Introduction to Data Science 2022
Assignment 3

François Lauze, Thomas Hamelryck

Your solution to Assignment 3 must be uploaded to Absalon no later than **March 8th, 22.00 2022** . Guidelines for the assigment:

- **The assignments in IDS must be completed and written individually.** This means that your code and report must be written completely by yourself.
- Upload your report as a single PDF file (no Word) named `firstname_lastname.pdf`.
- Upload your Python code. Upload it in a zip archive, even if there is only one file. **Remember to add your dataset files in the zip file and do not upload a IPython Notebook!**

Total 130 points.

Covariance and Principal component analysis

In the first part of this assignment you will learn how principal component analysis (PCA) can be used for reducing dimensionality and visualizing global dataset structure. In the second part, which is theoretical, you will test your understanding of Bayesian Statistics through a few questions.

Exercise 1 (Performing PCA, 50 points)**.**

a) Implement PCA. You might want to use the supplied template function `pca.py`, especially for its comments. Your function should return i) unit vectors spanning the principal components, and ii) the variance captured by each of these components, where the principal components are sorted so that the variance is monotonically decreasing.

b) Perform PCA on the *murder* dataset as discussed in the appendix of this document. Remember that each principal component (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. Produce a scatter plot of the dataset along with the mean and the principal eigenvectors pointing out of the mean, each eigenvector with a length scaled by the standard deviation of the data projected onto that eigenvector.

c) Perform PCA on the *pesticide* dataset (see description in the Appendix below). Make a plot of variance versus PC index, where you should see the variance stabilizing (capturing primarily noise).

d) Perform PCA on the Occupancy dataset `occupancy_data.csv`. This is a modified version of the one used in the previous assignment: larger amount of data, and the time of day, in seconds, is included as the first feature. Perform first PCA on un-normalised data. Make a plot of variance versus PC index, where you should see the variance stabilizing (capturing primarily noise).

e) This time standardise the Occupancy dataset `occupancy_data.csv` first: each column (i.e., scalar feature) should be normalised to standard deviation 1. Column deviations can be computed as follows:

```
stddev = np.std(data, axis=0)
stddev.shape = (1, -1)
norm_data = data/stddev
```

Make a plot of variance versus PC index in that case.

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, b) the plot, and b) plot of variance versus PC; plot of cumulative variance versus PC; the numbers of dimensions needed to capture 90% and 95%.

Exercise 2 (Visualization in 2D, 40 points)**.** *Multidimensional scaling* is the process of visualizing a dataset in 2D or 3D while preserving pairwise distances between data points as well as possible. A classical way to do this is by projecting data points onto the first 2 or 3 principal components of the dataset.

a) Write a function to project the *Pesticide* dataset onto the first 2 principal components. You may want to use the supplied template `mds.py`, and produce a plot of the dataset. If you have not completed exercise 1, you may use a built-in PCA package from e.g. `scikit-learn`. The display the projected data.

b) Use/modify you function (or the `scikit-lean` code) to project the *Building occupancy* dataset of the first 2 principal components using the unnormalised data and the normalised one (two different plots).

c) **Virtual Chocolate Box question (i.e., really only for fun, no point).** *Only if you are familiar with `Matplotlib3D` or want to play with it.* Still using the *Building occupancy* data set, modify your function to project your data on its first 3 principal components. Display it. Redo it with the standardised data. With `Matplotlib3D`, you can rotate your plot so as to see data from different viewing angles. Try and comment.

Deliverables. Uploaded code and plot.

Clustering I

In this exercise, you are supposed to do clustering using the k-means algorithm as introduced in the lecture. You are encouraged to implement the algorithm on your own. However, you can also use scikit-learn:

```
from sklearn.cluster import KMeans kmeans = KMeans(..., algorithm='full',
...).fit(XTrain)
```

The option `algorithm='full'` ensures that scikit-learn applies the algorithm as defined in the lecture.

Set the number of cluster centers (centroids) to two (i.e., $k = 2$) and cluster the training data in `IDSWeedCropTrain.csv` (see description in the appendix). Of course, you should only cluster the input vectors (i.e., not the labels).

You are supposed to initialize the cluster centers in the k-means algorithm to the first two data points in `IDSWeedCropTrain.csv` . This is in general a **bad idea**. However, as the data in the file `IDSWeedCropTrain.csv` is in a random order, here this amounts to initializing the algorithm with two random training points. In the lecture, we discussed better initialization schemes. Still, you are supposed to use the first two data points for two reasons. First, this makes it easy for us to grade the assignments as we can compare to a reference solution. Second, this simplifies verifying your own implementation using scikit-learn as a reference. This is how initialization with the first two training data points can be done in scikit-learn:

```
startingPoint = np.vstack((XTrain[0,],XTrain[1,]))
kmeans = KMeans(..., n_init=1, init=startingPoint, ...).fit(XTrain)
```

Typically, you may want to start k-means with different initialization and take the best solution in terms of the optimization criterion. This is supported scikit-learn by setting `n_init` to the number of different initializations. In this assignment, you should perform just a single trial. You can use `print(kmeans.cluster_centers_)` to output the final cluster centers.

Exercise 3 (Clustering, 25 points)**.** Perform 2-means clustering of the input data in `IDSWeedCropTrain.csv`. For the submission, initialize the cluster centers with the first two data points in `IDSWeedCropTrain.csv` (that is not a recommended initialization technique, but makes it easier to correct the exam).

Deliverables. Description of software used; two cluster centers

On Bayesian Statistics

Exercise 4 (Bayesian Statistics, 15 points)**.** A few questions related to the Thursday morning lecture.

- How is probability interpreted differently in the frequentist and Bayesian views?
- Cheap, efficient computers played a major role in making Bayesian methods mainstream. Why?
- What is the difference between a Bayesian credible interval and a frequentist confidence interval?
- How does a maximum likelihood estimate approximate full Bayesian inference?
- When will point estimates be a good approximation of full Bayesian inference?

Deliverables. One to two lines of answer per question.

Appendix: The data material

This appendix contains information about the used datasets and how to read them.

The murder data

For visualization purposes we will work with the following classical, small-scale dataset. The file `murderdata2d.txt` contains 2 features from the murder dataset of Spaeth [1991], Kleinbaum and Kupper [1978], corresponding to ”percent unemployed” and ”murders per annum per 1,000,000 inhabitants. The full dataset can be obtained from `http://people.sc.fsu.edu/~jburkardt/datasets/regression`. The file `murderdata2d.txt'` can be read as follows:

```
data = np.loadtxt('murderdata2d.txt')
```

Building occupancy data

This dataset comes from the same source as the occupancy data from the previous assignment (see Candanedo and Feldheim [2016]). An other subsample of the full dataset was extracted, with the time of day field kept, but converted from ”hour:minute:second” to seconds. The different fields are described below

- time of the day, in second,

- temperature, in Celcius,
- relative humidity, %,
- Light in lux,
- CO2 in ppm,
- humidity ratio, derived quantity from temperature and relative humidity, in kg-water-vapor/kg-air,
- occupancy, 0=not occupied, 1=occupied – the class label.

You can load `occupancy_data.csv` as follows:

```
data = np.loadtxt('occupancy_data.csv', delimiter=',')
```

Remove the last column containing the labels both for PCA and clustering studies.

The pesticide data

The data for the following tasks are taken from a research project financed by Miljøstyrelsen and involving researchers from DIKU and PLEN/KU. Selected results from the project are described by Rasmussen et al. [2016] and Olsen et al. [2017]. While the problem setting is inspired by Olsen et al. [2017], the data were processed differently.

Introduction to the problem (not relevant for solving the exercises). Pesticide regulations and a relatively new EU directive on integrated pest management create strong incentives to limit herbicide applications. In Denmark, several pesticide action plans have been launched since the late 1980s with the aim to reduce herbicide use. One way to reduce the herbicide use is to apply site-specific weed management, which is an option when weeds are located in patches, rather than spread uniformly over the field. Site-specific weed management can effectively reduce herbicide use, since herbicides are only applied to parts of the field. This requires reliable remote sensing and sprayers with individually controllable boom sections or a series of controllable nozzles that enable spatially variable applications of herbicides. Preliminary analysis [Rasmussen et al., 2016] indicates that the amount of herbicide use for pre-harvest thistle (Cirsium arvense) control with glyphosate can be reduced by at least 60 % and that a reduction of 80 % is within reach. See Figure 1 for an example classification. The problem is to generate reliable and cost-effective maps of the weed patches. One approach is to use user-friendly drones equipped with RGB cameras as the basis for image analysis and mapping.

The use of drones as acquisition platform has the advantage of being cheap, hence allowing the farmers to invest in the technology. Also, images of sufficiently high resolution may be obtained from an altitude allowing a complete coverage of a normal sized Danish field in one flight.

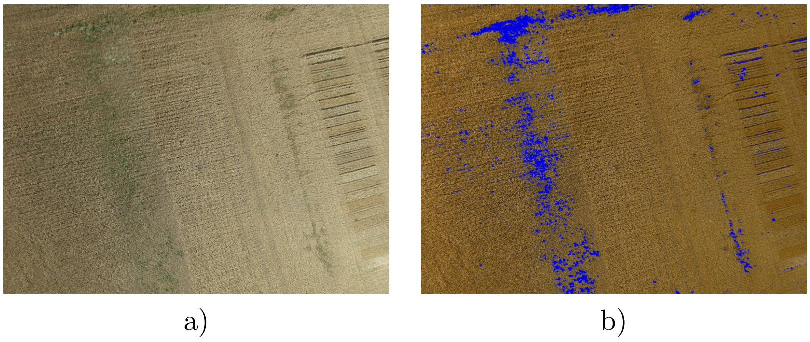

Figure 1: Example from another approach. a) An original image. b) Initial pixel based detection.

Your data is taken from a number of images of wheat fields taken by a drone carrying a 3K by 4K Canon Powershot camera. The flying height was 30 meters. A number of image patches, all showing a

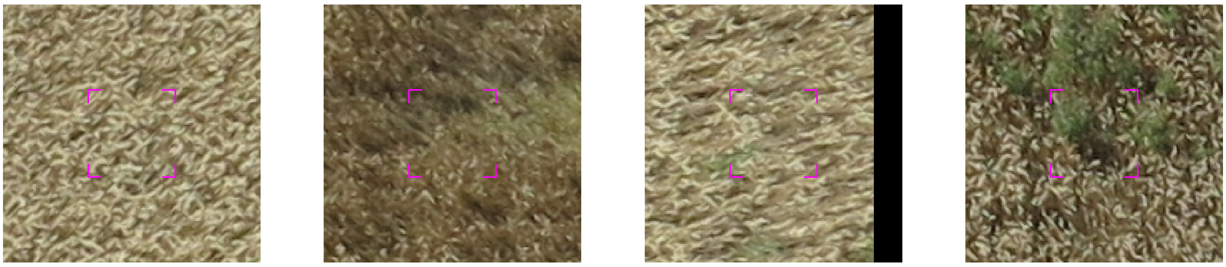

Figure 2: The two images on the left are classified as crop. The two images on the right are classified as weed. The classification of the middle two patches is debatable. The central area used for performance evaluation is indicated by the small magenta markers.

field area of 3×3 meters were extracted. Approximately half of the patches showed crop, the remaining thistles. For each patch only the central 1 × 1 meter sub-patch is used for performance measurement. The full patch was presented to an expert from agriculture and classified as showing either weed (class 0) or only crop (class 1).

In Figure 2 two patches classified as crop and two patches classified as weed are shown. Two of the patches are easy to classify (expert or not), while the remaining two less clearly belong to either of the classes.

For each of the central sub-patches (here of size 100 × 100

pixels), 13 rotation and translation invariant features were extracted. In more detail, the RGB-values were transformed to HSV and the hue values were extracted. The 13 features were obtained by taking a 13-bin histogram of the relevant color interval.

Reading in the data. The training and test data are in the files `IDSWeedCropTrain.csv` and `IDSWeedCropTest.csv`, respectively, available on Absalon. Each line contains the features and the label for one patch. The last column corresponds to the class label.

This is one way to read in the data in Python:

```
import numpy as np
# read in the data
dataTrain = np.loadtxt('IDSWeedCropTrain.csv', delimiter=',')
dataTest = np.loadtxt('IDSWeedCropTest.csv', delimiter=',')
# split input variables and labels
XTrain = dataTrain[:, :-1]
YTrain = dataTrain[:, -1]
XTest = dataTest[:, :-1]
YTest = dataTest[:, -1]
```

References

L. M. Candanedo and V. Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. *Energy and Buidlings*, pages 28–39, 2016.

D. Kleinbaum and L. Kupper. *Applied Regression Analysis and Other Multivariable Methods.* Duxbury Press, 1978.

S. I. Olsen, J. Nielsen, and R. J. Thistle detection. In *Scandinavian Conference on Image Analysis*, 2017. Submitted.

J. Rasmussen, J. Nielsen, S. I. Olsen, K. Steenstrup Petersen, J. E. Jensen, and J. Streibig. Droner til monitorering af flerårigt ukrudt i korn. Bekæmpelsesmiddelforskning 165, Miljøstyrelsen, Miljøministeriet, 2016.

H. Spaeth. *Mathematical Algorithms for Linear Regression.* Academic Press, 1991.