Statistical Machine Learning 2018
Frederik Hagelskjær and Norbert Kruger

Exercise Sheet 2: Data Preprocessing

Introduction

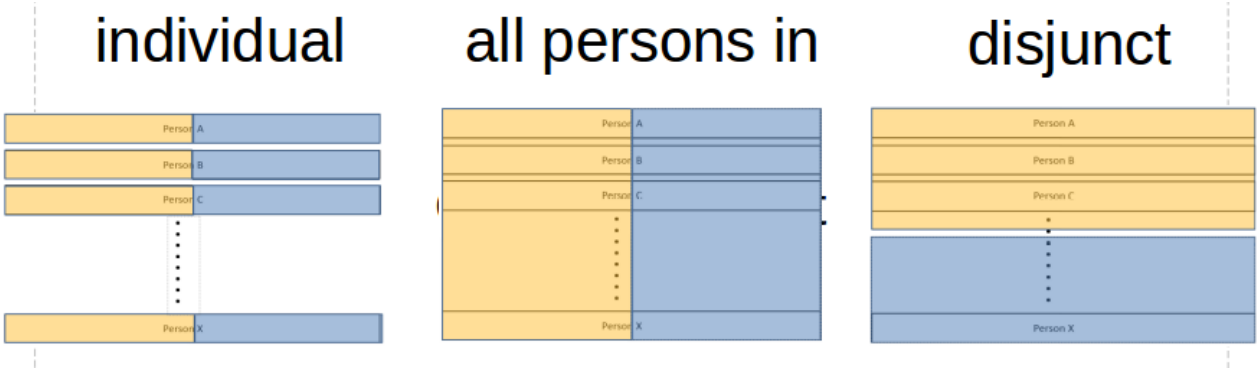
The goal of this exercise is to deepen your understanding of PCA as a tool for compressing the size of data. Furthermore, the exercise should enable you to better understand the effects of normalization, but in the report you should also discuss when normalization is important in general.

It is important that you have an understanding of the theory behind the algorithms and you are able to describe the advantages and disadvantages of the different techniques.

During analysis of execution times, it is important to consider what part of the execution time would be an issue in real world analysis.

You are also welcome to use preprocessing, you have determined in previous exercises, such as smoothing and down sampling. But remember to specify the methods you have used.

To improve the results when using data from different people, you can consider thresholding the images into binary images, since this should improve the similarity of people drawing with differently saturated colors.



In R various functions can simplify the implementations such as:

prcomp: a simple tool for performing Eigenvalue decomposition PCA

- Remember to use ?prcomp

- prcomp gives both the new values (scores), the PCA parameters (scaling, centering and loadings) and the eigenvalues
- you can use “predict” to evaluate new data with the found rotations

min, max, mean, sd: Simple tools for determining min,max, mean and the standard deviation. This should make normalization easy to implement.

Exercise 2.1: Principal Component Analysis (PCA)

Perform a PCA on the data for the “all persons in” and “disjunct” data set.

1.1 Show the standard deviation (From prcomp Eigenvalues), the proportion of variance and the accumulated proportion of variance of the principal components. (In the report the first 10-20 principal components, should be sufficient to illustrate the tendencies.)

1.2 Show the performance of selecting enough principal components to represent 80%, 90%, 95%, 99% of the accumulated variance. For each test vary “k” in kNN, try 3 reasonable values.

1.3 Measure run times for the prediction step of the kNN-classifier with PCA based dimensionality reduction. How does the feature vector dimensionality effect performance.

1.4 Interpret the results.

Exercise 2.2: Normalization

2.1 Perform one of the two normalizations suggested in the lecture (min-max normalization and z-score standardization) for the best parameter setting found under 2.1.3 and apply kNN with 10 fold cross-validation (10 runs, 90% training and 10% test set).
Apply the normalization before and after PCA independently and compare the results.

2.2 Analyze the results

Exercise 2.3: Reconstruction using PCA

3.1 This task is about reconstructing data using PCA. First using these functions we can plot an image of a single cipher (for plotting images do not convert **id** to data frame):

- id <- loadSinglePersonsData(**DPI**,**group**,**member**,folder)

- rotateSelf <- function(x) t(apply(x, 2, rev))
- imageSize <- sqrt(ncol(id) - 1)
- imageM <- matrix(id[**cipherNumber**,2:ncol(id)],nrow = imageSize,ncol = imageSize,byrow = FALSE)
- imageM <- rotateSelf(imageM) # rotate is a function to rotate the image
- image(imageM)

Plot one of each cipher.

3.2. Plot the first 10 eigenvectors/loadingvectors as images. Can you describe what you see?

(The first eigenvector is retrieved as id_pca$rotation[,1])

3.3. Plot a reconstruction of the images you displayed in 3.1 using all PC's. This can be done by multiplying the loadings with the scores and adding the removed centering.

- trunc <- pca_res$x[cipherNumber,1:nrow(pca_res$rotation)] %*% t(pca_res$rotation[,1:nrow(pca_res$rotation)])
- trunc <- scale(trunc, center = -1 * pca_res$center, scale=FALSE)

3.4. Now re-recreate using 80% of variance, 90% and 95%.

Can you describe what you see?

How much have you reduced the data size?

3.5. The last exercise concerns the comparison between two different ciphers. Take one instance of two different ciphers, (eg. 43 and 456 would give one '0' and one '1'), compare the 10 first scores and see if you can spot a difference. Try also this were you take the mean for all 400 instances of these ciphers and compare the first 10 scores. Can you spot a pattern when comparing with the loadings.