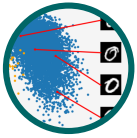


## Dimensionality Reduction with Principal Component Analysis



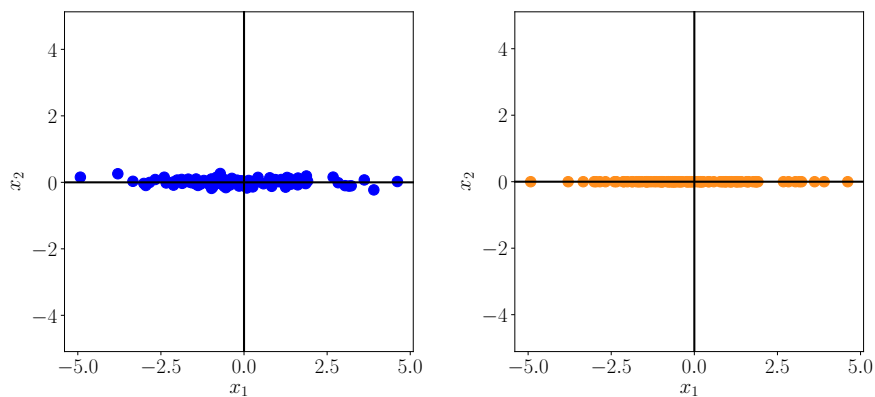
A  $640 \times 480$  pixel color image is a data point in a million-dimensional space, where every pixel responds to three dimensions, one for each color channel (red, green, blue).

Working directly with high-dimensional data, such as images, comes with some difficulties: it is hard to analyze, interpretation is difficult, visualization is nearly impossible, and (from a practical point of view) storage of the data vectors can be expensive. However, high-dimensional data often has properties that we can exploit. For example, high-dimensional data is often overcomplete, i.e., many dimensions are redundant and can be explained by a combination of other dimensions. Furthermore, dimensions in high-dimensional data are often correlated so that the data possesses an intrinsic lower-dimensional structure. Dimensionality reduction exploits structure and correlation and allows us to work with a more compact representation of the data, ideally without losing information. We can think of dimensionality reduction as a compression technique, similar to jpeg or mp3, which are compression algorithms for images and music.

principal component analysis  
dimensionality reduction

In this chapter, we will discuss *principal component analysis* (PCA), an algorithm for linear *dimensionality reduction*. PCA, proposed by Pearson (1901b) and Hotelling (1933), has been around for more than 100 years and is still one of the most commonly used techniques for data compression and data visualization. It is also used for the identification of simple patterns, latent factors and structures of high-dimensional data. In the sig-

**Figure 10.1**  
Illustration: Dimensionality reduction. (a) The original dataset does not vary much along the  $x_2$  direction. (b) The data from (a) can be represented using the  $x_1$ -coordinate alone with nearly no loss.



(a) Dataset with  $x_1$  and  $x_2$  coordinates. (b) Compressed dataset where only the  $x_1$  coordinate is relevant.

5848 nal processing community, PCA is also known as the *Karhunen-Loève trans-*  
 5849 *form*. In this chapter, we derive PCA from first principles, drawing on our  
 5850 understanding of basis and basis change (Sections 2.6.1 and 2.7.2), pro-  
 5851 jections (Section 3.7), eigenvalues (Section 4.2), Gaussian distributions  
 5852 (Section 6.5) and constrained optimization (Section 7.2).

Karhunen-Loève  
transform

5853 Dimensionality reduction generally exploits a property of high-dimen-  
 5854 sional data (e.g., images) that it often lies on a low-dimensional subspace.  
 5855 Figure 10.1 gives an illustrative example in two dimensions. Although  
 5856 the data in Figure 10.1(a) does not quite lie on a line, the data does not  
 5857 vary much in the  $x_2$ -direction, so that we can express it as if it was on  
 5858 a line – with nearly no loss, see Figure 10.1(b). To describe the data in  
 5859 Figure 10.1(b), only the  $x_1$ -coordinate is required, and the data lies in a  
 5860 one-dimensional subspace of  $\mathbb{R}^2$ .

5861 **10.1 Problem Setting**

5862 In PCA, we are interested in finding projections  $\tilde{x}_n$  of data points  $x_n$  that  
 5863 are as similar to the original data points as possible, but which have a sig-  
 5864 nificantly lower intrinsic dimensionality. Figure 10.1 gives an illustration  
 5865 what this could look like.

More concretely, we consider an i.i.d. dataset  $\mathcal{X} = \{x_1, \dots, x_N\}$ ,  $x_n \in \mathbb{R}^D$ , with mean  $\mathbf{0}$  that possesses the *data covariance matrix* (6.42)

data covariance  
matrix

$$S = \frac{1}{N} \sum_{n=1}^N x_n x_n^\top. \tag{10.1}$$

Furthermore, we assume there exists a low-dimensional compressed representation (code)

$$z_n = B^\top x_n \in \mathbb{R}^M \tag{10.2}$$

of  $x_n$ , where we define the projection matrix

$$B := [b_1, \dots, b_M] \in \mathbb{R}^{D \times M}. \tag{10.3}$$

5866 We assume that the columns of  $B$  are orthonormal (Definition 3.7) so that  
 5867  $b_i^\top b_j = 0$  if and only if  $i \neq j$  and  $b_i^\top b_i = 1$ . We seek an  $M$ -dimensional  
 5868 subspace  $U \subseteq \mathbb{R}^D$ ,  $\dim(U) = M < D$  onto which we project the data. We  
 5869 denote the projected data by  $\tilde{x}_n \in U$ , and their coordinates (with respect  
 5870 to the basis vectors  $b_1, \dots, b_M$  of  $U$ ) by  $z_n$ . Our aim is to find projections  
 5871  $\tilde{x}_n \in \mathbb{R}^D$  (or equivalently the codes  $z_n$  and the basis vectors  $b_1, \dots, b_M$ )  
 5872 so that they are as similar to the original data  $x_n$  and minimize the loss  
 5873 due to compression.

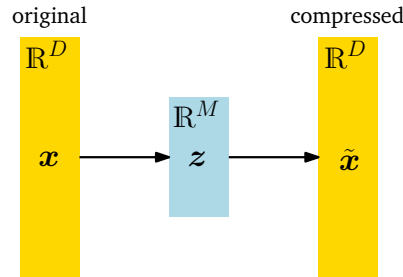
The columns  $b_1, \dots, b_M$  of  $B$  form a basis of the  $M$ -dimensional subspace in which the projected data  $\tilde{x} = BB^\top x \in \mathbb{R}^D$  live.

**Example 10.1 (Coordinate Representation/Code)**

Consider  $\mathbb{R}^2$  with the canonical basis  $e_1 = [1, 0]^\top$ ,  $e_2 = [0, 1]^\top$ . From

**Figure 10.2**

Graphical illustration of PCA. In PCA, we find a compressed version  $\tilde{x}$  of original data  $x$  that has an intrinsic lower-dimensional representation  $z$ .



Chapter 2 we know that  $x \in \mathbb{R}^2$  can be represented as a linear combination of these basis vectors, e.g.,

$$\begin{bmatrix} 5 \\ 3 \end{bmatrix} = 5e_1 + 3e_2. \quad (10.4)$$

However, when we consider vectors of the form

$$\tilde{x} = \begin{bmatrix} 0 \\ z \end{bmatrix} \in \mathbb{R}^2, \quad z \in \mathbb{R}, \quad (10.5)$$

they can always be written as  $0e_1 + ze_2$ . To represent these vectors it is sufficient to remember/store the *coordinate/code*  $z$  of  $\tilde{x}$  with respect to the  $e_2$  vector.

More precisely, the set of  $\tilde{x}$  vectors (with the standard vector addition and scalar multiplication) forms a vector subspace  $U$  (see Section 2.4) with  $\dim(U) = 1$  because  $U = \text{span}[e_2]$ .

The dimension of a vector space corresponds to the number of its basis vectors (see Section 2.6.1).

5874 In Section 10.2, we will find low-dimensional representations that re-  
 5875 tain as much information as possible and minimize the compression loss.  
 5876 An alternative derivation of PCA is given in Section 10.3, we will be look-  
 5877 ing at minimizing the squared reconstruction error  $\|x_n - \tilde{x}_n\|^2$  between  
 5878 the original data  $x_n$  and its projection  $\tilde{x}_n$ .

5879 Figure 10.2 illustrates the setting we consider in PCA, where  $z$  repre-  
 5880 sents the intrinsic lower dimension of the compressed data  $\tilde{x}$  and plays  
 5881 the role of a bottleneck, which controls how much information can flow  
 5882 between  $x$  and  $\tilde{x}$ . In PCA, we consider a linear relationship between the  
 5883 original data  $x$  and its low-dimensional code  $z$  so that  $z = B^\top x$  and  
 5884  $\tilde{x} = Bz$  for a suitable matrix  $B$ . Based the motivation of thinking of  
 5885 PCA as a data compression technique, we can interpret the arrows in Fig-  
 5886 ure 10.2 as a pair of operations representing encoders and decoders. The  
 5887 linear mapping represented by  $B$  can be thought of a decoder, which maps  
 5888 the low-dimensional code  $z \in \mathbb{R}^M$  back into the original data space  $\mathbb{R}^D$ .  
 5889 Similarly,  $B^\top$  can be thought of an encoder, which encodes the original  
 5890 data  $x$  as a low-dimensional (compressed) code  $z$ .

5891 Throughout this chapter, we will use the MNIST digits dataset as a re-



**Figure 10.3**  
Examples of  
handwritten digits  
from the MNIST  
dataset. [http:  
//yann.lecun.  
com/exdb/mnist/](http://yann.lecun.com/exdb/mnist/)

5892 occurring example, which contains 60,000 examples of handwritten digits  
5893 0–9. Each digit is a grayscale image of size  $28 \times 28$ , i.e., it contains 784  
5894 pixels so that we can interpret every image in this dataset as a vector  
5895  $\mathbf{x} \in \mathbb{R}^{784}$ . Examples of these digits are shown in Figure 10.3.

## 5896 10.2 Maximum Variance Perspective

5897 Figure 10.1 gave an example of how a two-dimensional dataset can be  
5898 represented using a single coordinate. In Figure 10.1(b), we chose to ig-  
5899 nore the  $x_2$ -coordinate of the data because it did not add too much in-  
5900 formation so that the compressed data is similar to the original data in  
5901 Figure 10.1(a). We could have chosen to ignore the  $x_1$ -coordinate, but  
5902 then the compressed data had been very dissimilar from the original data,  
5903 and much information in the data would have been lost.

5904 If we interpret information content in the data as how “space filling”  
5905 the data set is, then we can describe the information contained in the data  
5906 by looking at the spread of the data. From Section 6.4.1 we know that the  
5907 variance is an indicator of the spread of the data, and we can derive PCA as  
5908 a dimensionality reduction algorithm that maximizes the variance in the  
5909 low-dimensional representation of the data to retain as much information  
5910 as possible. Figure 10.4 illustrates this.

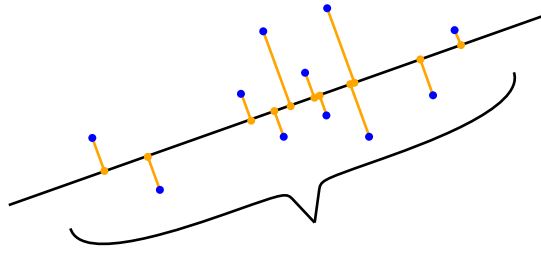
5911 Considering the setting discussed in Section 10.1, our aim is to find  
5912 a matrix  $\mathbf{B}$  (see (10.3)) that retains as much information as possible  
5913 when compressing data by projecting it onto the subspace spanned by  
5914 the columns  $\mathbf{b}_1, \dots, \mathbf{b}_M$  of  $\mathbf{B}$ . Retaining most information after data com-  
5915 pression is equivalent to capturing the largest amount of variance in the  
5916 low-dimensional code (Hotelling, 1933).

*Remark.* (Centered Data) For the data covariance matrix in (10.1) we  
assumed centered data. We can make this assumption without loss of gen-  
erality: Let us assume that  $\boldsymbol{\mu}$  is the mean of the data. Using the properties  
of the variance, which we discussed in Section 6.4.4 we obtain

$$\mathbb{V}_z[\mathbf{z}] = \mathbb{V}_x[\mathbf{B}^\top(\mathbf{x} - \boldsymbol{\mu})] = \mathbb{V}_x[\mathbf{B}^\top\mathbf{x} - \mathbf{B}^\top\boldsymbol{\mu}] = \mathbb{V}_x[\mathbf{B}^\top\mathbf{x}], \quad (10.6)$$

5917 i.e., the variance of the low-dimensional code does not depend on the  
5918 mean of the data. Therefore, we assume without loss of generality that the  
5919 data has mean  $\mathbf{0}$  for the remainder of this section. With this assumption  
5920 the mean of the low-dimensional code is also  $\mathbf{0}$  since  $\mathbb{E}_z[\mathbf{z}] = \mathbb{E}_x[\mathbf{B}^\top\mathbf{x}] =$   
5921  $\mathbf{B}^\top\mathbb{E}_x[\mathbf{x}] = \mathbf{0}$ .  $\diamond$

**Figure 10.4** PCA finds a lower-dimensional subspace (line) that maintains as much variance (spread of the data) as possible when the data (blue) is projected onto this subspace (orange).



5922

### 10.2.1 Direction with Maximal Variance

The vector  $\mathbf{b}_1$  will be the first column of the matrix  $\mathbf{B}$  and therefore the first of  $M$  orthonormal basis vectors that span the lower-dimensional subspace.

We maximize the variance of the low-dimensional code using a sequential approach. We start by seeking a single vector  $\mathbf{b}_1 \in \mathbb{R}^D$  that maximizes the variance of the projected data, i.e., we aim to maximize the variance of the first coordinate  $z_1$  of  $\mathbf{z} \in \mathbb{R}^M$  so that

$$V_1 := \mathbb{V}[z_1] = \frac{1}{N} \sum_{n=1}^N z_{1n}^2 \quad (10.7)$$

is maximized, where we exploited the i.i.d. assumption of the data and defined  $z_{1n}$  as the first coordinate of the low-dimensional representation  $\mathbf{z}_n \in \mathbb{R}^M$  of  $\mathbf{x}_n \in \mathbb{R}^D$ . Note that first component of  $\mathbf{z}_n$  is given by

$$z_{1n} = \mathbf{b}_1^\top \mathbf{x}_n, \quad (10.8)$$

i.e., it is the coordinate of the orthogonal projection of  $\mathbf{x}_n$  onto the one-dimensional subspace spanned by  $\mathbf{b}_1$  (Section 3.7). We substitute (10.8) into (10.7), which yields

$$V_1 = \frac{1}{N} \sum_{n=1}^N (\mathbf{b}_1^\top \mathbf{x}_n)^2 = \frac{1}{N} \sum_{n=1}^N \mathbf{b}_1^\top \mathbf{x}_n \mathbf{x}_n^\top \mathbf{b}_1 \quad (10.9a)$$

$$= \mathbf{b}_1^\top \left( \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top \right) \mathbf{b}_1 = \mathbf{b}_1^\top \mathbf{S} \mathbf{b}_1, \quad (10.9b)$$

where  $\mathbf{S}$  is the data covariance matrix defined in (10.1). In (10.9a) we have used the fact that the dot product of two vectors is symmetric with respect to its arguments, that is  $\mathbf{b}_1^\top \mathbf{x}_n = \mathbf{x}_n^\top \mathbf{b}_1$ .

Notice that arbitrarily increasing the magnitude of the vector  $\mathbf{b}_1$  increases  $V_1$ , that is, a vector  $\mathbf{b}_1$  that is two times longer can result in  $V_1$  that is potentially four times larger. Therefore, we restrict all solutions to  $\|\mathbf{b}_1\|^2 = 1$ , which results in a constrained optimization problem in which we seek the direction along which the data varies most.

With the restriction of the solution space to unit vectors the vector  $\mathbf{b}_1$  that points in the direction of maximum variance can be found by the

$$\|\mathbf{b}_1\|^2 = 1 \\ \iff \|\mathbf{b}_1\| = 1.$$

5930

constrained optimization problem

$$\begin{aligned} & \max_{\mathbf{b}_1} \mathbf{b}_1^\top \mathbf{S} \mathbf{b}_1 \\ & \text{subject to } \|\mathbf{b}_1\|^2 = 1. \end{aligned} \tag{10.10}$$

Following Section 7.2, we obtain the Lagrangian

$$\mathcal{L}(\mathbf{b}_1, \lambda) = \mathbf{b}_1^\top \mathbf{S} \mathbf{b}_1 + \lambda_1(1 - \mathbf{b}_1^\top \mathbf{b}_1) \tag{10.11}$$

to solve this constrained optimization problem. The partial derivatives of  $\mathcal{L}$  with respect to  $\mathbf{b}_1$  and  $\lambda_1$  are

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}_1} = 2\mathbf{b}_1^\top \mathbf{S} - 2\lambda_1 \mathbf{b}_1^\top, \quad \frac{\partial \mathcal{L}}{\partial \lambda_1} = 1 - \mathbf{b}_1^\top \mathbf{b}_1, \tag{10.12}$$

respectively. Setting these partial derivatives to  $\mathbf{0}$  gives us the relations

$$\mathbf{S} \mathbf{b}_1 = \lambda_1 \mathbf{b}_1, \tag{10.13}$$

$$\mathbf{b}_1^\top \mathbf{b}_1 = 1. \tag{10.14}$$

By comparing with the definition of an eigenvalue decomposition (Section 4.4), we see that  $\mathbf{b}_1$  is an eigenvector of the data covariance matrix  $\mathbf{S}$ , and the Lagrange multiplier  $\lambda_1$  plays the role of the corresponding eigenvalue. This eigenvector property (10.13) allows us to rewrite our variance objective (10.10) as

$$V_1 = \mathbf{b}_1^\top \mathbf{S} \mathbf{b}_1 = \lambda_1 \mathbf{b}_1^\top \mathbf{b}_1 = \lambda_1, \tag{10.15}$$

i.e., the variance of the data projected onto a one-dimensional subspace equals the eigenvalue that is associated with the basis vector  $\mathbf{b}_1$  that spans this subspace. Therefore, to maximize the variance of the low-dimensional code we choose the basis vector associated with the largest eigenvalue of the data covariance matrix. This eigenvector is called the *first principal component*. We can determine the effect/contribution of the principal component  $\mathbf{b}_1$  in the original data space by mapping the coordinate  $z_{1n}$  back into data space, which gives us the projected data point

$$\tilde{\mathbf{x}}_n = \mathbf{b}_1 z_{1n} = \mathbf{b}_1 \mathbf{b}_1^\top \mathbf{x}_n \in \mathbb{R}^D \tag{10.16}$$

5931 in the original data space.

5932 *Remark.* Although  $\tilde{\mathbf{x}}_n$  is a  $D$ -dimensional vector it only requires a single  
5933 coordinate  $z_{1n}$  to represent it with respect to the basis vector  $\mathbf{b}_1 \in \mathbb{R}^D$ .  $\diamond$

The quantity  $\sqrt{\lambda_1}$  is also called the *loading* of the unit vector  $\mathbf{b}_1$  and represents the standard deviation of the data accounted for by the principal subspace  $\text{span}\{\mathbf{b}_1\}$ .  
principal component

5934 **10.2.2  $M$ -dimensional Subspace with Maximal Variance**

Assume we have found the first  $m - 1$  principal components as the  $m - 1$  eigenvectors of  $\mathbf{S}$  that are associated with the largest  $m - 1$  eigenvalues. Since  $\mathbf{S}$  is symmetric, these eigenvectors form an orthonormal basis (ONB) of an  $(m - 1)$ -dimensional subspace of  $\mathbb{R}^D$ . Generally, the  $m$ th

principal component can be found by subtracting the effect of the first  $m - 1$  principal components  $\mathbf{b}_1, \dots, \mathbf{b}_{m-1}$  from the data, thereby trying to find principal components that compress the remaining information. We achieve this by first subtracting the contribution of the  $m - 1$  principal components from the data, similar to (10.16), so that we arrive at the new data matrix

$$\hat{\mathbf{X}} := \mathbf{X} - \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top \mathbf{X}, \quad (10.17)$$

5935 where  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$  contains the data points as column  
 5936 vectors. The matrix  $\hat{\mathbf{X}} := [\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_N] \in \mathbb{R}^{D \times N}$  in (10.17) contains the  
 5937 data that only contains the information that has not yet been compressed.

5938 *Remark* (Notation). Throughout this chapter, we do not follow the con-  
 5939 vention of collecting data  $\mathbf{x}_1, \dots, \mathbf{x}_N$  as the rows of the data matrix, but  
 5940 we define them to be the columns of  $\mathbf{X}$ . This means that our data ma-  
 5941 trix  $\mathbf{X}$  is a  $D \times N$  matrix instead of the conventional  $N \times D$  matrix. The  
 5942 reason for our choice is that the algebra operations work out smoothly  
 5943 without the need to either transpose the matrix or to redefine vectors as  
 5944 row vectors that are left-multiplied onto matrices.  $\diamond$

To find the  $m$ th principal component, we maximize the variance

$$V_m = \mathbb{V}[z_m] = \frac{1}{N} \sum_{n=1}^N z_{mn}^2 = \frac{1}{N} \sum_{n=1}^N (\mathbf{b}_m^\top \mathbf{x}_n)^2 = \mathbf{b}_m^\top \hat{\mathbf{S}} \mathbf{b}_m, \quad (10.18)$$

5945 subject to  $\|\mathbf{b}_m\|^2 = 1$ , where we followed the same steps as in (10.9b)  
 5946 and defined  $\hat{\mathbf{S}}$  as the data covariance matrix of the transformed dataset  
 5947  $\hat{\mathcal{X}} := \{\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_N\}$ . As previously, when we looked at the first principal  
 5948 component alone, we solve a constrained optimization problem and dis-  
 5949 cover that the optimal solution  $\mathbf{b}_m$  is the eigenvector of  $\hat{\mathbf{S}}$  that is associated  
 5950 with the largest eigenvalue of  $\hat{\mathbf{S}}$ .

However, it also turns out that  $\mathbf{b}_m$  is an eigenvector of  $\mathbf{S}$ . It holds that

$$\hat{\mathbf{S}} = \frac{1}{N} \sum_{n=1}^N \hat{\mathbf{x}}_n \hat{\mathbf{x}}_n^\top \stackrel{(10.17)}{=} \frac{1}{N} \sum_{n=1}^N \left( \mathbf{x}_n - \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top \mathbf{x}_n \right) \left( \mathbf{x}_n - \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top \mathbf{x}_n \right)^\top \quad (10.19a)$$

$$= \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top - 2 \mathbf{x}_n \mathbf{x}_n^\top \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top + \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top \mathbf{x}_n \mathbf{x}_n^\top \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top, \quad (10.19b)$$

where we exploited the symmetries  $\mathbf{x}_n^\top \mathbf{b}_i = \mathbf{b}_i^\top \mathbf{x}_n$  and  $\mathbf{b}_i \mathbf{x}_n^\top = \mathbf{x}_n \mathbf{b}_i^\top$  to summarize

$$-\mathbf{x}_n \mathbf{x}_n^\top \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top - \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top \mathbf{x}_n \mathbf{x}_n^\top = -2 \mathbf{x}_n \mathbf{x}_n^\top \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top. \quad (10.20)$$

If we take a vector  $\mathbf{b}_m$  with  $\|\mathbf{b}_m\| = 1$  that is orthogonal to all  $\mathbf{b}_1, \dots, \mathbf{b}_{m-1}$  and right-multiply  $\mathbf{b}_m$  to  $\hat{\mathbf{S}}$  in (10.19b) we obtain

$$\hat{\mathbf{S}}\mathbf{b}_m = \frac{1}{N} \sum_{n=1}^N \hat{\mathbf{x}}_n \hat{\mathbf{x}}_n^\top \mathbf{b}_m = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top \mathbf{b}_m = \mathbf{S}\mathbf{b}_m = \lambda_m \mathbf{b}_m. \quad (10.21)$$

5951 Here we applied the orthogonality property  $\mathbf{b}_i^\top \mathbf{b}_m = 0$  for  $i = 1, \dots, m-1$   
 5952 (all terms involving sums up to  $m-1$  vanish). Equation (10.21) reveals  
 5953 that  $\mathbf{b}_m$  is an eigenvector of both  $\hat{\mathbf{S}}$  and the original data covariance ma-  
 5954 trix  $\mathbf{S}$ . In other words,  $\lambda_m$  is the largest eigenvalue of  $\hat{\mathbf{S}}$  and  $\lambda_m$  is the  
 5955  $m$ th largest eigenvalue of  $\mathbf{S}$ , and both have the associated eigenvector  
 5956  $\mathbf{b}_m$ . This derivation shows that there is an intimate connection between  
 5957 the  $M$ -dimensional subspace with maximal variance and the eigenvalue  
 5958 decomposition. We will revisit this connection in Section 10.4.

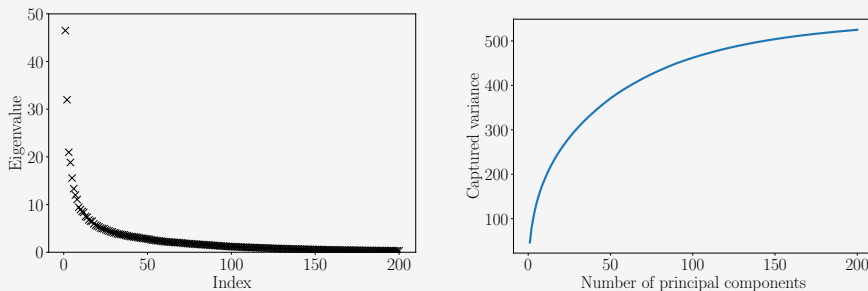
With the relation (10.21) and  $\mathbf{b}_m^\top \mathbf{b}_m = 1$  the variance of the data projected onto the  $m$ th principal component is

$$V_m = \mathbf{b}_m^\top \mathbf{S} \mathbf{b}_m \stackrel{(10.21)}{=} \lambda_m \mathbf{b}_m^\top \mathbf{b}_m = \lambda_m. \quad (10.22)$$

This means that the variance of the data, when projected onto an  $M$ -dimensional subspace, equals the sum of the eigenvalues that are associated with the corresponding eigenvectors of the data covariance matrix.

**Example 10.2 (Eigenvalues of MNIST ‘8’)**

Taking all digits ‘8’ in the MNIST training data, we compute the eigenvalues of the data covariance matrix.



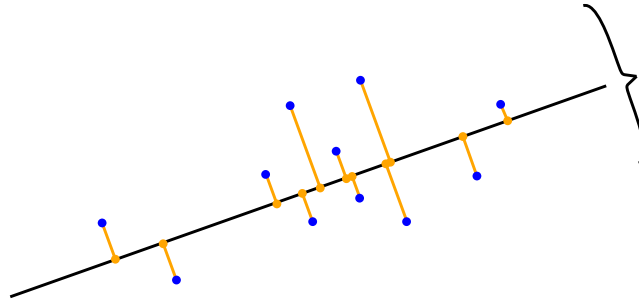
(a) Eigenvalues (sorted in descending order) of the data covariance matrix of all digits ‘8’ in the MNIST training set. (b) Variance captured by the principal components.

**Figure 10.5** Properties of the training data of MNIST ‘8’. (a) Eigenvalues sorted in descending order; (b) Variance captured by the principal components associated with the largest eigenvalues.

Figure 10.5(a) shows the 200 largest eigenvalues of the data covariance matrix. We see that only a few of them have a value that differs significantly from 0. Therefore, most of the variance, when projecting data onto the subspace spanned by the corresponding eigenvectors, is captured by only a few principal components as shown in Figure 10.5(b).



**Figure 10.6**  
 Illustration of the projection approach: Find a subspace (line) that minimizes the length of the difference vector between projected (orange) and original (blue) data.



Overall, to find an  $M$ -dimensional subspace of  $\mathbb{R}^D$  that retains as much information as possible, PCA tells us to choose the columns of the matrix  $\mathbf{B}$  in (10.3) as the  $M$  eigenvectors of the data covariance matrix  $\mathbf{S}$  that are associated with the  $M$  largest eigenvalues. The maximum amount of variance PCA can capture with the first  $M$  principal components is

$$V_M = \sum_{m=1}^M \lambda_m, \quad (10.23)$$

where the  $\lambda_m$  are the  $M$  largest eigenvalues of the data covariance matrix  $\mathbf{S}$ . Consequently, the variance lost by data compression via PCA is

$$J_M := \sum_{j=M+1}^D \lambda_j = V_D - V_M. \quad (10.24)$$

5959 Instead of these absolute quantities, we can define the relative variance  
 5960 captured as  $\frac{V_M}{V_D}$ , and the relative variance lost by compression as  $1 - \frac{V_M}{V_D}$ .

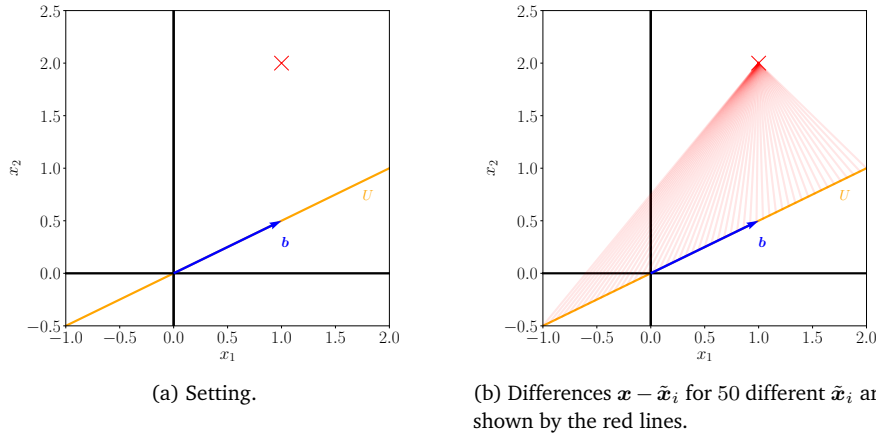
### 5961 10.3 Projection Perspective

5962 In the following, we will derive PCA as an algorithm that directly mini-  
 5963 mizes the average reconstruction error. This perspective allows us to in-  
 5964 terpret PCA as implementing an optimal linear auto-encoder. We will draw  
 5965 heavily from Chapters 2 and 3.

5966 In the previous section, we derived PCA by maximizing the variance  
 5967 in the projected space to retain as much information as possible. In the  
 5968 following, we will look at the difference vectors between the original data  
 5969  $\mathbf{x}_n$  and their reconstruction  $\tilde{\mathbf{x}}_n$  and minimize this distance so that  $\mathbf{x}_n$  and  
 5970  $\tilde{\mathbf{x}}_n$  are as close as possible. Figure 10.6 illustrates this setting.

#### 5971 10.3.1 Setting and Objective

5972 Assume an (ordered) orthonormal basis (ONB)  $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_D)$  of  $\mathbb{R}^D$ ,  
 5973 i.e.,  $\mathbf{b}_i^\top \mathbf{b}_j = 1$  if and only if  $i = j$  and 0 otherwise.



**Figure 10.7**  
Simplified projection setting. (a) A vector  $\mathbf{x} \in \mathbb{R}^2$  (red cross) shall be projected onto a one-dimensional subspace  $U \subseteq \mathbb{R}^2$  spanned by  $\mathbf{b}$ . (b) shows the difference vectors between  $\mathbf{x}$  and some candidates  $\tilde{\mathbf{x}}$ .

*Remark.* (Orthogonal Complement) Consider a  $D$ -dimensional vector space  $V$  and an  $M$ -dimensional subspace  $U \subseteq V$ . Then its *orthogonal complement*  $U^\perp$  is a  $(D - M)$ -dimensional subspace of  $V$  and contains all vectors in  $V$  that are orthogonal to every vector in  $U$ . Furthermore,  $U \cap U^\perp = \{\mathbf{0}\}$  so that any vector  $\mathbf{x} \in V$  can be (uniquely) decomposed into

orthogonal complement

$$\mathbf{x} = \sum_{m=1}^M \lambda_m \mathbf{b}_m + \sum_{j=1}^{D-M} \psi_j \mathbf{b}_j^\perp, \quad \lambda_m, \psi_j \in \mathbb{R}, \quad (10.25)$$

5974 where  $(\mathbf{b}_1, \dots, \mathbf{b}_M)$  is a basis of  $U$  and  $(\mathbf{b}_1^\perp, \dots, \mathbf{b}_{D-M}^\perp)$  is a basis of  $U^\perp$ .  
5975  $\diamond$

From Section 2.5 we know that for a basis  $(\mathbf{b}_1, \dots, \mathbf{b}_D)$  of  $\mathbb{R}^D$  any  $\mathbf{x} \in \mathbb{R}^D$  can be written as a linear combination of the basis vectors of  $\mathbb{R}^D$ , i.e.,

$$\mathbf{x} = \sum_{d=1}^D \zeta_d \mathbf{b}_d = \sum_{m=1}^M \zeta_m \mathbf{b}_m + \sum_{j=M+1}^D \zeta_j \mathbf{b}_j \quad (10.26)$$

5976 for suitable coordinates  $\zeta_d \in \mathbb{R}$ .

We are interested in finding vectors  $\tilde{\mathbf{x}} \in \mathbb{R}^D$ , which live in lower-dimensional subspace  $U \subseteq \mathbb{R}^D$ ,  $\dim(U) = M$ , so that

Vectors  $\tilde{\mathbf{x}} \in U$  could be vectors on a plane in  $\mathbb{R}^3$ . The dimensionality of the plane is 2, but the vectors still have three coordinates with respect to the standard basis of  $\mathbb{R}^3$ .

$$\tilde{\mathbf{x}} = \sum_{m=1}^M z_m \mathbf{b}_m \in U \subseteq \mathbb{R}^D \quad (10.27)$$

5977 is as similar to  $\mathbf{x}$  as possible. Note that at this point we need to assume  
5978 that the coordinates  $z_m$  of  $\tilde{\mathbf{x}}$  and  $\zeta_m$  of  $\mathbf{x}$  are not identical.

5979 In the following, we use exactly this kind of representation of  $\tilde{\mathbf{x}}$  to find  
5980 optimal coordinates  $\mathbf{z}$  and basis vectors  $\mathbf{b}_1, \dots, \mathbf{b}_M$  such that  $\tilde{\mathbf{x}}$  is as similar  
5981 to the original data point  $\mathbf{x}$ , i.e., we aim to minimize the (Euclidean)  
5982 distance  $\|\mathbf{x} - \tilde{\mathbf{x}}\|$ . Figure 10.7 illustrates this setting.

5983 Without loss of generality, we assume that the dataset  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ ,

5984  $\mathbf{x}_n \in \mathbb{R}^D$ , is centered at  $\mathbf{0}$ , i.e.,  $\mathbb{E}[\mathcal{X}] = \mathbf{0}$ . Without the zero-mean assumption,  
 5985 tion, we would arrive at exactly the same solution but the notation would  
 5986 be substantially more cluttered.

principal subspace

We are interested in finding the best linear projection of  $\mathcal{X}$  onto a lower-dimensional subspace  $U$  of  $\mathbb{R}^D$  with  $\dim(U) = M$  and orthonormal basis vectors  $\mathbf{b}_1, \dots, \mathbf{b}_M$ . We will call this subspace  $U$  the *principal subspace*. The projections of the data points are denoted by

$$\tilde{\mathbf{x}}_n := \sum_{m=1}^M z_{mn} \mathbf{b}_m = \mathbf{B} \mathbf{z}_n \in \mathbb{R}^D, \quad (10.28)$$

5987 where  $\mathbf{z}_n := [z_{1n}, \dots, z_{Mn}]^\top \in \mathbb{R}^M$  is the coordinate vector of  $\tilde{\mathbf{x}}_n$  with  
 5988 respect to the basis  $(\mathbf{b}_1, \dots, \mathbf{b}_M)$ . More specifically, we are interested in  
 5989 having the  $\tilde{\mathbf{x}}_n$  as similar to  $\mathbf{x}_n$  as possible.

reconstruction error

The similarity measure we use in the following is the squared Euclidean norm  $\|\mathbf{x} - \tilde{\mathbf{x}}\|^2$  between  $\mathbf{x}$  and  $\tilde{\mathbf{x}}$ . We therefore define our objective as the minimizing the average squared Euclidean distance (*reconstruction error*) (Pearson, 1901b)

$$J_M := \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2, \quad (10.29)$$

5990 where we make it explicit that the dimension of the subspace onto which  
 5991 we project the data is  $M$ . In order to find this optimal linear projection,  
 5992 we need to find the orthonormal basis of the principal subspace and the  
 5993 coordinates  $\mathbf{z}_n \in \mathbb{R}^M$  of the projections with respect to this basis.

5994 To find the coordinates  $\mathbf{z}_n$  and the ONB of the principal subspace we  
 5995 follow a two-step approach. First, we optimize the coordinates  $\mathbf{z}_n$  for a  
 5996 given ONB  $(\mathbf{b}_1, \dots, \mathbf{b}_M)$ ; second, we find the optimal ONB.

### 10.3.2 Finding Optimal Coordinates

5997 Let us start by finding the optimal coordinates  $z_{1n}, \dots, z_{Mn}$  of the projec-  
 5998 tions  $\tilde{\mathbf{x}}_n$  for  $n = 1, \dots, N$ . Consider Figure 10.7(b) where the principal  
 5999 subspace is spanned by a single vector  $\mathbf{b}$ . Geometrically speaking, finding  
 6000 the optimal coordinates  $z$  corresponds to finding the representation of the  
 6001 linear projection  $\tilde{\mathbf{x}}$  with respect to  $\mathbf{b}$  that minimizes the distance between  
 6002  $\tilde{\mathbf{x}} - \mathbf{x}$ . From Figure 10.7(b) it is clear that this will be the orthogonal  
 6003 projection, and in the following we will show exactly this.  
 6004

We assume an ONB  $(\mathbf{b}_1, \dots, \mathbf{b}_M)$  of  $U \subseteq \mathbb{R}^D$ . To find the optimal coordinates  $\mathbf{z}_m$  with respect to this basis, we require the partial derivatives

$$\frac{\partial J_M}{\partial z_{in}} = \frac{\partial J_M}{\partial \tilde{\mathbf{x}}_n} \frac{\partial \tilde{\mathbf{x}}_n}{\partial z_{in}}, \quad (10.30a)$$

$$\frac{\partial J_M}{\partial \tilde{\mathbf{x}}_n} = -\frac{2}{N} (\mathbf{x}_n - \tilde{\mathbf{x}}_n)^\top \in \mathbb{R}^{1 \times D}, \quad (10.30b)$$

$$\frac{\partial \tilde{\mathbf{x}}_n}{\partial z_{in}} \stackrel{(10.28)}{=} \frac{\partial}{\partial z_{in}} \left( \sum_{m=1}^M z_{mn} \mathbf{b}_m \right) = \mathbf{b}_i \quad (10.30c)$$

for  $i = 1, \dots, M$ , such that we obtain

$$\frac{\partial J_M}{\partial z_{in}} \stackrel{(10.30b)}{\stackrel{(10.30c)}{=}} -\frac{2}{N} (\mathbf{x}_n - \tilde{\mathbf{x}}_n)^\top \mathbf{b}_i \stackrel{(10.28)}{=} -\frac{2}{N} \left( \mathbf{x}_n - \sum_{m=1}^M z_{mn} \mathbf{b}_m \right)^\top \mathbf{b}_i \quad (10.31a)$$

$$\stackrel{\text{ONB}}{=} -\frac{2}{N} (\mathbf{x}_n^\top \mathbf{b}_i - z_{in} \mathbf{b}_i^\top \mathbf{b}_i) = -\frac{2}{N} (\mathbf{x}_n^\top \mathbf{b}_i - z_{in}). \quad (10.31b)$$

since  $\mathbf{b}_i^\top \mathbf{b}_i = 1$ . Setting this partial derivative to 0 yields immediately the optimal coordinates

$$z_{in} = \mathbf{x}_n^\top \mathbf{b}_i = \mathbf{b}_i^\top \mathbf{x}_n \quad (10.32)$$

for  $i = 1, \dots, M$  and  $n = 1, \dots, N$ . This means that the optimal coordinates  $z_{in}$  of the projection  $\tilde{\mathbf{x}}_n$  are the coordinates of the orthogonal projection (see Section 3.7) of the original data point  $\mathbf{x}_n$  onto the one-dimensional subspace that is spanned by  $\mathbf{b}_i$ . Consequently:

- The optimal linear projection  $\tilde{\mathbf{x}}_n$  of  $\mathbf{x}_n$  is an orthogonal projection.
- The coordinates of  $\tilde{\mathbf{x}}_n$  with respect to the basis  $(\mathbf{b}_1, \dots, \mathbf{b}_M)$  are the coordinates of the orthogonal projection of  $\mathbf{x}_n$  onto the principal subspace.
- An orthogonal projection is the best linear mapping given the objective (10.29).
- The coordinates  $\zeta_m$  of  $\mathbf{x}$  in (10.26) and the coordinates  $z_m$  of  $\tilde{\mathbf{x}}$  in (10.27) must be identical for  $m = 1, \dots, M$  since  $U^\perp = \text{span}[\mathbf{b}_{M+1}, \dots, \mathbf{b}_D]$  is the orthogonal complement of  $U = \text{span}[\mathbf{b}_1, \dots, \mathbf{b}_M]$ .

The coordinates of the optimal projection of  $\mathbf{x}_n$  with respect to the basis vectors  $\mathbf{b}_1, \dots, \mathbf{b}_M$  are the coordinates of the orthogonal projection of  $\mathbf{x}_n$  onto the principal subspace.

*Remark* (Orthogonal Projections with Orthonormal Basis Vectors). Let us briefly recap orthogonal projections from Section 3.7. If  $(\mathbf{b}_1, \dots, \mathbf{b}_D)$  is an orthonormal basis of  $\mathbb{R}^D$  then

$$\tilde{\mathbf{x}} = \mathbf{b}_j (\mathbf{b}_j^\top \mathbf{b}_j)^{-1} \mathbf{b}_j^\top \mathbf{x} = \mathbf{b}_j \mathbf{b}_j^\top \mathbf{x} \in \mathbb{R}^D \quad (10.33)$$

is the orthogonal projection of  $\mathbf{x}$  onto the subspace spanned by the  $j$ th basis vector, and  $z_j = \mathbf{b}_j^\top \mathbf{x}$  is the coordinate of this projection with respect to the basis vector  $\mathbf{b}_j$  that spans that subspace since  $z_j \mathbf{b}_j = \tilde{\mathbf{x}}$ . Figure 10.8 illustrates this setting.

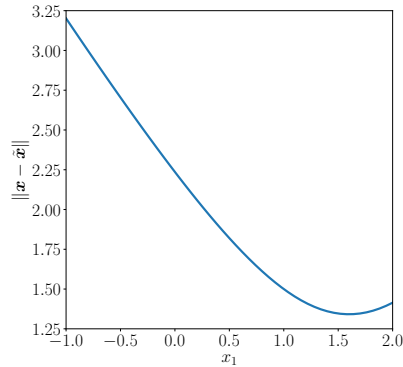
$\mathbf{b}_j^\top \mathbf{x}$  is the coordinate of the orthogonal projection of  $\mathbf{x}$  onto the subspace spanned by  $\mathbf{b}_j$ .

More generally, if we aim to project onto an  $M$ -dimensional subspace of  $\mathbb{R}^D$ , we obtain the orthogonal projection of  $\mathbf{x}$  onto the  $M$ -dimensional subspace with orthonormal basis vectors  $\mathbf{b}_1, \dots, \mathbf{b}_M$  as

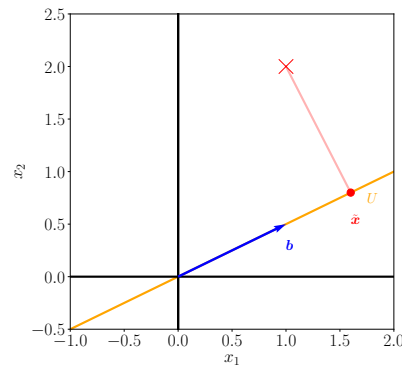
$$\tilde{\mathbf{x}} = \mathbf{B} \underbrace{(\mathbf{B}^\top \mathbf{B})^{-1}}_{=\mathbf{I}} \mathbf{B}^\top \mathbf{x} = \mathbf{B} \mathbf{B}^\top \mathbf{x}, \quad (10.34)$$

where we defined  $\mathbf{B} := [\mathbf{b}_1, \dots, \mathbf{b}_M] \in \mathbb{R}^{D \times M}$ . The coordinates of this

**Figure 10.8**  
Optimal projection  
of a vector  $\mathbf{x} \in \mathbb{R}^2$   
onto a  
one-dimensional  
subspace  
(continuation from  
Figure 10.7).  
(a) Distances  
 $\|\mathbf{x} - \tilde{\mathbf{x}}\|$  for some  
 $\tilde{\mathbf{x}} \in U$ .  
(b) Orthogonal  
projection and  
optimal coordinates.



(a) Distances  $\|\mathbf{x} - \tilde{\mathbf{x}}\|$  for some  $\tilde{\mathbf{x}} = z_1 \mathbf{b} \in U = \text{span}[\mathbf{b}]$ , see panel (b) for the setting.



(b) The vector  $\tilde{\mathbf{x}}$  that minimizes the distance in panel (a) is its orthogonal projection onto  $U$ . The coordinate of the projection  $\tilde{\mathbf{x}}$  with respect to the basis vector  $\mathbf{b}$  that spans  $U$  is the factor we need to scale  $\mathbf{b}$  in order to “reach”  $\tilde{\mathbf{x}}$ .

6023 projection with respect to the ordered basis  $(\mathbf{b}_1, \dots, \mathbf{b}_M)$  are  $\mathbf{z} := \mathbf{B}^\top \mathbf{x}$   
6024 as discussed in Section 3.7.

6025 We can think of the coordinates as a representation of the projected  
6026 vector in a new coordinate system defined by  $(\mathbf{b}_1, \dots, \mathbf{b}_M)$ . Note that al-  
6027 though  $\tilde{\mathbf{x}} \in \mathbb{R}^D$  we only need  $M$  coordinates  $z_1, \dots, z_M$  to represent this  
6028 vector; the other  $D - M$  coordinates with respect to the basis vectors  
6029  $(\mathbf{b}_{M+1}, \dots, \mathbf{b}_D)$  are always 0.  $\diamond$

6030 So far we have shown that for a given ONB we can find the optimal  
6031 coordinates of  $\tilde{\mathbf{x}}$  by an orthogonal projection onto the principal subspace.  
6032 In the following, we will determine what the best basis is.

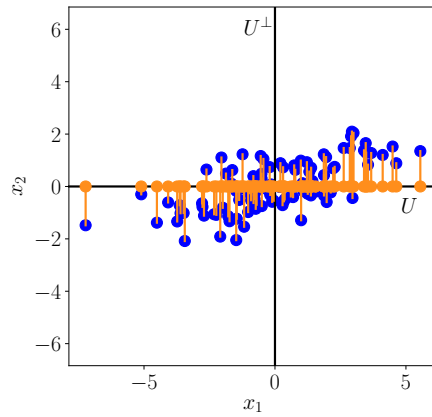
### 6033 10.3.3 Finding the Basis of the Principal Subspace

To determine the basis vectors  $\mathbf{b}_1, \dots, \mathbf{b}_M$  of the principal subspace, we rephrase the loss function (10.29) using the results we have so far. This will make it easier to find the basis vectors. To reformulate the loss function, we exploit our results from before and obtain

$$\tilde{\mathbf{x}}_n = \sum_{m=1}^M z_{mn} \mathbf{b}_m \stackrel{(10.32)}{=} \sum_{m=1}^M (\mathbf{x}_n^\top \mathbf{b}_m) \mathbf{b}_m. \quad (10.35)$$

We now exploit the symmetry of the dot product, which yields

$$\tilde{\mathbf{x}}_n = \left( \sum_{m=1}^M \mathbf{b}_m \mathbf{b}_m^\top \right) \mathbf{x}_n. \quad (10.36)$$



**Figure 10.9** Orthogonal projection and displacement vectors. When projecting data points  $\mathbf{x}_n$  (blue) onto subspace  $U_1$  we obtain  $\tilde{\mathbf{x}}_n$  (orange). The displacement vector  $\tilde{\mathbf{x}}_n - \mathbf{x}_n$  lies completely in the orthogonal complement  $U_2$  of  $U_1$ .

Since we can generally write the original data point  $\mathbf{x}_n$  as a linear combination of all basis vectors, it holds that

$$\mathbf{x}_n = \sum_{d=1}^D z_{dn} \mathbf{b}_d \stackrel{(10.32)}{=} \sum_{d=1}^D (\mathbf{x}_n^\top \mathbf{b}_d) \mathbf{b}_d = \left( \sum_{d=1}^D \mathbf{b}_d \mathbf{b}_d^\top \right) \mathbf{x}_n \quad (10.37a)$$

$$= \left( \sum_{m=1}^M \mathbf{b}_m \mathbf{b}_m^\top \right) \mathbf{x}_n + \left( \sum_{j=M+1}^D \mathbf{b}_j \mathbf{b}_j^\top \right) \mathbf{x}_n, \quad (10.37b)$$

where we split the sum with  $D$  terms into a sum over  $M$  and a sum over  $D - M$  terms. With this result, we find that the displacement vector  $\mathbf{x}_n - \tilde{\mathbf{x}}_n$ , i.e., the difference vector between the original data point and its projection, is

$$\mathbf{x}_n - \tilde{\mathbf{x}}_n = \left( \sum_{j=M+1}^D \mathbf{b}_j \mathbf{b}_j^\top \right) \mathbf{x}_n \quad (10.38a)$$

$$= \sum_{j=M+1}^D (\mathbf{x}_n^\top \mathbf{b}_j) \mathbf{b}_j. \quad (10.38b)$$

6034 This means the difference is exactly the projection of the data point onto  
 6035 the orthogonal complement of the principal subspace: We identify the matrix  
 6036  $\sum_{j=M+1}^D \mathbf{b}_j \mathbf{b}_j^\top$  in (10.38a) as the projection matrix that performs this  
 6037 projection. Hence, the displacement vector  $\mathbf{x}_n - \tilde{\mathbf{x}}_n$  lies in the subspace  
 6038 that is orthogonal to the principal subspace as illustrated in Figure 10.9.

*Remark (Low-Rank Approximation).* In (10.38a), we saw that the projection matrix, which projects  $\mathbf{x}$  onto  $\tilde{\mathbf{x}}$ , is given by

$$\sum_{m=1}^M \mathbf{b}_m \mathbf{b}_m^\top = \mathbf{B} \mathbf{B}^\top. \quad (10.39)$$

By construction as a sum of rank-one matrices  $\mathbf{b}_m \mathbf{b}_m^\top$  we see that  $\mathbf{B} \mathbf{B}^\top$  is

symmetric and has rank  $M$ . Therefore, the average squared reconstruction error can also be written as

$$\frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \mathbf{B}\mathbf{B}^\top \mathbf{x}_n\|^2 \quad (10.40a)$$

$$= \frac{1}{N} \sum_{n=1}^N \|(\mathbf{I} - \mathbf{B}\mathbf{B}^\top) \mathbf{x}_n\|^2. \quad (10.40b)$$

PCA finds the best  
rank- $M$   
approximation of  
the identity matrix.

Finding orthonormal basis vectors  $\mathbf{b}_1, \dots, \mathbf{b}_M$ , which minimize the difference between the original data  $\mathbf{x}_n$  and their projections  $\tilde{\mathbf{x}}_n$ , is equivalent to finding the best rank- $M$  approximation  $\mathbf{B}\mathbf{B}^\top$  of the identity matrix  $\mathbf{I}$  (see Section 4.6).  $\diamond$

Now, we have all the tools to reformulate the loss function (10.29).

$$J_M = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 \stackrel{(10.38b)}{=} \frac{1}{N} \sum_{n=1}^N \left\| \sum_{j=M+1}^D (\mathbf{b}_j^\top \mathbf{x}_n) \mathbf{b}_j \right\|^2. \quad (10.41)$$

We now explicitly compute the squared norm and exploit the fact that the  $\mathbf{b}_j$  form an ONB, which yields

$$J_M = \frac{1}{N} \sum_{n=1}^N \sum_{j=M+1}^D (\mathbf{b}_j^\top \mathbf{x}_n)^2 = \frac{1}{N} \sum_{n=1}^N \sum_{j=M+1}^D \mathbf{b}_j^\top \mathbf{x}_n \mathbf{b}_j^\top \mathbf{x}_n \quad (10.42a)$$

$$= \frac{1}{N} \sum_{n=1}^N \sum_{j=M+1}^D \mathbf{b}_j^\top \mathbf{x}_n \mathbf{x}_n^\top \mathbf{b}_j, \quad (10.42b)$$

where we exploited the symmetry of the dot product in the last step to write  $\mathbf{b}_j^\top \mathbf{x}_n = \mathbf{x}_n^\top \mathbf{b}_j$ . We now swap the sums and obtain

$$J_M = \sum_{j=M+1}^D \mathbf{b}_j^\top \left( \underbrace{\frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top}_{=: \mathbf{S}} \right) \mathbf{b}_j = \sum_{j=M+1}^D \mathbf{b}_j^\top \mathbf{S} \mathbf{b}_j \quad (10.43a)$$

$$= \sum_{j=M+1}^D \text{tr}(\mathbf{b}_j^\top \mathbf{S} \mathbf{b}_j) \sum_{j=M+1}^D \text{tr}(\mathbf{S} \mathbf{b}_j \mathbf{b}_j^\top) = \text{tr} \left( \underbrace{\left( \sum_{j=M+1}^D \mathbf{b}_j \mathbf{b}_j^\top \right)}_{\text{projection matrix}} \mathbf{S} \right), \quad (10.43b)$$

where we exploited the property that the trace operator  $\text{tr}(\cdot)$ , see (4.18), is linear and invariant to cyclic permutations of its arguments. Since we assumed that our dataset is centered, i.e.,  $\mathbb{E}[\mathcal{X}] = \mathbf{0}$ , we identify  $\mathbf{S}$  as the data covariance matrix. Since the projection matrix in (10.43b) is constructed as a sum of rank-one matrices  $\mathbf{b}_j \mathbf{b}_j^\top$  it itself is of rank  $D - M$ .

Equation (10.43a) implies that we can formulate the average squared reconstruction error equivalently as the covariance matrix of the data,

6050 projected onto the orthogonal complement of the principal subspace. Minimizing the average squared reconstruction error is therefore equivalent to  
 6051 minimizing the variance of the data when projected onto the subspace we  
 6052 ignore, i.e., the orthogonal complement of the principal subspace. Equiva-  
 6053 lently, we maximize the variance of the projection that we retain in the  
 6054 principal subspace, which links the projection loss immediately to the  
 6055 maximum-variance formulation of PCA discussed in Section 10.2. But this  
 6056 then also means that we will obtain the same solution that we obtained  
 6057 for the maximum-variance perspective. Therefore, we omit a derivation  
 6058 that is identical to the one Section 10.2 and summarize the results from  
 6059 earlier in the light of the projection perspective.  
 6060

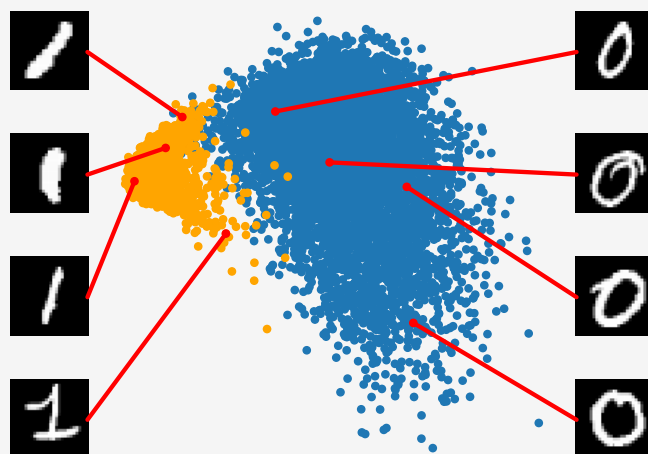
The average squared reconstruction error, when projecting onto the  $M$ -dimensional principal subspace, is

$$J_M = \sum_{j=M+1}^D \lambda_j, \quad (10.44)$$

6061 where  $\lambda_j$  are the eigenvalues of the data covariance matrix. Therefore,  
 6062 to minimize (10.44) we need to select the smallest  $D - M$  eigenvalues,  
 6063 which then implies that their corresponding eigenvectors are the basis  
 6064 of the orthogonal complement of the principal subspace. Consequently,  
 6065 this means that the basis of the principal subspace are the eigenvectors  
 6066  $\mathbf{b}_1, \dots, \mathbf{b}_M$  that are associated with the largest  $M$  eigenvalues of the data  
 6067 covariance matrix.

Minimizing the average squared reconstruction error is equivalent to minimizing the projection of the data covariance matrix onto the orthogonal complement of the principal subspace. Minimizing the average squared reconstruction error is equivalent to maximizing the variance of the projected data.

### Example 10.3 (MNIST Digits Embedding)



**Figure 10.10** Embedding of MNIST digits 0 (blue) and 1 (orange) in a two-dimensional principal subspace using PCA. Four embeddings of the digits '0' and '1' in the principal subspace are highlighted in red with their corresponding original digit.



Figure 10.10 visualizes the training data of the MMIST digits ‘0’ and ‘1’ embedded in the vector subspace spanned by the first two principal components. We observe a relatively clear separation between ‘0’s (blue dots) and ‘1’s (orange dots), and we see the variation within each individual cluster. Four embeddings of the digits ‘0’ and ‘1’ in the principal subspace are highlighted in red with their corresponding original digit. The figure reveals that the variation within the set of ‘0’ is significantly greater than the variation within the set of ‘1’.

#### 10.4 Eigenvector Computation and Low-Rank Approximations

In the previous sections, we obtained the basis of the principal subspace as the eigenvectors that are associated with the largest eigenvalues of the data covariance matrix

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top = \frac{1}{N} \mathbf{X} \mathbf{X}^\top, \quad (10.45)$$

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}. \quad (10.46)$$

Note that  $\mathbf{X}$  is a  $D \times N$  matrix, i.e., it is the transpose of the “typical” data matrix (Bishop, 2006; Murphy, 2012). To get the eigenvalues (and the corresponding eigenvectors) of  $\mathbf{S}$ , we can follow two approaches:

Eigendecomposition  
or SVD to compute  
eigenvectors.

- We perform an eigendecomposition (see Section 4.2) and compute the eigenvalues and eigenvectors of  $\mathbf{S}$  directly.
- We use a singular value decomposition (see Section 4.5). Since  $\mathbf{S}$  is symmetric and factorizes into  $\mathbf{X} \mathbf{X}^\top$  (ignoring the factor  $\frac{1}{N}$ ), the eigenvalues of  $\mathbf{S}$  are the squared singular values of  $\mathbf{X}$ .

More specifically, the SVD of  $\mathbf{X}$  is given by

$$\underbrace{\mathbf{X}}_{D \times N} = \underbrace{\mathbf{U}}_{D \times D} \underbrace{\mathbf{\Sigma}}_{D \times N} \underbrace{\mathbf{V}^\top}_{N \times N}, \quad (10.47)$$

where  $\mathbf{U} \in \mathbb{R}^{D \times D}$  and  $\mathbf{V}^\top \in \mathbb{R}^{N \times N}$  are orthogonal matrices and  $\mathbf{\Sigma} \in \mathbb{R}^{D \times N}$  is a matrix whose only non-zero entries are the singular values  $\sigma_{ii} \geq 0$ . It then follows that

$$\mathbf{S} = \frac{1}{N} \mathbf{X} \mathbf{X}^\top = \frac{1}{N} \mathbf{U} \mathbf{\Sigma} \underbrace{\mathbf{V}^\top \mathbf{V}}_{=\mathbf{I}_N} \mathbf{\Sigma}^\top \mathbf{U}^\top = \frac{1}{N} \mathbf{U} \mathbf{\Sigma} \mathbf{\Sigma}^\top \mathbf{U}^\top. \quad (10.48)$$

The columns of  $\mathbf{U}$   
are the eigenvectors  
of  $\mathbf{S}$ .

With the results from Section 4.5 we get that the columns of  $\mathbf{U}$  are the eigenvectors of  $\mathbf{X} \mathbf{X}^\top$  (and therefore  $\mathbf{S}$ ). Furthermore, the eigenvalues  $\lambda_d$  of  $\mathbf{S}$  are related to the singular values of  $\mathbf{X}$  via

$$\lambda_d = \frac{\sigma_d^2}{N}. \quad (10.49)$$

6077 This relationship between the eigenvalues of  $\mathbf{S}$  and the singular values  
 6078 of  $\mathbf{X}$  provides the connection between the maximum variance view (Sec-  
 6079 tion 10.2) and the singular value decomposition.

### 6080 10.4.1 PCA using Low-rank Matrix Approximations

To maximize the variance of the projected data (or minimize the average squared reconstruction error), PCA chooses the columns of  $\mathbf{U}$  in (10.48) to be the eigenvectors that are associated with the  $M$  largest eigenvalues of the data covariance matrix  $\mathbf{S}$  so that we identify  $\mathbf{U}$  as the projection matrix  $\mathbf{B}$  in (10.3), which projects the original data onto a lower-dimensional subspace of dimension  $M$ . The *Eckart-Young Theorem* (Section 4.6) offers a direct way to estimate the low-dimensional representation. Consider the best rank- $M$  approximation

Eckart-Young  
Theorem

$$\tilde{\mathbf{X}}_M := \operatorname{argmin}_{\operatorname{rk}(\mathbf{A}) \leq M} \|\mathbf{X} - \mathbf{A}\|_2 \in \mathbb{R}^{D \times N} \quad (10.50)$$

of  $\mathbf{X}$ , where  $\|\cdot\|_2$  is the spectral norm defined in (4.110). The Eckart-Young Theorem states that  $\tilde{\mathbf{X}}_M$  is given by truncating the SVD at the top- $M$  singular value. In other words, we obtain

$$\tilde{\mathbf{X}}_M = \underbrace{\mathbf{U}_M}_{D \times M} \underbrace{\boldsymbol{\Sigma}_M}_{M \times M} \underbrace{\mathbf{V}_M^\top}_{M \times N} \in \mathbb{R}^{D \times N} \quad (10.51)$$

6081 with orthogonal matrices  $\mathbf{U}_M := [\mathbf{u}_1, \dots, \mathbf{u}_M] \in \mathbb{R}^{D \times M}$  and  $\mathbf{V}_M :=$   
 6082  $[\mathbf{v}_1, \dots, \mathbf{v}_M] \in \mathbb{R}^{N \times M}$  and a diagonal matrix  $\boldsymbol{\Sigma}_M \in \mathbb{R}^{M \times M}$  whose diago-  
 6083 nal entries are the  $M$  largest singular values of  $\mathbf{X}$ .

### 6084 10.4.2 Practical Aspects

6085 Finding eigenvalues and eigenvectors is also important in other funda-  
 6086 mental machine learning methods that require matrix decompositions. In  
 6087 theory, as we discussed in Section 4.2, we can solve for the eigenvalues as  
 6088 roots of the characteristic polynomial. However, for matrices larger than  
 6089  $4 \times 4$  this is not possible because we would need to find the roots of a poly-  
 6090 nomial of degree 5 or higher. However, the Abel-Ruffini theorem (Ruffini,  
 6091 1799; Abel, 1826) states that there exists no algebraic solution to this  
 6092 problem for polynomials of degree 5 or more. Therefore, in practice, we  
 6093 solve for eigenvalues or singular values using iterative methods, which are  
 6094 implemented in all modern packages for linear algebra.

`np.linalg.eigh`  
or  
`np.linalg.svd`

In many applications (such as PCA presented in this chapter), we only require a few eigenvectors. It would be wasteful to compute the full decomposition, and then discard all eigenvectors with eigenvalues that are beyond the first few. It turns out that if we are interested in only the first few eigenvectors (with the largest eigenvalues), then iterative processes,

power iteration

If  $\mathbf{S}$  is invertible, it is sufficient to ensure that  $\mathbf{x}_0 \neq \mathbf{0}$ .

which directly optimize these eigenvectors, are computationally more efficient than a full eigendecomposition (or SVD). In the extreme case of only needing the first eigenvector, a simple method called the *power iteration* is very efficient. Power iteration chooses a random vector  $\mathbf{x}_0$  that is not in the null space of  $\mathbf{S}$  and follows the iteration

$$\mathbf{x}_{k+1} = \frac{\mathbf{S}\mathbf{x}_k}{\|\mathbf{S}\mathbf{x}_k\|}, \quad k = 0, 1, \dots \quad (10.52)$$

6095 This means the vector  $\mathbf{x}_k$  is multiplied by  $\mathbf{S}$  in every iteration and then  
 6096 normalized, i.e., we always have  $\|\mathbf{x}_k\| = 1$ . This sequence of vectors converges  
 6097 to the eigenvector associated with the largest eigenvalue of  $\mathbf{S}$ . The  
 6098 original Google PageRank algorithm (Page et al., 1999) uses such an algorithm  
 6099 for ranking web pages based on their hyperlinks.

## 6100 10.5 PCA in High Dimensions

6101 In order to do PCA, we need to compute the data covariance matrix. In  $D$   
 6102 dimensions, the data covariance matrix is a  $D \times D$  matrix. Computing the  
 6103 eigenvalues and eigenvectors of this matrix is computationally expensive  
 6104 as it scales cubically in  $D$ . Therefore, PCA, as we discussed earlier, will be  
 6105 infeasible in very high dimensions. For example, if our  $\mathbf{x}_n$  are images with  
 6106 10,000 pixels (e.g.,  $100 \times 100$  pixel images), we would need to compute  
 6107 the eigendecomposition of a  $10,000 \times 10,000$  covariance matrix. In the  
 6108 following, we provide a solution to this problem for the case that we have  
 6109 substantially fewer data points than dimensions, i.e.,  $N \ll D$ .

Assume we have a data set  $\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{x}_n \in \mathbb{R}^D$ . Assuming the data is centered, the data covariance matrix is given as

$$\mathbf{S} = \frac{1}{N} \mathbf{X} \mathbf{X}^\top \in \mathbb{R}^{D \times D}, \quad (10.53)$$

6110 where  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$  is a  $D \times N$  matrix whose columns are the data  
 6111 points.

6112 We now assume that  $N \ll D$ , i.e., the number of data points is smaller  
 6113 than the dimensionality of the data. If there are no duplicate data points  
 6114 the rank of the covariance matrix  $\mathbf{S}$  is  $N$ , so it has  $D - N + 1$  many eigenvalues  
 6115 that are 0. Intuitively, this means that there are some redundancies.

6116 In the following, we will exploit this and turn the  $D \times D$  covariance  
 6117 matrix into an  $N \times N$  covariance matrix whose eigenvalues are all greater  
 6118 than 0.

In PCA, we ended up with the eigenvector equation

$$\mathbf{S}\mathbf{b}_m = \lambda_m \mathbf{b}_m, \quad m = 1, \dots, M, \quad (10.54)$$

where  $\mathbf{b}_m$  is a basis vector of the principal subspace. Let us re-write this

equation a bit: With  $\mathbf{S}$  defined in (10.53), we obtain

$$\mathbf{S}\mathbf{b}_m = \frac{1}{N}\mathbf{X}\mathbf{X}^\top\mathbf{b}_m = \lambda_m\mathbf{b}_m. \quad (10.55)$$

We now multiply  $\mathbf{X}^\top \in \mathbb{R}^{N \times D}$  from the left-hand side, which yields

$$\frac{1}{N}\underbrace{\mathbf{X}^\top\mathbf{X}}_{N \times N}\underbrace{\mathbf{X}^\top\mathbf{b}_m}_{=: \mathbf{c}_m} = \lambda_m\mathbf{X}^\top\mathbf{b}_m \iff \frac{1}{N}\mathbf{X}^\top\mathbf{X}\mathbf{c}_m = \lambda_m\mathbf{c}_m, \quad (10.56)$$

6119 and we get a new eigenvector/eigenvalue equation:  $\lambda_m$  remains eigen-  
 6120 value, which confirms our results from Section 4.5.3 that the non-zero  
 6121 eigenvalues of  $\mathbf{X}\mathbf{X}^\top$  equal the non-zero eigenvalues of  $\mathbf{X}^\top\mathbf{X}$ . We obtain  
 6122 the eigenvector of the matrix  $\frac{1}{N}\mathbf{X}^\top\mathbf{X} \in \mathbb{R}^{N \times N}$  associated with  $\lambda_m$   
 6123 as  $\mathbf{c}_m := \mathbf{X}^\top\mathbf{b}_m$ . Assuming we have no duplicate data points, this matrix  
 6124 has rank  $N$  and is invertible. This also implies that  $\frac{1}{N}\mathbf{X}^\top\mathbf{X}$  has the same  
 6125 (non-zero) eigenvalues as the data covariance matrix  $\mathbf{S}$ . But this is now  
 6126 an  $N \times N$  matrix, so that we can compute the eigenvalues and eigenvectors  
 6127 much more efficiently than for the original  $D \times D$  data covariance  
 6128 matrix.

Now, that we have the eigenvectors of  $\frac{1}{N}\mathbf{X}^\top\mathbf{X}$ , we are going to recover the original eigenvectors, which we still need for PCA. Currently, we know the eigenvectors of  $\frac{1}{N}\mathbf{X}^\top\mathbf{X}$ . If we left-multiply our eigenvalue/eigenvector equation with  $\mathbf{X}$ , we get

$$\frac{1}{N}\underbrace{\mathbf{X}\mathbf{X}^\top}_{\mathbf{S}}\mathbf{X}\mathbf{c}_m = \lambda_m\mathbf{X}\mathbf{c}_m \quad (10.57)$$

6129 and we recover the data covariance matrix again. This now also means  
 6130 that we recover  $\mathbf{X}\mathbf{c}_m$  as an eigenvector of  $\mathbf{S}$ .

6131 *Remark.* If we want to apply the PCA algorithm that we discussed in Sec-  
 6132 tion 10.6 we need to normalize the eigenvectors  $\mathbf{X}\mathbf{c}_m$  of  $\mathbf{S}$  so that they  
 6133 have norm 1.  $\diamond$

## 6134 10.6 Key Steps of PCA in Practice

6135 In the following, we will go through the individual steps of PCA using a  
 6136 running example, which is summarized in Figure 10.11. We are given a  
 6137 two-dimensional dataset (Figure 10.11(a)), and we want to use PCA to  
 6138 project it onto a one-dimensional subspace.

6139 **1 Mean subtraction** We start by centering the data by computing the  
 6140 mean  $\boldsymbol{\mu}$  of the dataset and subtracting it from every single data point.  
 6141 This ensures that the dataset has mean  $\mathbf{0}$  (Figure 10.11(b)). Mean sub-  
 6142 traction is not strictly necessary but reduces the risk of numerical prob-  
 6143 lems.

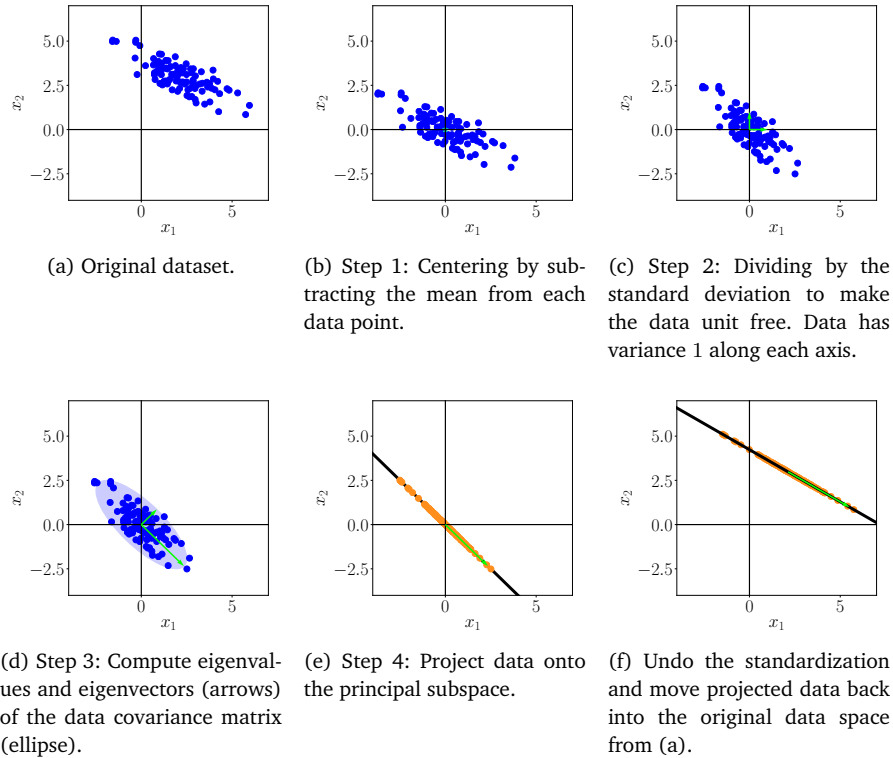


Figure 10.11 Steps of PCA.

- 6144 **2 Standardization** Divide the data points by the standard deviation  $\sigma_d$   
 6145 of the dataset for every dimension  $d = 1, \dots, D$ . Now the data is unit  
 6146 free, and it has variance 1 along each axis, which is indicated by the  
 standardization 6147 two arrows in Figure 10.11(c). This step completes the *standardization*  
 6148 of the data.
- 6149 **3 Eigendecomposition of the covariance matrix** Compute the data  
 6150 covariance matrix and its eigenvalues and corresponding eigenvectors.  
 6151 Since the covariance matrix is symmetric, the eigenvectors form an ortho-  
 6152 gonal basis. In Figure 10.11(d), the eigenvectors are scaled by the  
 6153 magnitude of the corresponding eigenvalue. The longer vector spans  
 6154 the principal subspace, which we denote by  $U$ . The data covariance  
 6155 matrix is represented by the ellipse.
- 4 Projection** We can project any data point  $\mathbf{x}_* \in \mathbb{R}^D$  onto the principal  
 subspace: To get this right, we need to standardize  $\mathbf{x}_*$  using the mean  
 $\mu_d$  and standard deviation  $\sigma_d$  of the training data in the  $d$ th dimension,  
 respectively, so that

$$x_*^{(d)} \leftarrow \frac{x_*^{(d)} - \mu_d}{\sigma_d}, \quad d = 1, \dots, D, \quad (10.58)$$

where  $x_*^{(d)}$  is the  $d$ th component of  $x_*$ . We obtain the projection as

$$\tilde{x}_* = BB^\top x_* \tag{10.59}$$

with coordinates

$$z_* = B^\top x_* \tag{10.60}$$

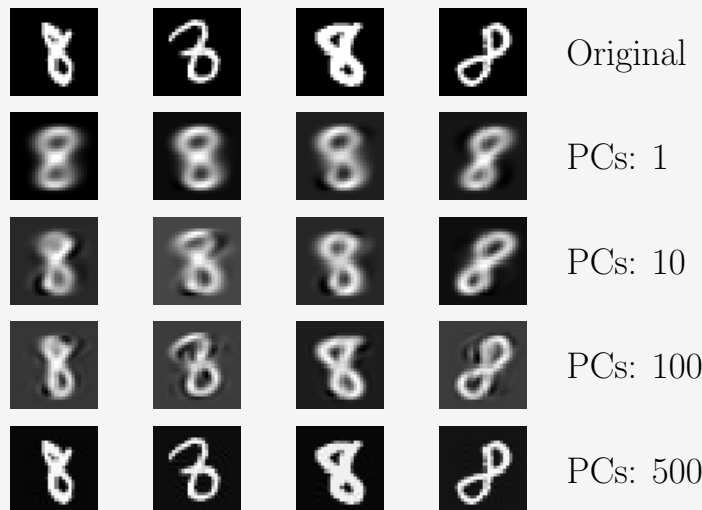
6156 with respect to the basis of the principal subspace. Here,  $B$  is the ma-  
 6157 trix that contains the eigenvectors that are associated with the largest  
 6158 eigenvalues of the data covariance matrix as columns. PCA returns the  
 6159 coordinates (10.60), not the projections  $x_*$ .

Having standardized our dataset, (10.59) only yields the projections in the context of the standardized dataset. To obtain our projection in the original data space (i.e., before standardization), we need to undo the standardization (10.58) and multiply by the standard deviation before adding the mean so that we obtain

$$\tilde{x}_*^{(d)} \leftarrow \tilde{x}_*^{(d)} \sigma_d + \mu_d, \quad d = 1, \dots, D. \tag{10.61}$$

6160 Figure 10.11(f) illustrates the projection in the original data space.

**Example 10.4 (MNIST Digits: Reconstruction)**



**Figure 10.12** Effect of increasing number of principal components on reconstruction.

In the following, we will apply PCA to the MNIST digits dataset, which contains 60,000 examples of handwritten digits 0–9. Each digit is an image of size  $28 \times 28$ , i.e., it contains 784 pixels so that we can interpret

<http://yann.lecun.com/exdb/mnist/>

every image in this dataset as a vector  $\mathbf{x} \in \mathbb{R}^{784}$ . Examples of these digits are shown in Figure 10.3. For illustration purposes, we apply PCA to a subset of the MNIST digits, and we focus on the digit ‘8’. We used 5,389 training images of the digit ‘8’ and determined the principal subspace as detailed in this chapter. We then used the learned projection matrix to reconstruct a set of test images, which is illustrated in Figure 10.12. The first row of Figure 10.12 shows a set of four original digits from the test set. The following rows show reconstructions of exactly these digits when using a principal subspace of dimensions 1, 10, 100, 500, respectively. We see that even with a single-dimensional principal subspace we get a half-way decent reconstruction of the original digits, which, however, is blurry and generic. With an increasing number of principal components (PCs) the reconstructions become sharper and more details are accounted for. With 500 principal components, we effectively obtain a near-perfect reconstruction. If we were to choose 784 PCs we would recover the exact digit without any compression loss.

**Figure 10.13**

Average squared reconstruction error as a function of the number of principal components. The average squared reconstruction error is the sum of the eigenvalues in the orthogonal complement of the principal subspace.

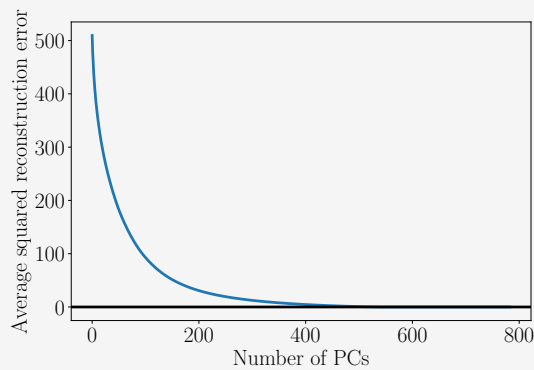


Figure 10.13 shows the average squared reconstruction error, which is

$$\frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 = \sum_{i=M+1}^D \lambda_i, \quad (10.62)$$

as a function of the number  $M$  of principal components. We can see that the importance of the principal components drops off rapidly, and only marginal gains can be achieved by adding more PCs. This matches exactly our observation in Figure 10.5 where we discovered that the most of the variance of the projected data is captured by only a few principal components. With about 550 PCs, we can essentially fully reconstruct the training data that contains the digit ‘8’ (some pixels around the boundaries show no variation across the dataset as they are always black).

## 10.7 Latent Variable Perspective

In the previous sections, we derived PCA without any notion of a probabilistic model using the maximum-variance and the projection perspectives. On the one hand this approach may be appealing as it allows us to sidestep all the mathematical difficulties that come with probability theory, but on the other hand a probabilistic model would offer us more flexibility and useful insights. More specifically, a probabilistic model would

- come with a likelihood function, and we can explicitly deal with noisy observations (which we did not even discuss earlier),
- allow us to do Bayesian model comparison via the marginal likelihood as discussed in Section 8.5,
- view PCA as a generative model, which allows us to simulate new data,
- allow us to make straightforward connections to related algorithms
- deal with data dimensions that are missing at random by applying Bayes' theorem,
- give us a notion of the novelty of a new data point,
- give us a principled way to extend the model, e.g., to a mixture of PCA models,
- have the PCA we derived in earlier sections as a special case,
- allow for a fully Bayesian treatment by marginalizing out the model parameters.

By introducing a continuous-valued latent variable  $\mathbf{z} \in \mathbb{R}^M$  it is possible to phrase PCA as a probabilistic latent-variable model. Tipping and Bishop (1999) proposed this latent-variable model as *Probabilistic PCA* (PPCA). PPCA addresses most of the issues above, and the PCA solution that we obtained by maximizing the variance in the projected space or by minimizing the reconstruction error is obtained as the special case of maximum likelihood estimation in a noise-free setting.

Probabilistic PCA

### 10.7.1 Generative Process and Probabilistic Model

In PPCA, we explicitly write down the probabilistic model for linear dimensionality reduction. For this we assume a continuous latent variable  $\mathbf{z} \in \mathbb{R}^M$  with a standard-Normal prior  $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$  and a linear relationship between the latent variables and the observed  $\mathbf{x}$  data where

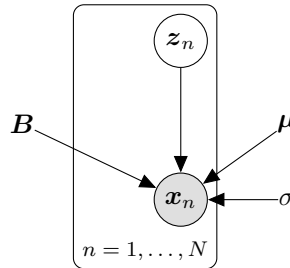
$$\mathbf{x} = \mathbf{B}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \in \mathbb{R}^D, \quad (10.63)$$

where  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$  is Gaussian observation noise,  $\mathbf{B} \in \mathbb{R}^{D \times M}$  and  $\boldsymbol{\mu} \in \mathbb{R}^D$  describe the linear/affine mapping from latent to observed variables. Therefore, PPCA links latent and observed variables via

$$p(\mathbf{x}|\mathbf{z}, \mathbf{B}, \boldsymbol{\mu}, \sigma^2) = \mathcal{N}(\mathbf{x} | \mathbf{B}\mathbf{z} + \boldsymbol{\mu}, \sigma^2 \mathbf{I}). \quad (10.64)$$



**Figure 10.14**  
Graphical model for probabilistic PCA. The observations  $\mathbf{x}_n$  explicitly depend on corresponding latent variables  $z_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . The model parameters  $\mathbf{B}$ ,  $\boldsymbol{\mu}$  and the likelihood parameter  $\sigma$  are shared across the dataset.



Overall, PPCA induces the following generative process:

$$z_n \sim \mathcal{N}(z | \mathbf{0}, \mathbf{I}) \quad (10.65)$$

$$\mathbf{x}_n | z_n \sim \mathcal{N}(\mathbf{x} | \mathbf{B}z_n + \boldsymbol{\mu}, \sigma^2 \mathbf{I}) \quad (10.66)$$

6190 To generate a data point that is typical given the model parameters, we  
ancestral sampling 6191 follow an *ancestral sampling* scheme: We first sample a latent variable  $z_n$   
6192 from  $p(z)$ . Then, we use  $z_n$  in (10.64) to sample a data point conditioned  
6193 on the sampled  $z_n$ , i.e.,  $\mathbf{x}_n \sim p(\mathbf{x} | z_n, \mathbf{B}, \boldsymbol{\mu}, \sigma^2)$ .

This generative process allows us to write down the probabilistic model (i.e., the joint distribution of all random variables, see Section 8.3) as

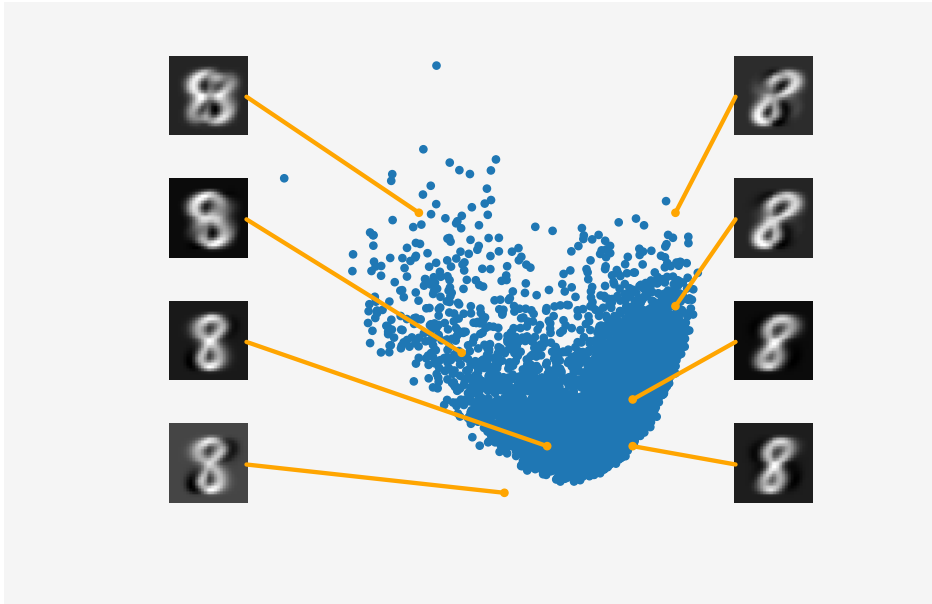
$$p(\mathbf{x}, \mathbf{z} | \mathbf{B}, \boldsymbol{\mu}, \sigma^2) = p(\mathbf{x} | \mathbf{z}, \mathbf{B}, \boldsymbol{\mu}, \sigma^2) p(\mathbf{z}), \quad (10.67)$$

6194 which immediately gives rise to the graphical model in Figure 10.14 using  
6195 the results from Section 8.4.

6196 *Remark.* Note the direction of the arrow that connects the latent variables  
6197  $\mathbf{z}$  and the observed data  $\mathbf{x}$ : The arrow points from  $\mathbf{z}$  to  $\mathbf{x}$ , which means  
6198 that the PPCA model assumes a lower-dimensional latent cause  $\mathbf{z}$  for high-  
6199 dimensional observations  $\mathbf{x}$ . In the end, we are obviously interested in  
6200 finding something out about  $\mathbf{z}$  given some observations. To get there we  
6201 will apply Bayesian inference to “invert” the arrow implicitly and go from  
6202 observations to latent variables.  $\diamond$

### Example 10.5 (Generating New Data using Latent Variables)

Figure 10.15 shows the latent coordinates of the MNIST digits ‘8’ found by PCA when using a two-dimensional principal subspace (blue dots). We can query any vector  $z_*$  in this latent space and generate an image  $\tilde{\mathbf{x}}_* = \mathbf{B}z_*$  that resembles the digit ‘8’. We show eight of such generated images with their corresponding latent space representation. Depending on where we query the latent space, the generated images look different (shape, rotation, size, ...). If we query away from the training data, we see more artefacts, e.g., the top-left and top-right digits. Note that the intrinsic dimensionality of these generated images is only two.



**Figure 10.15** Generating new MNIST digits. The latent variables  $z$  can be used to generate new data  $\tilde{x} = Bz$ . The closer we stay to the training data the more realistic the generated data.

6203

**10.7.2 Likelihood and Joint Distribution**

Using the results from Chapter 6, we obtain the likelihood of this probabilistic model by integrating out the latent variable  $z$  (see Section 8.3.4) so that

$$\begin{aligned}
 p(\mathbf{x} \mid \mathbf{B}, \boldsymbol{\mu}, \sigma^2) &= \int p(\mathbf{x} \mid z, \boldsymbol{\mu}, \sigma^2) p(z) dz \\
 &= \int \mathcal{N}(\mathbf{x} \mid \mathbf{B}z + \boldsymbol{\mu}, \sigma^2 \mathbf{I}) \mathcal{N}(z \mid \mathbf{0}, \mathbf{I}) dz.
 \end{aligned}
 \tag{10.68}$$

The likelihood does not depend on the latent variables  $z$ .

From Section 6.5, we know that the solution to this integral is a Gaussian distribution with mean

$$\mathbb{E}_{\mathbf{x}}[\mathbf{x}] = \mathbb{E}_z[\mathbf{B}z + \boldsymbol{\mu}] + \mathbb{E}_{\epsilon}[\epsilon] = \boldsymbol{\mu}
 \tag{10.69}$$

and with covariance matrix

$$\begin{aligned}
 \mathbb{V}[\mathbf{x}] &= \mathbb{V}_z[\mathbf{B}z + \boldsymbol{\mu}] + \mathbb{V}_{\epsilon}[\epsilon] = \mathbb{V}_z[\mathbf{B}z] + \sigma^2 \mathbf{I} \\
 &= \mathbf{B} \mathbb{V}_z[z] \mathbf{B}^T + \sigma^2 \mathbf{I} = \mathbf{B} \mathbf{B}^T + \sigma^2 \mathbf{I}.
 \end{aligned}
 \tag{10.70}$$

The likelihood in (10.68) can be used for maximum likelihood or MAP estimation of the model parameters.

*Remark.* We cannot use the conditional distribution in (10.64) for maximum likelihood estimation as it still depends on the latent variables. The likelihood function we require for maximum likelihood (or MAP) estimation should only be a function of the data  $\mathbf{x}$  and the model parameters, but must not depend on the latent variables.  $\diamond$

From Section 6.5 we know that a Gaussian random variable  $z$  and a linear/affine transformation  $\mathbf{x} = \mathbf{B}z$  of it are jointly Gaussian distributed. We already know the marginals  $p(z) = \mathcal{N}(z | \mathbf{0}, \mathbf{I})$  and  $p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \mathbf{B}\mathbf{B}^\top + \sigma^2\mathbf{I})$ . The missing cross-covariance is given as

$$\text{Cov}[\mathbf{x}, z] = \text{Cov}_z[\mathbf{B}z + \boldsymbol{\mu}] = \mathbf{B} \text{Cov}_z[z, z] = \mathbf{B}. \quad (10.71)$$

Therefore, the probabilistic model of PPCA, i.e., the joint distribution of latent and observed random variables is explicitly given by

$$p(\mathbf{x}, z | \mathbf{B}, \boldsymbol{\mu}, \sigma^2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x} \\ z \end{bmatrix} \middle| \begin{bmatrix} \boldsymbol{\mu} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{B}\mathbf{B}^\top + \sigma^2\mathbf{I} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{I} \end{bmatrix}\right), \quad (10.72)$$

6211 with a mean vector of length  $D + M$  and a covariance matrix of size  
6212  $(D + M) \times (D + M)$ .

### 6213 10.7.3 Posterior Distribution

The joint Gaussian distribution  $p(\mathbf{x}, z | \mathbf{B}, \boldsymbol{\mu}, \sigma^2)$  in (10.72) allows us to determine the posterior distribution  $p(z | \mathbf{x})$  immediately by applying the rules of Gaussian conditioning from Section 6.5.1. The posterior distribution of the latent variable given an observation  $\mathbf{x}$  is then

$$p(z | \mathbf{x}) = \mathcal{N}(z | \mathbf{m}, \mathbf{C}), \quad (10.73a)$$

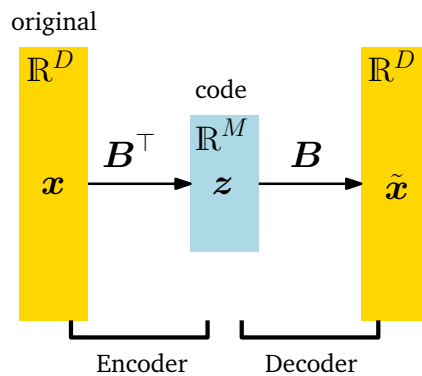
$$\mathbf{m} = \mathbf{B}^\top (\mathbf{B}\mathbf{B}^\top + \sigma^2\mathbf{I})^{-1} (\mathbf{x} - \boldsymbol{\mu}), \quad (10.73b)$$

$$\mathbf{C} = \mathbf{I} - \mathbf{B}^\top (\mathbf{B}\mathbf{B}^\top + \sigma^2\mathbf{I})^{-1} \mathbf{B}. \quad (10.73c)$$

6214 Note that the posterior covariance does not depend on the observed data  
6215  $\mathbf{x}$ . For a new observation  $\mathbf{x}_*$  in data space, we use (10.73) to determine  
6216 the posterior distribution of the corresponding latent variable  $z_*$ . The co-  
6217 variance matrix  $\mathbf{C}$  allows us to assess how confident the embedding is. A  
6218 covariance matrix  $\mathbf{C}$  with a small determinant (which measures volumes)  
6219 tells us that the latent embedding  $z_*$  is fairly certain. If we obtain a pos-  
6220 terior distribution  $p(z_* | \mathbf{x}_*)$  with much variance, we may be faced with  
6221 an outlier. However, we can explore this posterior distribution to under-  
6222 stand what other data points  $\mathbf{x}$  are plausible under this posterior. To do  
6223 this, we exploit the generative process underlying PPCA, which allows us  
6224 to explore the posterior distribution on the latent variables by generating  
6225 new data that are plausible under this posterior:

- 6226 1 Sample a latent variable  $z_* \sim p(z | \mathbf{x}_*)$  from the posterior distribution
- 6227 over the latent variables (10.73)
- 6228 2 Sample a reconstructed vector  $\tilde{\mathbf{x}}_* \sim p(\mathbf{x} | z_*, \mathbf{B}, \boldsymbol{\mu}, \sigma^2)$  from (10.64)

6229 If we repeat this process many times, we can explore the posterior dis-  
6230 tribution (10.73) on the latent variables  $z_*$  and its implications on the  
6231 observed data. The sampling process effectively hypothesizes data, which  
6232 is plausible under the posterior distribution.



**Figure 10.16** PCA can be viewed as a linear auto-encoder. It encodes the high-dimensional data  $\mathbf{x}$  into a lower-dimensional representation (code)  $\mathbf{z} \in \mathbb{R}^M$  and decodes  $\mathbf{z}$  using a decoder. The decoded vector  $\tilde{\mathbf{x}}$  is the orthogonal projection of the original data  $\mathbf{x}$  onto the  $M$ -dimensional principal subspace.

### 10.8 Further Reading

6233

6234 We derived PCA from two perspectives: a) maximizing the variance in the  
 6235 projected space; b) minimizing the average reconstruction error. However,  
 6236 PCA can also be interpreted from different perspectives. Let us re-cap what  
 6237 we have done: We took high-dimensional data  $\mathbf{x} \in \mathbb{R}^D$  and used a matrix  
 6238  $\mathbf{B}^\top$  to find a lower-dimensional representation  $\mathbf{z} \in \mathbb{R}^M$ . The columns of  
 6239  $\mathbf{B}$  are the eigenvectors of the data covariance matrix  $\mathbf{S}$  that are associated  
 6240 with the largest eigenvalues. Once we have a low-dimensional representa-  
 6241 tion  $\mathbf{z}$ , we can get a high-dimensional version of it (in the original data  
 6242 space) as  $\mathbf{x} \approx \tilde{\mathbf{x}} = \mathbf{B}\mathbf{z} = \mathbf{B}\mathbf{B}^\top \mathbf{x} \in \mathbb{R}^D$ , where  $\mathbf{B}\mathbf{B}^\top$  is a projection  
 6243 matrix.

We can also think of PCA as a linear *auto-encoder* as illustrated in Fig-  
 ure 10.16. An auto-encoder encodes the data  $\mathbf{x}_n \in \mathbb{R}^D$  to a *code*  $\mathbf{z}_n \in \mathbb{R}^M$   
 and decodes it to a  $\tilde{\mathbf{x}}_n$  similar to  $\mathbf{x}_n$ . The mapping from the data to the  
 code is called the *encoder*, and the mapping from the code back to the orig-  
 inal data space is called the *decoder*. If we consider linear mappings where  
 the code is given by  $\mathbf{z}_n = \mathbf{B}^\top \mathbf{x}_n \in \mathbb{R}^M$  and we are interested in minimiz-  
 ing the average squared error between the data  $\mathbf{x}_n$  and its reconstruction  
 $\tilde{\mathbf{x}}_n = \mathbf{B}\mathbf{z}_n$ ,  $n = 1, \dots, N$ , we obtain

$$\frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \mathbf{B}^\top \mathbf{B}\mathbf{x}_n\|^2. \quad (10.74)$$

6244 This means we end up with the same objective function as in (10.29) that  
 6245 we discussed in Section 10.3 so that we obtain the PCA solution when we  
 6246 minimize the squared auto-encoding loss. If we replace the linear map-  
 6247 ping of PCA with a nonlinear mapping, we get a nonlinear auto-encoder.  
 6248 A prominent example of this is a deep auto-encoder where the linear func-  
 6249 tions are replaced with deep neural networks. In this context, the encoder  
 6250 is also known as a *recognition network* or *inference network*, whereas the  
 6251 decoder is also called a *generator*.

auto-encoder  
code  
encoder  
decoder

6252 Another interpretation of PCA is related to information theory. We can  
 6253 think of the code as a smaller or compressed version of the original data

recognition network  
inference network  
generator

6254 point. When we reconstruct our original data using the code, we do not  
 6255 get the exact data point back, but a slightly distorted or noisy version  
 The code is a 6256 of it. This means that our compression is “lossy”. Intuitively we want  
 compressed version 6257 of the original data. to maximize the correlation between the original data and the lower-  
 6258 dimensional code. More formally, this is related to the mutual information.  
 6259 We would then get the same solution to PCA we discussed in Section 10.3  
 6260 by maximizing the mutual information, a core concept in information the-  
 6261 ory (MacKay, 2003a).

In our discussion on PPCA, we assumed that the parameters of the  
 model, i.e.,  $\mathbf{B}$ ,  $\boldsymbol{\mu}$  and the likelihood parameter  $\sigma^2$  are known. Tipping  
 and Bishop (1999) describe how to derive maximum likelihood estimates  
 for these parameters in the PPCA setting (note that we use a different  
 notation in this chapter). The maximum likelihood parameters, when project-  
 ing  $D$ -dimensional data onto an  $M$ -dimensional subspace, are given  
 by

The matrix  $\boldsymbol{\Lambda} - \sigma^2 \mathbf{I}$   
 in (10.76) is  
 guaranteed to be  
 positive  
 semi-definite as the  
 smallest eigenvalue  
 of the data  
 covariance matrix is  
 bounded from  
 below by the noise  
 variance  $\sigma^2$ .

$$\boldsymbol{\mu}_{\text{ML}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n, \quad (10.75)$$

$$\mathbf{B}_{\text{ML}} = \mathbf{T}(\boldsymbol{\Lambda} - \sigma^2 \mathbf{I})^{\frac{1}{2}} \mathbf{R}, \quad (10.76)$$

$$\sigma_{\text{ML}}^2 = \frac{1}{D - M} \sum_{j=M+1}^D \lambda_j, \quad (10.77)$$

6262 where  $\mathbf{T} \in \mathbb{R}^{D \times M}$  contains  $M$  eigenvectors of the data covariance matrix,  
 6263  $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_M) \in \mathbb{R}^{M \times M}$  is a diagonal matrix with the eigenvalues  
 6264 associated with the principal axes on its diagonal, and  $\mathbf{R} \in \mathbb{R}^{M \times M}$  is  
 6265 an arbitrary orthogonal matrix. The maximum likelihood solution  $\mathbf{B}_{\text{ML}}$  is  
 6266 unique up to an arbitrary orthogonal transformation, e.g., we can right-  
 6267 multiply  $\mathbf{B}_{\text{ML}}$  with any rotation matrix  $\mathbf{R}$  so that (10.76) essentially is a  
 6268 singular value decomposition (see Section 4.5). An outline of the proof is  
 6269 given by Tipping and Bishop (1999).

6270 The maximum likelihood estimate for  $\boldsymbol{\mu}$  given in (10.75) is the sample  
 6271 mean of the data. The maximum likelihood estimator for the observation  
 6272 noise variance  $\sigma^2$  given in (10.77) is the average variance in the orthog-  
 6273 onal complement of the principal subspace, i.e., the average leftover vari-  
 6274 ance that we cannot capture with the first  $M$  principal components are  
 6275 treated as observation noise.

In the noise-free limit where  $\sigma \rightarrow 0$ , PPCA and PCA provide identical  
 solutions: Since the data covariance matrix  $\mathbf{S}$  is symmetric, it can be di-  
 agonalized (see Section 4.4), i.e., there exists a matrix  $\mathbf{T}$  of eigenvectors  
 of  $\mathbf{S}$  so that

$$\mathbf{S} = \mathbf{T} \boldsymbol{\Lambda} \mathbf{T}^{-1}. \quad (10.78)$$

In the PPCA model, the data covariance matrix is the covariance matrix of  
 the Gaussian likelihood  $p(\mathbf{x} | \mathbf{B}, \boldsymbol{\mu}, \sigma^2)$ , which is  $\mathbf{B} \mathbf{B}^\top + \sigma^2 \mathbf{I}$ , see (10.70).

For  $\sigma \rightarrow 0$ , we obtain  $\mathbf{B}\mathbf{B}^\top$  so that this data covariance must equal the PCA data covariance (and its factorization given in (10.78)) so that

$$\text{Cov}[\mathcal{X}] = \mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1} = \mathbf{B}\mathbf{B}^\top \iff \mathbf{B} = \mathbf{T}\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{R}, \quad (10.79)$$

i.e., we obtain the maximum likelihood estimate in (10.76) for  $\sigma = 0$ .

From (10.76) and (10.78) it becomes clear that (P)PCA performs a decomposition of the data covariance matrix.

In a streaming setting, where data arrives sequentially, it is recommended to use the iterative expectation maximization (EM) algorithm for maximum likelihood estimation (Roweis, 1998).

To determine the dimensionality of the latent variables (the length of the code, the dimensionality of the lower-dimensional subspace onto which we project the data) Gavish and Donoho (2014) suggest the heuristic that, if we can estimate the noise variance  $\sigma^2$  of the data, we should discard all singular values smaller than  $\frac{4\sigma\sqrt{D}}{\sqrt{3}}$ . Alternatively, we can use (nested) cross validation (Section 8.5.1) or Bayesian model selection criteria (discussed in Section 8.5.2) to determine a good estimate of the intrinsic dimensionality of the data (Minka, 2001b).

Similar to our discussion on linear regression in Chapter 9, we can place a prior distribution on the parameters of the model and integrate them out. By doing so, we a) avoid point estimates of the parameters and the issues that come with these point estimates (see Section 8.5) and b) allow for an automatic selection of the appropriate dimensionality  $M$  of the latent space. In this *Bayesian PCA*, which was proposed by Bishop (1999), a prior  $p(\boldsymbol{\mu}, \mathbf{B}, \sigma^2)$  is placed on the model parameters. The generative process allows us to integrate the model parameters out instead of conditioning on them, which addresses overfitting issues. Since this integration is analytically intractable, Bishop (1999) proposes to use approximate inference methods, such as MCMC or variational inference. We refer to the work by Gilks et al. (1996) and Blei et al. (2017) for more details on these approximate inference techniques.

Bayesian PCA

In PPCA, we considered the linear model  $p(\mathbf{x}_n | \mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n | \mathbf{B}\mathbf{z}_n + \boldsymbol{\mu}, \sigma^2\mathbf{I})$  with prior  $p(\mathbf{z}_n) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ , where all observation dimensions are affected by the same amount of noise. If we allow each observation dimension  $d$  to have a different variance  $\sigma_d^2$  we obtain *factor analysis* (FA) (Spearman, 1904; Bartholomew et al., 2011). This means, FA gives the likelihood some more flexibility than PPCA, but still forces the data to be explained by the model parameters  $\mathbf{B}$ ,  $\boldsymbol{\mu}$ . However, FA no longer allows for a closed-form maximum likelihood solution so that we need to use an iterative scheme, such as the Expectation Maximization algorithm, to estimate the model parameters. While in PPCA all stationary points are global optima, this no longer holds for FA. Compared to PPCA, FA does not change if we scale the data, but it does return different solutions if we rotate the data.

factor analysis

An overly flexible likelihood would be able to explain more than just the noise.

Independent Component Analysis 6316 An algorithm that is also closely related to PCA is *Independent Component Analysis* (ICA, (Hyvarinen et al., 2001)) Starting again with the latent-variable perspective  $p(\mathbf{x}_n | \mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n | \mathbf{B}\mathbf{z}_n + \boldsymbol{\mu}, \sigma^2 \mathbf{I})$  we now change the prior on  $\mathbf{z}_n$  to non-Gaussian distributions. ICA can be used for *blind-source separation*. Imagine you are in a busy train station with many people talking. Your ears play the role of microphones, and they linearly mix different speech signals in the train station. The goal of blind-source separation is to identify the constituent parts of the mixed signals. As discussed above in the context of maximum likelihood estimation for PPCA, the original PCA solution is invariant to any rotation. Therefore, PCA can identify the best lower-dimensional subspace in which the signals live, but not the signals themselves (Murphy, 2012). ICA addresses this issue by modifying the prior distribution  $p(\mathbf{z})$  on the latent sources to require non-Gaussian priors  $p(\mathbf{z})$ . We refer to the books by Hyvarinen et al. (2001) and Murphy (2012) for more details on ICA.

6317  
6318  
6319  
6320  
6321  
6322  
6323  
6324  
6325  
6326  
6327  
6328  
6329  
6330  
6331  
6332  
6333  
6334  
6335  
6336  
6337  
6338  
6339  
6340  
6341  
6342  
6343  
6344  
6345  
6346  
6347  
6348  
6349  
6350  
6351  
6352  
6353  
6354  
6355  
6356  
6357  
6358  
6359

blind-source separation

kernel trick  
kernel PCA

deep auto-encoder

Gaussian Process Latent Variable Model

Bayesian GP-LVM

PCA, factor analysis and ICA are three examples for dimensionality reduction with linear models. Cunningham and Ghahramani (2015) provide a broader survey of linear dimensionality reduction.

The (P)PCA model we discussed here allows for several important extensions. In Section 10.5, we explained how to do PCA when the input dimensionality  $D$  is significantly greater than the number  $N$  of data points. By exploiting the insight that PCA can be performed by computing (many) inner products, this idea can be pushed to the extreme by considering infinite-dimensional features. The *kernel trick* is the basis of *kernel PCA* and allows us to implicitly compute inner products between infinite-dimensional features (Schölkopf et al., 1998; Schölkopf and Smola, 2002).

There are nonlinear dimensionality reduction techniques that are derived from PCA (Burges (2010) provide a good overview). The auto-encoder perspective of PCA that we discussed above can be used to render PCA as a special case of a *deep auto-encoder*. In the deep auto-encoder, both the encoder and the decoder are represented by multi-layer feedforward neural networks, which themselves are nonlinear mappings. If we set the activation functions in these neural networks to be the identity, the model becomes equivalent to PCA. A different approach to nonlinear dimensionality reduction is the *Gaussian Process Latent Variable Model* (GP-LVM) proposed by Lawrence (2005). The GP-LVM starts off with the latent-variable perspective that we used to derive PPCA and replaces the linear relationship between the latent variables  $\mathbf{z}$  and the observations  $\mathbf{x}$  with a Gaussian process (GP). Instead of estimating the parameters of the mapping (as we do in PPCA), the GP-LVM marginalizes out the model parameters and makes point estimates of the latent variables  $\mathbf{z}$ . Similar to Bayesian PCA, the *Bayesian GP-LVM* proposed by Titsias and Lawrence (2010) maintains a distribution on the latent variables  $\mathbf{z}$  and uses approximate inference to integrate them out as well.