1 Practical: Dimensionality Reduction

In Brightspace, you will find the file `COIL20.mat`, which contains an array of size 1440×1024. This array contains 1440 images of size 32×32 pixels that are flattened to a vector of 1440 dimension. These images come from 20 objects that are rotated 72 times (5 degrees per image)[1].

Implement the principal component analysis (PCA) as introduced and discussed in the lecture to reduce the dimensionality of the given data.

Your code should have the following structures:

Algorithm PCA

Input: data set X (each column contains a data point), the dimensionality d of the projection

Output: a matrix containing principal components U_d, a matrix (or a vector) containing eigen-values, and reduced version of the data set Z_d

Function PCA(X,d)

centralize the data set $Z = X - \mu$ (μ is computed as the mean of all samples),

compute principal components U and eigen-values D (you can use the built-in function *np.linalg.eig* in Python or *eig* in Matlab)

pick the first d principal components

reduce the dimensionality of the data $Z_d = U_d^T Z$

Report

You should hand in a structured report comprising:

- **(1 point)** An **Introduction** section that describes your assignment.
- **(3 points)** A **Methods** section in which you explain the PCA algorithm in a general manner. You need to implement the PCA algorithm yourself. Code and implementation itself will also be taken into account for the grading of this section.
- **(4 points)** An **Experimental results** section in which you provide the following details:
  - A plot showing the eigen-value profile of the data set, i.e. x-axis is the eigen-values indices (1, 2, $\cdots$ 1024) and y-axis is the eigen-value.
  - A table reporting the dimensionality d if we want to keep 0.9, 0.95 and 0.98 fraction of the total variance. Use the following formula: $\frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{n} \lambda_i}$
  - A plot showing the reduced data Z_d (e.g. $d = 40$) using t-SNE in a 2-dim feature space. You should give different colors to the data points generated

[1] You can visualize the *i*-th original image using *plt.imshow(np.reshape(X[i,:],(32,32)))* in Python or similar functions in Matlab.

from different objects (Every 72 data examples are from one object). You can use the built-in function *sklearn.manifold.TSNE* in Python or *tsne* in Matlab with the default parameter settings.

- **(2 points)** A **Discussion** section that includes your observations on the results.