Machine Learning (HWS22)
Assignment 3: Singular Value Decomposition

The archive provided to you contains this assignment description, a dataset in a binary format, as well as Python code fragments for you to complete. Comments and documentation in the code provide further information.

It suffices to fill out the "holes" that are marked in the code fragments provided to you, but feel free to modify the code to your liking. You need to stick with Python though.

Provide a single ZIP archive with name `ml22-<assignment number>-<your ILIAS login>.zip`. The archive needs to contain:

- A **single PDF report** that contains answers to the tasks specified in the assignment. Do not simply convert your Jupyter notebook to a PDF! Write a separate document, stay focused and brief. **Use at most 10 single-column pages.**
- All the **code** that you created and used in its original format.
- A PDF document that renders your Jupyter notebook with all figures. (If you don't use Jupyter, then you obviously do not need to provide this.)

You need to adhere to the above guidelines in your submission, otherwise we may grade your solution as a FAIL.

Generally, your report should

- include a high-level description of your approach and helpful figures,
- be self-explanatory (i.e., refer to code *only* for implementation-only tasks),
- follow standard scientific practice,
- include appropriate references if you used additional sources or material,
- not include any hand-written notes,
- label all figures/tables and refer to figures/tables via their labels,
- use one section per task and one subsection per subtask, each numbered with the (sub)task numbers from the assignment sheet.

Your report will be downgraded if you do not follow these points (e.g., you can't get EXCELLENT).

Hand-in your solution via ILIAS until the date specified there. This is a hard deadline.

Preliminaries

To compute the SVD of a matrix `X` using NumPy, write

```
import numpy as np
U, s, Vt = np.linalg.svd(X)
S = np.diag(s)
```

Note that `Vt` then has value $\boldsymbol{V}^\top$, not $\boldsymbol{V}$.
To obtain the rank-k approximation `Xk`, write

```
Xk = U[:,:k] @ np.diag(s[:k]) @ Vt[:k,:]
```

We provide a function called `svdcomp` that performs the two steps mentioned above via

```
Xk = svdcomp(X, range(k))
```

1 Intuition on SVD

a) Try to manually obtain the rank of each of the following matrices, as well as its singular values, and the left and right singular vectors corresponding to the *non-zero* singular values. Do this by "looking" at the data and try to infer how the compact SVD needs to look like.

   Do not use computational methods such as solving the characteristic equations. If you fail at this task for some matrices, just write that you failed and don't worry.

b) Compute the SVD (e.g., using NumPy) and compare. Were you correct?

c) What does the best rank-1 approximation look like? Is it "intuitive"?

d) How many non-zero singular values does $\boldsymbol{M}_6$ have, i.e., what is the rank of $\boldsymbol{M}_6$? How many non-zero singular values are reported by NumPy? Discuss!

For convenience, each matrix is also defined in the provided Python script.

$$\boldsymbol{M}_1 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{M}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 2 & 0 \\ 0 & 2 & 1 & 2 & 0 \\ 0 & 2 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{M}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \qquad \boldsymbol{M}_4 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\boldsymbol{M}_5 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \qquad \boldsymbol{M}_6 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

2 The SVD on Weather Data

Load the `worldclim` data as described in the provided code. You can find a description of the dataset under `data/worldclim.txt`.

a) Normalize the data (`climate` matrix) to z-scores and store the result in variable `X`. Considering the data we are using, are the assumptions for normalizing the data reasonable?

b) Compute the SVD $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ and the rank of the normalized data.

c) Plot each of the first 5 columns of $\boldsymbol{U}$. Use the longitude and latitude of each data point as the x and y coordinates, respectively, and the corresponding entry in the left singular vector to color each point (see provided code). Can you interpret the result?

d) Plot some scatterplots between the columns of $\boldsymbol{U}$ using colors to distinguish either their North–South or East–West location (see provided code). Can you interpret the results?

e) Try the different rank selection methods listed below to decide what would be a good size for a truncated SVD. Report the size each method suggests (and when subjective evaluation is needed, say why you picked your choice).

   (i) Guttman–Kaiser criterion
   (ii) 90% of squared Frobenius norm
   (iii) Scree test
   (iv) Entropy-based method
   (v) Random flipping of signs

   What, if any, would be your ultimate choice?

f) Define the root-mean-square error (RMSE) between an $m \times n$ matrix $\boldsymbol{A}$ and an $m \times n$ approximation $\hat{\boldsymbol{A}}$ as

$$\operatorname{rmse}(\boldsymbol{A}, \hat{\boldsymbol{A}}) = \frac{1}{\sqrt{mn}} \|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_F.$$

   Create a noisy version of the normalized climate data by adding i.i.d. Gaussian noise from $\mathcal{N}(0, \epsilon^2)$, where ϵ is a parameter that corresponds to the standard deviation of the noise. You can do this in NumPy via

```
X_noise = X + np.random.randn(*X.shape) * epsilon
```

   Do this for various choices of $\epsilon \in [0, 2]$. Now create a plot with ϵ on the x-axis and the RMSE on the y-axis. For $k \in \{1, 2, 5, 10, 48\}$, add a line for the RMSE between the original data (`X`) and the rank-k truncated SVD reconstruction from the noisy data (i.e., of `X_noise`). Discuss the results.

3 SVD and Clustering

For this task, our goal is to cluster the rows of the data into five clusters and visualize the result.

The entire process is explained in the provided code and works as follows: We first load the coordinates, then cluster the data (without the coordinates as before) into five clusters using the k-means algorithm, and finally create a plot where the x-y coordinates coorespond to longitude and latitude, respectively, and the color to the cluster identifier.

a) Look at the resulting clustering and explain what the clusters may represent (remember, the data contains temperature and rainfall information).

b) For another visualization of the results, plot the data so that the x-axis position comes from the first left singular vector, the y-axis position comes from the second left singular vector, and the color of each point is defined by the cluster identifier. Are the clusters well-separated from each other in the plot or are they mixed? Do some of the clusters look like outliers?

c) Compute the PCA scores of the data points (in `X`) for the first k principal components for $k \in \{1, 2, 3\}$, thereby reducing dimensionality to k. Do this solely (!) using the SVD of the appropriate version of the data. Repeat the clustering and visualization steps of a) with this new data. Did the results change? Why do you think the results changed or did not change?