10

Dimensionality Reduction with Principal Component Analysis

Data in real life is often high dimensional. For example, if we want to estimate the price of our house in a year's time, we can use data that helps us to do this: the type of house, the size, the number of bedrooms and bathrooms, the value of houses in the neighborhood when they were bought, the distance to the next train station and park, the number of crimes committed in the neighborhood, the economic climate etc. – there are many things that influence the house price, and we collect this information in a data set that we can use to estimate the house price. Another example is a 640×480 pixels color image, which 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 comes with some difficulties: It is hard to analyze, interpretation is difficult, visualization is nearly impossible, and (from a practical point of view) storage can be expensive. However, high-dimensional data also has some nice properties: For example, high-dimensional data is often overcomplete, i.e., many dimensions are redundant and can be explained by a combination of other dimensions. 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 jpg or mp3, which are compression algorithms for images and music.

principal component analysis

dimensionality reduction

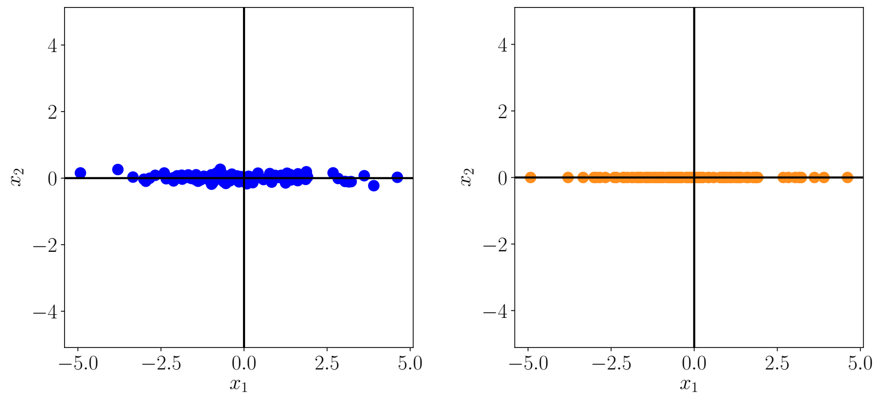
In this chapter, we will discuss *principal component analysis* (PCA), an algorithm for linear *dimensionality reduction*. PCA, proposed by Pearson (1901) and Hotelling (1933), has been around for more than 100 years and is still one of the most commonly used techniques for data compression, data visualization and the identification of simple patterns, latent factors and structures of high-dimensional data. In the signal processing community, PCA is also known as the *Karhunen-Loève transform*. In this chapter, we will explore the concept of linear dimensionality reduction with PCA in more detail, drawing on our understanding of basis and basis change (see Sections 2.6.1 and 2.7.2), projections (see Section 3.6), eigenvalues (see Section 4.2), Gaussian distributions (see Section 6.6) and constrained optimization (see Section 7.2).

Karhunen-Loève transform

Dimensionality reduction generally exploits the property of high-dimen-

Draft chapter (July 10, 2018) from "Mathematics for Machine Learning" ©2018 by Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. To be published by Cambridge University Press. Report errata and feedback to http://mml-book.com. Please do not post or distribute this file, please link to https://mml-book.com.

ure 10.1
stration:
nensionality
uction. (a) The
ginal dataset not
y much along the
direction. (b)
e data from (a)
be represented
ng the
coordinate alone
h nearly no loss.



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

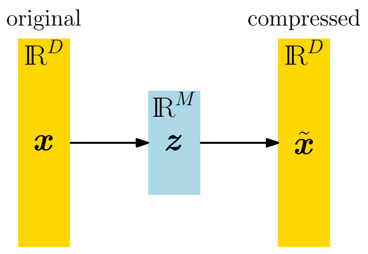


Figure 10.2 Graphical illustration of PCA. In PCA, we find a compressed version $\tilde{\boldsymbol{x}}$ of original data $\boldsymbol{x}$ that has an intrinsic lower-dimensional representation $\boldsymbol{z}$.

sional data (e.g., images) that it often lies on a low-dimensional subspace, and that many dimensions are highly correlated, redundant or contain little information. Figure 10.1 gives an illustrative example in two dimensions. Although the data in Figure 10.1(a) does not quite lie on a line, the data does not vary much in the x_2-direction, so that we can express it as if it was on a line – with nearly no loss, see Figure 10.1(b). The data in Figure 10.1(b) requires only the x_1-coordinated to describe and lies in a one-dimensional subspace of $\mathbb{R}^2$.

10.1 Problem Setting

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

Figure 10.2 illustrates the setting we consider in PCA, where $\boldsymbol{z}$ represents the intrinsic lower dimension of the compressed data $\tilde{\boldsymbol{x}}$ and plays the role of a bottleneck, which controls how much information can flow between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$.

More concretely, we consider i.i.d. data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^D$, and

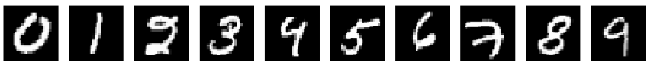


Figure 10.3 Examples of handwritten digits from the MNIST dataset.

we search a low-dimensional, compressed representation (code) $\boldsymbol{z}_n$ of $\boldsymbol{x}_n$. If our observed data lives in $\mathbb{R}^D$, we look for an M-dimensional subspace $U \subseteq \mathbb{R}^D$, $\dim(U) = M < D$ onto which we project data. We denote the projected data as $\tilde{\boldsymbol{x}}_n \in U$, and their coordinates (with respect to an appropriate basis in U) with $\boldsymbol{z}_n$. Our aim is to find $\tilde{\boldsymbol{x}}_n$ so that they are as similar to the original data $\boldsymbol{x}_n$ as possible.

Example 10.1 (Coordinate Representation/Code)
Consider $\mathbb{R}^2$ with the canonical basis $\boldsymbol{e}_1 = [1, 0]^\top$, $\boldsymbol{e}_2 = [0, 1]^\top$. From Chapter 2 we know that $\boldsymbol{x} \in \mathbb{R}^2$ can be represented as a linear combination of these basis vectors, e.g.,

$$\begin{bmatrix} 5 \\ 3 \end{bmatrix} = 5\boldsymbol{e}_1 + 3\boldsymbol{e}_2 \,. \tag{10.1}$$

However, when we consider the set of vectors

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} 0 \\ z \end{bmatrix} \in \mathbb{R}^2 \,, \quad z \in \mathbb{R} \,, \tag{10.2}$$

they can always be written as $0\boldsymbol{e}_1 + z\boldsymbol{e}_2$. To represent these vectors it is sufficient to remember/store the *coordinate/code* z of the $\boldsymbol{e}_2$ vector.

More precisely, the set of $\tilde{\boldsymbol{x}}$ vectors (with the standard vector addition and scalar multiplication) forms a vector subspace U (see Section 2.4) with $\dim(U) = 1$ because $U = \mathrm{span}[\boldsymbol{e}_2]$.

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

In PCA, we consider the relationship between the original data $\boldsymbol{x}$ and its low-dimensional code $\boldsymbol{z}$ to be linear so that $\boldsymbol{z} = \boldsymbol{B}^\top \boldsymbol{x}$ for a suitable matrix $\boldsymbol{B}$.

Throughout this chapter, we will use the MNIST digits dataset as a reoccurring example, which contains $60,000$ examples of handwritten digits 0–9. Each digit is an image of size 28×28, i.e., it contains 784 pixels so that we can interpret every image in this dataset as a vector $\boldsymbol{x} \in \mathbb{R}^{784}$. Examples of these digits are shown in Figure 10.3.

`http://yann.lecun.com/exdb/mnist/`

In the following, we will derive PCA from two different perspectives. First, we derive PCA by maintaining as much variance as possible in the projected space. Second, we will derive PCA by minimizing the average squared reconstruction error, which directly links to many concepts in Chapters 3 and 4.

10.2 Maximum Variance Perspective

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

If we interpret information content in the data as how "space filling" the data set is, then we can describe the information contained in the data by looking at the spread of the data. From Chapter 6 we know that the variance is an indicator of the spread of the data, and it is possible to formulate PCA as a dimensionality reduction algorithm that maximizes the variance in the low-dimensional representation of the data to retain as much information as possible. Now, let us formulate this objective more concretely.

Consider a dataset $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, $\boldsymbol{x}_n \in \mathbb{R}^D$, with mean $\mathbf{0}$ that possesses the *data covariance matrix* (empirical covariance)

data covariance matrix

$$\boldsymbol{S} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top . \tag{10.3}$$

Furthermore, we assume a low-dimensional representation $\boldsymbol{z}_n = \boldsymbol{B}^\top \boldsymbol{x}_n \in \mathbb{R}^M$ of $\boldsymbol{x}_n$, where $\boldsymbol{B} \in \mathbb{R}^{D \times M}$.

Our aim is to find a matrix $\boldsymbol{B}$ that retains as much information as possible when compressing data. We assume that $\boldsymbol{B}$ is an orthogonal matrix so that $\boldsymbol{b}_i^\top \boldsymbol{b}_j = 0$ if and only if $i \neq j$. Retaining most information is formulated as capturing the largest amount of variance in the low-dimensional code (Hotelling, 1933).

The columns $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M$ of $\boldsymbol{B}$ form a basis of the M-dimensional subspace in which the projected data $\tilde{\boldsymbol{x}} = \boldsymbol{B}\boldsymbol{B}^\top \boldsymbol{x} \in \mathbb{R}^D$ live.

Remark. (Centered Data) Let us assume that $\boldsymbol{\mu} = \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}]$ is the (empirical) mean of the data. Using the properties of the variance, which we discussed in Section 6.4.4 we obtain

$$\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{z}] = \mathbb{V}_{\boldsymbol{x}}[\boldsymbol{B}^\top(\boldsymbol{x} - \boldsymbol{\mu})] = \mathbb{V}_{\boldsymbol{x}}[\boldsymbol{B}^\top \boldsymbol{x} - \boldsymbol{B}^\top \boldsymbol{\mu}] = \mathbb{V}_{\boldsymbol{x}}[\boldsymbol{B}^\top \boldsymbol{x}] , \tag{10.4}$$

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

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

$$V_1 := \mathbb{V}[z_1] = \frac{1}{N}\sum_{n=1}^{N} z_{1n}^2 \tag{10.5}$$

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 $\boldsymbol{z}_n \in \mathbb{R}^M$ of $\boldsymbol{x}_n \in \mathbb{R}^D$. Note that first component of $\boldsymbol{z}_n$ is given by

$$z_{1n} = \boldsymbol{b}_1^\top \boldsymbol{x}_n \,. \tag{10.6}$$

We use this relationship now in (10.5), which yields

$$V_1 = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{b}_1^\top \boldsymbol{x}_n)^2 = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{b}_1^\top \boldsymbol{x}_n \boldsymbol{x}_n^\top \boldsymbol{b}_1 \tag{10.7a}$$

$$= \boldsymbol{b}_1^\top \left(\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top\right) \boldsymbol{b}_1 = \boldsymbol{b}_1^\top \boldsymbol{S} \boldsymbol{b}_1 \,, \tag{10.7b}$$

where $\boldsymbol{S}$ is the data covariance matrix defined in (10.3).

It is clear that arbitrarily increasing the magnitude of the vector $\boldsymbol{b}_1$ increases V_1. Therefore, we restrict all solutions to $\|\boldsymbol{b}_1\| = 1$, 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 we end up with the constrained optimization problem

$$\max_{\boldsymbol{b}_1} \boldsymbol{b}_1^\top \boldsymbol{S} \boldsymbol{b}_1 \tag{10.8}$$

$$\text{subject to } \|\boldsymbol{b}_1\|^2 = 1 \,. \tag{10.9}$$

Following Section 7.2, we obtain the Lagrangian

$$\mathfrak{L} = V_1 + \lambda_1(1 - \boldsymbol{b}_1^\top \boldsymbol{b}_1) = \boldsymbol{b}_1^\top \boldsymbol{S} \boldsymbol{b}_1 + \lambda_1(1 - \boldsymbol{b}_1^\top \boldsymbol{b}_1) \tag{10.10}$$

to solve this constrained optimization problem. The partial derivatives of $\mathfrak{L}$ with respect to $\boldsymbol{b}_1$ and λ_1 are

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{b}_1} = 2\boldsymbol{b}_1^\top \boldsymbol{S} - 2\lambda_1 \boldsymbol{b}_1^\top \tag{10.11}$$

$$\frac{\partial \mathfrak{L}}{\partial \lambda_1} = 1 - \boldsymbol{b}_1^\top \boldsymbol{b}_1 \,, \tag{10.12}$$

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

$$\boldsymbol{b}_1^\top \boldsymbol{b}_1 = 1 \,, \tag{10.13}$$

$$\boldsymbol{S}\boldsymbol{b}_1 = \lambda_1 \boldsymbol{b}_1 \,, \tag{10.14}$$

i.e., we see that $\boldsymbol{b}_1$ is an eigenvector of the data covariance matrix $\boldsymbol{S}$, and the Lagrange multiplier λ_1 plays the role of the corresponding eigenvalue. This eigenvector property allows us to rewrite our variance objective as

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

i.e., the variance of the data projected onto a one-dimensional subspace equals the eigenvalue that is associated with the basis vector $\boldsymbol{b}_1$ that spans this subspace. Therefore, to maximize the variance of the low-dimensional code we choose the basis vector belonging to 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 $\boldsymbol{b}_1$ in the original data space by mapping the coordinate z_{1n} back into data space, which gives us the projected data point

principal component

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

in the original data space.

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

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

Generally, the mth principal component can be found by subtracting the effect of the first $m-1$ principal components 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

$$\hat{\boldsymbol{X}} := \boldsymbol{X} - \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^\top \boldsymbol{X}\,, \tag{10.17}$$

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

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

To find the mth principal component, we maximize the variance

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

subject to $\|\boldsymbol{b}_m\|^2 = 1$, where we followed the same steps as in (10.7b) and defined $\hat{\boldsymbol{S}}$ as the data covariance matrix of $\hat{\boldsymbol{X}}$. As previously, when we looked at the first principal component alone, we solve a constrained

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

optimization problem and discover that the optimal solution $\boldsymbol{b}_m$ is the eigenvector of $\hat{\boldsymbol{S}}$ that belongs to the largest eigenvalue of $\hat{\boldsymbol{S}}$.

However, it also turns out that $\boldsymbol{b}_m$ is an eigenvector of $\boldsymbol{S}$. Since

$$\hat{\boldsymbol{S}} = \frac{1}{N}\sum_{n=1}^{N} \hat{\boldsymbol{x}}_n \hat{\boldsymbol{x}}_n^\top \overset{(10.17)}{=} \frac{1}{N}\sum_{n=1}^{N}\Big(\boldsymbol{x}_n - \sum_{i=1}^{m-1}\boldsymbol{b}_i\boldsymbol{b}_i^\top\boldsymbol{x}_n\Big)\Big(\boldsymbol{x}_n - \sum_{i=1}^{m-1}\boldsymbol{b}_i\boldsymbol{b}_i^\top\boldsymbol{x}_n\Big)^\top \tag{10.19a}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{x}_n\boldsymbol{x}_n^\top - 2\boldsymbol{x}_n\boldsymbol{x}_n^\top\sum_{i=1}^{m-1}\boldsymbol{b}_i\boldsymbol{b}_i^\top + \boldsymbol{x}_n^\top\sum_{i=1}^{m-1}\boldsymbol{b}_i\boldsymbol{b}_i^\top\sum_{i=1}^{m-1}\boldsymbol{b}_i\boldsymbol{b}_i^\top\right) \tag{10.19b}$$

we can multiply $\boldsymbol{b}_m$ onto $\hat{\boldsymbol{S}}$ and obtain

$$\hat{\boldsymbol{S}}\boldsymbol{b}_m = \frac{1}{N}\sum_{n=1}^{N}\hat{\boldsymbol{x}}_n\hat{\boldsymbol{x}}_n^\top\boldsymbol{b}_m = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^\top\boldsymbol{b}_m = \boldsymbol{S}\boldsymbol{b}_m = \lambda_m\boldsymbol{b}_m\,. \tag{10.20}$$

Here we applied the orthogonality conditions $\boldsymbol{b}_i^\top\boldsymbol{b}_m = 0$ for all $i = 1,\dots,m-1$ (all terms involving sums up to $m-1$ vanish). In the end, we exploited the fact that $\boldsymbol{b}_m$ is an eigenvector of $\hat{\boldsymbol{S}}$. Therefore, $\boldsymbol{b}_m$ is also an eigenvector of the original data covariance matrix $\boldsymbol{S}$, and the corresponding eigenvalue is λ_m is the mth largest eigenvalue of $\boldsymbol{S}$. Moreover, the variance of the data projected onto the mth principal component

$$V_m = \boldsymbol{b}_m^\top\boldsymbol{S}\boldsymbol{b}_m \overset{(10.20)}{=} \lambda_m\boldsymbol{b}_m^\top\boldsymbol{b}_m = \lambda_m \tag{10.21}$$

since $\boldsymbol{b}_m^\top\boldsymbol{b}_m = 1$. This means that the variance of the data, when projected onto an M-dimensional subspace, equals the sum of the eigenvalues that belong to the corresponding eigenvectors of the data covariance matrix.

To maximize the variance of the projected data, we choose the columns of $\boldsymbol{B}$ to be the eigenvectors that belong to the M largest eigenvalues of the data covariance matrix.

In practice, we do not have to compute principal components sequentially, but we can compute all of them at the same time. If we are looking for a projection onto an M-dimensional subspace so that as much variance as possible is retained in the projection, then PCA tells us to choose the columns of $\boldsymbol{B}$ to be the eigenvectors that belong to the M largest eigenvalues of the data covariance matrix. The maximum amount of variance PCA can capture with the first M principal components is

$$V = \sum_{m=1}^{M}\lambda_m\,, \tag{10.22}$$

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

$$J = \sum_{j=M+1}^{D}\lambda_j\,. \tag{10.23}$$

To summarize, to determine the M-dimensional subspace for which an

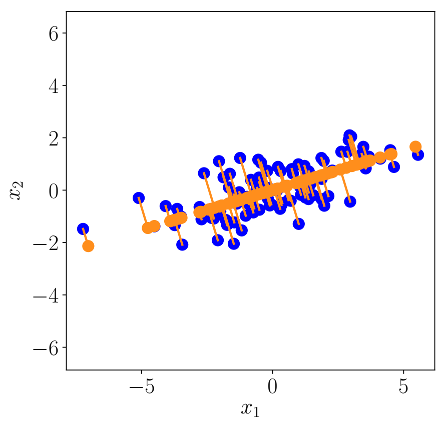


Figure 10.4 Illustration of the projection approach to PCA. We aim to find a one-dimensional subspace (line) of $\mathbb{R}^2$ so that the distance vector between projected (orange) and original (blue) data is as small as possible.

orthogonal projection maximizes the variance of the data we need to compute the M eigenvectors that belong to the M largest eigenvalues of the data covariance matrix. In Section 10.4, we will return to this point and discuss how to efficiently compute these M eigenvectors.

10.3 Projection Perspective

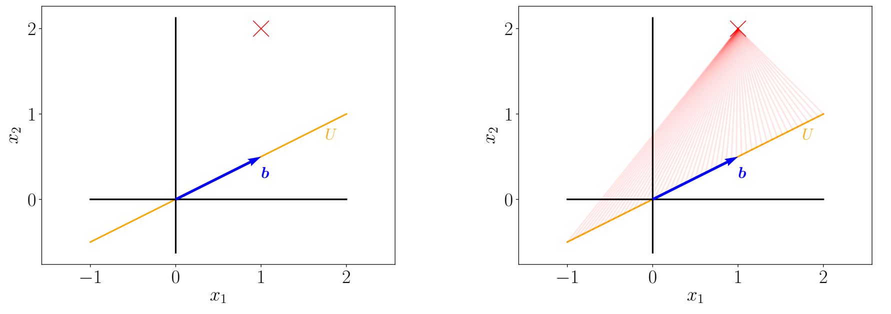
In the following, we will derive PCA as an algorithm for linear dimensionality reduction that minimizes the average projection error. We will draw heavily from Chapters 2 and 3. In the previous section, we derived PCA by maximizing the variance in the projected space to retain as much information as possible. In the following, we will look at the difference vectors between the original data $\boldsymbol{x}_n$ and their reconstruction $\tilde{\boldsymbol{x}}_n$ and minimize this distance so that $\boldsymbol{x}_n$ and $\tilde{\boldsymbol{x}}_n$ are as close as possible. Figure 10.4 illustrates this setting.

10.3.1 Setting and Objective

Assume an (ordered) orthonormal basis (ONB) $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_D)$ of $\mathbb{R}^D$, i.e., $\boldsymbol{b}_i^\top \boldsymbol{b}_j = 1$ if and only if $i = j$ and 0 otherwise. From Section 2.5 we know that every $\boldsymbol{x} \in \mathbb{R}^D$ can be written as a linear combination of the basis vectors of $\mathbb{R}^D$, i.e.,

$$\boldsymbol{x} = \sum_{d=1}^{D} z_d \boldsymbol{b}_d \tag{10.24}$$

for $z_d \in \mathbb{R}$. We are interested in finding vectors $\tilde{\boldsymbol{x}} \in \mathbb{R}^D$, which live in lower-dimensional subspace U of $\mathbb{R}^D$, so that $\tilde{\boldsymbol{x}}$ is as similar to $\boldsymbol{x}$ as possible. As $\tilde{\boldsymbol{x}} \in U \subseteq \mathbb{R}^D$, we can also express $\tilde{\boldsymbol{x}}$ as a linear combination



(a) Setting.

(b) Differences $\boldsymbol{x} - \tilde{\boldsymbol{x}}$ for 50 candidates $\tilde{\boldsymbol{x}}$ are shown by the red lines.

Figure 10.5 Simplified projection setting. (a) A vector $\boldsymbol{x} \in \mathbb{R}^2$ (red cross) shall be projected onto a one-dimensional subspace $U \subseteq \mathbb{R}^2$ spanned by $\boldsymbol{b}$. (b) shows the difference vectors between $\boldsymbol{x}$ and some candidates $\tilde{\boldsymbol{x}}$.

of the basis vectors of $\mathbb{R}^D$ so that

$$\tilde{\boldsymbol{x}} = \sum_{d=1}^{D} z_d \boldsymbol{b}_d \,. \tag{10.25}$$

For example, vectors $\tilde{x} \in U$ could be vectors on a plane in $\mathbb{R}^3$. The dimensionality of the plane is 2, but the vectors still have three coordinates in $\mathbb{R}^3$.

Let us assume $\dim(U) = M$ where $M < D = \dim(\mathbb{R}^D)$. Then, we can find basis vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_D$ of $\mathbb{R}^D$ so that at least $D - M$ of the coefficients z_d are equal to 0, and we can rearrange the way we index the basis vectors $\boldsymbol{b}_d$ such that the coefficients that are zero appear at the end. This allows us to express $\tilde{\boldsymbol{x}}$ as

$$\tilde{\boldsymbol{x}} = \sum_{m=1}^{M} z_m \boldsymbol{b}_m + \sum_{j=M+1}^{D} 0 \boldsymbol{b}_j = \sum_{m=1}^{M} z_m \boldsymbol{b}_m = \boldsymbol{B}\boldsymbol{z} \in \mathbb{R}^D \,, \tag{10.26}$$

where we defined

$$\boldsymbol{B} := [\boldsymbol{b}_1, \dots, \boldsymbol{b}_M] \in \mathbb{R}^{D \times M} \,, \tag{10.27}$$
$$\boldsymbol{z} := [z_1, \dots, z_M]^\top \in \mathbb{R}^M \,. \tag{10.28}$$

In the following, we use exactly this kind of representation of $\tilde{\boldsymbol{x}}$ to find optimal coordinates $\boldsymbol{z}$ and basis vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_M$ such that $\tilde{\boldsymbol{x}}$ is as similar to the original data point $\boldsymbol{x}$, i.e., we aim to minimize the (Euclidean) distance $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$. Figure 10.5 illustrates this setting.

Without loss of generality, we assume that the dataset $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, $\boldsymbol{x}_n \in \mathbb{R}^D$, is centered at $\boldsymbol{0}$, i.e., $\mathbb{E}[\boldsymbol{X}] = \boldsymbol{0}$.

Remark. Without the zero-mean assumption, we would arrive at exactly the same solution but the notation would be substantially more cluttered. ◇

principal subspace

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

$$\tilde{\boldsymbol{x}}_n := \sum_{m=1}^{M} z_{mn}\boldsymbol{b}_m = \boldsymbol{B}\boldsymbol{z}_n \in \mathbb{R}^D\,, \tag{10.29}$$

where $\boldsymbol{B} \in \mathbb{R}^{D\times M}$ is given in (10.27) and

$$\boldsymbol{z}_n := [z_{1n}, \dots, z_{Mn}]^\top \in \mathbb{R}^M\,, \quad n = 1, \dots N\,, \tag{10.30}$$

is the coordinate vector of $\tilde{\boldsymbol{x}}_n$ with respect to the basis $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_M)$. More specifically, we are interested in having the $\tilde{\boldsymbol{x}}_n$ as similar to $\boldsymbol{x}_n$ as possible. There are many ways to measure similarity.

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

reconstruction error

$$J := \frac{1}{N}\sum_{n=1}^{N} \|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2\,. \tag{10.31}$$

In order to find this optimal linear projection, we need to find the orthonormal basis of the principal subspace and the coordinates $\boldsymbol{z}_n$ of the projections with respect to these basis vectors. All these parameters enter our objective (10.31) through $\tilde{\boldsymbol{x}}_n$.

In order to find the coordinates $\boldsymbol{z}_n$ and the ONB of the principal subspace we optimize J by computing the partial derivatives of J with respect to all parameters of interest (i.e., the coordinates and the basis vectors), setting them to $\boldsymbol{0}$, and solving for the parameters. We detail these steps next. We will first determine the optimal coordinates z_{in} and then the basis vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_M$ of the principal subspace, i.e., the subspace in which $\tilde{\boldsymbol{x}}$ lives.

10.3.2 Optimization

Since the parameters we are interested in, i.e., the basis vectors $\boldsymbol{b}_i$ and the coordinates z_{in} of the projection with respect to the basis of the principal subspace, only enter the objective J through $\tilde{\boldsymbol{x}}_n$, we obtain

$$\frac{\partial J}{\partial z_{in}} = \frac{\partial J}{\partial \tilde{\boldsymbol{x}}_n}\frac{\partial \tilde{\boldsymbol{x}}_n}{\partial z_{in}}\,, \tag{10.32}$$

$$\frac{\partial J}{\partial \boldsymbol{b}_i} = \frac{\partial J}{\partial \tilde{\boldsymbol{x}}_n}\frac{\partial \tilde{\boldsymbol{x}}_n}{\partial \boldsymbol{b}_i} \tag{10.33}$$

for $i = 1, \dots, M$ and $n = 1, \dots, N$, where

$$\frac{\partial J}{\partial \tilde{\boldsymbol{x}}_n} = -\frac{2}{N}(\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n)^\top \in \mathbb{R}^{1\times D}\,. \tag{10.34}$$

In the following, we determine the optimal coordinates z_{in} first before finding the ONB of the principal subspace.

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

Coordinates

Let us start by finding the coordinates $z_{1n}, \dots, z_{Mn}$ of the projections $\tilde{\boldsymbol{x}}_n$ for $n = 1, \dots, N$. We assume that $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_D)$ is an ordered ONB of $\mathbb{R}^D$. From (10.32) we require the partial derivative

$$\frac{\partial \tilde{\boldsymbol{x}}_n}{\partial z_{in}} \overset{(10.29)}{=} \frac{\partial}{\partial z_{in}} \left(\sum_{m=1}^{M} z_{mn} \boldsymbol{b}_m \right) = \boldsymbol{b}_i \tag{10.35}$$

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

$$\frac{\partial J}{\partial z_{in}} \overset{\substack{(10.34)\\(10.35)}}{=} -\frac{2}{N} (\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n)^\top \boldsymbol{b}_i \overset{(10.29)}{=} -\frac{2}{N} \left(\boldsymbol{x}_n - \sum_{m=1}^{M} z_{mn} \boldsymbol{b}_m \right)^\top \boldsymbol{b}_i \tag{10.36}$$

$$\overset{\text{ONB}}{=} -\frac{2}{N} (\boldsymbol{x}_n^\top \boldsymbol{b}_i - z_{in} \underbrace{\boldsymbol{b}_i^\top \boldsymbol{b}_i}_{=1}) = -\frac{2}{N} (\boldsymbol{x}_n^\top \boldsymbol{b}_i - z_{in}) \,. \tag{10.37}$$

Setting this partial derivative to 0 yields immediately the optimal coordinates

$$z_{in} = \boldsymbol{x}_n^\top \boldsymbol{b}_i = \boldsymbol{b}_i^\top \boldsymbol{x}_n \tag{10.38}$$

for $i = 1, \dots, M$ and $n = 1, \dots, N$. This means, the optimal coordinates z_{in} of the projection $\tilde{\boldsymbol{x}}_n$ are the coordinates of the orthogonal projection (see Section 3.6) of the original data point $\boldsymbol{x}_n$ onto the one-dimensional subspace that is spanned by $\boldsymbol{b}_i$. Consequently:

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

- The optimal projection $\tilde{\boldsymbol{x}}_n$ of $\boldsymbol{x}_n$ is an orthogonal projection.
- The coordinates of $\tilde{\boldsymbol{x}}_n$ with respect to the basis $\boldsymbol{b}_1, \dots, \boldsymbol{b}_M$ are the coordinates of the orthogonal projection of $\boldsymbol{x}_n$ onto the principal subspace.
- An orthogonal projection is the best linear mapping we can find given the objective (10.31).

Remark (Orthogonal Projections with Orthonormal Basis Vectors). Let us briefly recap orthogonal projections from Section 3.6. If $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_D)$ is an orthonormal basis of $\mathbb{R}^D$ then

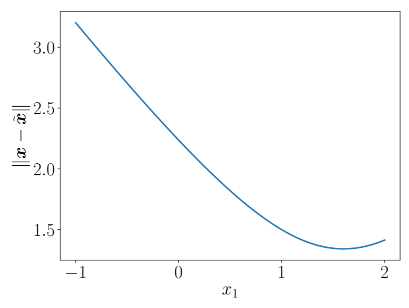
$$\tilde{\boldsymbol{x}} = \boldsymbol{b}_j (\underbrace{\boldsymbol{b}_j^\top \boldsymbol{b}_j}_{=1})^{-1} \boldsymbol{b}_j^\top \boldsymbol{x} = \boldsymbol{b}_j \boldsymbol{b}_j^\top \boldsymbol{x} \in \mathbb{R}^D \tag{10.39}$$

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

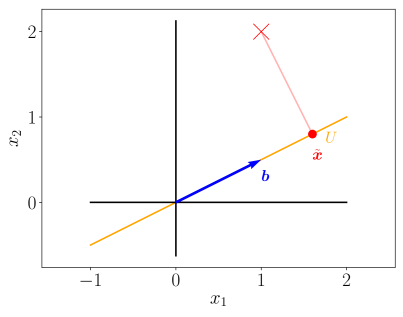
is the orthogonal projection of $\boldsymbol{x}$ onto the subspace spanned by the jth basis vector, and $z_j = \boldsymbol{b}_j^\top \boldsymbol{x}$ is the coordinate of this projection with respect to the basis vector $\boldsymbol{b}_j$ that spans that subspace since $z_j \boldsymbol{b}_j = \tilde{\boldsymbol{x}}$. Figure 10.6 illustrates this setting.

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

$$\tilde{\boldsymbol{x}} = \boldsymbol{B}(\underbrace{\boldsymbol{B}^\top \boldsymbol{B}}_{=\boldsymbol{I}})^{-1} \boldsymbol{B}^\top \boldsymbol{x} = \boldsymbol{B}\boldsymbol{B}^\top \boldsymbol{x} \,, \tag{10.40}$$



(a) Distances $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$ for some $\tilde{\boldsymbol{x}} \in U$, see panel (b) for the setting.



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

Figure 10.6 Optimal projection of a vector $\boldsymbol{x} \in \mathbb{R}^2$ onto a one-dimensional subspace (continuation from Figure 10.5). (a) Distances $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$ for some $\tilde{\boldsymbol{x}} \in U$. (b) Orthogonal projection and optimal coordinates.

where we defined $\boldsymbol{B} := [\boldsymbol{b}_1, \dots, \boldsymbol{b}_M] \in \mathbb{R}^{D \times M}$. The coordinates of this projection with respect to the ordered basis $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_M)$ are $\boldsymbol{z} := \boldsymbol{B}^\top \boldsymbol{x}$ as discussed in Section 3.6.

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

Basis of the Principal Subspace

We already determined the optimal coordinates of the projected data for a given ONB $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_D)$ of $\mathbb{R}^D$, only M of which were non-zero. What remains is to determine the basis vectors that span the principal subspace.

Before we get started, let us briefly introduce the concept of an orthogonal complement.

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, every vector $\boldsymbol{x} \in V$ can be (uniquely) decomposed into

orthogonal complement

$$\boldsymbol{x} = \sum_{m=1}^{M} \lambda_m \boldsymbol{b}_m + \sum_{j=1}^{D-M} \psi_j \boldsymbol{b}_j^\perp, \quad \lambda_i, \psi_j \in \mathbb{R}, \tag{10.41}$$

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

To determine the basis vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_M$ of the principal subspace, we rephrase the loss function (10.31) 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

$$\tilde{\boldsymbol{x}}_n = \sum_{m=1}^{M} z_{mn}\boldsymbol{b}_m \overset{(10.38)}{=} \sum_{m=1}^{M}(\boldsymbol{x}_n^\top \boldsymbol{b}_m)\boldsymbol{b}_m\,. \tag{10.42}$$

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

$$\tilde{\boldsymbol{x}}_n = \left(\sum_{m=1}^{M} \boldsymbol{b}_m\boldsymbol{b}_m^\top\right)\boldsymbol{x}_n\,. \tag{10.43}$$

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

$$\begin{aligned}\boldsymbol{x}_n &= \sum_{d=1}^{D} z_{dn}\boldsymbol{b}_d \overset{(10.38)}{=} \sum_{d=1}^{D}(\boldsymbol{x}_n^\top \boldsymbol{b}_d)\boldsymbol{b}_d = \left(\sum_{d=1}^{D}\boldsymbol{b}_d\boldsymbol{b}_d^\top\right)\boldsymbol{x}_n && \text{(10.44a)}\\ &= \left(\sum_{m=1}^{M}\boldsymbol{b}_m\boldsymbol{b}_m^\top\right)\boldsymbol{x}_n + \left(\sum_{j=M+1}^{D}\boldsymbol{b}_j\boldsymbol{b}_j^\top\right)\boldsymbol{x}_n\,, && \text{(10.44b)}\end{aligned}$$

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 $\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n$, i.e., the difference vector between the original data point and its projection, is

$$\begin{aligned}\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n &= \left(\sum_{j=M+1}^{D}\boldsymbol{b}_j\boldsymbol{b}_j^\top\right)\boldsymbol{x}_n && \text{(10.45)}\\ &= \sum_{j=M+1}^{D}(\boldsymbol{x}_n^\top\boldsymbol{b}_j)\boldsymbol{b}_j\,. && \text{(10.46)}\end{aligned}$$

This means the difference is exactly the projection of the data point onto the orthogonal complement of the principal subspace: We identify the matrix $\sum_{j=M+1}^{D}\boldsymbol{b}_j\boldsymbol{b}_j^\top$ in (10.45) as the projection matrix that performs this projection. This also means the displacement vector $\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n$ lies in the subspace that is orthogonal to the principal subspace as illustrated in Figure 10.7.

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

Remark (Low-Rank Approximation). In (10.45), we saw that the projection matrix, which projects $\boldsymbol{x}$ onto $\tilde{\boldsymbol{x}}$ is given by

$$\sum_{m=1}^{M}\boldsymbol{b}_m\boldsymbol{b}_m^\top = \boldsymbol{B}\boldsymbol{B}^\top\,. \tag{10.47}$$

By construction as a sum of rank-one matrices $\boldsymbol{b}_m\boldsymbol{b}_m^\top$ we see that $\boldsymbol{B}\boldsymbol{B}^\top$ is symmetric and has rank M. Therefore, the average reconstruction error

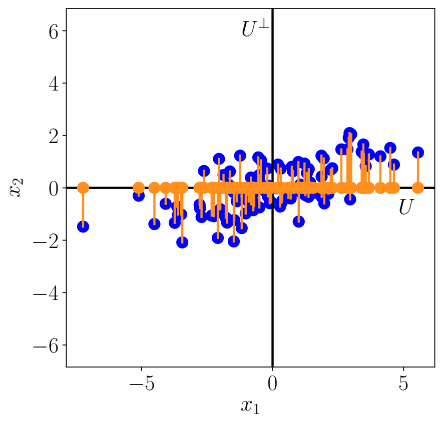


Figure 10.7 Orthogonal projection and displacement vectors. When projecting data points $\boldsymbol{x}_n$ (blue) onto subspace U_1 we obtain $\tilde{\boldsymbol{x}}_n$ (orange). The displacement vector $\tilde{\boldsymbol{x}}_n - \boldsymbol{x}_n$ lies completely in the orthogonal complement U_2 of U_1.

can also be written as

$$\sum_{n=1}^{N} \|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2 = \sum_{n=1}^{N} \left\|\boldsymbol{x}_n - \boldsymbol{B}\boldsymbol{B}^\top \boldsymbol{x}_n\right\| = \sum_{n=1}^{N} \left\|(\boldsymbol{I} - \boldsymbol{B}\boldsymbol{B}^\top)\boldsymbol{x}_n\right\|^2 . \tag{10.48}$$

Finding orthonormal basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M$ so that the difference between the original data $\boldsymbol{x}_n$ and their projections $\tilde{\boldsymbol{x}}_n$, $n = 1, \ldots, N$, is minimized is equivalent to finding the best rank-M approximation $\boldsymbol{B}\boldsymbol{B}^\top$ of the identity matrix $\boldsymbol{I}$, see Section 4.6. $\diamondsuit$

Now, we have all the tools to reformulate the loss function (10.31).

$$J = \frac{1}{N}\sum_{n=1}^{N} \|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2 \overset{(10.46)}{=} \frac{1}{N}\sum_{n=1}^{N} \left\| \sum_{j=M+1}^{D} (\boldsymbol{b}_j^\top \boldsymbol{x}_n)\boldsymbol{b}_j \right\|^2 . \tag{10.49}$$

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

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{j=M+1}^{D} (\boldsymbol{b}_j^\top \boldsymbol{x}_n)^2 = \frac{1}{N}\sum_{n=1}^{N}\sum_{j=M+1}^{D} \boldsymbol{b}_j^\top \boldsymbol{x}_n \boldsymbol{b}_j^\top \boldsymbol{x}_n \tag{10.50a}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{j=M+1}^{D} \boldsymbol{b}_j^\top \boldsymbol{x}_n \boldsymbol{x}_n^\top \boldsymbol{b}_j , \tag{10.50b}$$

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

$$J = \sum_{j=M+1}^{D} \boldsymbol{b}_j^\top \underbrace{\left(\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top\right)}_{=:\boldsymbol{S}} \boldsymbol{b}_j = \sum_{j=M+1}^{D} \boldsymbol{b}_j^\top \boldsymbol{S} \boldsymbol{b}_j \tag{10.51a}$$

$$= \sum_{j=M+1}^{D} \mathrm{tr}(\boldsymbol{b}_j^\top \boldsymbol{S}\boldsymbol{b}_j) \sum_{j=M+1}^{D} \mathrm{tr}(\boldsymbol{S}\boldsymbol{b}_j\boldsymbol{b}_j^\top) = \mathrm{tr}\Big(\underbrace{\Big(\sum_{j=M+1}^{D} \boldsymbol{b}_j\boldsymbol{b}_j^\top\Big)}_{\text{projection matrix}} \boldsymbol{S}\Big)\,, \tag{10.51b}$$

where we exploited the property that the trace operator $\mathrm{tr}(\cdot)$, see (4.16), is linear and invariant to cyclic permutations of its arguments. Since we assumed that our dataset is centered, i.e., $\mathbb{E}[\boldsymbol{X}] = \boldsymbol{0}$, we identify $\boldsymbol{S}$ as the data covariance matrix. We see that the projection matrix in (10.51b) is constructed as a sum of rank-one matrices $\boldsymbol{b}_j\boldsymbol{b}_j^\top$ so that it itself is of rank $D - M$.

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.

Equation (10.51a) implies that we can formulate the average squared reconstruction error equivalently as the covariance matrix of the data, projected onto the orthogonal complement of the principal subspace. Minimizing the average squared reconstruction error is therefore equivalent to minimizing the variance of the data when projected onto the subspace we ignore, i.e., the orthogonal complement of the principal subspace. Equivalently, we maximize the variance of the projection that we retain in the principal subspace, which links the projection loss immediately to the maximum-variance formulation of PCA discussed in Section 10.2. But this then also means that we will obtain the same solution that we obtained for the maximum-variance perspective. Therefore, we skip the slightly lengthy derivation here and summarize the results from earlier in the light of the projection perspective.

Minimizing the average squared reconstruction error is equivalent to maximizing the variance of the projected data.

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

$$J = \sum_{j=M+1}^{D} \lambda_j\,, \tag{10.52}$$

where λ_j are the eigenvalues of the data covariance matrix. Therefore, to minimize (10.52) we need to select the smallest $D - M$ eigenvalues, which then implies that their corresponding eigenvectors are the basis of the orthogonal complement of the principal subspace. Consequently, this means that the basis of the principal subspace are the eigenvectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_M$ that belong to the largest M eigenvalues of the data covariance matrix.

Example 10.2 (MNIST Digits Embedding)

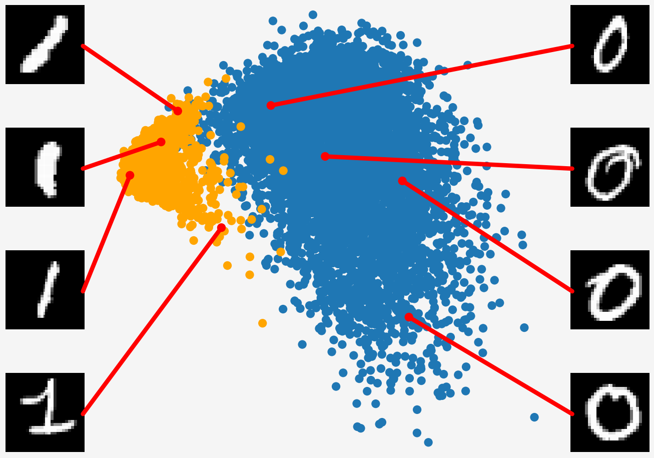


Figure 10.8 Embedding of MNIST digits 0 (blue) and 1 (orange) in a two-dimensional principal subspace using PCA. Four examples embeddings of the digits '0' and '1' in the principal subspace are highlighted in red with their corresponding original digit.

Figure 10.8 visualizes the training data of the MMIST digits '0' and '1' embedded in the vector subspace spanned by the first two principal components. We can see a relatively clear separation between '0's (blue dots) and '1's (orange dots), and we can see the variation within each individual cluster.

5450 10.4 Eigenvector Computation

In the previous sections, we obtained the basis of the principal subspace as the eigenvectors that belong to the largest eigenvalues of the data covariance matrix

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top , \tag{10.53}$$

$$\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_N] \in \mathbb{R}^{D\times N} . \tag{10.54}$$

5451 To get the eigenvalues (and the corresponding eigenvectors) of $\boldsymbol{S}$, we can
5452 follow two approaches:

Eigendecomposition or SVD to compute eigenvectors.

- 5453 We perform an eigendecomposition (see Section 4.2) and compute the
5454 eigenvalues and eigenvectors of $\boldsymbol{S}$ directly.
- We use a singular value decomposition (see Section 4.5). Since $\boldsymbol{S}$ is symmetric and factorizes into $\boldsymbol{X}\boldsymbol{X}^\top$ (ignoring the factor $\frac{1}{N}$), the eigenvalues of $\boldsymbol{S}$ are the squared singular values of $\boldsymbol{X}$. More specifically, if

the SVD of $\boldsymbol{X}$ is given by

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top , \tag{10.55}$$

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

$$\boldsymbol{S} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top = \frac{1}{N}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top = \frac{1}{N}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top . \tag{10.56}$$

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

$$\lambda_i = \frac{\sigma_i^2}{N} . \tag{10.57}$$

Practical aspects Finding eigenvalues and eigenvectors is also important in other fundamental machine learning methods that require matrix decompositions. In theory, as we discussed in Section 4.2, we can solve for the eigenvalues as roots of the characteristic polynomial. However, for matrices larger than 4×4 this is not possible because we would need to find the roots of a polynomial of degree 5 or higher. However, the Abel-Ruffini theorem (Ruffini, 1799; Abel, 1826) states that there exists no algebraic solution to this problem for polynomials of degree 5 or more. Therefore, in practice, we solve for eigenvalues or singular values using iterative methods, which are 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) iterative processes, 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 $\boldsymbol{x}_0$ and follows the iteration

power iteration

$$\boldsymbol{x}_{k+1} = \frac{\boldsymbol{S}\boldsymbol{x}_k}{\|\boldsymbol{S}\boldsymbol{x}_k\|} , \quad k = 0, 1, \dots . \tag{10.58}$$

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

10.5 PCA Algorithm

In the following, we will go through the individual steps of PCA using a running example, which is summarized in Figure 10.9. We are given a two-dimensional data set (Figure 10.9(a)), and we want to use PCA to project it onto a one-dimensional subspace.

1. **Mean subtraction** We start by centering the data by computing the mean $\boldsymbol{\mu}$ of the dataset and subtracting it from every single data point. This ensures that the data set has mean **0** (Figure 10.9(b)). Mean subtraction is not strictly necessary but reduces the risk of numerical problems.
2. **Standardization** Divide the data points by the standard deviation σ_d of the dataset for every dimension $d = 1, \dots, D$. Now the data is unit free, and it has variance 1 along each axis, which is indicated by the two arrows in Figure 10.9(c). This step completes the *standardization* of the data.
3. **Eigendecomposition of the covariance matrix** Compute the data covariance matrix and its eigenvalues and corresponding eigenvectors. In Figure 10.9(d), the eigenvectors are scaled by the magnitude of the corresponding eigenvalue. The longer vector spans the principal subspace, which we denote by U. The data covariance matrix is represented by the ellipse.
4. **Projection** We can project any data point $\boldsymbol{x}_* \in \mathbb{R}^D$ onto the principal subspace: To get this right, we need to standardize $\boldsymbol{x}_*$ using the mean and standard deviation of the data set that we used to compute the data covariance matrix, i.e.,

$$x_*^{(d)} \leftarrow \frac{x_*^{(d)} - \mu^{(d)}}{\sigma_d}, \quad d = 1, \dots, D, \tag{10.59}$$

where $x^{(d)}$ is the dth component of $\boldsymbol{x}$. Then, we obtain the projected data point as

$$\tilde{\boldsymbol{x}}_* = \boldsymbol{B}\boldsymbol{B}^\top \boldsymbol{x}_* \tag{10.60}$$

with coordinates $\boldsymbol{z}_* = \boldsymbol{B}^\top \boldsymbol{x}_*$ with respect to the basis of the principal subspace. Here, $\boldsymbol{B}$ is the matrix that contains the eigenvectors that belong to the largest eigenvalues of the data covariance matrix as columns.
5. **Moving back to data space** To see our projection in the original data format (i.e., before standardization), we need to undo the standardization (10.59) and multiply by the standard deviation before adding the mean so that we obtain

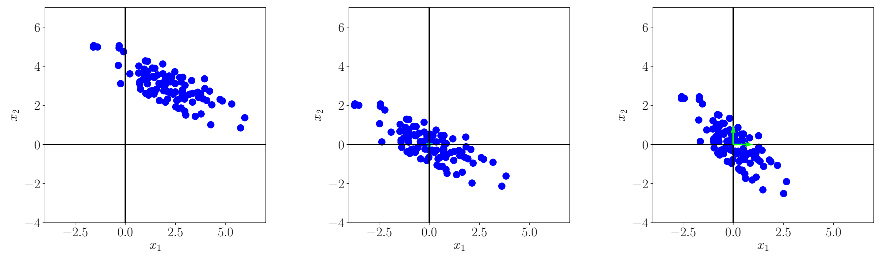
$$\tilde{x}_*^{(d)} \leftarrow \tilde{x}_*^{(d)} \sigma_d + \mu^{(d)}, \quad d = 1, \dots, D, \tag{10.61}$$

where $\mu^{(d)}$ and σ_d are the mean and standard deviation of the training

standardization

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

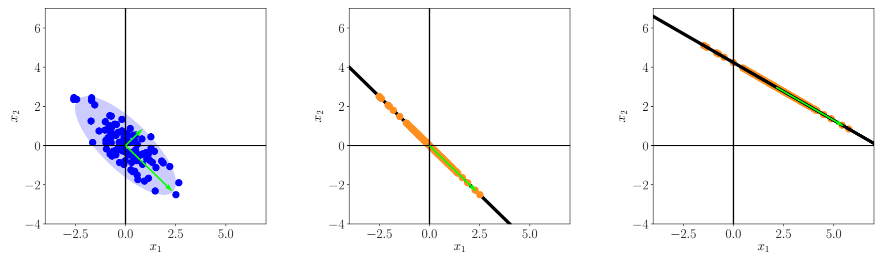
Figure 10.9 Steps of PCA.



(a) Original dataset.

(b) Step 1: Centering by subtracting the mean from each data point.

(c) Step 2: Dividing by the standard deviation to make the data unit free. Data has variance 1 along each axis.



(d) Step 3: Compute eigenvalues and eigenvectors (arrows) of the data covariance matrix (ellipse).

(e) Step 4: Project data onto the subspace spanned by the eigenvectors belonging to the largest eigenvalues (principal subspace).

(f) Step 5: Undo the standardization and move projected data back into the original data space from (a).

data in the dth dimension, respectively. Figure 10.9(f) illustrates the projection in the original data format.

Example 10.3 (MNIST Digits: Reconstruction)

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

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 28×28, i.e., it contains 784 pixels so that we can interpret every image in this dataset as a vector $\boldsymbol{x} \in \mathbb{R}^{784}$. 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 10.10. The first row of Figure 10.10 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 can see that even with a single-dimensional principal subspace we get a halfway 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 can be 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.

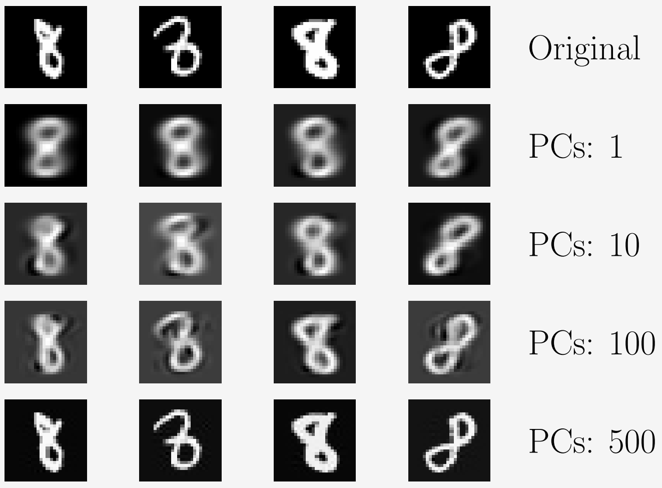


Figure 10.10 Effect of increasing number of principal components on reconstruction.

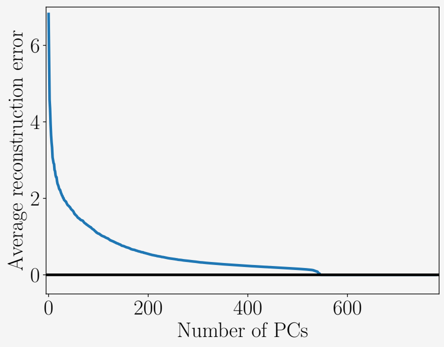


Figure 10.11 Average reconstruction error as a function of the number of principal components.

Figure 10.11 shows the average reconstruction error, which is

$$\frac{1}{N}\sum_{n=1}^{N}\|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\| = \sum_{d=1}^{D}\sqrt{\lambda_d}\,, \tag{10.62}$$

as a function of the number 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. With about 550 PCs,

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

we can essentially fully reconstruct the training data that contains the digit ‘8’.

10.6 PCA in High Dimensions

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

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

$$\boldsymbol{S} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{D\times D}\,, \tag{10.63}$$

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

We now assume that $N \ll D$, i.e., the number of data points is smaller than the dimensionality of the data. Then the rank of the covariance matrix $\boldsymbol{S}$ is N, and it has $D - N + 1$ many eigenvalues that are 0. Intuitively, this means that there are some redundancies.

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

In PCA, we ended up with the eigenvector equation

$$\boldsymbol{S}\boldsymbol{b}_m = \lambda_m \boldsymbol{b}_m\,, \quad m = 1, \dots, M\,, \tag{10.64}$$

where $\boldsymbol{b}_m$ is a basis vector of the principal subspace. Let us re-write this equation a bit: With $\boldsymbol{S}$ defined in (10.63), we obtain

$$\boldsymbol{S}\boldsymbol{b}_m = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{b}_m = \lambda_m \boldsymbol{b}_m\,. \tag{10.65}$$

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

$$\frac{1}{N}\underbrace{\boldsymbol{X}^\top\boldsymbol{X}}_{N\times N}\underbrace{\boldsymbol{X}^\top\boldsymbol{b}_m}_{=:\boldsymbol{c}_m} = \lambda_m \boldsymbol{X}^\top \boldsymbol{b}_m \iff \frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{c}_m = \lambda_m \boldsymbol{c}_m\,, \tag{10.66}$$

and we get a new eigenvector/eigenvalue equation: λ_m is still the eigenvalue, but the eigenvector is now $\boldsymbol{c}_m := \boldsymbol{X}^\top\boldsymbol{b}_m$ of the matrix $\frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X} \in \mathbb{R}^{N\times N}$. Assuming we have no duplicate data points, this matrix has rank N

and is invertible. This also implies that $\frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X}$ has the same (non-zero) eigenvalues as the data covariance matrix $\boldsymbol{S}$. But this is now an $N \times N$ matrix, so that we can compute the eigenvalues and eigenvectors much more efficiently than for the original $D \times D$ data covariance matrix.

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

$$\underbrace{\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top}_{\boldsymbol{S}}\boldsymbol{X}\boldsymbol{c}_m = \lambda_m\boldsymbol{X}\boldsymbol{c}_m \tag{10.67}$$

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

Remark. If we want to apply the PCA algorithm that we discussed in Section 10.5 we need to normalize the eigenvectors $\boldsymbol{X}\boldsymbol{c}_m$ of $\boldsymbol{S}$ so that they have norm 1. ◇

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, 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 8.4,
- 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,
- allow us to extend the model fairly straightforwardly, 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 $\boldsymbol{z} \in \mathbb{R}^M$ 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

Probabilistic PCA

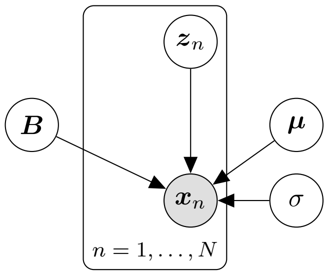


Figure 10.12 Graphical model for probabilistic PCA. The observations $\boldsymbol{x}_n$ explicitly depend on corresponding latent variables $\boldsymbol{z}_n \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. The model parameters $\boldsymbol{B}, \boldsymbol{\mu}$ and the likelihood parameter σ are shared across the dataset.

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.

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 $\boldsymbol{z} \in \mathbb{R}^M$ with a standard-Normal prior $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and a linear relationship between the latent variables and the observed $\boldsymbol{x}$ data where

$$\boldsymbol{x} = \boldsymbol{B}\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \in \mathbb{R}^D , \tag{10.68}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$ is Gaussian observation noise, $\boldsymbol{B} \in \mathbb{R}^{D\times M}$ 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

$$p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2) = \mathcal{N}(\boldsymbol{x} \,|\, \boldsymbol{B}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2\boldsymbol{I}) . \tag{10.69}$$

Overall, PPCA induces the following generative process:

$$\boldsymbol{z}_n \sim \mathcal{N}(\boldsymbol{z} \,|\, \boldsymbol{0}, \boldsymbol{I}) \tag{10.70}$$

$$\boldsymbol{x}_n \,|\, \boldsymbol{z}_n \sim \mathcal{N}(\boldsymbol{x} \,|\, \boldsymbol{B}\boldsymbol{z}_n + \boldsymbol{\mu}, \sigma^2\boldsymbol{I}) \tag{10.71}$$

To generate a data point that is typical given the model parameters, we follow an *ancestral sampling* scheme: We first sample a latent variable $\boldsymbol{z}_n$ from $p(\boldsymbol{z})$. Then, we use $\boldsymbol{z}_n$ in (10.69) to sample a data point conditioned on the sampled $\boldsymbol{z}_n$, i.e., $\boldsymbol{x}_n \sim p(\boldsymbol{x} \,|\, \boldsymbol{z}_n, \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2)$.

ancestral sampling

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

$$p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{B}, \boldsymbol{\mu}, \sigma^2) = p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2)p(\boldsymbol{z}) , \tag{10.72}$$

which immediately gives rise to the graphical model in Figure 10.12 using the results from Section 8.5.

Remark. Note the direction of the arrow that connects the latent variables $\boldsymbol{z}$ and the observed data $\boldsymbol{x}$: The arrow points from $\boldsymbol{z}$ to $\boldsymbol{x}$, which means that the PPCA model assumes a lower-dimensional latent cause $\boldsymbol{z}$ for high-dimensional observations $\boldsymbol{x}$. In the end, we are obviously interested in

5568 finding something out about z given some observations. To get there we
5569 will apply Bayesian inference to "invert" the arrow implicitly and go from
5570 observations to latent variables. ◇

Example 10.4 (Generating Data from Latent Variables)

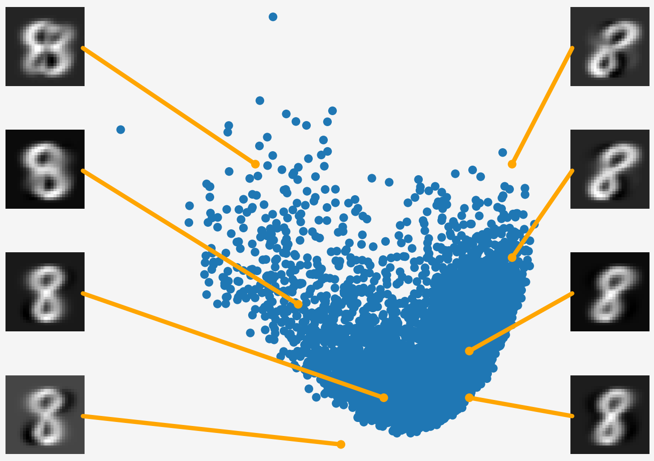


Figure 10.13 Generating new MNIST digits. The latent variables $\boldsymbol{z}$ can be used to generate new data $\tilde{\boldsymbol{x}} = \boldsymbol{B}\boldsymbol{z}$. The closer we stay to the training data the more realistic the generated data.

Figure 10.13 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 $\boldsymbol{z}_*$ in this latent space an generate an image $\tilde{\boldsymbol{x}}_* = \boldsymbol{B}\boldsymbol{z}_*$ 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 an more artefacts, e.g., the top-left and top-right digits. Note that the intrinsic dimensionality of these generated images is only two.

5571 *10.7.2 Likelihood and Joint Distribution*

Using the results from Chapter 6, we obtain the marginal distribution of the data $\boldsymbol{x}$ by integrating out the latent variable $\boldsymbol{z}$ so that

$$\begin{aligned} p(\boldsymbol{x} \mid \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2) &= \int p(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\mu}, \sigma^2) p(\boldsymbol{z}) \mathrm{d}\boldsymbol{z} \\ &= \int \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{B}\boldsymbol{z} + \boldsymbol{\mu},\, \sigma^2 \boldsymbol{I}) \mathcal{N}(\boldsymbol{z} \mid \boldsymbol{0},\, \boldsymbol{I}) \mathrm{d}\boldsymbol{z}\,. \end{aligned} \tag{10.73}$$

From Section 6.6, we know that the solution to this integral is a Gaussian distribution with mean

$$\mathbb{E}[\boldsymbol{x}] = \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{B}\boldsymbol{z} + \boldsymbol{\mu}] + \mathbb{E}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon}] = \boldsymbol{\mu} \tag{10.74}$$

and with covariance matrix

$$\begin{aligned} \mathbb{V}[\boldsymbol{x}] &= \mathbb{V}_{\boldsymbol{z}}[\boldsymbol{B}\boldsymbol{z} + \boldsymbol{\mu}] + \mathbb{V}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon}] = \mathbb{V}_{\boldsymbol{z}}[\boldsymbol{B}\boldsymbol{z}] \\ &= \boldsymbol{B}\mathbb{V}_{\boldsymbol{z}}[z]\boldsymbol{B}^\top + \sigma^2\boldsymbol{I} = \boldsymbol{B}\boldsymbol{B}^\top + \sigma^2\boldsymbol{I}\,. \end{aligned} \tag{10.75}$$

PPCA likelihood

The marginal distribution in (10.73) is the *PPCA likelihood*, which we can use for maximum likelihood or MAP estimation of the model parameters.

Remark. Although the conditional distribution in (10.69) is also a likelihood, we cannot use it 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 $\boldsymbol{x}$ and the model parameters, but not on the latent variables. ◇

From Section 6.6 we also know that the joint distribution of a Gaussian random variable $\boldsymbol{z}$ and a linear/affine transformation $\boldsymbol{x} = \boldsymbol{B}\boldsymbol{z}$ of it are jointly Gaussian distributed. We already know the marginals $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}\,|\,\boldsymbol{0},\,\boldsymbol{I})$ and $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}\,|\,\boldsymbol{\mu},\,\boldsymbol{B}\boldsymbol{B}^\top + \sigma^2\boldsymbol{I})$. The missing cross-covariance is given as

$$\mathrm{Cov}[\boldsymbol{x}, \boldsymbol{z}] = \mathrm{Cov}_{\boldsymbol{z}}[\boldsymbol{B}\boldsymbol{z} + \boldsymbol{\mu}] = \boldsymbol{B}\,\mathrm{Cov}_{\boldsymbol{z}}[\boldsymbol{z}, \boldsymbol{z}] = \boldsymbol{B}\,. \tag{10.76}$$

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

$$p(\boldsymbol{x}, \boldsymbol{z}\,|\,\boldsymbol{B}, \boldsymbol{\mu}, \sigma^2) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{z}\end{bmatrix} \,\middle|\, \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{0}\end{bmatrix}, \begin{bmatrix}\boldsymbol{B}\boldsymbol{B}^\top + \sigma^2\boldsymbol{I} & \boldsymbol{B}\\ \boldsymbol{B}^\top & \boldsymbol{I}\end{bmatrix}\right), \tag{10.77}$$

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

10.7.3 Posterior Distribution

The joint Gaussian distribution $p(\boldsymbol{x}, \boldsymbol{z}\,|\,\boldsymbol{B}, \boldsymbol{\mu}, \sigma^2)$ in (10.77) allows us to determine the posterior distribution $p(\boldsymbol{z}\,|\,\boldsymbol{x})$ immediately by applying the rules of Gaussian conditioning from Section 6.6.1. The posterior distribution of the latent variable given an observation $\boldsymbol{x}$ is then

$$p(\boldsymbol{z}\,|\,\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}\,|\,\boldsymbol{m},\,\boldsymbol{C})\,, \tag{10.78a}$$

$$\boldsymbol{m} = \boldsymbol{B}^\top(\boldsymbol{B}\boldsymbol{B}^\top + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\,, \tag{10.78b}$$

$$\boldsymbol{C} = \boldsymbol{I} - \boldsymbol{B}^\top(\boldsymbol{B}\boldsymbol{B}^\top + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{B}\,. \tag{10.78c}$$

The posterior distribution in (10.78) reveals a few things. First, the posterior mean $\boldsymbol{m}$ is effectively an orthogonal projection of the mean-centered data $\boldsymbol{x} - \boldsymbol{\mu}$ onto the vector subspace spanned by the columns of $\boldsymbol{B}$. If we ignore the measurement noise contribution we immediately recover the projection equation (3.55) from Section 3.6. Second, the posterior covariance matrix $\boldsymbol{C}$ does not directly depend on the observed data $\boldsymbol{x}$.

If we now have a new observation $\boldsymbol{x}_*$ in data space, we can use (10.78) to determine the posterior distribution of the corresponding latent variable $\boldsymbol{z}_*$. The covariance matrix $\boldsymbol{C}$ allows us to assess how confident the embedding is. A covariance matrix $\boldsymbol{C}$ with a small determinant (which measures volumes) tells us that the latent embedding $\boldsymbol{z}_*$ is fairly certain. However, if we obtain a posterior distribution $p(\boldsymbol{z}_* \,|\, \boldsymbol{x}_*)$ that is uncertain, we may be faced with an outlier. However, we can explore this posterior distribution to understand what other data points $\boldsymbol{x}$ are plausible under this posterior. To do this, we can exploit PPCA's generative process. The generative process underlying PPCA allows us to explore the posterior distribution on the latent variables by generating new data that are plausible under this posterior. This can be achieved as follows:

1. Sample a latent variable $\boldsymbol{z}_* \sim p(\boldsymbol{z} \,|\, \boldsymbol{x}_*)$ from the posterior distribution over the latent variables (10.78)
2. Sample a reconstructed vector $\tilde{\boldsymbol{x}}_* \sim p(\boldsymbol{x} \,|\, \boldsymbol{z}_*, \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2)$ from (10.69)

If we repeat this process many times, we can explore the posterior distribution (10.78) on the latent variables $\boldsymbol{z}_*$ and its implications on the observed data. The sampling process effectively hypothesizes data, which is plausible under the posterior distribution.

10.8 Further Reading

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

We can also think of PCA as a linear *auto-encoder* as illustrated in Figure 10.14. An auto-encoder encodes the data $\boldsymbol{x}_n \in \mathbb{R}^D$ to a *code* $\boldsymbol{z}_n \in \mathbb{R}^M$ and tries to decode it to a $\tilde{\boldsymbol{x}}_n$ similar to $\boldsymbol{x}_n$. The mapping from the data to the code is called the *encoder*, the mapping from the code back to the original data space is called the *decoder*. If we consider linear mappings where the code is given by $\boldsymbol{z}_n = \boldsymbol{B}^\top\boldsymbol{x}_n \in \mathbb{R}^M$ and we are interested in minimizing the average squared error between the data $\boldsymbol{x}_n$ and its reconstruction

auto-encoder
code
encoder
decoder

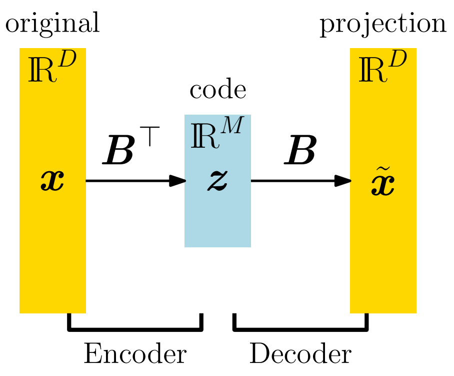


Figure 10.14 PCA can be viewed as a linear auto-encoder. It encodes the high-dimensional data $\boldsymbol{x}$ into a lower-dimensional representation (code) $\boldsymbol{z} \in \mathbb{R}^M$ and decode $\boldsymbol{z}$ using a decoder. The decoded vector $\tilde{\boldsymbol{x}}$ is the orthogonal projection of the original data $\boldsymbol{x}$ onto the M-dimensional principal subspace.

$\tilde{\boldsymbol{x}}_n = \boldsymbol{B}\boldsymbol{z}_n$, $n = 1, \ldots, N$, we obtain

$$\frac{1}{N}\sum_{n=1}^{N}\|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2 = \frac{1}{N}\sum_{n=1}^{N}\left\|\boldsymbol{x}_n - \boldsymbol{B}^\top\boldsymbol{B}\boldsymbol{x}_n\right\|^2 . \tag{10.79}$$

This means we end up with the same objective function as in (10.31) that we discussed in Section 10.3 so that we obtain the PCA solution when we minimize the squared auto-encoding loss. If we replace the linear mapping of PCA with a nonlinear mapping, we get a nonlinear auto-encoder. A prominent example of this is a deep auto-encoder where the linear functions are replaced with deep neural networks. In this context, the encoder is also know as *recognition network* or *inference network*, whereas the decoder is also called a *generator*.

recognition network
inference network
generator

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

The code is a compressed version of the original data.

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

$$\boldsymbol{\mu}_{\text{ML}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n , \tag{10.80}$$

$$\boldsymbol{B}_{\text{ML}} = \boldsymbol{T}(\boldsymbol{\Lambda} - \sigma^2\boldsymbol{I})^{\frac{1}{2}} , \tag{10.81}$$

$$\sigma^2_{\mathrm{ML}} = \frac{1}{D-M} \sum_{j=M+1}^{D} \lambda_j \,, \tag{10.82}$$

where $\boldsymbol{T} \in \mathbb{R}^{D \times M}$ contains M eigenvectors of the data covariance matrix, and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_M) \in \mathbb{R}^{M \times M}$ is a diagonal matrix with the eigenvalues belonging to the principal axes on its diagonal. The maximum likelihood solution $\boldsymbol{B}_{\mathrm{ML}}$ is unique up to a arbitrary rotations, i.e., we can right-multiply $\boldsymbol{B}_{\mathrm{ML}}$ with any rotation matrix $\boldsymbol{R} \in \mathbb{R}^{M \times M}$ so that (10.81) essentially is a singular value decomposition (see Section 4.5). An outline of the proof is given by Tipping and Bishop (1999).

The maximum likelihood estimate for $\boldsymbol{\mu}$ given in (10.80) is the sample mean of the data. The maximum likelihood estimator for the observation noise variance σ^2 given in (10.82) is the sum of all variances in the orthogonal complement of the principal subspace, i.e., the leftover variance that we cannot capture with the first M principal components are treated as observation noise.

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

$$\boldsymbol{S} = \boldsymbol{T}\boldsymbol{\Lambda}\boldsymbol{T}^{-1} \,. \tag{10.83}$$

In the PPCA model, the data covariance matrix is the covariance matrix of the likelihood $p(\boldsymbol{X} \,|\, \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2)$, which is $\boldsymbol{B}\boldsymbol{B}^\top + \sigma^2 \boldsymbol{I}$, see (10.75). For $\sigma \to 0$, we obtain $\boldsymbol{B}\boldsymbol{B}^\top$ so that this data covariance must equal the PCA data covariance (and its factorization given in (10.83)) so that

$$\mathrm{Cov}[\boldsymbol{X}] = \boldsymbol{T}\boldsymbol{\Lambda}\boldsymbol{T}^{-1} = \boldsymbol{B}\boldsymbol{B}^\top \iff \boldsymbol{B} = \boldsymbol{T}\boldsymbol{\Lambda}^{\frac{1}{2}} \,, \tag{10.84}$$

which is exactly the maximum likelihood estimate in (10.81) for $\sigma = 0$.

From (10.81) and (10.83) it becomes clear that (P)PCA performs a decomposition of the data covariance matrix. We can also think of PCA as a method for finding the best rank-M approximation of the data covariance matrix so that we can apply the Eckart-Young Theorem from Section 4.6, which states that the optimal solution can be determined using a singular value decomposition.

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).

Similar to our discussion on linear regression in Chapter 9, we can place a prior distribution on the parameters of the model and to integrate them out, thereby avoiding a) point estimates of the parameters and the issues that come with these point estimates (see Section 8.4) and b) allowing for an automatic selection of the appropriate dimensionality M of the latent space. In this *Bayesian PCA*, which was proposed by Bishop (1999),

Bayesian PCA

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.

places a (hierarchical) prior $p(\boldsymbol{\mu}, \boldsymbol{B}, \sigma^2)$ on the model parameters. The generative process allows us to integrates 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.

factor analysis

In PPCA, we considered the linear model $\boldsymbol{x}_n = \boldsymbol{B}\boldsymbol{z}_n + \boldsymbol{\epsilon}$ with $p(\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$, i.e., all observation dimensions are affected by the same amount of noise. If we allow each observation dimension d to have a different variance σ_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 $\boldsymbol{B}$, $\boldsymbol{\mu}$. However, FA no longer allows for a closed-form solution to maximum likelihood so that we need to use an iterative scheme, such as the EM 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.

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

Independent Component Analysis

Independent Component Analysis (ICA) is also closely related to PCA. Starting again with the model $\boldsymbol{x}_n = \boldsymbol{B}\boldsymbol{z}_n + \boldsymbol{\epsilon}$ we now change the prior on $\boldsymbol{z}_n$ 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 $p(\boldsymbol{z})$ on the latent sources to require non-Gaussian priors $p(\boldsymbol{z})$. We refer to the book by *Murphy2012* for more details on ICA.

blind-source separation

Murphy2012

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 10.6, 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).

kernel trick
kernel PCA

There are nonlinear dimensionality reduction techniques that are derived from PCA. 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 $\boldsymbol{z}$ and the observations $\boldsymbol{x}$ 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 $\boldsymbol{z}$. Similar to Bayesian PCA, the *Bayesian GP-LVM* proposed by Titsias and Lawrence (2010) maintains a distribution on the latent variables $\boldsymbol{z}$ and uses approximate inference to integrate them out as well.

deep auto-encoder

Gaussian Process Latent Variable Model

Bayesian GP-LVM

©2018 Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. To be published by Cambridge University Press.