All source code is hosted here.

Problem 1.1 (Cross Validation)
Cross validation is a model validation tecnique used for testing the skill of machine learning models.
To perform Cross Validation the data set used to test a classifier is splitted in:

- a Training set: it is used to train the classifier
- a Test set: it is used to evaluate the classifier

Both sets are chosen from the original data set, eventually randomly to increase the performance. To make sets homogeneous a stratification can be performed: every set must contain the same number of instances of each class. To test a classifier some hold out methods can be used:

- Stratified: Training set and Test set are randomly chosen using stratification
- Repeated: training is repeated m times with different random samples to handle sample bias

K-fold Cross Validation
K-fold Cross Validation is a repeated hold out method. It consists of dividing a data set into k groups named folds.

- the first fold is used as test set
- the remaining k-1 folds are used as training set

This method is repeated k time, each time a different fold is used as test set and an accuracy for each test is obtained. The mean of accuracies returns the classifier precision.

Problem 1.2 (Feature)
The answers for this problem are given into `pearson.py` script's console output. Here it is reported:

```
-----Computation and documentation of: -----

Feature - Feature correlations
a-a) 1.0000000000000002
a-b) 1.0000000000000002
a-c) -0.7615128014360222
b-b) 1.0000000000000002
b-c) -0.7615128014360222
c-c) 0.9999999999999999

(Note that the correlation between same class (e.g. a-a) should result 1, but because of
    calculation's approssimation it is not perfectly 1)

Feature - Class correlations
1) 0.8812188319210076
2) 0.8812188319210076
3) -0.8641585652180932

Subset of features          Merit
(0,)                   0.8812188319210076
(1,)                   0.8812188319210076
(2,)                   0.8641585652180932
(0, 1)                 0.8812188319210076
(0, 2)                 0.9298897226715618
(1, 2)                 0.9298897226715618
(0, 1, 2)              0.9259806690392741

-----Answer 1.2-a-----
```

```
If k=2 Naive selector will choose feature 1 and feature 2
It's because these two features are the ones with the highest feature-class correlation.

-----Answer 1.2-b-----
According to the merit the subsets with cardinality 2 that we can choose are both (0,2)=(A,C)
    and (1,2)=(B,C) because they have the same highest merits

-----Answer 1.2-c-----
If we could choose from an arbitrary cardinality of subset, we will choose one of the same two
    subsets of above. Indeed there is no subset with higher merit value
```

Problem 1.3 (Feature selection/Dimension reduction)

a) **Theory**

i) Imagine we are working with a dataset of vehicles. These vehicles have a lot of properties like number of wheels, number of doors, weight, height, shape etc. These are the features, but a lot of them are not so important, in the sense of to help us to distinguish the difference between these vehicles. For example a lot of vehicles have 4 wheels, so this property is not usful to distinguish a car from a bus, however the height is much more relevant. PCA can help us to find and eliminate these redundant features.

ii) In general PCA is used to reduce the dimension of the data by eliminating the redundant features from the dataset.
The goal is to reduce the size of the data and lose as less information as possible. It can be also used to select the most significant features.

b) **Implementation**

At first the dataset `data` is loaded by the provided function `load_dataset_from_folder()`, then due to the function `pca()` (the implementation is discussed below), eigenvalues and eigenvectors are retrieved from `data` with number of principal components `k`. By subtracting the `mean_vector` obtained from the `dataset`, we get the normalized dataset called `mean_adjusted_data` and after that we combine the eigenvectors matrix with `mean_adjusted_data` to get the linear combination. To retrieve the original data, the transformation is inverted in `inverse_transformed_data` by applying a linear combination of the transposed data and adding the `mean_vector` back.

```
if __name__ == '__main__':
    k = 4

    data, datashape = load_dataset_from_folder('data/')
    data = np.array(data, dtype='float')

    eigenvectors, eigenvalues = pca(data, k)

    mean_vector = get_mean_vector(data)
    mean_adjusted_data = subtract_mean_vector(data, mean_vector)

    transformed_data = np.array(np.dot(eigenvectors.T, mean_adjusted_data.T),
        dtype='float').T

    inverse_transformed_data = np.array(add_mean_vector(np.dot(eigenvectors,
        transformed_data.T), mean_vector),
                                    dtype='float').T
```

Below the implementation of the PCA algorithm can be seen. The function works with two parameters: the `dataset` (data) and the number of the principal components which we want to keep (`k`). First it subtracts the mean vector, which means it removes the mean from each feature vector (from each column). In this way we get the normalized dataset.

After, it calculates the covariance matrix of the transpose of the data, and then it extracts the eigenvectors and the eigenvalues from the matrix with the help of an preimplemented function from the libary: `numpy.linalg`.
Then it organises the eigenvectors and the eigenvalues in descending order based on the values of the eigenvalues, and return the first k biggest eigenvectors and eigenvalues.

```
def pca(data, k):
    X = subtract_mean_vector(data, get_mean_vector(data))
    cov = np.cov(X.T)
    eigenvalues, eigenvectors = np.linalg.eig(cov)

    idx = eigenvalues.argsort()[::-1]
    sorted_eigenvalues = eigenvalues[idx]
    sorted_eigenvectors = eigenvectors[:, idx]

    return sorted_eigenvectors[:, :k].real.astype('float'),
        sorted_eigenvalues[:k].real.astype('float')
```

Results:

i) Image reconstruction from the first and the last 2400 principal components. Obviously, reconstruction from the last principal components is worse, because the variance of these components are less, they are not relevant for the image. The main purpose of the PCA is to get rid of these components. Even it used almost every components the reconstruction from the last ones are still very bad, because it skips the most important components.

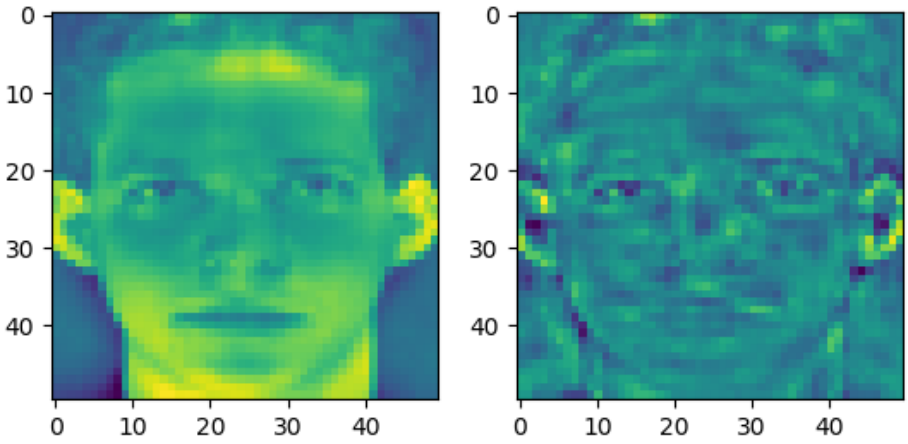

Abbildung 1: Image reconstruction from the first and the last 2400 principal components

ii) Image reconstructed from the first 100 principal components. Reconstrucion from the first 100 principal components is resulting much more precise result, than the reconstruction in the previous case. The eigenvectors with the highest eigenvalues are on the face, for example near the eyes, near the mouth etc.

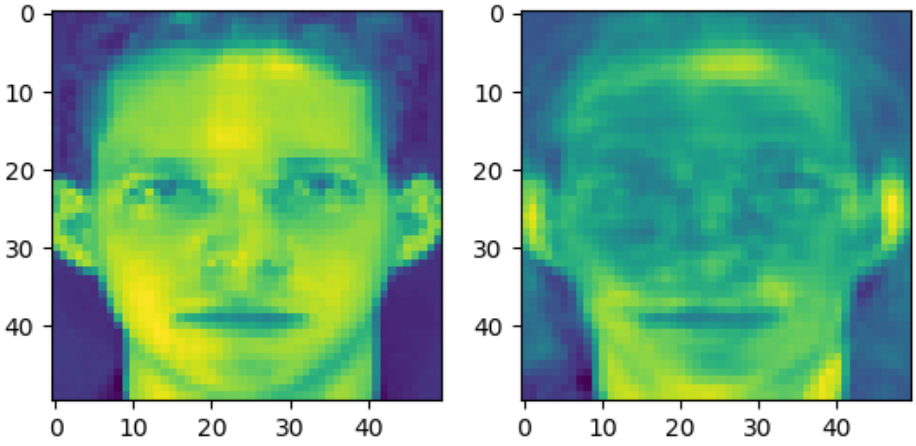

Abbildung 2: Image reconstructed from the first 100 principal componenets

iii) Image reconstructed from the first 400 principal components. Reconstruction from the first 400 component is resulting almost a perfect result. This reconstruction is the best we can get from this dataset, because of the lack of the data points. (We get the same result when we are using all the principal componenets)

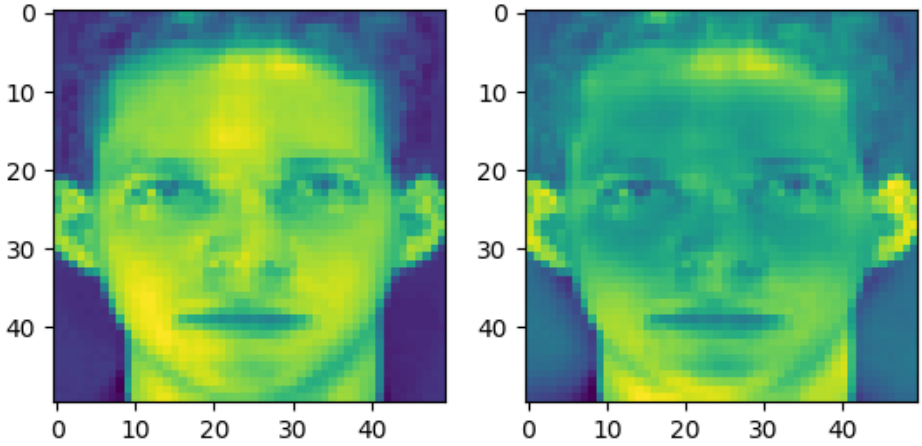

Abbildung 3: Image reconstructed from the first 400 principal componenets

The whole transformation is available here.

c) **Transfer/Understanding**

i) PCA is used in this exercise to find the principal components of faces, and to reduce the dimension of the data.
It can be used further to find the relevant parts of the image, detect the most significant colors, reduce the size of the data etc.

ii) There is no general rule for this question, it all depends on how good you want reconstruct the dataset, how many examples you have, how the variables are correlated, what is the distribution of the data etc. In this concrete example the maximum you can take is 400 components, because we only have 400 instance in our dataset. In this case we will lose some information, because the number of the features are 2500, but the reconstruction will be still good. (Usually the situation is the opposite, so that we have more data than features) If you select all the principal components the reconstruction will be perfect (except for calculation inaccuracies) but also in this case you are not reducing the dimension of the data. In general the best you can do is to

plot the fraction of total variance retained compered to the number of the eigenvectors and make your decision based on it.

iii) Other two approaches are Kernel PCA or Autoencoders:

- **Kernel PCA** Kernel PCA is an extension of the normal PCA algorithm using the kernel trick to transform the data from the feature space into a lower dimensional kernel space. With this technique we are able to construct a nonlinear mapping of the eigenvectors which maximaze the variance of the data. It can perform better when the problem is not linearly separable.
- **Autoencoders** Autoencoder is a type of neural networks, which consists of two network: a generator and a discriminator network and it can be used to reduce the dimension of the data. (They can be used also for feature extractions, or as a generative model) The encoder network tries to describe the input with a fewer dimension, and the decoder tries to reconstruct the original image from this compressed code. The goal is to find a projection method that maps the data from a high a feature space to a low feature space. If you are using the autoencoder with linear activation function or only one sigmoid hidden layers, they work very similar to the PCA, where the hidden layer size if equals with number of the chosed principal components. But the weights of the autoencoder are not the equals to the principal components. The big advantage of the autoencoders compared to the PCA, that the nonlinearity and therefore the more generalization capability. However they require training.

Problem 1.4 (Fourier Transform)
The problem given consists of removing an artificially added white noise by applying a filter on the image "asguard2.png". The filtering tecnique is basically the convolution integral between the function representing the image and a properly selected function. To make the filtering process simpler we need to apply a Fourier transform to convert the image in the frequency domain, in order to obtain a multiplication between the image and the filter instead of a convolution integral. The basic formula of filtering is:

$$G(u,v) = F(u,v)H(u,v)$$

where $F(u,v)$ is the Fourier transform of the image, $H(u,v)$ is the filter and $G(u,v)$ is the spectrum of the filtered image. After we applied the filter we have to apply a Fourier antitransform to obtain the filtered image. The solution for this exercise are below.

a) The filter implemented consists of a low pass filter in order to suppress the high frequency values introduced by a white noise. The circular structure of the low pass filter allows to cut frequecies on the 2D spectrum placed at distances greater than a range given from the center of the image.

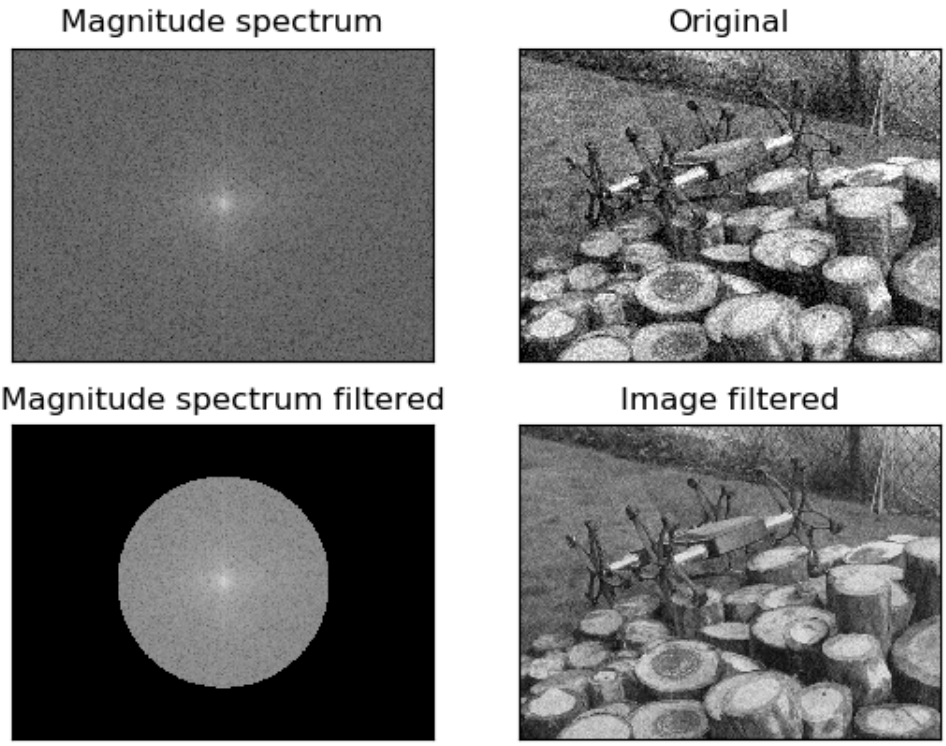

Abbildung 4: Comparison between the original and the filtered image

b) The mean value of grey is represented by the central point of the magnitude spectrum. The cross in the middle shows that the sine waves that only vary along the x and y axes play a big part in the final image. The interesting features are the horizontal bands parallel to X-axis. Having these bands shows that the variation of grey on the Y-axis varies only on some frequecies (the ones where we see the bands). In our case, this feature is explained by the structure of the skyscrapers in the picture. If we "go up" (i.e. move in the Y-axis) on the picture the grey variation is (very often) from the windows to the wall, therefore, we have some "fixed type of variation of the grey". Meanwhile if we look to the other axe (right-left) there are variations from window to window, but they are of various nature, so we will not find a "band" in the magnitude spectrum.

   If we rotate the image, we will see the magnitude spectrum rotated himself by the same value. This is because the variation of greyscale is now following another direction, and thus the sinusional waves (that represent the variation of grey) have a different direction, but same frequency, so that these have the same distance to the center of the spectre (with respect to the magnitude of the original image), that is the total frequency of the wave, but they have different x and y component (accordingly to the rotation of the image).

c) The Fourier transform nowadays is applied in many scientific fields as astronomy, communication, control theory, image processing etc...

   Between the vast field of Fourier transform applications in real world, in this paper, two have been selected:

   - Applying Fourier-Transform Infrared Spectroscopy and Self-Organizing Maps for Forensic Classification of White-Copy Papers [1]
   - Fast Fourier Transform and a Complimentary Filter Based Control of a Robotic System [2]

As it can be seen those two examples are very different applications. The first one is a forensic application where a Fourier transform is used to obtain an infrared spectrum and then analyze it by various tecniques used in machine learning as self organizing maps, PCA, and Cross Validation.

In the second paper a Fast Fourier Transformation is used in a robotic system to handle by a control system the balancing system of a robot. Signals detected by sensors are transposed in the Frequency domain, then they are filtered to remove unwanted measurement noises. A control system then is used to keep the equilibrium of the robot so to modify the measurement of sensors so to work as retroaction.

Problem 1.5 (Reflection)

a) Our expectation from this course is to get a deeper understanding how the ML algorithms works, understand and practice the math behind these algorithms, and later maybe create or extend one. Machine learning is also one of the most popular subject in informatics and other fields (like economics, medical science etc.), and this knowledge is necessary for every computer engineer today.

b) i) We hope that our work make some people life easier. We want to create something which will be useful for future researches (for example algorithms).

ii) Maybe a lot of knowledge that we will learn here will be useful in the future, for our career.

Literatur

[1] Lee, Loong Liong. *Applying Fourier-Transform Infrared Spectroscopy and Self-Organizing Maps for Forensic Classification of White-Copy Papers*. International Journal on Advanced Science, Engineering and Information Technology. 6. 1033. 10.18517/ijaseit.6.6.1425. 2016.

[2] Agon Kokaj, Ines Bula, Artan Dermaku. *Fast Fourier Transform and a Complimentary Filter Based Control of a Robotic System*. IFAC-PapersOnLine Volume 51, Issue 30, 2018, Pages 561-564.