matrices. PCA provides an m-rank matrix factorization of X, by imposing orthogonality on the structure of the involved factors. Later on, we are going to discuss other approaches.

Finally, it is important to emphasize that the bridge between PCA and SVD establishes a connection between the low-rank factorization of a matrix, X, and the intrinsic dimensionality of the subspace in which its column vectors reside, since this is the subspace where maximum variance of the data is guaranteed.

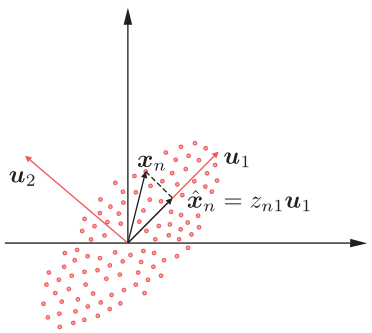

FIGURE 19.2

The projection of $\boldsymbol{x}_n$ on the principle axis $\boldsymbol{u}_1$ is given by $\hat{\boldsymbol{x}}_n = z_{n1}\boldsymbol{u}_1$, where $z_{n1} = \boldsymbol{u}_1^T\boldsymbol{x}_n$.

Minimum error interpretation

Having established the bridge between PCA and SVD, another interpretation of the PCA method becomes readily available. Because $\hat{X}$ is the best m-rank matrix approximation of X in the Frobenius sense, we have that the quantity

$$||\hat{X} - X||_F^2 := \sum_i \sum_j |\hat{X}(i,j) - X(i,j)|^2 = \sum_{n=1}^{N} ||\hat{\boldsymbol{x}}_n - \boldsymbol{x}_n||^2$$

is minimum; that is, obtaining any other m-dimensional approximation (say $\tilde{\boldsymbol{x}}_n$) of $\boldsymbol{x}_n$, by choosing to project onto another m-dimensional subspace, would result in higher squared error norm approximation, compared to that resulting from PCA. This is also a strong result that establishes a notable merit of the PCA method as a dimensionality reduction technique. This interpretation goes back to Pearson, [137].

PCA and information retrieval

The previous minimum error interpretation paves the way to build around PCA an efficient searching procedure in identifying similar patterns in large databases. Assume that a number N of prototypes are represented in terms of l features, giving rise to feature vectors, $\boldsymbol{x}_n \in \mathbb{R}^l$, $n = 1, 2, \ldots, N$, which are stored in a database. Given an unknown object, which is represented by a feature vector $\boldsymbol{x}$, the task is to identify to which one among the prototypes this pattern is most similar. Similarity is measured in terms of the Euclidean distance $||\boldsymbol{x} - \boldsymbol{x}_n||^2$. If N and l are large, searching for the minimum Euclidean distance can be computationally very expensive. The idea is to keep in the database the components $\boldsymbol{z}_n^{(m)} := [z_{n1}, \ldots, z_{nm}]^T$ (see Eq. (19.12)) that describe the projections of the N prototypes in $\text{span}\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m\}$, instead of the original l dimensional feature vectors. Assuming that m is large enough to capture most of the variability of the original data (i.e., the intrinsic dimensionality of the data is m to a good approximation), then $\boldsymbol{z}_n^{(m)}$ is a good feature vector description because we know that in this case $\hat{\boldsymbol{x}}_n \approx \boldsymbol{x}_n$. Given now an unknown pattern, $\boldsymbol{x}$, we first project it onto $\text{span}\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m\}$ resulting in

$$\hat{\boldsymbol{x}} = \sum_{i=1}^{m} (\boldsymbol{u}_i^T \boldsymbol{x}) \boldsymbol{u}_i := \sum_{i=1}^{m} z_i \boldsymbol{u}_i. \tag{19.13}$$

Then we have

$$\begin{aligned} ||\boldsymbol{x}_n - \boldsymbol{x}||^2 \approx ||\hat{\boldsymbol{x}}_n - \hat{\boldsymbol{x}}||^2 &= \left\| \sum_{i=1}^{m} z_{ni} \boldsymbol{u}_i - \sum_{i=1}^{m} z_i \boldsymbol{u}_i \right\|^2 \\ &= ||\boldsymbol{z}_n^{(m)} - \boldsymbol{z}||^2, \end{aligned}$$

where $\boldsymbol{z} := [z_1, \ldots, z_m]^T$. In other words, Euclidean distances are computed in the lower dimensional subspace, which leads to substantial computational gains; see, for example, [21, 58, 150] and the references therein. This method is also known as *latent semantics indexing*.

Orthogonalizing properties of PCA and feature generation

We will now shed light on PCA from a different angle. We have just discussed, in the context of the information retrieval application, that PCA can also be seen as a feature generation method that generates a set of new feature vectors, $\boldsymbol{z}$, whose components describe a pattern in terms of the principle axes. Let us now assume (to make life easier) that N is large enough and the sample covariance matrix

is a good approximation of the (full rank) covariance matrix $\Sigma = \mathbb{E}[\mathbf{x}\mathbf{x}^T]$. We know that any vector $\boldsymbol{x} \in \mathbb{R}^l$ can be described in terms of $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l$, that is,

$$\boldsymbol{x} = \sum_{i=1}^{l} z_i \boldsymbol{u}_i = \sum_{i=1}^{l} (\boldsymbol{u}_i^T \boldsymbol{x}) \boldsymbol{u}_i.$$

Our focus now turns to the covariance matrix of the random vectors, $\mathbf{z}$, as $\mathbf{x}$ changes randomly. Taking into account that

$$\mathrm{z}_i = \boldsymbol{u}_i^T \mathbf{x}, \tag{19.14}$$

and the definition of U in Eqs. (19.7)–(19.8), we can write as $\mathbf{z} = U^T \mathbf{x}$, hence,

$$\mathbb{E}[\mathbf{z}\mathbf{z}^T] = \mathbb{E}\left[U^T \mathbf{x}\mathbf{x}^T U\right] = U^T \Sigma U.$$

However, we know from linear algebra (Appendix A.2) that U is the matrix that diagonalizes Σ, hence,

$$\mathbb{E}[\mathbf{z}\mathbf{z}^T] = \mathrm{diag}\{\lambda_1, \ldots, \lambda_l\}. \tag{19.15}$$

In other words, the new features are *uncorrelated*, that is,

$$\boxed{\mathbb{E}[\mathrm{z}_i \mathrm{z}_j] = 0, \quad i \neq j, \; i,j = 1, 2, \ldots, l.} \tag{19.16}$$

Furthermore, note that, the variances of z_i are equal to the eigenvalues λ_i, $i = 1, 2, \ldots, l$, respectively. Hence, by selecting as features the ones that correspond to the dominant eigenvalues, one has maximally retained the total variance associated with the original features, x_i; indeed, the corresponding total variance is given by the trace of the covariance matrix, which in turn is equal to the sum of the eigenvalues, as we know from linear algebra. In other words, the new set of features, z_i, $i = 1, 2, \ldots, m$, represent the patterns in a more compact way, as they are *mutually uncorrelated* and most of the variance is retained. It is common in practice, when the goal is that of feature generation, for each one of the z_i's to be normalized to unit variance.

Later on, we will see that a more recent method, known as independent component analysis (ICA), imposes the constraint that after a linear transformation (a projection is a linear transformation, after all) the obtained latent variables (components) are statistically independent, which is a much stronger condition than being uncorrelated.

Latent variables

The random components, z_i, $i = 1, 2, \ldots, m$, are known as *principle components*. Sometimes, their observed values, z_i, are known as *principle scores*. As a matter of fact, the principle components comprise the *latent* variables, which we mentioned at the beginning of this section.

According to the general (linear) latent variables modeling approach, we assume that our l variables comprising, $\mathbf{x}$, are modeled as

$$\mathbf{x} \approx A\mathbf{z}, \tag{19.17}$$

where A is an $l \times m$ matrix and $\mathbf{z} \in \mathbb{R}^m$ is the corresponding set of latent variables. Adopting the PCA model, we have shown that

$$A = U = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m],$$

and the model implies that each one of the l components of $\mathbf{x}$ is (approximately) generated in terms of these mutually uncorrelated m latent random variables, that is,

$$\mathrm{x}_i \approx u_{i1}\mathrm{z}_1 + \ldots + u_{im}\mathrm{z}_m. \tag{19.18}$$

Alternatively, in the linear latent variables modeling, we can assume that the latent variables can also be recovered by a linear model from the original random variables, as for example,

$$\mathbf{z} = W\mathbf{x}. \tag{19.19}$$

In the case of the PCA approach, we have already seen that

$$W = U^T.$$

Equations (19.17) and (19.19) constitute the backbone of this chapter, and different methods provide different solutions for computing A or W.

Let us now collect all the principle score vectors, z_n, $n = 1, 2, \ldots, N$, as the columns of the $m \times N$ score matrix Z, that is,

$$Z := [z_1, \ldots, z_N]. \tag{19.20}$$

Then (19.10) can be rewritten in terms of the score matrix

$$\boxed{X \approx UZ.} \tag{19.21}$$

Moreover, taking into account the definition of the principle components in Eq. (19.14), we can also write

$$\boxed{Z = U^T X.} \tag{19.22}$$

Remarks 19.1.

- A major issue in practice is to select the m dominant eigenvalues. One way is to rank them in descending order, and determine m so that the gap between λ_m and λ_{m+1} is "large." The interested reader can obtain more on this issue in [51, 97].
- The treatment so far involved centered quantities. In case we want to approximate the original observation vectors by taking into consideration the respective mean value of the data set, Eq. (19.13) is rephrased as

$$\hat{\boldsymbol{x}} = \bar{\boldsymbol{x}} + \sum_{i=1}^{m} \boldsymbol{u}_i^T(\boldsymbol{x} - \bar{\boldsymbol{x}})\boldsymbol{u}_i, \tag{19.23}$$

  where $\bar{\boldsymbol{x}}$ is the sample mean (mean if it is known)

$$\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n,$$

  and $\boldsymbol{x}$ denotes the original (not centered) vector.
- PCA builds upon *global* information spread over *all* the data observations in the set $\mathcal{X}$. Indeed, the main source of information is the sample covariance matrix (XX^T). Thus, PCA is effective if the covariance matrix provides a sufficiently rich description of the data at hand. For example, this is the case for Gaussian-like distributions. In [40], modifications of the standard approach are

suggested in order to deal with data having a clustered nature. Soon, we are going to discuss alternative to PCA techniques in order to overcome this drawback.

- Computing the SVD of large matrices can be computationally costly and a number of efficient techniques have been proposed (see, e.g., [1, 77, 183]). In a number of cases in practice, it turns out that $l > N$. Of course, in this case, the sample covariance is not invertible and some of the eigenvalues are zero. In such scenarios, it is preferable to work with $X^T X$ ($N \times N$) instead of XX^T ($l \times l$) matrix. To this end, the relationships given in Section 6.4, in order to obtain $\boldsymbol{u}_i$ from $\boldsymbol{v}_i$, can be employed.
- The treatment of PCA bears a similarity with the Fisher's linear discriminant method (FLD) (Chapter 7). They both rely on the eigenstructure of matrices that, in one way or another, encode (co)variance information. However, note that PCA is an *unsupervised* method in contrast to FLD, which is a *supervised* one. As a consequence, PCA performs dimensionality reduction so as to preserve data variability (variance) while FLD class separability. Figure 19.3 demonstrates the difference in the resulting (hyper)planes.
- *Multidimensional Scaling* (MDS) is another linear technique used to project in a lower dimensional space, while respecting certain constraints. Given the set $\mathcal{X} \subset \mathbb{R}^l$, the goal is to project onto a lower dimensional space, so that inner products are optimally preserved; that is, the cost

$$E = \sum_i \sum_j \left(\boldsymbol{x}_i^T \boldsymbol{x}_j - \boldsymbol{z}_i^T \boldsymbol{z}_j \right)^2$$

is minimized, where $\boldsymbol{z}_i$ is the image of $\boldsymbol{x}_i$ and the sum runs over all the training points in $\mathcal{X}$. The problem is similar to the PCA and it can be shown that the solution is given by the eigendecomposition of the Gram matrix,[3] $\mathcal{K} := X^T X$. Another side of the same coin is to require the Euclidean distances, instead of the inner products, to be optimally preserved. A Gram matrix, consistent with the squared Euclidean distances, can then be formed, leading to the same solution

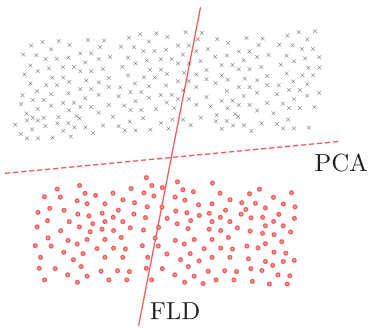

FIGURE 19.3

The case of a two-class task in the two-dimensional space. PCA computes the direction along which the variance is maximally retained after the projections of the data on it. In contrast, FLD computes the line so that the class separability is maximized.

[3] In order to avoid confusion, recall that here X has been defined as the transpose of what we called a data matrix in previous chapters.

as before. It turns out that the solutions obtained by PCA and MDS are equivalent. This can readily be understood as X^TX and XX^T share the same (nonzero) eigenvalues. The corresponding eigenvectors are different, yet they are related, as we have seen while introducing the SVD in Section 6.4.

More on these issues can be found in [27, 56]. As we will soon see in Section 19.9, the main idea behind MDS of preserving the distances is used, in one way or another, in a number of more recently developed nonlinear dimensionality reduction techniques.

- In a variant of the basic PCA, known as *supervised PCA* [15, 184], the output variables in regression or in classification (depending on the problem at hand) are used together with the input ones, in order to determine the principle directions.

Example 19.1. This example demonstrates the power of PCA as a method to represent data in a lower dimension space. Each pattern in a database, described in terms of a feature vector, $\boldsymbol{x}_n \in \mathbb{R}^l$, will be represented by a corresponding vector of a reduced dimensionality, $\boldsymbol{z}_n^{(m)} \in \mathbb{R}^m$, $n = 1, 2, \ldots, N$.

In this example, each feature vector comprises the pixels of a 168×168 face image. These face images are members of the software-based aligned version, [180], of the *labeled faces in the wild* (LFW) database [95]. In particular, among the over 13,000 face images of this database, $N = 1924$ have been selected with criteria such as the quality of the image and the face angle (portraits were of preference). Moreover, the images are zoomed in order to omit most of the background. Examples of the face images used are depicted in Figure 19.4 and the full collection of all the 1924 images can be found in the companion site of this book.

The images are first vectorized (in $\mathbb{R}^l$, $l = 168 \times 168 = 28,224$) and in the sequel are concatenated in the columns of the $28,224 \times 1924$ matrix X. Moreover, the mean value across each one of the rows is computed and then subtracted from the corresponding element of each column.

In this case, where $l > N$, it is convenient to compute the eigenvectors of X^TX, denoted by $\boldsymbol{v}_i$, $i = 1, \ldots, N$, and then the principle axes directions, that is, the eigenvectors of XX^T are computed by $\boldsymbol{u}_i \propto X\boldsymbol{v}_i$ (Chapter 6, Eq. 6.16). These eigenvectors can be rearranged in a matrix form to give 168×168 images known as *eigenimages*, which in the particular case of face images are referred to as *eigenfaces*. Figure 19.5 shows examples of eigenfaces resulted by the PCA of matrix X and specifically those corresponding, from top left to bottom right, to the 1st, 2nd, 6th, 7th, 8th, 10th, 11th, and 17th larger eigenvalues.

Next, the quality of reconstruction of an original image, in terms of its lower dimensional representation, is examined according to Eq. (19.13) for different values of m. As an example, the images depicting Marilyn Monroe and Andy Warhol, shown in Figure 19.4, are chosen. The results

FIGURE 19.4

Indicative examples of the face images used.

FIGURE 19.5

Examples of eigenfaces.

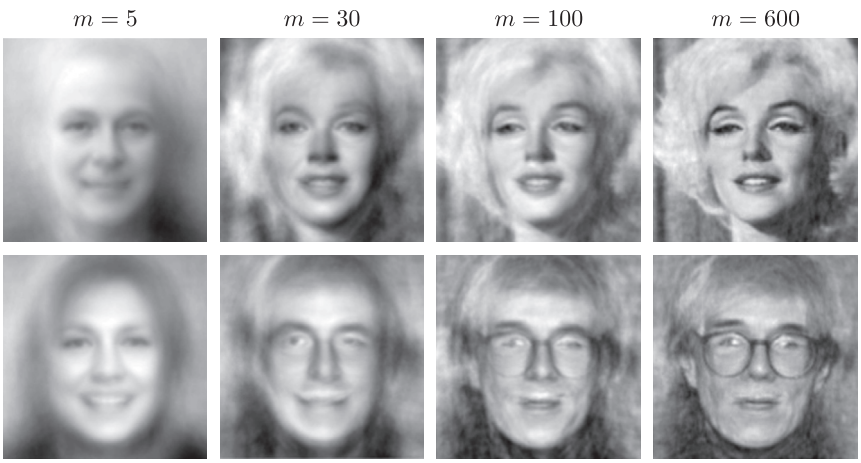

FIGURE 19.6

Image compression and reconstruction based on the first m eigenvectors.

are illustrated in Figure 19.6. It is observed that, for $m = 100$ or even better for $m = 600$, the resulting approximation is very close to the original images. Note that exact reconstruction will be achieved when the full set of the 1924 eigenfaces are used.

To put our previous findings in an information retrieval context, assume than one has available an image and wants to know what person is depicted in it. Assuming that the image of this person is in the database, the procedure would be (a) to vectorize the image, (b) to project it onto the subspace spanned by the, say, $m = 100$ eigenfaces, and (c) to search in this lower dimensional space to identify

the vectorized image in the database that is closer in the Euclidean norm sense. Usually, it is preferable to identify the, say, five or ten most similar images and rank them according to the Euclidean distance (or any other distance) similarity. Then, through the database, he/she can have the name and all the associated information that is kept in the database.

In information retrieval, each one of the images in the database, could be stored in terms of the corresponding vector of the principle scores.

Example 19.2. In this example, the use of PCA for image compression is demonstrated. In the previous example, PCA was performed across the different images of a database. Here, the focus will be on a single image.

The pixel values of the image are stored in an $l \times N$ matrix X and the columns of this matrix are considered to be the observation vectors $\boldsymbol{x}_n \in \mathbb{R}^l$, $n = 1, 2, \ldots, N$. Note that X needs to be zero-mean along the rows so the mean vector, $\bar{\boldsymbol{x}}$, is computed and subtracted from each column. Then the eigenvectors corresponding to the m, $1 \leq m < l$ largest eigenvalues are obtained either via the sample covariance matrix or directly through SVD. Exploiting the matrix factorization formulation of PCA in Eq. (19.22) a compressed representation of X, comprising m instead of l rows, is given by

$$Z^{(m)} = \underbrace{[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m]^T}_{m \times l} X, \tag{19.24}$$

where the dimensionality m has been explicitly brought into the notation. Thus, only $Z^{(m)}$ and $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m$, are needed in order to get an estimate of the, meansubtracted, X via Eq. (19.21). Finally, in order to reconstruct the image, the mean vector $\bar{\boldsymbol{x}}$ need to be added back to each column, see Eq. (19.23).

The effectiveness of the PCA-based image compression will be demonstrated with the aid of the top-left image depicted in Figure 19.7. This image is square having $l = N = 400$. For any m chosen, the compression ratio is easily computed considering that instead of 400×400 values, of the original image, after compression the storage of $2 \times m \times 400$ values, for the matrix, $Z^{(m)}$ and the eigenvectors, $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m$, plus 400 values for the mean vector $\bar{\boldsymbol{x}}$ are needed. This amounts to a compression ratio of $400 : (2m + 1)$. The reconstructed images together with the corresponding MSE, between the original and the reconstructed image, for different compression rates, are shown in Figure 19.7.

Remarks 19.2.

- *Subspace Tracking*: Online subspace tracking is another old area with a revived interest recently.

  A well-known algorithm of relative low complexity, for tracking the signal subspace, is the so-called projection approximation subspace tracking (PAST) proposed in [186]. In PAST, the recursive least-squares (RLS) technique is employed for subspace estimation. Alternative algorithms in this line of philosophy have been presented in, for example, [64, 108, 160, 169].

  More recently, the work in [47, 80, 129] tackles the problem of subspace tracking with missing/unobserved data. The methodology presented in [80] is based on gradient descent iterations on the Grassmannian manifold. Furthermore, the algorithms of [47, 129] attempt to estimate the unknown subspace by minimizing properly constructed loss functions.

  Finally, [48, 49, 85, 124, 152] attack the subspace tracking problem in environments where observations are contaminated by outlier noise.

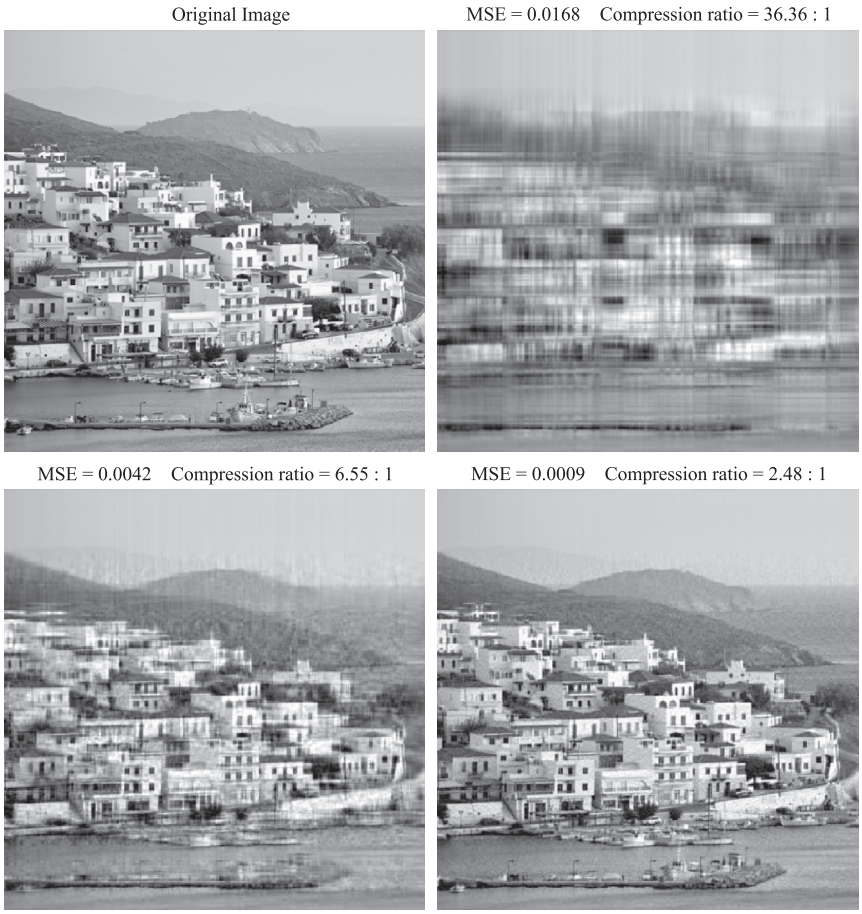

FIGURE 19.7

PCA-based image compression. The image is from the Greek island Andros.

19.4 CANONICAL CORRELATION ANALYSIS

PCA is a dimensionality reduction technique focusing on a *single* data set. However, in a number of cases, one has to deal with multiple data sets, which although they may originate from different sources, they are closely related. For example, many problems in medical imaging fall under this umbrella. A typical case occurs in the study of brain activity where one can use different modalities, for example, electroencephalogram (EEG), functional magnetic resonance imaging (fMRI), or structural MRI. Each one of these modalities can grasp a different type of information and it is beneficial to exploit all of them in a complementary fashion. Thus, the respective experimental data can appropriately be fused

in order to get a better description concerning the brain activity that gives birth to the data. Another scenario where multiple data sets are of interest, is when a single modality is used but different data are available measured on different subjects; thus, jointly analyzing the results can be beneficial for the finally reached conclusions (see, e.g., [52]).

Canonical correlation analysis (CCA) is an old technique developed in [89] in order to process two data sets jointly. Our starting point is the fact that when two sets of random variables (two random vectors) are involved, the value of their correlation *does depend* on the coordinate system in which the random vectors are represented. The goal behind CCA is to seek a pair of linear transformations, one for each set of variables, such that after the transformation, the resulting transformed variables are *maximally correlated*.

Let us assume that we are given two sets of random variables comprising the components of two random vectors, $\mathbf{x} \in \mathbb{R}^p$ and $\mathbf{y} \in \mathbb{R}^q$, and let the corresponding sets of observations be $\boldsymbol{x}_n, \boldsymbol{y}_n$, $n = 1, 2, \ldots, N$, respectively. Following a step-wise procedure, as we did for PCA, we will first compute a single pair of directions, namely $\boldsymbol{u}_{x,1}, \boldsymbol{u}_{y,1}$, so that the correlation between the projections onto these directions is maximized. Let $\mathrm{z}_{x,1} := \boldsymbol{u}_{x,1}^T \mathbf{x}$ and $\mathrm{z}_{y,1} := \boldsymbol{u}_{y,1}^T \mathbf{y}$ be the (zero-mean) random variables after the linear transformation (projection). Note that these variables are the counterparts of what we called principle components in PCA. The corresponding correlation coefficient (normalized covariance) is defined as

$$\rho := \frac{\mathbb{E}[\mathrm{z}_{x,1}\mathrm{z}_{y,1}]}{\sqrt{\mathbb{E}\left[\mathrm{z}_{x,1}^2\right]\mathbb{E}\left[\mathrm{z}_{y,1}^2\right]}} = \frac{\mathbb{E}\left[(\boldsymbol{u}_{x,1}^T\mathbf{x})(\mathbf{y}^T\boldsymbol{u}_{y,1})\right]}{\sqrt{\mathbb{E}\left[(\boldsymbol{u}_{x,1}^T\mathbf{x})^2\right]\mathbb{E}\left[(\boldsymbol{u}_{y,1}^T\mathbf{y})^2\right]}}$$

or

$$\rho := \frac{\boldsymbol{u}_{x,1}^T \Sigma_{xy} \boldsymbol{u}_{y,1}}{\sqrt{(\boldsymbol{u}_{x,1}^T \Sigma_{xx} \boldsymbol{u}_{x,1})(\boldsymbol{u}_{y,1}^T \Sigma_{yy} \boldsymbol{u}_{y,1})}}, \tag{19.25}$$

where

$$\mathbb{E}\left[\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}[\mathbf{x}^T,\ \mathbf{y}^T]\right] := \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}. \tag{19.26}$$

Note that, by the respective definition, we have $\Sigma_{xy} = \Sigma_{yx}^T$. When expectations are not available, covariances are replaced by the corresponding sample covariance values. This is the most common case in practice, so we will adhere to it and use the notation with the "hat." Furthermore, it can easily be checked out that the correlation coefficient is invariant to scaling (changing, e.g., $\mathbf{x} \to b\mathbf{x}$). Thus, maximizing it with respect to the directions $\boldsymbol{u}_{x,1}$ and $\boldsymbol{u}_{y,1}$, can equivalently be cast as the following constrained optimization task,

$$\boxed{\begin{aligned} \max_{\boldsymbol{u}_x, \boldsymbol{u}_y} \quad & \boldsymbol{u}_x^T \hat{\Sigma}_{xy} \boldsymbol{u}_y, && (19.27)\\ \text{s.t.} \quad & \boldsymbol{u}_x^T \hat{\Sigma}_{xx} \boldsymbol{u}_x = 1, && (19.28)\\ & \boldsymbol{u}_y^T \hat{\Sigma}_{yy} \boldsymbol{u}_y = 1. && (19.29) \end{aligned}}$$

Compare Eqs. (19.27)–(19.29) with the optimization task defining PCA in Eqs. (19.2)–(19.3). For CCA, two directions have to be computed and the constraints involve the weighted Σ-norm instead of the

Euclidean one. Moreover, in PCA the variance is maximized, while CCA cares for the correlation between the projections of the two involved vectors onto the new axes.

Employing Lagrange multipliers, the corresponding Lagrangian of Eqs. (19.27)–(19.29) is given by

$$L(\boldsymbol{u}_x, \boldsymbol{u}_y, \lambda_x, \lambda_y) = \boldsymbol{u}_x^T \hat{\Sigma}_{xy} \boldsymbol{u}_y - \frac{\lambda_x}{2} \left(\boldsymbol{u}_x^T \hat{\Sigma}_{xx} \boldsymbol{u}_x - 1 \right) - \frac{\lambda_y}{2} \left(\boldsymbol{u}_y^T \hat{\Sigma}_{yy} \boldsymbol{u}_y - 1 \right).$$

Taking the gradients with respect to $\boldsymbol{u}_x$ and $\boldsymbol{u}_y$ and equating to zero, we obtain (Problem 19.2)

$$\lambda_x = \lambda_y := \lambda,$$

and

$$\hat{\Sigma}_{xy} \boldsymbol{u}_y = \lambda \hat{\Sigma}_{xx} \boldsymbol{u}_x, \tag{19.30}$$

$$\hat{\Sigma}_{yx} \boldsymbol{u}_x = \lambda \hat{\Sigma}_{yy} \boldsymbol{u}_y. \tag{19.31}$$

Solving the latter of the two with respect to $\boldsymbol{u}_y$ and substituting to the first one, we finally get

$$\hat{\Sigma}_{xy} \hat{\Sigma}_{yy}^{-1} \hat{\Sigma}_{yx} \boldsymbol{u}_x = \lambda^2 \hat{\Sigma}_{xx} \boldsymbol{u}_x, \tag{19.32}$$

and

$$\boldsymbol{u}_y = \frac{1}{\lambda} \hat{\Sigma}_{yy}^{-1} \hat{\Sigma}_{yx} \boldsymbol{u}_x, \tag{19.33}$$

assuming, of course, invertibility of $\hat{\Sigma}_{yy}$. Furthermore, assuming invertibility of $\hat{\Sigma}_{xx}$, too, we end up with the following eigenvalue-eigenvector problem:

$$\left(\hat{\Sigma}_{xx}^{-1} \hat{\Sigma}_{xy} \hat{\Sigma}_{yy}^{-1} \hat{\Sigma}_{yx} \right) \boldsymbol{u}_x = \lambda^2 \boldsymbol{u}_x. \tag{19.34}$$

Thus, the axis $\boldsymbol{u}_{x,1}$ is obtained as an eigenvector of the product of matrices in the parentheses in Eq. (19.34). Taking into account Eq. (19.30) and the constraints, it turns out that the corresponding optimal value of the correlation, ρ, is equal to

$$\rho = \boldsymbol{u}_{x,1}^T \hat{\Sigma}_{xy} \boldsymbol{u}_{y,1} = \lambda \boldsymbol{u}_{x,1}^T \hat{\Sigma}_{xx} \boldsymbol{u}_{x,1} = \lambda.$$

Hence, selecting $\boldsymbol{u}_{x,1}$ to be the eigenvector corresponding to the maximum eigenvalue, λ^2, results in maximum correlation.

The eigenvectors $\boldsymbol{u}_{x,1}$, $\boldsymbol{u}_{y,1}$ are known as the *normalized canonical correlation basis vectors*, the eigenvalue λ^2 as the squared *canonical correlation*, and the projections $z_{x,1}$, $z_{y,1}$ as the *canonical variates*.

The previous idea can now be taken further and compute a pair of subspaces, $\text{span}\{\boldsymbol{u}_{x,1}, \ldots, \boldsymbol{u}_{x,m}\}$, $\text{span}\{\boldsymbol{u}_{y,1}, \ldots, \boldsymbol{u}_{y,m}\}$, where $m \leq \min(p, q)$. One way to achieve this goal is in a step-wise fashion, as it was done for the PCA. Assuming that k pairs of basis vectors have already been computed, the $k+1$ is obtained by solving the following constrained optimization task,

$$\max_{\boldsymbol{u}_x, \boldsymbol{u}_y} \ \boldsymbol{u}_x^T \hat{\Sigma}_{xy} \boldsymbol{u}_y, \tag{19.35}$$

$$\text{s.t.} \ \ \boldsymbol{u}_x^T \hat{\Sigma}_{xx} \boldsymbol{u}_x = 1, \ \ \boldsymbol{u}_y^T \hat{\Sigma}_{yy} \boldsymbol{u}_y = 1, \tag{19.36}$$

$$\boldsymbol{u}_x^T \hat{\Sigma}_{xx} \boldsymbol{u}_{x,i} = 0, \ \ \boldsymbol{u}_y^T \hat{\Sigma}_{yy} \boldsymbol{u}_{y,i} = 0, \ \ i = 1, 2, \ldots, k, \tag{19.37}$$

$$\boldsymbol{u}_x^T \hat{\Sigma}_{xy} \boldsymbol{u}_{y,i} = 0, \ \ \boldsymbol{u}_y^T \hat{\Sigma}_{yx} \boldsymbol{u}_{x,i} = 0, \ \ i = 1, 2, \ldots, k. \tag{19.38}$$

In other words, every new pair of vectors is computed so as to be normalized (19.36) and at the same time, each one to be orthogonal (in the generalized sense) to those obtained in the previous iteration steps (Eqs. (19.37) and (19.38)). Note that this guarantees that the derived canonical variates are uncorrelated to all previously derived ones. This reminds us of the uncorrelatedness property of the principle components in PCA. The only nonzero correlation in CCA, which is maximized at every iteration step, is the one between $z_{x,k} = \boldsymbol{u}_{x,k}^T \mathbf{x}$ and $z_{y,k} = \boldsymbol{u}_{y,k}^T \mathbf{y}$, $k = 1, 2, \ldots, m$.

More on CCA can be found in [6, 24]. Extensions of CCA in reproducing kernel Hilbert spaces have also been developed and used; see, for example, [9, 82, 110] and the references therein. In [82], the kernel CCA is used for content-based image retrieval. The aim is to allow retrieval of images from a text query but without reference to any labeling associated with the image. The task is treated as a cross-modal problem. A probabilistic Bayesian formulation of CCA has been given in [14, 106]. A regularized CCA version, using sparsity-based arguments, has been derived in [83]. In [60], a variant of CCA is proposed, named *correlated component analysis*; instead of two directions (subspaces), a common direction is derived for both data sets. The idea behind this method is that the two data sets may not be much different, so a single direction is enough. In this way, the task has fewer free parameters to estimate. Moreover, the constraint on orthogonality is dropped, which in some cases may not be physically justifiable. A Bayesian extension of the method is provided in [138].

Example 19.3. Let $\mathrm{x} \in \mathbb{R}^2$ be a normally distributed random vector, $\mathcal{N}(\mathbf{0}, I)$. The pair of random variables, $(\mathrm{y}_1,\ \mathrm{y}_2)$, are related to $(\mathrm{x}_1,\ \mathrm{x}_2)$ as

$$\mathbf{y} = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix} \mathbf{x}.$$

Note the strong correlation that exists between the involved variables, because

$$\mathrm{y}_1 + \mathrm{y}_2 = \mathrm{x}_1 + \mathrm{x}_2.$$

However, the cross-covariance matrix Σ_{yx},

$$\Sigma_{yx} = AI = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix},$$

indicates a rather low correlation. After performing CCA, the resulting directions are

$$\boldsymbol{u}_{x,1} = \boldsymbol{u}_{y,1} = -\frac{1}{\sqrt{2}}[1, 1]^T,$$

which actually is the direction where the linear equality of the involved variables lies. The maximum correlation coefficient value is equal to 1, indicating strong correlation indeed.

19.4.1 RELATIVES OF CCA

CCA is not the only multivariate technique to process and deal with different data sets jointly. Various techniques have been developed, using different optimizing criteria/constraints, each one serving different needs and goals.

The aim of this subsection is to briefly discuss some of these methods under a common framework. Recall that the eigenvalue-eigenvector problem for computing the pair of canonical basis vectors results

from the pair of equations in Eqs. (19.30)–(19.31). These can be combined into a single one [24], namely

$$C\boldsymbol{u} = \lambda B\boldsymbol{u}, \tag{19.39}$$

where

$$\boldsymbol{u} := [\boldsymbol{u}_x^T, \boldsymbol{u}_y^T]^T,$$

and

$$C := \begin{bmatrix} O & \hat{\Sigma}_{xy} \\ \hat{\Sigma}_{yx} & O \end{bmatrix}, \quad B := \begin{bmatrix} \hat{\Sigma}_{xx} & O \\ O & \hat{\Sigma}_{yy} \end{bmatrix}.$$

Changing the structure of the two matrices, C and B, different methods result. For example, if we set $C = \hat{\Sigma}_{xx}$ and $B = I$, we get the eigenvalue-eigenvector task of PCA.

In [178], algorithmic procedures for the solution of the related equations, in a numerically robust way, are discussed.

Partial least-squares

Partial least-squares (PLS) method was first introduced in [175] and it has been used extensively in a number of applications, such as chemometrics, bioinformatics, food research, medicine, pharmacology, social sciences, and physiology, to name but a few. The corresponding eigenanalysis problem results if we set in Eq. (19.39)

$$B = \begin{bmatrix} I & O \\ O & I \end{bmatrix},$$

and keep C the same as for CCA. This eigenvalue-eigenvector problem arises (try it) if instead of maximizing the correlation coefficient ρ in Eq. (19.25), one maximizes the covariance, that is,

$$\text{cov}\left(\mathrm{z}_{x,1}, \mathrm{z}_{y,1}\right) = \mathbb{E}[\mathrm{z}_{x,1}\mathrm{z}_{y,1}]. \tag{19.40}$$

This means that while trying to reduce the dimensionality, our concern not only focuses on the correlation but *at the same* we want to identify directions that *also* care for maximum variance for both sets of variables. The optimizing task for identifying the first pair of axes, $\boldsymbol{u}_{x,1}, \boldsymbol{u}_{y,1}$, now becomes

$$\begin{aligned} \text{maximize} \quad & \boldsymbol{u}_x^T \hat{\Sigma}_{xy} \boldsymbol{u}_y, && (19.41) \\ \text{s.t.} \quad & \boldsymbol{u}_x^T \boldsymbol{u}_x = 1, && (19.42) \\ & \boldsymbol{u}_y^T \boldsymbol{u}_y = 1. && (19.43) \end{aligned}$$

PLS has been used both for classification as well as for regression tasks. For example in Chapter 6, we used PCA for regression in order to reduce the dimensionality of the space and the LS solution was expressed in this lower dimensional space. However, the principle axes were determined only on the basis of the input data so as to retain maximum variance. In contrast, PLS can be employed by considering the output observations as the second set of variables, and one can select the axes so as to maximize the variances as well as the correlation between the two data sets. The latter can be understood from the fact that by maximizing the covariance (PLS) is equivalent to maximizing the product of the correlation coefficient (used for CCA) times the two variance terms.

The literature on PLS is extensive and the method has been studied both algorithmically and from its performance point of view. The interested reader can obtain more on PLS from [143]. In all the

techniques we have discussed so far, a major focus is on computing the eigenvalues-eigenvectors. To this end, although one can use general packages and algorithms, a number of more efficient alternatives have been derived. A common approach is to solve the task in a two-step iterative procedure. In the first step, the largest eigenvalue (eigenvector) is computed, for which there exist efficient algorithms, such as the power method (e.g., [77]). Then, a procedure known as *deflation* is adopted; this consists of removing from the covariance matrices the variance that has been explained with the features extracted from the first step; see, for example, [127]. Kernelized versions of PLS have also been proposed, for example, [9, 142].

Remarks 19.3.

- Another dimensionality reduction method results if we set in Eq. (19.39)

$$B = \begin{bmatrix} \hat{\Sigma}_{xx} & O \\ O & I \end{bmatrix}.$$

  The resulting method is known as *multivariate linear regression* (MLR). This is the task of finding a set of basis vectors and corresponding regressors such that the mean-square error in a regression problem is minimized, [24].
- CCA is *invariant* with respect to affine transformations. This is an important advantage with respect to the ordinary correlation analysis, for example, [6].
- Extensions of CCA and PLS to more than two data sets have also been proposed; see, for example, [52, 103, 174].

19.5 INDEPENDENT COMPONENT ANALYSIS

The latent variable interpretation of PCA was summarized in Eqs. (19.17)–(19.19), where each one of the observed random variables, x_i, is (approximately) written as a linear combination of the latent variables (principle components in this case), z_i, which are in turn obtained via Eq. (19.19), imposing the uncorrelatedness constraint.

The kick-off point for ICA is to assume that the following latent model is true, that is,

$$\mathbf{x} = A\mathbf{s}, \tag{19.44}$$

where the (unknown) latent variables of $\mathbf{s}$ are assumed to be mutually statistically *independent* and we refer to them as the *independent components* (ICs). The task then comprises obtaining estimates of both the matrix A as well as the independent components. We will focus on the case where A is an $l \times l$ square matrix. Extensions to fat and tall matrices, corresponding to scenarios where the number of latent variables, m, is smaller or larger than the number of the observed random variables, l, have also been considered and developed (see, e.g., [93]).

Matrix A is known as the *mixing matrix* and its elements, a_{ij}, as the *mixing coefficients*. The resulting estimates of the latent variables will be denoted as z_i, $i = 1, 2 \ldots, l$, and we will also refer to them as independent components. The observed random variables, x_i, $i = 1, 2, \ldots, l$, are sometimes called the *mixture variables* or simply mixtures.

To obtain the estimates of the latent variables, we adopt the model

$$\hat{\mathbf{s}} := \mathbf{z} = W\mathbf{x}, \tag{19.45}$$

where W is also known as the *unmixing* or *separating* matrix. Note that

$$\mathbf{z} = WA\mathbf{s},$$

and we have to estimate the unknown parameters, so that $\mathbf{z}$ is as close to $\mathbf{s}$, that is, to be independent. For square matrices, $A = W^{-1}$, assuming invertibility.

19.5.1 ICA AND GAUSSIANITY

Although in general in statistics adopting the Gaussian assumption for a pdf seems to be rather a "blessing," in the case of ICA this is not true any more. This can easily be understood if we look at the consequences of adopting the Gaussian assumption. If the independent components follow Gaussian distributions, their joint pdf is given by

$$p(\boldsymbol{s}) = \frac{1}{(2\pi)^{l/2}} \exp\left(-\frac{||\boldsymbol{s}||^2}{2}\right), \tag{19.46}$$

where for simplicity, we have assumed that all the variables are normalized to unit variance. Let the mixing matrix, A, be an orthogonal one, that is, $A^{-1} = A^T$. Then, the joint pdf of the mixtures is readily obtained as (see, Eq. (2.45))

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{l/2}} \exp\left(-\frac{||A^T\boldsymbol{x}||^2}{2}\right) |\det(A^T)|. \tag{19.47}$$

However due to the orthogonality of A, we have that $||A^T\boldsymbol{x}||^2 = ||\boldsymbol{x}||^2$ and $|\det(A^T)| = 1$, which makes $p(\boldsymbol{s})$ indistinguishable from $p(\boldsymbol{x})$. That is, no conclusion about A can be drawn by observing $\mathbf{x}$, as all related information has been lost. Seen from another point of view, the mixtures x_i are mutually uncorrelated, as $\Sigma_x = I$, and ICA can provide no further information. This is a direct consequence of the fact that uncorrelatedness for jointly Gaussian variables is equivalent to independence (see Section 2.3.2). In other words, if the latent variables are Gaussians, ICA cannot take us any further than PCA, because the latter provides uncorrelated components. That is, the mixing matrix, A, *is not identifiable* for Gaussian independent components. In a more general setting, in a case where some of the components are Gaussians and some are not, ICA can identify the non-Gaussian ones. Thus, for a matrix A to be identifiable, *at most one* of the independent components can be Gaussian.

From a mathematical point of view, the ICA task is ill-posed for Gaussian variables. Indeed, assume that a set of independent Gaussian components, $\mathbf{z}$, have been obtained; then, any linear transformation on $\mathbf{z}$ by a unitary matrix will also be a solution (as shown previously). Note that this problem is bypassed in PCA, because the latter imposes a specific structure on the transformation matrix.

In order to deal with independence one has to involve, in one way or another, higher order statistical information. Second-order statistical information suffices for imposing uncorrelatedness, as is the case with PCA, but it is not enough for ICA. To this end, a large number of techniques and algorithms have been developed over the years and reviewing all these techniques is far beyond the limits imposed on a book section. The goal here is to provide the reader with the essence behind these techniques and emphasize the need to bring higher order statistics into the game. The interested reader can delve deeper in this field from [51, 54, 75, 93, 113].

19.5.2 ICA AND HIGHER ORDER CUMULANTS

Imposing the constraint on the components of $\mathbf{z}$ to be independent is equivalent to demanding all higher order *cross-cumulants* (Appendix B.3) to be zero. One possibility to achieve this is to restrict ourselves up to the fourth-order cumulants [53]. As it is stated in Appendix B.3, the first three cumulants for zero-mean variables are equal to the corresponding moments, that is,

$$\kappa_1(\mathrm{z}_i) = \mathbb{E}[\mathrm{z}_i] = 0,$$

$$\kappa_2(\mathrm{z}_i, \mathrm{z}_j) = \mathbb{E}[\mathrm{z}_i \mathrm{z}_j],$$

$$\kappa_3(\mathrm{z}_i, \mathrm{z}_j, \mathrm{z}_k) = \mathbb{E}[\mathrm{z}_i \mathrm{z}_j \mathrm{z}_k],$$

and the fourth-order cumulants are given by

$$\begin{aligned}\kappa_4(\mathrm{z}_i, \mathrm{z}_j, \mathrm{z}_k, \mathrm{z}_r) = \mathbb{E}[\mathrm{z}_i \mathrm{z}_j \mathrm{z}_k \mathrm{z}_r] - \mathbb{E}[\mathrm{z}_i \mathrm{z}_j]\,\mathbb{E}[\mathrm{z}_k \mathrm{z}_r] \\ - \mathbb{E}[\mathrm{z}_i \mathrm{z}_k]\,\mathbb{E}[\mathrm{z}_j \mathrm{z}_r] - \mathbb{E}[\mathrm{z}_i \mathrm{z}_r]\,\mathbb{E}[\mathrm{z}_j \mathrm{z}_k].\end{aligned}$$

An assumption that is employed is that the involved pdfs are symmetric, which renders odd order cumulants to zero. Thus, we are left only with the second- and fourth-order cumulants. Under the previous assumptions, our goal is to estimate the unmixing matrix, W, so that (a) the second-order and (b) the fourth-order cumulants to become zero. This is achieved in two steps.

Step 1: Compute

$$\hat{\mathbf{z}} = U^T \mathbf{x}, \tag{19.48}$$

where U is the unitary $l \times l$ matrix associated with PCA. This transformation guarantees that the components of $\hat{\mathbf{z}}$ are uncorrelated, that is,

$$\mathbb{E}[\hat{\mathrm{z}}_i \hat{\mathrm{z}}_j] = 0, \quad i \neq j,\ i,j = 1, 2, \ldots, l.$$

Step 2: Compute a orthogonal matrix, $\hat{U}$, such that the fourth-order cross-cumulants of the components of the transformed random vector,

$$\mathbf{z} = \hat{U}^T \hat{\mathbf{z}}, \tag{19.49}$$

are zero. In order to achieve this, the following maximization task is solved:

$$\max_{\hat{U}\hat{U}^T = I} \sum_{i=1}^{l} \kappa_4^2(\mathrm{z}_i). \tag{19.50}$$

Step 2 is justified as follows. It can be shown [53] that, the sum of the squares of the fourth-order cumulants is invariant under a linear transformation by an orthogonal matrix. Therefore, as the sum of the squares of the fourth-order cumulants is fixed for $\mathbf{z}$, maximizing the sum of the squares of the autocumulants of $\mathbf{z}$ will force the corresponding cross-cumulants to zero. Observe that this is basically a diagonalization problem of the fourth-order cumulant multidimensional array. In practice, this can be achieved by generalizing the method of Givens rotations, used for matrix diagonalization, [53]. Note that the sum that is maximized is a function of (a) the elements of the unknown matrix $\hat{U}$, (b) the elements of the known (for this step) matrix U, and (c) the cumulants of the random components of the mixtures $\mathbf{x}$, which have to be estimated prior to the application of the method. In practice, it usually turns

out that setting the cross-cumulants to zero is only approximately achieved. This is because the model in Eq. (19.44) may not be exact, for example, due to the existence of noise. Also, the cumulants of the mixtures are only approximately known, because they are estimated by the available observations.

Once U and $\hat{U}$ have been computed, the unmixing matrix is readily available and we can write

$$\mathbf{z} = W\mathbf{x} = (U\hat{U})^T\mathbf{x},$$

and the mixing matrix is given as $A = W^{-1}$.

A number of algorithms have been developed around the idea of higher order cumulants, which are also known as *tensorial methods*. Tensors are generalizations of matrices and cumulant tensors are generalizations of the covariance matrix. Moreover, note that as the eigenanalysis of the covariance matrix leads to uncorrelated (principle) components, the eigenanalysis of the cumulant tensor leads to independent components. The interested reader can obtain a more detailed account of such techniques from [38, 53, 112].

ICA ambiguities

Any ICA method can (approximately) recover the independent components within the following two indeterminacies.

- Independent components (ICs) are recovered to within a constant factor. Indeed, if A and $\mathbf{z}$ are the recovered quantities by an ICA algorithm, then $(1/a)A$ and $a\mathbf{z}$ is also a solution, as is readily seen from Eq. (19.44). Thus, usually the recovered latent variables (ICs) are normalized to unit variance.
- We cannot determine the order of the ICs. Indeed, if A and $\mathbf{z}$ have been recovered and P is a permutation matrix, then AP^{-1} and $P\mathbf{z}$ is also a solution, because the components of $P\mathbf{z}$ are the same as those of $\mathbf{z}$ in a different order (with the same statistical properties).

19.5.3 NON-GAUSSIANITY AND INDEPENDENT COMPONENTS

The fourth-order (auto)cumulant, of a random variable, z,

$$\kappa_4(\mathrm{z}) = \mathbb{E}[\mathrm{z}^4] - 3\left(\mathbb{E}[\mathrm{z}^2]\right)^2,$$

is known as the *kurtosis* of the variable and it is a measure of *non-Gaussianity*. Variables following the Gaussian distribution have zero kurtosis. Sub-Gaussian variables (variables whose pdf falls at a slower rate than the Gaussian, for the same variance) have negative kurtosis. Super-Gaussian variables (corresponding to pdfs that fall at a faster rate than the Gaussian) have positive kurtosis. Thus, if we keep the variance fixed (e.g., for variables normalized to unit variance), maximizing the sum of squared kurtosis, it results in maximizing the nonGaussianity of the recovered ICs. Usually, the absolute value of the kurtosis of the recovered ICs is used as a measure of ranking them. This is important if ICA is used as a feature generation technique. Figure 19.8 shows some typical examples of a sub-Gaussian and a super-Gaussian together with the corresponding Gaussian distribution. Also, another typical example of a sub-Gaussian distribution is the uniform one.

Recall from Chapter 12 (Section 12.4.1) that the Gaussian distribution is the one that maximizes the entropy under the variance and mean constraints. In other words, it is the most random one, under these

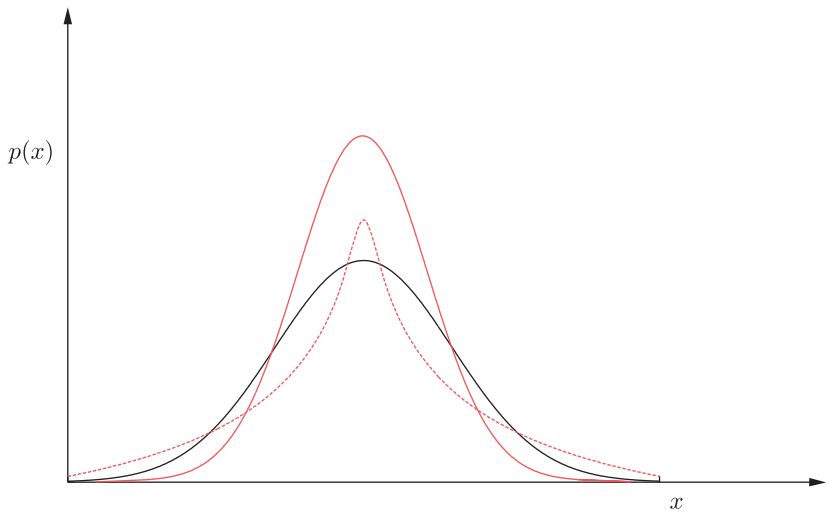

FIGURE 19.8

A Gaussian (full gray line) a super-Gaussian (dotted red line) and a sub-Gaussian (full red line).

constraints, and from this point of view the least informative with respect to the underlying structure of the data. In contrast, distributions that have the least resemblance to the Gaussian are more interesting as they are able to better unveil the structure associated with the data. This observation is at the heart of *projection pursuit*, which is closely related to the ICA family of techniques. The essence of these techniques is to search for directions in the feature space where the data projections are described in terms of non-Gaussian distributions [90, 99].

19.5.4 ICA BASED ON MUTUAL INFORMATION

The approach based on zeroing the second- and fourth-order cross-cumulants is not the only one. An alternative path is to estimate W by minimizing the *mutual information* among the latent variables. The notion of mutual information was introduced in Section 2.5. Elaborating a bit on Eq. (2.158) and performing the integrations on the right-hand side (for the case of more than two variables), it is readily shown that

$$I(\mathbf{z}) = -H(\mathbf{z}) + \sum_{i=1}^{l} H(\mathrm{z}_i), \tag{19.51}$$

where $H(\mathrm{z}_i)$ is the associated entropy of z_i, defined in Eq. (2.157). In Section 2.5 it has been shown that, $I(\mathbf{z})$ is equal to the Kullback-Leibler (KL) divergence between the joint pdf $p(\boldsymbol{z})$ and the product of the respective marginal probability densities, $\prod_{i=1}^{l} p_i(z_i)$. The KL divergence (and, hence, the associated mutual information $I(\mathbf{z})$) is a nonnegative quantity and it becomes zero if the components z_i are statistically independent. This is because only in this case the joint pdf becomes equal to the product of the corresponding marginal pdfs, leading the KL divergence to zero. Hence, the idea now becomes

to compute W so as to force $I(\mathbf{z})$ to be minimum, as this will make the components of $\mathbf{z}$ *as independent as possible*. Plugging Eq. (19.45) into Eq. (19.51) and taking into account the formula that relates the two pdfs associated with $\mathbf{x}$ and $\mathbf{z}$ (Eq. (2.45)), we end up with

$$I(\mathbf{z}) = -H(\mathbf{x}) - \ln|\det(W)| - \sum_{i=1}^{l} \int p_i(z_i) \ln p_i(z_i)\, dz_i. \tag{19.52}$$

The elements of the unknown matrix, W, are also hidden in the marginal pdfs of the latent variables, z_i. However, it is not easy to express this dependence explicitly. One possibility is to expand each one of the marginal densities around the Gaussian pdf, denoted here as $g(z)$, following Edgeworth's expansion (Appendix B), and truncate the series to a reasonable approximation. For example, keeping the first two terms in the Edgeworth expansion we have

$$p_i(z_i) = g(z_i)\left(1 + \frac{1}{3!}\kappa_3(z_i)H_3(z_i) + \frac{1}{4!}\kappa_4(z_i)H_4(z_i)\right), \tag{19.53}$$

where $H_k(z_i)$ is the Hermite polynomial of order k (Appendix B). To obtain an approximate expression for $I(\mathbf{z})$, in terms of cumulants of z_i and W, we can (a) insert in Eq. (19.52) the pdf approximation in Eq. (19.53), (b) adopt the approximation $\ln(1+y) \simeq y - y^2$, and (c) perform the integrations. This is no doubt a rather painful task! For the case of Eq. (19.53) and constraining W to be orthogonal the following is obtained (e.g., [93]):

$$I(\mathbf{z}) \approx C - \sum_{i=1}^{l}\left(\frac{1}{12}\kappa_3^2(z_i) + \frac{1}{48}\kappa_4^2(z_i) + \frac{7}{48}\kappa_4^4(z_i) - \frac{1}{8}\kappa_3^2(z_i)\kappa_4(z_i)\right), \tag{19.54}$$

where C in a quantity independent of W. Under the assumption that the pdfs are symmetric (thus, third-order cumulants are zero), it can be shown that minimizing the approximate expression of the mutual information in Eq. (19.54) is equivalent to maximizing the sum of the squares of the fourth-order cumulants. Note that the orthogonal W constraint is not necessary, and if it is not adopted other approximate expressions for $I(\mathbf{z})$ result, for example, [84].

Minimization of $I(\mathbf{z})$ in Eq. (19.54) can be carried out by a gradient descent technique (Chapter 5), where the involved expectations (associated with the cumulants) are replaced by the respective instantaneous values. Although we will not treat the derivation of algorithmic schemes in detail, in order to get a flavor of the involved tricks, let us go back to Eq. (19.52), before we apply the approximations. Because $H(\mathbf{x})$ does not depend on W, minimizing $I(\mathbf{z})$ is equivalent with the maximization of

$$J(W) = \ln|\det(W)| + \mathbb{E}\left[\sum_{i=1}^{l} \ln p_i(\mathrm{z}_i)\right]. \tag{19.55}$$

Taking the gradient of the cost function with respect to W results in

$$\frac{\partial J(W)}{\partial W} = W^{-T} - \mathbb{E}[\boldsymbol{\phi}(\mathbf{z})\mathbf{x}^T], \tag{19.56}$$

where

$$\boldsymbol{\phi}(\mathbf{z}) := \left[-\frac{p_1'(\mathrm{z}_1)}{p_1(\mathrm{z}_1)}, \ldots, -\frac{p_l'(\mathrm{z}_l)}{p_l(\mathrm{z}_l)}\right]^T, \tag{19.57}$$

and

$$p_i'(z_i) := \frac{dp_i(z_i)}{dz_i}, \tag{19.58}$$

and we used the formula

$$\frac{\partial \det(W)}{\partial W} = W^{-T}\det(W).$$

Obviously, the derivatives of the marginal probability densities depend on the type of approximation adopted in each case. The general gradient ascent scheme at the ith iteration step can now be written as

$$W^{(i)} = W^{(i-1)} + \mu_i \left((W^{(i-1)})^{-T} - \mathbb{E}\left[\boldsymbol{\phi}(\mathbf{z})\mathbf{x}^T \right] \right),$$

or

$$W^{(i)} = W^{(i-1)} + \mu_i \left(I - \mathbb{E}\left[\boldsymbol{\phi}(\mathbf{z})\mathbf{z}^T \right] \right) (W^{(i-1)})^{-T}. \tag{19.59}$$

In practice, the expectation operator is neglected and random variables are replaced by respective observations, in the spirit of the stochastic approximation rationale (Chapter 5).

The update equation in Eq. (19.59) involves the inversion of the transpose of the current estimate of W. Besides the computational complexity issues, there is no guarantee of the invertibility in the process of adaptation. The use of the so called *natural gradient* [63], instead of the gradient in Eq. (19.56), results in

$$W^{(i)} = W^{(i-1)} + \mu_i \left(I - \mathbb{E}[\boldsymbol{\phi}(\mathbf{z})\mathbf{z}^T] \right) W^{(i-1)}, \tag{19.60}$$

which does not involve matrix inversion and at the same time improves convergence. A more detailed treatment of this issue is beyond the scope of this book. Just to give an incentive to the mathematically inclined reader for indulging more deeply this field, it suffices to say that our familiar gradient, that is, Eq. (19.56), points to the steepest ascent direction if the space is Euclidean. However, in our case the parameter space consists of all the nonsingular $l \times l$ matrices, which is a multiplicative group. The space is Riemannian and it turns out that the natural gradient, pointing to the steepest ascent direction, results if we multiply the gradient in Eq. (19.56) by $W^T W$, which is the corresponding Riemannian metric tensor [63].

Remarks 19.4.

- From the gradient in Eq. (19.56), it is easy to see that at a stationary point the following is true:

$$\frac{\partial J(W)}{\partial W} W^T = \mathbb{E}[I - \boldsymbol{\phi}(\mathbf{z})\mathbf{z}^T] = 0. \tag{19.61}$$

  In other words, what we achieve with ICA is a *nonlinear generalization* of PCA. Recall that for the latter, the uncorrelatedness condition can be written as

$$\mathbb{E}[I - \mathbf{z}\mathbf{z}^T] = 0. \tag{19.62}$$

  The presence of the *nonlinear* function, $\boldsymbol{\phi}$, takes us beyond simple uncorrelatedness, and brings the cumulants into the scene. As a matter of fact, Eq. (19.61) was the one that inspired the early pioneering work on ICA, as a direct nonlinear generalization of PCA, [86, 100].

- The origins of ICA are traced back to the seminal paper [86]. For a number of years, it remained an activity pretty much within the French signal processing and statistics communities. Two papers were catalytic for its widespread use and popularity, namely [17] in the mid-nineties and the development of the FastICA,[4] [92], which allowed for efficient implementations; see [101] for a related review.
- In machine learning, the use of ICA as a feature generation technique is justified by the following argument. In [16], it is suggested that the outcome of the early processing performed by the visual cortical feature detectors might be the result of a *redundancy reduction* process. Thus, searching for independent features, conditioned on the input data, is in line with such a claim; see, for example, [70, 107] and the references therein.
- Although we have focused on the noiseless case, extensions of ICA to noisy tasks have also been proposed (see, e.g., [93]). For an extension of ICA in the complex-valued case, see [2]. Nonlinear extensions have also been considered, including kernelized ICA versions, for example, [13].
- In [3], the treatment of ICA also involves random processes and a wider class of signals, including Gaussians, can be identified.
- In [7], the multiset ICA framework of *independent vector analysis* (IVA) is discussed. It is shown that it generalizes the multiset CCA, if higher-order, besides second-order statistics, are taken into account.

19.5.5 ALTERNATIVE PATHS TO ICA

Besides the previously discussed two paths to ICA, a number of alternatives have been suggested, shedding light on different aspects of the problem. Some notable directions are

- *Infomax Principle*: This method assumes that the latent variables are the outputs of a nonlinear system (neural network, Chapter 18) of the form

$$z_i = \phi_i(\boldsymbol{w}_i^T \mathbf{x}) + \eta, \quad i = 1, 2, \ldots, l,$$

  where ϕ_i are nonlinear functions and η additive Gaussian noise. The weight vectors, $\boldsymbol{w}_i$, are computed so as to maximize the entropy of the outputs; the reasoning is based on some information theoretic arguments concerning the information flow in the network, [17].
- *Maximum Likelihood*: Starting from Eq. (19.44), the pdf of the observed variables is expressed in terms of the pdfs of the independent components

$$p(\boldsymbol{x}) = |\det(W)| \prod_{i=1}^{l} p_i(\boldsymbol{w}^T x_i),$$

  where we used

$$W := A^{-1}.$$

  Assuming that we have N observations, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, and taking the logarithm of the joint $p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, one can maximize the log-likelihood with respect to the W. It is straightforward to derive the log-likelihood function and to observe that it is very similar to $J(W)$ given in Eq. (19.55). The p_i's are chosen so as to belong to families of non-Gaussians, for example, [93]. A connection between the infomax approach and the maximum likelihood one has been established in [36, 37].

[4] http://research.ics.aalto.fi/ica/fastica/index.shtml.

- *Negentropy*: According to this method, the starting point is to maximize the non-Gaussianity, which is now measured in terms of the *negentropy*, defined as

$$J(\mathbf{z}) := H(\mathbf{z}_{\text{gauss}}) - H(\mathbf{z}),$$

  where $\mathbf{z}_{\text{gauss}}$ corresponds to Gaussian distributed variables of the same covariance matrix, which we know corresponds to the maximum entropy, H. Thus, maximizing the negentropy, which is a nonnegative function, is equivalent to making the latent variables as less Gaussian as possible. Usually, approximations of the negentropy are employed, which are expressed in terms of higher order cumulants, or by matching the nonlinearity to source distribution, [93, 132].
- If the unmixing matrix is constrained to be orthogonal, the negentropy and the maximum likelihood approaches become equivalent, [2].

The cocktail party problem

A classical application that demonstrates the power of the ICA is the so-called *cocktail party problem*. In a party, there are various people speaking; in our case, we are going to consider music as well. Let us say that there are people (a female and a male) and there is also monophonic music, making three sources of sound in total. Then, three microphones (as many as the sources) are placed in different places in the room and the mixed speech signals are recorded. We denote the inputs to the three microphones as $x_1(t)$, $x_2(t)$, $x_3(t)$, respectively. In the simplest of the models, the three recorded signals can be considered as linear combinations of the individual source signals. Delays are not considered. The goal is to use ICA and recover the original speech and music from the recorded mixed signals.

To this end and in order to bring the task in the formulation we have previously adopted, we consider the values of the three signals at different time instants as different observations of the corresponding random variables, x_1, x_2, x_3, which are put together to form the random vector $\mathbf{x}$. We further adopt the very reasonable assumption that the original source signals, denoted as $s_1(t)$, $s_2(t)$, $s_3(t)$, are independent and (similarly as before) the values at different time instants correspond to the values of three latent variables, denoted together as a random vector $\mathbf{s}$.

We are ready now to apply ICA to compute the unmixing matrix W, from which we can obtain the estimates of the ICs corresponding to the observations received by the three microphones,

$$\boldsymbol{z}(t) = [z_1(t), z_2(t), z_3(t)]^T = W[x_1(t), x_2(t), x_3(t)]^T.$$

Figure 19.9a shows the three different signals, which are linearly combined (by a set of mixing coefficients defining a mixing matrix A) to form the three "microphone signals." Figure 19.9b shows the resulting signals, which are then used as described before for the ICA analysis. Figure 19.9c shows the recovered original signals, as the corresponding ICs. The FastICA algorithm was employed.[5] Figure 19.10 is the result when PCA is used and the original signals are obtained via the (three) principle components.

One can observe that ICA manages to separate the signals with very good accuracy, whereas PCA fails. The reader can also listen to the signals by downloading the corresponding ".wav" files from the site of this book.

Note that the cocktail party problem is representative of a large class of tasks where a number of recorder signals result as linear combinations of other independent signals; the goal is the recovery of the latter. A notable application of this kind is found in electroencephalogram (EEG). The EEG data

[5] http://research.ics.aalto.fi/ica/fastica/.

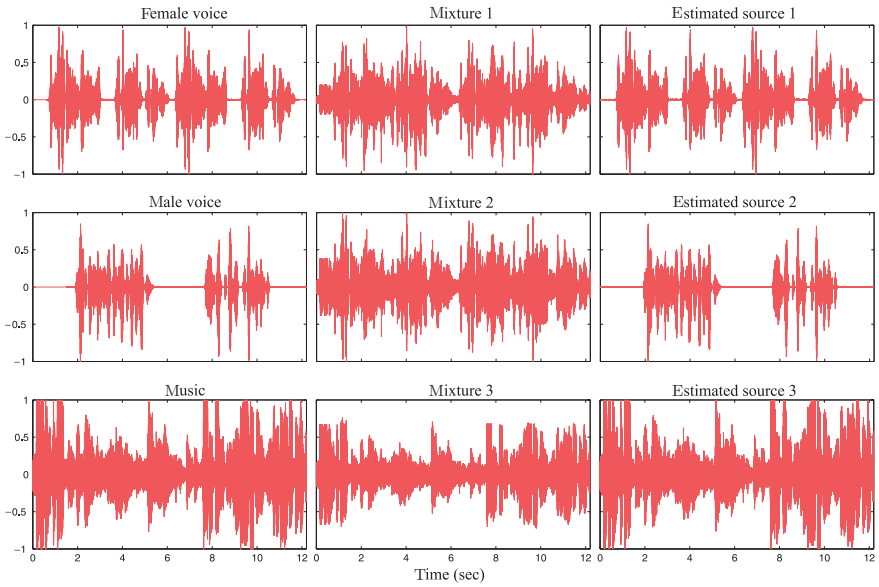

FIGURE 19.9

ICA source separation in the cocktail party setting.

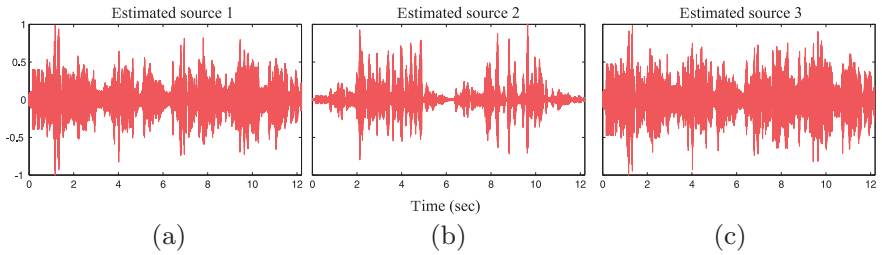

FIGURE 19.10

PCA source separation in the cocktail party setting.

consists of electrical potentials recorded at different locations on the scalp (or more recently in the ear, [105]), which are generated by the combination of different underlying components of brain and muscle activity. The task is to use ICA to recover the components, which in turn can unveil useful information about the brain activity, for example, [148].

The cocktail party problem is a typical example of a more general class of tasks known as *blind source separation* (BSS). The goal in these tasks is to estimate the "causes" (sources, original signals) based only on information residing in the observations, without any other extra information, and this is the reason that the word "blind" is used. Viewed in another way, BSS is an example of unsupervised learning. ICA is, probably, the most widely used technique for such problems.

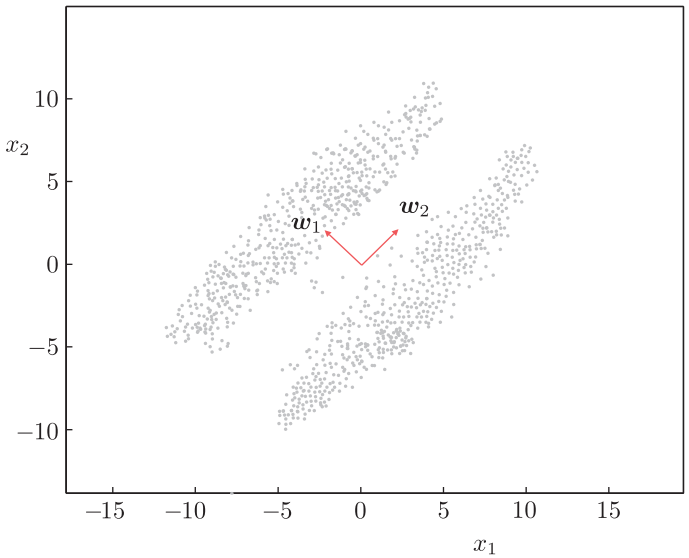

FIGURE 19.11

The setup for the ICA simulation example. The two vectors point to the projection directions resulting from the analysis. The optimal direction for projection, resulting from the ICA analysis, is that of w_2.

Example 19.4. The goal of this example is to demonstrate the power of ICA as a feature generation technique, where the most informative of the generated features are to be kept.

The example is a realization of the case shown in Figure 19.11. A number of 1024 samples of a two-dimensional normal distribution was generated.

The mean and covariance matrix of the normal pdf were

$$\boldsymbol{\mu} = [-2.6042, 2.5]^T, \quad \Sigma = \begin{bmatrix} 10.5246 & 9.6313 \\ 9.6313 & 11.3203 \end{bmatrix}$$

Similarly, 1024 samples from a second normal pdf were generated with the same covariance matrix and mean $-\boldsymbol{\mu}$. For the ICA, the method based on the second- and fourth-order cumulants, presented in this section, was used. The resulting transformation matrix W is

$$W = \begin{bmatrix} -0.7088 & 0.7054 \\ 0.7054 & 0.7088 \end{bmatrix} := \begin{bmatrix} \boldsymbol{w}_1^T \\ \boldsymbol{w}_2^T \end{bmatrix}$$

The vectors $\boldsymbol{w}_2$ and $\boldsymbol{w}_1$ point in the principle and minor axis directions, respectively, obtained from the PCA analysis. According to PCA, the most informative direction is along the principle axis $\boldsymbol{w}_2$, which is the one with maximum variance. However, the most interesting direction for projection, according to the ICA analysis, is that of $\boldsymbol{w}_1$. Indeed, the kurtosis of the obtained ICs z_1, z_2, along these directions are

$$\kappa_4(z_1) = -1.7,$$
$$\kappa_4(z_2) = 0.1,$$

respectively. Thus, projection in the principle (PCA) axis direction results in a variable with a pdf close to a Gaussian. The projection on the minor axis direction results in a variable with a pdf that deviates from the Gaussian (it is bimodal) and it is the more interesting one from the classification point of view. This can be easily verified by looking at the figure; projecting on the direction $\boldsymbol{w}_2$ leads to class overlapping.

19.6 DICTIONARY LEARNING: THE *k*-SVD ALGORITHM

The concept of *overcomplete dictionaries* and their importance in modeling real-world signals has been introduced in Chapter 9. We return to this topic, this time in a more general setting. There, the dictionary was assumed known with preselected atoms. In this section, the blind version of this task is considered; that is, the atoms of the dictionary are unknown and have to be estimated from the observed data. Recall that, this was the case with ICA; however, instead of the independence concept, used for ICA, sparsity arguments will be mobilized here. Giving the freedom to the dictionary to adapt to the needs of the specific, each time, input can lead to enhanced performance compared to dictionaries with preselected atoms.

Our starting point is that the observed l random variables are expressed in terms of $m > l$ latent ones according to the linear model

$$\mathbf{x} = A\mathbf{z},\ \mathbf{x} \in \mathbb{R}^l, \quad \mathbf{z} \in \mathbb{R}^m, \tag{19.63}$$

and A is an unknown $l \times m$ matrix. Usually, $m \gg l$. Even if A were known and fixed, it does not need special mathematical skills to see that this task has not a single solution and one has to embed constraints into the problem. To this end, we are going to adopt sparsity-promoting constraints, as we have already discussed in various parts in this book.

Let $\boldsymbol{x}_n,\ n = 1, 2, \ldots, N$, be the observations that will constitute the only available information. The task is to obtain the atoms (columns of A) of the dictionary as well as the latent variables that are assumed to be sparse; that is, we are going to establish a sparse representation of our input observations (vectors). No doubt, there are different paths to achieve the goal. We are going to focus on one of the most widely known and used methods, known as *k-SVD*, proposed in [4].

Let $X := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$, $A := \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m\}$, and $Z := [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N]$, where $\boldsymbol{z}_n$ is the latent vector corresponding to the input $\boldsymbol{x}_n,\ n = 1, 2, \ldots, N$. The dictionary learning (DL) task is cast as the following optimization problem

$$\text{minimize with respect to } A, Z \quad ||X - AZ||_F^2, \tag{19.64}$$

$$\text{subject to} \quad ||\boldsymbol{z}_n||_0 \leq T_0,\ n = 1, 2, \ldots, N, \tag{19.65}$$

where T_0 is a threshold value. This is a nonconvex optimization task, and it is performed iteratively; each iteration comprises two stages. In the first one, A is assumed to be fixed and optimization is carried out with respect to $\boldsymbol{z}_n,\ n = 1, 2, \ldots, N$. In the second stage, the latent vectors are assumed fixed and optimization is carried out with respect to the columns of A.

In k-SVD, a slightly different rationale is adopted. While optimizing with respect to the columns of A, one at a time, an update of some of the elements of Z is also performed. This is a crucial difference of

the k-SVD, compared to the more standard optimization techniques, which appears to lead to improved performance in practice.

Stage 1: Assume A to be known and fixed to the value obtained from the previous iteration. Then, the associated optimization task becomes

$$\begin{aligned} \min_{Z} \quad & ||X - AZ||_F^2, \\ \text{s.t.} \quad & ||\boldsymbol{z}_n||_0 \leq T_0, \quad n = 1, 2, \ldots, N, \end{aligned}$$

which, due to the definition of the Frobenius norm, is equivalent to solving N distinct optimization tasks,

$$\min_{\boldsymbol{z}_n} \quad ||\boldsymbol{x}_n - A\boldsymbol{z}_n||^2, \tag{19.66}$$

$$\text{s.t.} \quad ||\boldsymbol{z}_n||_0 \leq T_0, \quad n = 1, 2, \ldots, N. \tag{19.67}$$

A similar objective is met if the following optimization tasks are considered instead,

$$\begin{aligned} \min_{\boldsymbol{z}_n} \quad & ||\boldsymbol{z}_n||_0, \\ \text{s.t.} \quad & ||\boldsymbol{x}_n - A\boldsymbol{z}_n||^2 < \epsilon, \quad n = 1, 2, \ldots, N, \end{aligned}$$

where ϵ is a constant acting as an upper bound of the error.

The task in Eqs. (19.66)–(19.67) can be solved by any one of the ℓ_0 minimization solvers, which have been considered in Chapter 10, for example, the OMP. This stage is known as *sparse coding*.

Stage 2: This stage is known as the *codebook update*. Having obtained $\boldsymbol{z}_n$, $n = 1, 2, \ldots, N$, (for fixed A) from stage 1, the goal now is to optimize with respect to the columns of A. This is achieved on a *column-by-column* basis. Assume that we currently consider the update of $\boldsymbol{a}_k$; this is carried out so as to minimize the (squared) Frobenius norm, $||X - AZ||_F^2$. To this end, we can write the product AZ as a sum of rank-one matrices, that is,

$$AZ = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m][\boldsymbol{z}_1^r, \ldots, \boldsymbol{z}_m^r]^T = \sum_{i=1}^{m} \boldsymbol{a}_i \boldsymbol{z}_i^{rT}, \tag{19.68}$$

where $\boldsymbol{z}_i^{rT}$, $i = 1, 2, \ldots, m$, are the *rows* of Z. Note that in the above sum, the vectors for indices, $i = 1, 2, \ldots, k-1$, are fixed to their recently updated values during this second stage of the current iteration step, while and vectors corresponding to $i = k+1, \ldots, m$, are fixed to the values that are available from the previous iteration step. This strategy allows for the use of the most recent updated information. We will now minimize with respect to the rank-one outer product matrix, $\boldsymbol{a}_k \boldsymbol{z}_k^{rT}$. Observe that this product, besides the kth column of A, also involves the kth row of Z; both of them will be updated. The rank-one matrix is estimated so as to minimize

$$||E_k - \boldsymbol{a}_k \boldsymbol{z}_k^{rT}||_F^2, \tag{19.69}$$

where,

$$E_k := X - \sum_{i=1, i \neq k}^{m} \boldsymbol{a}_i \boldsymbol{z}_i^{rT}.$$

In other words, we seek to find the best, in the Frobenius sense, rank-one approximation of E_k. Recall from Chapter 6 (Section 6.4) that the solution is given via the SVD of E_k. However, if we do that, there

is no guarantee that whatever sparse structure has been embedded in $\boldsymbol{z}_k^r$, from the update in stage 1, will be retained. According to the k-SVD, this is bypassed by focusing on the active set, that is, involving only the nonzero of its coefficients. Thus, we first search for the locations of the nonzero coefficients in $\boldsymbol{z}_k^r$ and let

$$\omega_k := \left\{ j_k, 1 \leq j_k \leq N : \ z_k^r(j_k) \neq 0 \right\}.$$

Then, we form the reduced vector $\tilde{\boldsymbol{z}}_k^r \in \mathbb{R}^{|\omega_k|}$, where $|\omega_k|$ denotes the cardinality of ω_k, which contains only the nonzero elements of $\boldsymbol{z}_k^r$. A little thought reveals that when writing $X = AZ$, the column of current interest, $\boldsymbol{a}_k$, contributes (as part of the corresponding linear combination) only to the columns $\boldsymbol{x}_{j_k}$, $j_k \in \omega_k$, of X. We then collect the corresponding columns of E_k to construct a reduced order matrix, $\tilde{E}_k$, which comprises the columns that are associated with the locations of the nonzero elements of $\boldsymbol{z}_k^r$, and select $\boldsymbol{a}_k \tilde{\boldsymbol{z}}_k^{rT}$ so that to minimize

$$||\tilde{E}_k - \boldsymbol{a}_k \tilde{\boldsymbol{z}}_k^{rT}||_F^2. \tag{19.70}$$

Performing SVD, $\tilde{E}_k = UDV^T$, $\boldsymbol{a}_k$ is set equal to $\boldsymbol{u}_1$ corresponding to the largest of the singular values and $\tilde{\boldsymbol{z}}_k^r = D(1,1)\boldsymbol{v}_1$. Thus, the atoms of the dictionary are obtained in normalized form (recall from the theory of SVD, that $||\boldsymbol{u}_1|| = 1$). In the sequel, the updated values obtained for $\tilde{\boldsymbol{z}}_k^r$ are placed in the corresponding locations in $\boldsymbol{z}_k^r$. The latter now has at least as many zeros as it had before, as some of the elements in $\boldsymbol{v}_1$ may be zeros. Simple arguments (Problem 19.3) show that at each iteration the error decreases and the algorithm converges to a local minimum. The success of the algorithm depends on the ability of the greedy algorithm to provide a sparse solution during the first stage. As we know from Chapter 10, greedy algorithms work well for sparsity levels, T_0, small enough compared to l.

In summary, each iteration step of the k-SVD algorithm comprises the following computation steps.

- Initialize $A^{(0)}$ with columns normalized to unit ℓ_2 norm.
- Set $i = 1$.
- *Stage 1*: Solve the optimization task in Eqs. (19.66)–(19.67) to obtain the sparse coding representation vectors, $\boldsymbol{z}_n$, $n = 1, 2, \ldots, N$; use any algorithm developed for this task.
- *Stage 2*: For any column, $k = 1, 2, \ldots, m$, in $A^{(i-1)}$, update it according to the following:
  - Identify the locations of the nonzero elements in the kth row of the computed, from stage 1, matrix Z.
  - Select the columns in X, which correspond to the locations of the nonzero elements of the kth row of Z and form a reduced order error matrix, $\tilde{E}_k$.
  - Perform SVD on $\tilde{E}_k$: $\tilde{E}_k = UDV^T$.
  - Update the kth column of $A^{(i)}$ to be the eigenvector corresponding to the largest singular value, $\boldsymbol{a}_k^{(i)} = \boldsymbol{u}_1$.
  - Update Z, by embedding in the nonzero locations of its kth row, the values $D(1,1)\boldsymbol{v}_1^T$.
- Stop if a convergence criterion is met.
- If not, $i = i + 1$, and continue.

Why the name k-SVD

The SVD part of the name is pretty obvious. However, the reader may wonder about the presence of "k" in front. As stated in [4], the algorithm can be considered a generalization of the k-means algorithm,

introduced in Chapter 12 (Algorithm 12.1). There, we can consider the mean values, which represent each cluster, as the code words (atoms) of a dictionary. During the first stage of the k-means learning, given the representatives of each cluster, a sparse coding scheme is performed; that is, each input vector is assigned to a single cluster. Thus, we can think of the k-means clustering as a sparse coding scheme, that associates a latent vector with each one the observations. Note that each of the latent vectors has only one nonzero element, pointing to the cluster where the respective input vector is assigned, according to the smallest Euclidean distance from all cluster representatives. This is a major difference with the k-SVD dictionary learning, during which each observation vector can be associated with more than one atoms; hence, the sparsity level of the corresponding latent vector can be larger than one. Furthermore, based on the assignment of the input vectors to the clusters, in the second stage of the k-means algorithm, an update of the cluster representatives is performed, and for each representative only the input vectors assigned to it are used. This is also similar in spirit to what happens in the second stage of the k-SVD. The difference is that each input observation may be associated with more than one atom. As it is pointed out in [4], if one sets $T_0 = 1$, the k-means algorithm can result from the k-SVD.

Remarks 19.5.

- Alternative to k-SVD paths to dictionary learning have also been suggested. For example, in [68] a DL technique referred to as method of optimal directions (MOD) was proposed, which differs from k-SVD in the dictionary update step. In particular, the full dictionary is updated via direct minimization of the Frobenious norm. Moreover, in [185] a majorization approach is followed, which allows the incorporation of more general sparsity constraints. On the other hand, in [119, 134] probabilistic arguments are employed, using a Laplacian prior to enforce sparsity. We know from Chapter 13 (Section 13.5) that in this case, the involved integrations are not analytically tractable and the different methods differ in the different approximations used to bypass this obstacle. In the former, the maximum value of the integrand is used and in the latter a Gaussian approximation of the posterior is adopted in order to handle the integration. In [76], variational bound techniques are mobilized, see Section 13.9.
- The method proposed in [118] bears some similarities to the k-SVD, because it also revolves around the SVD, but the dictionary is constrained to be a union of orthonormal bases. This can lead to some computational advantages; on the other hand, k-SVD puts no constraints on the atoms of the dictionary, which gives more freedom in modeling the input. Another difference lies in the column-by-column update introduced in k-SVD.
- A more detailed comparative study of k-SVD with other methods is given in [4].
- Dictionary learning is essentially a matrix factorization problem where a certain type of constraint is imposed on the right matrix factor. This approach can be considered to be just a manifestation of a wider class of constrained matrix factorization methods that allow several types of constraints to hold. Such techniques include the regularized PCA where functional and/or sparsity constraints are imposed to the left and to the right factors, [12, 179, 189], as well as the structured sparse matrix factorization in [28] together with its online counterpart [131].

Example 19.5. The goal of this example is to show the performance of the DL technique in the context of the image denoising task. In the case study of Section 9.10, image denoising, based on a predetermined and fixed DCT dictionary, was considered. Here, the k-SVD will be employed in order to learn the dictionary using information of the image *itself*. The two (256×256) images, without and

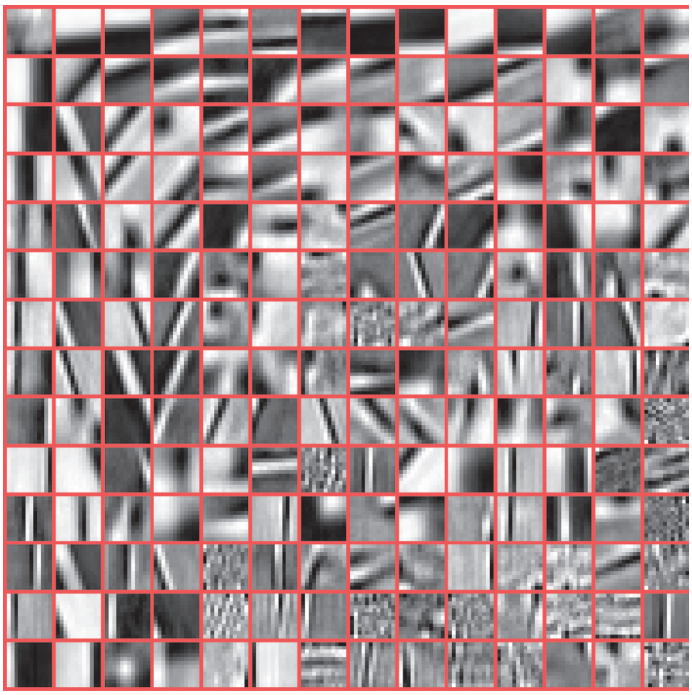

FIGURE 19.12

Dictionary resulting from *k*-SVD.

with noise corresponding to PSNR = 22, are shown in Figures 19.13a,b, respectively. The noisy image is divided in overlapped patches of size 12×12 (144), resulting in $(256 - 12 + 1)^2 = 60,025$ patches in total; these will constitute the training data set used for the learning of the dictionary. Specifically, the patches are sequentially extracted from the noisy image, then vectorized in lexicographic order, and used as columns, one after the other to define the $(144 \times 60,025)$ matrix X. Then, *k*-SVD is mobilized in order to train an overcomplete dictionary of size 144×196. The resulting atoms, reshaped in order to form 12×12 pixel patches, are shown in Figure 19.12. Compare the atoms of this dictionary with atoms of the fixed DCT dictionary of Figure 9.14.

Next, we follow the same procedure as in Section 9.10, by replacing the DCT dictionary with the one obtained by the *k*-SVD method. The resulting denoised image is shown in Figure 19.13c. Note that, although the dictionary was trained based on *the noisy data*, it led to about 2dB PSNR improvement over the fixed-dictionary case. As a matter of fact, because the number of patches is large and each one of them is carrying a different noise realization, the noise, during the dictionary learning stage, is averaged out leading to nearly noise-free dictionary atoms. More advanced use of dictionary learning techniques to further improve performance in tasks such as denoising and inpainting can be found in [66, 67, 130].

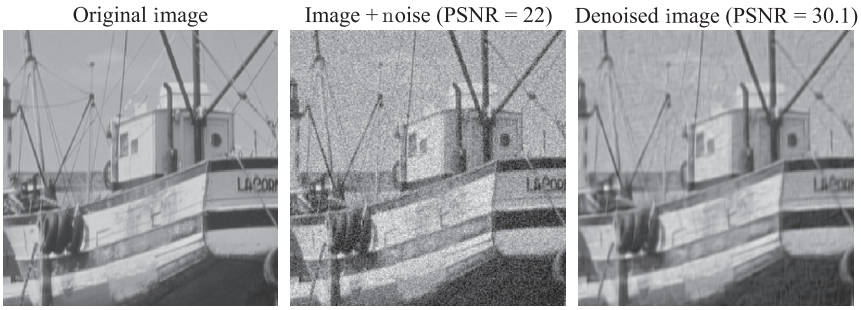

FIGURE 19.13

Image de-noising based on dictionary learning.

19.7 NONNEGATIVE MATRIX FACTORIZATION

The strong connection between dimensionality reduction and low-rank matrix factorization has already been stressed while discussing PCA. ICA can also be considered as a low-rank matrix factorization, if a smaller number, compared to the l observed random variables, of independent components is retained (e.g., selecting the $m < l$ least Gaussian ones).

An alternative to the previously discussed low-rank matrix factorization schemes was suggested in [135, 136], which guarantees the *nonnegativity* of the elements of the resulting matrix factors. Such a constraint is enforced in certain applications because negative elements contradict physical reality. For example, in image analysis, the intensity values of the pixels cannot be negative. Also, probability values cannot be negative. The resulting factorization is known as *nonnegative matrix factorization* (NMF) and it has been used successfully in a number of applications including document clustering [181], molecular pattern discovery [26], image analysis [115], clustering [161], music transcription and music instrument classification [19, 156], and face verification [187].

Given an $l \times N$ matrix X, the task of NMF consists of finding an approximate factorization of X, that is,

$$X \approx AZ, \tag{19.71}$$

where A and Z are $l \times m$ and $m \times N$ matrices, respectively, $m \leq \min(N, l)$ and all the matrix elements are nonnegative, that is, $A(i,k) \geq 0,\ Z(k,j) \geq 0,\ i = 1, 2, \ldots, l,\ k = 1, 2, \ldots, m,\ j = 1, 2, \ldots, N$. Clearly, if matrices A and Z are of low rank, their product is also a low rank, at most m, approximation of X. The significance of the above is that every column vector in X is represented by the expansion

$$\boldsymbol{x}_i \approx \sum_{k=1}^{m} Z(k,i)\boldsymbol{a}_k,\ i = 1, 2, \ldots, N,$$

where $\boldsymbol{a}_k,\ k = 1, 2, \ldots, m$, are the column vectors of A and constitute the basis of the expansion. The number of vectors in the basis is less than the dimensionality of the vector itself. Hence, NMF can also be seen as a method for *dimensionality reduction*.

To get a good approximation in Eq. (19.71) one can adopt different costs. The most common cost is the Frobenius norm of the error matrix. In such a setting, the NMF task is cast as follows:

$$\min_{A,Z} \; ||X - AZ||_F^2 := \sum_{i=1}^{l} \sum_{j=1}^{N} \big(X(i,j) - [AZ](i,j)\big)^2 \tag{19.72}$$

$$\text{s.t.} \;\; A(i,k) \geq 0, \; Z(k,j) \geq 0, \tag{19.73}$$

where $[AZ](i,j)$ is the (i,j) element of matrix AZ, and i, j, k run over all possible values. Besides the Frobenius norm, other costs have also been suggested (see, e.g., [158]).

Once the problem has been formulated, the major issue rests at the solution of the optimization task. To this end, a number of algorithms have been proposed, for example, Newton type or gradient descent type. Such algorithmic issues, as well as a number of related theoretic ones, are beyond the scope of this book and the interested reader may consult, for example, [50, 62, 168]. More recently, regularized versions, including sparsity-promoting regularizers, have been proposed; see, for example, [51] for a more recent review on the topic.

19.8 LEARNING LOW-DIMENSIONAL MODELS: A PROBABILISTIC PERSPECTIVE

In this section, the emphasis is to look at the dimensionality reduction task from a Bayesian perspective. Our focus will be more on presenting the main ideas and less on algorithmic procedures; the latter depend on the specific model and can be dug out from the palette of algorithms that have already been presented in Chapters 12 and 13. Our path to low-dimensional modeling traces its origin to the so-called *factor analysis*.

19.8.1 FACTOR ANALYSIS

Factor analysis was originally proposed in the work of Charles Spearman [159]. Charles Spearman (1863-1945) was an English psychologist who has made important contributions to statistics. Spearman was interested in human intelligence and developed the method in 1904, for analyzing multiple measures of cognitive performance. He argued that there exists a general intelligence factor (the so-called g-factor) that can be extracted by applying the factor analysis method on intelligence test data. However, this notion has been strongly disputed, as intelligence comprises a multiplicity of components (see, e.g., [78]).

Let $\mathbf{x} \in \mathbb{R}^l$. The factor analysis model assumes that there are $m < l$ underlying (latent) zero-mean variables or *factors* $\mathbf{z} \in \mathbb{R}^m$ so that

$$\mathrm{x}_i - \mu_i = \sum_{j=1}^{m} a_{ij} \mathrm{z}_j + \epsilon_i, \quad i = 1, 2, \ldots, l, \tag{19.74}$$

or

$$\mathbf{x} - \boldsymbol{\mu} = A\mathbf{z} + \boldsymbol{\epsilon}, \tag{19.75}$$

where $\boldsymbol{\mu}$ is the mean of $\mathbf{x}$ and $A \in \mathbb{R}^{l \times m}$ is formed by the weights a_{ij} known as *factor loadings*. The variables z_j, $j = 1, 2, \ldots, m$, are sometimes called *common factors*, because they contribute to

all observed variables, x_i, and ϵ_i are the *unique* or *specific factors*. As we have already done so far and without loss of generality, we will assume our data is centered, that is, $\boldsymbol{\mu} = \mathbf{0}$. In factor analysis, we assume ϵ_i to be of zero-mean and mutually uncorrelated, that is, $\Sigma_\epsilon = \mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T] := \text{diag}\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_l^2\}$. We also assume that $\mathbf{z}$ and $\boldsymbol{\epsilon}$ are independent. The m ($< l$) columns of A form a lower dimensional subspace, and $\boldsymbol{\epsilon}$ is that part of $\mathbf{x}$ not contained in this subspace. The first question that is now raised is whether the model in Eq. (19.75) is any different from our familiar regression task. The answer is in the affirmative. Note that here the matrix A is not known. All that we are given is the set of observations, $\boldsymbol{x}_n$, $n = 1, 2, \ldots, N$, and we have to obtain the subspace described by A. It is basically the same linear model that we have considered so far in this chapter, with the difference that now we have introduced the noise term. Once A is known, z_n can be obtained for each $\boldsymbol{x}_n$.

From Eq. (19.75), it is readily seen that

$$\Sigma_x = \mathbb{E}[\mathbf{x}\mathbf{x}^T] = A\,\mathbb{E}[\mathbf{z}\mathbf{z}^T]A^T + \Sigma_\epsilon.$$

We will further assume that $\mathbb{E}[\mathbf{z}\mathbf{z}^T] = I$; hence, we can write

$$\Sigma_x = AA^T + \Sigma_\epsilon. \tag{19.76}$$

Hence, A results as a factor of $(\Sigma_x - \Sigma_\epsilon)$. However, such a factorization, if it exists, is not unique. This can be easily checked out if we consider $\bar{A} = AU$, where U is an orthonormal matrix. Then, $\bar{A}\bar{A}^T = AA^T$. This has brought a lot of controversy around the factor analysis method when it comes to interpreting individual factors; see, for example, [43] for a discussion. To remedy this drawback, a number of authors have suggested methods and criteria that deal with the rotation (orthogonal or oblique) in order to gain improved interpretation of the factors [147]. However, from our perspective, where our goal is to express our problem in a lower dimensional space, this is not a problem. Any orthonormal matrix imposes a rotation within the subspace spanned by the columns of A; but we do not care about the exact choice of the coordinates, that is, the common factors.

There are different methods to obtain A (see, e.g., [59]). A popular one is to assume $p(\boldsymbol{x})$ to be Gaussian and employ the ML method to optimize with respect to the unknown parameters that define Σ_x in Eq. (19.76). Once A becomes available, one way to estimate the factors is to further assume that these can be expressed as linear combinations of the observations, that is,

$$\mathbf{z} = W\mathbf{x}.$$

Post multiplying by $\mathbf{x}$, taking expectations, recalling Eq. (19.75) and that $\mathbb{E}[\mathbf{z}\mathbf{z}^T] = I$, we get

$$\mathbb{E}[\mathbf{z}\mathbf{x}^T] = \mathbb{E}[\mathbf{z}\mathbf{z}^T A^T] + \mathbb{E}[\mathbf{z}\boldsymbol{\epsilon}^T] = A^T. \tag{19.77}$$

Also,

$$\mathbb{E}[\mathbf{z}\mathbf{x}^T] = W\,\mathbb{E}[\mathbf{x}\mathbf{x}^T] = W\Sigma_x. \tag{19.78}$$

Hence,

$$W = A^T\Sigma_x^{-1}.$$

Thus, given a value $\boldsymbol{x}$, the values of the corresponding latent variables are obtained by

$$\boldsymbol{z} = A^T\Sigma_x^{-1}\boldsymbol{x}. \tag{19.79}$$

19.8.2 PROBABILISTIC PCA

New light on this old problem was shed via the Bayesian rationale in the late nineties, [144, 165, 166]; the task was treated for the special case $\Sigma_\epsilon = \sigma^2 I$ and it was named *probabilistic PCA* (PPCA). The latent variables, $\mathbf{z}$, are dressed with a Gaussian prior,

$$p(z) = \mathcal{N}(z|\mathbf{0}, I),$$

which is in agreement with the earlier assumption $\mathbb{E}[\mathbf{z}\mathbf{z}^T] = I$, and the conditional pdf is chosen as

$$p(\boldsymbol{x}|z) = \mathcal{N}\left(\boldsymbol{x}|Az, \sigma^2 I\right),$$

where, for simplicity, we assume $\boldsymbol{\mu} = 0$ (otherwise the mean would be $Az + \boldsymbol{\mu}$). We are by now pretty familiar with writing down

$$p(z|\boldsymbol{x}) = \mathcal{N}\left(z|\boldsymbol{\mu}_{z|x}, \Sigma_{z|x}\right), \tag{19.80}$$

and

$$p(\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}|\mathbf{0}, \Sigma_x\right), \tag{19.81}$$

where (see Eqs. (12.10), (12.15) and (12.17), Chapter 12)

$$\Sigma_{z|x} = \left(I + \frac{1}{\sigma^2} A^T A\right)^{-1}, \tag{19.82}$$

$$\mu_{z|x} = \frac{1}{\sigma^2} \Sigma_{z|x} A^T \boldsymbol{x}, \tag{19.83}$$

$$\Sigma_x = \sigma^2 I + AA^T. \tag{19.84}$$

Note that using the Bayesian framework, the computation of the latent variables corresponding to a given set of observations, $\boldsymbol{x}$, can naturally be obtained via the posterior $p(z|\boldsymbol{x})$ in Eq. (19.80). For example, one can pick the respective mean value

$$z = \frac{1}{\sigma^2} \Sigma_{z|x} A^T \boldsymbol{x}. \tag{19.85}$$

Using the matrix inversion lemma (Problem 19.4), it turns out that Eqs. (19.79) and (19.85) are exactly the same; however, now, it comes as a natural consequence of our Bayesian assumptions.

One way to compute A is to apply the ML method on $\prod_{n=1}^{N} p(\boldsymbol{x}_n)$ and maximize with regard to A, σ^2 (and $\boldsymbol{\mu}$, if $\boldsymbol{\mu} \neq 0$). It turns out that the maximum likelihood solution for A is given by ([165]),

$$A_{\mathrm{ML}} = U_m \mathrm{diag}\left\{\lambda_1 - \sigma^2, \ldots, \lambda_m - \sigma^2\right\} R,$$

where U_m is the $l \times m$ matrix with columns the eigenvectors corresponding to the m largest eigenvalues, λ_i, $i = 1, 2, \ldots, m$, of the sample covariance matrix of $\mathbf{x}$, and R is an arbitrary orthogonal matrix ($RR^T = I$). Setting $R = I$, the columns of A are the (scaled) principle directions as computed by the classical PCA, discussed in Section 19.3. In any case, the columns of A span the principle subspace of the standard PCA. Note that as $\sigma^2 \to 0$, PPCA tends to PCA (Problem 19.5). Also, it turns out that

$$\sigma^2_{\mathrm{ML}} = \frac{1}{l - m} \sum_{i=m+1}^{l} \lambda_i. \tag{19.86}$$

The previously established connection with PCA does not come as a surprise. It has been well known for a long time (e.g., [5]) that if in the factor analysis model, one assumes $\Sigma_\epsilon = \sigma^2 I$, then at stationary points of the likelihood function the columns of A are scaled eigenvectors of the sample covariance matrix. Furthermore, σ^2 is the average of the discarded eigenvalues, as suggested in Eq. (19.86).

Another way to estimate A and σ^2 is via the EM algorithm [144, 165]. This is possible because we have $p(\boldsymbol{z}|\boldsymbol{x})$ in an analytic form. Given the set $(\boldsymbol{x}_n, \boldsymbol{z}_n)$, $n = 1, 2, \ldots, N$, of the observed and latent variables, the complete log-likelihood function is given by

$$\begin{aligned}\ln p(\mathcal{X}, \mathcal{Z}; A, \sigma^2) &= \sum_{n=1}^{N} \Big(\ln p(\boldsymbol{x}_n|\boldsymbol{z}_n; A, \sigma^2) + \ln p(\boldsymbol{z}_n) \Big) \\ &= -\sum_{n=1}^{N} \left(\frac{l}{2} \ln(2\pi) - \frac{l}{2} \ln \beta + \frac{\beta}{2} \|\boldsymbol{x}_n - A\boldsymbol{z}_n\|^2 \right. \\ &\quad \left. + \frac{m}{2} \ln(2\pi) + \frac{1}{2} \boldsymbol{z}_n^T \boldsymbol{z}_n \right),\end{aligned}$$

which is of the same form as the one given in Eq. (12.72). We have used $\beta = \frac{1}{\sigma^2}$. Thus, following similar steps as for Eq. (12.72) and rephrasing Eqs. (12.73)–(12.77) to our current notation, the E-step becomes

- E-step:

$$\begin{aligned}\mathcal{Q}(A, \beta; A^{(j)}, \beta^{(j)}) = -\sum_{n=1}^{N} \Big(&- \frac{l}{2} \ln \beta + \frac{1}{2} \big\| \boldsymbol{\mu}_{z|x}^{(j)}(n) \big\|^2 + \frac{1}{2} \text{trace} \left\{ \Sigma_{z|x}^{(j)} \right\} \\ &+ \frac{l}{2} \big\| \boldsymbol{x}_n - A \boldsymbol{\mu}_{z|x}^{(j)}(n) \big\|^2 + \frac{\beta}{2} \text{trace} \left\{ A \Sigma_{z|x}^{(j)} A^T \right\} \Big) + C\end{aligned}$$

  where C is a constant and

$$\boxed{\boldsymbol{\mu}_{z|x}^{(j)}(n) = \beta^{(j)} \Sigma_{z|x}^{(j)} A^{(j)T} \boldsymbol{x}_n, \;\; \Sigma_{z|x}^{(j)} = \left(I + \beta^{(j)} A^{(j)T} A^{(j)} \right)^{-1}.}$$

- M-step: Taking the derivatives with regard to β and A and equating to zero (Problem 19.6), we obtain

$$\boxed{A^{(j+1)} = \left(\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{\mu}_{z|x}^{(j)T}(n) \right) \left(N \Sigma_{z|x}^{(j)} + \sum_{n=1}^{N} \boldsymbol{\mu}_{z|x}^{(j)}(n) \boldsymbol{\mu}_{z|x}^{(j)T}(n) \right)^{-1},} \tag{19.87}$$

  and

$$\boxed{\beta^{(j+1)} = \frac{Nl}{\sum_{n=1}^{N} \left(\big\| \boldsymbol{x}_n - A^{(j+1)} \boldsymbol{\mu}_{z|x}^{(j)}(n) \big\|^2 + \text{trace} \left\{ A^{(j+1)} \Sigma_{z|x}^{(j)} A^{(j+1)T} \right\} \right)}.} \tag{19.88}$$

Observe that having adopted the EM algorithm, one does not need to compute the eigenvalues/eigenvectors of Σ_x. Even retrieving only the m principle components, the lowest cost one has to pay is $\mathcal{O}(ml^2)$ operations. Beyond that, $\mathcal{O}(Nl^2)$ are needed to compute Σ_x. For the EM approach, the covariance matrix need not be computed and the most demanding part comprises the matrix vector

products, which amount to $\mathcal{O}(Nml)$. Hence for $m \ll l$, computational savings are expected compared to the classical PCA. Keep in mind, though, that, the two methods optimize different criteria. PCA guarantees minimum least-squares error reconstruction, PPCA via the EM optimizes the likelihood. Thus, for applications where the error reconstruction is important, such as in compression, one has to be aware of this fact; see [165] for a related discussion.

The other alternative route to solve PPCA is by considering A and σ^2 as random variables with appropriate priors and apply the variational EM algorithm, see [23]. This has the added advantage that if one uses as the prior

$$p(A|\boldsymbol{\alpha}) = \prod_{k=1}^{m} \left(\frac{\alpha_k}{2\pi}\right)^{l/2} \exp\left(-\frac{\alpha_k}{2}\boldsymbol{a}_k^T\boldsymbol{a}_k\right),$$

with different precisions, $\alpha_k, k = 1, 2, \ldots, m$, per column, then using large enough m one can achieve pruning of the unnecessary components; this was discussed in Section 13.5. Hence, such an approach could provide the means of automatic determination of m. The interested reader, besides the references given before, can dig out useful related information from [45].

Example 19.6. Figure 19.14 shows a set of data, which have been generated via a two-dimensional Gaussian, with zero-mean value and covariance matrix equal to

$$\Sigma = \begin{bmatrix} 5.05 & -4.95 \\ -4.95 & 5.05 \end{bmatrix}.$$

The corresponding eignvalues/eigenvectors are computed as

$$\lambda_1 = 0.05, \quad \boldsymbol{a}_1 = [1, 1]^T,$$
$$\lambda_2 = 5.00, \quad \boldsymbol{a}_2 = [-1, 1]^T.$$

Observe that the data are distributed mainly around a straight line. The EM PPCA algorithm was run on this set of data, for $m = 1$. The resulting matrix A, which now becomes a vector, is

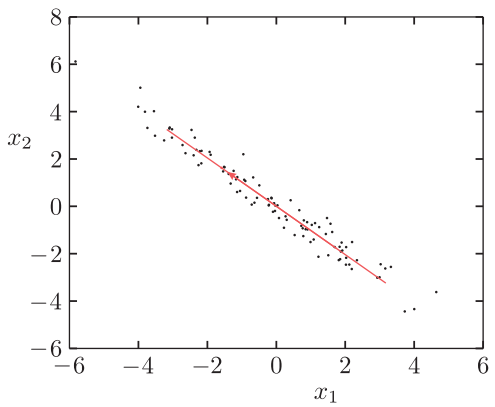

FIGURE 19.14

Data points are distributed around a straight line (one-dimensional subspace) in $\mathbb{R}^2$. The subspace is fully recovered by the PPCA, running the EM algorithm.

$$\boldsymbol{a} = [-1.71, 1.71]^T$$

and $\beta = 0.24$. Note that the obtained vector $\boldsymbol{a}$ points in the direction of the line (subspace) around which the data are distributed.

Remarks 19.6.

- In PPCA, a special diagonal structure was assumed for Σ_ϵ. An EM algorithm for the more general case has also been derived in the early eighties, [146]. Moreover, if the Gaussian prior imposed on the latent variables is replaced by another one, different algorithms result. For example, if non-Gaussian priors are used, then ICA versions are obtained. As a matter of fact, employing different priors, probabilistic versions of the canonical correlation analysis (CCA) and the partial least-squares (PLS) methods result; related references have already been given in the respective sections. Sparsity-promoting priors have also been used, resulting in what is known as *sparse factor analysis*, for example, [8, 23]. Once the priors have been adopted, one uses standard arguments, more or less, to solve the task, like those discussed in Chapters 12 and 13.
- Besides real-valued variables, extensions to categorical variables have also been considered, for example, [104]. A unifying view of various probabilistic dimensionality reduction techniques is provided in [133].

19.8.3 MIXTURE OF FACTORS ANALYZERS: A BAYESIAN VIEW TO COMPRESSED SENSING

Let us go back to our original model in Eq. (19.75) and rephrase it into a more "trendy" fashion. Matrix A had dimensions $l \times m$ with $m < l$, and $\mathbf{z} \in \mathbb{R}^m$. Let us now make $m > l$. For example, the columns of A may comprise vectors of an overcomplete dictionary. Thus, this section can be considered as the probabilistic counterpart of Section 19.6. The required low dimensionality of the modeling is expressed by imposing sparsity on $\mathbf{z}$; we can rewrite the model in terms of the respective observations as, [45],

$$\boldsymbol{x}_n = A(\boldsymbol{z}_n \circ \boldsymbol{b}) + \boldsymbol{\epsilon}_n, \quad n = 1, 2, \ldots, N,$$

where N is the number of our training points and the vector $\boldsymbol{b} \in \mathbb{R}^m$ has elements $b_i \in \{0, 1\}$, $i = 1, 2, \ldots, m$. The product $\boldsymbol{z}_n \circ \boldsymbol{b}$ is the point-wise vector product, that is,

$$\boldsymbol{z}_n \circ \boldsymbol{b} = [z_n(1)b_1, z_n(2)b_2, \ldots, z_n(m)b_m]^T. \tag{19.89}$$

If $\|\boldsymbol{b}\|_0 \ll l$, then $\boldsymbol{x}_n$ is sparsely represented in terms of the columns of A and its intrinsic dimensionality is equal to $\|\boldsymbol{b}\|_0$. Adopting the same assumptions as before, that is,

$$p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \beta^{-1} I_l), \quad p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \alpha^{-1} I_m),$$

where now we have explicitly brought l and m into the notation in order to remind us of the associated dimensions. Also, for the sake of generality, we have assumed that the elements of $\mathbf{z}$ correspond to precision values different than one. Following our familiar standard arguments (as for Eq. (12.15)), it is readily shown that the observations $\boldsymbol{x}_n$, $n = 1, 2, \ldots, N$, are drawn from

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{x}|\mathbf{0}, \Sigma_x), \tag{19.90}$$

$$\Sigma_x = \alpha^{-1} A \Lambda A^T + \beta^{-1} I_l, \tag{19.91}$$

where

$$\Lambda = \text{diag}\{b_1, \ldots, b_m\}, \tag{19.92}$$

which guarantees that in Eq. (19.91) only the columns of A, which correspond to nonzero values of $\boldsymbol{b}$, contribute to the formation of Σ_x. We can rewrite the matrix product in the following form:

$$A\Lambda A^T = \sum_{i=1}^{m} b_i \boldsymbol{a}_i \boldsymbol{a}_i^T,$$

and because only $\|\boldsymbol{b}\|_0 := k \ll l$ nonzero terms contribute to the summation, this corresponds to a rank $k < l$ matrix, provided that the respective columns of A are linear independent. Furthermore, assuming that β^{-1} is small, then Σ_x turns out to have a rank approximately equal to k.

Our goal now becomes the learning of the involved parameters; that is, A, β, α, and Λ. This can be done in a standard Bayesian setting by imposing priors on α, β (typically gamma pdfs) and for the columns of A,

$$p(\boldsymbol{a}_i) = \mathcal{N}\left(\boldsymbol{a}_i | 0, \frac{1}{l} I_l\right), \quad i = 1, 2, \ldots, m,$$

which guarantees unit expected norm for each column. The prior for the elements of $\boldsymbol{b}$ are chosen to follow a Bernoulli distribution (see [45] for more details).

Before generalizing the model, let us see the underlying geometric interpretation of the adopted model. Recall from our statistics basics (see also, Section 2.3.2, Chapter 2) that most of the activity of a set of jointly Gaussian variables takes place within an (hyper)ellipsoid whose principle axes are determined by the eigenstructure of the covariance matrix. Thus, assuming that the values of $\mathbf{x}$ lie close to a subspace/(hyper)plane, the resulting Gaussian model Eqs. (19.90)–(19.91) can sufficiently model it by adjusting the elements of Σ_x (after training) so that the corresponding high probability region forms a sufficiently flat ellipsoid; see Figure 19.15 for an illustration.

Once we have established the geometric interpretation of our factor model, let us leave our imagination free to act. Can this viewpoint be extended for modeling data that originate from a union of subspaces? A reasonable response to this challenge would be to resort to a mixture of factors; one for each subspace. However, there is more to it than that. It has been shown, for example, [25] that a compact manifold can be covered by a finite number of topological disks, whose dimensionality is equal to the dimensionality of the manifold. Associating topological disks with the principle hyperplanes that define sufficiently flat hyperellipsoids, one can model the data activity, which takes place along a manifold, by a sufficient number of factors, one per ellipsoid ([45]).

A *mixture of factor analyzers* (MSA) is defined as

$$p(\boldsymbol{x}) = \sum_{j=1}^{J} P_j \mathcal{N}\left(\boldsymbol{x} | \boldsymbol{\mu}_j, \alpha_j^{-1} A_j \Lambda_j A_j^T + \beta^{-1} I_l\right), \tag{19.93}$$

where $\sum_{j=1}^{J} P_j = 1$, $\Lambda_j = \text{diag}\left\{b_{j_1}, \ldots, b_{j_m}\right\}$, $b_{j_i} \in \{0, 1\}$, $i = 1, 2, \ldots, m$. The expansion in Eq. (19.93) for fixed J and preselected Λ_j, for the jth factor, has been known for some time; in this context, learning of the unknown parameters is achieved in the Bayesian framework, by imposing appropriate priors and mobilizing techniques such as the variational EM (e.g., [74]), the EM ([165]),

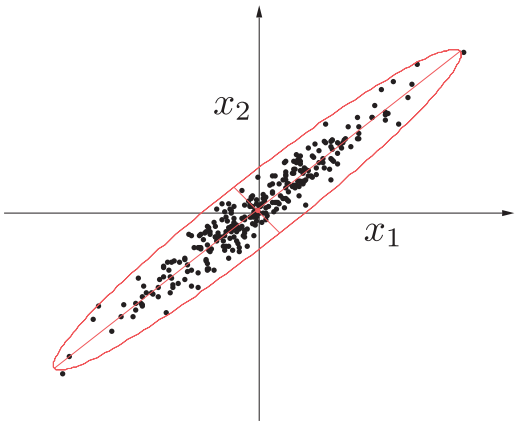

FIGURE 19.15

Data points that lie close to a hyperplane can be sufficiently modeled by a Gaussian pdf whose high probability region corresponds to a sufficiently flat (hyper)ellipsoid.

and the maximum likelihood [170]. In a more recent treatment of the problem, the dimensionality of each Λ_j, $j = 1, 2, \ldots, J$, as well as the number of factors, J, can be learned by the learning scheme. To this end, *nonparametric priors* are mobilized; see, for example, [39, 45, 87], and Section 13.12. The model parameters are then computed via Gibbs sampling (Chapter 14) or variational Bayesian techniques. Note that in general, different factors may turn out to have different dimensionality.

For *nonlinear manifold learning*, the geometric interpretation of Eq. (19.93) is illustrated in Figure 19.16. The number J is the number of flat ellipsoids used to cover the manifold, $\boldsymbol{\mu}_j$ are the sampled points on the manifold, the columns $A_j\Lambda_j$ (approximately) span the local $k-$dimensional tangent subspace, the noise variance β^{-1} depends on the manifold curvature, and the weights P_j reflect the respective density of the points across the manifold. The method has also been used for matrix completion (see also Section 19.10.1) via a low rank matrix approximation of the involved matrices ([39]).

Once the model has been learned, using Bayesian inference on a set of training data $\boldsymbol{x}_n$, $n = 1, 2, \ldots, N$, it can subsequently be used for compressed sensing; that is, to be able to obtain any $\boldsymbol{x}$, which belongs in the ambient space $\mathbb{R}^l$ but "lives" in the learned k-dimensional manifold, modeled by Eq. (19.93), using $K \ll l$ measurements. To this end, in a complete analogy with what has been said in Chapter 9, one has to determine a sensing matrix, which is denoted here as $\Phi \in \mathbb{R}^{K\times l}$, so that to be able to recover $\boldsymbol{x}$ from the measured (projection) vector

$$\boldsymbol{y} = \Phi\boldsymbol{x} + \boldsymbol{\eta},$$

where $\boldsymbol{\eta}$ denotes the vector of the (unobserved) samples of the measurement noise; all that is now needed is to compute the posterior $p(\boldsymbol{x}|\boldsymbol{y})$. Assuming the noise samples follow a Gaussian and because $p(\boldsymbol{x})$ is a sum of Gaussians, it can be readily seen that the posterior is also a sum of Gaussians determined by the parameters in Eq. (19.93) and the covariance matrix of the noise vector. Hence, $\boldsymbol{x}$ can be recovered

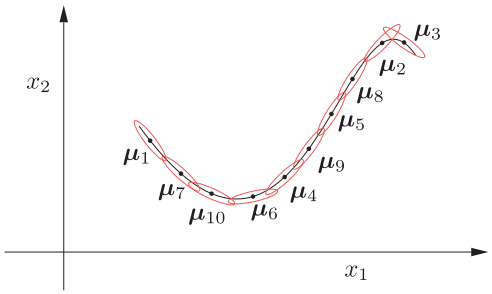

FIGURE 19.16

The curve (manifold) is covered by a number of sufficiently flat ellipsoids centered at the respective mean values.

by a substantially smaller number of measurements, K, compared to l. In [45], a theoretical analysis is carried out that relates the dimensionality of the manifold, k, the dimensionality of the ambient space, l, and Gaussian/sub-Gaussian types of sensing matrices Φ; this is an analogy to the RIP so that a stable embedding is guaranteed (Section 9.9).

One has to point out a major difference between the techniques developed in Chapter 9 and the current section. There, the model that generates the data was assumed to be known; the signal, denoted there by $\boldsymbol{s}$, was written as

$$\boldsymbol{s} = \Psi\boldsymbol{\theta},$$

where Ψ was the matrix of the dictionary and $\boldsymbol{\theta}$ the sparse vector. That is, the signal was assumed to reside in a subspace, which is spanned by some of the columns of Ψ; in order to recover the signal vector, one had to search for it in a union of subspaces. In contrast, in the current section, we had to "learn" the manifold in which the signal, denoted here by $\boldsymbol{x}$, lies.

19.9 NONLINEAR DIMENSIONALITY REDUCTION

All the techniques that have been considered so far build around linear models, which relate the observed and the latent variables. In this section, we turn our attention to their nonlinear relatives. Our aim is to discuss the main directions that are currently popular and we will not delve into many details. The interested reader can get a deeper understanding and related implementation details from the references provided in the text.[6]

19.9.1 KERNEL PCA

As its name suggests, this is a kernelized version of the classical PCA, and it was first introduced in [151]. As we have seen in Chapter 11, the idea behind any kernelized version of a linear method is to map the variables that originally lie in a low dimensional space, $\mathbb{R}^l$ into a high (possibly infinite)

[6] Much of this section is based on [164].

dimensional reproducing kernel Hilbert space (RKHS). This is achieved by adopting an implicit mapping,

$$\boldsymbol{x} \in \mathbb{R}^l \longmapsto \boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{H}. \tag{19.94}$$

Let $\boldsymbol{x}_n,\ n = 1, 2, \ldots, N$, be the available training points. The sample covariance matrix of the images, after mapping into $\mathbb{H}$ and assuming centered data, is given by[7]

$$\hat{\Sigma} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_n)\boldsymbol{\phi}(\boldsymbol{x}_n)^T. \tag{19.95}$$

The goal is to perform the eigendecomposition of $\hat{\Sigma}$, that is,

$$\hat{\Sigma}\boldsymbol{u} = \lambda \boldsymbol{u}. \tag{19.96}$$

By the definition of $\hat{\Sigma}$, it can be shown that $\boldsymbol{u}$ lies in the span$\{\boldsymbol{\phi}(\boldsymbol{x}_1), \boldsymbol{\phi}(\boldsymbol{x}_2), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_N)\}$. Indeed,

$$\lambda \boldsymbol{u} = \left(\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_n)\boldsymbol{\phi}^T(\boldsymbol{x}_n)\right)\boldsymbol{u} = \frac{1}{N}\sum_{n=1}^{N} \big(\boldsymbol{\phi}^T(\boldsymbol{x}_n)\boldsymbol{u}\big)\boldsymbol{\phi}(\boldsymbol{x}_n),$$

and for $\lambda \neq 0$ we can write

$$\boldsymbol{u} = \sum_{n=1}^{N} a_n \boldsymbol{\phi}(\boldsymbol{x}_n). \tag{19.97}$$

Combining Eqs. (19.96) and (19.97), it turns out (Problem 19.7) that the problem is equivalent to performing an eigendecomposition of the corresponding kernel matrix (Chapter 11)

$$\mathcal{K}\boldsymbol{a} = N\lambda \boldsymbol{a}, \tag{19.98}$$

where

$$\boldsymbol{a} := [a_1, a_2, \ldots, a_N]^T. \tag{19.99}$$

As we already know (Section 11.5.1), the elements of the kernel matrix are $\mathcal{K}(i, j) = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with $\kappa(\cdot, \cdot)$ being the adopted kernel function. Thus, the kth eigenvector of $\hat{\Sigma}$, corresponding to the kth (nonzero) eigenvalue of $\mathcal{K}$ in Eq. (19.98), is expressed as

$$\boldsymbol{u}_k = \sum_{n=1}^{N} a_{kn}\boldsymbol{\phi}(\boldsymbol{x}_n), \quad k = 1, 2, \ldots, p \tag{19.100}$$

where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$ denote the respective eigenvalues in descending order and λ_p is the smallest nonzero one and $\boldsymbol{a}_k^T := [a_{k1}, \ldots, a_{kN}]$ is the kth eigenvector of the kernel matrix. The latter is assumed to be normalized so that $\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle = 1,\ k = 1, 2, \ldots, p$, where $\langle \cdot, \cdot \rangle$ is the inner product in the Hilbert space $\mathbb{H}$. This imposes an equivalent normalization on the respective $\boldsymbol{a}_k$'s, resulting from

$$1 = \langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle = \left\langle \sum_{i=1}^{N} a_{ki}\boldsymbol{\phi}(\boldsymbol{x}_i), \sum_{j=1}^{N} a_{kj}\boldsymbol{\phi}(\boldsymbol{x}_j) \right\rangle$$

[7] If the dimension of $\mathbb{H}$ is infinite, the definition of the covariance matrix needs a special interpretation, but we will not bother with it here.

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ki} a_{kj} \mathcal{K}(i,j)$$

$$= \boldsymbol{a}_k^T \mathcal{K} \boldsymbol{a}_k = N\lambda_k \boldsymbol{a}_k^T \boldsymbol{a}_k, \quad k = 1, 2, \ldots, p. \tag{19.101}$$

We are now ready to summarize the basic steps for performing a kernel PCA; that is, to compute the corresponding latent variables (kernel principle components). Given $\boldsymbol{x}_n \in \mathbb{R}^l,\ n = 1, 2, \ldots, N$, and a kernel function $\kappa(\cdot, \cdot)$

- Compute the $N \times N$ kernel matrix, with elements $\mathcal{K}(i,j) = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
- Compute the m dominant eigenvalues/eigenvectors $\lambda_k,\ \boldsymbol{a}_k,\ k = 1, 2, \ldots, m$, of $\mathcal{K}$ (Eq. (19.98)).
- Perform the required normalization (Eq. (19.101)).
- Given a feature vector $\boldsymbol{x} \in \mathbb{R}^l$, obtain its low-dimensional representation by computing the m projections onto each one of the dominant eigenvectors,

$$z_k := \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{u}_k \rangle = \sum_{n=1}^{N} a_{kn} \kappa(\boldsymbol{x}, \boldsymbol{x}_n),\ k = 1, 2, \ldots, m. \tag{19.102}$$

The operations given in Eq. (19.102) correspond to a *nonlinear mapping* in the input space. Note that, in contrast to the linear PCA, the dominant eigenvectors $\boldsymbol{u}_k,\ k = 1, 2, \ldots, m$, are not computed explicitly. All we know are the respective (nonlinear) projections, z_k along them. However, after all, this is what we are finally interested in.

Remarks 19.7.

- Kernel PCA is equivalent to performing a standard PCA in the RKHS $\mathbb{H}$. It can be shown that all the properties associated with the dominant eigenvectors, as discussed for the PCA, are still valid for the kernel PCA. That is, (a) the dominant eigenvector directions optimally retain most of the variance; (b) the MSE in approximating a vector (function) in $\mathbb{H}$ in terms of the m dominant eigenvectors is minimal, with respect to any other m directions; and (c) projections onto the eigenvectors are uncorrelated [151].
- Recall from Remarks 19.1 that the eigendecomposition of the Gram matrix was required for the metric multidimensional scaling (MDS) method. Because the kernel matrix is the Gram matrix in RKHS, kernel PCA can be considered as a kernelized version of MDS, where inner products in the input space have been replaced by kernel operations in the Gram matrix.
- Note that the kernel PCA method does not consider an explicit underlying structure of the manifold on which the data reside.
- A variant of the kernel PCA, known as the kernel entropy component analysis (ECA), has been developed in [96], where the dominant directions are selected so as to maximize the Renyi entropy.

19.9.2 GRAPH-BASED METHODS

Laplacian eigenmaps

The starting point of this method is the assumption that the points in the data set, $\mathcal{X}$, lie on a smooth manifold $\mathcal{M} \supset \mathcal{X}$, whose intrinsic dimension is equal to $m < l$ and it is embedded in $\mathbb{R}^l$, that is, $\mathcal{M} \subset$

$\mathbb{R}^l$. The dimension m is given as a parameter by the user. In contrast, this is not required in the kernel PCA, where m is the number of dominant components, which, in practice, is determined so that the gap between λ_m and λ_{m+1} has a "large" value.

The main philosophy behind the method is to compute the low-dimensional representation of the data so that *local neighborhood information* in $\mathcal{X} \subset \mathcal{M}$ is optimally preserved. In this way, one attempts to get a solution that reflects the geometric structure of the manifold. To achieve this, the following steps are in order:

Step 1: Construct a graph $G = (V, E)$, where $V = \{v_n,\ n = 1, 2, \ldots, N\}$ is a set of vertices and $E = \{e_{ij}\}$ is the corresponding set of edges connecting vertices $(v_i, v_j),\ i,j = 1, 2, \ldots, N$ (see also Chapter 15). Each node, v_n, of the graph corresponds to a point, $\boldsymbol{x}_n$, in the data set, $\mathcal{X}$. We connect v_i, v_j, that is, insert the edge e_{ij} between the respective nodes, if points $\boldsymbol{x}_i, \boldsymbol{x}_j$ are "close" to each other. According to the method, there are two ways of quantifying "closeness." Vertices v_i, v_j are connected with an edge if:

1. $||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 < \epsilon$, for some user-defined parameter ϵ, where $||\cdot||$ is the Euclidean norm in $\mathbb{R}^l$, or
2. $\boldsymbol{x}_j$ is among the k-nearest neighbors of $\boldsymbol{x}_i$ or $\boldsymbol{x}_i$ is among the k-nearest neighbors of $\boldsymbol{x}_j$, where k is a user-defined parameter and neighbors are chosen according to the Euclidean distance in $\mathbb{R}^l$. The use of the Euclidean distance is justified by the smoothness of the manifold that allows to approximate, locally, manifold geodesics by Euclidean distances in the space where the manifold is embedded. The latter is a known result from differential geometry.

   For those who are unfamiliar with such concepts, think of a sphere embedded in the three-dimensional space. If somebody is constrained to live on the surface of the sphere, the shortest path to go from one point to another is the geodesic between these two points. Obviously this is not a straight line but an arc across the surface of the sphere. However, if these points are close enough, their geodesic distance can be approximated by their Euclidean distance, computed in the three-dimensional space.

Step 2: Each edge, e_{ij}, is associated with a weight, $W(i,j)$. For nodes that are not connected, the respective weights are zero. Each weight, $W(i,j)$, is a measure of the "closeness" of the respective neighbors, $\boldsymbol{x}_i, \boldsymbol{x}_j$. A typical choice is

$$W(i,j) = \begin{cases} \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{\sigma^2}\right), & \text{if } v_i, v_j \text{ correspond to neighbors,} \\ 0 & \text{otherwise,} \end{cases}$$

where σ^2 is a user-defined parameter. We form the $N \times N$ weight matrix W having as elements the weights $W(i,j)$. Note that W is symmetric and it is *sparse* because, in practice, many of its elements turn out to be zero.

Step 3: Define the diagonal matrix D with elements $D_{ii} = \sum_j W(i,j),\ i = 1, 2, \ldots, N$, and also the matrix $L := D - W$. The latter is known as the *Laplacian matrix of the graph*, $G(V, E)$. Perform the generalized eigendecomposition

$$L\boldsymbol{u} = \lambda D\boldsymbol{u}.$$

Let $0 = \lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_m$ be the smallest $m+1$ eigenvalues.[8] Ignore the $\boldsymbol{u}_o$ eigenvector corresponding to $\lambda_0 = 0$ and choose the next m eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m$. Then map

$$\boldsymbol{x}_n \in \mathbb{R}^l \longmapsto \boldsymbol{z}_n \in \mathbb{R}^m, \quad n = 1, 2, \ldots, N,$$

where

$$\boldsymbol{z}_n^T = [u_{1n}, u_{2n}, \ldots, u_{mn}], \quad n = 1, 2, \ldots, N. \tag{19.103}$$

That is, $\boldsymbol{z}_n$ comprises the nth components of the m previous eigenvectors. The computational complexity of a general eigendecomposition solver amounts to $O(N^3)$ operations. However, for sparse matrices, such as the Laplacian matrix, L, efficient schemes can be employed to reduce complexity to be subquadratic in N, e.g., the Lanczos algorithm [77].

The proof concerning the statement of step 3, will be given for the case of $m = 1$. For this case, the low-dimensional space is the real axis. Our path evolves along the lines adopted in [18]. The goal is to compute $z_n \in \mathbb{R}$, $n = 1, 2, \ldots, N$, so that connected points (in the graph, i.e., neighbors) stay as close as possible after the mapping onto the one-dimensional subspace. The criterion used to satisfy the closeness after the mapping is

$$E_L = \sum_{i=1}^{N} \sum_{j=1}^{N} (z_i - z_j)^2 W(i,j), \tag{19.104}$$

to become minimum. Observe that if $W(i,j)$ has a large value (i.e., $\boldsymbol{x}_i, \boldsymbol{x}_j$ are close in $\mathbb{R}^l$), then if the respective z_i, z_j are far apart in $\mathbb{R}$ it incurs a heavy penalty in the cost function. Also, points that are not neighbors do not affect the minimization as the respective weights are zero. For the more general case, where $1 < m < l$, the cost function becomes

$$E_L = \sum_{i=1}^{N} \sum_{j=1}^{N} ||\boldsymbol{z}_i - \boldsymbol{z}_j||^2 W(i,j).$$

Let us now reformulate Eq. (19.104). After some trivial algebra, we obtain

$$\begin{aligned} E_L &= \sum_i z_i^2 \sum_j W(i,j) + \sum_j z_j^2 \sum_i W(i,j) - 2 \sum_i \sum_j z_i z_j W(i,j) \\ &= \sum_i z_i^2 D_{ii} + \sum_j z_j^2 D_{jj} - 2 \sum_i \sum_j z_i z_j W(I,j) \\ &= 2 \boldsymbol{z}^T L \boldsymbol{z}, \end{aligned} \tag{19.105}$$

where

$$\boxed{L := D - W: \quad \text{Laplacian Matrix of the Graph,}} \tag{19.106}$$

and $\boldsymbol{z}^T = [z_1, z_2, \ldots, z_N]$. The Laplacian matrix, L, is symmetric and positive semidefinite. The latter is readily seen from the definition in Eq. (19.105), where E_L is always a nonnegative scalar. Note that the

[8] In contrast to the notation used for PCA, the eigenvalues here are marked in ascending order. This is because, in this subsection, we are interested in determining the smallest values and such a choice is notationally more convenient.

larger the value of D_{ii} the more "important" is the sample $\boldsymbol{x}_i$. This is because it implies large values for $W(i,j)$, $j = 1, 2, \ldots, N$, and plays a dominant role in the minimization process. Obviously, the minimum of E_L is achieved by the trivial solution $z_i = 0,\ i = 1, 2, \ldots, N$. To avoid this, as it is common in such cases, we constrain the solution to a prespecified norm. Hence, our problem now becomes

$$\begin{aligned} \min_{z} \quad & \boldsymbol{z}^T L \boldsymbol{z}, \\ \text{s.t.} \quad & \boldsymbol{z}^T D \boldsymbol{z} = 1. \end{aligned}$$

Although we can work directly on the previous task, we will slightly reshape it in order to use tools that are more familiar to us. Define

$$\boldsymbol{y} = D^{1/2}\boldsymbol{z}, \tag{19.107}$$

and

$$\tilde{L} = D^{-1/2} L D^{-1/2}, \tag{19.108}$$

which is known as the *normalized graph Laplacian* matrix. It is now readily seen that our optimization problem becomes

$$\min_{y} \quad \boldsymbol{y}^T \tilde{L} \boldsymbol{y}, \tag{19.109}$$

$$\text{s.t.} \quad \boldsymbol{y}^T \boldsymbol{y} = 1. \tag{19.110}$$

Using Lagrange multipliers and equating the gradient of the Lagrangian to zero, it turns out that the solution is given by

$$\tilde{L}\boldsymbol{y} = \lambda \boldsymbol{y}. \tag{19.111}$$

In other words, computing the solution becomes equivalent to solving an eigenvalue-eigenvector problem. Substituting Eq. (19.111) into the cost function in (19.109) and taking into account the constraint (19.110), it turns out that the value of the cost associated with the optimal $\boldsymbol{y}$ is equal to λ. Hence, the solution is the eigenvector corresponding to the minimum eigenvalue. However, the minimum eigenvalue of $\tilde{L}$ is zero and the corresponding eigenvector corresponds to a trivial solution. Indeed, observe that

$$\tilde{L} D^{1/2}\mathbf{1} = D^{-1/2} L D^{-1/2} D^{1/2}\mathbf{1} = D^{-1/2}(D - W)\mathbf{1} = \mathbf{0},$$

where $\mathbf{1}$ is the vector having all its elements equal 1. In words, $\boldsymbol{y} = D^{1/2}\mathbf{1}$ is an eigenvector corresponding to the zero eigenvalue and it results in the trivial solution, $z_i = 1,\ i = 1, 2, \ldots, N$. That is, all the points are mapped onto the same point in the real line. To exclude this undesired solution, recall that $\tilde{L}$ is a positive semidefinite matrix and, hence, 0 is its smallest eigenvalue. In addition, if the graph is assumed to be connected, that is, there is at least one path (see Chapter 15) that connects any pair of vertices, $D^{1/2}\mathbf{1}$ is the only eigenvector associated with the zero eigenvalue, λ_0, [18]. Also, as $\tilde{L}$ is a symmetric matrix, we know (Appendix A.2) that its eigenvectors are orthogonal to each other. In the sequel, we impose an extra constraint and we now require the solution to be *orthogonal* to $D^{1/2}\mathbf{1}$. Constraining the solution to be orthogonal to the eigenvector corresponding to the smallest (zero) eigenvalue, drives the solution to the next eigenvector corresponding to the next smallest (nonzero) eigenvalue λ_1. Note that the eigendecomposition of $\tilde{L}$ is equivalent to what we called generalized eigendecomposition of L in step 3 before.

For the more general case of $m > 1$, we have to compute the m eigenvectors associated with $\lambda_1 \leq \ldots \leq \lambda_m$. As a matter of fact, for this case, the constraints prevent us from mapping into a subspace of dimension less than the desired m. For example, we do not want to project in a three-dimensional space and the points to lie on a two-dimensional plane or on a one-dimensional line. For more details, the interested reader is referred to the insightful paper [18].

Local linear embedding (LLE)

As was the case with the Laplacian eigenmap method, *local linear embedding* (LLE) assumes that the data points rest on a smooth enough manifold of dimension m, which is embedded in the $\mathbb{R}^l$ space, with $m < l$ [145]. The smoothness assumption allows us to further assume that, provided there is sufficient data and the manifold is "well" sampled, nearby points lie on (or close to) a "locally" *linear* patch of the manifold (see, also, related comments in Section 19.8.3). The algorithm in its simplest form is summarized in the following three steps:

Step 1: For each point, $\boldsymbol{x}_n,\ n = 1, 2, \ldots, N$, search for its nearest neighbors.

Step 2: Compute the weights $W(i,j),\ i,j = 1, 2, \ldots, N$, that best reconstruct each point, $\boldsymbol{x}_n$, from its nearest neighbors, so as to minimize the cost

$$\arg\min_{W} \ E_W = \sum_{n=1}^{N} \left\| \boldsymbol{x}_n - \sum_{j=1}^{N} W(i,j)\boldsymbol{x}_{n_j} \right\|^2, \tag{19.112}$$

where $\boldsymbol{x}_{n_j}$ denotes the jth neighbor of the nth point. The weights are constrained: (a) to be zero for points which are not neighbors and (b) the rows of the weight matrix add to one, that is,

$$\sum_{j=1}^{N} W(i,j) = 1. \tag{19.113}$$

That is, the sum of the weights, over all neighbors, must be equal to one.

Step 3: Once the weights have been computed from the previous step, use them to obtain the corresponding points $\boldsymbol{z}_n \in \mathbb{R}^m,\ n = 1, 2, \ldots, N$, so that to minimize the cost with respect to the unknown set of points $\mathcal{Z} = \{\boldsymbol{z}_n,\ n = 1, 2, \ldots, N\}$,

$$\arg\min_{\boldsymbol{z}_n:\ n=1,\ldots,N} \ E_{\mathcal{Z}} = \sum_{n=1}^{N} \left\| \boldsymbol{z}_n - \sum_{j=1}^{N} W(n,j)\boldsymbol{z}_j \right\|^2. \tag{19.114}$$

The above minimization takes place subject to two constraints, to avoid degenerate solutions: (a) the outputs are centered, $\sum_n \boldsymbol{z}_n = \boldsymbol{0}$, and (b) the outputs have unit covariance matrix [149]. Nearest points, in step 1, are searched in the same way as it is carried out for the Laplacian eigenmap method. Once again, the use of the Euclidean distance is justified by the smoothness of the manifold, as long as the search is limited "locally" among neighboring points. For the second step, the method exploits the local linearity of a smooth manifold and tries to predict *linearly* each point by its neighbors using the least-squares error criterion. Minimizing the cost subject to the constraint given in Eq. (19.113) results in a solution that satisfies the following three properties:

1. Rotation invariance.
2. Scale invariance.
3. Translation invariance.

The first two can easily be verified by the form of the cost function and the third one is the consequence of the imposed constraints. The implication of this is that the computed weights encode information about the intrinsic characteristics of each neighborhood and they do not depend on the particular point.

The resulting weights, $W(i,j)$, reflect the intrinsic properties of the local geometry underlying the data, and because our goal is to retain the local information after the mapping, these weights are used to reconstruct each point in the $\mathbb{R}^m$ subspace by its neighbors. As is nicely stated in [149], it is as if we take a pair of scissors to cut small linear patches of the manifold and place them in the low-dimensional subspace.

It turns out that solving (19.114) for the unknown points, $\boldsymbol{z}_n,\ n = 1, 2, \ldots, N$, is equivalent to

- Performing an eigendecomposition of the matrix $(I - W)^T(I - W)$.
- Discarding the eigenvector that corresponds to the smallest eigenvalue.
- Taking the eigenvectors that correspond to the next (smaller) eigenvalues. These yield the low-dimensional latent variable scores, $\boldsymbol{z}_n,\ n = 1, 2, \ldots, N$.

Once again, the involved matrix W is sparse and if this is taken into account the eigenvalue problem scales relatively well to large data sets with complexity subquadratic in N. The complexity for step 2 scales as $O(Nk^3)$ and it is contributed by the solver of the linear set of equations with k unknowns for each point. The method needs two parameters to be provided by the user, the number of nearest neighbors, k (or ϵ) and the dimensionality m. The interested reader can find more on the LLE method in [149].

Isometric mapping (ISOMAP)

In contrast to the two previous methods that unravel the geometry of the manifold on a local basis, the ISOMAP algorithm adopts the view that only the geodesic distances between all pairs of the data points can reflect the true structure of the manifold. Euclidean distances between points in a manifold cannot represent it properly, because points that lie far apart, as measured by their geodesic distance, may be close when measured in terms of their Euclidean distance (see Figure 19.17). ISOMAP is basically a variant of the multidimensional scaling (MDS) algorithm, in which the Euclidean distances are substituted by the respective geodesic distances along the manifold. The essence of the method is to estimate geodesic distances between points that lie faraway. To this end, a two-step procedure is adopted.

Step 1: For each point, $\boldsymbol{x}_n,\ n = 1, 2, \ldots, N$, compute the nearest neighbors and construct a graph $G(V, E)$ whose vertices represent the data points and the edges connect nearest neighbors. (Nearest neighbors are computed with either of the two alternatives used for the Laplacian eigenmap method. The parameters k or ϵ are user-defined parameters.) The edges are assigned weights based on the respective Euclidean distance (for nearest neighbors, this is a good approximation of the respective geodesic distance).

Step 2: Compute the pairwise geodesic distances among all pairs $(i,j),\ i,j = 1, 2, \ldots, N$, along shortest paths through the graph. The key assumption is that the geodesic between any two points on the manifold can be approximated by the *shortest path* connecting the two points along the graph $G(V, E)$. To this end, efficient algorithms can be used to achieve it with complexity $\mathcal{O}(N^2 \ln N + N^2 k)$ (e.g., Djikstar's algorithm, [55]). This cost can be prohibitive for large values of N.

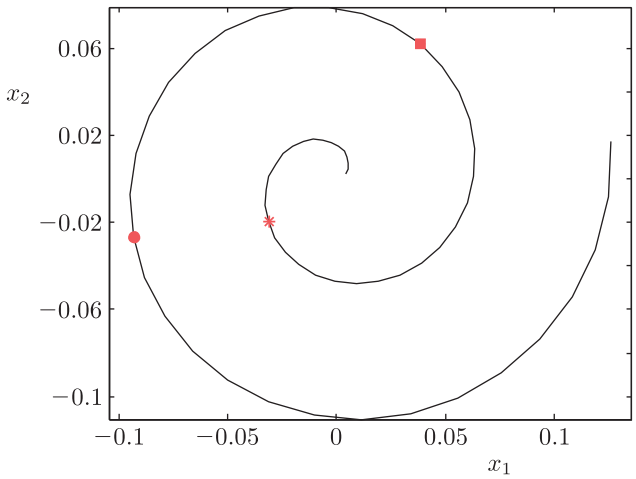

FIGURE 19.17

The point denoted by a "star," is deceptively closer to the point denoted by a "dot" than to the point denoted by a "box," if distance is measured in terms of the Euclidean distance. However, if one is constrained to travel along the spiral, the geodesic distance is the one that determines closeness and it is the "box" point that is closer to the "star."

Having estimated the geodesics between all pairs of point, the MDS method is mobilized. Thus, the problem becomes equivalent to performing the eigendecomposition of the respective Gram matrix and selecting the m most dominant eigenvectors to represent the low-dimensional space. After the mapping, Euclidean distances between points in the low-dimensional subspace match the respective geodesic distances on the manifold in the original high-dimensional space. As it is the case in PCA and MDS, m is estimated by the number of significant eigenvalues. It can be shown that ISOMAP is guaranteed asymptotically ($N \longrightarrow \infty$) to recover the true dimensionality of a class of nonlinear manifolds [61, 163].

All three graph-based methods share a common step for computing nearest neighbors in a graph. This is a problem of complexity $\mathcal{O}(N^2)$ but more efficient search techniques can be used by employing a special type of data structures, for example, [22]. A notable difference between the ISOMAP on the one side and the Laplacian eigenmap and LLE methods on the other is that the latter two approaches rely on the eigendecomposition of sparse matrices as opposed to the ISOMAP that relies on the eigendecomposition of the dense Gram matrix. This gives a computational advantage to the Laplacian eigenmap and LLE techniques. Moreover, the calculation of the shortest paths in the ISOMAP is another computationally demanding task. Finally, it is of interest to note that the three graph-based techniques perform the task of dimensionality reduction while trying to unravel, in one way or another, the geometric properties of the manifold on which the data (approximately) lie. In contrast, this is not the case with the kernel PCA, which shows no interest in any manifold learning. However, as the world is very small, in [81] it is pointed out that the graph-based techniques can be seen as special cases of the kernel PCA! This becomes possible if data-dependent kernels, derived from graphs encoding neighborhood information, are used in place of predefined kernel functions.

The goal of this section was to present some of the most basic directions that have been suggested for nonlinear dimensionality reduction. Besides the previous basic schemes, a number of variants have been proposed in the literature (e.g., [20, 65, 153]). In [140] and [111] (*diffusion maps*), the low-dimensional embedding is achieved so as to preserve certain measures that reflect the connectivity of the graph $G(V, E)$. In [29, 94], the idea of preserving the local information in the manifold has been carried out to define linear transforms of the form $\boldsymbol{z} = A^T\boldsymbol{x}$, and the optimization is now carried out with respect to the elements of A. The task of incremental manifold learning for dimensionality reduction was more recently considered in [114]. In [162, 173], the *maximum variance unfolding* method is introduced. The variance of the outputs is maximized under the constraint that (local) distances and angles are preserved among neighbors in the graph. Like the ISOMAP, it turns out that the top eigenvectors of a Gram matrix have to be computed, albeit avoiding the computationally demanding step of estimating geodesic distances, as it required by the ISOMAP. In [154], a general framework, called *graph embedding*, is presented that offers a unified view for understanding and explaining a number of known (including PCA and nonlinear PCA) dimensionality reduction techniques and it also offers a platform for developing new ones. For a more detailed and insightful treatment of the topic, the interested reader is referred to [27]. A review of nonlinear dimensionality reduction techniques can be found in, for example, [31, 116].

Example 19.7. Let a data set consisting of 30 points be in the two-dimensional space. The points result from sampling the spiral of Archimedes (see Figure 19.18a), described by

$$x_1 = a\theta\cos\theta, \quad x_2 = a\theta\sin\theta.$$

The points of the data set correspond to the values $\theta = 0.5\pi,\ 0.7\pi,\ 0.9\pi, \ldots, 2.05\pi$ (θ is expressed in radians), and $a = 0.1$. For illustration purposes and in order to keep track of the "neighboring" information, we have used a sequence of six symbols, "x", "+", "$\star$", "$\square$", "$\diamond$", "$\circ$" with black color, followed by the same sequence of symbols in red color, repeatedly.

To study the performance of PCA for this case, where data lie on a nonlinear manifold, we first performed the eigendecomposition of the covariance matrix, estimated from the data set. The resulting eigenvalues are

$$\lambda_2 = 0.089 \quad \text{and} \quad \lambda_1 = 0.049.$$

Observe that, the eigenvalues are comparable in size. Thus, if one would trust the "verdict" coming from PCA, the answer concerning the dimensionality of the data would be that it is equal to 2. Moreover, after projecting along the direction of the principle component (the straight line in Figure 19.18b), corresponding to λ_2, neighboring information is lost because points from different locations are mixed together.

In the sequel, the Laplacian eigenmap technique for dimensionality reduction is employed, with $\epsilon = 0.2$ and $\sigma = \sqrt{0.5}$. The obtained results are shown in Figure 19.18c. Looking from right to left, we see that the Laplacian method nicely "unfolds" the spiral in a one-dimensional straight line. Furthermore, neighboring information is retained in this one-dimensional representation of the data. Black and red areas are succeeding each other in the right order, and also, observing the symbols, one can see that neighbors are mapped to neighbors.

Example 19.8. Figure 19.19 shows samples from a three-dimensional spiral, parameterized as $x_1 = a\theta\cos\theta$, $x_2 = a\theta\sin\theta$, and sampled at $\theta = 0.5\pi,\ 0.7\pi\ , 0.9\pi, \ldots, 2.05\pi$ (θ is expressed in radians), $a = 0.1$ and $x_3 = -1,\ -0.8,\ -0.6, \ldots, 1$.

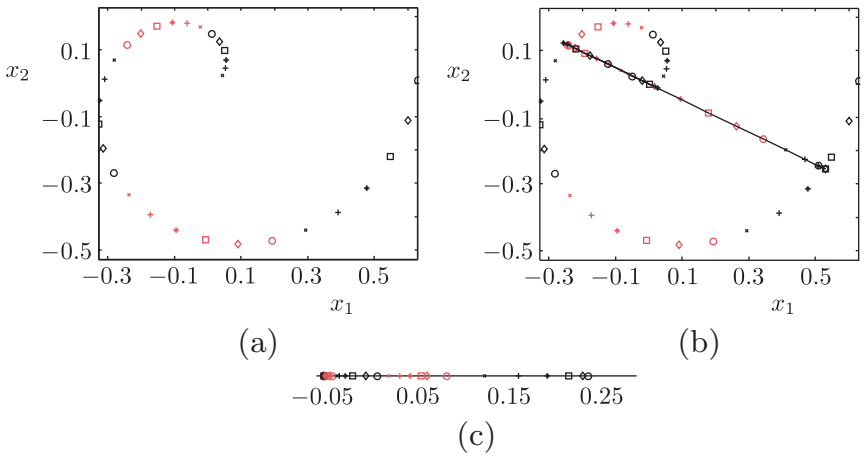

FIGURE 19.18

(a) A spiral of Archimedes in the two-dimensional space. (b) The previous spiral together with the projections of the sampled points on the direction of the first principle component, resulting from PCA. It is readily seen that neighboring information is lost after the projection and points corresponding to different parts of the spiral overlap. (c) The one-dimensional map of the spiral using the Laplacian method. In this case, the neighboring information is retained after the nonlinear projection and the spiral nicely unfolds to a one-dimensional line.

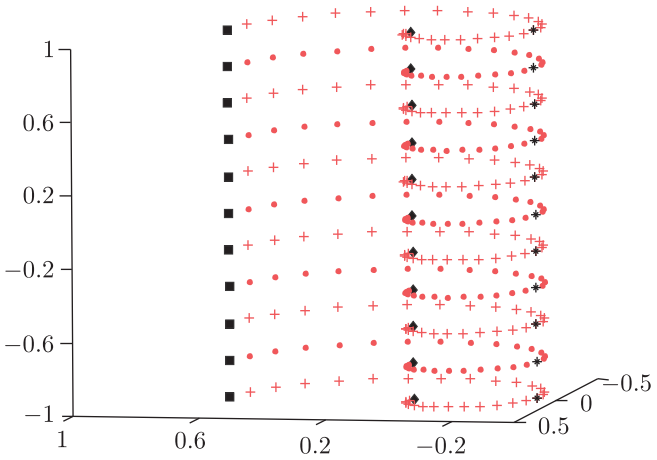

FIGURE 19.19

Samples from a three-dimensional spiral. One can think of it as a number of two-dimensional spirals one above the other. Different symbols have been used in order to track neighboring information.

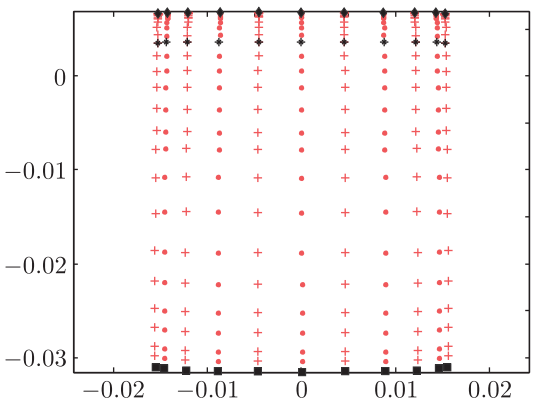

FIGURE 19.20

Two-dimensional mapping of the spiral of Figure 19.19 using the Laplacian eigenmap method. The three-dimensional structure is unfolded to the two-dimensional space by retaining the neighboring information.

For illustration purposes and in order to keep track of the "identity" of each point, we have used red crosses and dots interchangeably, as we move upward in the x_3 dimension. Also, the first, the middle, and the last points for each level of x_3 are denoted by black "$\diamond$", black "$\star$," and black "$\square$", respectively. Basically, all points at the same level lie on a two-dimensional spiral.

Figure 19.20 shows the two-dimensional mapping of the three-dimensional spiral using the Laplacian method for dimensionality reduction, with parameter values $\epsilon = 0.35$ and $\sigma = \sqrt{0.5}$. Comparing Figures 19.19 and 19.20, we see that all points corresponding to the same level x_3 are mapped across the same line, with the first point being mapped to the first one and so on. That is, as it was the case of the Example 19.7, the Laplacian method unfolds the three-dimensional spiral into a two-dimensional surface, while retaining neighboring information.

19.10 LOW-RANK MATRIX FACTORIZATION: A SPARSE MODELING PATH

The low-rank matrix factorization task has already been discussed from different perspectives. In this section, the task will be considered in a specific context; that of missing entries and/or in the presence of outliers. Such a focus is dictated by a number of more recent applications, especially in the framework of big data problems. To this end, sparsity-promoting arguments will be mobilized to offer a fresh look at this old problem. We are not going to delve into many details and our purpose is to highlight the main directions and methods, which have been considered.

19.10.1 MATRIX COMPLETION

To recapitulate some of the main findings in Chapters 9 and 10, let us consider a signal vector $\boldsymbol{s} \in \mathbb{R}^l$, where only N of its components are observed and the rest are unknown. This is equivalent with sensing $\boldsymbol{s}$ via a sensing matrix having its N rows picked uniformly at random from the standard (canonical) basis

$\Phi = I$, where I is the $l \times l$ identity matrix. The question that was posed there was whether it is possible to recover $\boldsymbol{s}$ exactly based on these N components. From the theory presented in Chapter 9, we know that one can recover all the components of $\boldsymbol{s}$, provided that $\boldsymbol{s}$ is sparse in some basis or dictionary, Ψ, which exhibits low mutual coherence with $\Phi = I$, and N is large enough, as has been pointed out in Section 9.9.

Inspired by the theoretical advances in compressed sensing, a question similar in flavor and with a prominent impact regarding practical applications was posed in [32]. Given an $l_1 \times l_2$ matrix M, assume that only $N \ll l_1 l_2$ among its entries are known. Concerning notation, we refer to a general matrix M, irrespective of how this matrix was formed. For example, it may correspond to an image array. The question now is whether one is able to recover the exact full matrix. This problem is widely known as *matrix completion* [32]. The answer, although it might come as a surprise, is "yes" with high probability, provided that (a) the matrix is *well structured* and complies with certain assumptions, (b) it has a *low rank*, $r \ll l$, where $l = \min(l_1, l_2)$, and (c) that N is large enough. Intuitively, this is plausible because a low-rank matrix is fully described in terms of a number of parameters (degrees of freedom), which is much smaller than its total number of entries. These parameters are revealed via its singular value decomposition (SVD)

$$M = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T = U \begin{bmatrix} \sigma_1 & & O \\ & \ddots & \\ O & & \sigma_r \end{bmatrix} V^T, \tag{19.115}$$

where r is the rank of the matrix, $\boldsymbol{u}_i \in \mathbb{R}^{l_1}$ and $\boldsymbol{v}_i \in \mathbb{R}^{l_2}$, $i = 1, 2, \ldots, r$, are the left and right orthonormal singular vectors, spanning the column and row spaces of M, respectively, σ_i, $i = 1, 2, \ldots, r$, are the corresponding singular values, and $U = [\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_r]$, $V = [\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_r]$.

Let $\boldsymbol{\sigma}_M$ denote the vector containing all the singular values of M, that is, $\boldsymbol{\sigma}_M = [\sigma_1, \sigma_2, \cdots, \sigma_l]^T$, then $\text{rank}(M) := \|\boldsymbol{\sigma}_M\|_0$. Counting the parameters associated with the singular values and vectors in Eq. (19.115), it turns out that the number of degrees of freedom of a rank r matrix is equal to $d_M = r(l_1 + l_2) - r^2$ (Problem 19.8). When r is small, d_M is much smaller than l.

Let us denote with Ω the set of N pairs of indices, (i,j), $i = 1, 2, \ldots, l_1$, $j = 1, 2, \ldots, l_2$, of the locations of the known entries of M, which have been sampled uniformly at random. Adopting a similar rationale to the one running across the backbone of sparsity-aware learning, one would attempt to recover M based on the following rank minimization problem,

$$\begin{aligned} \min_{\hat{M} \in \mathbb{R}^{l_1 \times l_2}} \quad & \|\boldsymbol{\sigma}_{\hat{M}}\|_0 \\ \text{s.t.} \quad & \hat{M}(i,j) = M(i,j), \quad (i,j) \in \Omega. \end{aligned} \tag{19.116}$$

It turns out that, assuming that there exists a unique low-rank matrix having as elements the specific known entries, then the task in (19.116) leads to the exact solution [32]. However, compared to the case of sparse vectors, in the matrix completion problem the uniqueness issue gets much more involved. The following issues play a crucial part concerning the uniqueness of the task in (19.116).

1. If the number of known entries is lower than the degrees of freedom, that is, $N < d_M$, then there is no way to recover the missing entries whatsoever, because there is an infinite number of low-rank matrices consistent with the N observed entries.
2. Even if $N \geq d_M$, uniqueness is still not guaranteed. It is required that the N elements with indices in Ω are such that at least one entry per column and one entry per row are observed. Otherwise,

even a rank-1 matrix $M = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T$ cannot be recovered. This becomes clear with a simple example. Assume that M is a rank-1 matrix and that no entry in the first column as well as in the last row is observed. Then, because for this case $M(i,j) = \sigma_1 u_{1i} v_{1j}$, it is clear that no information concerning the first component of $\boldsymbol{v}_1$ as well as the last component of $\boldsymbol{u}_1$ is available; hence, it is impossible to recover these singular vector components, regardless of which method is used. As a consequence, the matrix cannot be completed. On the other hand, if the elements of Ω are picked at random and N is large enough, one can only hope that Ω is such as to comply with the requirement above; i.e., at least one entry per row and column to be observed, with high probability. It turns out that this problem resembles the famous theorem in probability theory known as the *coupon collector's* problem. According to this, at least $N = C_0 l \ln l$ entries are needed, where C_0 is a constant [126]. This is the information theoretic limit for exact matrix completion [34] of any low-rank matrix.

3. Even if points (1) and (2) above are fulfilled, uniqueness is still not guaranteed. In fact, not every low-rank matrix is liable to exact completion, regardless of the number and the positions of the observed entries. Let us demonstrate this via an example. Let one of the singular vectors be sparse. Assume, without loss of generality, that the third left singular vector, $\boldsymbol{u}_3$, is sparse with sparsity level $k = 1$ and also that its nonzero component is the first one, that is, $u_{31} \neq 0$. The rest of $\boldsymbol{u}_i$ and all $\boldsymbol{v}_i$ are assumed to be dense. Let us return to the SVD for awhile in Eq. (19.115). Observe that the matrix M is written as the sum of r, $l_1 \times l_2$ matrices $\sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$, $i = 1, \ldots, r$. Thus, in this specific case where $\boldsymbol{u}_3$ is $k = 1$ sparse, the matrix $\sigma_3 \boldsymbol{u}_3 \boldsymbol{v}_3^T$ has zeros everywhere except for its first row. In other words, the information that $\sigma_3 \boldsymbol{u}_3 \boldsymbol{v}_3^T$ brings to the formation of M is concentrated in its first row only. This argument can also be viewed from another perspective; the entries of M obtained from any row except the first one, do not provide any useful information with respect to the values of the free parameters σ_3, $\boldsymbol{u}_3$, $\boldsymbol{v}_3$. As a result, in this case, unless one incorporates extra information about the sparse nature of the singular vector, the entries from the first row that are missed are not recoverable, because the number of parameters concerning this row is larger than the available number of data.

Intuitively, when a matrix has dense singular vectors it is better rendered for exact completion as each one among the observed entries carries information associated with all the d_M parameters that fully describe it. To this end, a number of conditions, which evaluate the suitability of the singular vectors, have been established. The simplest one is given next [32]:

$$\|\boldsymbol{u}_i\|_\infty \leq \sqrt{\frac{\mu_B}{l_1}}, \quad \|\boldsymbol{v}_i\|_\infty \leq \sqrt{\frac{\mu_B}{l_2}}, \; i = 1, \ldots, r. \tag{19.117}$$

where μ_B is a bound parameter. In fact, μ_B is a measure of the coherence of matrix U (and similarly of V),[9] (vis-à-vis the standard basis), defined as follows:

$$\mu(U) := \frac{l_1}{r} \max_{1 \leq i \leq l_1} \|P_U \boldsymbol{e}_i\|^2, \tag{19.118}$$

where P_U defines the orthogonal projection to subspace U and $\boldsymbol{e}_i$ is the ith vector of the canonical basis. Note that when U results from SVD, then $\|P_U \boldsymbol{e}_i\|^2 = \left\| U^T \boldsymbol{e}_i \right\|^2$. In essence, coherence is an index quantifying the extent to which the singular vectors are correlated with the standard basis $\boldsymbol{e}_i$, $i = 1, 2, \ldots, l$. The smaller the μ_B, the less "spiky" the singular vectors are likely to be, and the

[9] This is a quantity different than the mutual-coherence already discussed in Section 9.6.1.

corresponding matrix is better suited for exact completion. Indeed, assuming for simplicity a square matrix M, that is, $l_1 = l_2 = l$, then if *any one* among the singular vectors is sparse having a single nonzero component only, then, taking into account that $\boldsymbol{u}_i^T \boldsymbol{u}_i = \boldsymbol{v}_i^T \boldsymbol{v}_i = 1$, this value will have magnitude equal to one and the bound parameter will take its largest value possible, that is, $\mu_B = l$. On the other hand, the smaller value that μ_B can get is 1, something that occurs when the components of *all* the singular vectors assume the same value (in magnitude). Note that in this case, due to the normalization, this common component value has magnitude $\frac{1}{l}$. Tighter bounds to a matrix coherence result from the more elaborate incoherence property [32, 141] and the strong incoherence property [34]. In all cases, the larger the bound parameter the larger the number of known entries becomes, which is required in order to guarantee uniqueness.

In section 19.10.3, the aspects of uniqueness will be discussed in the context of a real-life application.

The task formulated in (19.116) is of limited practical interest because it is an NP-hard task. Thus, borrowing the arguments used in Chapter 9, the ℓ_0 (pseudo)norm is replaced by a *convexly* relaxed counterpart of it, that is,

$$\begin{aligned} \min_{\hat{M} \in \mathbb{R}^{l_1 \times l_2}} \quad & \left\| \boldsymbol{\sigma}_{\hat{M}} \right\|_1, \\ \text{s.t.} \quad & \hat{M}(i,j) = M(i,j), \quad (i,j) \in \Omega, \end{aligned} \tag{19.119}$$

where $\left\| \boldsymbol{\sigma}_{\hat{M}} \right\|_1$, that is, the sum of the singular values, is referred to as the *nuclear norm* of the matrix $\hat{M}$, often denoted as $\left\| \hat{M} \right\|_*$. The nuclear norm minimization was proposed in [69] as a convex approximation of rank minimization, which can be cast as a semidefinite programming task.

Theorem 19.1. *Let M be an $l_1 \times l_2$ matrix of rank r, which is a constant much smaller than $l = \min(l_1, l_2)$, obeying (19.117). Suppose that we observe N entries of M with locations sampled uniformly at random. Then there is a positive constant C such that if*

$$N \geq C \mu_B^4 l \ln^2 l, \tag{19.120}$$

then M is the unique solution to the task in (19.119) with probability at least $1 - l^{-3}$.

There might be an ambiguity on how small the rank should be in order for the corresponding matrix to be characterized as "low rank." More rigorously, a matrix is said to be of low rank if $r = \mathcal{O}(1)$, which means that r is a constant with no dependence (not even logarithmic), on l. Matrix completion is also possible for more general rank cases where, instead of the mild coherence property of (19.117), the incoherence and the strong incoherence properties [32, 34, 79, 141] are mobilized in order to get similar theoretical guaranties. The detailed exposition of these alternatives is beyond the scope of this book. In fact, Theorem 19.1 embodies the essence of the matrix completion task: with high probability, nuclear-norm minimization recovers all the entries of a low-rank matrix, M, with no error. More importantly, the number of entries, N, that the convexly relaxed problem needs is only by a logarithmic factor larger than the information theoretic limit, which, as it was mentioned before, equates to $C_0 l \ln l$. Moreover, similar to compressed sensing, robust matrix completion in the presence of noise is also possible as long as the request $\hat{M}(i,j) = M(i,j)$ in Eqs. (19.116) and (19.119) is replaced by $\|\hat{M}(i,j) - M(i,j)\|_2 \leq \epsilon$ [33]. Furthermore, the notion of matrix completion has also been extended to tensors, for example, [72, 155].

19.10.2 ROBUST PCA

The developments on matrix completion theory led, more recently, to the formulation and solution of another problem of high significance. To this end, the notation $\|M\|_1$, that is, the ℓ_1 norm of a matrix, is introduced and defined as the sum of the absolute values of its entries, that is, $\|M\|_1 = \sum_{i=1}^{l_1}\sum_{j=1}^{l_2}|M(i,j)|$. In other words, it acts on the matrix as if this were a long vector.

Assume now that M is expressed as the sum of a low-rank matrix, L, and a sparse matrix, S, that is, $M = L + S$. Consider the following convex minimization problem task, [35, 42, 176, 182], which is usually referred to as *principle component pursuit* (PCP),

$$\min_{\hat{L},\hat{S}} \quad \|\boldsymbol{\sigma}_M\|_1 + \lambda\|\hat{S}\|_1, \tag{19.121}$$

$$\text{s.t.} \quad \hat{L} + \hat{S} = M, \tag{19.122}$$

$\hat{L}$, $\hat{S}$ are both $l_1 \times l_2$ matrices. It can be shown that solving the task in (19.121)–(19.121) recovers *both* L and S according to the following theorem. [35]:

Theorem 19.2. *The PCP recovers both L and S with probability at least* $1 - cl_1^{-10}$*, where c is a constant, provided that:*

1. *The support set* Ω *of S is uniformly distributed among all sets of cardinality N.*
2. *The number, k, of nonzero entries of S is relatively small, that is,* $k \leq \rho l_1 l_2$*, where* ρ *is a sufficiently small positive constant.*
3. *L obeys the incoherence property.*
4. *The regularization parameter,* λ*, is constant with value* $\lambda = \frac{1}{\sqrt{l_2}}$*,*
5. $\text{rank}(L) \leq C\frac{l_2}{\ln^2 l_1}$*, with C being a constant.*

In other words, based on *all* the entries of a matrix M, which is known to be the sum of two unknown matrices L and S, with the first one being of low-rank matrix and the second being sparse, then PCP recovers exactly, with probability almost 1, both L and S, irrespective of how large the magnitude of the entries of S are, provided *that both r and k are sufficiently small*.

The applicability of the previous task is very broad. For example, PCP can be employed in order to find a low-rank approximation of M. In contrast to the standard PCA (SVD) approach, PCP is robust and insensitive in the presence of outliers, as these are naturally modeled, via the presence of S. Note that outliers are sparse by their nature. For this reason, the above task is widely known as *robust PCA via nuclear norm minimization*. (More classical PCA techniques are known to be sensitive to outliers and a number of alternative approaches have in the past been proposed toward its robustification, for example, [91, 102].)

When PCP serves as a robust PCA approach, the matrix of interest is L and S accounts for the outliers. However, PCP estimates both L and S. As will be discussed soon, another class of applications are well accommodated when the focus of interest is turned to the sparse matrix S itself.

Remarks 19.8.

- Just as ℓ_1 -minimization is the tightest convex relaxation of the combinatorial ℓ_0-minimization problem in sparse modeling, the nuclear-norm minimization is the tightest convex relaxation of the NP-hard rank minimization task. Besides the nuclear norm, other heuristics have also been proposed such as the log-determinant heuristic [69] and the max-norm [71].

- The nuclear norm as a rank minimization approach is the generalization of the trace-related cost, which is often used in the control community for the rank minimization of positive semidefinite matrices [125]. Indeed, when the matrix is symmetric and positive semidefinite, the nuclear norm of M is the sum of the eigenvalues and, thus, it is equal to the trace of M. Such problems arise when, for example, the rank minimization task refers to covariance matrices and positive semidefinite Toeplitz or Hankel matrices (see, e.g., [69]).
- Both matrix completion (19.119) and PCP (19.122) can be formulated as semidefinite programs and are solved based on interior-point methods. However, whenever the size of a matrix becomes large (e.g., 100×100), these methods are deemed to fail in practice due to excessive computational load and memory requirements. As a result, there is an increasing interest, which has propelled intensive research efforts, for the development of efficient methods to solve both optimization tasks, or related approximations, which scale well with large matrices. Many of these methods revolve around the philosophy of the iterative soft and hard thresholding techniques, as discussed in Chapter 9. However, in the current low-rank approximation setting, it is the singular values of the estimated matrix that are thresholded. As a result, in each iteration, the estimated matrix, after thresholding its singular values, tends to be of lower rank. The thresholding of the singular values is either imposed, such as in the case of the singular value thresholding (SVT) algorithm [30], or it results as a solution of regularized versions of (19.119) and (19.122) (see, e.g., [46, 167]). Moreover, algorithms inspired by greedy methods such as CoSaMP, have also been proposed (e.g., [117, 172]).
- Improved versions of PCP that allow for exact recovery even if some of the constraints of Theorem 19.2 are relaxed have also been developed (see, e.g., [73]). Fusions of PCP with matrix completion and compressed sensing are possible, in the sense that only a subset of the entries of M is available and/or linear measurements of the matrix in a compressed sensing fashion can be used instead of matrix entries, for example, [172, 177]. Moreover, stable versions of PCP dealing with noise have also been investigated, for example, [188].

19.10.3 APPLICATIONS OF MATRIX COMPLETION AND ROBUST PCA

The number of applications in which these techniques are involved is ever increasing and their extensive presentation is beyond the scope of this book. Next, some key applications are selectively discussed in order to reveal the potential of these methods and at the same time to assist the reader in better understanding the underlying notions.

Matrix completion

A typical application where the matrix completion problem arises is in the *collaborative filtering* task (e.g., [157]), which is essential for building up successful recommender systems. Let us consider that a group of individuals provide their ratings concerning products that they have enjoyed. Then, a matrix with ratings can be filled, where each row indexes a different individual and the columns index the products. As a popular example, take the case where the products are different movies. Inevitably, the associated matrix will be partially filled because it is not common that all customers have watched all the movies and submitted ratings for all of them. Matrix completion comes to provide an answer, potentially in the affirmative, to the following question: Can we predict the ratings that the users would give to films that they have not seen yet? This is the task of a recommender system in order to encourage

users to watch movies, which are likely to be of their preference. The exact objective of competition for the famous Netflix prize (http://www.netflixprize.com/) was the development of such a recommender system.

The aforementioned problem provides a good opportunity to build up our intuition about the matrix completion task. First, an individual's preferences or taste in movies are typically governed by a small number of factors, such as gender, the actors that appear in it, the continent of origin, and so on. As a result, a matrix fully filled with ratings is expected to be low rank. Moreover, it is clear that each user needs to have at least one movie rated in order to have any hope of filling out her/his ratings across all movies. The same is true for each movie. This requirement complies with the second assumption in Section 19.10.1, concerning uniqueness; that is, one needs to know at least one entry per row and column. Finally, imagine a single user who rates movies with criteria that are completely different from those used by the rest of the users. One could, for example, provide ratings at random or depending on, let's say, the first letter of the movie title. The ratings of this particular user cannot be described in terms of the singular vectors that model the ratings of the rest of the users. Accordingly, for such a case, the rank of the matrix increases by one and the user's preferences will be described by an extra set of left and right singular vectors. However, the corresponding left singular vector will comprise a single nonzero component, at the place corresponding to the row dedicated to this user, and the right singular vector will comprise her/his ratings normalized to unit norm. Such a scenario complies with the third point concerning the uniqueness in the matrix completion problem, as previously discussed. Unless all the ratings of the specific user are known, the matrix cannot get fully completed.

Other applications of matrix completion includes system identification [120], recovering structure from motion [44], multitask learning [10], and sensor network localization [128].

Robust PCA/PCP

In the collaborative filtering task, robust PCA offers an extra attribute compared to matrix completion, which can be proved very crucial in practice. The users are allowed to even tamper with some of the ratings without affecting the estimation of the low-rank matrix. This seems to be the case whenever the rating process involves many individuals in an environment, which is not strictly controlled, because some of them occasionally are expected to provide ratings in an ad hoc, or even malicious manner.

One of the first applications of PCP was in video surveillance systems (e.g., [35]) and the main idea behind it appeared to be popular and extendable to a number of computer vision applications. Take the example of a camera recording a sequence of frames consisting of a merely static background and a foreground with a few moving objects, for example, vehicles and/or individuals. A common task in surveillance video is to extract from the background the foreground, in order, for example, to detect any activity or to proceed with further processing such as face recognition. Suppose the successive frames are converted to vectors in lexicographic order and then are placed as columns in a matrix M. Due to the background, even though this may slightly vary due, for example, to changes in illumination, successive columns are expected to be highly correlated. As a result, the background contribution to the matrix M can be modeled as an approximately low-rank matrix L. On the other hand, the objects in the foreground appear as "anomalies" and correspond to only a fraction of pixels in each frame; that is, to a limited number of entries in each column of M. Moreover, due to the motion of the foreground objects, the positions of these anomalies are likely to change from one column of M to the next. Therefore, they can be modeled as a sparse matrix S.

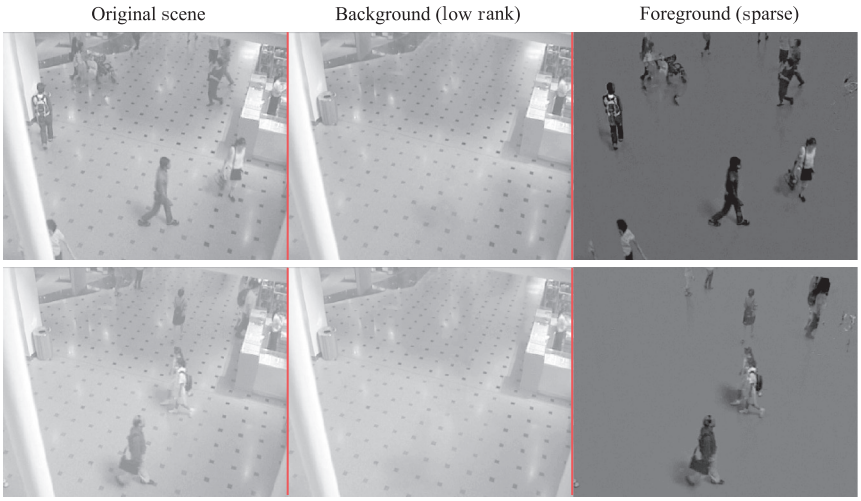

FIGURE 19.21

Background-Foreground separation via PCP.

Next, the above discussed philosophy is applied to a video acquired from a shopping mall surveillance camera, [121], with the corresponding PCP task being solved with a dedicated accelerated proximal gradient algorithm, [122]. The results are shown in Figure 19.21. In particular, two randomly selected frames are depicted together with the corresponding columns of the matrices L and S reshaped back to pictures.

19.11 A CASE STUDY: fMRI DATA ANALYSIS

In the brain, tasks involving action, perception, cognition, and so forth, are performed via the simultaneous activation of a number of the so-called *functional brain networks* (FBN), which are engaged in proper interactions in order to effectively execute the task. Such networks are usually related to low-level brain functions and they are defined as a number of *segregated* specialized small brain regions, potentially distributed over the whole brain. For each FBN, the involved segregated brain regions define the *spatial map*, which characterizes the specific FBN. Moreover, these brain regions, irrespective of their anatomical proximity or remoteness, exhibit *strong* functional connectivity, which is expressed as strong coherence in the activation timepatterns of these regions. Examples of such functional brain networks are the visual, sensorimotor, auditory, default-mode, dorsal attention, and executive control networks [139].

Functional magnetic resonance imaging (fMRI) [123] is a powerful noninvasive tool for detecting brain activity along time. Most commonly, it is based on *blood oxygenation level-dependent* (BOLD)

contrast, which translates to detecting localized changes in the hemodynamic flow of oxygenated blood in activated brain areas. This is achieved by exploiting the different magnetic properties of oxygen-saturated versus oxygen-desaturated hemoglobin. The detected fMRI signal is recorded in both the spatial (3-D) as well as the temporal (1-D) domain. The spatial domain is segmented with a 3-D grid to elementary cubes of edge size 3-5 mm, which are named *voxels*. Indicatively, a complete volume scan typically consists of $64 \times 64 \times 48$ voxels and it is acquired in one or two seconds, [123]. Relying on adequate postprocessing, which effectively compensates for possible time lags and other artifacts, [123], it is fairly accurate to assume that each acquisition is performed instantly. The, say l, in total voxel values, corresponding to a single scan, are collected in a flattened (row) 1-D vector, $\boldsymbol{x}_n \in \mathbb{R}^l$. Considering $n = 1, 2, \ldots, N$, successive acquisitions, the full amount of data is collected in a data matrix $X \in \mathbb{R}^{N \times l}$. Thus, each column, $i = 1, 2, \ldots, l$, of X represents the evolution in time of the values of the corresponding ith voxel. Each row, $n = 1, 2, \ldots, N$, corresponds to the activation pattern, at the corresponding time n, over all l voxels.

The recorded voxel values result from the cumulative contribution of several FBNs, where each one of them is activated following certain time patterns, depending on the tasks that the brain is performing. The above can be mathematically modeled according to the following factorization of the data matrix:

$$X = \sum_{j=1}^{m} \boldsymbol{a}_j \boldsymbol{z}_j^T := AZ, \tag{19.123}$$

where $\boldsymbol{z}_j \in \mathbb{R}^l$ is a sparse vector of latent variables, representing the spatial map of the jth FBN having nonzero values only in positions that correspond to brain regions associated with the specific FBN, and $\boldsymbol{a}_j \in \mathbb{R}^N$ represents the activation *time course* of the respective FBN. The model assumes that m FBNs have been activated. In order to understand better the previous model, take as an example the extreme case where only one set of brain regions (one FBN) is activated. Then, matrix X is written as

$$X = \boldsymbol{a}_1 \boldsymbol{z}_1^T := \begin{bmatrix} a_1(1) \\ a_1(2) \\ \vdots \\ a_1(N) \end{bmatrix} \underbrace{[\ldots, *, \ldots, *, \ldots, *, \ldots,]}_{l \text{ (voxels)}},$$

where $*$ denotes a nonzero element (active voxel in the FBN) and the dots zero ones. Observe that according to this model, all nonzero elements in the nth row of X result from the nonzero elements of $\boldsymbol{z}_1$ multiplied by the *same* number, $a_1(n)$, $n = 1, 2, \ldots, N$. If now two FBNs are active, the model for the data matrix becomes

$$X = \boldsymbol{a}_1 \boldsymbol{z}_1^T + \boldsymbol{a}_2 \boldsymbol{z}_2^T = [\boldsymbol{a}_1, \boldsymbol{a}_2] \begin{bmatrix} \boldsymbol{z}_1^T \\ \boldsymbol{z}_2^T \end{bmatrix}.$$

Obviously, for m FBNs, Eq. (19.123) results.

One of the major goals of the fMRI analysis is to detect, study, and characterize the different FBNs and to relate them to particular mental and physical activities. In order to achieve this, the subject (person) subjected to fMRI is presented with carefully designed experimental procedures, so that the activation of the FBNs will be as controlled as possible.

ICA has been successfully employed for fMRI unmixing, that is, for estimating matrices A and Z above. If we consider each column of X to be a realization of a random vector $\mathbf{x}$, the fMRI data generation mechanism can be modeled to follow the classical ICA latent model, that is, $\mathbf{x} = A\mathbf{s}$, where the components of $\mathbf{s}$ are statistically independent and A is an unknown mixing matrix. The goal of ICA is to recover the unmixing matrix, W and Z. Matrix A is then obtained from W. The use of ICA in the fMRI task could be justified by the following argument. Nonzero elements of Z in the same column contribute to the formation of a single element of X, for each time instant, n, and correspond to different FBNs. Thus, they are assumed to correspond to two statistically independent sources.

As a result of the application of ICA on X, one hopes that each row of the obtained matrix Z could be associated with an FBN; that is, to a spatial activity map. Furthermore, the corresponding column of A could represent the respective time activation pattern.

This approach will be applied next for the case of the following experimental procedure, [57]: A visual pattern was presented to the subject, in which an 8 Hz reversing black and white checkerboard was shown intermittently in the left and right visual ends for 30 s at a time. This is a typical block design paradigm in fMRI, consisting of three different conditions to which the subject is exposed. Checkerboard on the left (red block) checkerboard on the right (black block) and no visual stimulus (white block). The subject was instructed to focus on the cross at the central during the full time of the experiment, Figure 19.22. More details about the scanning procedure and the preprocessing of the data can be found in [57]. The Group ICA of fMRI Toolbox (GIFT)[10] simulation tool was used.

When ICA is performed on the obtained data set,[11] the aforementioned matrices Z and A are computed. Ideally, at least some of the rows of Z should constitute spatial maps of the true FBNs, and the corresponding columns of A should represent the activation patterns of the respective FBNs, which correspond to the specific experimentation procedure.

The news is good, as shown in Figure 19.23. In particular, Figures 19.23a and 19.23b show two time courses (columns of A). For each one of them, one section of the associated spatial map (corresponding

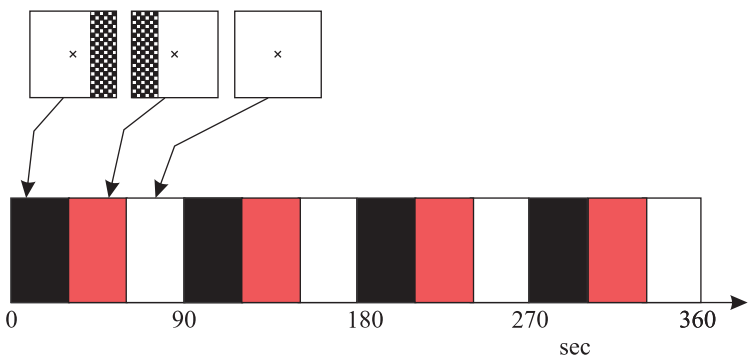

FIGURE 19.22

The fMRI experimental procedure used.

[10] It can be obtained from http://mialab.mrn.org/software/gift/.
[11] The specific data set is provided as test data with GIFT.

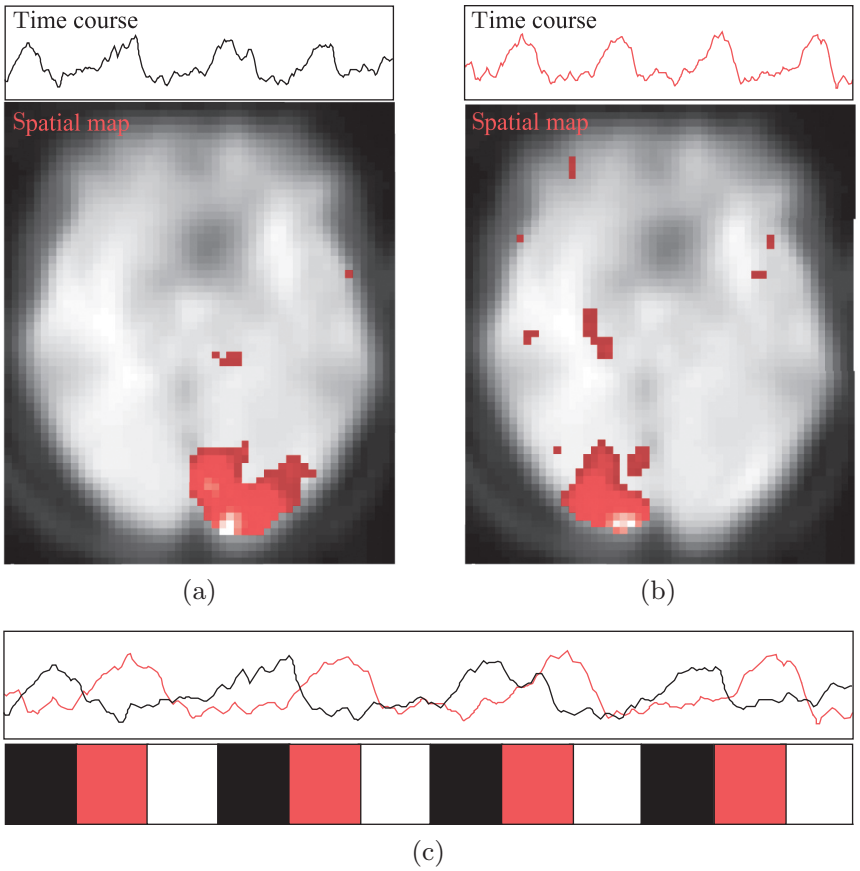

FIGURE 19.23

The two time courses follow well the fMRI experimental setup used.

row of Z) is considered, which represents voxels of a slice in the brain. The areas that are activated (red) are those corresponding to the left (a) and to the right (b) visual cortex, which are the regions of the brain responsible for processing visual information. Activation of this part should be expected according to the characteristics of the specific experimentation procedure. More interesting, as it is seen in Figure 19.23c, the two activation patterns, as represented by the two time courses, follow closely the two different conditions; namely the checkerboard to be placed on the left or on the right from the point the subject is focusing on.

Besides ICA, alternative methods, discussed in this chapter, can also be used, which can better exploit the low-rank nature of X. Dictionary learning is a promising candidate leading to notably good results, see, for example, [11, 109, 171].

PROBLEMS

19.1 Show that the second principle component in PCA is given as the eigenvector corresponding to the second largest eigenvalue.

19.2 Show that the pair of directions, associated with CCA, which maximize the respective correlation coefficient, satisfy the following pair of relations,

$$\Sigma_{xy}\boldsymbol{u}_y = \lambda\,\Sigma_{xx}\boldsymbol{u}_x,$$
$$\Sigma_{yx}\boldsymbol{u}_x = \lambda\,\Sigma_{yy}\boldsymbol{u}_y.$$

19.3 Establish the arguments that verify the convergence of the k-SVD.

19.4 Prove that Eqs. (19.79) and (19.85) are the same.

19.5 Show that the ML PPCA tends to PCA as $\sigma^2 \to 0$.

19.6 Show Eqs. (19.87)–(19.88).

19.7 Show Eq. (19.98).

19.8 Show that the number of degrees of freedom of a rank r matrix is equal to $r(l_1 + l_2) - r^2$.

MATLAB Exercises

19.9 This exercise reproduces the results of Example 19.1. Download the faces from this book's website and read them one by one using the imread.m MATLAB function. Store them as columns in a matrix X. Then, compute and subtract the mean in order for the rows to become zero-mean.

A direct way to compute the eigenvectors (eigenfaces) would be to use the svd.m MATLAB function in order to perform an SVD to the matrix X, that is $X = UDV^T$. In this way, the eigenfaces are the columns of U. However, this needs a lot of computational effort because X has too many rows. Alternatively, you can proceed as follows. First compute the product $A = X^TX$ and then the SVD of A (using the SVD.m MATLAB function) in order to compute the right singular vectors of X via $A = VD^2V^T$. Then calculate each eigenface according to $\boldsymbol{u}_i = \frac{1}{\sigma_i}X\boldsymbol{v}_i$, where σ_i is the ith singular value of the SVD. In the sequel, select one face at random in order to reconstruct it using the first 5, 30, 100, and 600 eigenvectors.

19.10 Recompute the eigenfaces as in the MATLAB Exercise 19.9, using all the face images apart from one, that you choose. Then reconstruct that face that did not take part in the computation of the eigenfaces, using the first 300 and 1000 eigenvectors. Is the reconstructed face anywhere close to the true one?

19.11 Download the fast ICA MATLAB software package from http://research.ics.aalto.fi/ica/fastica/ in order to reproduce the results of the cocktail party example described in Subsection 19.5.5. The two voice and the music signals can be downloaded from this book's website and read using the wavread.m MATLAB function. Generate a random mixing matrix A, (3×3) and produce with it the 3 mixture signals. Each one of them simulates the signal received in each microphone. Then, apply FastICA in order to estimate the source signals. Use the MATLAB function wavplay.m in order to listen to the original signals, to the mixtures, and the recovered ones. Repeat the previous steps performing PCA instead of ICA and compare the results.

19.12 This exercise reproduces the dictionary learning-based denoising Example 19.5. The image depicting the boat can be obtained from this book's website. Moreover, the k-SVD either need

to be implemented according to Section 19.6 or an implementation available on the web can be downloaded and used, e.g., http://www.cs.technion.ac.il/~elad/software/.

Then the next steps to be followed are: First, extract from the image all the possible sliding patches of size 12×12 using the im2col.m MATLAB function and store them as columns in a matrix X. Using this matrix, train an overcomplete dictionary, Ψ, of size, (144×196), for 100 k-SVD iterations with $T_0 = 5$. For the first iteration, the initial dictionary atoms are drawn from a zero-mean Gaussian distribution and then normalized to unit norm. As a next step, de-noise each image patch separately. In particular, assuming that $\boldsymbol{y}_i$ is the ith patch reshaped in column vector use the OMP (Section 10.2.1) in order to estimate a sparse vector $\boldsymbol{\theta}_i, \ \in \mathbb{R}^{196}$ with $\|\boldsymbol{\theta}_i\|_0 = 5$, such that $\left\|\boldsymbol{y}_i - A\boldsymbol{\theta}_i\right\|$ is small. Then, $\hat{\boldsymbol{y}}_i = \Psi\boldsymbol{\theta}_i$ is the ith de-noised patch. Finally, average the values of the overlapped patches in order to form the full de-noised image.

19.13 Download one of the videos (they are provided in the form of a sequence of bitmap images) from http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html.

In Section 19.10.3, the "shopping center" bitmap image sequence has been used. Read one by one the bitmap images using the imread.m MATLAB function then convert them from color to grayscale using rgb2gray.m and finally store them as columns in a matrix X. Download one of the MATLAB implementations of an algorithm performing the robust PCA task from http://perception.csl.illinois.edu/matrix-rank/sample_code.html.

The "Accelerated Proximal Gradient" method and the accompanied proximal_gradient _rpca.m MATLAB function is a good and easy to use choice. Set $\lambda = 0.01$. Note, however, that depending on the video used, this regularization parameter might need to get fine-tuned.

REFERENCES

[1] D. Achlioptas, F. McSherry, Fast computation of low rank approximations, in: Proceedings of the ACM STOC Conference, 2001, pp. 611-618.

[2] T. Adali, H. Li, M. Novey, J.F. Cardoso, Complex ICA using nonlinear functions, IEEE Trans. Signal Process. 56 (9) (2008) 4356-4544.

[3] T. Adali, M. Anderson, G.S. Fu, Diversity in independent component and vector analyses: Identifiability, algorithms, and applications in medical imaging, IEEE Signal Process. Mag. 31 (3) (2014) 18-33.

[4] M. Aharon, M. Elad, A. Bruckstein, k-SVD: an algorithm for designing overcomplete dictionaries for sparse representation. IEEE Trans. Signal Process. 54 (11) (2006) 4311-4322.

[5] T.W. Anderson, Asymptotic theory for principal component analysis, Ann. Math. Stat. 34 (1963) 122-148.

[6] T.W. Anderson, An Introduction to Multivariate Analysis, second ed., John Wiley, New York, 1984.

[7] M. Anderson, X.L. Li, T. Adali, Joint blind source separation with multivariate Gaussian model: Algorithms and performance analysis, IEEE Trans. Signal Process. 60 (4) (2012) 2049-2055.

[8] C. Archambeau, F. Bach, Sparse probabilistic projections, in: D. Koller, D. Schuurmans, Y. Bengio, L. Bottou (Eds.), Neural Information Processing Systems, NIPS, Vancouver, Canada, 2008.

[9] J. Arenas-García, K.B. Petersen, G. Camps-Valls, L.K. Hansen, Kernel multivariate analysis framework for supervised subspace learning, IEEE Signal Process. Mag. 30 (4) (2013) 16-29.

[10] A. Argyriou, T. Evgeniou, M. Pontil, Multi-task feature learning, in: Advances in Neural Information Processing Systems, vol. 19, MIT Press, Cambridge, MA, 2007.

[11] V. Abolghasemi, S. Ferdowsi, S. Sanei, Fast and incoherent dictionary learning algorithms with application to fMRI, Signal Image Video Process. 2013. DOI: 10.1007/s11760-013-0429-2.

[12] G.I. Allen, Sparse and Functional Principal Components Analysis, 2013, arXiv preprint arXiv:1309.2895.

[13] F.R. Bach, M.I. Jordan, Kernel independent component analysis, J. Mach. Learn. Res. 3 (2002) 1-48.
[14] F. Bach, M. Jordan, A probabilistic interpretation of canonical correlation analysis, Technical Report 688, University of Berkeley, 2005.
[15] E. Barshan, A. Ghodsi, Z. Azimifar, M.Z. Jahromi, Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds, Pattern Recognit. 44 (2011) 1357-1371.
[16] H.B. Barlow, Unsupervised learning, Neural Comput. 1 (1989) 295-311.
[17] A.J. Bell, T.J. Sejnowski, An information maximization approach to blind separation and blind deconvolution, Neural Comput. 7 (1995) 1129-1159.
[18] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (6) (2003) 1373-1396.
[19] E. Benetos, M. Kotti, C. Kotropoulos, Applying supervised classifiers based on non-negative matrix factorization to musical instrument classification, in: Proceedings IEEE International Conference on Multimedia and Expo, Toronto, Canada, 2006, pp. 2105-2108.
[20] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, M. Quimet, Out of sample extensions for LLE, Isomap, MDS, eigenmaps and spectral clustering, in: S. Thrun, L. Saul, B. Schölkopf (Eds.), Advances in Neural Information Processing Systems Conference, MIT Press, Cambridge, MA, 2004.
[21] M. Berry, S. Dumais, G. O'Brie, Using linear algebra for intelligent information retrieval, SIAM Rev. 37 (1995) 573-595.
[22] A. Beygelzimer, S. Kakade, J. Langford, Cover trees for nearest neighbor, in: Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, 2006.
[23] C.M. Bishop, Variational principal components, in: Proceedings 9th International Conference on Artificial Neural Networks, ICANN, vol. 1, 1999, pp. 509-514.
[24] M. Borga, Canonical correlation analysis: A tutorial, Technical Report, 2001, www.imt.liu.se/~magnus/cca/tutorial/tutorial.pdf.
[25] M. Brand, Charting a manifold, in: Advances in Neural Information Processing Systems, vol. 15, MIT Press, Cambridge, MA, 2003, pp. 985-992.
[26] J.-P. Brunet, P. Tamayo, T.R. Golub, J.P. Mesirov, Meta-genes and molecular pattern discovery using matrix factorization, Proc. Natl. Acad. Sci. 101 (2) (2004) 4164-4169.
[27] C.J.C. Burges, Geometric methods for feature extraction and dimensional reduction: A guided tour, Technical Report MSR-TR-2004-55, Microsoft Research, 2004.
[28] F. Bach, R. Jenatton, J. Mairal, G. Obozinski, Structured sparsity through convex optimization, Stat. Sci. 27 (4) (2012) 450-468.
[29] D. Cai, X. He, Orthogonal locally preserving indexing, in: Proceedings 28th Annual International Conference on Research and Development in Information Retrieval, 2005.
[30] J.-F. Cai, E.J. Candès, Z. Shen, A singular value thresholding algorithm for matrix completion, SIAM J. Optim. 20 (4) (2010) 1956-1982.
[31] F. Camastra, Data dimensionality estimation methods: A survey, Pattern Recognit. 36 (2003) 2945-2954.
[32] E.J. Candès, B. Recht, Exact matrix completion via convex optimization, Found. Comput. Math. 9 (6) (2009) 717-772.
[33] E.J. Candès, P. Yaniv, Matrix completion with noise, Proc. IEEE 98 (6) (2010) 925-936.
[34] E.J. Candès, T. Tao, The power of convex relaxation: Near-optimal matrix completion, IEEE Trans. Inform. Theory. 56 (3) (2010) 2053-2080.
[35] E.J. Candès, X. Li, Y. Ma, J. Wright, Robust principal component analysis J. ACM, 58 (3) (2011) 1-37.
[36] J.F. Cardoso, Infomax and maximum likelihood for blind source separation, IEEE Signal Process. Lett. 4 (1997) 112-114.
[37] J.-F. Cardoso, Blind signal separation: Statistical principles, Proc. IEEE 9 (10) (1998) 2009-2025.

[38] J.-F. Cardoso, High-order contrasts for independent component analysis, Neural Comput. 11 (1) (1999) 157-192.
[39] L. Carin, R.G. Baraniuk, V. Cevher, D. Dunson, M.I. Jordan, G. Sapiro, M.B. Wakin, Learning low-dimensional signal models, IEEE Signal Process. Mag. 34 (2) (2011) 39-51.
[40] V. Casteli, A. Thomasian, C.-S. Li, CSVD: Clustering and singular value decomposition for approximate similarity searches in high-dimensional space, IEEE Trans. Knowl. Data Eng. 15 (3) (2003) 671-685.
[41] V. Cevher, P. Indyk, L. Carin, R.G. Baraniuk, Sparse signal recovery and acquisition with graphical models, IEEE Signal Process. Mag. 27 (6) (2010) 92-103.
[42] V. Chandrasekaran, S. Sanghavi, P.A. Parrilo, A.S. Willsky, Rank-sparsity incoherence for matrix decomposition, SIAM J. Optim. 21 (2) (2011) 572-596.
[43] C. Chatfield, A.J. Collins, Introduction to Multivariate Analysis, Chapman Hall, London, 1980.
[44] P. Chen, D. Suter, Recovering the missing components in a large noisy low-rank matrix: Application to SFM, IEEE Trans. Pattern Anal. Mach. Intell. 26 (8) (2004) 1051-1063.
[45] M. Chen, J. Silva, J. Paisley, C. Wang, D. Dunson, L. Carin, Compressive sensing on manifolds using nonparametric mixture of factor analysers: Algorithms and performance bounds, IEEE Trans. Signal Process. 58 (12) (2010) 6140-6155.
[46] C. Chen, B. He, X. Yuan, Matrix completion via an alternating direction method, IMA J. Numer. Anal. 32 (2012) 227-245.
[47] Y. Chi, Y.C. Eldar, R. Calderbank, PETRELS: Subspace estimation and tracking from partial observations, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012, pp. 3301-3304.
[48] S. Chouvardas, Y. Kopsinis, S. Theodoridis, An adaptive projected subgradient based algorithm for robust subspace tracking, in: Proc. International Conference on Acoustics Speech and Signal Processing, ICASSP, Florence, Italy, May 4-9, 2014.
[49] S. Chouvardas, Y. Kopsinis, S. Theodoridis, Robust subspace tracking with missing entries: The set-theoretic approach, IEEE Trans. Signal Process., to appear, 2015.
[50] M. Chu, F. Diele, R. Plemmons, S. Ragni, Optimality, Computation and Interpretation of the Nonnegative Matrix Factorization, 2004, available at http://www.wfu.edu/~plemmons.
[51] A. Cichoki, Unsupervised learning algorithms and latent variable models: PCA/SVD, CCA, ICA, NMF, in: R. Chelappa, S. Theodoridis (Eds.), E-Reference for Signal Processing, Academic Press, Boston, 2014.
[52] N.M. Correa, T. Adal, Y.-Q. Li, V.D. Calhoun, Canonical correlation analysis for group fusion and data inferences, IEEE Signal Process. Mag. 27 (4) (2010) 39-50.
[53] P. Comon, Independent component analysis: A new concept, Signal Process. 36 (1994) 287-314.
[54] P. Comon, C. Jutten, Handbook of Blind Source Separation: Independent Component Analysis and Applications, Academic Press, 2010.
[55] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., MIT Press/McGraw-Hill, Cambridge, MA, 2001.
[56] T. Cox, M. Cox, Multidimensional Scaling, Chapman & Hall, London, 1994.
[57] V. Calhoun, T. Adali, G. Pearlson, J. Pekar, A method for making group inferences from functional MRI data using independent component analysis, Hum. Brain Mapp. 14 (3) (2001) 140-151.
[58] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, R. Harshman, Indexing by latent semantic analysis, J. Soc. Inform. Sci. 41 (1990) 391-407.
[59] W.R. Dillon, M. Goldstein, Multivariable Analysis Methods and Applications, John Wiley, New York, 1984.
[60] J.P. Dmochowski, P. Sajda, J. Dias, L.C. Parra, Correlated components of ongoing EEG point to emotionally laden attention—a possible marker of engagement? Front. Hum. Neurosci. 6 (2012). DOI: 10.3389/fnhum.2012.00112.

[61] D.L. Donoho, C.E. Grimes, When does ISOMAP recover the natural parameterization of families of articulated images? Technical Report 2002-27, Department of Statistics, Stanford University, 2002.
[62] D. Donoho, V. Stodden, When does nonnegative matrix factorization give a correct decomposition into parts? in: S. Thrun, L. Saul, B. Schölkopf (Eds.), Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 2004.
[63] S.C. Douglas, S. Amari, Natural gradient adaptation, in: S. Haykin (Ed.), Unsupervised Adaptive Filtering, Part I: Blind Source Separation, John Wiley & Sons, New York, 2000, pp. 13-61.
[64] X. Doukopoulos, G.V. Moustakides, Fast and stable subspace tracking, IEEE Trans. Signal Process. 56 (4) (2008) 1452-1465.
[65] V. De Silva, J.B. Tenenbaum, Global versus local methods in nonlinear dimensionality reduction, in: S. Becker, S. Thrun, K. Obermayer (Eds.), Advances in Neural Information Processing Systems, vol. 15, MIT Press, Cambridge, MA, 2003, pp. 721-728.
[66] M. Elad, M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, IEEE Trans. Image Process. 15 (12) (2006) 3736-3745.
[67] M. Elad, Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing, Springer, New York, 2010.
[68] K. Engan, S.O. Aase, J.H.A. Husy, Multi-frame compression: theory and design, Signal Process. 80 (10) (2000) 2121-2140.
[69] M. Fazel, H. Hindi, S. Boyd, Rank minimization and applications in system theory, in: Proceedings American Control Conference, vol. 4, 2004, pp. 3273-3278.
[70] D.J. Field, What is the goal of sensory coding? Neural Comput. 6 (1994) 559-601.
[71] R. Foygel, N. Srebro, Concentration-based guarantees for low-rank matrix reconstruction, in: Proceedings, 24th Annual Conference on Learning Theory (COLT), 2011.
[72] S. Gandy, B. Recht, I. Yamada, Tensor completion and low-n-rank tensor recovery via convex optimization, Inverse Prob. 27 (2) (2011) 1-19.
[73] A. Ganesh, J. Wright, X. Li, E.J. Candès, Y. Ma, Dense error correction for low-rank matrices via principal component pursuit, in: Proceedings IEEE International Symposium on Information Theory, 2010, pp. 1513-1517.
[74] Z. Ghahramani, M. Beal, Variational inference for Bayesian mixture of factor analysers, in: Advances in Neural Information Processing Systems, vol. 12, MIT Press, Cambridge, MA, 2000, pp. 449-455.
[75] M. Girolami, Self-organizing Neural Networks, Independent Component Analysis and Blind Source Separation, Springer-Verlag, New York, 1999.
[76] M. Girolami, A variational method for learning sparse and overcomplete representations, Neural Comput. 13 (2001) 2517-2532.
[77] G.H. Golub, C.F. Van Loan, Matrix Computations, Johns Hopkins Press, Baltimore, 1989.
[78] S. Gould, The Mismeasure of Man, second ed., Norton, New York, 1981.
[79] D. Gross, Recovering low-rank matrices from few coefficients in any basis, IEEE Trans. Inform. Theory 57 (3) (2011) 1548-1566.
[80] J. He, L. Balzano, J. Lui, Online robust subspace tracking from partial information, 2011, arXiv preprint arXiv:1109.3827.
[81] J. Ham, D.D. Lee, S. Mika, B. Schölkopf, A kernel view of the dimensionality reduction of manifolds, in: Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004, pp. 369-376.
[82] D.R. Hardoon, S. Szedmak, J. Shawe-Taylor, Canonical correlation analysis: an overview with application to learning methods, Neural Comput. 16 (2004) 2639-2664.
[83] D.R. Hardoon, J. Shawe-Taylor, Sparse canonical correlation analysis, Mach. Learn. 83 (3) (2011) 331-353.
[84] S. Haykin, Neural Networks: A Comprehensive Foundation, second ed., Prentice Hall, Upper Saddle River, NJ, 1999.

[85] J. He, L. Balzano, J. Lui, Online robust subspace tracking from partial information, 2011, arXiv preprint arXiv:1109.3827.
[86] J. Hérault, C. Jouten, B. Ans, Détection de grandeurs primitive dans un message composite par une architecture de calcul neuroimimétique en apprentissage non supervisé, in: Actes du Xème colloque GRETSI, Nice, France, 1985, pp. 1017-1022.
[87] N. Hjort, C. Holmes, P. Muller, S. Walker, Bayesian Nonparametrics, Cambridge University Press, Cambridge, 2010.
[88] H. Hotelling, Analysis of a complex of statistical variables into principal components, J. Educ. Psychol. 24 (1933) 417-441.
[89] H. Hotelling, Relations between two sets of variates, Biometrika 28 (34) (1936) 321-377.
[90] P.J. Huber, Projection pursuit, Ann. Stat. 13 (2) (1985) 435-475.
[91] M. Hubert, P.J. Rousseeuw, K. Vanden Branden, ROBPCA: a new approach to robust principal component analysis, Technometrics 47 (1) (2005) 64-79.
[92] A. Hyvärinen, Fast and robust fixed-point algorithms for independent component analysis, IEEE Trans. Neural Netw. 10 (3) (1999) 626-634.
[93] A. Hyvärinen, J. Karhunen, E. Oja, Independent Component Analysis, John Wiley, New York, 2001.
[94] X. He, P. Niyogi, Locally preserving projections, in: Proceedings Advances in Neural Information Processing Systems Conference, 2003.
[95] B.G. Huang, M. Ramesh, T. Berg, Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments, Technical Report, University of Massachusetts, Amherst, No. 07-49, 2007.
[96] R. Jenssen, Kernel entropy component analysis, IEEE Trans. Pattern Anal. Mach. Intell. 32 (5) (2010) 847-860.
[97] J.E. Jackson, A User's Guide to Principle Components, John Wiley, New York, 1991.
[98] I. Jolliffe, Principal Component Analysis, Springer-Verlag, New York, 1986.
[99] M.C. Jones, R. Sibson, What is projection pursuit? J. R. Stat. Soc. A 150 (1987) 1-36.
[100] C. Jutten, J. Herault, Blind separation of sources, Part I: an adaptive algorithm based on neuromimetic architecture, Signal Process. 24 (1991) 1-10.
[101] C. Jutten, Source separation: From dusk till dawn, in: Proceedings 2nd International Workshop on Independent Component Analysis and Blind Source Separation, ICA'2000, Helsinki, Finland, 2000, pp. 15-26.
[102] J. Karhunen, J. Joutsensalo, Generalizations of principal component analysis, optimization problems, and neural networks, Neural Netw. 8 (4) (1995) 549-562.
[103] J. Kettenring, Canonical analysis of several sets of variables, Biometrika 58 (3) (1971) 433-451.
[104] M.E. Khan, M. Marlin, G. Bouchard, K.P. Murphy, Variational bounds for mixed-data factor analysis, in: J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, A. Culotta (Eds.), Neural Information Processing Systems, NIPS, Vancouver, Canada, 2010.
[105] P. Kidmose, D. Looney, M. Ungstrup, M.L. Rank, D.P. Mandic, A study of evoked potentials from ear-EEG, IEEE Trans. Biomed. Eng. 60 (10) (2103) 2824-2830.
[106] A. Klami, S. Virtanen, S. Kaski, Bayesian canonical correlation analysis, J. Mach. Learn. Res. 14 (2013) 965-1003.
[107] O.W. Kwon, T.W. Lee, Phoneme recognition using the ICA-based feature extraction and transformation, Signal Process. 84 (6) (2004) 1005-1021.
[108] S.-Y. Kung, K.I. Diamantaras, J.-S. Taur, Adaptive principal component extraction (APEX) and applications, IEEE Trans. Signal Process. 42 (5) (1994) 1202-1217.
[109] Y. Kopsinis, H. Georgiou, S. Theodoridis, fMRI unmixing via properly adjusted dictionary learning, in: Proceedings of the 20th European Signal Processing Conference (EUSIPCO), 2012, pp. 61-65.

[110] P.L. Lai, C. Fyfe, Kernel and nonlinear canonical correlation analysis, Int. J. Neural Syst. 10 (5) (2000) 365-377.
[111] S. Lafon, A.B. Lee, Diffusion maps and coarse-graining: a unified framework for dimensionality reduction, graph partitioning and data set parameterization, IEEE Trans. Pattern Anal. Mach. Intell. 28 (9) (2006) 1393-1403.
[112] L.D. Lathauer, Signal Processing by Multilinear Algebra, Ph.D. Thesis, Faculty of Engineering, K.U. Leuven, Belgium, 1997.
[113] T.W. Lee, Independent Component Analysis: Theory and Applications, Kluwer, Boston, MA, 1998.
[114] M.H.C. Law, A.K. Jain, Incremental nonlinear dimensionality reduction by manifold learning, IEEE Trans. Pattern Anal. Mach. Intell. 28 (3) (2006) 377-391.
[115] D.D. Lee, S. Seung, Learning the parts of objects by nonnegative matrix factorization, Nature 401 (1999) 788-791.
[116] J.A. Lee, M. Verleysen, Nonlinear Dimensionality Reduction, Springer, New York, 2007.
[117] K. Lee, Y. Bresler, ADMiRA: atomic decomposition for minimum rank approximation, IEEE Trans. Inform. Theory 56 (9) (2010) 4402-4416.
[118] S. Lesage, R. Gribonval, F. Bimbot, L. Benaroya, Learning unions of orthonormal bases with thresholded singular value decomposition, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2005.
[119] M.S. Lewicki, T.J. Sejnowski, Learning overcomplete representations, Neural Comput. 12 (2000) 337-365.
[120] Z. Liu, L. Vandenberghe, Interior-Point Method for Nuclear Norm Approximation with Application to System Identification, SIAM J. Matrix Anal. Appl. 31 (3) (2010) 1235-1256.
[121] L. Li, W. Huang, I.-H. Gu, Q. Tian, Statistical modeling of complex backgrounds for foreground object detection, IEEE Trans. Image Process. 13 (11) (2004) 1459-1472.
[122] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, Y. Ma, Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix, in: Intl. Workshop on Comp. Adv. in Multi-Sensor Adapt. Processing, Aruba, Dutch Antilles, 2009.
[123] M.A. Lindquist, The statistical analysis of fMRI data, Stat. Sci. 23 (4) (2008) 439-464.
[124] G. Mateos, G.B. Giannakis, Robust PCA as bilinear decomposition with outlier-sparsity regularization, IEEE Trans. Signal Process. 60 (2012) 5176-5190.
[125] M. Mesbahi, G.P. Papavassilopoulos, On the rank minimization problem over a positive semidefinite linear matrix inequality, IEEE Trans. Autom. Control 42 (2) (1997) 239-243.
[126] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, 1995.
[127] L. Mackey, Deflation methods for sparse PCA, in: D. Koller, D. Schuurmans, Y. Bengio, L. Bottou (Eds.), Advances in Neural Information Processing Systems, vol. 21, 2009, pp. 1017-1024.
[128] G. Mao, B. Fidan, B.D.O. Anderson, Wireless sensor network localization techniques, Comput. Netw. 51 (10) (2007) 2529-2553.
[129] M. Mardani, G. Mateos, G.B. Giannakis, Subspace Learning and Imputation for Streaming Big Data Matrices and Tensors, 2014, arXiv preprint arXiv:1404.4667.
[130] J. Mairal, M. Elad, G. Sapiro, Sparse Representation for Color Image Restoration, IEEE Trans. Image Process. 17 (1) (2008) 53-69.
[131] J. Mairal, F. Bach, J. Ponce, G. Sapiro, Online learning for matrix factorization and sparse coding, J. Mach. Learn. Res. 11 (2010).
[132] M. Novey, T. Adali, Complex ICA by negentropy maximization, IEEE Trans. Neural Netw. 19 (4) (2008) 596-609.
[133] M.A. Nicolaou, S. Zafeiriou, M. Pantic, A unified framework for probabilistic component analysis, in: European Conference Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD'14), Nancy, France, 2014.

[134] B.A. Olshausen, B.J. Field, Sparse coding with an overcomplete basis set: a strategy employed by v1, Vis. Res. 37 (1997) 3311-3325.
[135] P. Paatero, U. Tapper, R. Aalto, M. Kulmala, Matrix factorization methods for analysis diffusion battery data, J. Aerosol Sci. 22 (Supplement 1) (1991) 273-276.
[136] P. Paatero, U. Tapper, Positive matrix factor model with optimal utilization of error, Environmetrics 5 (1994) 111-126.
[137] K. Pearson, On lines and planes of closest fit to systems of points in space, in: The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, Sixth Series, vol. 2, 1901, pp. 559-572.
[138] A.T. Poulsen, S. Kamronn, L.C. Parra, L.K. Hansen, Bayesian correlated component analysis for inference of joint EEG activation, in: 4th International Workshop on Pattern Recognition in Neuroimaging, 2014.
[139] V. Perlbarg, G. Marrelec, Contribution of exploratory methods to the investigation of extended large-scale brain networks in functional MRI: methodologies, results, and challenges, Int. J. Biomed. Imaging 2008 (2008) 1-14.
[140] H. Qui, E.R. Hancock, Clustering and embedding using commute times, IEEE Trans. Pattern Anal. Mach. Intell. 29 (11) (2007) 1873-1890.
[141] B. Recht, A simpler approach to matrix completion, J. Mach. Learn. Res. 12 (2011) 3413-3430.
[142] R. Rosipal, L.J. Trejo, Kernel partial least squares regression in reproducing kernel Hilbert spaces, J. Mach. Learn. Res. 2 (2001) 97-123.
[143] R. Rosipal, N. Krämer, Overview and recent advances in partial least squares, in: C. Saunders, M. Grobelnik, S. Gunn, J. Shawe-Taylor (Eds.), Subspace, Latent Structure and Feature Selection, Springer, New York, 2006.
[144] S. Roweis, EM algorithms for PCA and SPCA, in: M.I. Jordan, M.J. Kearns, S.A. Solla (Eds.), Advances in Neural Information Processing Systems, vol. 10, MIT Press, Cambridge, MA, 1998, pp. 626-632.
[145] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323-2326.
[146] D.B. Rubin, D.T. Thayer, EM algorithm for ML factor analysis, Psychometrika 47 (1) (1982) 69-76.
[147] C.A. Rencher, Multivariate Statistical Inference and Applications, John Wiley & Sons, New York, 2008.
[148] S. Sanei, Adaptive Processing of Brain Signals, John Wiley, New York, 2013.
[149] L.K. Saul, S.T. Roweis, An introduction to locally linear embedding, http://www.cs.toronto.edu/~~roweis/lle/papers/lleintro.pdf.
[150] N. Sebro, T. Jaakola, Weighted low-rank approximations, in: Proceedings of the ICML Conference, 2003, pp. 720-727.
[151] B. Schölkopf, A. Smola, K.R. Muller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (1998) 1299-1319.
[152] F. Seidel, C. Hage, M. Kleinsteuber, pROST: A smoothed ℓ_p-norm robust online subspace tracking method for background subtraction in video, in: Machine Vision and Applications, 2013, pp. 1-14.
[153] F. Sha, L.K. Saul, Analysis and extension of spectral methods for nonlinear dimensionality reduction, in: Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 2005.
[154] Y. Shuicheng, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: a general framework for dimensionality reduction, IEEE Trans. Pattern Anal. Mach. Intell. 29 (1) (2007) 40-51.
[155] M. Signoretto, R. Van de Plas, B. De Moor, J.A.K. Suykens, Tensor versus matrix completion: a comparison with application to spectral data, IEEE Signal Process. Lett. 18 (7) (2011) 403-406.
[156] P. Smaragdis, J.C. Brown, Nonnegative matrix factorization for polyphonic music transcription, in: Proceedings IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2003.
[157] X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, Adv. Artif. Intell. 2009 (2009) 1-19.

[158] S. Sra, I.S. Dhillon, Non-negative matrix approximation: algorithms and applications, Technical Report TR-06-27, University of Texas at Austin, 2006.
[159] C. Spearman, The proof and measurement of association between two things, Am. J. Psychol. 100 (3-4) (1987) 441-471 (republished).
[160] G.W. Stewart, An updating algorithm for subspace tracking, IEEE Trans. Signal Process. 40 (6) (1992) 1535-1541.
[161] A. Szymkowiak-Have, M.A. Girolami, J. Larsen, Clustering via kernel decomposition, IEEE Trans. Neural Netw. 17 (1) (2006) 256-264.
[162] J. Sun, S. Boyd, L. Xiao, P. Diaconis, The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem, SIAM Rev. 48 (4) (2006) 681-699.
[163] J.B. Tenenbaum, V. De Silva, J.C. Langford, A global geometric framework for dimensionality reduction, Science 290 (2000) 2319-2323.
[164] S. Theodoridis, K. Koutroumbas, Pattern Recognition, fourth ed., Academic Press, Boston, 2009.
[165] M.E. Tipping, C.M. Bishop, Probabilistic principal component analysis, J. R. Stat. Soc. B 21 (3) (1999) 611-622.
[166] M.E. Tipping, C.M. Bishop, Mixtures probabilistic principal component analysis, Neural Comput. 11 (2) (1999) 443-482.
[167] K.C. Toh, S. Yun, An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems, Pac. J. Optim. 6 (2010) 615-640.
[168] J.A. Tropp, Literature survey: Nonnegative matrix factorization, Unpublished note, 2003, http://www.personal.umich.edu/~jtropp/.
[169] C.G. Tsinos, A.S. Lalos, K. Berberidis, Sparse subspace tracking techniques for adaptive blind channel identification in OFDM systems, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012, pp. 3185-3188.
[170] N. Ueda, R. Nakano, Z. Ghahramani, G.E. Hinton, SMEM algorithm for mixture models, Neural Comput. 12 (9) (2000) 2109-2128.
[171] G. Varoquaux, A. Gramfort, F. Pedregosa, V. Michel, B. Thirion, Multi-subject dictionary learning to segment an atlas of brain spontaneous activity, in: Information Processing in Medical Imaging, Springer, Berlin/Heidelberg.
[172] A.E. Waters, A.C. Sankaranarayanan, R.G. Baraniuk, SpaRCS: recovering low-rank and sparse matrices from compressive measurements, in: Advances in Neural Information Processing Systems (NIPS), Granada, Spain, 2011.
[173] K.Q. Weinberger, L.K. Saul, Unsupervised learning of image manifolds by semidefinite programming, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, Washington, DC, USA, 2004, pp. 988-995.
[174] J. Westerhuis, T. Kourti, J. MacGregor, Analysis of multiblock and hierarchical PCA and PLS models, J. Chemometr. 12 (1998) 301-321.
[175] H. Wold, Nonlinear estimation by iterative least squares procedures, in: F. David (Ed.), Research Topics in Statistics, John Wiley, New York, 1966, pp. 411-444.
[176] J. Wright, Y. Peng, Y. Ma, A. Ganesh, S. Rao, Robust principal component analysis: exact recovery of corrupted low-rank matrices by convex optimization, in: Neural Information Processing Systems (NIPS), 2009.
[177] J. Wright, A. Ganesh, K. Min, Y. Ma, Compressive Principal Component Pursuit, 2012, arXiv:1202.4596.
[178] D. Weenink, Canonical Correlation Analysis, Institute of Phonetic Sciences, University of Amsterdam, Proceedings, vol. 25, 2003, pp. 81-99.
[179] D.M. Witten, R. Tibshirani, T. Hastie, A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis, Biostatistics 10 (3) (2009) 515-534.
[180] L. Wolf, T. Hassner, Y. Taigman, Effective unconstrained face recognition by combining multiple descriptors and learned background statistics, IEEE Trans. Pattern Anal. Mach. Intell. 33 (10) (2011) 1978-1990.

[181] W. Xu, X. Liu, Y. Gong, Document clustering based on nonnegative matrix factorization, in: Proceedings 26th Annual International ACM SIGIR Conference, ACM Press, New York, 2003, pp. 263-273.
[182] H. Xu, C. Caramanis, S. Sanghavi, Robust PCA via outlier pursuit, IEEE Trans. Inform. Theory 58 (5) (2012) 3047-3064.
[183] J. Ye, Generalized low rank approximation of matrices, in: Proceedings of the 21st International Conference on Machine Learning, Banff, Alberta, Canada, 2004, pp. 887-894.
[184] S.K. Yu, V. Yu, K.H.-P. Tresp, M. Wu, Supervised probabilistic principal component analysis, in: Proceedings International Conference on Knowledge Discovery and Data Mining, 2006.
[185] M. Yaghoobi, T. Blumensath, M.E. Davies, Dictionary learning for sparse approximations with the majorization method, IEEE Trans. Signal Process. 57 (6) (2009) 2178-2191.
[186] B. Yang, Projection approximation subspace tracking, IEEE Trans. Signal Process. 43 (1) (1995) 95-107.
[187] S. Zafeiriou, A. Tefas, I. Buciu, I. Pitas, Exploiting discriminant information in non-negative matrix factorization with application to frontal face verification, IEEE Trans. Neural Netw. 17 (3) (2006) 683-695.
[188] Z. Zhou, X. Li, J. Wright, E.J. Candès, Y. Ma, Stable principal component pursuit, in: Proceedings, IEEE International Symposium on Information Theory, 2010, pp. 1518-1522.
[189] H. Zou, T. Hastie, R. Tibshirani, Sparse principal component analysis, J. Comput. Graph. Stat. 15 (2) (2006) 265-286.