Problem Set 4

Advanced data analysis and machine learning

> **Important**
>
> You must upload your problem set to Moodle before 23:55 on 2023-05-25.
>
> Your submission must include:
>
> - a PDF that responds to all questions; and
> - a single Python script (`.py`) that you used to answer all questions.
>
> Answers without accompanying Python code will recieve zero marks.

Use the following Python code to generate data for Questions 1-4 inclusive.

```
import numpy as np
from sklearn.datasets import make_blobs

def generate_data(student_id, N=500, K=5):
    np.random.seed(student_id)
    n_samples = np.random.multinomial(N, np.ones(K)/K)
    X, assignment = make_blobs(
        n_samples=n_samples,
        cluster_std=np.exp(np.random.uniform(-1, 0, size=K)),
        random_state=student_id
    )
    return X
```

Question 1 [30 marks]

Write a function called `expectation(...)` which returns a two-length tuple containing (1) the log likelihood of the data given some model parameters, and (2) a (N, K) array of weight values.

Question 2 [30 marks]

Write a function called `maximization(...)` which returns an new estimate of the model parameters, given the data and the weighting matrix from Question 1.

Question 3 [10 marks]

Select an initial guess for θ that is 'far' away from the true values.

Write a function called `expectation_maximization(...)` that accepts the arguments `y`, `y_err`, and your initial guess of the parameters. This function should alternate between the `expectation()` and `maximization()` steps until the increase in the log likelihood is less than 10^{-5}.

Make a figure with two panels: (1) showing the log likelihood as a function of E-M step, and (2) showing the difference in log-likelihood as a function of E-M step.

Question 4 [10 marks]

Make a figure with two panels that both show the data. On the first panel you should show the initial probability density of each of the K mixtures in different colours. On the second panel you should show the final probability density of each of the K mixtures in the same colours used in the first panel.

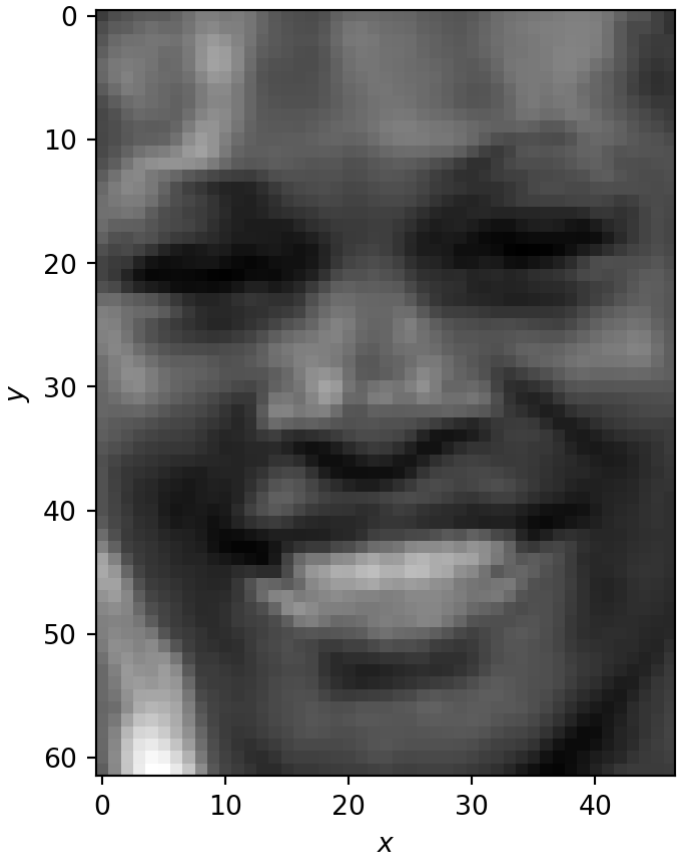
Question 5 [5 points]

You are tasked with performing dimensionality reduction on a set of images of human faces. In this question you will use an existing package to do the PCA for you. Remember that if you were to do this yourself, your data must be mean-centered and unit variance in every dimension before you do PCA!

```
from sklearn.datasets import fetch_lfw_people
faces = fetch_lfw_people(min_faces_per_person=50)

# Who are these people?!
print(faces.target_names)
['Ariel Sharon' 'Colin Powell' 'Donald Rumsfeld' 'George W Bush'
  'Gerhard Schroeder' 'Hugo Chavez' 'Jacques Chirac' 'Jean Chretien'
  'John Ashcroft' 'Junichiro Koizumi' 'Serena Williams' 'Tony Blair']

# What do their faces look like?
print(faces.images.shape)
(1560, 62, 47)

# The target name index for each image (0 = Ariel Sharon, etc)
print(faces.target.shape)
1560

print(faces.target)
[11,  4,  2, ...,  3, 11,  5]
```

You can see that we have 1,560 images, where each image is 62 pixels by 47 pixels. Let's plot an image.

```
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_lfw_people
faces = fetch_lfw_people(min_faces_per_person=50)

fig, ax = plt.subplots(figsize=(4, 4.75))
ax.imshow(faces.images[12], cmap="binary_r")
ax.set_xlabel(r"$x$")
ax.set_ylabel(r"$y$")
fig.tight_layout()
```



Use PCA to fit these images (`sklearn.decomposition.PCA`) and find the first 150 principal components. You will find that the dimensionality of this data is large, so vanilla PCA will have a hard time. Use the 'random' algorithm as the SVD solver, which takes random subsets of the data to find the first components. Make a figure showing the first 50 components (eigenfaces) as images, with one image per panel. This code example might help you set up your figure:

```
fig, axes = plt.subplots(
    5, 10,
    figsize=(10, 5),
    subplot_kw={'xticks':[], 'yticks':[]},
    gridspec_kw=dict(hspace=0.1, wspace=0.1)
)

for i, ax in enumerate(axes.flat):
    ...
```

Question 6 [5 points]

Plot the cumulative explained variance as a function of the number of principal components.

Question 7 [10 points]

Use PCA to compute the contributions from the first 150 principal components for every data point. You should have an array that is 1,560 by 150 (the number of images by the number of components). Now use these components to take an inverse transform with PCA, to show projected images using

only those 150 components. For each person in the data set, plot an original image of them compared to the reconstructed image using 150 principal components.