Introduction to Data Science 2023
Assignment 4

Daniel Hershcovich, Thomas Hamelryck, Stella Frank, François Lauze, Andreas Manoukian

Your solution to Assignment 4 must be uploaded to Absalon no later than Monday, March 20th, 2023, 23:59.

Guidelines for the assignment:

- **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 .docx file) named `firstname_lastname.pdf`. The report should include all the points of the deliverables stated at the end of each exercise. Adding code snippets to the report is optional, but nice to have especially when you implement things from scratch.
- Upload your `Python` code. Upload it in a zip archive, even if there is only one file. Expected is either one or several "`.py`" files or a "`.ipynb`" notebook.
- We will grade using Python 3.10 and the specific package versions specified in requirements.txt. Other versions may work, or may break; using these versions is the safest. You can create an appropriate environment with all the requirements using Anaconda or reuse the same environment you had in the previous assignment and install the requirements file.
- To reuse a previous environment, 1) open a terminal or command prompt and navigate to the directory containing the requirements.txt using `cd` (e.g. `cd Desktop/courses/ids/as4`). 2) activate your environment (make sure to replace `name_of_env` with the name you specified in the previous assignment).

  ```
  conda activate name_of_previous_env
  pip install -r requirements.txt
  ```

  Otherwise, if you want to create a new environment you first need to run:

  ```
  conda create --name my_env python=3.10
  ```

  Then use the above two command lines to install the corresponding requirements

Covariance and Principal component analysis

In this assignment, you will learn how principal component analysis (PCA) can be used for reducing dimensionality and visualizing global dataset structure.

Exercise 1 (Performing PCA)**.**

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 *occupancy* dataset `OccupancyTrain.csv`. This is a modified version of the one used in the previous assignment (see description in Appendix). Make a plot of variance versus PC index, where you should see the variance stabilizing at a low level (capturing primarily noise) for larger PC indices (your y axis should be the variances found from PCA).

c) Perform PCA on the *pesticide* dataset `IDSWeedCropTrain.csv` (see description in the Appendix below). Again, make a plot of variance versus PC index.

Next, plot the cumulative variance versus the index of the PCs, sorted by the amount of variance they capture. You can determine the cumulative variance by dividing the eigenvalues by their sum and then calculating their cumulative sum. This way you capture how large a proportion of the variance is described by the first, second, etc Principal Component (PC).

d) How many PCs (dimensions) do you need to capture 90% of the variance in your dataset? How many do you need to capture 95%?

Deliverables. a) Description of your implementation b) the plot of variance versus PC for *occupancy*; c) 2 plots, one plot of variance versus PC for *pesticide* and one plot of cumulative variance versus PC for *pesticide*, d) the numbers of dimensions needed to capture 90% and 95% for *pesticide*.

Exercise 2 (Visualization in 2D)**.** 2D-projection is the process of visualizing a dataset in 2D 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 principal components of the dataset.

a) Write a function to project the *Pesticide* training dataset onto the first 2 principal components. Produce a plot of the training dataset. If you have not completed exercise 1, you may use a built-in PCA package from e.g. `scikit-learn` to display the projected data.

b) Use/modify your function (or the `scikit-lean` code) to project the *Occupancy* training dataset on the first 2 principal components. Plot the projected data as PC1 versus PC2.

Deliverables. a) Description of your implementation and b) the plot.

Critical thinking

Exercise 3.
In the occupancy dataset there are features such as *humidity* which is on a 100 scale, and there are features such as *time* which is on a 10.000 scale.

Assume that you perform 2 preprocessing steps *prior* to performing PCA with the covariance matrix.

i) Centering

ii) Normalization

Will applying these preprocessing steps make a difference or not? You do not need to implement this. Discuss the effect of centering and normalization and if it is meaningful before doing PCA with the covariance matrix.

Deliverables. Two short discussions (one for centering and one for normalization).

Clustering II

This exercise continues Exercise 3 from the previous Assignment 3.

In Exercise 3 from Assignment 3, you were asked to cluster the data in using k-means clustering with $k = 2$ using the first two data points as starting points and the data visualized using two pairs of features. Now, you are going to perform PCA on `IDSWeedCropTrain.csv` in order to visualize the data by projecting it on the first two 2 principal components.

Exercise 4 (Clustering II)**.** Visualize the data in `IDSWeedCropTrain.csv` by projecting it onto its first two principal components (as in Exercise 2). Colour the data points according to their class. Take the cluster centers you found in Exercise 3 from Assignment 3 (2-means clustering of the input data in `IDSWeedCropTrain.csv`, the cluster centres initialized with the first two data points). Then project the centers onto the first two principal components found in the previous step and visualize them together with the data points (i.e., in the same plot). Briefly discuss whether you got meaningful clusters.

Deliverables. a) Projection of the two cluster centers (i.e., two two-dimensional vectors); b) a 2D plot visualizing the data and the cluster centers; c) short discussion of results

Exercise 5 (Clustering III)**.** Perform 2-means clustering on the `OccupancyTrain.csv` and report the center values as you did with the crop dataset. Initialize your 2-means the same way (first two datapoints). Visualize the data in `OccupancyTrain.csv` by projecting it onto its first two principal components. Colour the data points according to their class. Take the centers you found above and project them onto the first two principal components found in the previous step and visualize them together with the data points (i.e., in the same plot). Briefly discuss whether you got meaningful cluster centers.

Deliverables. a) projection of the two cluster centers (i.e., two two-dimensional vectors); b) a 2D plot visualizing the data and the cluster centers; c) short discussion of results

Linear regression

In this part of the assignment, you will use and evaluate linear regression for predicting the occupancy from its properties.

Exercise 6 (Linear Regression)**.** Your task is to use linear regression to predict the occupancy of buildings (0=not occupied, 1=occupied) based on its properties (humidity, temperature, light etc). You are going to learn a linear model

$$t = f(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_D x_D = \mathbf{w}^T \mathbf{x},$$

where t is the predicted output variable (occupancy), $\mathbf{x} = (1, x_1, x_2, \ldots, x_D)^T$ is the vector-valued input variable (properties), and $\mathbf{w} = (w_0, w_1, \ldots, w_D)$ are the free parameters. The parameters w_i *define* the regression model, and once they have been estimated, the model can be used to predict outputs for new input values $\mathbf{x}_0$.

a) Implement linear regression. Call your function `multivarlinreg()`; a template is provided in the `multivarlinreg.py` file. Your code should load the data matrix X containing the input variables, as well as the output vector t, and output an estimate of the free parameters in the model, that is, the w_i in the form of the vector $\mathbf{w}$. Remember the offset parameter w_0. You should not use a built-in function for regression.
b) Run your regression function on the training set by using only the last column (the humidity ratio) and report your estimated parameters w_0 (the offset parameter) and w_1. What do you see?
c) Now run your regression function on the whole training set and report your estimated parameters w_i. What can you say about the different properties shown in the data?

Deliverables. a) Discussion of your implementation, b) Parameters w_0 and w_1 and a short description including your observations, c) parameters w_i and a small description including the different properties shown in the data

Exercise 7 (Evaluating Linear Regression)**.**

a) Implement the root means square error

$$RMSE(\mathbf{w}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|y_i - f(\mathbf{x}_i, \mathbf{w})\|^2}$$

as function `rmse()` for the linear model. A template is provided in the `rmse.py` file. For a set of known input-output values $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$, where y_i is the recorded output value associated to the i^{th} data point $\mathbf{x}_i$ in the dataset, the parameter $\mathbf{w}$ allows to compute $f(\mathbf{x}_i, \mathbf{w})$, the output value associated to the input $\mathbf{x}_i$ as predicted by your regression model. Your code should take as input the predicted values $f(\mathbf{x}_i, \mathbf{w})$ and ground truth output values y_i. You should not use a built-in function for RMSE.

b) Build the regression model for the input variables using the last column of the training set as in 6b). Use the last column of the test set to compute the RMSE of this model. How good is the model?

c) Build the regression model for the input variables using the whole training set as in 6c). Use the whole test set to compute the RMSE of this model. How good is the model now?

In b) and c) it is OK to use a built-in function for regression if you have not completed exercise 6.

Deliverables. a) the RMSE for the humidity and a short description, b) the RMSE for all the columns and a short description.

Appendix: Data material

Building occupancy data

This dataset comes from the same source as the occupancy data from the previous assignment. The dataset is larger and includes as a first feature the time of day, but converted from 'hour:minute:second' to seconds after midnight. The dataset fields are described below

- time of the day, in seconds,
- temperature, in Celcius,
- 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 the Occupancy training and test datasets as follows:

```
occu_train = np.loadtxt('OccupancyTrain.csv', delimiter=',')
occu_test = np.loadtxt('OccupancyTest.csv', delimiter=',')
```

Important: Separate the last column containing the labels for PCA, clustering, and regression studies. PCA only uses the features to identify the principal components, not the label data (y). The labels (y_occu_train, and y_occu_test) should only be used in the plots and regression parts, but not in PCA.

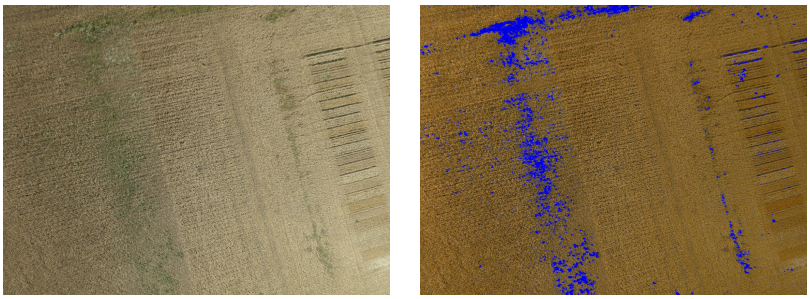
```
X_occu_train = occu_train[:,:-1]
y_occu_train = occu_train[:,-1]
X_occu_test = occu_test[:,:-1]
y_occu_test = occu_test[:,-1]
```

The pesticide data

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 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 an 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 complete coverage of a normal sized Danish field in one flight.



a) b)

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 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).

Figure 2 displays four patches, with two classified as crop and two classified as weed. While two of the patches are easily distinguishable as either crop or weed by an expert, the other two patches are less clearly assignable to either class.

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. You can load the Weed training and test datasets as follows:

```
weed_train = np.loadtxt('IDSWeedCropTrain.csv', delimiter=',')
weed_test = np.loadtxt('IDSWeedCropTest.csv', delimiter=',')
```

Important: Separate the last column containing the labels for PCA and clustering. PCA only uses the features to identify the principal components, not the label data (y). The labels (y_weed_train, and y_weed_test) should only be used in the plots.

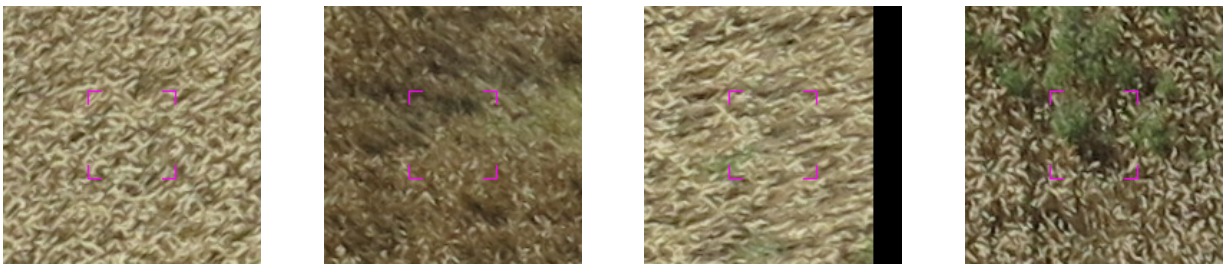


Figure 2: The two images on the left are classified as crop. The two images on the right are classified as weed. The classification of the middle two patches is debatable. The central area used for performance evaluation is indicated by the small magenta markers.

```
X_weed_train = weed_train[:,:-1]
y_weed_train = weed_train[:,-1]
X_weed_test = weed_test[:,:-1]
y_weed_test = weed_test[:,-1]
```

References

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.