Chemometrics I
Exercises

Ron Wehrens – Lionel Blanchet,
Institute for Molecules and Materials
Analytical Chemistry/ Chemometrics
Radboud University Nijmegen
Last edit: September 16, 2015 (Geert Postma)

1 Introduction to the Chemometrics I course exercises

The following hands-on exercises teach you step by step how to perform and understand various chemometrical analysis techniques. These exercises go hand in hand with studying the theory using the Study Guide, the lectures / lecture sheets and the background material. Each exercise contains a number of questions which address a specific issue of the data analysis. In many of the exercises, you will write functions that eventually will form your own data analysis toolbox. Store the functions in your local directory so you can apply them in the remainder of the exercises.

You are requested to write your functions and scripts **before** the first seminar / problem session following each lecture.

During this problem session you will have the opportunity to check your functions and scripts, discuss the results of the exercises with the teaching assistants and repair if necessary possible errors observed. The functions and scripts are then used for the assignments that will be provided next.

Matlab should be used. Although programming is not the goal of the Chemometrics I course, still for properly and efficiently doing the exercises, you should have some knowledge of and experience with Matlab. You should know how to write a function with if-then statements and for-loops and you should be able to plot data with the plot command.
Some helpful commands and functions are provided in the header of each exercise, but this is only for reference (also R commands are listed there: you can skip that information). Once a command or function has been introduced, it is assumed that you are able to use it in the other exercises too.

The data for the exercises are available at the course website:

http://webchem2.science.ru.nl/chemometrics-i/

The data are in dat-format which can be read by Matlab.

Appendix A contains some typical examples of (sequences of) Matlab commands and functions.

The exercises

During the first problem session following each lecture (about one week after the lecture) a quick check of the functions and scripts will take place by one of the teaching assistants and a test data set with expected outcome will be provided. Using this test data set and the expected outcome you can check the proper functioning of your functions and scripts. The teaching assistants are available during this problem session to assist you in case of problems and errors in the functions and scripts that you cannot solve yourself.

The Matlab code of your functions and scripts should be saved in .m files. Make sure your code complies to the following rules:

- the header of every function includes a description, including input/output variables, goal of the function and name of the author(s);
- the actual code is sufficiently explained by making use of comments – have a look Appendix A for some examples;
- variables and functions use logical names (e.g. don’t hand in functions named sterretje3.m, but use PCA.m, QDA.m, etc.);
- variables and functions are named in a consistent manner throughout the course (e.g. use P in all functions to indicate a column mean);
- the code does not make use of built-in Matlab-functions. Of course you are allowed to use generic functions like mean, sd, size, etc.

As stated earlier: note that you will have to use the same functions in multiple chapters, so store them in separate .m files and make them generally applicable for different problems! (E.g. do not put the command ‘load 'iris.dat' ‘ in every function in order to have access to your data: the data on which the function is to be applied should be addressed with (an) input argument(s).) Also note that you are responsible for the proper functioning of your functions and scripts. They are to be used for the assignments that follow.

2 Multivariate analysis: PCA

Exercise 2.1 – Making a plot

Product: the plot function

Useful Matlab commands: help, function, load, size, plot
Useful R commands: ?help, source, function, read.table, dim, plot

As a start, load the *iris* data containing the input variables and species labels.

Question 2.1.1: Inspect the size of the loaded data: how many objects (samples) and variables does the iris data matrix contain?

Question 2.1.2: Plot the sepal length versus petal length (1st column versus 3rd column). In the appendices you will find examples of how to make informative plots.
Label the x-axis and y-axis and also give the plot a title. Note that labelling of plots is convenient for yourself and others.

Write a plot function which makes a plot of two columns of a data matrix, with a title and axes labels.

Input: X (data matrix),
C1 (column index 1),
C2 (column index 2),
Title, Xlabel, Ylabel.

Output: None.
Note: this function is given in Appendix A.

Is there any relation between sepal length and petal length? Do you observe a relation between sepal and petal width?

Question 2.1.3: To make the visualization easier, you can use different colours and plotting symbols to indicate the different species (see the appendices). Create such a plot. Which iris species has the largest sepal length, the smallest sepal length, and the largest and smallest petal length?

Exercise 2.2 – Summary statistics

Product: the barplot function

Useful Matlab commands: mean, std, bar, hold
Useful R commands: colMeans, sd, barplot

Question 2.2.1: Calculate the means of the columns (variables) of the iris data. How would you calculate the means of the rows (i.e., the mean value per object)? Which type of leaf is – on average – the biggest? Visualize this in a bar plot (do not forget title, labels, etcetera).

Question 2.2.2: Next, calculate the standard deviations of the columns. Which variable has the largest standard deviation? Visualize this by plotting the standard deviation as a bar plot.

Write a barplot function which shows for each column of X the mean as well as the ±1 standard deviation limits.

Input: X (data matrix),
Title, Xlabel, Ylabel.

Output: None.
Note: this function is given in the Appendix A.

Exercise 2.3 – Scaling

Products:

- a function which performs mean-centering;
- a function for auto-scaling;
- a SNV scaling function;
- a function which performs mean-centering and auto-scaling based on the means and standard-deviations of another matrix.

Question 2.3.1: Calculate the mean-centered and auto-scaled versions of the iris data. Plot the first row of the original, unscaled data; use lines. Add the first rows of the mean-centered and auto-scaled matrices to the plot. Can you understand the differences?

Write a function that returns the mean-centered data matrix and the mean values per column.

Input: X (the data matrix).

Output: MCX (the mean-centered matrix),
MC (a vector containing the mean values of X per column).

Write a function that returns the auto-scaled data together with the mean and standard deviation per column.

Input: X (the data matrix).

Output: ASX (the auto-scaled matrix),
MC (a vector containing the mean values of X per column),
SD (a vector containing the standard deviation of X per column).

Repeat this for Standard Normal Variate (SNV) -scaling. Hint: SNV-scaling is similar to auto-scaling but now conducted row-wise.

Input: X (the data matrix).

Output: SNVX (the SNV-scaled matrix).

Question 2.3.2: Perform auto-scaling on the iris data, but now use 2, 3, 4 and 5 as column means, and use 1, 1, 0.02 and 0.04 as column standard-deviations, respectively, instead of the real column means and standard-deviations. What are the effects of this scaling? Plot the results!

Write a function which performs mean-centering and auto-scaling, given a vector of column means and – for auto-scaling – also column standard deviations.

Input: X2 (the data matrix),
MC (mean value per column of the data matrix X),
SD (the standard deviation per column of X).

Output: MCX2 (the mean-centered data matrix),
ASX2 (the auto-scaled data matrix).

Exercise 2.4 – The variance-covariance and correlation matrices

Product:

- the variance-covariance function;
- the correlation function.

Useful Matlab commands: cov, corr, corrcoef, imagesc, colorbar
Useful R commands: cov, cor, image

Question 2.4.1: Compute the variance-covariance matrix for the iris data, which is given by 1/(n-1) $\mathbf{X}^T\mathbf{X}$, where **X** is a mean-centered data matrix. What is the size of the variance-covariance matrix and what does it represent? Why is the matrix symmetric?

Write a function which calculates the variance-covariance matrix for a matrix X. Compare the output of your function with the covariance function provided by your software package.

Input: X (the data matrix).

Output: VC (the variance-covariance matrix).

Question 2.4.2: Calculate the variance-covariance matrix of the transposed data matrix. What is the size of the matrix and what does it represent? Make an image of this matrix. Verify that your color scale distinguishes between positive and negative values!

Question 2.4.3: The variance-covariance matrix depends on the unit of measurement. Suppose the lengths and widths were not measured in cm but in inches (1 inch = 2.54 cm). Calculate the covariance matrix of the transposed matrix, now based on inches instead of cm, and compare its image with that of the previously calculated covariance matrix.

Question 2.4.4: A measure, analogous to the variance-covariance matrix but independent of scaling, is the correlation matrix. The (i, j)-th element of this matrix contains the correlation between columns i and j. The correlation matrix can be calculated in a number of ways. First, calculate this matrix element-by-element using for-loops. What is the correlation between sepal length and petal length? And between sepal width and petal width? How do these numbers compare with the plots you made in question 2.1.2?

Question 2.4.5: A simpler way to calculate the correlation matrix is to calculate the variance-covariance matrix of the auto-scaled version of matrix **X**. Calculate the correlation matrix for the iris data set in this way and check that your results are equal to the previous question. Repeat this for the data measured in inches. Show that the correlation matrices for the two different units of measurement are equal.

Write a function to calculate the correlation matrix. Compare the output of your function with the correlation function provided by your software package.

Input: X (the data matrix).

Output: CM (the correlation matrix).

Exercise 2.5 – Principal Component Analysis

Products:

- a function returning scores, loadings and variance per PC;
- a function that creates a Scree plot.

Useful Matlab commands: svd (NB:you can also use economy size: svds), log, sum
Useful R commands: svd, log, sum

Up to now we have dealt with simple but useful ways to investigate your data. Next a multivariate way.

Question 2.5.1: Perform svd on the mean-centered iris data: this leads to left singular vectors **U**, singular values **D**, and right singular vectors **V**. Plot the first two left singular vectors (the first two columns of **U**) against each other; use different colours or plotting symbols for the three classes. Also make a plot of the third against the fourth left singular vectors. What differences do you see?

Question 2.5.2: What is the relation between the sizes of the **U** and **V** matrices on the one hand, and the size of the iris data on the other hand? Verify that the columns of **U** and **V** are orthogonal and have a norm of one (i.e., both matrices consist of orthonormal column vectors).

Question 2.5.3: Calculate the percentage of variance of each PC. Visualize this in a bar plot. Which PC corresponds to the largest amount of explained variance?

Question 2.5.4: Calculate the scores from **U** and **D**. Plot the first two columns of the score matrix versus each other and compare this plot with the plot produced in **2.5.1**. Are the columns in the score matrix orthonormal?

Write a function for PCA.

Input: X (the data matrix).

Output: SCR (a matrix containing the PCA scores),
LDS (the loading matrix),
VAR (the variance per PC).

Question 2.5.5: Make a Scree plot for the iris data: i.e. plot the variance explained as function of the number of PCs used. How many PCs are necessary to capture the essential information in the iris data set?

Question 2.5.6: Perform PCA on:

1. the original (unscaled) iris data;
2. the mean-centered iris data;
3. the auto-scaled iris data.

Compare the explained variances for the three different types of scaling. Which type of scaling leads to the highest variance for the first PC? Explain why!

Question 2.5.7: Plot the loadings for the three different scalings in separate bar plots. Which of the four variables are most important in PC 1 in each case? Can you explain the order of variable importance for each type of scaling?

Question 2.5.8: Thus far, you applied PCA to the entire iris data matrix. Now, create two matrices: matrix (**X1**) containing the odd rows of the iris data and the other (**X2**) containing the even rows (see the appendices!). Apply PCA to the auto-scaled **X1** matrix, to obtain loading vectors that span the space of **X1**.

Project the objects of **X2** into that space by right-multiplying them with the loadings you obtained for **X1**. Use the proper way of auto-scaling for matrix **X2**! This projection will result in scores for the objects in **X2** in the PC-space corresponding to the matrix **X1**.

Next, plot the first two score vectors against each other, where the objects from **X1** are shown in a different colour or plotting symbols than the objects from **X2**.

Question 2.5.9: Repeat the previous question, but now with **X1** containing the autoscaled rows 1 up to 50 of the original iris matrix, whereas **X2** contains the remaining rows: 51 up to 150. Compare the loadings of this PCA model with the loadings of the PCA model from **2.5.8**. Can you explain the differences between the PCA models? Which of the two models is most like the PCA model obtained from the complete data set?

Exercise 2.6 – The Biplot

Products:

- a function generating a score plot;
- a function that creates a loading plot;
- a biplot function.

Useful Matlab commands: min, max, abs, scatter, hold
Useful R commands: min, max, abs

Question 2.6.1: Again, make a score-plot for PC 1 against PC 2 for the meancentered iris data set. Label the axes with the percentages of variance explained by the individual PCs.

Write a score plot function for two arbitrary PCs. It should label the axes with the PC numbers and show the percentage of variance explained (see the appendices for useful examples).

Input: SCR (the score matrix,)
C1 and C2 (column identifiers),
VAR (the PC variances),
Class vector (for colour and/or plotting symbols).

Output: None.

Question 2.6.2: Create a loading plot for PC 1 and PC 2 in a new figure. By convention, loadings are indicated by arrows or lines from the origin (see the examples in the reader). You may label every object in the loading plot with variable name or variable number. Again, use appropriate axis labels. Which variables show the highest correlation? And does this correspond with your earlier findings?

Write a loading plot function for two arbitrary PCs. It should label the axes with the PC numbers and show the percentage of variance explained (see the appendices for useful examples).

Input: LDS (the loading matrix),
C1 and C2 (column identifiers),
VAR (the PC variances).

Output: None.

Question 2.6.3: Make a biplot by superimposing the score plot and the loading plot. You will need to scale either scores or loadings to get both types of information in the same figure; when comparing objects with variables, the main information is in the angles, rather than in the scale.
Compare your function with the biplot function from Matlab or R.

Write a biplot function for two arbitrary PCs. It should combine the properties of your score and loading plotting functions

Input: SCR (the score matrix),
LDS (the loading matrix),
C1 and C2 (column identifiers),
VAR (the PC variances),
Class vector (for colour and/or plotting symbols).

Output: None.

Question 2.6.4: Use your biplot function to investigate the auto-scaled iris data. Do you observe any differences, in particular in the grouping of patterns present in the biplot? Which type of scaling do you recommend for the iris data?

Exercise 2.7 – Reconstruction

Product: a function that generates the reconstructed matrix, the error matrix **E** and the total error.

Useful Matlab commands: sqrt
Useful R commands: sqrt

Question 2.7.1: For the auto-scaled iris data set, calculate the reconstructed matrix $\tilde{\mathbf{X}}$ from PC 1 and PC 2. Calculate the error matrix **E**. Visualize this matrix. What do you observe? Is there any structure present in the **E** matrix? Also calculate the square root of the sum of the squared elements in this **E** matrix: the total error. Are these two PCs sufficient for a proper reconstruction of the original data matrix **X**?

Write a function which calculates the reconstruction error matrix. This function must also return the normalized total error.

Input: X (the data matrix),
SCR (the scores),
LDS (the loadings),
a (the number of PCs used for the reconstruction).

Output: Xtilde (the reconstructed matrix),
ERR (the error matrix),
TE (the total error).

Question 2.7.2: Reconstruct the data matrix using 1, 2, 3, and 4 PCs, respectively. Make a bar plot showing the total errors for each of the reconstructions. How many PCs are optimal?

3 Cluster analysis

Exercise 3.1 – The distance matrix

Product: a function which calculates the Euclidean distances between the rows of a matrix X.

Useful Matlab commands: pdist (NB:the ouput is a vector instead of a symmetric matrix), reshape
Useful R commands: dist

Question 3.1.1: Use the formula, given in Table 3.1, to calculate the Euclidean distance between sample 1 and 2 of the iris data, and also between sample 1 and 100. Which of these two pairs of samples are most similar?

Question 3.1.2: In many clustering algorithms, a distance matrix is used. Such a (symmetric) matrix contains the distances between all pair of objects in a data set, i.e. at position (3,6) and (6,3) of the matrix, the distance between objects 3 and 6 is located. Investigation of such a distance matrix can lead to the grouping of similar objects.

Write a function that calculates the Euclidean distance matrix for any given data set.

Input: X (the data matrix).

Output: D (the distance matrix).

Calculate the distance matrix for the iris data set. Verify your answers from question 3.1.1.

Question 3.1.3: Visualize the distance matrix in an image. What can you say about the similarities between objects, given this image? Can you already say something about a grouping of samples?

Question 3.1.4: Calculate the distance matrix for $\mathbf{X}^\mathbf{T}$. What does it tell you?

Exercise 3.2 – Hierarchical clustering by Single Linkage

Useful Matlab commands: see the functions provided on the course web page
Useful R commands: hclust, cutree

Question 3.2.1: Hierarchical clustering starts by assigning every object to a separate cluster. In every iteration, the two most similar clusters are merged. Calculate the distance matrix for the columns of the iris data. Which variables are most similar? Consider the combination of these two as a cluster. Recalculate the distance matrix using the single linkage criterion, so that the distance matrix now reflects the distances between two original variables and the first cluster. Again, find the closest pair, merge them into a cluster, and repeat until only one cluster is present. You have just performed your first single linkage clustering!

Question 3.2.2: Now, perform single linkage on the transposed iris data set using the Matlab/R functions, and show the dendrogram. Does the order in which variables are merged correspond to your answers in the previous question?

Question 3.2.3: Visualise the dendrogram of the single linkage clustering of the *rows* of the iris data. Where would you cut it? What is the optimal number of clusters according to the dendrogram?

Question 3.2.4: Cluster the iris data (the rows!) into 2, 3, 4, 5 and 6 clusters, respectively. Visualize the results using PCA, with different colours for each cluster. What is the optimal number of clusters, according to these plots? Does this correspond with the structure of the dendrogram?

Exercise 3.3 – Complete and Average Linkage clustering

Product: a function that calculates the ‘cross-table’

Question 3.3.1: Visualize the dendrogram for complete and average linkage. What is the optimal number of clusters?

Question 3.3.2: Visualize the complete and average linkage results with PCA for 2-6 clusters. Compare these with single linkage (with equal numbers of clusters). What are the major differences?

Question 3.3.3: Compare the results of the hierarchical clustering methods with the true types of species, using three clusters. Which linkage strategy obtains the best results? Can you explain the behaviour of the individual methods?

Question 3.3.4: Consider the three-cluster solutions for single and complete linkage. How many objects, assigned to cluster 1 in single linkage are also assigned to cluster 1 in complete linkage? And how many of the objects, assigned to cluster 1 in single linkage, are assigned to cluster 2 in complete linkage? Complete the three-times-three matrix giving all possibilities. Such a matrix is called a ‘cross-table’(or confusion matrix).

Write a function which calculates and displays the ‘cross-table’ for two different clustering methods.

Input: LAB1 (labels of the objects according to clustering method A),
LAB2 (labels of the objects according to clustering method B).

Output: CT (the cross-table).

Question 3.3.5: Use your cross-table function to compare the clustering results of all methods with the true iris classes. Which clustering seems to reproduce the true labels best? Does this correspond with your answer in 3.3.3?

Question 3.3.6: The cross-table does not need to be square: calculate the cross-table for the comparison of the true labels and the four-cluster complete-linkage solution. Is the agreement better or worse than with three clusters?

Exercise 3.4 – The effect of outliers

Question 3.4.1: In the (contaminated) data set iris2 several objects have been replaced by outliers. Visualize the dendrograms for the single, complete and average linkage algorithms, applied on the iris2 data. How many clusters are required for the optimal partitioning? Show the PCA plots with colours indicating cluster labels.

Question 3.4.2: If you did not pick three as the optimal number of clusters, repeat the above with three clusters. As you probably can see, outliers are grouped in separate clusters. Is this logical? What would you expect if the data were partitioned in more than three clusters?

Question 3.4.3: Visualize the clustering results of single, complete and average linkage with six clusters and compare this with the three-cluster results of the original iris data. What are the differences? What conclusion can be drawn, concerning outliers?

Exercise 3.5 – k-means clustering

Useful Matlab commands: randperm
Useful R commands: sample

Products:
- a function which performs k-means clustering;
- a function that calculates the sum of squared distances.

Question 3.5.1: To obtain the initial cluster centers in k-means, use one of the two possible approaches: random selection of objects, or random values. The algorithm should iterate until the cluster centers - or the labels - do not change anymore.

Implement the k-means algorithm.

Input: X (the data matrix),
k (the number of clusters.

Output: C (a matrix containing the k cluster centroids),
L (a vector containing the class labels for the objects in X).

Perform k-means on the iris data using three clusters. Visualise the result in a PCA plot.

Question 3.5.2: The quality of the k-means clustering can be assessed by considering the overall distance to the respective centroids. Calculate the SSD of your three-cluster solution. Repeat the k-means clustering, visualise the result, and again calculate the SSD. Compare the two clusterings in a cross-table. What do you observe?

Question 3.5.3: For 2 to 10 clusters, perform k-means 100 times (or tell the computer to do this for you). Evaluate the quality of the different clusterings. Can you use this to determine the optimal number of clusters?

Write a function that calculates the sum of squared Euclidean distances between the objects and the corresponding cluster centroids.

Input: X (the data matrix),
L (the class label vector),
C (the centroid matrix).

Output: SSD (the sum of squared distances).

Exercise 3.6 – The effect of outliers on k-means clustering

Question 3.6.1: Use the iris2 data to obtain a partitioning of the data with three clusters. Apply k-means 100 times and evaluate the quality of the clusterings. Visualize the (best) results.

Question 3.6.2: What do you expect to happen when the data is grouped in more than three clusters? Obtain 100 partitionings with 6 clusters and calculate the average SSD value. What are the differences with the three-cluster situation? What conclusion can be drawn, concerning outliers?

Exercise 3.7 – The effect of scaling on the clustering

Question 3.7.1: Repeat the clustering analysis for the (original) auto-scaled iris data. Obtain the three-cluster results only. Apply k-means 100 times and select the best clustering. What are the differences with the clustering obtained by using the raw data?
Have the results improved after auto-scaling? What do you expect if the meancentered data are used?

4 Classification

Exercise 4.1 – Making a training and test set

Product: A function that generates randomly sampled training and test sets.

Useful Matlab commands: sort
Useful R commands: sort

Question 4.1.1: Construct a random training set of 75 objects from the iris data that will be used to set up the class characteristics. The remainder of the data will be used as a test set. Make a PCA plot of the training set, colour the objects according to their labelling. Project the test set and show the locations of the objects in your plot. Is the training set representative for the entire data set?

Question 4.1.2: Repeat this for small training sets, e.g. 10 objects. What do you expect? Repeat this several times, and monitor the "representativeness" of the training set. How would you counter the specific danger of this approach?

Write a function that divides a data set in randomly chosen training and test sets, given a number of objects in the training set, and taking into account class representation.

Input: X (data matrix),
C (vector of class labels),
P (percentage of training objects).

Output: Xtrain, Xtest (training and test sets),
Ctrain, Ctest (vectors of class labels for the two sets).

Now, construct a training set and a test set for later use (so make sure you save it in a file!); pick an appropriate size.

Exercise 4.2 – Maximum-Likelihood Linear Discriminant Analysis

Products:
- a function that calculates the pooled covariance matrix;
- a function that calculates the Mahalanobis distance;
- a function that performs ML-LDA;
- a function that performs QDA;
- a function that calculates the classification accuracy from a confusion matrix.

Question 4.2.1: Calculate the mean and covariance matrix of each class in your iris training set. Calculate the pooled covariance matrix.

Write a function which calculates the pooled covariance matrix from a data matrix and the corresponding class label vector.

Input: X (the data matrix),
L (the class label vector).

Output: CP (the pooled variance-covariance matrix).

Question 4.2.2: Calculate the Mahalanobis distance between the first object in the iris test set and the mean of each class, obtained from the training set; use the variance-covariance matrices calculated from the individual classes. For which class does this object have the highest similarity? Does this correspond with the true class of this object? Repeat this using the pooled covariance matrix. Are there any differences?

Write a function that calculates the Mahalanobis distance of objects in matrix X to a given point (vector Y, e.g. a class mean). The covariance matrix may for instance be the covariance matrix of the class with center Y, or a pooled covariance matrix.

Input: X (the data matrix),
Y (the target),
C (a variance-covariance matrix).

Output: D (a vector containing the Mahalanobis distances).

Write the ML-LDA function assigning every object of a test set to the most similar class in the corresponding training set.

Input: X1 (training data),
L1 (training labels),
X2 (test data).

Output: PL (predicted labels for the test set).

Write the QDA function assigning every object of a test set to the most similar class in the corresponding training set.

Input: X1 (training data),
L1 (training labels),
X2 (test data).

Output: PL (predicted labels for the test set).

Question 4.2.3: Calculate the classification of the objects in the test set using MLLDA and QDA; are there any objects whose classification differs? (Remember the cross table from the merry days of clustering? In the context of classification, it is called a *confusion matrix*.) Use the same functions to calculate the classification of the objects in the training set. Which set is predicted best? Which class is predicted best? Which class is worst?

Question 4.2.4: From the confusion matrix, one can easily calculate the performance – remember, in the case of classification one *knows* the class labels. Propose a performance measure that gives the percentage of correct classifications.

Modify the function from question 3.3.4 to calculate also the percentage of correct classifications, given a confusion matrix.

Input: LAB1 (labels of the objects according to clustering method A),
LAB2 (labels of the objects according to clustering method B).

Output: CT (the confusion matrix).
PCCD2 (the performance).

Question 4.2.5: Plot the misclassifications in a PCA plot for both ML-LDA and QDA.

Exercise 4.3 – Fisher Discriminant Analysis

Product: The Fisher LDA function.

Question 4.3.1: Use the equations given in the reader to calculate the within- and between-group sums of squares for the training set of the iris data. Calculate the first singular vector of $\mathbf{W}^{-1}\mathbf{B}$ and use it to calculate the discriminant scores.

Question 4.3.2: Determine the differences between the discriminant score for the first object in the iris data set and the discriminant scores for each class mean.
Which class mean has a discriminant score closest to the score of this particular object? Does this correspond to the true class type?

Write the QDA function assigning every object of a test set to the most similar class in the corresponding training set.

Input: X1 (training data),
L1 (training labels),
X2 (test data).

Output: PL (predicted labels for the test set).
Note: this function will be provided.

Question 4.3.3: Use your Fisher LDA function to determine the class labels of the training and test set. Visualize the results in a plot and compare the classes with the true class type.

Exercise 4.4 – K-Nearest Neighbours

Product: The kNN classification function.

Question 4.4.1: The first step in kNN classification is to calculate the distance between the unknown object and every object in the training set. Therefore, calculate the (Euclidean) distance between the first object of the iris test set and the objects in the training set, and determine the closest neighbour. To which class does this object belong? Does this result change if you consider the class type of three neighbours? What happens if you consider five neighbours?

Write a function that assigns every object of a test set to the most similar class of the training set by means of the kNN algorithm.

Input: X1 (training data),
L1 (training labels),
X2 (test data),
k (the number of nearest neighbours to take into account).

Output: PL (predicted labels for the test set).

Question 4.4.2: Classify the objects in the training set and the test set using 3NN. Compare the classification results with the true classes, by a visual inspection of the confusion matrix, and calculate the performance. Visualise the misclassifications in a PCA score plot. Repeat this for 1 and 5 neighbours, respectively. Which setting gives the best results?

Exercise 4.5 – Cross-validation

Product: A function that performs leave-one-out cross-validation.

Question 4.5.1: Predict the class of the first object in the iris training set, based on the remaining objects of the training set, using 3NN. Compare the prediction with the true class. Repeat this for the first 10 objects. This is boring, so write a function to do this…

Implement a function to perform leave-one-out (LOO) cross-validation suitable for the ML-LDA, QDA and kNN classification functions. The function should return the confusion matrix.

Input: X (the data matrix),
L (the class label vector),
T (the classifier type, T = 1: LDA, T=2: QDA etc.),
k (optional, required for kNN).

Output: C (the confusion matrix),
PCCD2XY (the classification performance).

Question 4.5.2: Perform cross-validation for the classification methods that are studied in these exercises. Use the entire data matrix! Compare the LOO classification performance to the performance estimated for the training / test set division.

Question 4.5.3: Now you are able to optimize the k value of kNN in a statistically sound way. Apply leave-one-out cross-validation to the *training set* for k = 2 up to k =10. Select the k value with the highest cross-validation performance. Use this k value to estimate the prediction error for the *test set*. Is this outcome in agreement with the performance estimated from the training set / test set divisions?

Question 4.5.4: Which of the classification methods performs best for the *wine* data? That is, which method do you expect to have the highest prediction performance for *newly measured* wines? What do you expect the performance to be in each case?

5 Multivariate Regression

Exercise 5.1 – Principal Component Regression

Product:

- the PCR model building function;
- the PCR prediction function;
- the RMSE function.

Question 5.1.1: In this exercise you will use the *stackloss* data set (use the sets 'stackloss' and 'stacklossy' as **X** and **Y**, respectively). Perform PCA on the meancentered **X**, and make a score-plot for the first two PCs. Is there a correlation between the PCs and the y-values? Based on the scores, do you expect to be able to predict **Y** from the scores?

Question 5.1.2: Perform a linear regression of **Y** on the first two PCs. Transform the regression coefficients back to the original **X** domain. Add an abscissa b_0 so that your model can predict the original data, i.e. data that have not been mean-centered. This is called a Principal Component Regression (PCR). Compare the regression coefficients with the numbers in the reader. Inspect the performance of the model by plotting the actual output values versus the predicted ones.

Implement a function that performs Principal Component Regression.

Input: X (data matrix),
Y (y-values),
PC (number of PCs).

Output: B (regression coefficients).

Question 5.1.3: Make a PCR model with three PCs and compare the regression coefficients with the regression coefficients of a MLR model (given by Equation 5.2 in the reader).

Question 5.1.4: B can be used to predict the y-values of unknown samples. When do you expect **B** to be a vector? When is it a matrix?

Implement a function that predicts y-values for new data.

Input: X (data matrix),
B (regression coefficients).

Output: YP (predicted y-values).

Question 5.1.5: The performance of the regression model can be determined by calculating the Root Mean Square Error (RMSE) between the true and predicted output values. The name is actually very good: the errors are squared, then summed, and divided by the number of predictions which leads to the mean squared error.
Finally, we take the root. In a formula:

$$RMSE = \sqrt{\frac{1}{n}\sum e_i^2}$$

Write a function that calculates the RMSE for the true and predicted output values.

Input: Y (true values),
YP (predicted values).

Output: RMSE (the error estimate).

Exercise 5.2 – Partial Least Squares regression

Product: the PLS function.

Question 5.2.1: Using again the stackloss data, calculate the **X** and **Y** loadings for the first latent variable in PLS regression. Also calculate the scores for both **X** and **Y**. Compare the **X**-scores with the PCR-scores on the first PC. Do the same for the loadings. Are all input variables equally important in these PLS and PCR models?

Write a function that returns the regression coefficients B for PLS, given a data matrix X with the corresponding Y, and a number of Latent Variables (LVs).

Input: X (data matrix),
Y (y-values),
LV (number of LVs).

Output: B (regression coefficients).

Question 5.2.1: Calculate **B** for the stackloss data, for 1, 2 and 3 latent variables. Check your results with the reader! Compare your results to the PCR coefficients.

Modify your own PLS function, so that it returns the scores and loadings of X and Y as well.

Input: X (data matrix),
Y (y-values),
LV (number of LVs).

Output: B (regression coefficients),
SCX (X scores),
SCY (Y scores),
LX (X loadings),
LY (Y loadings).

Exercise 5.3 – Application to the soil data

Question 5.3.1: Examine the soil data. You can start by answering the following questions:
- Which metal concentrations are highly correlated (say, with $r > 0.95$)?
- Which metals are not highly correlated (say, with $r < 0.8$)?
- Which other parameters show high correlations?
- Which parameters show negative correlations? Can you explain this?

Question 5.3.2: Select the variables Ni, Cu and Zn to predict the Cd concentrations by applying PCR and PLS on this subset after mean-centering of your data. Just to be perfectly clear, **X** will contain the Ni, Cu and Zn concentrations, while **Y** contains only the Cd concentrations. If you have implemented PCR and PLS correctly you will obtain the following **B**-matrices for PCR and PLS by selecting the first two PCs and LVs:

B-matrix PCR

	1 PC	**2 PCs**
b_0	0.311	-0.443
b_1	0.001	0.019
b_2	0.001	0.031
b_3	0.008	0.002

B-matrix PLS

	1 LV	**2 LVs**
b_0	0.310	-0.221
b_1	0.001	0.011
b_2	0.001	0.038
b_3	0.008	0.001

Do you have any explanation why PCR and PLS result in almost the same **B** coefficients for one PC/LV? Which of the two models obtains the smallest RMSE value? How many latent variables or PCs do you need?

Question 5.3.3: Examine these PCR and PLS models:
- Which objects are predicted well and which bad? Make a plot to show this.
- Inspect the loadings and interpret these.

Exercise 5.4 – Prediction Error Estimates

Product: a function returning the RMSECV for a leave-one-out cross-validation

Question 5.4.1: Split the soil data set into a training and test set considering the following samples:

Test set: ***objects 1 to 25***
Training set: ***objects 26 to 54***

Are the samples of the test set well represented by the training set?

Build PCR and PLS models for one to six PCs / LVs (so, twelve models in total) by selecting the variables Ni, Cu, Zn, Zn1, Pb, Se, Organic matter, moisture initial and moisture content after 5 month storage to predict cadmium concentrations. For every model, make a plot of the true **Y** versus the predicted **Y**. What do you observe?

Also calculate the RMSEC (the root mean square error of calibration, the RMSE of the training set) and the RMSEP (the root mean square error of prediction, based on the test set) for one to six PCs / LVs. Make a plot of the 6 RMSEC / RMSEP values for PCR and PLS, respectively. Why do the plots look different? Based on these plots, what do you consider to be the optimal number of PCs and LVs?

Question 5.4.2: The previous question demonstrated that the prediction error can increase when taking too many PCs/LVs. This is called overtraining. What do you think causes this effect? To prevent overtraining, we need to select the optimal number of PCs or LVs. In the reader it is shown that these numbers can be determined by plotting the number of PCs (or LVs) against the RMSE of Cross-Validation (RMSECV). Instead of leaving out a complete test set, we can leave out every object once, and continue until all objects have been used as test objects.

Write a general function that returns the RMSECV for PCR or PLS, based on leave-one-out cross-validation. Make use of your own PCR, PLS and RMSE functions.

Input: X (data matrix),
Y (y-values),
LV (number of LVs).

Output: RMSECV (the RMSE cross-validation errors).

Exercise 5.5 – A comparison between PCR and PLS

Question 5.5.1: So far, we have not seen the true strength of multivariate regression methods: the number of objects still was larger than the number of variables. Let us now look at the situation where the number of variables is much larger than the number of objects, something that often happens when we want to predict properties or concentrations from spectra. Load the spectra for the soil data; use this as new **X**, and use columns from the soil data as **Y**. Create suitable training and test sets. What does "suitable" mean in the case of regression?

Question 5.5.2: Predict the Cd and Zn metal concentrations using two separate PCR models. How many PCs do you need and what are the RMSECV and RMSEP values of the best model of those metal concentrations? Next, predict the Cd and Zn concentrations together. What do you observe? Can you explain this? Do you prefer to predict both metal concentrations separately or together?

Question 5.5.3: Repeat the PCR regression after auto-scaling of the data. Which scaling method results in the best regression model? How would you validate the optimal choice of scaling?

Question 5.5.4: Repeat the previous questions for PLS. Which difference between PLS and PCR is immediately obvious? Which method is best for this data set?

Appendix A – Matlab examples

Note: for debugging your functions you can use the debugging facilities of Matlab (e.g. through the Matlab editor set breakpoints in your code by clicking beside a line number; then during execution the function halts at that point and you can inspect the values of the variables and you can continue line by line; see for more the Matlab Help), or you can simply print values of variables, calculated during the execution of your function, by removing the semicolon after each line, use the disp() command and/or copy the content of you function line by line, loop by loop, etc., to the command line. A function or script can be stopped temporarily with the 'pause' command.

Load the iris data and the corresponding class labels.

```
>> load iris.dat
>> load irisclass.dat
```

This set of commands plots sepal length (variable 1) versus petal length (variable 3), adds per point in the plot the species type and provides some figure labels.

```
>> figure;
>> plot(iris(:,1), iris(:,3), '+');
>> text(iris(:,1), iris(:,3), num2str(irisclass));
>> xlabel('sepal length');
>> ylabel('petal length');
>> title('iris data');
```

This example generates a plot of the iris data set (again, for variable 1 vs 3), but now colors each point in the plot according to the specific species type (note that Matlab automatically assigns colors to the points).

```
>> plot([iris(1:50,1) iris(51:100,1) iris(101:150,1)], ...
[iris(1:50,3) iris(51:100,3) iris(101:150,3)], '+');
>> xlabel('sepal length');
>> ylabel('petal length');
>> title('iris data');
```

Because the species types in the iris data set are sorted, it is easy to assign different colors to the different species types. If a data set is not sorted, the appropriate class label for each sample should first be determined.

```
>> class1 = find(irisclass == 1);
>> class2 = find(irisclass == 2);
>> class3 = find(irisclass == 3);
>> figure;
>> hold on
>> plot(iris(class1,1), iris(class1,3), '+r');
>> plot(iris(class2,1), iris(class2,3), '+g');
>> plot(iris(class3,1), iris(class3,3), '+b');
>> legend('class 1', 'class 2', 'class 3');
>> xlabel('sepal length');
>> ylabel('petal length');
>> title('iris data');
```

How the 'ones' command is used for fast mean-centering. The last line refers to the calculation of the variance-covariance matrix of the mean-centered data matrix.

```
>> [m,n] = size(iris);
>> mn_iris = mean(iris, 1);
>> ones_mat = ones(m,1);
>> mean_mat = mn_iris(ones_mat,:);
>> mc_iris = iris – mean_mat;
>> cov_iris = (mc_iris' * mc_iris) / (m-1);
```

*Divide a data matrix **X** in two new matrices which contain the odd rows (matrix **X1**) and even rows (**X2**) of matrix **X**, respectively.*

```
>> [m,n] = size(X);
>> ind_odd = [1:2:m];
>> ind_even = [2:2:m];
>> X1 = X(ind_odd,:);
>> X2 = X(ind_even,:);
```

*To plot lines from the origin to points in space, specified by two columns of **X** (here, the columns 2 and 6 are selected), use the following plotting command.*

```
>> [n,m] = size(X);
>> zero = zeros(n, 1);
>> plot([zero X(:,2)]',[zero X(:,6)]');
```

Display the amount of variance, from Matlab variable PVAR, on the x-axis.

```
>> xlabel(['PC 1 (' num2str(PVAR(1,1)) ' % variance)']);
```

Remember that you should make general applicable functions. An example of a (start of a) generalized function to plot your data for a given set of 2 variables, related to the set of commands given above in order to plots sepal length (variable 1) versus petal length (variable 3):

```
        function plotvars(X, class, var1, var2, labels)
        %       plotvars(X, class, var1, var2, labels)
        % comments (input and output argument explanation and restrictions
        % on the input arguments (e.g. string, matrix, number), goal / task
        % of function, authors, date)

        figure
        plot( . . ) % complete the arguments yourself
        % etc.
        end
```

The function related to Exercise 2.1.2/3 can then be:

```
function C1_Plot(X, var1, var2, Class, Xlabel, Ylabel)
%
% C1_Plot2(X, var1, var2, Class, Xlabel, YLabel)
%
% Plots two columns of data matrix X
%
%
% INPUT:
% X         - data
% var1        - index of first column to be plotted
% var2        - index of second column to be plotted
% Class     - vector with class information of objects in X
%             (e.g. [1 1 2 2 .. etc.]) (optional)
% Xlabel    - label on x-axis (optional)
% Ylabel    - label on y-axis (optional)
%
% OUTPUT:
%

if nargin < 6
    Xlabel = ['Variable ', int2str(var1)];
end
if nargin < 5
    Ylabel = ['Variable ', int2str(var2)];
end
if nargin < 4
    Class = ones(1,size(X,1));
end

% check class information:
Classes = unique(Class);
nr_classes = length(Classes);

% generate class labels:
MStyle_unique                                                =
['b+';'ro';'g*';'cs';'md';'yx';'k.';'b^';'rv';'g>';'c<';'mp
';'kh'];

% generate plot:
figure;
hold on
for k= 1:nr_classes
```

```
        group_nr = Classes(k);
        pos_group = find(Class == group_nr);
        plot(X(pos_group,var1),              X(pos_group,var2),
    MStyle_unique(k,:));
    end
    % as simple alternative also gscatter can be used for above
    for-loop
    % gscatter(X(:,var1), X(:,var2), Class)

    xlabel(Xlabel);
    ylabel(Ylabel);

    return
```

A pairsplot function that generalizes the afore mentioned function for all combinations of variables is given below. The framework of this function can also be used to make score plots for different combinations of latent variables.

```
function C1_pairsplot(Data, Class_info, Number_samples)

%
%       C1_pairsplot(Data, Class_info, Number_samples)
%
% Plots the variables as pairs-plots of all possible variable
% combinations and labels/colors the datapoints according to the
% different classes.
% Assumed is that the variables are in the columns.
% The data will not be scaled but impicitly it still is due to the
% visual scaling by the matlab plot-command.
%
% input:
% Data         :  Data, rows are samples
% Class_info   :  vector with class numbers, If absent: all same class.
% Number_samples : 1 of 0: to number samples in de plots
%                  Default 0
%
% If required execute after the plot:
%         legend('location', 'NorthEastOutside')
%

[nr, nc] = size(Data);

if nargin < 2
    Class_info = ones(nr,1);
end

if nargin < 3
    Number_samples = 0;
end

Classes = unique(Class_info);
nr_classes = length(Classes);

% generate different labels:
```

```
CMap      = colormap(jet(2*nr_classes));
MStyle_unique = ['+','o','*','s','d','x','.','^','v','>','<','p','h']
MStyle  =  repmat(MStyle_unique,  ceil(nr_classes/length(MStyle_unique)),
1);

% subplot
% figure
% Not completely safe:
for k=1 : nc-1
    for l=k+1 : nc
        subplot(nc-1,nc-1,((k-1)*(nc-1) + l-1))
        for class_count = 1:nr_classes
            group_nr = Classes(class_count);
            pos_group = find(Class_info == group_nr);
            Plotvar1 = k;
            Plotvar2 = l;
            h1   =   plot(Data(pos_group,   Plotvar1),   Data(pos_group,
Plotvar2), ...
                MStyle(group_nr), 'Color', CMap((2*group_nr),:));
            hold on
            % number objects
            if Number_samples
                % text offfset berekenen:
                minscores1stvar = min(Data(pos_group, Plotvar1));
                maxscores1stvar = max(Data(pos_group, Plotvar1));
                textoffset= (maxscores1stvar - minscores1stvar)/20;
                for m=1:length(pos_group)
                    text(Data(pos_group(m),        Plotvar1)+textoffset,
Data(pos_group(m), Plotvar2), ....
                        int2str(pos_group(m)),                 'Color',
CMap((2*group_nr),:))
                end
            end
        %   title(['PC    ',   int2str(k),'-',   int2str(l),    '    ',
num2str(PCA_varperc(k)) '%',num2str(PCA_varperc(l)) '%' ])
            xlabel(['variable ' int2str(k) ])
            ylabel(['variable ' int2str(l) ])
        end
    end
end
% 16/2/2011: legend with last figure: seems not to work well.
% Execute afterwards.
% legend('location', 'NorthEastOutside')
disp('If   required   execute   after   the   plot:   legend(''location'',
''NorthEastOutside'')')
```

A barplot function (Exercise 2.2.2) that plots the mean +- one standard deviation for each variable, is given below:

```
function BarPlot(X, Title, Xlabel, Ylabel)
%
% BarPlot(X, Title, Xlabel, YLabel)
%
% Generates a barplot of the mean of data matrix X, including the
% standard deviations.
```

```
%
%
% INPUT:
% X          - data
% Title      - title (optional)
% Xlabel     - label on x-axis (optional)
% Ylabel     - label on y-axis (optional)
%
% OUTPUT:
%

if nargin < 4
    Ylabel=[];
end
if nargin < 3
    Xlabel=['Variable number'];
end
if nargin < 2
    Title=['Bar plot of mean +- standard deviation of each variable'];
end


mean_cols = mean(X, 1);
std_cols = std(X, 0, 1);

figure;
hold on
bar(mean_cols + std_cols, 'w');
bar(mean_cols, 'b');
bar(mean_cols - std_cols, 'b');
title(Title)
xlabel(Xlabel);
ylabel(Ylabel);

return
```