Introduction to Data Science 2023
Assignment 3

Thomas Hamelryck, Stella Frank, Daniel Hershcovich, François Lauze, Wenyan Li

Your solution to Assignment 3 must be uploaded to Absalon no later than Monday March 7th., 2023, 23:59.

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 .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/as3`). 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 corresponding requirments

Bayesian exercises

Exercise 1 (Theory)**.** Consider the following derivation of the ELBO, a quantity used in variational Bayes inference. For each of the 4 lines in the derivation, explain its justification (hint: remember the "three power tools of statistics").

$$
\begin{aligned}
&\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x})\right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z}\mid\mathbf{x})}\right]\right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}\mid\mathbf{x})}\frac{q_{\boldsymbol{\phi}}(\mathbf{z}\mid\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}\mid\mathbf{x})}\right]\right] \\
&= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}\mid\mathbf{x})}\right]\right]}_{=\mathcal{L}_{\boldsymbol{\theta},\phi}(\mathbf{x})} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_{\boldsymbol{\phi}}(\mathbf{z}\mid\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}\mid\mathbf{x})}\right]\right]}_{=D_{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\right)}
\end{aligned}
\tag{1}
$$

ELBO is $\mathcal{L}_{\boldsymbol{\theta},\phi}(\mathbf{x})$ in Equation 1.
Deliverables. Please format your answers as bullet points in your report, as following:

Answer:

1.
2.
3.
4.

Exercise 2 (Practical)**.** Consider the following pyMC3 model for Bayesian linear regression. There are 4 lines with errors. Identify them, correct the error, and explain why it was an error (i.e., not just what was wrong, but WHY it was wrong).

```
# Define the model
with pm.Model() as model:
    # Register the data - useful for later predictions,
    # when we replace x to predict y
    x = pm.Data('x', x_data)
    y = pm.Data('y_obs', y_data)

    # Define priors
    # Slope
    a = pm.Normal("slope", mu=100, sigma=100)
    # Intercapt
    b = pm.Normal("intercept", mu=100, sigma=100)
    # Standard deviation - Note HalfNormal!
    s = pm.Normal("sigma", sigma=0.001)

    # Define the likelihood (note the "observed" argument, and mu=ax+b)
    likelihood = pm.Normal("y", mu=a*x + b, sigma=s)

    # Now we define the inference engine
    # We will sample from the posterior using MCMC (Hamiltonian MC, NUTS)
    step = pm.NUTS()

    # The trace variable contains the samples a,b,s ~ P(a,b,s|D)
    trace = pm.sample(1000, tune=1000, init=None, step=step, cores=2)
```

Deliverables. Please format your answers as bullet points in your report, as following:

Answer:

1.
2.
3.
4.

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. Either implementations is graded equally.

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:

```
import numpy as np
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. 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).

[Visualization] Assuming we can get a rough idea about our data clusters by taking a look at 2 of the 13 features. Make two scatter plots of your **test** split clusters by using:

- Feature 6 and Feature 7 (e.g. you can get feature 6 by using the 6th column `XTest[:, 6]`)
- Feature 1 and Feature 2

Compare the difference in the scatter plots and summarize your observations on using the different selected features. (Hints: you can use `sns.scatterplot()` for visualization)

Deliverables. 1) Description of software used; 2) report values of two cluster centers; 3) display two plots side by side and your observation

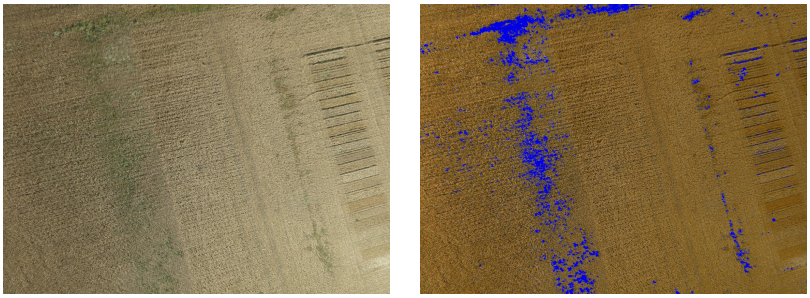
Reading in the data. The training and test data are in the files `IDSWeedCropTrain.csv` and `IDSWeedCropTest.csv`, respectively, available on Absalon. Each line contains the features and the label for one patch. The last column corresponds to the class label.

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]
```

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.



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).

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

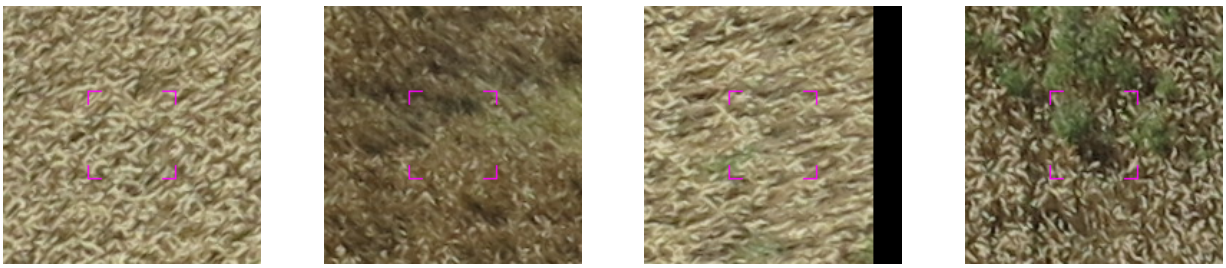


Figure 2: The two images on the left are classified as crop. The two images on the right are classified as weed. The classification of the middle two patches is debatable. The central area used for performance evaluation is indicated by the small magenta markers.

interval.

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.