2D Image Processing - Exercise Sheet 4: Statistical Methods and Face Detection

The notebook guides you through the exercise. Read all instructions carefully.

- **Deadline**: 30.06.2022 @ 23:59
- **Contact**: Michael.Fuerst@dfki.de
- Submission: As PDF Printout; filename is `ex04_2DIP_group_XY.pdf`, where XY is replaced by your group number.
- Allowed Libraries: Numpy, OpenCV and Matplotlib (unless a task specifically states differently).
- Copying or sharing code is NOT permitted and results in failing the exercise. However, you could compare produced outputs if you want to. (Btw, this includes copying code from the internet.)

Submission as PDF printout. You can generate a PDF directly from jupyterlab or if that does not work, export as HTML and then use your webbrowser to convert the HTML to a PDF. For the printout make sure, that all text/code is visible and readable. And that the figures have an appropriate size. (Check your file before submitting, without outputs you will not pass!)

---

0. Infrastructure: Cloud Image Loader

This is an image loader function, that loads the images needed for the exercise from the dfki-cloud into an opencv usable format. This allows for easy usage of colab, since you only need this notebook and no other files.

In [2]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

import requests
from PIL import Image

def get_image(name, no_alpha=True):
    url = f'https://cloud.dfki.de/owncloud/index.php/s/THLirfoB6SYTetn/download?path=&files={name}'
    image = np.asarray(Image.open(requests.get(url, stream=True).raw))
    if no_alpha and len(image) > 2 and image.shape[2] == 4:
        image = image[:,:,:3]
    return image[:,:,::-1].copy()
```

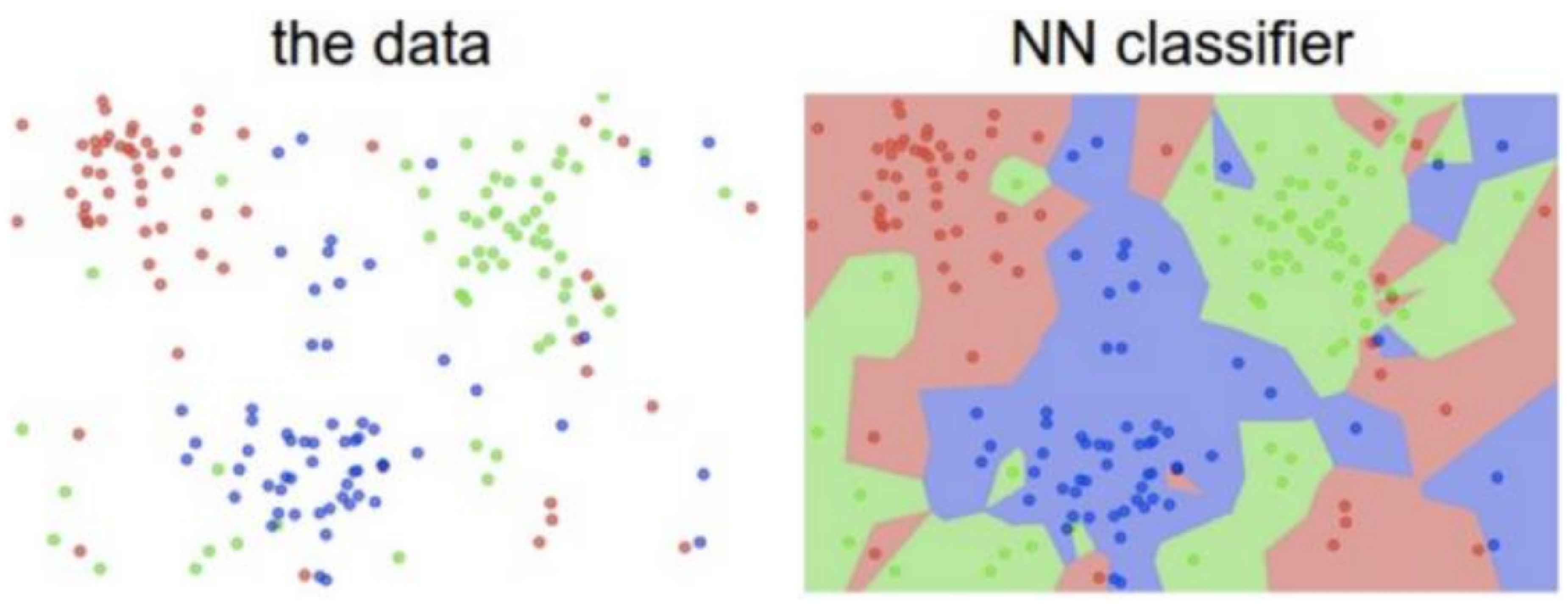
---

1. K-Nearest Neighbour

The K-nearest neighbours (KNN) algorithm is a supervised machine learning algorithm that can be used to solve both classification and regression problems.

Theory (4.5 Pts)

1. What is supervised method with respect to classification? (1 Pt)
2. See Figure beneath, can you briefly explain the workflow of K-Nearest Neighbours (KNN)? (3.5 Pts)



Solution:

1. The idea behind supervised classification is that a user can pick a sample set of pixels from an image that most accurately depicts a certain class, and then instruct the image processing algorithm to use these pixels as references when classifying all other pixels in the image.

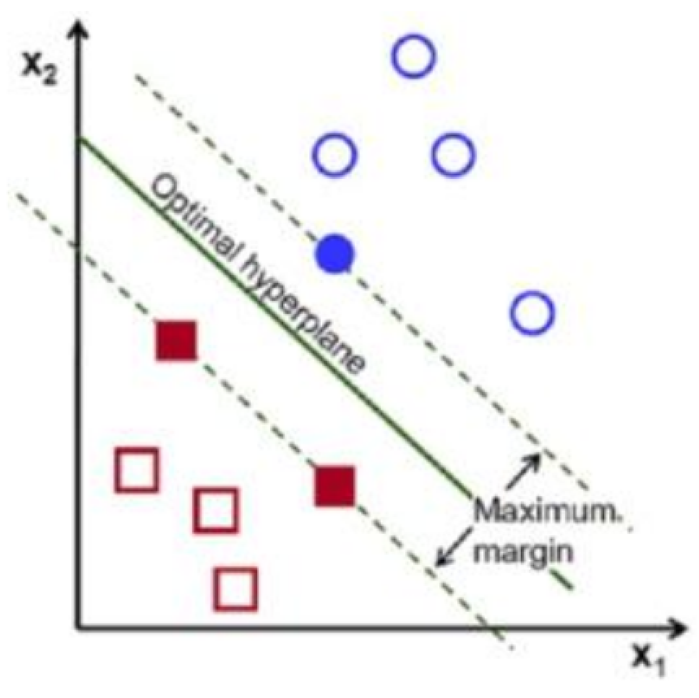
2. The workflow for the K-Nearest Neighbours (KNN) for the above figure will be as following:

   - First, we need to choose the value of K i.e the nearest data points.
   - For each point in test set:
     - Calculate the distance between test data point and training data point.
     - Then, using the labels of the k points, choose the class label.

---

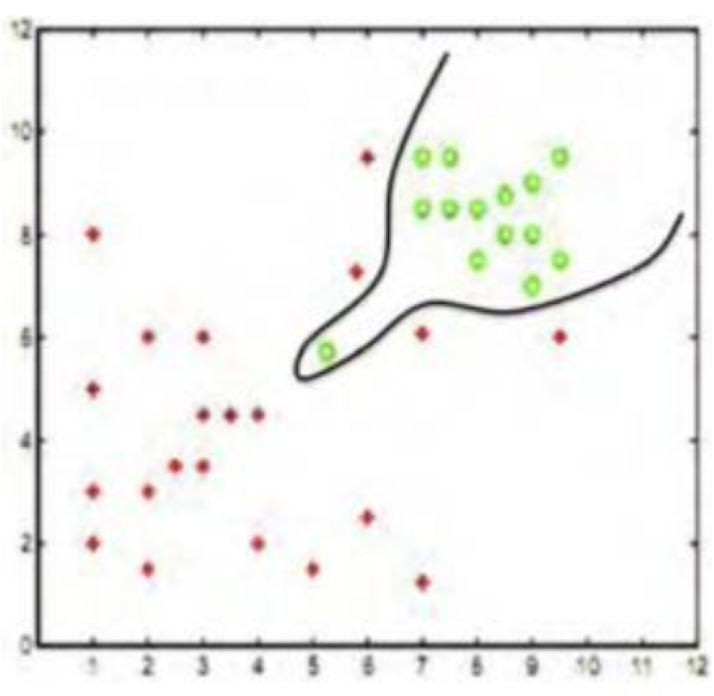
2. Support Vector Machine (SVM)

SVM is a type of classifier which classifies positive and negative examples, here blue and red data points. As shown in the Figure, the max margin is found in order to avoid overfitting and the optimal hyperplane is at the maximum distance from the positive and negative examples (equal distant from the bundary lines).



Theory (3 Pts)

1. What is the intuition of a max margin classifier? (1 Pt)
2. What is a kernel in SVM? Why do we sue kernels in SVM? (1 Pt)
3. See the given Figure (below) and explain the phenomenon that SVM may suffer. (1 Pt)



Solution:

1. SVM works by seperating the features by a hyperplane represented by $w^T x + b = 0$ The goal is to maximize this hyperplain by a margin represented by γ.
   γ in turn can be represented in terms of w by

   $$w^T x_1 + b = 1$$

   (x_1 representing the nearest point in front of hyperplain) Similarly a point behind the hyperlain x_2 can represented as

   $$w^T x_2 + b = -1$$

   This in turn can be represented as

   $$\max_{\gamma, b \epsilon \mathbb{R}, w \epsilon \mathbb{R}^d} \gamma$$

   $$s.\, t y_i (w^T x_i + b) \geq \|w\| \gamma$$

2. Kernals are transformations functions incorporated to transform non seperable data to a higher dimention where it is possible to seperate data using hyperplain. In the higher dimention the input is represented using the dot product which can be easily comuted in the lower dimention. Some examples of kernals are gaussian kernal, polynomial kernal, sigmoid kernal etc.

3. To make svm fit data which is not easily seperable, a penality function is incorporated to accomodate the points lying inside the maximized margin. The complete equation for the same is

   $$\min_{\mathbf{w}, b, \zeta} \left(\frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=0}^{m} \zeta_i \right)$$

   s. t

   $$y_i (\mathbf{w} \cdot \mathbf{x} + b) \geq 1 - \zeta$$

   In the shown image C increases ie, it does not penalize due to which overfitting occours. The model represented in the image tries to fit all the points perfectly thus resulting in overfitting.

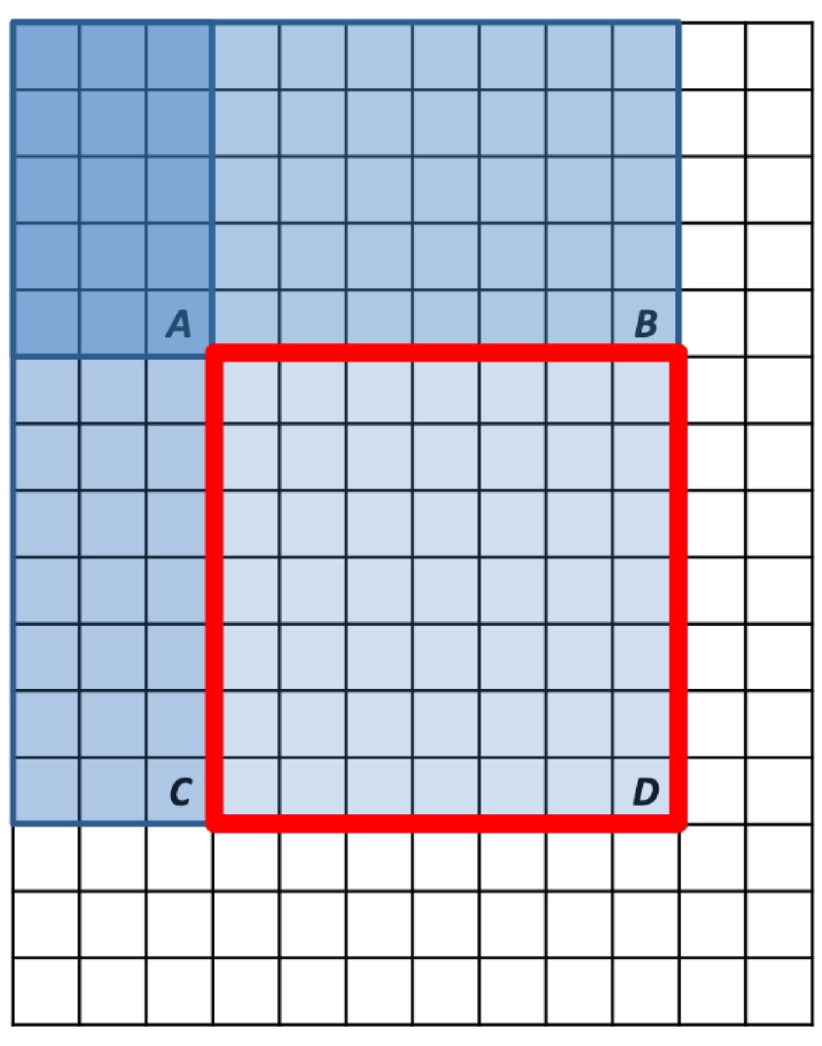
---

3. AdaBoost Face Detection

Theory (4 Pts)

1. What is the intuition of AdaBoost and how can you make a set of weak classifiers a strong classifier? (3 Pts)
2. In the very first exercise of this lecture, you have learned about integral image, see Figure, how do we compute the sum of the pixels in the red box efficiently? Explain using a formula. You may assume you have the integral image $I(x, y)$ as Points A,B,C,D. (1 Pts)

243	239	240	225	206	185	188	218	211	206	216	225
242	239	218	110	67	31	34	152	213	206	208	221
243	242	123	58	94	82	132	77	108	208	208	215
235	217	115	212	243	236	247	139	91	209	208	211
233	208	131	222	219	226	196	114	74	208	213	214
232	217	131	116	77	150	69	56	52	201	228	223
232	232	182	186	184	179	159	123	93	232	235	235
232	236	201	154	216	133	129	81	175	252	241	240
235	238	230	128	172	138	65	63	234	249	241	245
237	236	247	143	59	78	10	94	255	248	247	251
234	237	245	193	55	33	115	144	213	255	253	251
248	245	161	128	149	109	138	65	47	156	239	255
190	107	39	102	94	73	114	58	17	7	51	137
23	32	33	148	168	203	179	43	27	17	12	8
17	26	12	160	255	255	109	22	26	19	35	24



Solution:

1. AdaBoost is build on the intuition of building a good classifier using just the small subsets of all the possible feature. It does so by going through multiple boosting rounds. In each round, a weak learner is selected which does well with the examples which other classifiers were handling hard previously. The weights associated with each example will capture the hardness. Theoretically, to make a strong classifier from of weak learners, a summation of product all considered learners with their respective weight is taken. For h(n) be 'n' number of weak classifiers with 'An' as 'n' no. of weights of weak classifier,
   ```
   (strong classifier) h(x) = (Sign)(Sum t=1,t->n(h(t)*At)
                n = number of weak classifiers
   ```
2. The pixel value (x,y) of integral image is computed iteratively through single pass as the sum of pixel value above and to the left of the current pixel value (x,y) inclusive. In the given image, the sum of pixels in red box ABCD can be calculated using :
   ```
   sum = D + A - C - B
   ```
   Here, D,A,C and B are the integral images at the given point. To calculate the integral image A,B,C and D, sum of integral image of a pixel above and cumulative row sum of a pixel to the left is calculated. Thus,
   ```
   CumulativeRowSum(x,y) = CumulativeRowSum(x-1,y) + i(x,y)
   IntegralImage(x,y) = CumulativeRowSum(x,y) + CumulativeRowSum(x,y)
   ```

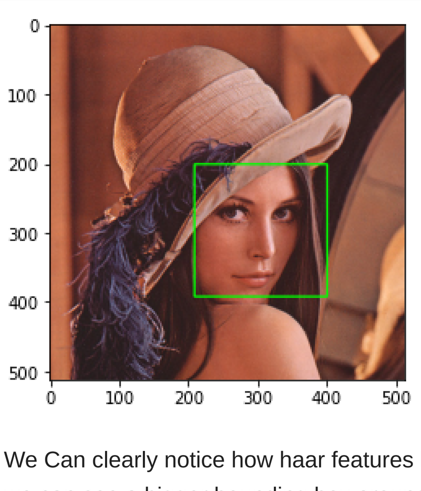
Programming Task (2 Pts)

Apply the face detection from opencv to the lena image and visualize the bounding box around the face.

In [3]:
```
# Solution
img = get_image("lena.png")

gray_image =cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
haar_cascade_face =cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
faces_rect = haar_cascade_face.detectMultiScale(gray_image, scaleFactor=1.2, minNeighbors=8)

for (x,y,w,h) in faces_rect:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
plt.imshow(img[:,:,::-1])
plt.show()
```



We Can clearly notice how haar features help to find the bounding box around the detected image. We have currently selected our scale factor to 1.2. If we increase the scale factore we can see a bigger bounding box aroung the face.

4. Principal Component Analysis (PCA)

In the lecture we have learned that eigenfaces use PCA to represent the images as eigenvectors.

Theory (2 Pts)

1. Can you explain the idea of eigenfaces and how is it used for face recognition.

The basic idea of Eigenfaces is to build a low-dimensional linear subspace that best captures the variation of the set of face images.

For facial recognition, we may use it as follows:

- Assume that the first k directions of maximum variance define a low-dimensional subspace where the majority of face photos are located.
- Find the vectors or "eigenfaces" u1,...uk that span that subspace using PCA.
- Convert all of the dataset's face images into linear combinations of eigenfaces.

In [ ]: