<u>Exercise 2</u> Timo Marks, Munther Odeh 03.05.22

1) a) 1-dim normal distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

μ = mean, σ = standard deviation

b) If we already know that $x^{(1)}, \dots, x^{(N)}$ is normal distributed we can compute

the mean $\mu = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}$

and

the standard deviation $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$

for the normal distribution.

2) If we have different data sets and we want to compare them with eachother, we can use z-scored data, which tells us how many „z"-standard deviations the data point is from the mean μ.

$$z_i = \frac{x_i - \bar{\mu}}{\sigma}$$

Now the same z_i means the same quantity of σ from μ regardless of the specific unit of $x^{(1)}, \dots, x^{(N)}$.

3)

a) N-dimensional normal distribution of a random variable $\vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}$

$$f_x(\vec{x}) = \frac{1}{\sqrt{(2\pi)^N \det(\Sigma)}} e^{\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})\right)}$$

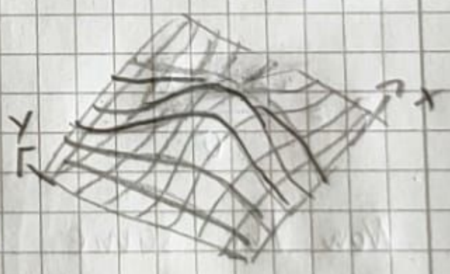
with $\vec{\mu}$ as N-dimensional vector of the mean/expected values $\vec{\mu} = (E(X_1), \dots, E(X_N))^T$

and Σ as the covariance matrix

$\Sigma_{ij} = E[(X_i - \mu_i)(X_j - \mu_j)]$

b) One possible approach could be, if we know that the dimensional entries $(X_1, \dots, X_N)^T$ are all independent from another, that we compute μ_i for each i-th dimension and then compute Σ_{ij} with it. In other courses like „machine Learning I" there was also given the EM-algorithm / Expectation-Maximization-algorithm

c) We can only plot the 2D-normal distribution but in general: $\vec{\mu}$ shows us the peak of the distribution and moves it to the $\vec{\mu}$ position. If Σ is small, then the distribution is narrow and large entries in Σ correspond to a broad distribution, just like in 1D.



4) The mahalanobis distance (MD) is a possible approach to define the distance between two points in a multivariate space or between a vector and a distribution. For the points $\vec{x}, \vec{y}$ it is defined as

$$\Delta(\vec{x}, \vec{y}) = \sqrt{(\vec{x}-\vec{y})^T \Sigma^{-1} (\vec{x}-\vec{y})}$$ with Σ as the covariance matrix.

Intuitively MD gives the distance between $\vec{x}, \vec{y}$ as the multiple of the standard deviations of the corresponding distribution.

This is similar to the z-score, but now only defined for higher dimensions. z-score (=1-dim) also gives the distance of a point $x^{(i)}$ to the mean μ as a multiple of standard deviations σ.

Ex2_part5

May 3, 2022

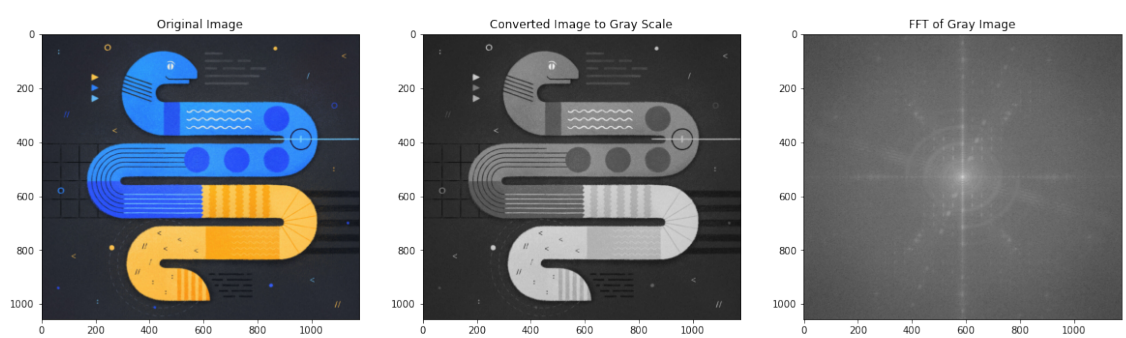
[ ]:
```python
# Done by Munther Odeh and Timo Marks
import numpy as np
from matplotlib import pyplot as plt
from PIL import Image
from skimage.color import rgb2gray
```

[ ]:
```python
# Function for converting image to gray scale -> fft of gray image -> Plot
def fft_of_image(filepath):
    fig = plt.figure(figsize=(20, 20))
    img = plt.imread(filepath)
    gray_img = rgb2gray(img)
    fig1 = fig.add_subplot(1,3,1)
    fig1.imshow(img)
    fig1.title.set_text('Original Image')

    fig2 = fig.add_subplot(1,3,2)
    fig2.imshow(gray_img, cmap="gray")
    fig2.title.set_text('Converted Image to Gray Scale')

    # Fouriertransformation
    gray_img_fft = np.fft.fftshift(np.fft.fft2(gray_img))
    fig3 = fig.add_subplot(1,3,3)
    fig3.imshow(np.log(abs(gray_img_fft)), cmap="gray")
    fig3.title.set_text('FFT of Gray Image')
```

[ ]:
```python
fft_of_image('python-hero.jpg')
```



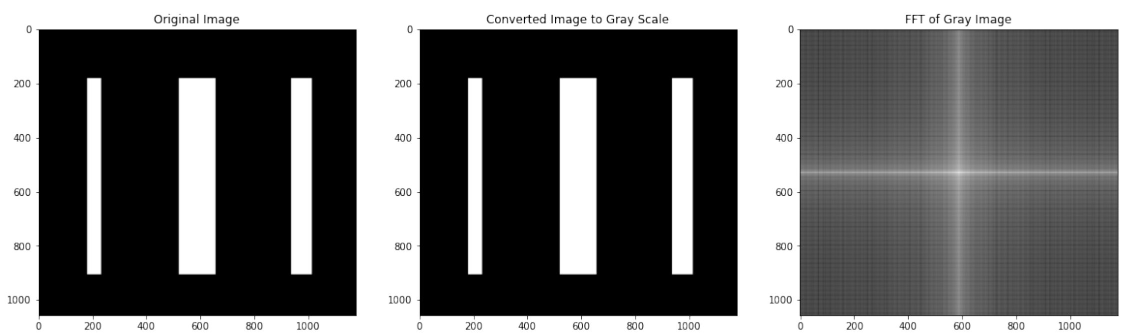
The plots show the absolute value of the frequency components in the image

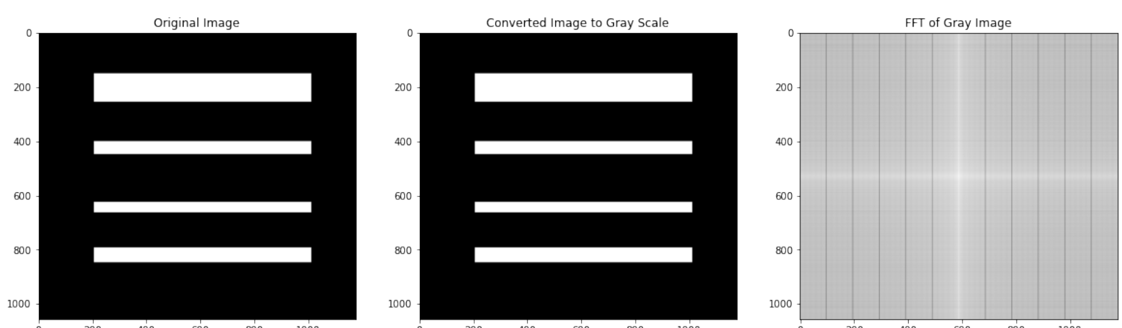
The Python image has a rather complex fourier transformation with bright lines on the main horizontal and vertical axis and a very bright sport in the middle. But we can also see some circular shapes and other radial lines to the outside.

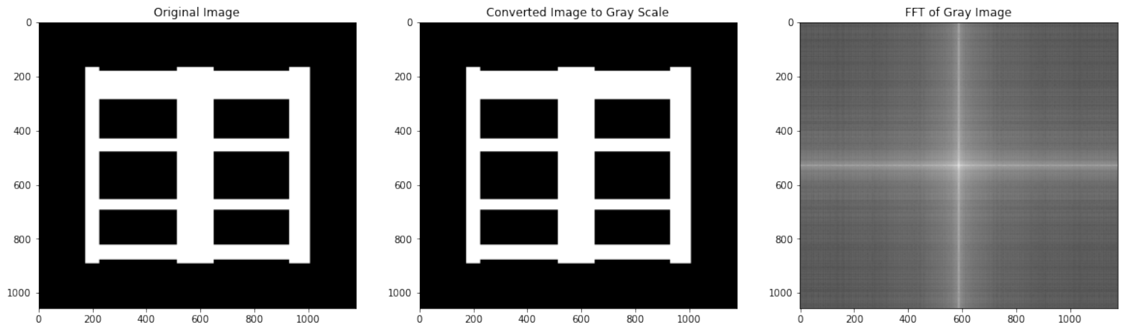
The fourier transformations of the less complex images (with vertical and horizontal white rectangles) have a distinct grid like pattern. In the fourier transformation of the vertical rectangles, we have some vertical dark lines, which means that these frequencies do not occur in the image. They are always a multiple of one another. If we combine both images (Both.png) we also see a combination of both fourier transformations.

One thing to be noted: The vertical/horizontal images have more higher frequency components than the Python image because we need more high frequency components to "create" the sharp edges of the rectangles in the image. We also have these sharp edges/high frequency components in the Python image but not as pronounced.

[ ]:
```
fft_of_image('Horizontal.png')
fft_of_image('Vertical.png')
fft_of_image('Both.png')
```






Original Image
Converted Image to Gray Scale
FFT of Gray Image
0
200
400
600
800
1000