of these spectra covering the interval 3200–7800 Å in 1000 wavelength bins. While a spectrum defined as $x(\lambda)$ may not immediately be seen as a point in a high-dimensional space, it can be represented as such. The function $x(\lambda)$ is in practice sampled at D discrete flux values, and written as a D-dimensional vector. And just as a three-dimensional vector is often visualized as a point in a three-dimensional space, this spectrum (represented by a D-dimensional vector) can be thought of as a single point in D-dimensional space. Analogously, a $D = N \times K$ image may also be expressed as a vector with D elements, and therefore a point in a D-dimensional space. So, while we use spectra as our proxy for high-dimensional space, the algorithms and techniques described in this chapter are applicable data as diverse as catalogs of multivariate data, two-dimensional images, and spectral hypercubes.

7.3. Principal Component Analysis

Figure 7.2 shows a two-dimensional distribution of points drawn from a Gaussian centered on the origin of the x- and y-axes. While the points are strongly correlated along a particular direction, it is clear that this correlation does not align with the initial choice of axes. If we wish to reduce the number of features (i.e., the number of axes) that are used to describe these data (providing a more compact representation) then it is clear that we should rotate our axes to align with this correlation (we have already encountered this rotation in eq. 3.82). Any rotation preserves the relative ordering or configuration of the data so we choose our rotation to maximize the ability to discriminate between the data points. This is accomplished if the rotation maximizes the variance along the resulting axes (i.e., defining the first axis, or principal component, to be the direction with maximal variance, the second principal component to be orthogonal to the first component that maximizes the residual variance, and so on). As indicated in figure 7.2, this is mathematically equivalent to a regression that minimizes the square of the orthogonal distances from the points to the principal axes.

This dimensionality reduction technique is known as a principal component analysis (PCA). It is also referred to in the literature as a Karhunen–Loéve [21, 25] or Hotelling transform. PCA is a linear transform, applied to multivariate data, that defines a set of uncorrelated axes (the principal components) ordered by the variance captured by each new axis. It is one of the most widely applied dimensionality reduction techniques used in astrophysics today and dates back to Pearson who, in 1901, developed a procedure for fitting lines and planes to multivariate data; see [28].

There exist a number of excellent texts on PCA that review its use across a broad range of fields and applications (e.g., [19] and references therein). We will, therefore, focus our discussion of PCA on a brief description of its mathematical formalism then concentrate on its application to astronomical data and its use as a tool for classification, data compression, regression, and signal-to-noise filtering of high-dimensional data sets.

Before progressing further with the application of PCA, it is worth noting that many of the applications of PCA to astronomical data describe the importance of the orthogonal nature of PCA (i.e., the ability to project a data set onto a set of uncorrelated axes). It is often forgotten that the observations themselves are already a

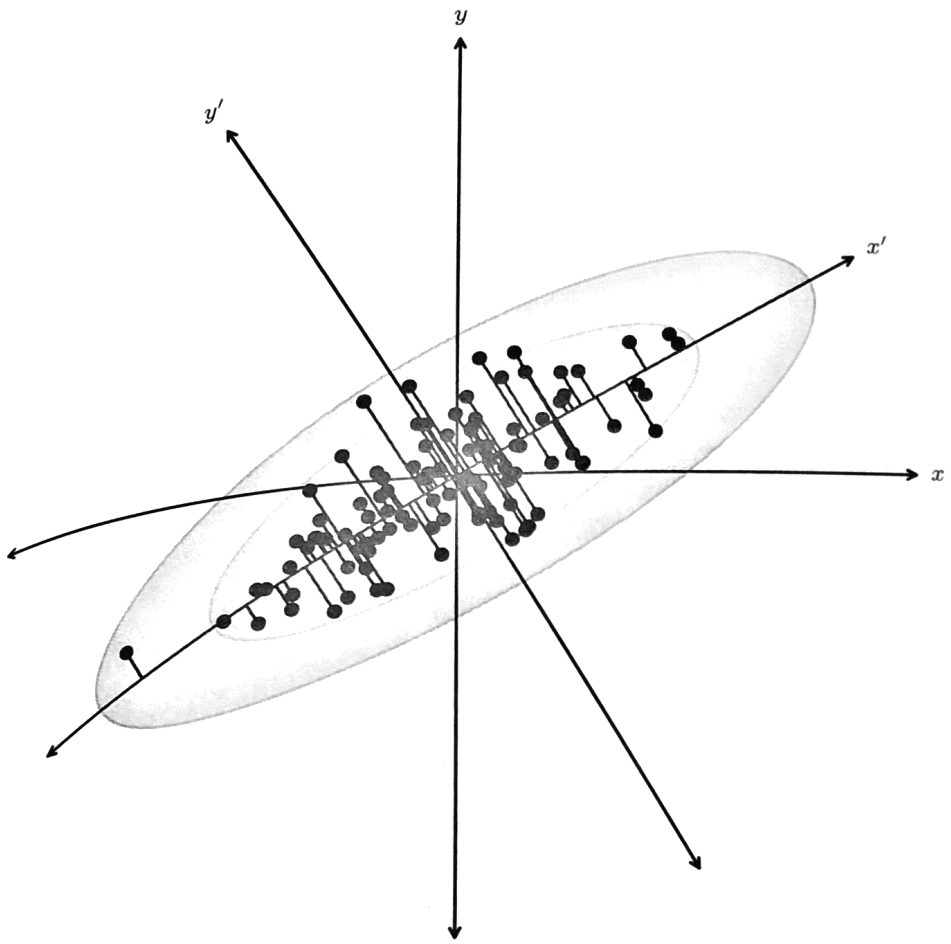

Figure 7.2. A distribution of points drawn from a bivariate Gaussian and centered on the origin of x and y. PCA defines a rotation such that the new axes (x' and y') are aligned along the directions of maximal variance (the principal components) with zero covariance. This is equivalent to minimizing the square of the perpendicular distances between the points and the principal components.

representation of an orthogonal basis (e.g., the axes {1,0,0,0, ...}, {0,1,0,0,0, ...}, etc.). As we will show, the importance of PCA is that *the new axes are aligned with the direction of maximum variance within the data* (i.e., the direction with the maximum signal).

7.3.1. The Derivation of Principal Component Analyses

Consider a set of data, $\{x_i\}$, comprising a series of N observations with each observation made up of K measured features (e.g., size, color, and luminosity, or the wavelength bins in a spectrum). We initially center the data by subtracting the mean of each feature in $\{x_i\}$ and then write this $N \times K$ matrix as X.[1] The covariance

[1]Often the opposite convention is used: that is, N points in K dimensions are stored in a $K \times N$ matrix rather than an $N \times K$ matrix. We choose the latter to align with the convention used in Scikit-learn and AstroML.

of the centered data, C_X, is given by

$$C_X = \frac{1}{N-1} X^T X, \tag{7.6}$$

where the $N - 1$ term comes from the fact that we are working with the sample covariance matrix (i.e., the covariances are derived from the data themselves).

Nonzero off-diagonal components within the covariance matrix arise because there exist correlations between the measured features (as we saw in figure 7.2; recall also the discussion of bivariate and multivariate distributions in §3.5). PCA wishes to identify a projection of $\{x_i\}$, say, R, that is aligned with the directions of maximal variance. We write this projection as $Y = XR$ and its covariance as

$$C_Y = R^T X^T X R = R^T C_X R \tag{7.7}$$

with C_X the covariance of X as defined above.

The first principal component, r_1, of R is defined as the projection with the maximal variance (subject to the constraint that $r_1^T r_1 = 1$). We can derive this principal component by using Lagrange multipliers and defining the cost function, $\phi(r_1, \lambda)$, as

$$\phi(r_1, \lambda_1) = r_1^T C_X r_1 - \lambda_1 (r_1^T r_1 - 1). \tag{7.8}$$

Setting the derivative of $\phi(r_1, \lambda)$ (with respect to r_1) to zero gives

$$C_X r_1 - \lambda_1 r_1 = 0. \tag{7.9}$$

λ_1 is, therefore, the root of the equation $\det(C_X - \lambda_1 \mathbf{I}) = 0$ and is an eigenvalue of the covariance matrix. The variance for the first principal component is maximized when

$$\lambda_1 = r_1^T C_X r_1 \tag{7.10}$$

is the largest eigenvalue of the covariance matrix. The second (and further) principal components can be derived in an analogous manner by applying the additional constraint to the cost function that the principal components are uncorrelated (e.g., $r_2^T C_X r_1 = 0$).

The columns of R are then the eigenvectors or principal components, and the diagonal values of C_Y define the amount of variance contained within each component. With

$$C_X = R C_Y R^T \tag{7.11}$$

and ordering the eigenvectors by their eigenvalue we can define the set principal components for X.

Efficient computation of principal components

One of the most direct methods for computing the PCA is through the eigenvalue decomposition of the covariance or correlation matrix, or equivalently through the

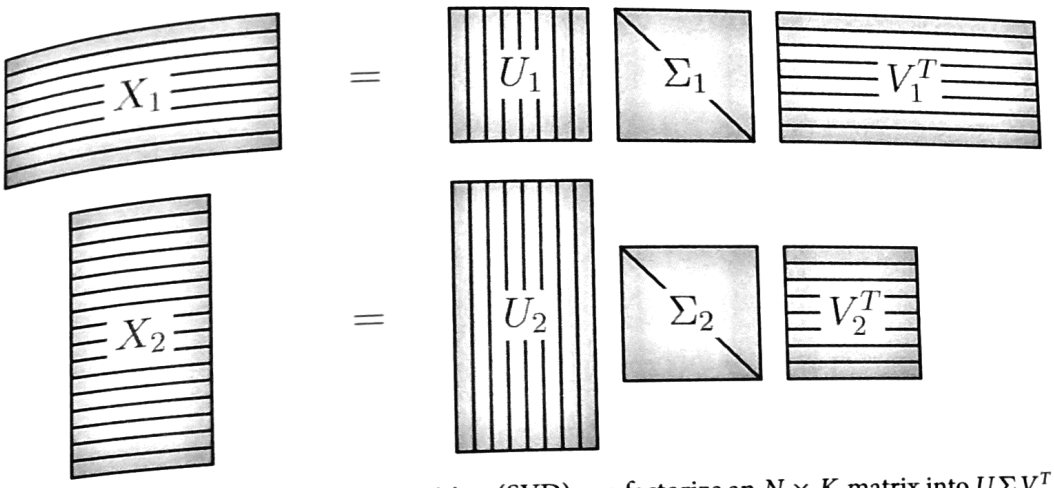

Figure 7.3. Singular value decomposition (SVD) can factorize an $N \times K$ matrix into $U\Sigma V^T$. There are different conventions for computing the SVD in the literature, and this figure illustrates the convention used in this text. The matrix of singular values Σ is always a square matrix of size $[R \times R]$ where $R = \min(N, K)$. The shape of the resulting U and V matrices depends on whether N or K is larger. The columns of the matrix U are called the left-singular vectors, and the columns of the matrix V are called the right-singular vectors. The columns are orthonormal bases, and satisfy $U^TU = V^TV = I$.

singular value decomposition (SVD) of the data matrix itself. The scaled SVD can be written

$$U\Sigma V^T = \frac{1}{\sqrt{N-1}}X, \tag{7.12}$$

where the columns of U are the *left-singular vectors*, and the columns of V are the *right-singular vectors*. There are many different conventions for the SVD in the literature; we will assume the convention that the matrix of singular values Σ is always a square, diagonal matrix, of shape $[R \times R]$ where $R = \min(N, K)$ is the rank of the matrix X (assuming all rows and columns of X are independent). U is then an $[N \times R]$ matrix, and V^T is an $[R \times K]$ matrix (see figure 7.3 for a visualization of this SVD convention). The columns of U and V form orthonormal bases, such that $U^TU = V^TV = I$.

Using the expression for the covariance matrix (eq. 7.6) along with the scaled SVD (eq. 7.12) gives

$$\begin{aligned} C_X &= \left[\frac{1}{\sqrt{N-1}}X\right]^T \left[\frac{1}{\sqrt{N-1}}X\right] \\ &= V\Sigma U^T U\Sigma V^T \\ &= V\Sigma^2 V^T. \end{aligned} \tag{7.13}$$

Comparing to eq 7.11, we see that the right singular vectors V correspond to the principal components R, and the diagonal matrix of eigenvalues C_Y is equivalent to the square of the singular values,

$$\Sigma^2 = C_Y. \tag{7.14}$$

Thus the eigenvalue decomposition of C_X, and therefore the principal components, can be computed from the SVD of X, without explicitly constructing the matrix C_X.

NumPy and SciPy contain powerful suites of linear algebra tools. For example, we can confirm the above relationship using `svd` for computing the SVD, and `eigh` for computing the symmetric (or in general Hermitian) eigenvalue decomposition:

```
>>> import numpy as np
>>> X = np.random.random((100, 3))
>>> CX = np.dot(X.T, X)
>>> U, Sdiag, VT = np.linalg.svd(X, full_matrices=
    False)
>>> CYdiag, R = np.linalg.eigh(CX)
```

The `full_matrices` keyword assures that the convention shown in figure 7.3 is used, and for both Σ and C_Y, only the diagonal elements are returned. We can compare the results, being careful of the different ordering conventions: `svd` puts the largest singular values first, while `eigh` puts the smallest eigenvalues first:

```
>>> np.allclose(CYdiag, Sdiag[::-1] ** 2)
 # [::-1] reverses the array
True
>>> np.set_printoptions(suppress=True)
 # clean output for below
>>> VT[::-1].T / R
array([[-1., -1.,  1.],
       [-1., -1.,  1.],
       [-1., -1.,  1.]])
```

The eigenvectors of C_X and the right singular vectors of X agree up to a sign, as expected. For more information, see appendix A or the documentation of `numpy.linalg` and `scipy.linalg`.

The SVD formalism can also be used to quickly see the relationship between the covariance matrix C_X, and the correlation matrix,

$$\begin{aligned} C_X &= \frac{1}{N-1} XX^T \\ &= U\Sigma V^T V \Sigma U^T \\ &= U\Sigma^2 U^T \end{aligned} \tag{7.15}$$

in analogy with above. The left singular vectors, U, turn out to be the eigenvectors of the correlation matrix, which has eigenvalues identical to those of the covariance matrix. Furthermore, the orthonormality of the matrices U and V means that if U is known, V (and therefore R) can be quickly determined using the linear algebraic

manipulation of eq. 7.12:

$$R = V = \frac{1}{\sqrt{N-1}} X^T U \Sigma^{-1}. \tag{7.16}$$

Thus we have three equivalent ways of computing the principal components R and the eigenvalues C_X: the SVD of X, the eigenvalue decomposition of C_X, or the eigenvalue decomposition of M_X. The optimal procedure will depend on the relationship between the data size N and the dimensionality K. If $N \gg K$, then using the eigenvalue decomposition of the $K \times K$ covariance matrix C_X will in general be more efficient. If $K \gg N$, then using the $N \times N$ correlation matrix M_X will be more efficient. In the intermediate case, direct computation of the SVD of X will be the most efficient route.

7.3.2. The Application of PCA

PCA can be performed easily using Scikit-learn:

```
import numpy as np
from sklearn.decomposition import PCA

X = np.random.normal(size=(100, 3))
    # 100 points in 3 dimensions
R = np.random.random((3, 10))  # projection matrix
X = np.dot(X, R)  # X is now 10-dim, with 5 intrinsic
    # dims
pca = PCA(n_components=4)  # n_components can be
      # optionally set
pca.fit(X)
comp = pca.transform(X)  # compute the subspace
       # projection of X

mean = pca.mean_  # length 10 mean of the data
components = pca.components_  # 4 x 10 matrix of
             # components
var = pca.explained_variance_  # the length 4 array
      # of eigenvalues
```

In this case, the last element of var will be zero, because the data is inherently three-dimensional. For larger problems, `RandomizedPCA` is also useful. For more information, see the Scikit-learn documentation.

To form the data matrix X, the data vectors are centered by subtracting the mean of each dimension. Before this takes place, however, the data are often preprocessed to ensure that the PCA is maximally informative. In the case of heterogeneous data (e.g., galaxy shape and flux), the columns are often preprocessed by dividing by

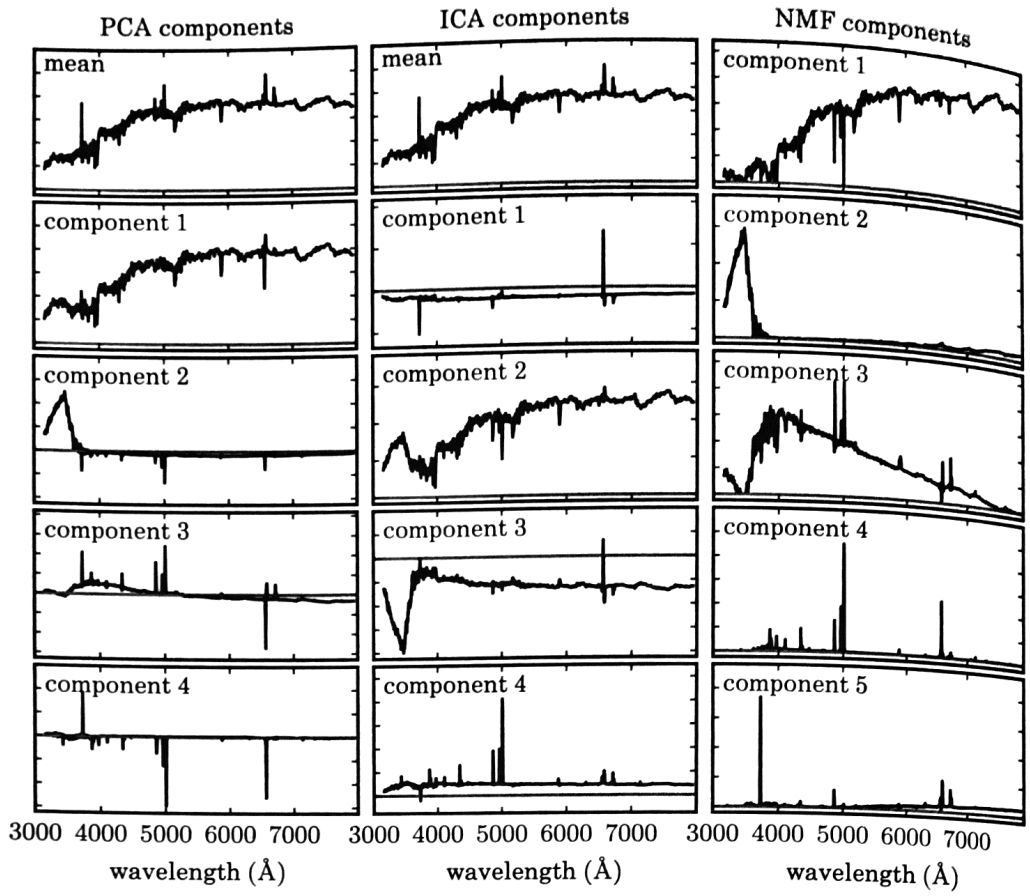

Figure 7.4. A comparison of the decomposition of SDSS spectra using PCA (left panel—see §7.3.1), ICA (middle panel—see §7.6) and NMF (right panel—see §7.4). The rank of the component increases from top to bottom. For the ICA and PCA the first component is the mean spectrum (NMF does not require mean subtraction). All of these techniques isolate a common set of spectral features (identifying features associated with the continuum and line emission). The ordering of the spectral components is technique dependent.

their variance. This so-called whitening of the data ensures that the variance of each feature is comparable, and can lead to a more physically meaningful set of principal components. In the case of spectra or images, a common preprocessing step is to normalize each row, such that the integrated flux of each object is one. This helps to remove uninteresting correlations based on the overall brightness of the spectrum or image.

For the case of the galaxy spectra in figure 7.1, each spectrum has been normalized to a constant total flux, before being centered such that the spectrum has zero mean (this subtracted mean spectrum is shown in the upper-left panel of figure 7.4). The principal directions found in the high-dimensional data set are often referred to as the "eigenspectra," and just as a vector can be represented by the sum of its components, a spectrum can be represented by the sum of its eigenspectra. The left panel of figure 7.4 shows, from top to bottom, the mean spectrum and the first four eigenspectra. The eigenspectra are ordered by their associated eigenvalues shown in figure 7.5. Figure 7.5 is often referred to as a scree plot (related to the shape of rock debris after it has fallen down a slope; see [6]) with the eigenvalues reflecting

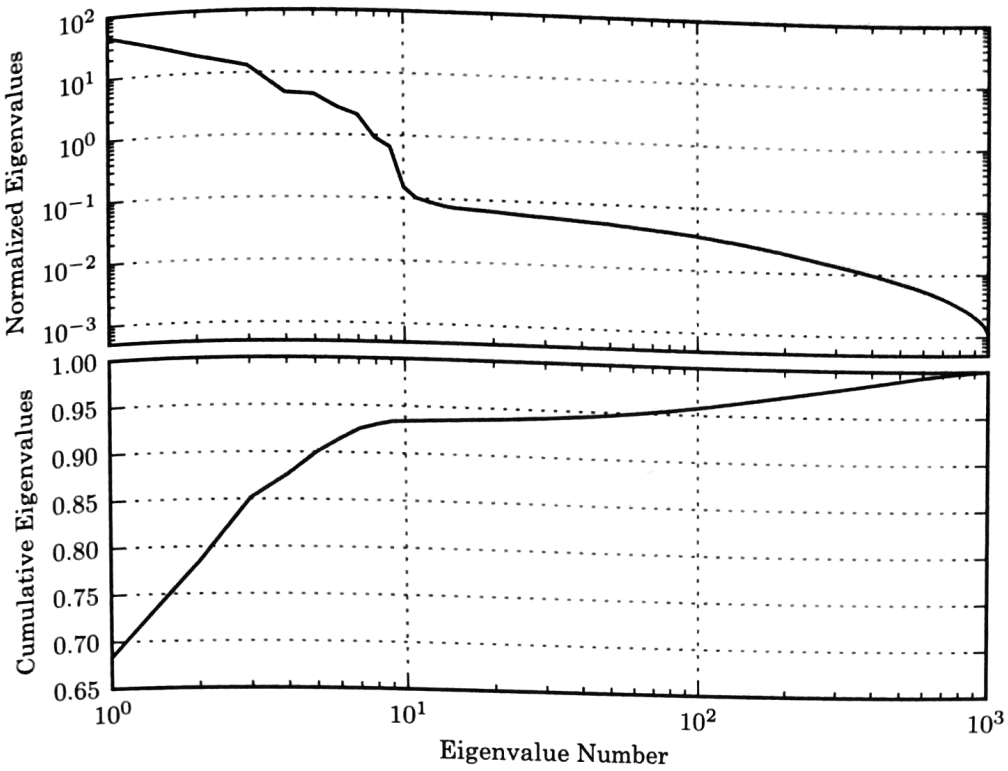

Figure 7.5. The eigenvalues for the PCA decomposition of the SDSS spectra described in §7.3.2. The top panel shows the decrease in eigenvalue as a function of the number of eigenvectors, with a break in the distribution at ten eigenvectors. The lower panel shows the cumulative sum of eigenvalues normalized to unity. 94% of the variance in the SDSS spectra can be captured using the first ten eigenvectors.

the amount of variance contained within each of the associated eigenspectra (with the constraint that the sum of the eigenvalues equals the total variance of the system).

The cumulative variance associated with the eigenvectors measures the amount of variance of the *entire data set* which is encoded in the eigenvectors. From figure 7.5, we see that ten eigenvectors are responsible for 94% of the variance in the sample: this means that by projecting each spectrum onto these first ten eigenspectra, an average of 94% of the "information" in each spectrum is retained, where here we use the term "information" loosely as a proxy for variance. This amounts to a compression of the data by a factor of 100 (using ten of the 1000 eigencomponents) with a very small loss of information. This is the sense in which PCA allows for dimensionality reduction.

This concept of data compression is supported by the shape of the eigenvectors. Eigenvectors with large eigenvalues are predominantly low-order components (in the context of astronomical data they primarily reflect the continuum shape of the galaxies). Higher-order components (with smaller eigenvalues) are predominantly made up of sharp features such as emission lines. The combination of continuum and line emission within these eigenvectors can describe any of the input spectra. The remaining eigenvectors reflect the noise within the ensemble of spectra in the sample.

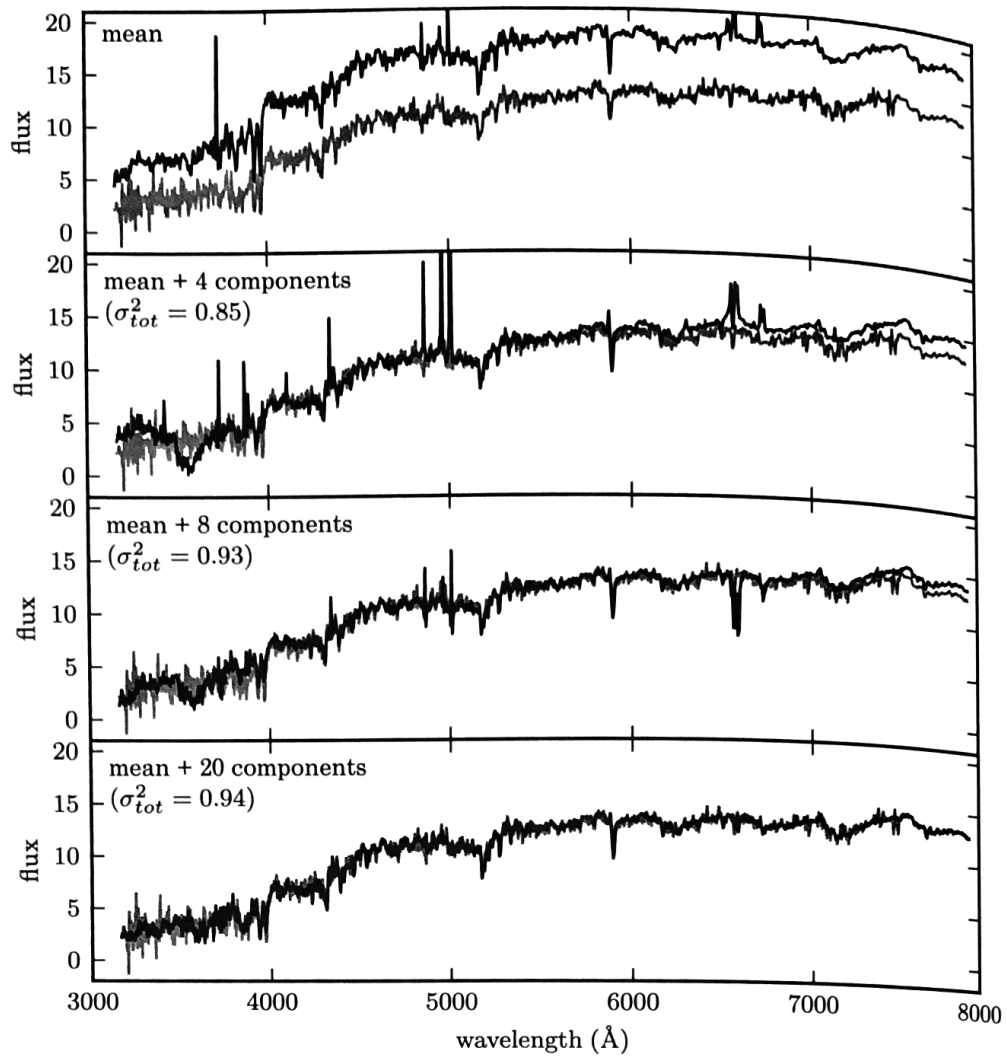

Figure 7.6. The reconstruction of a particular spectrum from its eigenvectors. The input spectrum is shown in gray, and the partial reconstruction for progressively more terms is shown in black. The top panel shows only the mean of the set of spectra. By the time 20 PCA components are added, the reconstruction is very close to the input, as indicated by the expected total variance of 94%.

The reconstruction of an example spectrum, $\boldsymbol{x}(k)$, from the eigenbasis, $\boldsymbol{e}_i(k)$ is shown in figure 7.6. Each spectrum $\boldsymbol{x}_i(k)$ can be described by

$$\boldsymbol{x}_i(k) = \boldsymbol{\mu}(k) + \sum_{j}^{R} \theta_{ij} \boldsymbol{e}_j(k), \tag{7.17}$$

where i represents the number of the input spectrum, j represents the number of the eigenspectrum, and, for the case of a spectrum, k represents the wavelength. Here, $\boldsymbol{\mu}(k)$ is the mean spectrum and θ_{ij} are the linear expansion coefficients derived from

$$\theta_{ij} = \sum_{k} \boldsymbol{e}_j(k)(\boldsymbol{x}_i(k) - \boldsymbol{\mu}(k)). \tag{7.18}$$

R is the total number of eigenvectors (given by the rank of X, $\min(N,K)$). If the summation is over all eigenvectors, the input spectrum is fully described with no loss of information. Truncating this expansion (i.e., $r < R$),

$$\boldsymbol{x}_i(k) = \sum_{i}^{r<R} \theta_i \boldsymbol{e}_i(k), \tag{7.19}$$

will exclude those eigencomponents with smaller eigenvalues. These components will, predominantly, reflect the noise within the data set. This is reflected in figure 7.6: truncating the reconstruction at 20 components captures the overall shape and important features of the spectrum: the differences between the reconstruction and the input spectrum are mostly high-frequency spectral noise.

A number of aspects of PCA are worth noting. Comparisons between the eigenvectors derived from PCA and known spectral types of galaxies have shown that these statistically orthogonal components correlate strongly with specific physical properties (i.e., they relate to the star formation and the composition of the stellar types within a galaxy spectrum, e.g., [33, 34]). Second, given the cumulative nature of the sum of variances used in PCA, astrophysically interesting components within the spectra (e.g., sharp spectral lines or transient features for certain galaxy populations) may not be reflected in the largest PCA components. Because of this, care must be taken when truncating at a small number of components. Additionally, the assumption that a sum of linear components can efficiently reconstruct the features within the data does not always hold. An example of this is the variation in broad emission lines (such as those from quasars). The variation in line width is an inherently nonlinear process and can require a large number of components to fully characterize: for broad line quasars over 30 components are required to reproduce the underlying spectra compared to the 10 required for quiescent and star-forming galaxies. In these cases, dimensionality reduction techniques based on the local structure need to be considered (see §7.5). Finally, up to this point we have ignored errors and missing data when considering the application of PCA. We address this in §7.3.3.

Choosing the Level of Truncation in an Expansion

One of the critical issues when reconstructing a data set from a linear combination of eigenvectors is choosing the number of components, r, to keep. Too many components will introduce noise into the reconstruction. Too few may not capture the complete physical correlations within the data. While many attempts have been made to place the choice of r on a sound statistical footing, the techniques that are used today are typically either based on empirical relations derived from simplified experiments or derived from a series of somewhat ad hoc assumptions (see [19] for a detailed discussion).

The most common criterion for defining r is based on the total variance captured in the first r eigenvectors. If we specify a bound, α, on the fraction of the variance we wish to capture then we can define r from the summation of the

eigenvalues, σ_i, that are the diagonals of the matrix Σ:

$$\frac{\sum_i^{i=r} \sigma_i}{\sum_i^{i=R} \sigma_i} < \alpha. \tag{7.20}$$

Typical values for α range from 0.70 to 0.95, though the choice of threshold is sensitive to the shape of the scree plot (figure 7.5); for a shallow slope in the scree plot, whether r or $r + 1$ crosses the threshold is somewhat arbitrary.

The shape of the scree plot can be used to define the level of truncation. Cattell [6], using factor analysis, proposed that a sharp change in the gradient of the eigenvalues (i.e., a knee in the scree plot) could be used to define the cutoff value, r. The knee is defined by [6] as $\Sigma_r^2 - \Sigma_{r+1}^2$. If no clearly defined break in the scree plot exists, the definition of this cutoff becomes problematic. A modification of the technique in [6] is the LEV diagram, which plots the logarithm of the eigenvalue against the number of eigenvectors. The rationale for this modification is that if noise decays geometrically, variation in the eigenvalues should drop as a linear function.

For the correlation matrix PCA, Kaiser's rule [20] or the Guttman–Kaiser criterion can be applied. This states that, if all of the elements of x are independent then all principal components would have unit variance. In this case, r can be set to the limit where the eigenvalues in the scree plot fall below unity. In the context of the covariance matrix this can be reformulated as the setting of r to the number of components at which the eigenvalue falls to the average of all eigenvalues. Jolliffe [19] proposed a modification of this truncation to 70% of the average eigenvalues to allow for sample variance (which increases the number of components returned). Experiments with Kaiser's rule show that it tends to overpredict the number of components that remain after truncation.

Each of the criteria described above are sensitive to the shape of the scree plot and the choice of truncation rule remains application specific.

7.3.3. PCA with Missing Data

Until now we have assumed that the data we are working with are complete, without gaps or censored elements. In real-world applications, the presence of detector glitches, variable noise, or masking effects (e.g., sky lines in astronomical spectra) can make these assumptions invalid. Truncation of the expansion provides a signal-to-noise filtering of the data. The PCA bases should also be able to correct for missing elements within the data: because the PCA components encode the correlation of each flux with the other measured fluxes, these components should provide a natural way to determine these missing values.

One complication we must address is that eigenspectra are only defined to be orthogonal over the data range on which they are constructed. If a data vector does not fully cover that space then projecting the data onto the eigenbases will result in a biased set of expansion coefficients. Everson and Sirovich [12] have, however, shown that, when we know how the input data are masked or censored, we can correct for the nonorthogonality of the eigenbases. Following Connolly and Szalay [9], we consider an observed spectrum, $\boldsymbol{x}^o$, as the combination of the true spectrum (i.e., without gaps), $\boldsymbol{x}$, and a wavelength-dependent weight, $\boldsymbol{w}$. This weight is zero where

data are missing and $1/\sigma^2$ for the remaining spectral range (with σ^2 the variance of the spectral elements). Minimizing the quadratic deviation between the original spectrum, $\boldsymbol{x}^o$, and its truncated reconstruction, $\sum_i \theta_i \boldsymbol{e}_i$ and solving for θ_i gives

$$\sum_k \theta_i \boldsymbol{w}(k)\boldsymbol{e}_i(k)\boldsymbol{e}_j(k) = \sum_k \boldsymbol{w}(k)\boldsymbol{x}^o(k)\boldsymbol{e}_j(k), \tag{7.21}$$

where $\sum_k$ represents the sum over the length of the vector $\boldsymbol{x}(k)$ (i.e., over wavelength for the case of the spectra). If we define $M_{ij} = \sum_k \boldsymbol{w}(k)\boldsymbol{e}_i(k)\boldsymbol{e}_j(k)$ and $F_i = \sum_k \boldsymbol{w}(k)\boldsymbol{x}^o(k)\boldsymbol{e}_i(k)$ then this simplifies to

$$\theta_i = \sum_j M_{ij}^{-1} F_j, \tag{7.22}$$

where F_j represent the coefficients derived from the gappy data and M_{ij}^{-1} expresses how correlated the eigenvectors are over the missing regions.

Figure 7.7 shows the reconstruction of missing data using the PCA components. The gray regions represent intervals in wavelength space where the spectra are censored (the underlying data values within these censored regions are shown by the black line). The gray lines represent the reconstruction of these spectral regions using the eigenbases. An estimate of the uncertainty on the reconstruction coefficients is given by

$$\mathrm{Cov}(\theta_i, \theta_j) = M_{ij}^{-1}. \tag{7.23}$$

The accuracy of this reconstruction will depend on the distribution of the gaps within the data vector (though this is reflected in $\mathrm{Cov}(\theta_i, \theta_j)$). For an uncorrelated set of gaps, reconstruction with the number of sampled points, $N_{\mathrm{samples}} > r$, is possible. This observation is at the heart of the fields of lossy compression and compressed sensing.

7.3.4. Scaling to Large Data Sets

There are a number of limitations of PCA that can make it impractical for application to very large data sets. Principal to this are the computational and memory requirements of the SVD, which scale as $\mathcal{O}(D^3)$ and $\mathcal{O}(2 \times D \times D)$, respectively. In §7.3.1 we derived the PCA by applying an SVD to the covariance and correlation matrices of the data X. Thus, the computational requirements of the SVD are set by the rank of the data matrix, X, with the covariance matrix the preferred route if $K < N$ and the correlation matrix if $K > N$. Given the symmetric nature of both the covariance and correlation matrix, eigenvalue decompositions (EVD) are often more efficient than SVD approaches.

Even given these optimizations, with data sets exceeding the size of the memory available per core, applications of PCA can be very computationally challenging. This is particularly the case for real-world applications when the correction techniques for missing data are iterative in nature. One approach to address these limitations is to make use of online algorithms for an iterative calculation of the mean. As shown in

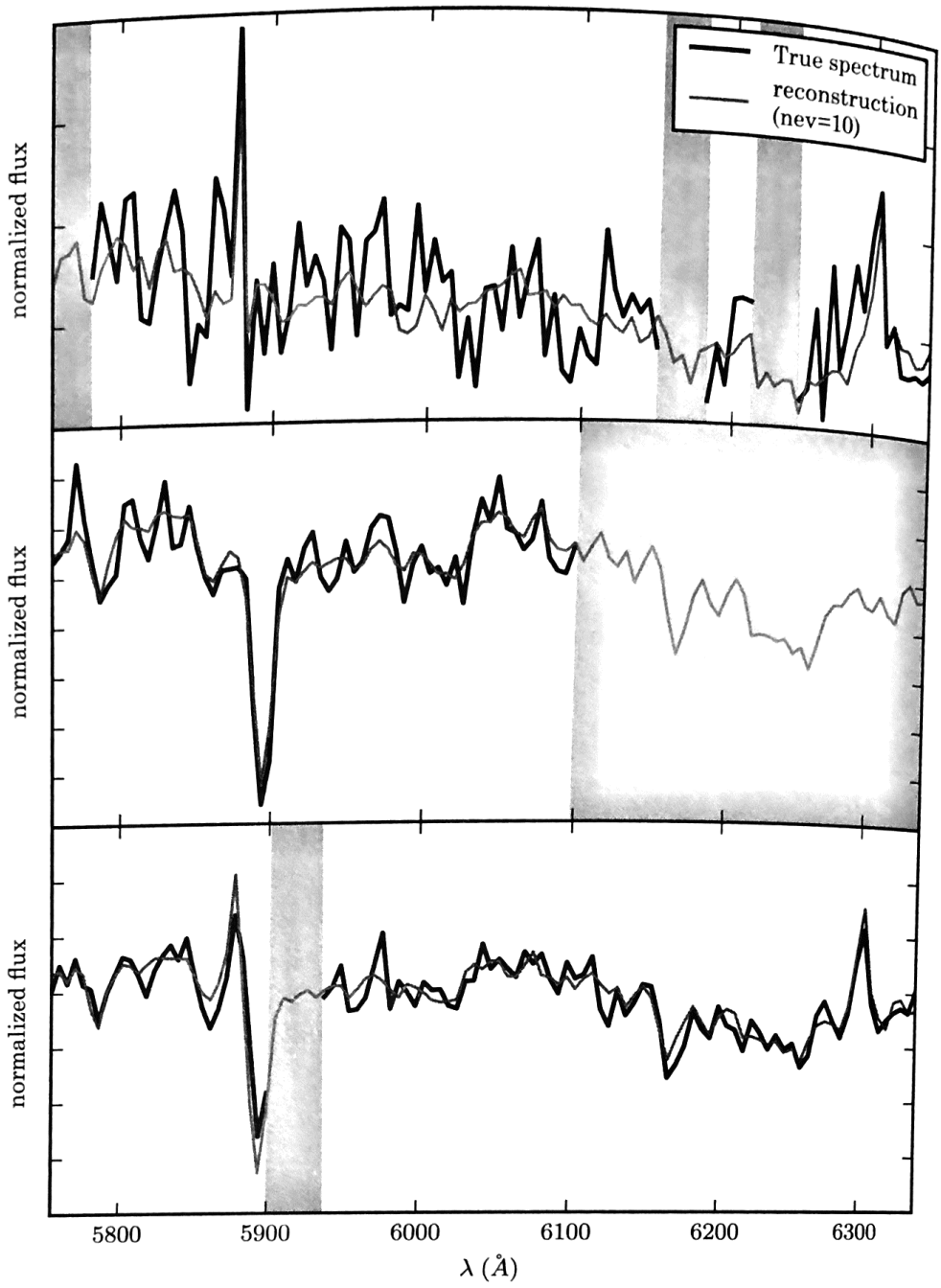

Figure 7.7. The principal component vectors defined for the SDSS spectra can be used to interpolate across or reconstruct missing data. Examples of three masked spectral regions are shown comparing the reconstruction of the input spectrum (black line) using the mean and the first ten eigenspectra (gray line) The gray bands represent the masked region of the spectrum.

[5], the sample covariance matrix can be defined as

$$C = \gamma C_{\text{prev}} + (1 - \gamma) x^T x \tag{7.24}$$

$$\sim \gamma Y_p D_p Y_p^T + (1 - \gamma) x^T x, \tag{7.25}$$

where C_{prev} is the covariance matrix derived from a previous iteration, x is the new observation, Y_p are the first p eigenvectors of the previous covariance matrix, D_p

are the associated eigenvalues, and γ is a weight parameter that can scale with step size.

For each new observation the covariance matrix is updated, and the PCA components based on this matrix enable the identification and filtering of outliers within a data set. Assuming that the number of required eigenvectors, p, is small, the cost of periodically reevaluating the SVD of the updated covariance matrix is a cheap operation.

7.4. Nonnegative Matrix Factorization

One of the challenges in interpreting PCA bases comes from the fact that the eigenvectors are defined relative to the mean data vector. This results in principal components that can be positive or negative. In contrast, for many physical systems we have a priori knowledge that a data vector can be represented by a linear sum of positive components. For example, in the case of the SDSS spectra, a galaxy spectrum can be assumed to be a linear sum of stellar components (consistent with the general form of the eigenspectra seen in figure 7.4).

Nonnegative matrix factorization (NMF) applies an additional constraint on the components that comprise the data matrix X; see [23]. It assumes that any data matrix can be factored into two matrices, W and Y, such that

$$X = WY, \tag{7.26}$$

where both W and Y are nonnegative (i.e., all elements in these matrices are nonnegative). WY is, therefore, an approximation of X. By minimizing the reconstruction error $||(X - WY)^2||$, it can be shown that nonnegative bases can be derived using a simple update rule,

$$W_{ki} = W_{ki} \frac{[XY^T]_{ki}}{[WYY^T]_{ki}}, \tag{7.27}$$

$$Y_{in} = Y_{in} \frac{[W^T X]_{in}}{[W^T WY]_{in}}, \tag{7.28}$$

where n, k, and i denote the wavelength, spectrum and template indices, respectively [23]. This iterative process does not guarantee nonlocal minima (as with many iterative machine learning approaches such as K-means and EM). With random initialization and cross-validation the solutions for the NMF bases are, however, often appropriate.

The central panel of figure 7.7 shows the results of NMF applied to the spectra used in the PCA analysis of §7.3.2. The components derived by NMF are broadly consistent with those from PCA but with a different ordering of the basis functions. Given that both applications assume a linear transform between X and Y this might not be that surprising. For the case of NMF, the assumption is that each spectrum can be represented by a smaller number of nonnegative components than the underlying dimensionality of the data. The number of components are, therefore, defined prior