Introduction to Data Science 2024
Assignment 4

Your solution to Assignment 4 must be uploaded to Absalon no later than Monday March 18th, 2024, 15:00. Responsible TA: Wenhao Gao(wega@di.ku.dk).

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.
- For rules of plagiarism and all uses of AI assistance, we have the same requirements as in previous assignments.

Note: 1. Please use standarization (also called z-score scaling) to preprocess the data before doing all exercises in this assignment; 2. The 6 exercises are equally weighted, each of them is worth 10 points. You can choose any 5 of them to solve. If you solve all 6, the lowest score will be dropped.

Principal component analysis

In this part, 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. 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. 1) Brief description of your implementation; 2) The plot of variance versus PC for *occupancy*; 3) 2 plots, one plot of variance versus PC for *pesticide* and one plot of cumulative variance versus PC for *pesticide*; 4) 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 *Performing PCA*, 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. 1) Brief description of your implementation; 2) The plot.

Exercise 3 (Critical thinking)**.** In the occupancy dataset there are features such as *humidity* which are 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 perform PCA with the covariance matrix.

i) Centering

ii) Normalization

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

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

Gradient descent

Exercise 4 (Gradient descent & learning rates)**.** Apply gradient descent to find the minimum of the function $f(x) = e^{-x/2} + 10x^2$.

Detailed instructions:

1. Manually derive the function $f(x)$ and implement a function that computes this derivative $f'(x)$.
2. Implement and apply gradient descent with learning rates $\eta \in \{0.1, 0.01, 0.001, 0.0001\}$. Your implementation should be done without any external Python packages (you may use NumPy, but it is not required).
3. For each of the learning rates, do the following:
   (a) Take $x = 1$ as a starting point.
   (b) Run the algorithm until the absolute value of the gradient falls below 10^{-10} or the algorithm has exceeded 10,000 iterations. Report the number of iterations it took the algorithm to converge and the function value at the final iteration (i.e., report 8 values in total, 2 values for each of the 4 learning rates). Discuss briefly which of the learning rates is preferable and why.
   (c) Visualize the tangent lines and gradient descent steps for the first three iterations (produce 4 plots for your report corresponding to gradient descent with the four learning rates). The first tangent line should be at the initial point $x = 1$.

(d) Visualize gradient descent steps for the first 10 iterations (another 4 plots; no visualization of tangent lines in this case).

Deliverables. 1) Number of iterations until convergence and function values at final iteration; 2) 4 plots (1 for each learning rate) visualizing the tangent lines and gradient descent steps when performing 3 iterations; 3) 4 plots (1 for each learning rate) visualizing gradient descent steps for 10 iterations; 4) Discussion about preferred learning rate.

Linear Regression

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

Exercise 5 (Linear Regression)**.** Your task is to use linear regression to predict the **temperature** of buildings based on other three properties (**relative humidity, light, CO2**; **Don't use humidity ratio**). 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 (temperature), $\mathbf{x} = (1, x_1, x_2, \ldots, x_D)^T$ 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. 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 can use a built-in function for regression (sklearn of other packages).

b) Run your regression function on the training set by using only the second column (**relative humidity**) and report your estimated parameters w_0 (the offset parameter) and w_1.

c) Now run your regression function on the training set by using the three properties (**relative humidity, light, CO2**) and report your estimated parameters w_i. What can you say about the relationship between the three properties and temperature?

Deliverables. 1) Discussion of your implementation; 2) Parameters w_0 and w_1; 3) Parameters w_i and a short description.

Exercise 6 (Evaluate 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. 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 can use a built-in function for RMSE.

b) Build the regression model for the input variables using **relative humidity** (second column) of the training set as in 5b). Use **relative humidity** (second 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 three properties (**relative humidity, light, CO2**) of training set as in 5c). Use the same properties in 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 *Linear Regression.*

Deliverables. 1) The RMSE for only using relative humidity; 2) The RMSE for using three properties; 3) A short discussion of the results.

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

- 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 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.

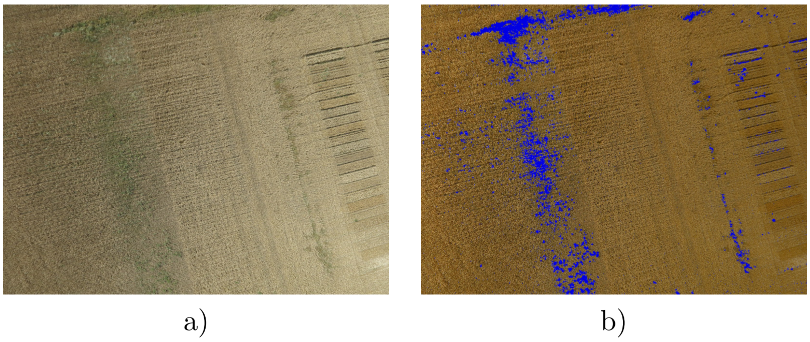

Figure 1: Example from another approach. a) An original image. b) Initial pixel based detection.

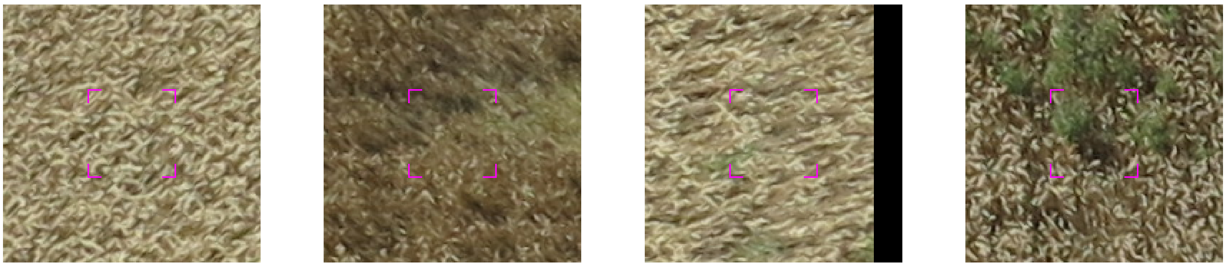

Figure 2: The two images on the left are classified as crop. The two images on the right are classified as weed. The classification of the middle two patches is debatable. The central area used for performance evaluation is indicated by the small magenta markers.

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.

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.

```
X_weed_train = occu_train[:,:-1]
y_weed_train = occu_train[:,-1]
X_weed_test = occu_test[:,:-1]
```

```
y_weed_test = occu_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.