TKO_3120 Machine Learning and Pattern Recognition

Image recognition

Exercise info

Petra Virjonen

pekavir@utu.fi

University of Turku

Department of Future Technologies

7.2.2019

Outline

- Task in general
- Data
  - Data set
  - Data import
  - Data preparation
- Feature extraction
  - Gray-Level Co-Occurence Matrix
- Example of nested cross validation with k-NN
- Task in detail
- Reporting
- Grading

Task in general

- Image recognition
  - Color images
  - One object to be recognized in each image
- Train classifiers to identify the class of an image
  - kNN
  - Regularized linear model
  - Multi-layer perceptron MLP
- Estimate the performance of the classifier
- Use Python or Matlab

Data

- image-net.org
  - An image dataset
  - Images are quality-controlled and human-annotated
  - Does not own the copyright of the images
  - Provides thumbnails and URLs of images
    - A lot of URLs with "Page not Found"…
- Three sets of image URLs provided as textfiles in Moodle
  - Bird nests
  - Lighthouses
  - Honeycombs
- Bonus task: Choose your own object, retrieve the URLs for it, and use it as the fourth image set

Data import

- Read the URLs from the text files

```
import numpy as np
from skimage import io

urls = np.loadtxt('lighthouse.txt', dtype='U100')
# read the first image
url = urls[0]
img = io.imread(url)
```

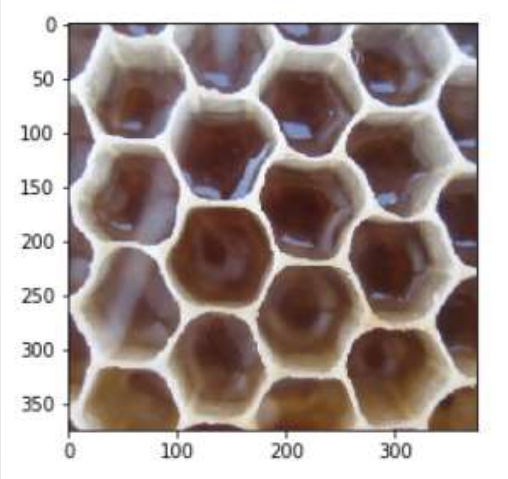
Data preparation

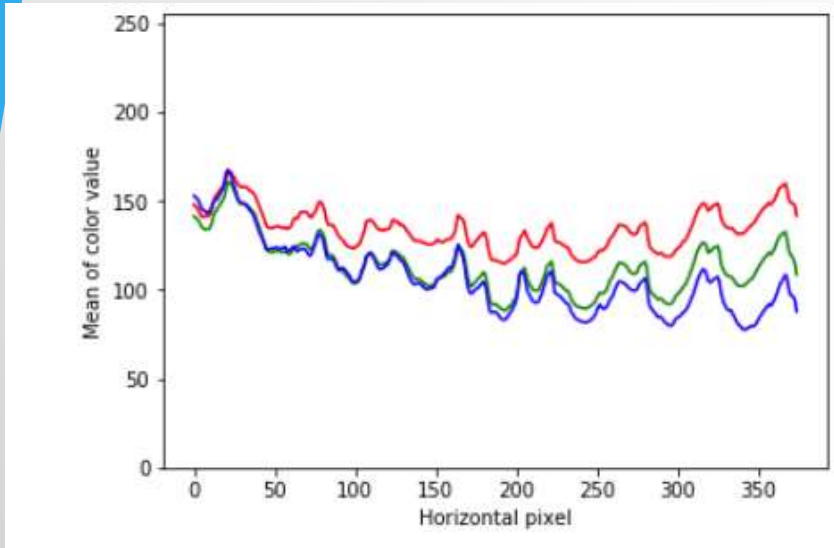
- Crop and/or resize the images into same size
- For GLCM
  - Change the images into grayscale
  - Reduce the quantization level e.g. to 8 levels

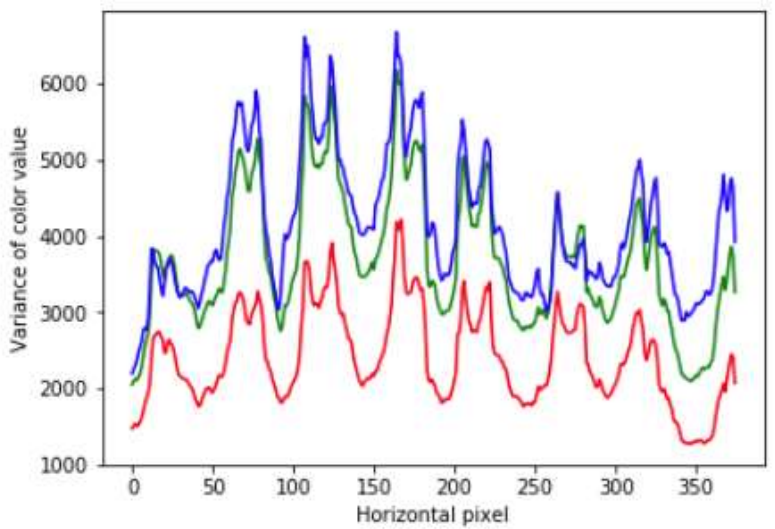
Feature extraction

- First order texture measures: statistics
  - Mean for each RGB color channel
  - Variance for each RGB color channel
- Second order texture measures: relationship among neighboring pixels
  - Gray-Level Co-Occurrence GLCM matrix
- Bonus task: use some other feature. Include a reference to the method you chose

Example of color channels







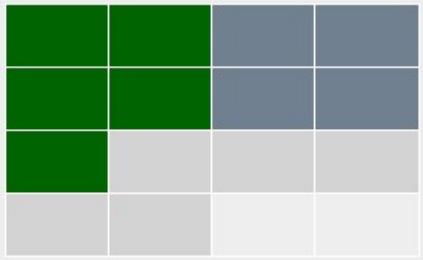

```
r=img_cropped[:,:,0]
g=img_cropped[:,:,1]
b=img_cropped[:,:,2]

# calculate the mean color value in vertical direction for each color
r_mean=np.mean(r, axis=1)
g_mean=np.mean(g, axis=1)
b_mean=np.mean(b, axis=1)

# calculate the variance of color value in vertical direction for each color
r_var=np.var(r, axis=1)
g_var=np.var(g, axis=1)
b_var=np.var(b, axis=1)
```

Gray-Level Co-Occurrence (GLCM) matrix

- A way to measure texture
  - Rough: Large difference between high and low points, i.e. brightness values
  - Silky: little difference between high and low points
- How often different combinations of pixel brightness values occur in an image?
- To the right, to the left, in the direction of diagonal
- https://www.ucalgary.ca/mhallbey/glcm_texture

Example of GLCM

Toy image



0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

neighbour pixel value -> ref pixel value:	0	1	2	3
0	0,0	0,1	0,2	0,3
1	1,0	1,1	1,2	1,3
2	2,0	2,1	2,2	2,3
3	3,0	3,1	3,2	3,3

matrix for one pixel to the right

2	2	1	0
0	2	0	0
0	0	3	1
0	0	0	1

- The number of rows and columns equals the quantization level of the image
  - Reduce the quantization level!
- For texture calculations, the matrix needs to be
  - Symmetric
  - Normalized

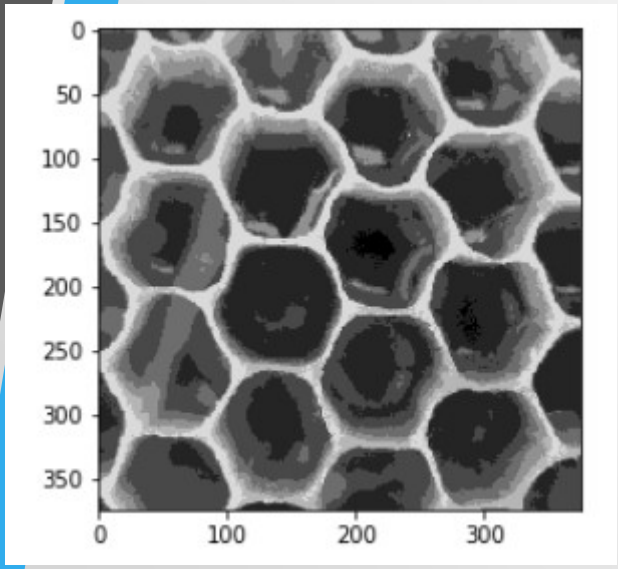
GLCM texture measures

- Contrast
- Dissimilarity
- Homogeneity
- Angular Second Moment ASM
- Correlation

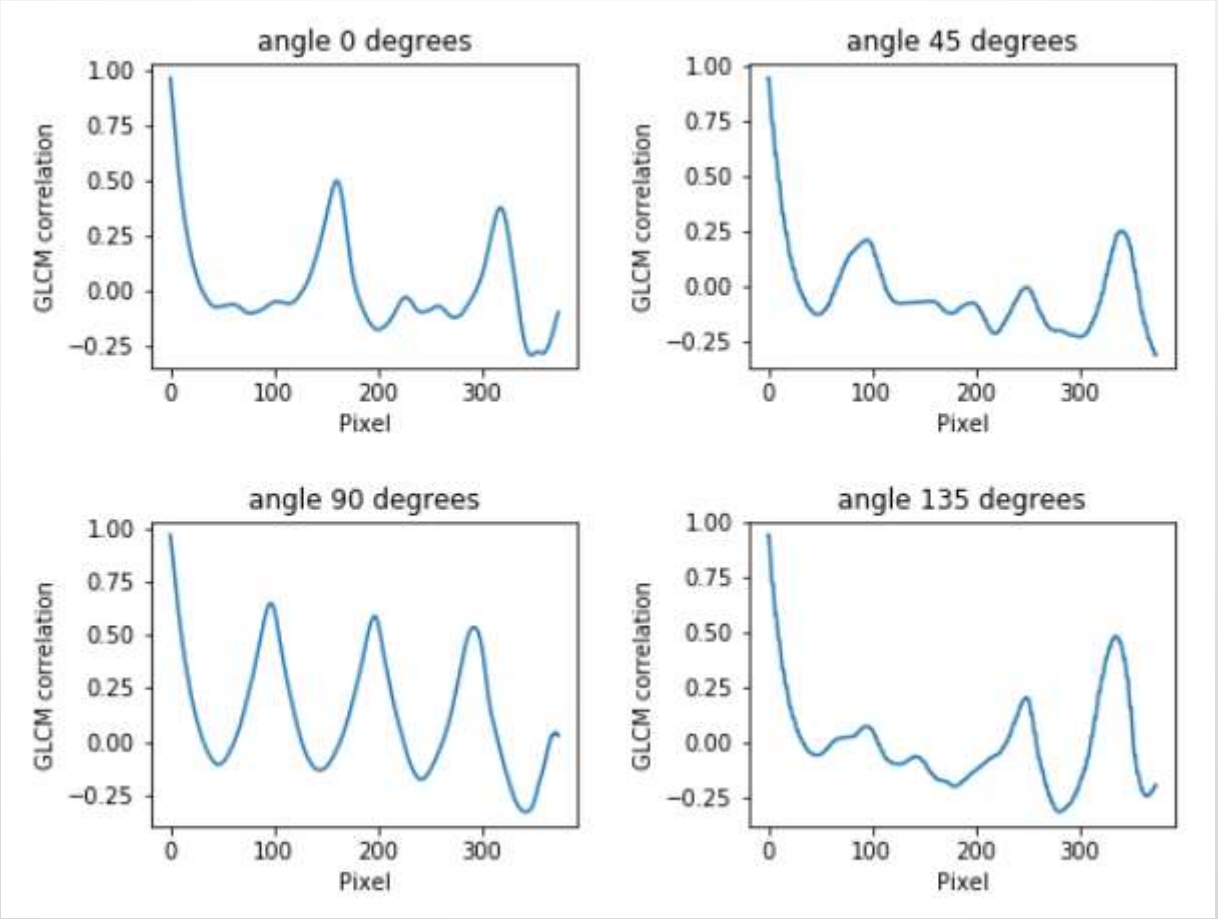
$$\sum_{i,j=0}^{N-1} P_{i,j} \left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right]$$

Example of GLCM correlation

Honeycomb image



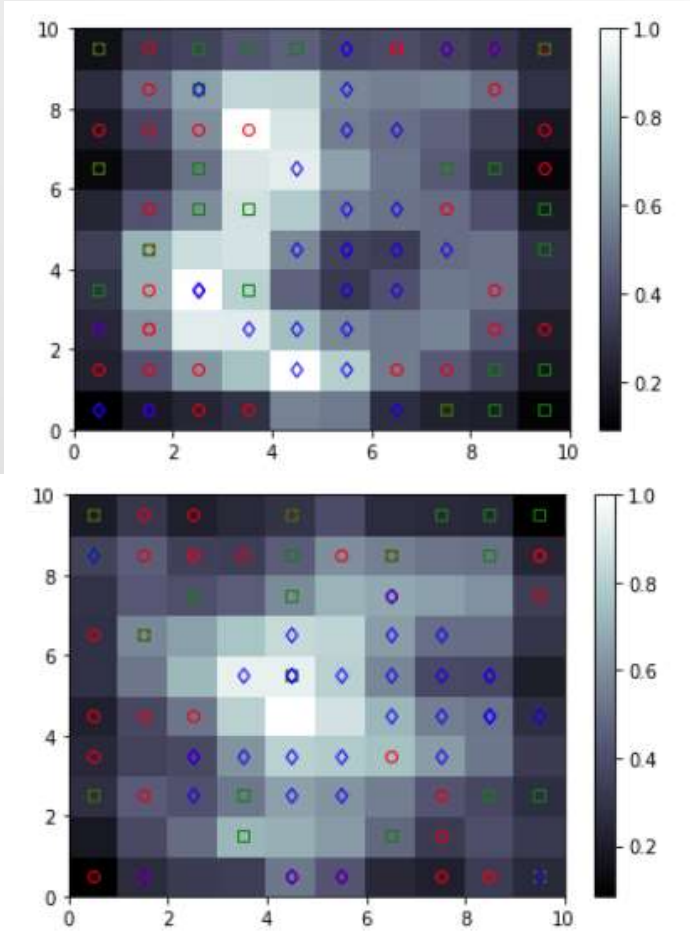
http://farm4.static.flickr.com/3045/2932572848_6c482ff3e0.jpg



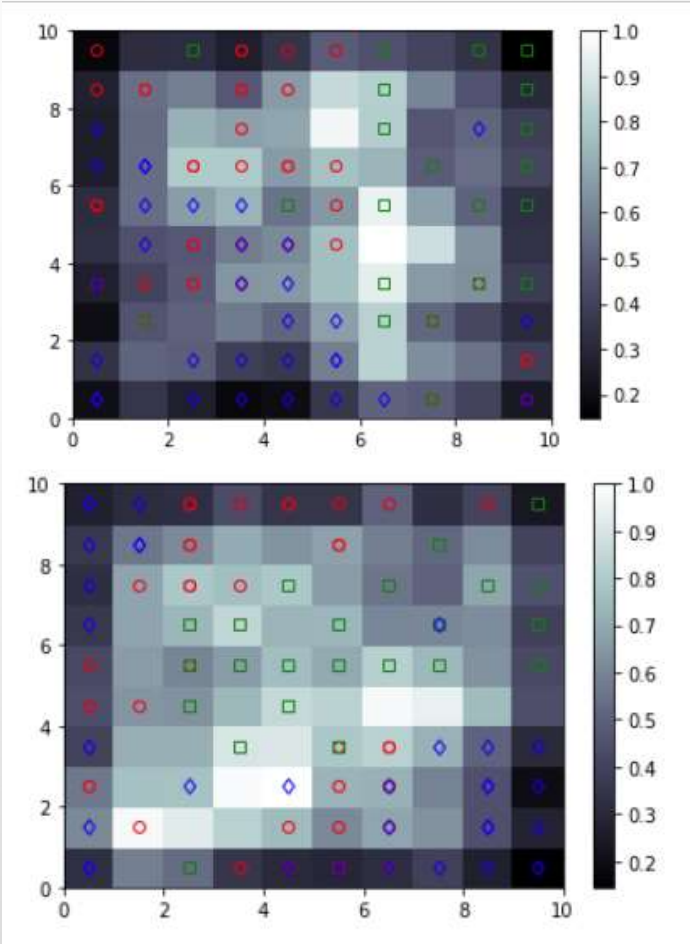
SOM illustration

- You can use e.g. https://github.com/JustGlowing/minisom

RGB mean and variance



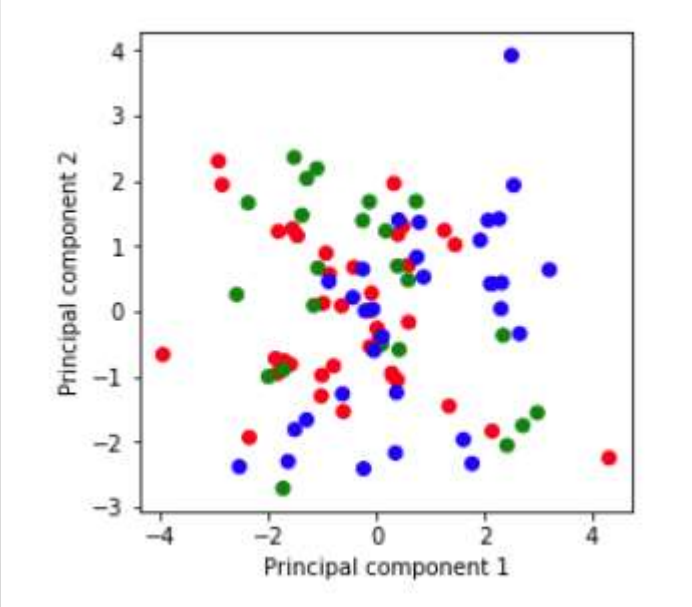
RGB mean and variance and GLCM 6 features



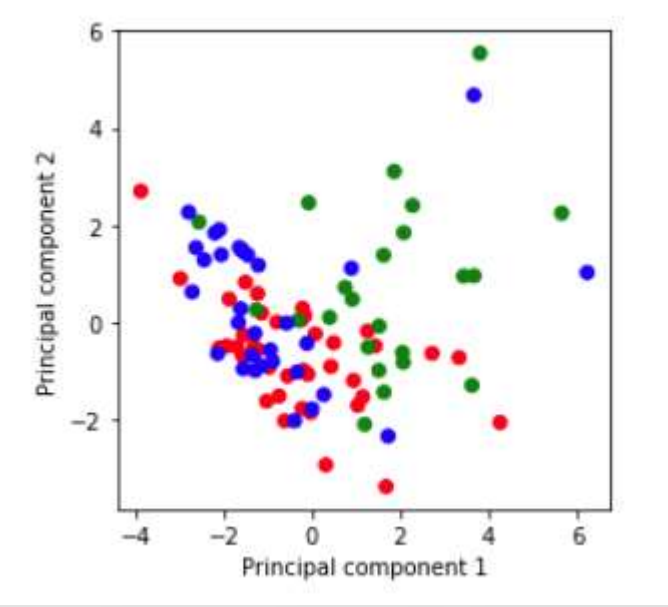
```
markers=['o', 's', 'd'] # birdnest, lighthouse, honeycomb
colors=['r', 'g', 'b']
```

PCA

RGB mean and variance



RGB mean and variance and GLCM 6 features



Example of estimating the accuracy of a classification method with nested leave-one-out cross validation

- Optimizes the hyper-parameters of the method in the inner loop
- Tests the method using the optimal parameters with unseen data in the outer loop
- Facilitates the comparison between different methods

```python
from sklearn import metrics
from sklearn.model_selection import LeaveOneOut
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix

k_range = range(1,21)
image_range = range(X.shape[0])

# Leave-one-out
loo = LeaveOneOut()

k_best=[]
y_preds=np.zeros(X.shape[0])
```

Estimation of kNN with optimized hyperparameter k

6 features:

- Mean for RGB color channels
- Variance for RGB color channels

```python
for image in image_range:

    # divide the data as test and train data for each loo loop
    test_index=[image]
    train_index=np.delete(image_range, image)

    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    perc=[]

    for k in k_range:
        checkfit=[]

        # make the knn search for each train-test-set made by loo.split(X)
        for subtrain_index, validation_index in loo.split(X_train):

            X_subtrain, X_validation = X[subtrain_index], X[validation_index]
            y_subtrain, y_validation = y[subtrain_index], y[validation_index]

            # make the class prediction using knn from sklearn
            knn = KNeighborsClassifier(n_neighbors=k)
            knn.fit(X_subtrain, y_subtrain)
            y_pred = knn.predict(X_validation)

            # save the accuracy of image type prediction
            checkfit.append(metrics.accuracy_score(y_validation, y_pred))

        # calculate the average prediction accuracy for each loo set
        perc.append(np.mean(checkfit))

    # find the best k-value
    k_best.append(k_range[perc.index(max(perc))])
    # train the knn using the best k-value
    knn = KNeighborsClassifier(n_neighbors=k_best[image])
    knn.fit(X_train, y_train)
    # save the predicted type of the test image
    y_preds[image]=knn.predict(X_test)[0]

print(confusion_matrix(y_preds, y))

accuracy=np.sum(np.diagonal(confusion_matrix(y_preds, y))/X.shape[0])
```

Estimation of kNN with optimized hyperparameter k cont.

Birdnests
Lighthouses
honeycombs

```
[[31 11  8]
 [ 6 10  1]
 [ 2  2 23]]
The estimated accuracy for the method is: 0.68
```

- Usually different k:s are acquired within different inner loops
- Only RGB information is not adequate for classifying the images → need better features

KERAS

- The Python Deep Learning Library
- https://keras.io/
- Does not work with Python 3.7
- Setup a new environment with Python 3.6, Tensorflow, Keras (and all the other packages you need for the task)
  - Instructions in the Moodle
- Watch tutorials in YouTube to get started, e.g.
  - https://youtu.be/wQ8BIBpya2k

Tasks

- Import the images and prepare them for analysis
  - Read the URL-lists and open the images
  - Bonus: Choose fourth class of images and find the URLs for images
  - Size the images properly
- Feature extraction
  - Choose at least 10 features, e.g.
    - RGB mean, RGB variance
    - Features derived from GLCM matrix
    - Bonus: some other feature(s)
- Standardization of the feature values
- Make illustrations of the feature relationships
  - SOM
  - PCA
- Try different classifiers, and optimize their parameters
  - K nearest neighbors
  - Regularized linear model with ridge regression
  - Multi-layer perceptron with 1 hidden layer and early stop committee
    - Output function Softmax
- Estimate the classifier accuracy
- Discuss which method performs the best? Why? What are the limitations? How could you improve the result?

Reporting

- Write a report from which also an unspecialized reader could grasp your work
- Return your output as
  - a working Jupyter notebook or
  - Matlab Live Script file or
  - Pdf report combined from text and code, and a working Matlab .m file
- You may use Python template MLPT19_exercise_templ.jpynb
- Name your file as MLTP19_exercise_surname.jpynb or .m
- Write a report with clear sections, e.g.
  - Introduction
  - Data set
  - Methods
  - Data preparation
  - Calculations
  - Results
  - Discussion
- Write easy readable code
  - Comment the code
  - Avoid redundant code! If you exploit some code from the web, provide a reference. Exploit relevant parts of the code and modify it for your purposes to produce what you need

TKO_3120 Machine Learning and Pattern Recognition

exercise
Steve Student
student number 000000
steve.student@utu.fi
2/7/2019

Image recognition of color images

Introduction

General characterization of the task, what was the purpose?

Data set

Describe the data set, what kind of data have you used? Where is it from, how did you acquire it?

Methods

What methods have you used? (SOM, PCA, GLCM, nested cross-validation, regularized linear model, MLP, other?) For what purpose? Write a short description of each. Report the parameters.

Data preparation

How did you prepare the data for analysis? Why? Did you exclude any data? Why?

In [4]:
```
# code with comments
```

Some example figures of the data

Calculations

Classifier 1

In [5]:
```
# code with comments
# training and evaluation of performance
```

Some example figures of correctly and erraneously classified figures.

Classifier 2

In [6]:
```
# code with comments
# training and evaluation of performance
```

Some example figures of correctly and erraneously classified figures.

etc

Results and discussion

What kind of results did you get? What affects on the accuracy of the classifiers? How could they be improved? Which method performs the best? Why?

Grading

- 4 points
- 2 bonus points
- Added as extra points to exam score (exam total 30 points)
- Course will be graded after you have completed and passed the exercise

- Start now!
- If you encounter problems, Google first and if you can't find an answer, ask help
  - pekavir@utu.fi
  - visit at Agora 4th floor room 452B
- **Deadline 30th of April at 23:55**
- No extension granted unless extremely justified reason