Chapter 6

Multivariate methods

Multivariate methods in data-analysis refer to the vast collection of methods that are applied to data with several variables. In principle, the regression analysis (linear or nonlinear models) with multiple variable data is also a multivariate method, but usually multivariate regression is treated separately. Different clustering, classification, pattern recognition and dimension reduction methods are in the core of multivariate data-analysis.

6.1 Multivariate distributions

Multivariate distributions are distributions for vector-valued random variables, and multivariate pdf's and cdf's are functions from $\mathbb{R}^n$ to positive real axis $\mathbb{R}^+$. Apart from the fact that the variable is multidimensional, they are just like one-dimensional distributions.

With one-dimensional distributions there are plenty of different types of choices available. With multiple dimensions, the multivariate normal distribution governs the field and other choices are rare. With independent variables this is not an issue, since the joint distribution of independent components is the product of the one-dimensional distributions. With just a few components these distributions are often called by the names of the individual components, e.g. gamma-normal distribution for the product distribution of gamma and normal distributed variables.

6.1.1 Multinormal distribution

Multinormal distribution for p-dimensional random vector $\boldsymbol{Y}$, $\mathcal{N}_p$, is parametrized by p-dimensional vector of expected values $\boldsymbol{\mu}$ and $p \times p$-dimensional covariance matrix $\boldsymbol{\Sigma}$. The pdf is

$$\mathrm{f}(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{p}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})\right), \tag{6.1}$$

where $\det(\cdot)$ is the determinant of a matrix.

The covariance matrix $\mathbf{\Sigma}$ has all the information about the dependencies between multinormal variables. Two variables Y_i and Y_j are independent if $[\mathbf{\Sigma}]_{ij} = \sigma_{ij} = \sigma_{ji} = 0$. In that case their correlation is also zero. Note that for other than multinormal variables it might be that the (linear) correlation between the variables is zero, but that they are not independent. For normal distribution, however, correlation is equivalent to dependency.

The possible dependency can be generalized to groups of variables. Let us say that the random vector $\boldsymbol{Y}$ constitutes of k components $A_1, \ldots, A_k$, and m components $B_1, \ldots, B_m$. The random vector, expected value vector and the covariance matrix can be partitioned into submatrices or -vectors:

$$\boldsymbol{Y} = [\boldsymbol{A}\ \boldsymbol{B}]^T = [A_1\ \cdots\ A_k\ B_1\ \cdots\ B_m]^T \tag{6.2}$$

$$\boldsymbol{\mu} = [\boldsymbol{\mu}_A\ \boldsymbol{\mu}_B]^T = [\mu_{A_1}\ \cdots\ \mu_{A_k}\ \mu_{B_1}\ \cdots\ \mu_{B_m}]^T \tag{6.3}$$

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_{AA} & \mathbf{\Sigma}_{AB} \\ \mathbf{\Sigma}_{AB} & \mathbf{\Sigma}_{BB} \end{bmatrix} \tag{6.4}$$

Now, if the variables $\boldsymbol{A}$ are all independent of $\boldsymbol{B}$, it means that $\mathbf{\Sigma}_{AB} = \mathbf{0}$. Furthermore, it holds now that $\mathbf{A} \sim \mathcal{N}_k(\boldsymbol{\mu}_A, \mathbf{\Sigma}_{AA})$ and similarly for $\mathbf{B}$. Two examples of pdf's of two-dimensional normal distribution are shown in Fig. 6.1. The variables are independent in the first example, and dependent on the second.

Construction of multinormal distribution

It might be useful to understand how a multinormally distributed variables are formed. First of all, we need p random variables Z_i that are independently and normally distributed. Without loss of generality, we can assume at this point that they all are distributed as $Z_i \sim \mathcal{N}(0, 1)$.

Second, let us have a $p \times p$ matrix of coefficients c_{ij}, $\mathbf{C}$. Third, we need a vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_p)$. Now we can construct a new random vector $\boldsymbol{Y}$ as

$$\begin{aligned} Y_1 &= c_{11}Z_1 + \ldots + c_{1p}Z_p + \mu_1 \\ Y_2 &= c_{21}Z_1 + \ldots + c_{2p}Z_p + \mu_2 \\ &\vdots \\ Y_p &= c_{p1}Z_1 + \ldots + c_{pp}Z_p + \mu_p \end{aligned} \tag{6.5}$$

which can be written shorter as

$$\boldsymbol{Y} = \mathbf{C}\boldsymbol{Z} + \boldsymbol{\mu} \tag{6.6}$$

After this transform $\boldsymbol{Y}$ has multinormal distribution $\boldsymbol{Y} \sim \mathcal{N}_p(\boldsymbol{\mu}, \mathbf{\Sigma})$, where $\mathbf{\Sigma} = \mathbf{C}\mathbf{C}^T$.

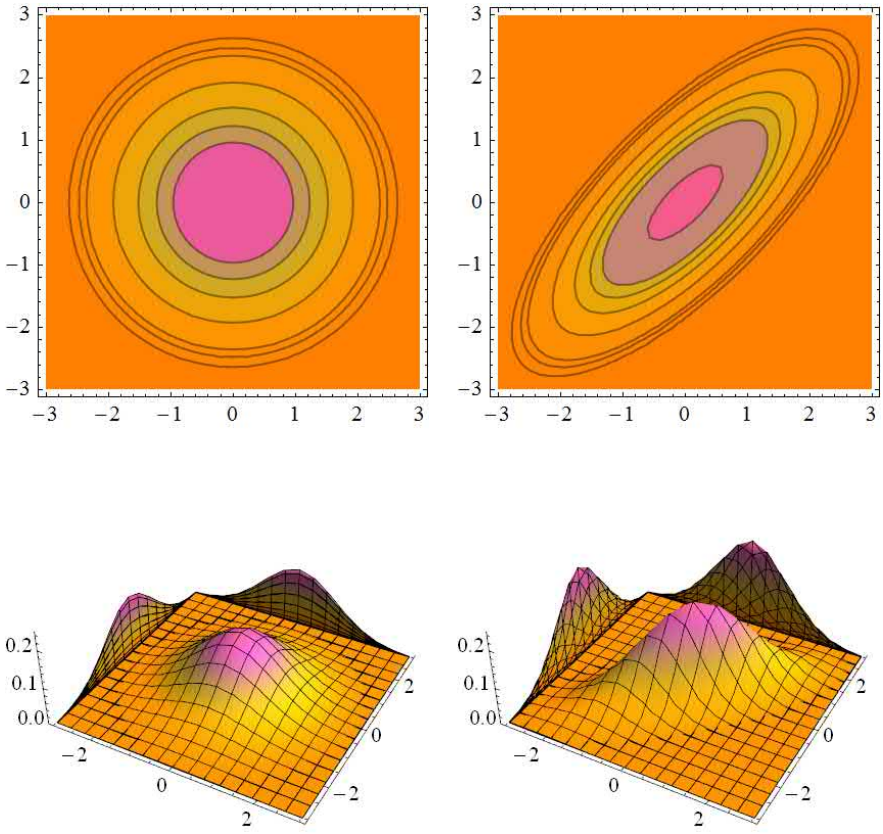

Figure 6.1: Contour plots (upper row) and 3D plots (lower row) of two-dimensional normal distribution. Distribution on left has no dependence ($\rho = 0$) between the variables, while distribution on the right has $\rho = 0.75$.

The construction of multinormal variables above can be used to create samples of (pseudo)random numbers from multinormal distribution. The creation of standard $(0, 1)$ normal random numbers is available in almost all software packages, so it is easy to create sample $\boldsymbol{Z} = (Z_1, \ldots, Z_p)$. The required covariance matrix should be decompositioned with Cholesky decomposition $\boldsymbol{\Sigma} = \mathbf{C}\mathbf{C}^T$, or preferably with eigendecomposition (*ominaisarvohajotelma*) $\boldsymbol{\Sigma} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$, where $\boldsymbol{\Lambda}$ is diagonal matrix of eigenvalues. In the latter case, $\mathbf{C} = \mathbf{U}\boldsymbol{\Lambda}^{1/2}$. Now Eq. (6.6) can be directly applied to $\boldsymbol{Z}$ to get the multivariate random sample:

$$\boldsymbol{Y} = \mathbf{U}\boldsymbol{\Lambda}^{1/2}\boldsymbol{Z} + \boldsymbol{\mu}. \tag{6.7}$$

Because $\boldsymbol{\Lambda}$ is diagonal matrix, the $\boldsymbol{\Lambda}^{1/2}$ is simply $\left[\sqrt{\Lambda_{11}} \cdots \sqrt{\Lambda_{pp}}\right]$. Note that the equation above is for one sample vector $\boldsymbol{Y}$. If you need to construct a matrix $\mathbf{Y}$ where all the rows are from the same multinormal distribution, $\boldsymbol{Y_i}. \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})\ \forall i$, then the matrix version of the construction is as

$$\mathbf{Y} = \mathbf{Z}\,\boldsymbol{\Lambda}^{1/2}\,\mathbf{U}^T + \mathbf{1}_{n,p}\,\mathrm{diag}(\boldsymbol{\mu}). \tag{6.8}$$

where $\mathbf{1}_{n,p}$ is $n \times p$ matrix full of ones, and $\mathrm{diag}(\cdot)$ is an operator that constructs a diagonal matrix of the values.

Mahalanobis distance

The Mahalanobis distance is a generalized distance measure that is suitable for multinormal distributed variables. Let us have an example of two-dimensional sample from multinormal distribution as in Fig. 6.2. The two variables might measure completely different quantities and thus have different scales. The expectancy of the distribution is at $(100, 1)$. Let us say that we have three interesting observations, the red, green and the blue dots in the figure. One might want to know which one is furthest from the expected value.

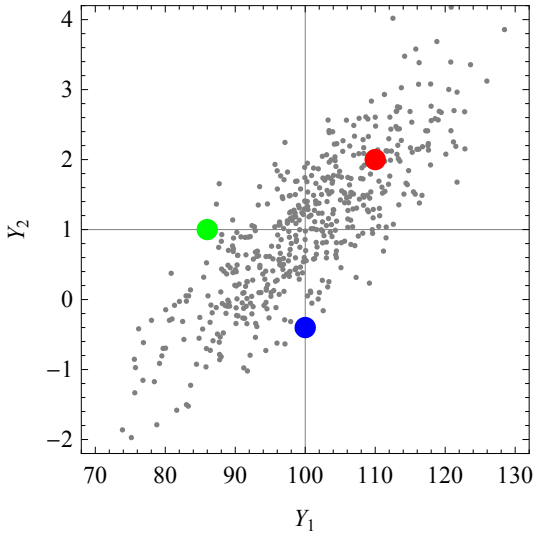

Figure 6.2: Random multinormal sample and Mahalanobis distance.

The expected value (mean) has coordinate $\overline{\boldsymbol{y}} = (\overline{y_1}, \overline{y_2})$. The squared Euclidean distance to mean would be $D_e^2 = (\boldsymbol{y} - \overline{\boldsymbol{y}})^T(\boldsymbol{y} - \overline{\boldsymbol{y}})$. In this case, the distances would be about 10 (red), 14 (green), and 1.4 (blue) for the three colored dots. Euclidean distance is clearly a bad measure in this case, since it assumes that both coordinate axes Y_1 and Y_2 have the same scale.

An improved version of the distance measure could be constructed if the observations would be normalized (scaled with their standard deviations) before taking the Euclidean distance. However, that procedure would not take into account the evident strong correlation between the variables. After normalization the points would have approximately the same Euclidean distances to mean. Still, based on the gray sample points from the distribution, it would seem that the red point is "more common" and should have smallest distance from mean.

The Mahalanobis distance takes both the scales of the different axis and the corre-

lation into account. The distance is defined as

$$D_m = \left((\boldsymbol{y} - \overline{\boldsymbol{y}})^T \, \mathbf{S}^{-1} \, (\boldsymbol{y} - \overline{\boldsymbol{y}}) \right)^{1/2}, \tag{6.9}$$

where $\mathbf{S}$ is the sample estimate of the covariance matrix. One can see that the Mahalanobis distance is Euclidean distance that is weighted by the inverse of the covariance. For multinormal sample this is the correct distance measure to be used.

Test of multinormality with Mahalanobis distance

There are a number of tests for multinormality, each focusing on different requirements for a multinormal sample. The Mahalanobis distance can also be used to test the multinormality. It can be shown that the squared Mahalanobis distances of multinormal sample should have the χ^2-distribution with p degrees of freedom. The Q-Q plot, as described in Fig. 3.7 and the related text, can be used to graphically check the distribution assumption. Sorted squared distances are plotted on the vertical axis, and quantiles from the $\chi^2(p)$-distribution of the squared distances on the horizontal axis. The points should lie close to diagonal line if the sample is from multinormal distribution.

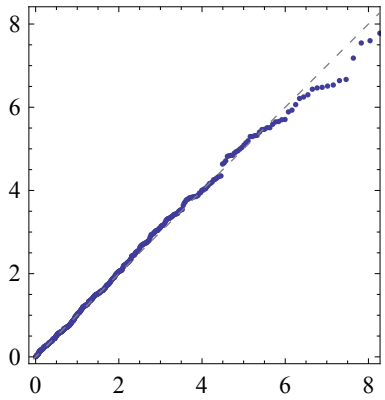

Figure 6.3: Q-Q-plot of the squared Mahalanobis distances against χ^2-distribution from the sample in Fig. 6.2

6.2 Dimension reduction

In multivariate analysis we often need to deal with data that has many (tens, hundreds) properties (i.e., variables) measured or observed. However, even the visual presentation of such data can be difficult, not even mentioning the statistical analysis. It is quite possible that there are underlying dependencies among the variables. Finding these dependencies can be valuable in such, but it can also help to reduce the number of variables.

With reduced dimensions the visual data mining, clustering, classification etc. will become easier. In this section we will go through two methods to analyze the underlying *linear* dependencies in the data, the principle component analysis (PCA) and the linear discriminant analysis (LDA). If correlated variables exist in the data, both the methods can be used to find them, to create a set of new variables based on these correlations, and to reduce the dimensionality of the data by dropping out the non-important new variables.

The difference between the PCA and the LDA is that the PCA is suitable for so-called *unsupervised* analysis where we have no prior knowledge about the possible groups/clusters/classes in the data, and LDA for *supervised* analysis where we have a 'training set' for which we already know the groups/clusters/classes of every observation.

6.2.1 Principle component analysis

Principle component analysis (PCA, *pääkomponenttianalyysi*) is one of the most important multivariate methods, especially in natural sciences. In social sciences Factor Analysis (*faktorianalyysi*) is similar and popular method, but PCA is more 'physical' while there are more possibilities to subjective judgment in factor analysis.

The importance of PCA comes from its wide applicability. PCA can be used in visual analysis, clustering, pattern recognition, exploratory data analysis, variable reduction, searching for dependency structures etc. Furthermore, PCA is quite straightforward to implement and is 'objective' in the sense that it does not need any parameters to be set.

PCA can be understood perhaps the easiest way be a geometrical approach. In Fig. 6.4 (a) there are contour ellipses from two-variate normal distribution. There is correlation between the variables, so the axis of the ellipsoids are not parallel to the coordinate axis. What the PCA does is that is searches for these axis of the contour ellipses and then transforms the data so that the ellipse axis are the new coordinate vectors. After PCA the new variables (coordinate axis) are uncorrelated, as shown in Fig. 6.4 (b).

Implementing principle component transform

The PCA can be implemented quite easily in a computing environment where there are tools for matrix algebra and for eigenvalue decomposition. The data matrix $\mathbf{Y}$ has n rows, one for each observation, and p columns for the variables. First the data matrix needs to be centered or standardized. If the data is only centered, the method is based on the covariances, and if standardized, it is based on the correlations.

The correct method can be chosen based on the quantities and scales the variables are measuring. If all the variables measure the same quantity, and we want to

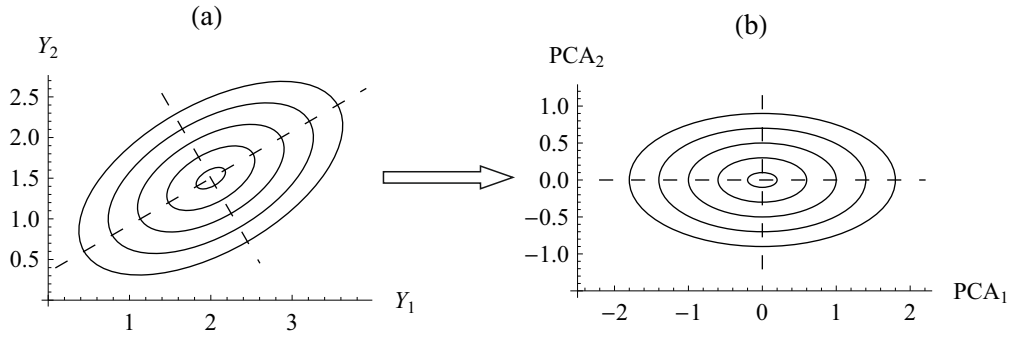

Figure 6.4: Sketch of the PCA in geometrical interpretation.

preserve the information that is in the variances of the variables, we should choose the covariance method. The centering of the data is done using the mean vector $\overline{\boldsymbol{y}}$ which holds the mean values over the observations for each variable, i.e.

$$\overline{\boldsymbol{y}} = (\overline{y}_1, \ldots, \overline{y}_p) = \frac{1}{n}(\sum_i^n y_{i1}, \ldots, \sum_i^n y_{ip}). \tag{6.10}$$

The centered data matrix $\mathbf{X}$ is computed from $\mathbf{Y}$ by:

$$\mathbf{X} = \mathbf{Y} - \mathbf{1}_{n,p}\,\mathrm{diag}(\overline{\boldsymbol{y}}), \tag{6.11}$$

where $\mathbf{1}_{n,p}$ is $n \times p$ matrix full of ones, and $\mathrm{diag}(\cdot)$ is an operator that constructs a diagonal matrix of the values.

However, if the variables measure different quantities and their variances cannot be compared with each other, we should choose the correlation method and use the standardized data matrix. In standardization the centered data is further divided by standard deviations, variable by variable. This can be formulated with the diagonal matrix of inverses of standard deviations, $[\mathbf{V}]_{ii} = 1/s_{ii}$ as

$$\mathbf{X}^* = \mathbf{X}\,\mathbf{V} \tag{6.12}$$

The rest of the PCA procedure is identical for correlation and covariance methods, so we use symbol $\mathbf{X}$ for both the cases. Next, the sample estimate to covariance matrix $\mathbf{S}$ is needed. If (and only if) the data matrix is centered, as with $\mathbf{X}$ here, the sample covariance matrix can be computed as

$$\mathbf{S} = \frac{1}{n-1}\mathbf{X}^T\,\mathbf{X}, \tag{6.13}$$

If $\mathbf{X}$ was standardized, $\mathbf{S}$ is actually correlation matrix.

Third step is to compute the eigenvalue decomposition of $\mathbf{S}$. Eigenvalue decomposition is such that

$$\mathbf{S} = \mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^T, \tag{6.14}$$

where $\mathbf{U}$ is the $p \times p$ matrix of eigenvectors, and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues*. Finally, the data is transformed into PCA space by

$$\mathbf{Z} = \mathbf{X}\,\mathbf{U}. \tag{6.15}$$

An example of PCA transform in shown in Fig. 6.5.

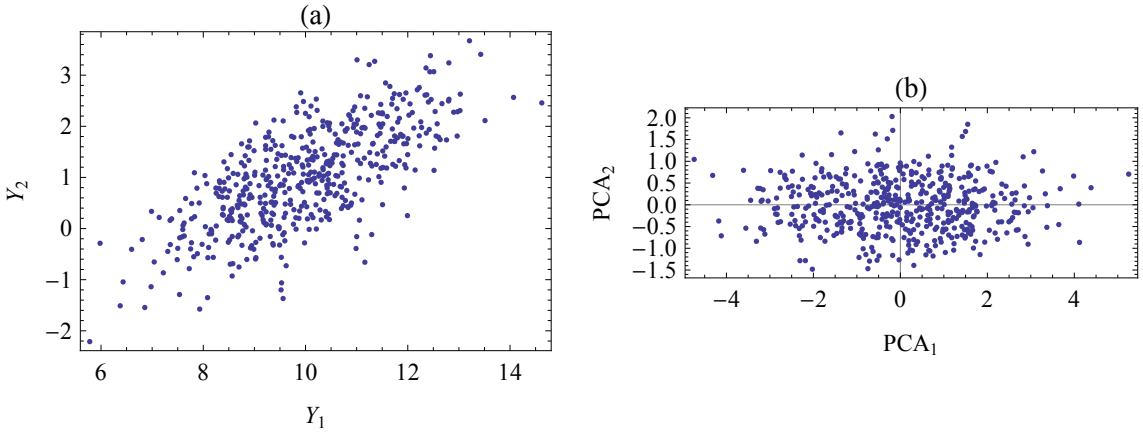

Figure 6.5: Example of PCA transform to 500 observations from two-dimensional multinormal distribution. Original observations are in subfigure (a), and data in PCA space in (b).

Interpretation of principal components

As can be seen from Eq. (6.15), PCA is a linear transform. If $\boldsymbol{u}_j$'s are the eigenvectors in $\mathbf{U} = [\boldsymbol{u}_1 \cdots \boldsymbol{u}_p]$, and $\mathbf{x}_i$ is the row in centered (standardized) data matrix, the value of jth new PCA variable for observation i is

$$z_{ij} = \boldsymbol{x}_i^T\,\boldsymbol{u}_j = x_{i1}u_{1j} + \ldots + x_{ip}u_{pj} \tag{6.16}$$

In that context, the eigenvectors $\boldsymbol{u}_j$ are the new coordinate basis, and map the original variables to the PCA space. The eigenvectors are often called *loadings*. Large absolute values in u_{kj} mean that original variable k has large impact, loading, to PCA variable j. Therefore by plotting eigenvectors one can visually inspect how the original variables influence the PCA variables.

*Note that some numerical eigensystem algorithms might return the matrix of eigenvectors so that the eigenvectors are the rows of $\mathbf{U}$. In the equations here we assume that the eigenvectors are the columns of $\mathbf{U}$. You can easily check which way it is by checking if $\mathbf{S} - \mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^T = \mathbf{0}$ or if $\mathbf{S} - \mathbf{U}^T\,\mathbf{\Lambda}\,\mathbf{U} = \mathbf{0}$. In the latter case the eigenvalues are the rows, and you need to compute $\mathbf{Z}$ as $\mathbf{Z} = \mathbf{X}\mathbf{U}^T$.

The eigenvalues, i.e. the diagonal values in $\mathbf{\Lambda}$ are the variances of the data in the PCA space. The PCA will preserve the total variance of the data, i.e.

$$\sum_{j}^{p}[\mathbf{\Lambda}]_{jj} = \sum_{j}^{p}[\mathbf{S}]_{jj} \tag{6.17}$$

In PCA based on the standardized data matrix the total correlation is preserved, so $\sum_j^p[\mathbf{\Lambda}]_{jj} = p$.

Principal component analysis in variable reduction

One of the applications of PCA is in variable or dimensionality reduction or data compression. The fact that the PCA variables are uncorrelated makes this possible. Unnecessary PCA variables can be removed without affecting the remaining variables. The variances of the PCA variables is used to judge which variables are ”unnecessary”.

Usually the procedure that computes eigenvalues and -vectors already sorts them so that the first eigenvalue is the largest and so forth. The eigenvectors are also sorted because the order of values and vectors must match. If this is not done by the procedure, one should do this manually. So, eigenvalues must be sorted so that $\Lambda_{[1]} \geq \Lambda_{[2]} \geq \cdots \geq \Lambda_{[p]}$. The same ordering must then be applied for eigenvectors, $\mathbf{U} = [\boldsymbol{u}_{[1]}\, \boldsymbol{u}_{[2]}\, \cdots\, \boldsymbol{u}_{[p]}]$.

If there are correlations between the original variables, it is often so that the total variance in the data is redistributed with PCA variables so that the first few PCA variables make up almost all the total variance. The interpretation is that the first few PCA variables with large variances are the ”real signal” and the rest of the PCA variables with variances close to zero are ”random noise”. Variable reduction is based on this.

The portion c of total variance that is reproduced with the first k PCA variables is derived with

$$c = \frac{\sum_j^k \Lambda_j}{\sum_j^p \Lambda_j}. \tag{6.18}$$

Usually the limit for c is set close to 100 %, to 95 % or 99 % for example. When the first k PCA variables can reproduce the required portion, the variable reduction is done by forming $\mathbf{U}^* = [\boldsymbol{u}_1 \cdots \boldsymbol{u}_k]$, i.e. taking only the first k eigenvectors and dropping out the rest. The reduced data $\mathbf{Z}^*$ in PCA space is received by $\mathbf{Z}^* = \mathbf{X}\,\mathbf{U}^*$. The reduced matrix has now only k variables. If the PCA variable reduction is successful, the reduced number of variables k can be significantly smaller than the original number of variables p.

One application for PCA variable reduction is the visualization of high-dimensional data. If the first two or three PCA variables can reproduce a large portion of the total variance, the data can be visualized in 2D or 3D plots in the reduced PCA

space. Another is in classification or clustering problems. While PCA is not itself optimized for classification, it can find structures in the data that can be both visualized in low dimensions, and used in classification. An example of this is shown in Fig. 6.6.

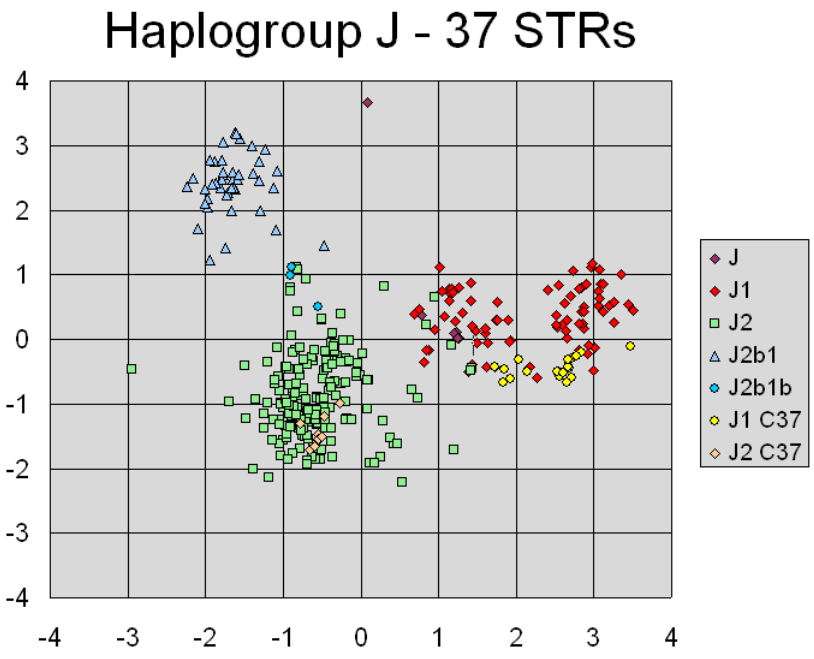

Figure 6.6: PCA example from Wikipedia. A PCA scatterplot of haplotypes calculated for 37 Y-chromosomal STR markers from 354 individuals. PCA has successfully found linear combinations of the different markers, that separate out different clusters corresponding to different lines of individuals' Y-chromosomal genetic descent.

6.2.2 Linear discriminant analysis

The PCA is a highly popular method for data analysis and dimension reduction, and is often used prior to clustering or classification analysis to produce (important) transformed variables that would be more optimal to be clustered. The PCA usually succeeds quite well, but it fails to take into account any prior knowledge about different classes of observations in the data. If these classes are actually known for the data, usually then called the training data, the linear discriminant analysis (LDA) is a very close relative to the PCA but with the ability to acknowledge the classes in the data.

A visual explanation about the difference between these methods is shown in Fig. 6.7. The original data, quite similar as in Fig. 6.5, is shown in the uppermost subplot. The data has two variables with evident linear correlation between them. In this

example, we actually know beforehand that the data has two distinct classes of observations, the blue and the red dots.

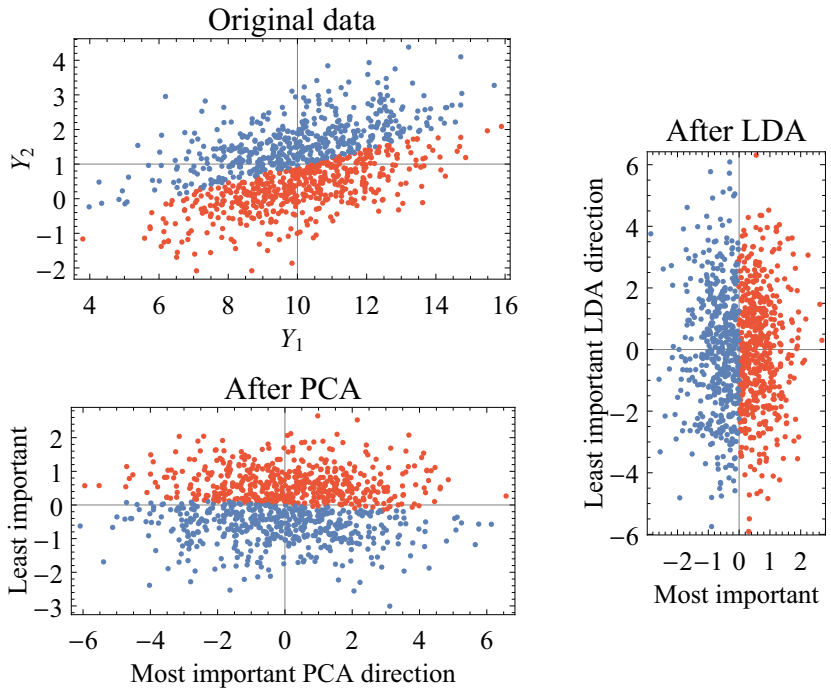

Figure 6.7: The difference between the PCA and LDA when there are known classes in the data.

As seen in the lower left subplot, the PCA will find the direction with the maximal variability in the data, which is the vertical axis in the subplot. However, the vertical axis is useless in classifying the blue and red dots from each other. Remember that the new axis are ordered from the most important to the least important. With only two variables, the second axis, which would separate the classes, is also the least important and would probably be left out in the dimension reduction procedure.

However, as seen from the subplot on the right side, the LDA does the opposite from the PCA and aligns the first and the most important axis along the direction which gives the most information for separating the classes. This is roughly what the LDA aims to do — find new variables to maximize the separation between the classes in the data.

Implementing linear discriminant analysis

Implementing LDA is somewhat similar to implementing the PCA, but it requires taking into account the different classes in the data. We will need to have the same standardized matrix $\mathbf{X}$ as in Eq. (6.11) or (6.12) for covariance- or correlation-based analysis. In addition, we will need the mean vectors $\overline{\boldsymbol{x}_c}$ and covariance matrices $\mathbf{S}_c$ for the standardized observations $\boldsymbol{x}_i$ for each group $c = 1, \ldots, k$.

With these group-based mean vectors and covariances, we compute the within-class covariance matrix

$$\mathbf{W} = \sum_c (n_c - 1)\mathbf{S}_c, \tag{6.19}$$

where n_c is the number of observations for group c, and the between-classes covariance matrix

$$\mathbf{B} = \sum_c n_c(\overline{\boldsymbol{x}_c}\ \overline{\boldsymbol{x}_c}^T). \tag{6.20}$$

From these we can form the LDA projection matrix with the help of eigenvalue decomposition as

$$\mathbf{L}\mathbf{\Lambda}\mathbf{L}^{-1} = \mathbf{W}^{-1}\mathbf{B}, \tag{6.21}$$

where the matrix $\mathbf{L}$ is not orthogonal anymore, because the matrix to be decomposed is not symmetric as with the PCA. Again, different algorithms might output either $\mathbf{L}$ or $\mathbf{L}^T$, please check this before implementing.

Finally, the LDA-transformed new variables are given by

$$\mathbf{Z} = \mathbf{X}\mathbf{L}. \tag{6.22}$$

Similar to the PCA, the most important variables are ordered as the first columns in the matrix $\mathbf{Z}$, so the dimension reduction can be done by discarding the columns after the most important ones.

6.3 Classification

Classification, also called supervised learning, is a set of techniques to assign new observations into pre-defined classes. To have pre-defined classes one needs a training data set where these classes are already known. The classifier is 'trained' using this data, and then used for future observations without the knowledge of the correct classification.

For classification, a pre-treatment to the data is usually needed, especially if the data has many variables. Dimension reduction such as PCA or LDA is recommended before the classification, since it makes the task for the classification algorithm easier. In an ideal case where the classes are actually known beforehand, LDA is recommended over PCA which does not exploit the class structure in the data.

However, sometimes the classification is done so that the classes are actually not known beforehand. In the analysis the PCA or similar is first applied to the training set without the class information. From the result a visual and subjective analysis is done to define the classes, and then the classifier is build upon these.

In what follows, three classification algorithms are introduced and applied for the classical Iris flower data set of three Iris flower species and the measures of their

sepal and pedal flower dimensions. The data with four variables is first treated with the LDA, and the two most important new variables are extracted, as seen in Fig. 6.8.

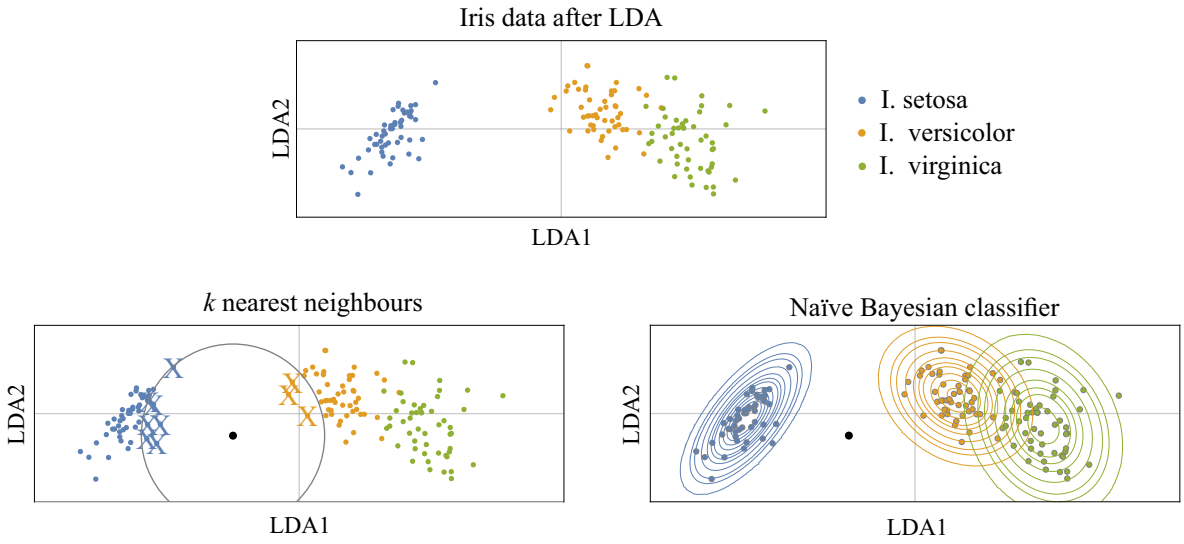

Figure 6.8: The Iris dataset after the LDA dimension reduction on the top panel. On the lower left panel, a sketch of the k-nearest neighbor algorithm and on the right, the naïve Bayesian classifier algorithm.

The first 'algorithm' to be introduced is a heuristic division of the variable space, often with some simple linear borders, to different class areas/volumes. This division is subjective and often done just 'by eye' by the researcher. Still, this kind of taxonomic systems are quite popularly used. An example of the variable space division for the Iris data is shown in Fig. 6.9.

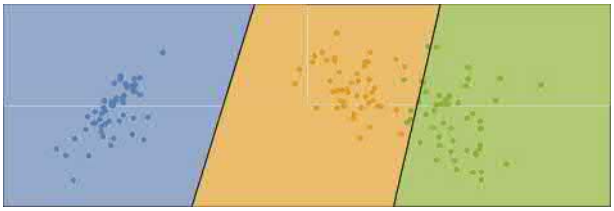

Figure 6.9: An example of a heuristic classification of the Iris data (after the LDA transform) and the resulting areas of the variable space for the three classes.

6.3.1 Nearest-neighbor method

The nearest-neighbor (N-N) method is also quite simple, but also robust, method for classification. The simple procedure is to find n closest points in the training set for the new observation (see Fig. 6.8). The class frequencies within these n points

are computed, and the most frequent class is assigned for the new observation. The most simple version searches only for the one nearest neighbor and classifies to the same class as this. This N-N method using only one point will actually create a Voronoi division of the variable space, as shown in Fig. 6.10.

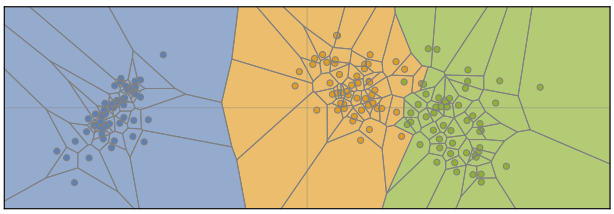

Figure 6.10: The Iris data after the LDA transform and the Voronoi division in the variable space corresponding to the one-nearest-neighbor classification areas.

If some kind of dimension reduction / variable transform technique is used before the N-N classification, the normal Euclidian distance should be a proper distance measure with this algorithm. However, if the original variables are used, one might want to apply more suitable distance measures such as the Mahalanobis or Manhattan distances when searching for the nearest neighbors.

6.3.2 Naïve Bayes classifier

The probabilistic classifier is based on the probability distribution assumption on the data, and if the *a priori* probabilities are also considered, this method is called the naïve Bayesian classifier (NBC). In most cases, it is assumed that the data in each class follows the multinormal distribution with the class mean vectors $\boldsymbol{\mu}_c$ and the covariance matrices $\mathbf{S}_c$. With new observation $\boldsymbol{x}$, the probability p_c to belong to class c is computed simply from the assumed (multinormal) probability distributions

$$p_c = \mathrm{f}_c(\mathbf{x}; \boldsymbol{\mu}_c, \mathbf{S}_c). \tag{6.23}$$

If Bayesian classification is sought for, the *a priori* probabilities a_c, which can be computed, for example, from the training data frequencies, are also considered as

$$p_c = a_c \, \mathrm{f}_c(\mathbf{x}; \boldsymbol{\mu}_c, \mathbf{S}_c). \tag{6.24}$$

Finally, the most probable class is selected, i.e., the class c with the highest value of p_c. This is the probabilistic or the naïve Bayesian classifier. The multinormal distributions are shown in Fig. 6.8 for the Iris data, and the areas in the variable space in Fig. 6.11.

6.4 Clustering

While classification is supervised learning with pre-defined class information, clustering is a non-supervised method without the prior knowledge of the possible

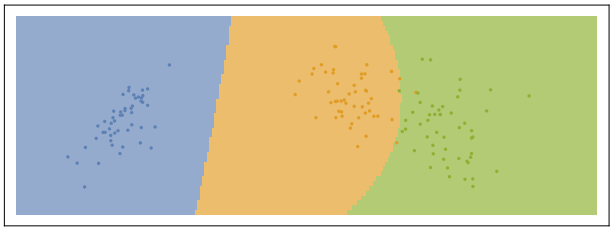

Figure 6.11: The Iris data after the LDA transform and the variable space division corresponding to the probabilistic classifier.

groups in the data. The clustering algorithms works by choosing groups for each observation by minimizing a chosen measure of "group conformance" while maximizing the difference between the groups in some sense. This is usually done for different number of groups, and the recommended number of groups is chosen so that it optimizes the ratio between "within-group" and "between-groups" variances. In clustering, as with classification, a pre-treatment to the data (PCA or similar) is recommended, or the distance measure can be chosen to be some other than the Euclidean distance. In Fig. 6.12, the Iris data is divided into 2 to 5 groups.

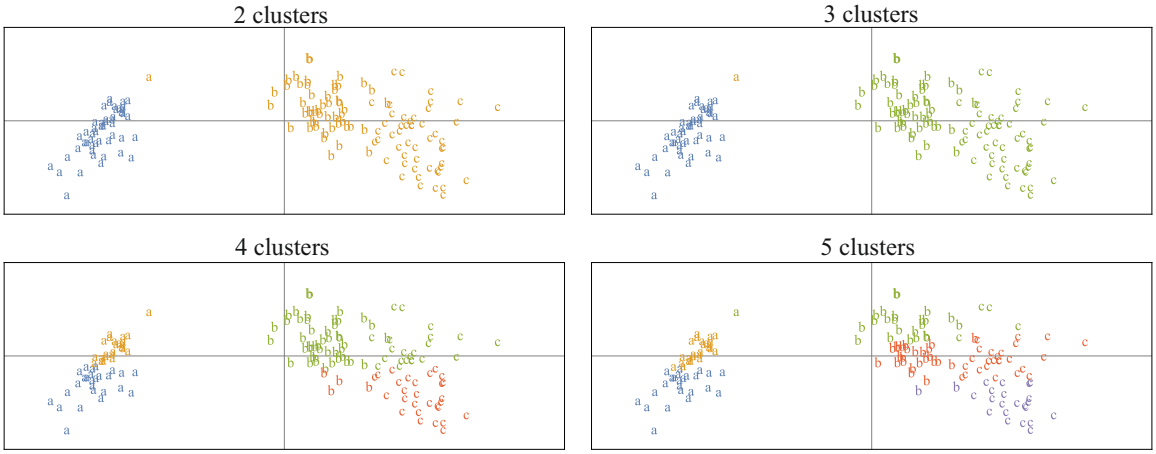

Figure 6.12: The Iris data after the LDA transform clustered into 2 to 5 groups. The letters a–c mark the real groups, and the colors the clusters found by the algorithm.

With the example in Fig. 6.12 one can see that at least the default clustering algorithm of the Mathematica software using the 'k-means' method cannot find the original flower subspecies from the data. The *setosa* species is found quite well when dividing into 2 or 3 clusters, but *versicolor* and *virginica* cannot be separated that well.

Clustering can be done in a tree-like structure where the observations are group together one-by-one and finally ending into one group. This structure can be plotted into *dendogram* or clustering tree, as in Fig. 6.13.

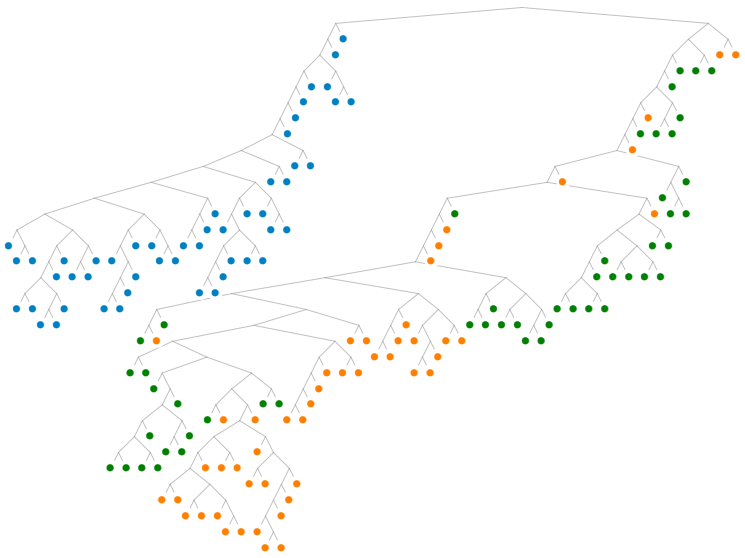

Figure 6.13: Clustering tree representation of the sequential grouping of the Iris flower data observations. The grouping is started from the bottom of the tree, and finally the two main groups are joined into one at the top of the tree. The colors of the points are the same as in Fig. 6.8