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 [1]:
```
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=&
    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()
```

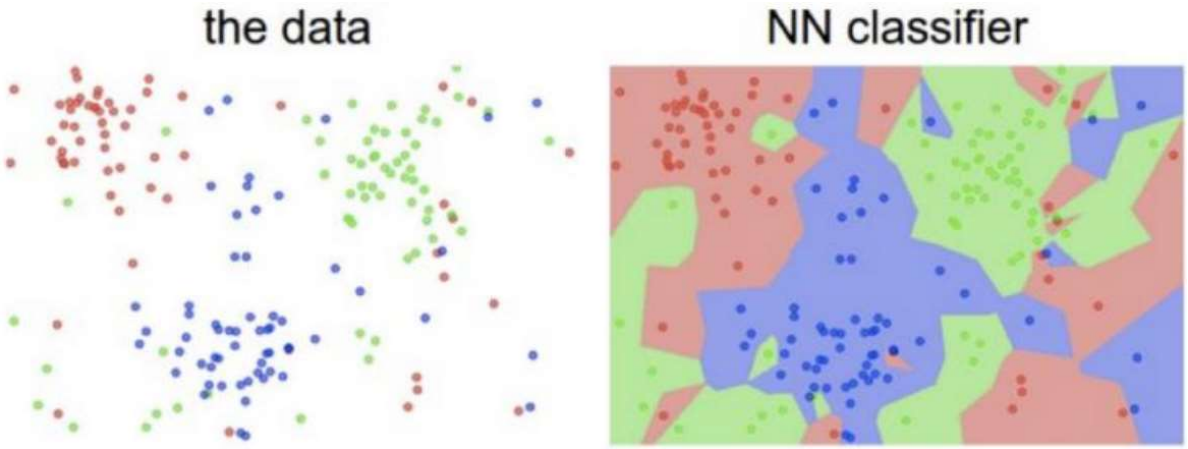
---

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. A supervised method as the name suggests is a method that is being observed and being directed. Classification deals in segregation of the data given. A supvised method in classification consists of algorithms and techniques that learn patterns in data to predict discrete values which are independent of each other.

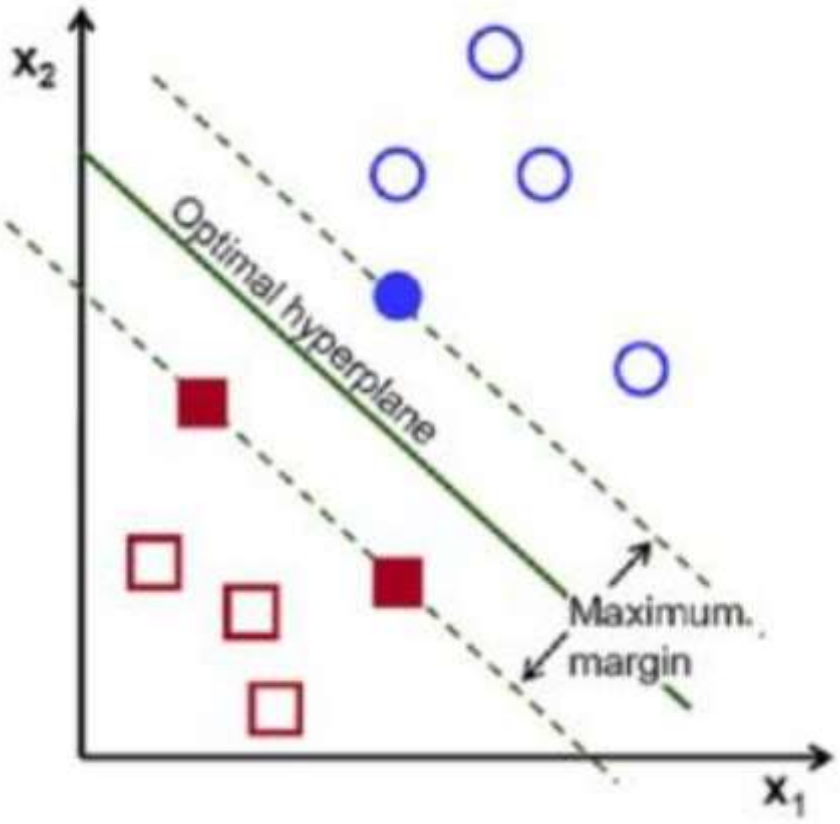
2. In Nearest Neighbour Classifier, only the label of the nearest image is used for prediction. In k-Nearest Neighbour Classifier, k closest images are found in the training set instead of the single closest image and then have them vote on the label of the test image. Therefore if k equals one, then it would function the same way as a Nearest Neighbour Classifier.
In the figure, three classes of data points are visible, namely, red, blue and green. The coloured regions show the decision boundaries obtained by the Euclidean distance in a two dimensional plane. Here as we can see the NN Classifier has been used where k is one and when the k value is low, the classifier doesn't smoothen the boundary, i.e., making it vulnerable to outliers. Here many single points and small regions can be avoided using a higher value of k. There is no unclassified region and has sharp boundaries. This training

data set is indeed specialised to the given data set but our aim is to generalise the boundary for a smoother result.

---

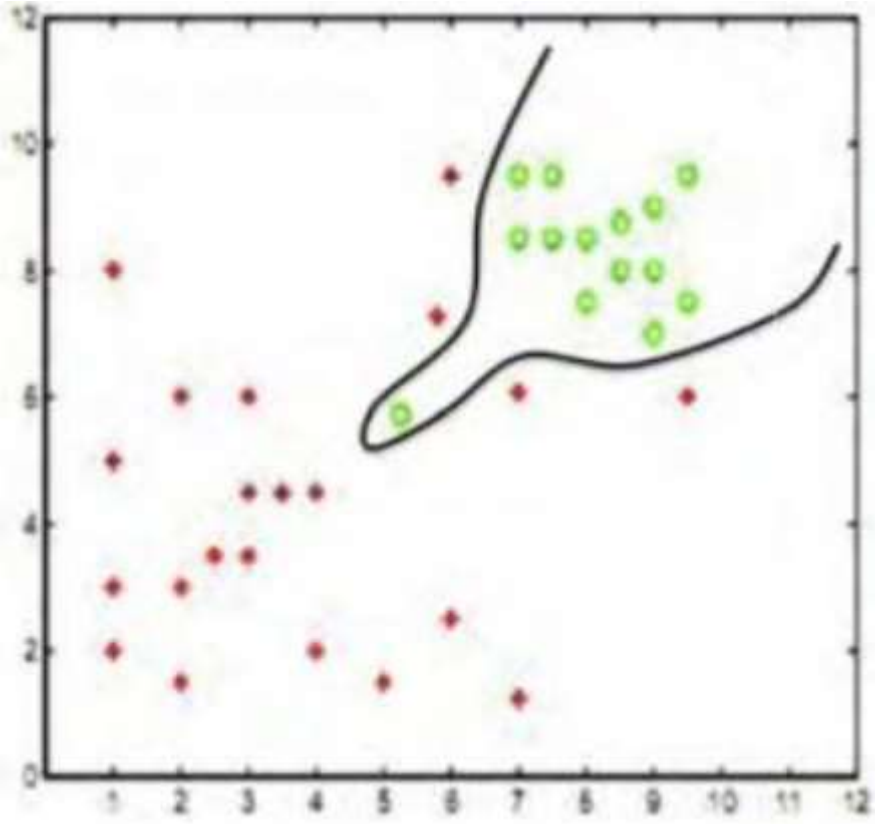
2. Support Vector Machine (SVM)

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. We know that the distance from a point (x_0, y_0) to a line:
   ax + by + c = 0, is:

$$\text{distance}(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}} ,$$

-> The distance between H0 and H1 is : |w•x+b|/||w||=1/||w||

->The total distance between H1 and H2 is thus: 2/||w||

->We need to minimize ||w|| in order to maximize the margin.

->Considering there are no datapoints between H1 and H2:

xi•w+b ≥ +1 when yi = +1,
xi•w+b ≤ –1 when yi = –1
These two equations can be combined into : yi(xi•w) ≥ 1

2. A kernel is a two-argument real-valued function over X × X (κ : X × X → ℜ) such that for any x, z ∈ X κ (x, z) = (φ (x), φ (z))F for some inner-product space F such that ∀ x ∈ X φ (x) ∈ F.

->The method which takes data as input and transform into the required form of processing data is a kernel function.

->Kernel is used due to set of mathematical functions used in Support Vector Machine provides the window to manipulate the data. The Kernel Function generally transforms the training set of

data so that a non-linear decision surface is able to transformed to a linear equation in a higher number of dimension spaces. It returns the inner product between two points in a standard feature dimension.

3. The SVM shown in the figure may suffer from overfitting.

-> The model fits irrelevant characteristics (noise or outlier) in the training data. The training error is low but the test error is high.

-> The amount of training data is not sufficient and the model is too complex.

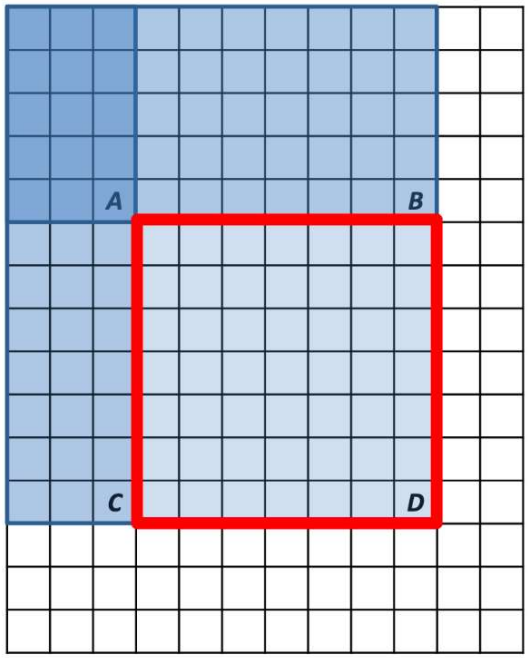
---

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. AdaBoosting is a classification scheme that works by combining weak learners into a more accurate ensemble classifier. A weak learner need only do better than chance.
   A set of weak classifiers can be made into strong classifiers by training.
   a. Training consists of multiple boosting rounds,During each boosting round, we select a

weak learner that does well on examples that were hard for the previous weak learners.
b. "Hardness" is captured by weights attached to training examples
c. Each weak classifier learns by considering one simple feature
d. T most beneficial features for classification should be selected by updating the weights
e. The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

2. The integral image computes a value at each pixel (x,y) that is the sum of the pixel values above and to the left of (x,y), inclusive.

Consider the integral image I(x, y) having points A, B, C, D. Need to calculate the sum of I (x, y) that is at point D. Also need to calculate the sum at each point having co-ordinates A = I (x-1, y-1), B= I (x, y-1) and C= I (x-1, y).
After calculating the sum at each point, that is sum of all the pixel values above and to the left of the point.
A = I(x-1, y-1) = 3168
B= I(x, y-1) = 8815
C= I(x-1, y) = 7856
D = I(x, y) = 20,256

After calculating the integral sum at each point, we have to use the below formula to calculate the integral sum of the image in the red box. We need to add sum at point D and subtract the sum of point B and C. Since we have subtracted the sum of A point twice, so we need to add sum of point A again to calculate the sum of integral image.

Integral sum = D – B – C + A
= I(x, y) - I(x, y-1) - I(x-1, y) + I(x-1, y-1)
= 20,256 – 8815 – 7856 + 3168
= 6753

Programming Task (2 Pts)

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

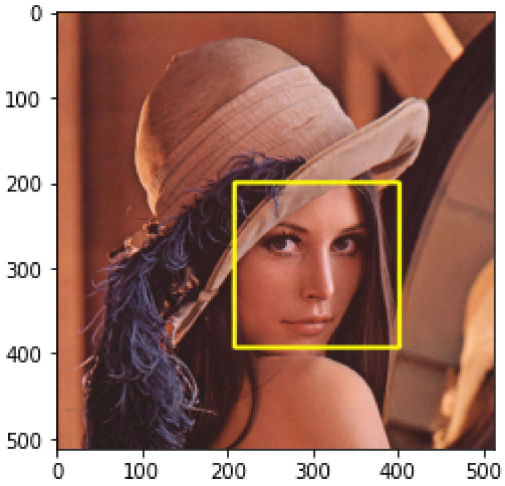
In [2]:
```
# Solution
import cv2

# Load the pretained algorithm
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_
img = get_image("lena.png")
gray_scale = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#Detect the face
faces = face_cascade.detectMultiScale(gray_scale, 1.1, 4)
# Draw rectangle around the face
for (x, y, w, h) in faces:
```

```
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 255), 4)

# Display the output
plt.imshow(img[:,:,::-1])
plt.show()
```



---

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.

Principle Component Analysis

- The direction that captures the maximum covariance of data is the eigen vector corresponding to the largest eigen value of the data covariance matrix
- Furthermore, the top k orthogonal directions that captures the most variance of the data are the k eigen vectors corresponding to the k largest eigen values Eigen faces
- Assume that most face images lie on a low-dimensional subspace determined by the first k (k<d) directions of maximum variance
- Use PCA to determine the vectors or "eigenfaces" $u_1, \ldots u_k$ that span that subspace
- Represent all the faces images in the dataset as linear combinations of eigenfaces
  Training images – $X_1, \ldots, X_n$
  Mean -μ

  Eigen faces examples:
- Face x in "face space" coordinates:
  $X -> (u_1^T(x_i - \mu), \ldots, u_k^T(x_i - \mu))$
- Reconstruction
  $\hat{x} = \mu + w_1 u_1 + w_2 u_2 + w_3 u_3 + \ldots.$

Recognition with eigen values

Process labelled training images:

• Find mean μ and covariance matrix Σ

• Find k principal components (eigenvectors of Σ) $u_1, \ldots u_k$

• Project each training image xi onto subspace spanned by principal components:

$(w_{i1}, \ldots, w_{ik}) = (u_1^T(x_i - \mu), \ldots, u_k^T(x_i - \mu))$

Given novel image x:

• Project onto subspace: $(w_1, \ldots, w_k) = (u_1^T(x - \mu), \ldots, u_k^T(x - \mu))$

• Optional: check reconstruction error $x - \hat{x}$ to determine whether image is really a face

• Classify as closest training face in k-dimensional subspace

In [ ]: