Numerical Methods – Second Homework

-Constantinescu Vlad – 314CB-

The archive contains the implementantion of a compression software that uses SVD and PCA, aswell as a digit recognition implementation using the PCA algorithm and the KNN (k-nearest neighbours) algorithm.

- **task1** – The function implements an image compression algorithm using singular value decomposition (SVD). By applying the SVD algorithm to an image matrix photo, reduced matrices U, S, and V are obtained. Using these matrices, an approximation of the original photo matrix is reconstructed, resulting in new_X, which represents the compressed image with a reduced number of principal components given by the parameter k.

⚠ The quality of the image compressed using the SVD algorithm depends on the number of principal components selected (k). Choosing a larger value of k improves the image quality.

- **task2 –** The function implements an image compression algorithm using PCA (principal component analysis). It takes an input image matrix and the desired number of principal components (‘pcs’) as parameters and returns the compressed image matrix by normalizing the initial matrix, resulting in a mean-centered matrix, used to construct the covariance matrix Z. SVD is used on the covariance matrix to obtain the matrices U, S, and V, creates the matrix W by selecting the first ‘pcs’ columns from the matrix V, calculates the matrix Y by multiplying the transpose of ‘ with the mean centered matrix and reconstructs the compressed image matrix new_X by multiplying W with Y and adding the mean of each row, afterwards converting the reconstructed matrix to the ‘uint8’ data type to obtain a valid image representation.

- **task3 -** The function is similar to the ‘task2’ function because it also implements an image compression algorithm using PCA. The difference is that it computes the eigenvectors and eigenvalues of the covariance matrix, the eigenvalues are sorted in descending order, and the corresponding eigenvectors are arranged. The algorithm keeps only the first ‘pcs’ columns of the eigenvector matrix, which represent the most important principal components and gets a new represatation of the image, converting it to ‘uint8’ to obtain a valid image.

⚠ The quality of the compressed image depends on the number of principal components chosen (‘pcs’). By selecting a higher value for ‘pcs’, more information from the original image is retained, resulting in a higher quality compressed image and selecting a lower value for ‘pcs’ leads to a decrease in quality.

- **prepare_data**: The function imports the necessary data from the "mnist.mat" file. It takes two inputs: the dataset name and the number of training images to be used. It returns the training matrix ‘train_mat’ and the corresponding labels ‘train_val’.

- **visualise_image**: The function visualizes a specific image from the dataset. It takes two inputs: the training matrix ‘train_mat’ and the image number ‘number’. It reshapes the image using the transopes variant of the reshaped matrix by using the ‘reshaped’ function and transforming the matrix intro a valid image with the ‘uint8’ function. Uncommenting the last line of code will allow to view the image

- **magic_with_pca:** The function applies PCA to the training matrix. It takes two inputs: the training matrix ‘train_mat’ and the number of principal components ‘pcs’. It implements steps 2 to 9 of the prediction algorithm and returns the transformed training matrix ‘train’, the mean ‘miu’, the projected matrix ‘Y’, and the principal component matrix ‘Vk’.

- **prepare_photo**: The function modifies and transforms a test image into a vector using the ‘reshape’ function to facilitate prediction. The training images have a black background and a white digit, while the test images have a white background and a black digit, so the colors need to be inverted. It takes the test image ‘im’ as input and returns the modified string ‘sir’.

- **KNN**: The function applies the k-nearest neighbors algorithm. It takes four inputs: the labels ‘labels’, the projected matrix ‘Y’, the test image ‘test’, and the number of nearest neighbors ‘k’. It performs the necessary calculations and returns the prediction.

- **classifyImage**: This function combines the previous functions to make the prediction. It takes five inputs: the test image ‘im’, the training matrix ‘train_mat’, the training labels ‘train_val’, the number of principal components ‘pcs’, and the number of nearest neighbors ‘k’. It calls the other functions in sequence and returns the prediction.