The usual homework policies (see Homework 1 policies)

Submission:

- Please submit your solutions in a *PDF file*, together with *a .zip file containing all the code needed to reproduce your results.* Mention the students with whom you discussed the homework.
- For the computer problems, include the printout of the code, inputs, outputs, required plots, and discussions needed to answer the questions (when appropriate).

Exercises:

1. (*40 pts*) In this problem you will study how initialization of the k-means clustering algorithm can affect the final clustering result. You will compare two types of initialization for the representative vectors $\{\mathbf{c}_l\}_{l=1,\cdots,k}$.

   - **Random initialization**: sampling components of the representative vectors from a distribution.
   - **k++ initialization**: this is an alternative to random initialization and can be described via the following steps:

     i Randomly select one of the points in the data set and assign it as the initial value of the representative vector $\mathbf{c_1}$.

     ii Select the point in the data set furthest away from $\mathbf{c_1}$. Assign this point as the initial value of the representative vector $\mathbf{c_2}$.

     iii Continue the procedure in step 2 for $\mathbf{c_3}, \mathbf{c_4}$, etc., ensuring that the initial value of the next representative vector $\mathbf{c_l}$ is the data point that is furthest away from the nearest vector among $\mathbf{c_1}, \cdots, \mathbf{c_{l-1}}$.

     iv The initialization is complete when initial values for the k representative vectors $\{\mathbf{c}_l\}_{l=1,\cdots,k}$ have been chosen in this manner.

   (a) Modify the provided script `kMeans_demo.py` to implement k-means clustering with:

      i. random initialization by sampling components using a uniform distribution for each component on the interval $[-2, 12]$.

      ii. The k++ initialization outlined above.

   (b) Test your algorithms on the provided data set. The data set for this problem should be loaded via `np.load("blobs.npy")` with $k = 5$. Create scatter plots for the initialized representative vectors in each cluster for both initialization schemes.

   (c) By running 10 realizations of the clustering for each initialization above with $k = 5$, compare the performance of the two initialization schemes. For both schemes, create appropriate plots to compare the overall coherence of the final clustering result. Determine which initialization yields better performance and explain why you think this occurs.

2. (*30 pts*) Consider the matrix

$$\mathbf{A} = \begin{bmatrix} 3 & 4 \\ -4 & -3 \end{bmatrix}.$$

   (a) Using the `svd()` function, compute the singular value decomposition of $\mathbf{A}$, $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$.

   (b) Find the left singular vectors, right singular vectors, and singular values σ_1, σ_2 based on the decomposition.

   (c) What is the rank of $\mathbf{A}$? How can it be read from the SVD?

   (d) Find the inverse matrix $\mathbf{A}^{-1}$ via the SVD. Note that both $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices.

   (e) Compute the eigenvalues λ_1 and λ_2 of the matrix $\mathbf{A}$ by hand.

   (f) Verify that the determinant satisfies $\det(\mathbf{A}) = \lambda_1\lambda_2$ and $|\det(\mathbf{A})| = \sigma_1\sigma_2$.

3. (*30 pts*) Write a Python program for image compression using low rank approximation via SVD.

   (a) Find an image file of your choice. Create an $m \times n$ matrix $\mathbf{A}$ that contains the gray-scale pixel data from the image, where the entries $0 \leq a_{ij} \leq 1$. If you use a color image, first convert it to a grayscale image. The functions `imread`, `rgb2gray`, and `imshow` may be helpful.

   (b) Create a rank - 5 approximation $\mathbf{A}_5 = \sum_{j=1}^{5} \sigma_j u_j v_j^T$ to the matrix $\mathbf{A}$ using SVD. Show both the original image and the low-rank approximation in the report.

   (c) For $1 \leq r \leq 10$, create a rank - r approximation $\mathbf{A}_r = \sum_{j=1}^{r} \sigma_j u_j v_j^T$ to the matrix $\mathbf{A}$ and compute the approximation error $\|\mathbf{A} - \mathbf{A}_r\|_2$ in 2-norm via `np.linalg.norm(A - Ar, ord=2)` . In the report, create a table showing the approximation errors $\|\mathbf{A} - \mathbf{A}_r\|_2$ for each value of r. How are $\|\mathbf{A} - \mathbf{A}_r\|_2$ related to the singular values of $\mathbf{A}$? Discuss your observations.