Principal Component Analysis

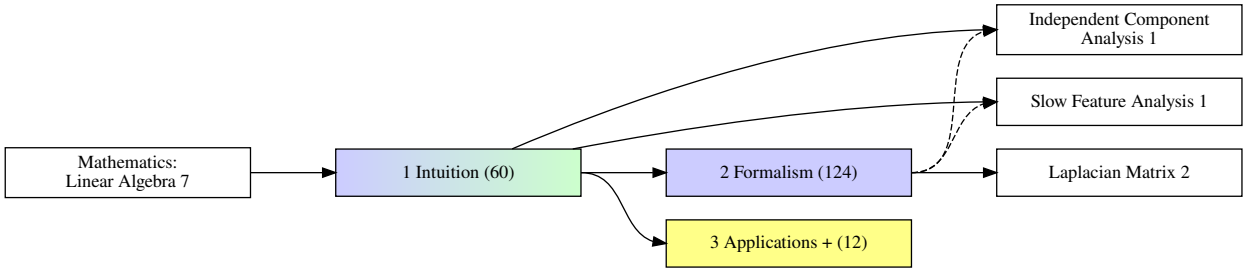
— Lecture Notes —

Laurenz Wiskott
Institut für Neuroinformatik
Fakultät für Informatik
Ruhr-Universität Bochum, Germany, EU

14 December 2016

— Summary —

Principal component analysis (PCA) can be applied to vectorial data and is probably the most common method to reduce the dimensionality of data for compression and visualization. It determines the dimensions of largest and smallest variance of the data, referred to as the principal components, which can then be used to discard the small variance dimensions for dimensionality reduction or select the two or three largest variance dimensions for visualization. For instance, if you have one thousand 100-dimensional data points, PCA might be used to reduce the dimensionality of the data down to 10 without loosing too much information, which corresponds to a compression by 90%. In some cases it is also the small variance directions that are of interest. PCA can also be used to normalize data such, that it has unit variance in all directions, a process called whitening or sphering, to eliminate correlations.



© 2004–2006, 2009, 2010, 2013, 2016 Laurenz Wiskott (ORCID https://orcid.org/0000-0001-6237-740X, homepage https://www.ini.rub.de/PEOPLE/wiskott/). This work (except for all figures from other sources, if present) is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License, see https://creativecommons.org/licenses/by-sa/4.0/. If figures are not included for copyright reasons, they are uni colored, but the word 'Figure', 'Image', or the like in the reference is often linked to a freely available copy.
If applicable, dark red marks core text and sometimes important formulas, a filled diamond ♦ marks important formulas, and an empty diamond ◊ marks less important formulas or items that I would usually also present in a lecture to give context or dive into the details; core text and important formulas are worth remembering and learning for an exam; + marks sections that I would usually skip in a lecture.
You can also download the teaching material of this topic as zip files and then view them locally on your computer.

— **Lecture 1/2** — → Lecture 1 Exercises, Lecture 1 Solutions

1 Intuition introduces the problem statement of principal component analysis and the general idea of the algorithm without much formalism.

2 Formalism provides a detailed mathematical treatment of principal component analysis, including whitening and singular value decommposition, an extension for very high dimensional spaces. (2.1–2.5)

— **Lecture 2/2** — → Lecture 2 Exercises, Lecture 2 Solutions

2 Formalism (2.6–2.19)

3 Application + presents an application to face processing. A large number of images of faces are aligned at the eyes and then treated as data vectors by concatenating the rows (or columns). The mean represents the average face and the principal components the dominant modes of variation. This can be used for face recognition and analysis.

Contents

LECTURE 1/2 **3**

1 Intuition **3**

1.1 Problem statement . . . 4
1.2 Projection and reconstruction error . . . 5
1.3 Reconstruction error and variance . . . 6
1.4 Covariance matrix . . . 6
1.5 Covariance matrix and higher order structure . . . 7
1.6 PCA by diagonalizing the covariance matrix . . . 8
1.7 Whitening . . . 8

2 Formalism **9**

2.1 Definition of the PCA-optimization problem . . . 9
2.2 Matrix $\mathbf{V}^T$: Mapping from high-dimensional old coordinate system to low-dimensional new coordinate system . . . 10
2.3 Matrix $\mathbf{V}$: Mapping from low-dimensional new coordinate system to subspace in old coordinate system . . . 11
2.4 Matrix $(\mathbf{V}^T\mathbf{V})$: Identity mapping within new coordinate system . . . 12
2.5 Matrix $(\mathbf{V}\mathbf{V}^T)$: Projection from high- to low-dimensional (sub)space within old coordinate system . . . 13

LECTURE 2/2 **14**
2.6 Variance . . . 14
2.7 Reconstruction error . . . 15
2.8 Covariance matrix . . . 15
2.9 Eigenvalue equation of the covariance matrix . . . 16
2.10 Total variance of the data $\mathbf{x}$ . . . 16
2.11 Diagonalizing the covariance matrix . . . 17
2.12 Variance of $\mathbf{y}$ for a diagonalized covariance matrix . . . 17
2.13 Constraints of matrix $\mathbf{V}'$ . . . 18
2.14 Finding the optimal subspace . . . 18
2.15 Interpretation of the result . . . 18
2.16 PCA Algorithm . . . 19
2.17 Intuition of the Results . . . 19
2.18 Whitening or sphering . . . 20
2.19 Singular value decomposition + . . . 21

3 Application + **22**
3.1 Face processing . . . 22

A Other resources + **24**
A.1 Written material . . . 24
A.2 Visualizations . . . 24
A.3 Videos . . . 24
A.4 Software . . . 26

LECTURE 1/2

- Lecture 1 Exercises, Lecture 1 Solutions

1 Intuition

Learning material:[1]

○ 2 min video 0 In a Nutshell

■ Text below

[1]Generic instruction: Consider the (possibly nested) list of resources like a horizontal tree with an invisible root on the very left, and decide from left to right what you want to select to work through. The invisible root node has to be selected. For any selected parent node all children nodes marked with ■ or ● are mandatory and have to be selected.

Children nodes marked with □ or ○ are optional and may be selected in addition to get a better understanding of the material. If a parent node has no mandatory child, then at least one optional child has to be selected. Children marked with + provide additional voluntary material that can be safely ignored, typically going beyond the scope of the section. Children of non-selected parents may be ignored. ■ and □ indicate children that cover (almost) the whole material of the section. Missing content might then be indicated by struck through references to the corresponding learning objectives. ● and ○ indicate children that do not cover (almost) the whole material of the section. They must be combined with others to be complete. Items tend to be ordered by precedence and/or recommended temporal order from top to bottom, assuming that you prefer to first watch a video before reading through lecture notes. If a detailed table of content for videos or lecture notes is given, references to learning objectives might be provided in green, 1:30 should be read as 1 min and 30 seconds, and 1'30 should be read as page 1 at about 30% of the page. Video times may be linked directly to the indicated position in the video, but be aware that the video might be downloaded anew each time you click on a time. Resources without author name are usually authored by Laurenz Wiskott and his team.

Quizzes and Exercises: Deepen and test your understanding with the following

- Demo
- Section 1 Exercises, Section 1 Solutions
- Python Exercises, Python Solutions

This section is partly based on (Hertz et al., 1991).

1.1 Problem statement

Experimental data to be analyzed is often represented as a number of vectors of fixed dimensionality. A single vector could for example be a set of temperature measurements across Germany. Taking such a vector of measurements at different times results in a number of vectors that altogether constitute the data. Each vector can also be interpreted as a point in a high dimensional space. Then the data are simply a cloud of points in this space (if one ignores the temporal order, otherwise it would be a trajectory).

When analyzing such data one often encounters the problem that the dimensionality of the data points is too high to be visualized or analyzed with some particular technique. Thus the problem arises to reduce the dimensionality of the data in some optimal way.

To keep things simple we insist that the dimensionality reduction is done linearly, i.e. we are looking for a low-dimensional linear subspace of the data space, onto which the data can be projected. As a criterion for what the optimal subspace might be it seems reasonable to require that it should be possible to reconstruct the original data points from the reduced ones as well as possible. Thus if one were to project the data back from the low-dimensional space into the original high-dimensional space, the reconstructed data points should lie as close as possible to the original ones, with the mean squared distance between original and reconstructed data points being the reconstruction error. The question is, how can we find the linear subspace that minimizes this reconstruction error.

It is useful and common practice to remove the mean value from the data first before doing the dimensionality reduction as stated above. Thus, we assume zero mean data throughout. As a result variances and 2nd moments are the same. This justifies the slightly confusing common practice to speak of variances but write the equations for 2nd moments. Please keep that in mind.

1.2 Projection and reconstruction error

The task of principal component analysis (PCA) is to reduce the dimensionality of some high-dimensional data points by linearly projecting them onto a lower-dimensional space in such a way that the reconstruction error made by this projection is minimal. In order to develop an intuition for PCA we first take a closer look at what it means to project the data points and to reconstruct them. Figure 1.1 illustrates the process.

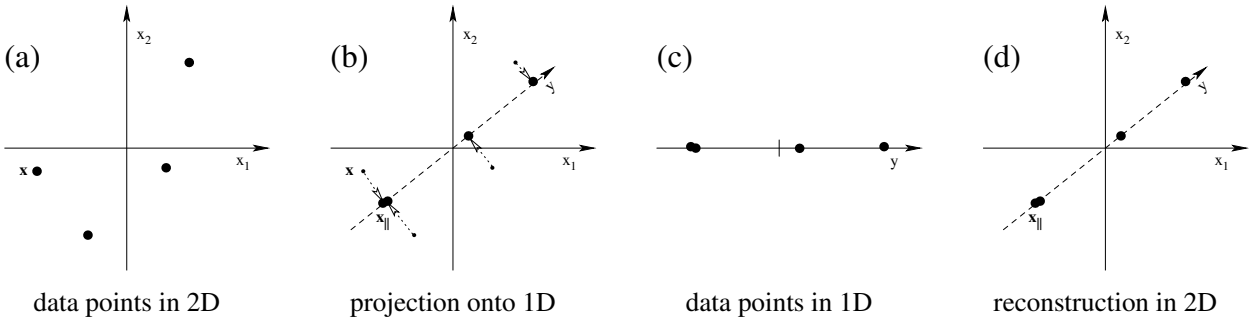


Figure 1.1: Projection of 2D data points onto a 1D subspace and their reconstruction.
© CC BY-SA 4.0

(a) A few data points are given in a two-dimensional space and are represented by two-dimensional vectors $\boldsymbol{x} = (x_1, x_2)$. (b) In order to reduce the dimensionality down to one, we have to choose a one-dimensional subspace defined by a unit vector $\boldsymbol{v}$ and project the data points onto it, which can be done by

$$\boldsymbol{x}_{\parallel} := \boldsymbol{v}\boldsymbol{v}^T\boldsymbol{x}\,. \tag{1.1}$$

(c) The points can now be represented by just one number,

$$y := \boldsymbol{v}^T\boldsymbol{x}\,, \tag{1.2}$$

and we do not care that they originally came from a two-dimensional space. (d) If we want to reconstruct the original two-dimensional positions of the data points as well as possible, we have to embed the one-dimensional space in the original two-dimensional space in exactly the orientation used during the projection,

$$\boldsymbol{x}_{\parallel} \overset{(1.1,1.2)}{=} \boldsymbol{v}y\,. \tag{1.3}$$

However, we cannot recover the accurate 2D-position; the points remain on the one-dimensional subspace. The reconstruction error is therefore the average distance of the original 2D-positions from the one-dimensional subspace (the length of the projection arrows in (b)). For mathematical convenience one actually takes the average squared distance

$$E := \langle \|\boldsymbol{x}^\mu - \boldsymbol{x}^\mu_{||}\|^2 \rangle_\mu \tag{1.4}$$

$$= \frac{1}{M}\sum_{\mu=1}^{M}\sum_{i=1}^{I}(x_i^\mu - x_{||\,i}^\mu)^2\,, \tag{1.5}$$

where μ indicates the different data points, M the number of data points, and I the dimensionality of the data vectors.

1.3 Reconstruction error and variance

The question now is how we can find the direction of the one-dimensional subspace that minimizes the reconstruction error. For that it is interesting to inspect more closely what happens as we rotate the subspace. Figure 1.2 illustrates the projection onto two different subspaces. Focus just on the one point $\boldsymbol{x}$

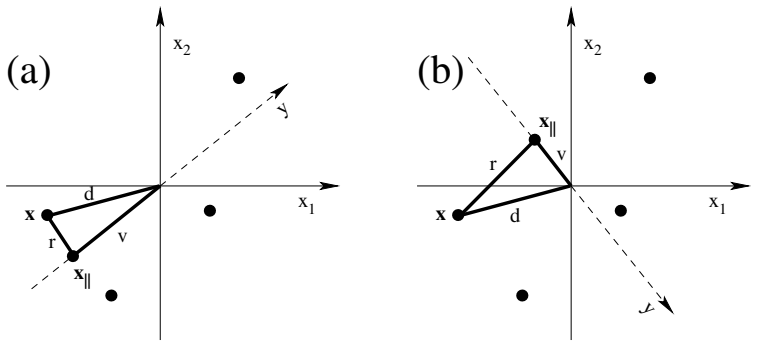


Figure 1.2: Variance of the projected data and reconstruction error as the linear subspace is rotated.
© CC BY-SA 4.0

and its projection $\boldsymbol{x}_{||}$. d is the distance of $\boldsymbol{x}$ from the origin, r is the distance of $\boldsymbol{x}$ from $\boldsymbol{x}_{||}$ in the subspace, and v is the distance of $\boldsymbol{x}_{||}$ from the origin. r and v depend on the direction of the subspace while d does not. Interestingly, since the triangles between $\boldsymbol{x}$, $\boldsymbol{x}_{||}$, and the origin are right-angled, r and v are related by Pythagoras' theorem, i.e. $r^2 + v^2 = d^2$. We know that r^2 contributes to the reconstruction error. v^2 on the other hand contributes to the variance of the projected data within the subspace. Thus we see that the sum over the reconstruction error plus the variance of the projected data is constant and equals the variance of the original data. Therefore, minimizing the reconstruction error is equivalent to maximizing the variance of the projected data.

1.4 Covariance matrix

How can we determine the direction of maximal variance? The first we can do is to determine the variances of the individual components. If the data points (or vectors) are written as $\boldsymbol{x} = (x_1, x_2)^T$ (T indicates transpose), then the variances of the first and second component can be written as $C_{11} := \langle x_1 x_1 \rangle$ and $C_{22} := \langle x_2 x_2 \rangle$, where angle brackets indicate averaging over all data points. Please remember that these are strictly speaking 2nd moments and not variances, but since we assume zero mean data that does not make a difference. If C_{11} is large compared to C_{22}, then the direction of maximal variance is close to $(1, 0)^T$, while if C_{11} is small, the direction of maximal variance is close to $(0, 1)^T$. (Notice that variance doesn't have a polarity, so that one could use the inverse vector $(-1, 0)^T$ instead of $(1, 0)^T$ equally well for indicating the direction of maximal variance.)

But what if C_{11} is of similar value as C_{22}, like in the example of Figure 1.1? Then the co-variance between the two components, $C_{12} := \langle x_1 x_2 \rangle$, can give us additional information (notice that $C_{21} := \langle x_2 x_1 \rangle$ is equal to C_{12}). A large positive value of C_{12} (large relative to $\sqrt{C_{11} C_{22}}$) indicates a strong correlation between x_1 and x_2 and that the data cloud is extended along the $(1, 1)^T$ direction. A negative value would indicate anti-correlation and an extension along the $(-1, 1)^T$ direction. A small value of C_{12} (again relative to $\sqrt{C_{11} C_{22}}$) would indicate no correlation and thus little structure of the data, i.e. no prominent direction of maximal variance. The variances and covariances are conveniently arranged in a matrix with components

$$\blacklozenge \quad C_{ij} := \langle x_i x_j \rangle \,, \tag{1.6}$$

which **is called covariance matrix (remember, assuming zero mean data)**. It can easily be shown that the components obey the relation

$$C_{ij}^2 \leq C_{ii} C_{jj} \,. \tag{1.7}$$

It is also easy to see that scaling the data by a factor α scales the covariance matrix by a factor α^2. Figure 1.3 shows several data clouds and the corresponding covariance matrices.

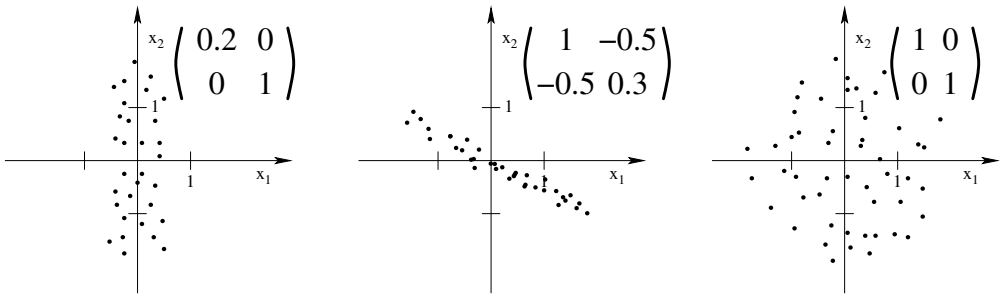


Figure 1.3: Several data distributions and their covariance matrices.
© CC BY-SA 4.0

1.5 Covariance matrix and higher order structure

Notice that **the covariance matrix only gives you information about the general extent of the data (the second order moments). It does not give you any information about detailed structure of the data cloud (the higher-order moments)**. Figure 1.4 shows different data distributions that all have the same covariance matrix. Thus as long as we consider only the covariance matrix, i.e. second order moments, we can always assume a Gaussian data distribution with an ellipsoid shape, because the covariance matrix does not represent any more structure in any case.

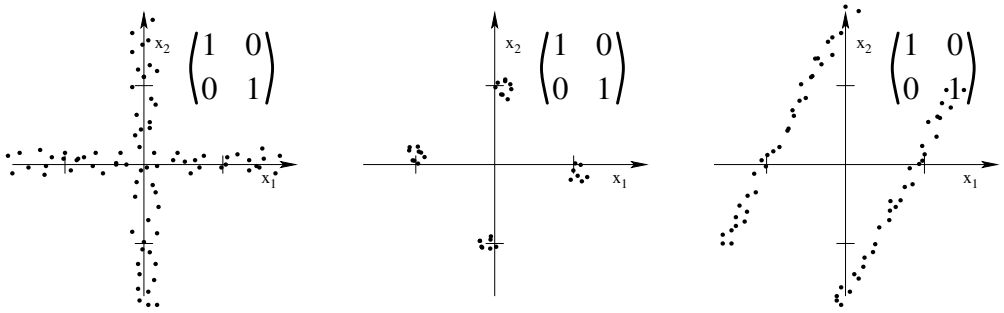


Figure 1.4: Different data distributions with identical covariance matrices. What is the scale of the axes in order to get the identity matrix?
© CC BY-SA 4.0

1.6 PCA by diagonalizing the covariance matrix

Now that we have learned that the covariance matrix in principle contains the information about the direction of maximal variance the question arises how we can get at this information. From Figure 1.3 (a) and (b) we can see that there are two fundamentally different situations: in (a) the data cloud is aligned with the axes of the coordinate system and the covariance matrix is diagonal; in (b) the data cloud is oblique to the axes and the matrix is not diagonal. In the former case the direction of maximal variance is simply the axis belonging to the largest value on the diagonal of the covariance matrix. In the latter case, we cannot directly say what the direction of maximal variance might be. Thus, since the case of a diagonal covariance matrix is so much simpler, the strategy we are going to take is to make a non-diagonal covariance matrix digonal by rotating the coordinate system accordingly. This is illustrated in Figure 1.5. From linear algebra we know that diagonalizing a matrix can be done by solving the corresponding eigenvalue equation. It will turn out that the eigenvectors of the covariance matrix point into the directions of maximal (and minimal) variance and that the eigenvalues are equal to the variances along these directions. Projecting the data onto the eigenvectors with largest eigenvalues is therefore the optimal linear dimensionality reduction.

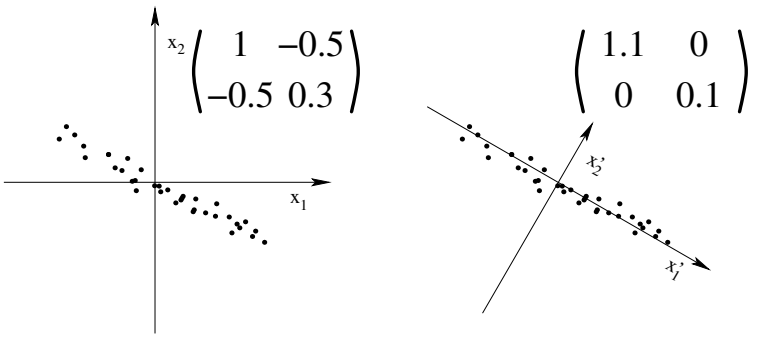


Figure 1.5: Diagonalizing the covariance matrix by rotating the coordinate system.
© CC BY-SA 4.0

1.7 Whitening

Whitening data means to first remove the mean and then stretch the data along the principal axes such that it has unit variance in all directions, i.e. the stretching must be done by one over the square root of the variance. Principal axes as well as the corresponding variances can be optained by PCA. If the original data is an unisotropic Gaussian to begin with, e.g. having the shape of a flying sausage or a cigar, it has a spherical shape after whitening, which is the reason to call it also sphering.

Whitened data has the advantage that you can project it onto any unit vector and it has unit variance. More formally, if $\boldsymbol{n}$ is an arbitrary unit vector and $\boldsymbol{x}^\mu$ are the whitened data vectors, the projected data $x_{||} := \boldsymbol{n}^T\boldsymbol{x}$ has zero mean $\langle x_{||}\rangle = 0$, since $\langle \boldsymbol{x}^\mu\rangle = \boldsymbol{0}$, and unit variance $\langle x_{||}^2\rangle = 1$. Projected onto two orthogonal unit vectors, the two projected data sets are uncorrelated. More formally, if $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are two arbitrary but orthogonal unit vectors, i.e. $\boldsymbol{n}_i^T\boldsymbol{n}_j = \delta_{ij}$ (Kronecker symbol), the two projected data sets $x_{||1} := \boldsymbol{n}_1^T\boldsymbol{x}$ and $x_{||2} := \boldsymbol{n}_2^T\boldsymbol{x}$ are uncorrelated, i.e. $\langle x_{||1}x_{||2}\rangle = 0$.

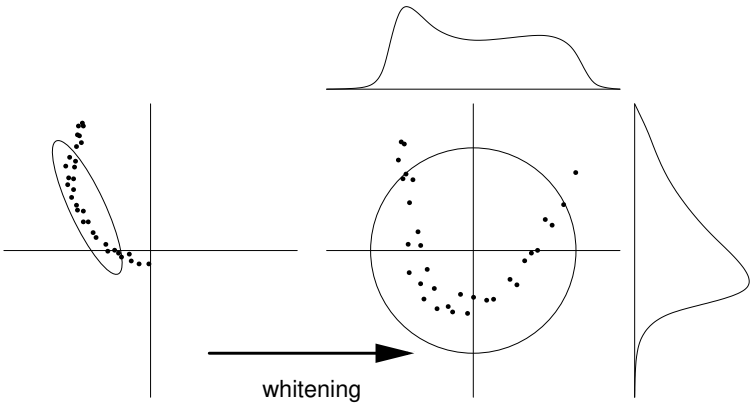


Figure 1.6: Whitening of some data. Left: Original data. The ellipse represents the second moment matrix of the data distribution. Right: The mean is removed and the data stretched such that it has variance one in all directions, indicated by a circle representing the identity second moment matrix. The graphs to the top and right represent the marginals of the first and second component, respectively.
© CC BY-SA 4.0

2 Formalism

Learning material:

■ Text below

Quizzes and Exercises: Deepen and test your understanding with the following

- Section 2 Exercises, Section 2 Solutions

This section is partly based on (Hertz et al., 1991).

2.1 Definition of the PCA-optimization problem

The problem of principal component analysis (PCA) can be formally stated as follows.

Principal Component Analysis (PCA): Given a set $\{\boldsymbol{x}^\mu : \mu = 1, ..., M\}$ of I-dimensional data points $\boldsymbol{x}^\mu = (x_1^\mu, x_2^\mu, ..., x_I^\mu)^T$ with zero mean, $\langle \boldsymbol{x}^\mu \rangle_\mu = \boldsymbol{0}_I$, find an orthogonal matrix $\boldsymbol{U}$ with determinant $|\boldsymbol{U}| = +1$ generating the transformed data points $\boldsymbol{x'}^\mu := \boldsymbol{U}^T \boldsymbol{x}^\mu$ such that for any given dimensionality P the data projected onto the first P axes, $\boldsymbol{x'}_{||}^\mu := (x'^\mu_1, x'^\mu_2, ..., x'^\mu_P, 0, ..., 0)^T$, have the smallest

$$\text{reconstruction error} \quad E := \langle \| \boldsymbol{x'}^\mu - \boldsymbol{x'}_{||}^\mu \|^2 \rangle_\mu \tag{2.1}$$

among all possible projections onto a P-dimensional subspace. The column vectors of matrix $\boldsymbol{U}$ define the new axes and are called the *principal components*.

Some remarks: (i) $\langle \boldsymbol{x}^\mu \rangle_\mu$ indicates the mean over all M data points indexed with μ. To simplify the notation we will from now on drop the index μ and indicate averages over the data points by $\langle \cdot \rangle$. (ii) If one has non-zero-mean data, one typically removes the mean before applying PCA. Even though all the math is valid also for non-zero-mean data, the results would typically be undesired and nonintuitive. (iii) Since matrix $\boldsymbol{U}$ is orthogonal and has determinant value $+1$, it corresponds simply to a rotation of the data $\boldsymbol{x}$. Thus, the 'shape' of the data cloud remains the same, just the 'perspective' changes. $|\boldsymbol{U}| = -1$ would imply a mirror reflection of the data distribution and is often permitted, too. Notice also that one can interpret the multiplication with matrix $\boldsymbol{U}^T$ either as a rotation of the data or as a rotation of the coordinate system. Either interpretation is valid. (iv) Projecting the data $\boldsymbol{x'}$ onto the P-dimensional linear subspace spanned by the first P axes is simply done by setting all components higher than P to zero. This can be done, because we still have an orthonormal coordinate system. If $\boldsymbol{U}$ and therefore the new coordinate system were not orthogonal then the projection became a mathematically more complex operation. (v) The reconstruction error has to be minimal for any P. This has the advantage that we do not need to decide on P before performing PCA. Often P is actually choosen based on information obtained during PCA and governed by a constraint, such as that the reconstruction error should be below a certain threshold.

2.2 Matrix $\mathbf{V}^T$: Mapping from high-dimensional old coordinate system to low-dimensional new coordinate system

Assume some data points $\boldsymbol{x}$ are given in an I-dimensional space and a linear subspace is spanned by P orthonormal vectors

$$\lozenge \quad \boldsymbol{v}_p := (v_{1p}, v_{2p}, ..., v_{Ip})^T \tag{2.2}$$

$$\lozenge \quad \text{with} \quad \boldsymbol{v}_p^T \boldsymbol{v}_q = \delta_{pq} := \begin{cases} 1 & \text{if } p = q \\ 0 & \text{otherwise} \end{cases} . \tag{2.3}$$

We will typically assume $P < I$ and speak of a high(I)-dimensional space and a low(P)-dimensional (sub)space. However, $P = I$ may be possible as a limiting case as well.

Arranging these vectors in a **matrix** yields

$$\blacklozenge \quad \boldsymbol{V} := (\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_P) \tag{2.4}$$

$$\overset{(2.2)}{=} \begin{pmatrix} v_{11} & v_{12} & ... & v_{1P} \\ v_{21} & v_{22} & ... & v_{2P} \\ \vdots & & \ddots & \vdots \\ v_{I1} & v_{I2} & ... & v_{IP} \end{pmatrix} . \tag{2.5}$$

This matrix **can be used to map the data points $\boldsymbol{x}$ into the subspace spanned by the vectors $\boldsymbol{v}_p$** yielding

$$\lozenge \quad \boldsymbol{y} := \boldsymbol{V}^T \boldsymbol{x} , \tag{2.6}$$

see also figure 2.1. If $P < I$ then the dimensionality is reduced and some information is lost; if $P = I$ all information is preserved. In any case the mapped data are now represented in a new coordinate system the axes of which are given by the vectors $\boldsymbol{v}_p$. With $P = 2$ and $I = 3$, for example, we have

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{v}_1^T \boldsymbol{x} \\ \boldsymbol{v}_2^T \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \end{pmatrix} \boldsymbol{x} = \boldsymbol{V}^T \boldsymbol{x}$$

$$\text{or} \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} v_{11} & v_{21} & v_{31} \\ v_{12} & v_{22} & v_{32} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \boldsymbol{V}^T \boldsymbol{x} .$$

Notice that $\boldsymbol{y}$ is P-dimensional while $\boldsymbol{x}$ is I-dimensional.

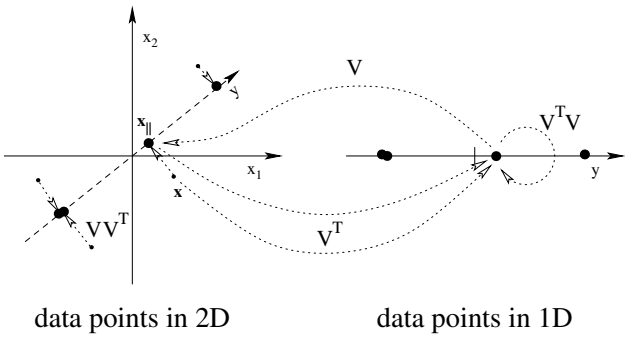


Figure 2.1: The effect of matrices $\boldsymbol{V}^T$ and $\boldsymbol{V}$ and combinations thereof for an example of a mapping from 2D to 1D.
© CC BY-SA 4.0

It is important to realize that we have done two things here: firstly, **we have moved the points from the high-dimensional space onto the low-dimensional subspace** (the points that were already in the subspace have not been moved, of course) **and** secondly, **we have represented the moved points in a new coordinate system** that is particularly suitable for the low-dimensional subspace. Thus, we went from the high-dimensional space and the old coordinate system to the low-dimensional subspace and a new coordinate system. Notice also that points in the high-dimensional space can generally not be represented accurately in the new coordinate system, because it does not have enough dimensions.

2.3 Matrix V: Mapping from low-dimensional new coordinate system to subspace in old coordinate system

Interestingly, since the vectors $\boldsymbol{v}_p$ are orthonormal, **matrix $\boldsymbol{V}$ can also be used to transform the points back from the new to the old coordinate system, although, the lost dimensions cannot be recovered**, of course. Thus the mapped points $\boldsymbol{y}$ in the new coordinate system become points $\boldsymbol{x}_{||}$ in the old coordinate system and are given by

$$\lozenge \quad \boldsymbol{x}_{||} := \boldsymbol{V}\boldsymbol{y} \tag{2.7}$$

$$\lozenge \quad \overset{(2.6)}{=} \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x}\,. \tag{2.8}$$

$\boldsymbol{y}$ and $\boldsymbol{x}_{||}$ are equivalent representations, i.e. they **contain the same information, just in different coordinate systems.**

2.4 Matrix ($\mathbf{V}^T\mathbf{V}$): Identity mapping within new coordinate system

Before we look at the combined matrix $\boldsymbol{V}\boldsymbol{V}^T$ consider $\boldsymbol{V}^T\boldsymbol{V}$. **The latter is** obviously **a $P \times P$-matrix and performs a transformation from the new (low-dimensional) coordinate system to the old (high-dimensional) coordinate system (2.7) and back again (2.6).** The back-transformation implies a dimensionality reduction, but **since all points** in the old coordinate system come from the new coordinate system and therefore **lie within the low-dimensional subspace already, the** mapping onto the low-dimensional space does not discard any information. Thus, only the back and forth (or rather forth and back) transformation between the two coordinate systems remains and that in **combination is without any effect** either. This means that **$\boldsymbol{V}^T\boldsymbol{V}$ is the identity matrix,** which can be easily verified

$$\left(\boldsymbol{V}^T\boldsymbol{V}\right)_{pq} = \boldsymbol{v}_p^T\boldsymbol{v}_q \overset{(2.3)}{=} \delta_{pq} \tag{2.9}$$

$$\Longleftrightarrow \quad \boldsymbol{V}^T\boldsymbol{V} = \mathbf{1}_P \tag{2.10}$$

with $\mathbf{1}_P$ indicating the identity matrix of dimensionality P. With $P = 2$, for example, we have

$$\boldsymbol{V}^T\boldsymbol{V} = \begin{pmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \end{pmatrix} (\boldsymbol{v}_1\boldsymbol{v}_2) = \begin{pmatrix} \boldsymbol{v}_1^T\boldsymbol{v}_1 & \boldsymbol{v}_1^T\boldsymbol{v}_2 \\ \boldsymbol{v}_2^T\boldsymbol{v}_1 & \boldsymbol{v}_2^T\boldsymbol{v}_2 \end{pmatrix} \overset{(2.3)}{=} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} .$$

2.5 Matrix ($\mathbf{V}\mathbf{V}^T$): Projection from high- to low-dimensional (sub)space within old coordinate system

As we have seen above (2.8) **the combined matrix $\boldsymbol{V}\boldsymbol{V}^T$ maps the points $\boldsymbol{x}$ onto the low-dimensional subspace** but in contrast to matrix $\boldsymbol{V}^T$ alone the mapped points are represented **within the old coordinate system** and not the new one. It turns out that **this is a projection operation with the characterizing property that** it does not make a difference whether you apply it once or twice, i.e. $\boldsymbol{P}\boldsymbol{P} = \boldsymbol{P}$. Let us therefore define the projection matrix

$$\blacklozenge \quad \boldsymbol{P} := \boldsymbol{V}\boldsymbol{V}^T \tag{2.11}$$

and verify that

$$\lozenge \quad \boldsymbol{P}\boldsymbol{P} \overset{(2.11)}{=} \boldsymbol{V}\underbrace{\boldsymbol{V}^T\boldsymbol{V}}_{\mathbf{1}_P}\boldsymbol{V}^T \tag{2.12}$$

$$\lozenge \quad \overset{(2.10)}{=} \boldsymbol{V}\boldsymbol{V}^T \tag{2.13}$$

$$\lozenge \quad \overset{(2.11)}{=} \boldsymbol{P}\,. \tag{2.14}$$

A closer look at $\boldsymbol{P}$ shows that

$$\lozenge \quad \boldsymbol{P} := \boldsymbol{V}\boldsymbol{V}^T \tag{2.15}$$

$$\lozenge \quad = (\boldsymbol{v}_1, ..., \boldsymbol{v}_P)\begin{pmatrix} \boldsymbol{v}_1^T \\ \vdots \\ \boldsymbol{v}_P^T \end{pmatrix} \tag{2.16}$$

$$\overset{(2.2)}{=} \begin{pmatrix} v_{11} & v_{12} & \cdots \\ v_{21} & \ddots & \\ v_{31} & \ddots & \\ \vdots & & v_{IP} \end{pmatrix}\begin{pmatrix} v_{11} & v_{21} & v_{31} & \cdots \\ v_{12} & \ddots & \ddots & \\ \vdots & & & v_{IP} \end{pmatrix} \tag{2.17}$$

$$= \begin{pmatrix} \sum_p v_{1p}v_{1p} & \sum_p v_{1p}v_{2p} & \cdots \\ \sum_p v_{2p}v_{1p} & \ddots & \\ \vdots & & \sum_p v_{Ip}v_{Ip} \end{pmatrix} \tag{2.18}$$

$$= \sum_{p=1}^{P}\begin{pmatrix} v_{1p}v_{1p} & v_{1p}v_{2p} & \cdots \\ v_{2p}v_{1p} & \ddots & \\ \vdots & & v_{Ip}v_{Ip} \end{pmatrix} \tag{2.19}$$

$$\lozenge \quad = \sum_{p=1}^{P}\boldsymbol{v}_p\boldsymbol{v}_p^T\,. \tag{2.20}$$

$\boldsymbol{P}$ is obviously an $I \times I$-matrix. **If $P = I$ then** projecting from the old to the new and back to the old coordinate system causes no information loss and $\boldsymbol{P} = \mathbf{1}_I$. **The smaller P the more information is lost**

and the more does $\boldsymbol{P}$ differ from the identity matrix. Consider, for example

$$\boldsymbol{v}_1 := \frac{1}{2}(\sqrt{2},-1,1)^T \quad \Rightarrow \quad \boldsymbol{v}_1\boldsymbol{v}_1^T = \frac{1}{4}\begin{pmatrix} 2 & -\sqrt{2} & \sqrt{2} \\ -\sqrt{2} & 1 & -1 \\ \sqrt{2} & -1 & 1 \end{pmatrix},$$

$$\boldsymbol{v}_2 := \frac{1}{2}(0,\sqrt{2},\sqrt{2})^T \quad \Rightarrow \quad \boldsymbol{v}_2\boldsymbol{v}_2^T = \frac{1}{4}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{pmatrix}, \text{ and}$$

$$\boldsymbol{v}_3 := \frac{1}{2}(-\sqrt{2},-1,1)^T \quad \Rightarrow \quad \boldsymbol{v}_3\boldsymbol{v}_3^T = \frac{1}{4}\begin{pmatrix} 2 & \sqrt{2} & -\sqrt{2} \\ \sqrt{2} & 1 & -1 \\ -\sqrt{2} & -1 & 1 \end{pmatrix}$$

for which you can easily verify that $\boldsymbol{P}$ (2.20) successively becomes the identity matrix as you take more of the $\boldsymbol{v}_p\boldsymbol{v}_p^T$-terms.

LECTURE 2/2

- Lecture 2 Exercises, Lecture 2 Solutions

2.6 Variance

The overall variance of a multi-dimensional data set is called ***total variance*** (D: totale Varianz) **is** defined as **the sum over the variances of its components. Since we assume zero-mean data, we have**

$$\blacklozenge \quad \mathrm{var}(\boldsymbol{x}) := \sum_{i=1}^{I} \langle x_i^2 \rangle \tag{2.21}$$

$$\lozenge = \left\langle \sum_{i=1}^{I} x_i^2 \right\rangle \tag{2.22}$$

$$\blacklozenge = \langle \boldsymbol{x}^T\boldsymbol{x} \rangle \tag{2.23}$$

This also holds for the projected data, of course, $\mathrm{var}(\boldsymbol{y}) = \langle \boldsymbol{y}^T\boldsymbol{y} \rangle$.

2.7 Reconstruction error

The reconstruction error E is defined as the mean square sum over the distances between the original data points $\boldsymbol{x}$ and the projected ones $\boldsymbol{x}_\parallel$. If we define the *orthogonal vectors* (D: Lotvektoren)

$$\lozenge \quad \boldsymbol{x}_\perp = \boldsymbol{x} - \boldsymbol{x}_\parallel \tag{2.24}$$

(in contrast to the projected vectors $\boldsymbol{x}_\parallel$) we can write **the reconstruction error** as the variance of the orthogonal vectors and find

$$\begin{aligned}
\blacklozenge|\lozenge \quad E &\overset{(2.1,2.24)}{=} \langle {\boldsymbol{x}_\perp}^T \boldsymbol{x}_\perp \rangle && (2.25)\\
&\overset{(2.24)}{=} \langle (\boldsymbol{x} - \boldsymbol{x}_\parallel)^T (\boldsymbol{x} - \boldsymbol{x}_\parallel) \rangle && (2.26)\\
\lozenge \quad &\overset{(2.8)}{=} \langle (\boldsymbol{x} - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x})^T (\boldsymbol{x} - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x}) \rangle && (2.27)\\
\lozenge \quad &= \langle \boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{V}\underbrace{(\boldsymbol{V}^T\boldsymbol{V})}_{=\mathbf{1}_P}\boldsymbol{V}^T\boldsymbol{x} \rangle && (2.28)\\
\lozenge \quad &\overset{(2.10)}{=} \langle \boldsymbol{x}^T\boldsymbol{x} \rangle - \langle \boldsymbol{x}^T\boldsymbol{V}\underbrace{(\boldsymbol{V}^T\boldsymbol{V})}_{=\mathbf{1}_P}\boldsymbol{V}^T\boldsymbol{x} \rangle && (2.29)\\
\lozenge \quad &\overset{(2.8)}{=} \langle \boldsymbol{x}^T\boldsymbol{x} \rangle - \langle {\boldsymbol{x}_\parallel}^T \boldsymbol{x}_\parallel \rangle && (2.30)\\
\blacklozenge \quad &\overset{(2.29,2.6)}{=} \langle \boldsymbol{x}^T\boldsymbol{x} \rangle - \langle \boldsymbol{y}^T\boldsymbol{y} \rangle \,. && (2.31)
\end{aligned}$$

This means that the reconstruction error **equals the difference between the variance of the data minus the variance of the projected data**. Thus, this verifies our intuition that minimizing the reconstruction error is equivalent to maximizing the variance of the projected data.

2.8 Covariance matrix

We have already argued heuristically that the covariance matrix $\boldsymbol{C}_x$ with $C_{xij} := \langle x_i x_j \rangle$ plays an important role in performing PCA. It is convenient to write the **covariance matrix in vector notation**:

$$\blacklozenge \quad \boldsymbol{C}_x := \langle \boldsymbol{x}\boldsymbol{x}^T \rangle \quad = \quad \frac{1}{M}\sum_\mu \boldsymbol{x}^\mu \boldsymbol{x}^{\mu T} \,. \tag{2.32}$$

It is an easy exercise to show that this definition is equivalent to the componentwise one given above. Since $(\boldsymbol{x}\boldsymbol{x}^T)^T = \boldsymbol{x}\boldsymbol{x}^T$ (remember $(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T\boldsymbol{A}^T$), one can also see that $\boldsymbol{C}_x$ **is symmetric**, i.e. $\boldsymbol{C}_x^T = \boldsymbol{C}_x$.

Keep in mind that $\boldsymbol{C}_x$ is strictly speaking a 2nd moment matrix and only identical to the covariance matrix since we assume zero mean data.

2.9 Eigenvalue equation of the covariance matrix

Since the covariance matrix is symmetric, its eigenvalues are real and a set of orthogonal eigenvectors always exists. In mathematical terms, **for a given covariance matrix $\boldsymbol{C}_x$ we can always find a complete set of real eigenvalues λ_i and corresponding eigenvectors $\boldsymbol{u}_i$ such that**

$$\begin{aligned}
\lozenge \quad \boldsymbol{C}_x \boldsymbol{u}_i &= \boldsymbol{u}_i \lambda_i \quad \text{(eigenvalue equation)}\,, && (2.33)\\
\blacklozenge \quad \lambda_i &\geq \lambda_{i+1} \quad \text{(eigenvalues are ordered)}\,, && (2.34)\\
\lozenge \quad \boldsymbol{u}_i^T \boldsymbol{u}_j &= \delta_{ij} \quad \text{(eigenvectors are orthonormal)}\,. && (2.35)
\end{aligned}$$

If we combine the eigenvectors into an orthogonal matrix $\boldsymbol{U}$ and the eigenvalues into a diagonal matrix $\boldsymbol{\Lambda}$,

$$\begin{aligned}
\blacklozenge \quad \boldsymbol{U} &:= (\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_I)\,, && (2.36)\\
\blacklozenge \quad \boldsymbol{\Lambda} &:= \text{diag}(\lambda_1, \lambda_2, ..., \lambda_I)\,, && (2.37)
\end{aligned}$$

then we can rewrite (2.35) and (2.33) as

$$\begin{aligned}
\blacklozenge \quad \boldsymbol{U}^T\boldsymbol{U} &\overset{(2.35,2.36)}{=} \boldsymbol{1}_I \quad \text{(matrix } \boldsymbol{U} \text{ is orthogonal)}\,, && (2.38)\\
\lozenge \quad \Longleftrightarrow \quad \boldsymbol{U}\boldsymbol{U}^T &= \boldsymbol{1}_I \quad \text{(since } \boldsymbol{U}^{-1} = \boldsymbol{U}^T \text{ and } \boldsymbol{U} \text{ is quadratic)}\,, && (2.39)\\
\blacklozenge \quad \boldsymbol{C}_x\boldsymbol{U} &\overset{(2.33,2.36,2.37)}{=} \boldsymbol{U}\boldsymbol{\Lambda} \quad \text{(eigenvalue equation)}\,, && (2.40)\\
\lozenge \quad \overset{(2.38)}{\Longleftrightarrow} \boldsymbol{U}^T\boldsymbol{C}_x\boldsymbol{U} &= \boldsymbol{\Lambda} && (2.41)\\
\lozenge \quad \overset{(2.38,2.39)}{\Longleftrightarrow} \boldsymbol{C}_x &= \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T\,. && (2.42)
\end{aligned}$$

2.10 Total variance of the data x

Given the eigenvector matrix $\boldsymbol{U}$ and the eigenvalue matrix $\boldsymbol{\Lambda}$ it is easy to compute the total variance of the data

$$\begin{aligned}
\blacklozenge|\lozenge \quad \langle \boldsymbol{x}^T\boldsymbol{x} \rangle &= \langle \text{tr}(\boldsymbol{x}^T\boldsymbol{x}) \rangle \quad (\text{since } s = \text{tr}(s) \text{ for any scalar } s) && (2.43)\\
\lozenge \quad &= \langle \text{tr}(\boldsymbol{x}\boldsymbol{x}^T) \rangle \quad (\text{since } \text{tr}(\boldsymbol{A}\boldsymbol{B}) = \text{tr}(\boldsymbol{B}\boldsymbol{A}) \text{ for any matrices } \boldsymbol{A}, \boldsymbol{B}) && (2.44)\\
\lozenge \quad &= \text{tr}(\langle \boldsymbol{x}\boldsymbol{x}^T \rangle) \quad (\text{since tr}(\cdot) \text{ and } \langle\cdot\rangle \text{ commute}) && (2.45)\\
\lozenge \quad &\overset{(2.32)}{=} \text{tr}(\boldsymbol{C}_x) && (2.46)\\
\lozenge \quad &\overset{(2.39)}{=} \text{tr}(\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{C}_x) && (2.47)\\
\lozenge \quad &= \text{tr}(\boldsymbol{U}^T\boldsymbol{C}_x\boldsymbol{U}) && (2.48)\\
\lozenge \quad &\overset{(2.41)}{=} \text{tr}(\boldsymbol{\Lambda}) && (2.49)\\
\blacklozenge \quad &\overset{(2.37)}{=} \sum_i \lambda_i\,. && (2.50)
\end{aligned}$$

Thus **the total variance of the data is simply the sum of the eigenvalues of its covariance matrix**.

Notice that on the way of this proof we have shown some very general properties. From line (2.43) to (2.46) we have shown that the total variance of some multi-dimensional data equals the trace of its covariance matrix. From line (2.46) to (2.48) we have shown that the trace remains invariant under any orthogonal transformation of the coordinate system. This implies that the total variance of some multi-dimensional data is invariant under any orthogonal transformation such as a rotation. This is intuitively clear.

2.11 Diagonalizing the covariance matrix

We can now use matrix $\boldsymbol{U}$ to **transform the data such that the covariance matrix becomes diagonal.** Define $\boldsymbol{x}' := \boldsymbol{U}^T\boldsymbol{x}$ and denote the new covariance matrix by $\boldsymbol{C}'_x$. We have

$$\begin{aligned}
\blacklozenge \quad \boldsymbol{x}' &:= \boldsymbol{U}^T\boldsymbol{x} && (2.51)\\
\blacklozenge|\lozenge \quad \boldsymbol{C}'_x &:= \left\langle \boldsymbol{x}'\boldsymbol{x}'^T \right\rangle && (2.52)\\
\lozenge &\overset{(2.51)}{=} \left\langle (\boldsymbol{U}^T\boldsymbol{x})(\boldsymbol{U}^T\boldsymbol{x})^T \right\rangle && (2.53)\\
\lozenge &= \boldsymbol{U}^T \left\langle \boldsymbol{x}\boldsymbol{x}^T \right\rangle \boldsymbol{U} && (2.54)\\
\lozenge &\overset{(2.32)}{=} \boldsymbol{U}^T\boldsymbol{C}_x\boldsymbol{U}\,, && (2.55)\\
\blacklozenge &\overset{(2.41)}{=} \boldsymbol{\Lambda} && (2.56)
\end{aligned}$$

and find that the transformed data $\boldsymbol{x}'$ have a diagonal covariance matrix. Working with $\boldsymbol{x}'$ instead of $\boldsymbol{x}$ will simplify further analysis without loss of generality.

2.12 Variance of y for a diagonalized covariance matrix

Now that we have the data represented in a coordinate system in which the covariance matrix is diagonal, we can try to answer the question, which is the P-dimensional subspace that minimizes the reconstruction error. Our intuition would predict that it is simply the space spanned by the first P eigenvectors. To show this analytically, we **take an arbitrary set of P orthonormal vectors $\boldsymbol{v}'_p$, and with $\boldsymbol{V}' := (\boldsymbol{v}'_1, \boldsymbol{v}'_2, ..., \boldsymbol{v}'_P)$** we **compute** the variance of $\boldsymbol{y}$.

$$\begin{aligned}
\lozenge \quad \boldsymbol{y} &:= \boldsymbol{V}'^T\boldsymbol{x}' && (2.57)\\
\lozenge \implies \langle \boldsymbol{y}^T\boldsymbol{y} \rangle &\overset{(2.57)}{=} \langle \boldsymbol{x}'^T\boldsymbol{V}'\boldsymbol{V}'^T\boldsymbol{x}' \rangle && (2.58)\\
&= \langle \mathrm{tr}(\boldsymbol{x}'^T\boldsymbol{V}'\boldsymbol{V}'^T\boldsymbol{x}') \rangle \quad (\text{since } s = \mathrm{tr}(s) \text{ for any scalar } s) && (2.59)\\
&= \langle \mathrm{tr}(\boldsymbol{V}'^T\boldsymbol{x}'\boldsymbol{x}'^T\boldsymbol{V}') \rangle \quad (\text{since } \mathrm{tr}(\boldsymbol{ABC}) = \mathrm{tr}(\boldsymbol{BCA}) \text{ if defined}) && (2.60)\\
\lozenge &\overset{(2.52)}{=} \mathrm{tr}(\boldsymbol{V}'^T\boldsymbol{C}'_x\boldsymbol{V}') \quad (\text{since } \mathrm{tr}(\cdot) \text{ and } \langle\cdot\rangle \text{ commute}) && (2.61)\\
\lozenge &\overset{(2.56)}{=} \mathrm{tr}(\boldsymbol{V}'^T\boldsymbol{\Lambda}\boldsymbol{V}') && (2.62)\\
\lozenge &= \sum_i \lambda_i \sum_p (v'_{ip})^2\,. \quad (\text{as one can work out on a sheet of paper}) && (2.63)
\end{aligned}$$

$\boldsymbol{V}'$ is a rectangular $I \times P$-matrix and projects the data into a lower-dimensional space. Since its columns are orthonormal to each other, it only performes a rotation and elimination of some dimensions (not necessarily along the axes) but no scaling of any sort or distortion beyond the elimnation of dimensions.

2.13 Constraints of matrix V′

Notice that, **since the vectors v'_p are orthonormal, $\boldsymbol{V}'$** can always be completed to an orthogonal $I \times I$-matrix by adding $I - P$ additional orthonormal vectors. Since we know that an orthogonal matrix has normalized row as well as column vectors, we see that, by taking away the $I - P$ additional column vectors, we are left with the constraints

$$\lozenge \quad \sum_i (v'_{ip})^2 = 1 \quad \text{(column vectors of } \boldsymbol{V}' \text{ have norm one)}\,, \tag{2.64}$$

$$\lozenge \implies \sum_{ip} (v'_{ip})^2 = P \quad \text{(square sum over all matrix elements equals } P)\,, \tag{2.65}$$

$$\lozenge \quad \sum_p (v'_{ip})^2 \leq 1 \quad \text{(row vectors of } \boldsymbol{V}' \text{ have norm less or equal one)}\,. \tag{2.66}$$

Notice that Constraint (2.65) is a direct consequence of Constraint (2.64) and does not need to be verified separately in the following considerations.

2.14 Finding the optimal subspace

Since the variance (2.63) of $\boldsymbol{y}$ as well as the constraints (2.64, 2.65, 2.66) of Matrix $\boldsymbol{V}'$ are linear in $(v'_{ip})^2$, maximization of the variance $\langle \boldsymbol{y}^T \boldsymbol{y} \rangle$ is obviously achieved by putting as much 'weight' as possible on the large eigenvalues, which are the first ones. The simplest way of doing that is to set

$$\lozenge \quad v'_{ip} := \delta_{ip} := \begin{cases} 1 & \text{if } i = p \\ 0 & \text{otherwise} \end{cases}\,, \tag{2.67}$$

with the Kronecker symbol δ_{ip}.

Since $I \geq P$ we can verify the constraints

$$\sum_p (v'_{ip})^2 \overset{(2.67)}{=} \sum_p \delta_{ip}^2 = \left\{ \begin{array}{ll} 1 & \text{if } i \leq P \\ 0 & \text{otherwise} \end{array} \right\} \leq 1\,, \tag{2.68}$$

$$\sum_i (v'_{ip})^2 \overset{(2.67)}{=} \sum_i \delta_{ip}^2 = \delta_{pp}^2 = 1\,, \tag{2.69}$$

and see from (2.68) that there is actually as much 'weight' on the first, i.e. large, eigenvalues as Constraint (2.66) permits.

2.15 Interpretation of the result

What does it mean to set $v'_{ip} := \delta_{ip}$? **It means that $\boldsymbol{V}'$ projects the data $\boldsymbol{x}'$ onto the first P axes, which in fact is a projection onto the first P eigenvectors of the covariance matrix $\boldsymbol{C}_x$.** Thus, **if we define**

$$\blacklozenge|\lozenge \quad \boldsymbol{V} := \boldsymbol{U}\boldsymbol{V}' \tag{2.70}$$

$$\blacklozenge \overset{(2.36,\ 2.67)}{=} (\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_P) \tag{2.71}$$

we can go back to the original coordinate system and **find**

$$\blacklozenge|\lozenge \quad \boldsymbol{y} \overset{(2.57)}{=} \boldsymbol{V'}^T \boldsymbol{x}' \tag{2.72}$$

$$\lozenge \overset{(2.51)}{=} \boldsymbol{V'}^T \boldsymbol{U}^T \boldsymbol{x} \tag{2.73}$$

$$\blacklozenge \overset{(2.70)}{=} \boldsymbol{V}^T \boldsymbol{x}\,, \tag{2.74}$$

which we know has maximal variance. Thus, if we start from the original data $\boldsymbol{x}$ we would set $\boldsymbol{v}_p := \boldsymbol{u}_p$. The variance of $\boldsymbol{y}$ is

$$\blacklozenge|\lozenge \quad \langle \boldsymbol{y}^T \boldsymbol{y} \rangle \overset{(2.63)}{=} \sum_{i=1}^{I} \lambda_i \sum_{p=1}^{P} (v'_{ip})^2 \tag{2.75}$$

$$\lozenge \quad \overset{(2.67)}{=} \sum_{i=1}^{I} \lambda_i \sum_{p=1}^{P} \delta_{ip}^2 \tag{2.76}$$

$$\blacklozenge \quad = \sum_{i=1}^{P} \lambda_i \,, \tag{2.77}$$

which is the sum over the first P largest eigenvalues of the covariance matrix. Likewise one can determine the reconstruction error as

$$\blacklozenge|\lozenge \quad E \overset{(2.31)}{=} \langle \boldsymbol{x}^T \boldsymbol{x} \rangle - \langle \boldsymbol{y}^T \boldsymbol{y} \rangle \tag{2.78}$$

$$\lozenge \quad \overset{(2.50,\ 2.77)}{=} \sum_{i=1}^{I} \lambda_i - \sum_{j=1}^{P} \lambda_j \tag{2.79}$$

$$\blacklozenge \quad = \sum_{i=P+1}^{I} \lambda_i \,. \tag{2.80}$$

Notice that this is just one optimal set of weights. We have seen above that the projected data, like any multi-dimensional data, can be rotated arbitrarily without changing its variance and therefore without changing its reconstruction error. This is equivalent to a rotation of the projection vectors $\boldsymbol{v}_p$ within the space spanned by the first eigenvectors.

2.16 PCA Algorithm

2.17 Intuition of the Results

Eigenvalue spectrum.

Projection onto a low-dimensional eigenspace.

Visualization of eigenvectors.

2.18 Whitening or sphering

Sometimes it is desirable **to transform a data set such that it has variance one in all directions**. Such a normalization operation **is called *whitening* or *sphering***, see Fig. 1.6. The latter term is quite intuitive, because a spherical data distribution has the same variance in all directions. Intuitively speaking sphering requires to stretch and compress the data distribution along the axes of the principal components such they have variance one. Technically speaking one first rotates the data into a coordinate system where the covariance matrix is diagonal, then performs the stretching along the axes, and then rotates the data back into the original coordinate system. Principal component analysis obviously gives all the required information. The eigenvectors of the covariance matrix provide the axes of the new coordinate system and the eigenvalues λ_i indicate the variances and therefore how much one has to stretch the data. If the original variance is λ_i then one obviously has to stretch by a factor of $1/\sqrt{\lambda_i}$ to get variance one. Thus, **sphering is achieved by multiplying the data with a sphering matrix**

$$\blacklozenge \quad \boldsymbol{W} := \boldsymbol{U}\,\mathrm{diag}\left(\frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}}, ..., \frac{1}{\sqrt{\lambda_I}}\right)\boldsymbol{U}^T \tag{2.81}$$

$$\blacklozenge \quad \hat{\boldsymbol{x}} := \boldsymbol{W}\boldsymbol{x}\,. \tag{2.82}$$

If the final orientation of the data does not matter, the sphering matrix is often defined without the first $\boldsymbol{U}$. It is easy to verify that **the sphering matrix is symmetrical, the sphered data $\hat{\boldsymbol{x}}$ have a unit covariance matrix**,

$$\begin{aligned}
\boldsymbol{C}_{\hat{\boldsymbol{x}}} &:= \langle \hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T \rangle && (2.83)\\
&\overset{(2.82)}{=} \boldsymbol{W}\langle \boldsymbol{x}\boldsymbol{x}^T \rangle \boldsymbol{W}^T && (2.84)\\
&\overset{(2.32,2.81)}{=} \boldsymbol{U}\,\mathrm{diag}\left(\frac{1}{\sqrt{\lambda_i}}\right)\boldsymbol{U}^T\boldsymbol{C}_{\boldsymbol{x}}\boldsymbol{U}\,\mathrm{diag}\left(\frac{1}{\sqrt{\lambda_i}}\right)\boldsymbol{U}^T && (2.85)\\
&\overset{(2.41)}{=} \boldsymbol{U}\,\mathrm{diag}\left(\frac{1}{\sqrt{\lambda_i}}\right)\boldsymbol{\Lambda}\,\mathrm{diag}\left(\frac{1}{\sqrt{\lambda_i}}\right)\boldsymbol{U}^T && (2.86)\\
&\overset{(2.37)}{=} \boldsymbol{U}\,\mathbf{1}\,\boldsymbol{U}^T && (2.87)\\
&\overset{(2.39)}{=} \mathbf{1}\,, && (2.88)
\end{aligned}$$

and they have variance one in all directions, since for any projection vector $\boldsymbol{n}$ of norm one the variance $\langle(\boldsymbol{n}^T\hat{\boldsymbol{x}})^2\rangle$ of the projected data is

$$\begin{aligned}
\langle(\boldsymbol{n}^T\hat{\boldsymbol{x}})^2\rangle &= \boldsymbol{n}^T\langle\hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T\rangle\boldsymbol{n} && (2.89)\\
&\overset{(2.88)}{=} \boldsymbol{n}^T\boldsymbol{n} && (2.90)\\
&= 1 && (2.91)
\end{aligned}$$

Similarly one can show that **the sphered data projected onto two orthogonal vectors are uncorrelated**.

2.19 Singular value decomposition +

Sometimes one has fewer data points than dimensions. For instance one might have 100 images with 10000 pixels each. Then doing direct PCA is very inefficient and the following method, known as *singular value decomposition* (SVD), is helpful.

Let $\boldsymbol{x}^\mu, \mu = 1, ..., M$ be the I-dimensional data with $M < I$. For convenience we write the data in one $I \times M$-matrix

$$\boldsymbol{X} \quad := \quad (\boldsymbol{x}^1, ..., \boldsymbol{x}^M) \,. \tag{2.92}$$

The second-moment matrix can then be written as

$$\boldsymbol{C}_1 \quad := \quad \boldsymbol{X}\boldsymbol{X}^T/M \,, \tag{2.93}$$

and its eigenvalue equation and decomposition read

$$\boldsymbol{C}_1\boldsymbol{U}_1 \quad = \quad \boldsymbol{U}_1\boldsymbol{\Lambda}_1 \tag{2.94}$$

$$\iff \quad \boldsymbol{C}_1 \quad = \quad \boldsymbol{U}_1\boldsymbol{\Lambda}_1\boldsymbol{U}_1^T \,. \tag{2.95}$$

The data represented in the coordinate system of the eigenvectors is

$$\boldsymbol{Y}_1 \quad := \quad \boldsymbol{U}_1^T\boldsymbol{X} \,, \tag{2.96}$$

which is still high-dimensional.

Now interpret the data matrix $\boldsymbol{X}$ transposed, i.e. swap the data point index for the dimension index. In our example this would correspond to having 10000 data points in a 100-dimensional space, which is, of course, much easier to deal with. We get the same equations as above just with $\boldsymbol{X}$ transposed.

$$\boldsymbol{C}_2 \quad := \quad \boldsymbol{X}^T\boldsymbol{X}/I \,, \tag{2.97}$$

$$\boldsymbol{C}_2\boldsymbol{U}_2 \quad = \quad \boldsymbol{U}_2\boldsymbol{\Lambda}_2 \tag{2.98}$$

$$\iff \quad \boldsymbol{C}_2 \quad = \quad \boldsymbol{U}_2\boldsymbol{\Lambda}_2\boldsymbol{U}_2^T \,, \tag{2.99}$$

$$\boldsymbol{Y}_2 \quad := \quad \boldsymbol{U}_2^T\boldsymbol{X}^T \,. \tag{2.100}$$

The interesting property of matrix $\boldsymbol{Y}_2$ now is that its rows are eigenvectors of matrix $\boldsymbol{C}_1$, as can be shown easily.

$$\boldsymbol{C}_1\boldsymbol{Y}_2^T \overset{(2.93,2.100)}{=} (\boldsymbol{X}\boldsymbol{X}^T/M)(\boldsymbol{X}\boldsymbol{U}_2) \tag{2.101}$$

$$= \quad \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X}/I)\boldsymbol{U}_2 I/M \tag{2.102}$$

$$\overset{(2.97)}{=} \quad \boldsymbol{X}\boldsymbol{C}_2\boldsymbol{U}_2 I/M \tag{2.103}$$

$$\overset{(2.98)}{=} \quad \boldsymbol{X}\boldsymbol{U}_2\boldsymbol{\Lambda}_2 I/M \tag{2.104}$$

$$\overset{(2.100)}{=} \quad \boldsymbol{Y}_2^T\boldsymbol{\Lambda}_2 I/M \,. \tag{2.105}$$

The corresponding eigenvalues are eigenvalues of $\boldsymbol{C}_2$ scaled by I/M.

However, $\boldsymbol{Y}_2$ yields only M eigenvectors and eigenvalues. The other eigenvalues are all zero, because M data points can only produce M non-zero variance dimensions or, in other words, M data points together with the origin can only span an M-dimensional subspace. The missing $(I - M)$ eigenvectors must all be orthogonal to the first M ones and orthogonal to each other but can otherwise be quite arbitrary, since their eigenvalues are all equal. A Gram-Schmidt orthogonalization procedure can be used to generate them.

Acknowledgment: I thank Agnieszka Grabska-Barwinska for working out the proof for singular value decomposition.

3 Application +

Learning material:

- ☐ 2 min video 3.1 Face Processing
- ☐ 14 min video PCA 10: eigen-faces by Victor Lavrenko on YouTube
- ☐ Text below

3.1 Face processing

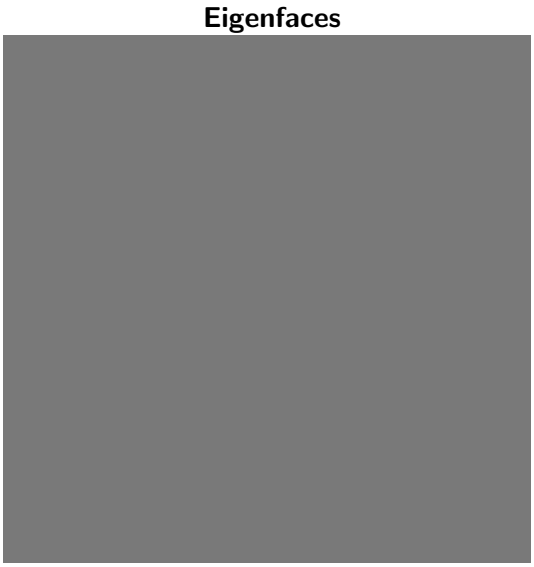


Figure: (http://www-white.media.mit.edu/ 2002-12-17 [2024-10-04, URL is outdated])

2/4

If one carefully shifts and scales many face images such that the eyes are in register, i.e. at identical positions, and performs PCA (or SVD) on them, meaningful eigenvectors can be calculated and are called eigenfaces. These are the principal grey value variations that distinguish faces from each other. The figure shows first the mean and then the eigenfaces ordered in rows. The first eigenface obviously accounts for the mean grey value, the second one for the difference in color between hair and face, the third one for illumination from the side, the fourth and seventh one at least partially for a beard. The higher components become increasingly difficult to interpret. The projection of face images onto the first eigenfaces is a suitable representation for face recognition, (cf. Turk and Pentland, 1991).

Figure: (http://www-white.media.mit.edu/ 2002-12-17 [2024-10-04, URL is outdated])

Eigenfaces - Texture



Figure: (Lanitis, Taylor, et al., 1995, Image and Vision Computing 13(5):393–401, Fig. 11, URL)

3/4

In this example many faces were warped to a standard geometry and then PCA was applied to calculate the eigenvectors, which are called eigenfaces. The Figure visualizes the first four eigenfaces by varying the average face (middle) along the direction of the eigenvectors by up to ± 3 standard deviations. The first eigenface (mode) accounts for overall illumination, the other three for some combination of beard, gender, and mimic expression.
Figure: (Lanitis et al., 1995, Fig. 11, URL)[3.1]

Eigenfaces - Shape

Figure: (Lanitis, Taylor, et al., 1995, Image and Vision Computing 13(5):393–401, Fig. 6, URL)

4/4

In this example PCA was applied to the geometry of faces. A graph with some standard structure was mapped onto many faces and the concatenated vector of xy-positions of the nodes (not shown in the graphs) of the graphs serve as the data for PCA. Visualization is done relative to the average geometry (middle) by adding the eigenvectors up to ± 2 standard deviations. The first three components mainly account for the orientation of the head along the three rotational axes. The fourth component accounts for some variation in width and mimic expression.
Figure: (Lanitis et al., 1995, Fig. 6, URL)[3.2]

A Other resources +

Numbers in square brackets indicate sections of these lecture notes to which the corresponding item is related.

A.1 Written material

- PCA in Wikipedia
  https://en.wikipedia.org/wiki/Principal_component_analysis
- A tutorial by Jonathon Shlens
  https://arxiv.org/pdf/1404.1100v1

A.2 Visualizations

- 2D, 3D, and 17D examples of PCA
  https://setosa.io/ev/principal-component-analysis/

A.3 Videos

- Abstract conceptual introduction to PCA from Georgia Tech.
  1. Part 1
     https://www.youtube.com/watch?v=kw9R0nD69OU (4:22) [1.1, 1.2, 1.3]
  2. Part 2
     https://www.youtube.com/watch?v=_nZUhV-qhZA (5:27) [1.2, 1.3]
  3. Part 3
     https://www.youtube.com/watch?v=kuzJJgPBrqc (5:01) [1.2, 1.3]
- Introduction to PCA with several practical examples by Rasmus Bro from the University of Copenhagen
  Note: In these videos the principal components are called *loadings*, and the projected values are called *scores*.
  1. Conceptual introduction
     https://www.youtube.com/watch?v=UUxIXU_Ob6E (12:32)
     - 00:00–00:38 Introduction
     - 00:38–07:52 Examples of multi-variate data
     - 07:52–12:32 Concept of PCA and bi-plot by means of a simple 3D example
  2. Example: Analysis of demographic data of many countries in the world with PCA
     https://www.youtube.com/watch?v=26YhtSJi1qc (11:36)
     - 00:11–02:26 Outlier detection
     - 02:26–05:47 Loadings help interpreting the projected data dimensions
     - 05:47–08:11 Exploration after removing outliers
     - 08:11–09:40 Using external information as markers
     - 09:40–10:36 From plots to numbers
     - 10:36–11:36 Summary

3. A bit more in depth discussion
   https://www.youtube.com/watch?v=2s-a62zSWL4 (14:31)
   - 00:00–00:50 Introduction and historical remarks
   - 00:50–03:18 Basic equation for data generation
   - 03:18–06:32 Example: Three-variable data set
     * 04:30–05:40 Removing the mean before PCA
     * 05:40–06:11 PCA as variations along loadings
     * 06:11–06:32 PCA as a rotation and truncation
   - 06:32–06:59 Scores are projections of the data onto the loadings
   - 06:59–14:26 PCA as finding common profiles and profile variations
     * 07:12–11:58 Example 1: Continuous spectral data of sugar samples
     * 12:22–14:26 Example 2: Discrete physical and chemical measurements of the same sugar samples
       · 12:48–14:26 Scaling the variables before PCA (covariance- vs. correlation-matrix)
4. Continuation of the previous video
   https://www.youtube.com/watch?v=sRsdF3rcAJc (08:22)
   - 00:05–14:26 PCA as finding common profiles and profile variations (cont.)
     * 00:05–04:02 Example 2: Discrete physical and chemical measurements of the same sugar samples (cont.)
     * 01:31–02:37 Relating results from Examples 1 and 2
     * 02:37–04:02 Bi-plot: Understanding the data by visualizing the loadings
   - 04:02–08:18 PCA reviewed

- Introduction to PCA in Geoscience by Matthew E. Clapham from the UC Santa Cruz
  https://www.youtube.com/watch?v=TSYL-oHx4T0 (15:27)
  - 00:00–00:33 Introduction
  - 00:33–01:17 Data is often multi-variate / high-dimensional
  - 01:17–02:50 Covariation allows to reduce dimensionality and remove redundancy
  - 02:50–03:36 skip (Indirect gradient analysis)
  - 03:36–04:44 skip (Types of ordination methods)
  - 04:44–06:10 Idea of PCA
  - 06:10–06:29 Covariance matrix
  - 06:29–09:22 Eigenvectors define a new coordinate system ordered by variance
    * 07:53–08:43 Eigenvalues measure the amount of variance along the axes of the new coordinate system
  - 09:22–10:42 Loadings
  - 10:42–12:17 Covariance- vs. correlation-matrix (scaling the variables before PCA)
  - 12:17–13:50 Using the eigenvalue spectrum to decide how many PCs to keep
  - 13:50–15:27 When is PCA applicable?

- Three lectures on singular value decomposition and PCA
  (See also the closely related tutorial by Jonathon Shlens https://arxiv.org/pdf/1404.1100v1.)

1. Singular value decomposition
   https://www.youtube.com/watch?v=EokL7E6o1AE (44:35)
   Explains singular value decomposition in general, which is quite an interesting fact in linear algebra, but not really necessary here. It is also not entirely obvious how this relates to Section 2.19. I list this here, because it belongs to the series.
2. PCA in relatively simple mathematical terms
   https://www.youtube.com/watch?v=a9jdQGybYmE (51:12)
   - 00:00–15:38 Why would you like to do PCA? [1.1]
   - 15:38–23:32 Variance and covariance [1.4]
     * 18:18–18:30 The vectors introduced 16:52-17:10 are row vectors, so that $\boldsymbol{a}\boldsymbol{b}^T$ is indeed an inner product. I always use column vectors, and then I write the inner product as $\boldsymbol{a}^T\boldsymbol{b}$
     * 22:26–22:54 The statement made here is not always true. Variables can be uncorrelated but statistically dependent, a simple example being $\cos(t)$ and $\sin(t)$ for $t \in [0, 2\pi]$. He uses the term stastistical independence for what really only is zero covariance or no correlation.
   - 23:32–25:06 Relation of variance and covariance to the motivational example
   - 25:06–35:49 Covariance matrix (assuming zero mean data) [2.8]
   - 35:49–39:44 Diagonalizing the covariance matrix [1.7]
   - 39:44–47:20 Diagonalization with eigenvalues and -vectors [2.11]
   - 47:20–49:43 Diagonalization with singular value decomposition
   - 49:43–51:12 Summary
3. Application of PCA to face recognition
   https://www.youtube.com/watch?v=8BTv-KZ2Bh8 (48:02)

A.4 Software

- General list of software for PCA
  https://en.wikipedia.org/wiki/Principal_component_analysis#Software.2Fsource_code
- PCA and variants thereof in scikit-learn, a python library for machine learning
  https://scikit-learn.org/stable/modules/decomposition.html#principal-component-analysis-pca
- Examples using PCA in scikit-learn
  https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA

References

Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, CA.

Lanitis, A., Taylor, C. J., and Cootes, T. F. (1995). An automatic face identification system using flexible appearance models. *Image and Vision Computing*, 13(5):393–401.

Turk, M. and Pentland, A. P. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86.

Notes

[3.1] Lanitis, Taylor, et al., 1995, Image and Vision Computing 13(5):393–401, Fig. 11, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=dcdfd33771b96da16bfd7f3babee237a16469540

[3.2] Lanitis, Taylor, et al., 1995, Image and Vision Computing 13(5):393–401, Fig. 6, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=dcdfd33771b96da16bfd7f3babee237a16469540