Data Analysis Methods 2015
Assignment 2

Aasa Feragen, Christian Igel

In Assignment 2 you will work with dimensionality reduction through PCA and its use for visualization of data and visualization of variance in visual data. Assignment 2 will be made available 23.2 and your report should be uploaded to Absalon no later than 8.3.

Guidelines for the assigment:

- **The assignments in DAM must be completed and written individually.** This means that your code and report must be written completely by yourself.
- You are, however, encouraged to discuss in small groups while working on the assignments, If you do this, please list your discussion partners in the beginning of your report.
- In DAM, we will be using an automatic code checking system for some of the exercises. These exercises are written in blue text below, and will be evaluated solely based on your uploaded code. Your solutions to these assignments should therefore be uploaded to the automatic code checking system. You can do this at any time, as many times as you want, until the deadline, and your auto-generated feedback will be immediately available.
- Upload your report as a single PDF file (no Word) named `firstname.lastname.pdf`.

Dimensionality reduction with PCA

In the first part of the assignment we will work with the DTU hand dataset, see the appendix.

Exercise 1 (Plotting hand shapes)**.** Plot one of the hands by plotting the landmark points and interpolating between subsequent landmark points.

Next, plot all the hands on top of each other. Can you see any dataset tendencies from this plot?

Deliverables. A plot of a hand, a plot of many hands, and a one-liner.

Exercise 2 (Performing PCA)**.**

a) Implement PCA using the supplied template function `pca.py`. Your function should return i) unit vectors spanning the principal components, ii) the variance captured by each of these components. Upload your code in the automatic code checking system.

b) Perform PCA on the dataset. Remember that each PC corresponds to an eigenvector of the covariance matrix, and that the corresponding eigenvalue corresponds to the variance of the data in the direction of the eigenvector. Make a plot of variance versus PC, where you should see the variance stabilizing (capturing primarily noise).

You can determine the cumulative normalized variance by normalizing the variance along all PCs such that the sum of all variances is 1, and then capture how large a proportion of the variance is described by the first, second, etc PC. Plot the cumulative variance versus the number of used PCs. How many PCs (dimensions) do you need to capture 90% of the variance in your dataset? 95%?

Deliverables. a) Uploaded code and b) Plot of variance versus PC; plot of cumulative variance versus PC; the numbers of dimensions needed to capture 90% and 95%.

Visualizing variance in visual data

Exercise 3. Now, you should visualize the variance of the hand-data by plotting some instances of the first three PCs. That is, if the mean of the data is given by m, you are going to plot the "hands"

$$\begin{array}{ccccc} m-2\sigma_1 e_1 & m-\sigma_1 e_1 & m & m+\sigma_1 e_1 & m+2\sigma_1 e_1 \\ m-2\sigma_2 e_2 & m-\sigma_2 e_2 & m & m+\sigma_2 e_2 & m+2\sigma_2 e_2 \\ m-2\sigma_3 e_3 & m-\sigma_3 e_3 & m & m+\sigma_3 e_3 & m+2\sigma_3 e_3. \end{array}$$

where the e_1, e_2 and e_3 are the eigenvectors defining the first three PCs, and σ_1, σ_2 and σ_3 denote the standard deviation of the data projected onto each of the first three PCs.

Plot the five hands corresponding to each PC in a single plot, and illustrate the temporal development with a changing color. This can, for instance, be done by importing a colormap with `blues = plt.get_cmap('Blues')`, where `blues(x)` returns a different shade of blue for every number x between 0 and 1.

Describe the variance captured by the three components.

Deliverables. Three plots with sequences of hands showing the variance. A description of the three components.

Classification with nearest neighbors

In the second part of Assignment 2, you will perform classification with the nearest neighbor classifier, which is a fundamental, non-linear, non-parametric method for classification.

Please download the datasets `ParkinsonsTrain.dt` and `ParkinsonsTest.dt` from Absalon, see description in the Appendix below.

Exercise 4 (k-NN Classification)**.**

a) Implement a k-nearest neighbor classifier (k-NN) using the supplied template function `knn.py`, which takes as input a training set, a test set, a set of training labels, and a parameter k, and outputs a vector of labels for the test set.

b) Apply your k-NN classifier using `ParkinsonsTrain.dt` as training data and report the accuracty (one minus the $0-1$ loss) of the corresponding classifier on both the training data and the test data in `ParkinsonsTest.dt` for $k = 1, 3, 5$.

Deliverables. a) Uploaded code; b) Training and test results of your k-NN classifier for $k = 1, 3, 5$; 2-3 lines of discussion of the results.

Data normalization

Data normalization is an important preprocessing step. A basic normalization is to generate zero-mean, unit variance input data.

Exercise 5 (Data normalization)**.**

a) Implement centering and normalization as described in Chapter 10 of the book [1], using the supplied template `cent_and_norm.py`. Your function should compute the mean and the variance of every input feature (i.e. of every component of the input vector) in your training set, and find the affine linear mapping $f_{\text{norm}} \colon \mathbb{R}^d \to \mathbb{R}^d$ that transforms the training data such that the mean and the variance of every feature in the transformed. In the supplied dataset $d = 22$, but please implement the function to work with any integer d. Your function should take both a training and test set as input. The mean and variance used for normalization should be computed *only* using the training set, but you should use these to normalize both the training and test sets. Upload your code in the automatic code checking system.

b) Center and normalize the training and test data in the Parkinsons dataset, and repeat the classification from Exercise 4.

Deliverables. a) Uploaded source code for the centering and normalization. b) Updated training and test results for the k-NN classifier for $k = 1, 3, 5$.

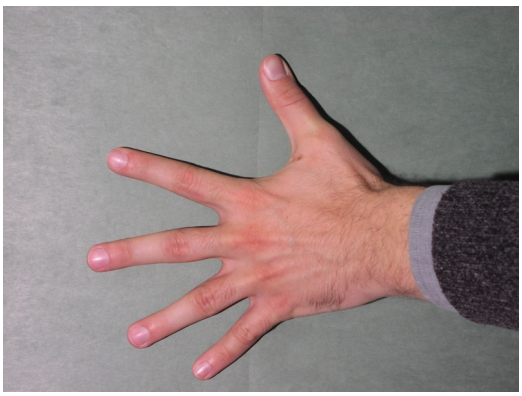

Figure 1: A hand from the DTU dataset.

Hyperparameter selection using cross validation

The performance of the nearest neighbor classifier depends on the choice of the parameter k determining the number of neighbors. Such a parameter of the learning *algorithm* (i.e. not a parameter of the resulting model) is called a *hyperparameter*.

Cross-validation is useful to determine proper hyperparameters.

Exercise 6 (Cross validation)**.** You are supposed to find a good value for k from $\{1, 3, 5, \ldots, 25\}$.

a) Implement 5-fold cross validation using the supplied template function `cv.py`. Your code should load a training set with the corresponding training set labels, estimate the performance of the k-NN classifier for every k using 5-fold cross validation, and output the k with the lowest average $0-1$ loss (classification error), which we will call k_{best} in the following. **NB!** Only use the *training data* in the cross validation process. Please upload your code to the automatic code checking system.

b) Please report your optimal parameter k_{best} on the centered and normalized training set `ParkinsonsTrain.dt`.

c) After you have determined k_{best}, please estimate the performance of the classifier using the centered and normalized test data `ParkinsonsTest.dt`.

Deliverables. a) Source code, b) found parameter k_{best}, c) training and test error percentage of k_{best}

As a general rule, you must not use the test data in the model building process at all (neither for training, data normalization, nor hyperparameter selection), because otherwise you may get a biased estimate of the generalization performance of the model.

The data material

The shape of hands

We will create a statistical model for hand shape using an annotated set showing hand shape. Download the DTU hand dataset from `http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=403`. This is a dataset consisting of $2D$ images of 40 hands, each annotated with 56 landmark points (x_i, y_i), $i = 1, \ldots, 56$. This means each hand is represented by a vector of dimension 112, and in the data material, each hand is parametrized as $(x_1 x_2, \ldots x_{56}, y_1, y_2, \ldots, y_{56})$. We will study the shape defined by these landmark points. They are found in the file `shapes.txt` in the `shapes` folder.

Application: Diagnosing Parkinson's disease voice signals

We consider the medical application of diagnosing Parkinson's disease from a person's voice. We consider the data from [2], which can be obtained from the well-known UCI benchmark repository [3].

The data were collected from 31 people, 23 suffering from Parkinson's disease. Several voice recordings of these people were processed. Each line in the data files corresponds to one recording. The first 22 columns are features derived from the recording, including minimum, average and maximum vocal fundamental frequency, several measures of variation in fundamental frequency, several measures of variation in amplitude, two measures of ratio of noise to tonal components in the voice status, two nonlinear dynamical complexity measures, a measure called signal fractal scaling exponent, as well as nonlinear measures of fundamental frequency variation [2, 3]. The last column is the target label indicating whether the subject is healthy (0) or suffers from Parkinson's disease (1).

References

[1] Joel Grus, *Data Science from Scratch*, 2015.

[2] M.A. Little, P.E. McSharry, E.J. Hunter, J. Spielman and L.O. Ramig, *Suitability of dysphonia measurements for telemonitoring of Parkinson's disease*, IEEE Transactions on Biomedical Engineering, Vol. 56, No. 4, pp. 1015–1022, 2009.

[3] A. Frank and A. Asuncion, *UCI Machine Learning Repository*, `http://archive.ics.uci.edu/ml`, University of California, Irvine, School of Information and Computer Sciences, 2010.