

Introduction to Data Science 2024

Assignment 3

In Assignment 3 you will work with Bayesian Statistics and K-means clustering.

Available from Tuesday, February 27th of 2024. Your report should be uploaded to Absalon no later than Tuesday 5th, at 15:00 latest. Total points: 50. Responsible TA: Francisca Alves(frsa@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.
- We will grade using Python 3.12.1 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=3.12.1
```

Then use the above two command lines to install corresponding requirements

Academic Code of Conduct

You are welcome to discuss the assignment with other students, but sharing of code is not permitted. Copying code or text from the report directly from other students will be treated as plagiarism. Please refer to the University's plagiarism regulations if in doubt. For questions regarding the assignment, please ask on the Absalon discussion forum.

In short, plagiarism means copying text or ideas from others without acknowledging the underlying sources. Crucially, this does not mean that you are prohibited from building on others' ideas or use external sources, but rather that you have to properly acknowledge all sources used in your work. This holds for instance for building on code from lectures or lab sessions. If in doubt, we recommend erring on the side of over- rather than under-acknowledging sources.

While guidance on AI assistance usage is not covered in this course, you are welcome to take a look at the Absalon course on Learning resources for Digital Literacy.

You are also welcome to use AI assistance (e.g., ChatGPT, GitHub Copilot) for tutoring purposes, augmenting the TAs for help with questions and issues. However, keep in mind that their output is not guaranteed to be either comprehensive, true or aligned with the course scope and expectations. Always check with the TAs in case of doubt. Importantly, the use of AI assistance while writing the assignment is allowed only for the following purposes:¹

- As coding tools (e.g., GitHub Copilot): no restrictions.
- As writing tools to improve the writing of original content, i.e. when the prompt you write contain all the ideas to be formulated: no restrictions.
- As search tools to identify related literature: no restrictions. Usual citation requirements apply (see plagiarism note above): you must cite the original work you identify, even if you used an LLM to find it. Just like you do not cite Google Search for papers you find using it, you should not cite ChatGPT for this either. In particular, always make sure that the citations it provides actually exist—LLMs are known to often generate plausible but nonexistent references.
- As generation tools for *new* ideas: generated content must be clearly highlighted even if post-edited by yourself. All prompts/transcripts from the tools used must be included as an appendix at the end of the submission in PDF, after the references.

For all uses of AI assistance, the purpose, tool, and version² must be stated in the submission—e.g, in a dedicated section. Here is such a statement for example:

ChatGPT August 3 Version was used as a writing assistance tool and as a serch tool to identify related literature. GitHub Copilot July 14 Version was used while developing the code for...

Bayesian exercises

Exercise 1 (Bayesian Statistics / 10 points).

Provide a **small** and **concise** example of how bayesian statistics can be applied in your study degree. You should provide references to your answer if you don't come up with an example on your own.

Deliverables. Provide the scenario, prior, likelihood function and posterior.

Exercise 2 (Theory / 8 points).

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”).

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

ELBO is $\mathcal{L}_{\theta, \phi}(\mathbf{x})$ in Equation 1.

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

¹Note that evaluating LLMs as models on task data is not considered “AI assistance” and is not restricted or affected by the rules here.

²If multiple version have been used throughout the assignment, list all of them. In the ChatGPT web interface, the version is specified at the bottom of the screen, under the text input field.

Answer:

- 1.
- 2.
- 3.
- 4.

Exercise 3 (Practical / 8 points).

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)
    # Intercept
    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 Exercises

Exercise 4 - Introduction

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='lloyd', ...).fit(XTrain)
```

The option `algorithm='lloyd'` 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 4. (12 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 assignment).

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.

Notes: 1) Plots should be complete, i.e., have proper title and labels. Failure to comply will result in point deduction.

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]
```

Exercise 5 - Introduction

Images are often compressed in order to facilitate portability and storage. In this exercise you'll work with image compression using the k -means clustering algorithm. In practice, what happens is similar pixels are clustered in the same group. But how do we know how many clusters to choose when initializing the algorithm? What is the optimal k value? These questions can be addressed with evaluation methods, of which the "elbow method" is a popular one.

Elbow method gives us an idea on what a good k number of clusters will be based on the sum of squared distance between data points and their assigned centroid (also known as within-cluster sum of squares, WCSS). Naturally, the smaller the WCSS value, the closer each data point will be on average to its assigned centroid, indicating the cluster division is well made. However, there comes a point at which increasing k no longer leads to a significant decrease in WCSS, and the rate of decrease slows down. This point is known as the "elbow point" and it represents a trade-off between minimizing the WCSS and avoiding overfitting. The elbow point thus represents the ideal k value.

On scikit-learn, WCSS corresponds to the attribute `k-means.inertia_`.

Similarly to the previous exercise you'll have to perform k -means clustering and [are encouraged to implement your own algorithm](#). However, you're welcome to use the one provided by scikit-learn (as both implementations are graded equally). Once again, to make your results reproducible and easier to grade, the initialization should go as follows:

```
from sklearn.cluster import KMeans
import numpy as np
startingPoint_list = np.unique(X, axis = 0)
kmeans = KMeans(..., algorithm='lloyd', n_init=1, init=startingPoint[:i], ...).fit(X)
```

Here `i` in `startingPoint[:i]` corresponds to the number of centroids in each initialization.

Exercise 5 (k -means, Image Segmentation / 12 points).

a) Open and read image into `np.array`. Use the elbow method to choose the number of k clusters (k varies between 1 and 10).

b) Now, with the best observed k value implement k -means clustering on the image and display original and compressed image.

Deliverables. 1) Display "elbow method" plot; 2) from direct observation of the plot report the optimal value for k and its corresponding inertia value; 3) provide one reason as to why the elbow method might not be the best evaluation method (Hint: What could be/are the struggles when visualizing

the plot?); 4) Display the original image and compressed image side by side. 5) Report the meaning of the k value in the context of this exercise.

Notes: 1) Depending on your code implementation (e.g. you use a for loop to calculate the inertia for each different number of k cluster), it might take a few minutes for the code to run. So don't be alarmed. 2) Plots should be complete, i.e., have proper title and labels. Failure to comply will result in point deduction.

Reading and compressing the data. The image for this exercise is the `dog.jpg` file, available on Absalon. Before applying k-means clustering, image data (pixels) must be prepared. Images are read from the storage media, then the data is read within (pixels with three bytes RGB color). These pixels are arranged in the form of a matrix with a number of columns and rows with a color for each point.

This is one way to read and work with the data in Python:

```
import numpy as np
from matplotlib.image import imread
# read in the data
img = imread('dog.jpg')
img_size = img.shape
# Reshape it to be 2-dimension
X = img.reshape(img_size[0] * img_size[1], img_size[2])
...
...
# your code
...
...
# Assert that compressed image is within [0,255] pixel value range.
X_compress = np.clip(cluster_centers_for_data_points.astype('uint8'), 0, 255)
# Reshape X_compress to have the same dimension as the original image 3265 * 4898 * 3
X_compress = X_compress.reshape(img_size[0], img_size[1], img_size[2])
```

A Appendix for exercise 4 (not relevant for solving the exercise).

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.

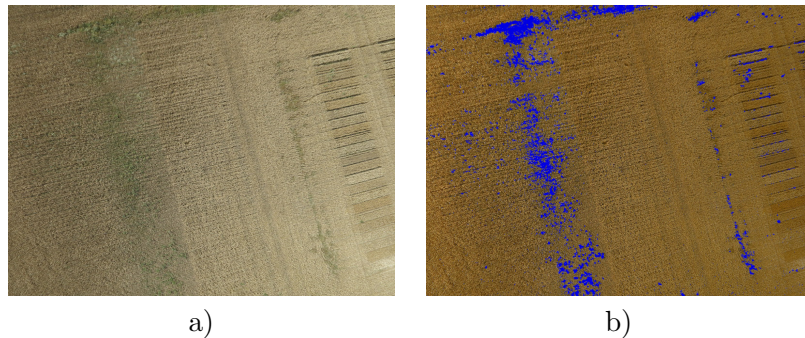


Figure 1: Example from another approach. a) An original image. b) Initial pixel based detection.

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 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.

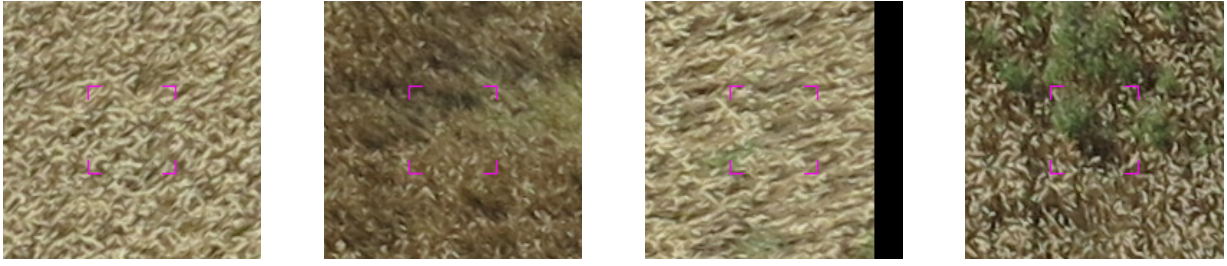


Figure 2: The two images on the left are classified as crop. The two images on the right are classified as weed. The classification of the middle two patches is debatable. The central area used for performance evaluation is indicated by the small magenta markers.

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.

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.