SGN-41007 Pattern Recognition and Machine Learning

Exercise Set 1: October 28 – November 1, 2019

Exercises consist of both pen&paper and computer assignments. Pen&paper questions are solved at home before exercises, while computer assignments are solved during exercise hours. The computer assignments are marked by `python` and Pen&paper questions by `pen&paper`

Before you start, load all the data for all questions from the following links. We will need them in later weeks as well.

```
http://www.cs.tut.fi/courses/SGN-41007/Ex1_data.zip
http://www.cs.tut.fi/courses/SGN-41007/least_squares_data.zip
http://www.cs.tut.fi/courses/SGN-41007/locationData.zip
```

---

1. `pen&paper` *Maximum likelihood estimation.* Consider the model

   $$x[n] = A\cos[n] + w[n], \qquad n = 0, 1, \ldots, N-1,$$

   where $w[n] \sim \mathcal{N}(0, \sigma^2)$. In other words, we assume that our measurement is a sinusoid at fixed frequency and phase and want to estimate the amplitude A. Derive the maximum likelihood estimator of A.

   Hint: The same procedure as on the Thursday 10.1. lecture applies: Just write the probability of observing $\mathbf{x} = (x[0], \ldots, x[N-1])$, and maximize with respect to A. The only difference to lecture case is that you have the known signal $\cos[n]$ as an additional variable. However, it is known, so just treat is at as a constant (it will be part of the end result).

2. `pen&paper` *Maximum likelihood estimation 2.* The *Poisson distribution* is a discrete probability distribution that expresses the probability of a number of events $x \geq 0$ occurring in a fixed period of time:

   $$p(x; \lambda) = \frac{e^{-\lambda}\lambda^x}{x!}$$

   We measure N samples: $x_0, x_1, \ldots, x_{N-1}$ and assume they are Poisson distributed and independent of each other. Find the maximum likelihood estimator of λ, *i.e.*, do the following steps.

   a) Compute the probability $p(\mathbf{x}; \lambda)$ of observing the samples $\mathbf{x} = (x_0, x_1, \ldots, x_{N-1})$.
   b) Compute the natural logarithm of p, *i.e.*, $\log p(\mathbf{x}; \lambda)$.
   c) Differentiate the result with respect to λ.
   d) Find the maximum of the function, *i.e.*, the value where $\frac{\partial}{\partial\lambda} \log p(\mathbf{x}; \lambda) = 0$.

3. `python` *Load Matlab data into Python and plot it.*

   a) Load the file `twoClassData.mat` into Python (found in `Ex1_data.zip`). This can be done as follows.

```
>>> from scipy.io import loadmat
>>> mat = loadmat("twoClassData.mat")
```

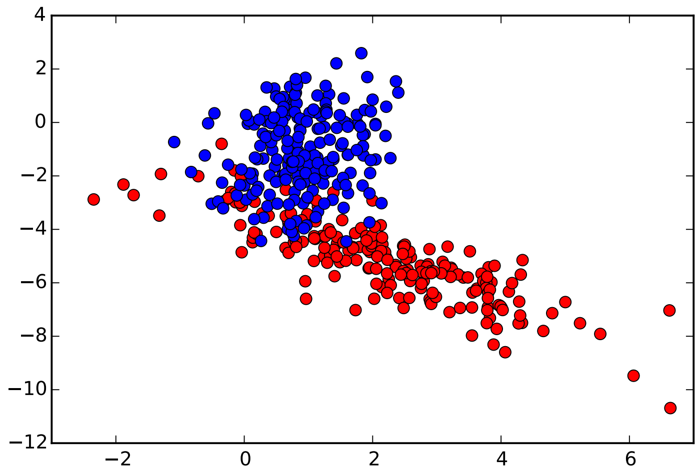
   This generates a **dict** structure, whose elements can be accessed through their names.

```
>>> print(mat.keys()) # Which variables mat contains?
['y', 'X', '__version__', '__header__', '__globals__']
>>> X = mat["X"] # Collect the two variables.
>>> y = mat["y"].ravel()
```

   The function `ravel()` transforms y from 400×1 matrix into a 400-length array. In Python these are different things unlike Matlab.

   b) The matrix X contains two-dimensional samples from two classes, as defined by y. Plot the data as a scatter plot like the picture below. Hints:

   - You can access all class 0 samples from X as: `X[y == 0, :]`.
   - The samples can be plotted like: `plt.plot(X[:, 0], X[:, 1], 'ro')`



4. `python` *Remove the uneven illumination from a microscope image using least squares fitting like we did at the lecture.*

   Load the following Python template and test image:

```
http://www.cs.tut.fi/courses/SGN-41007/exercises/task5_template.py
http://www.cs.tut.fi/courses/SGN-41007/exercises/uneven_illumination.jpg
```

Load the image into numpy array `Z` using the `imread` function of `matplotlib.image`[1]. Finally, show the image on screen (using `matplotlib.pyplot.imshow`) and check that the image shape is 1300×1030. Let's next fit a 2nd order surface to the grayscales.

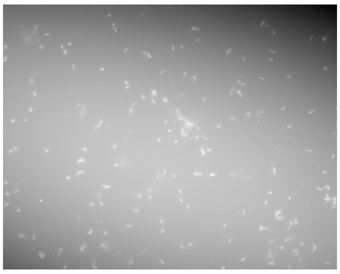
a) Create the explanatory variables (all x,y-coordinates in a matrix): `X, Y = np.meshgrid(range(1300), range(1030))`

b) Vectorize the matrices $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ using `ravel`, *e.g.*, `z = Z.ravel()`.

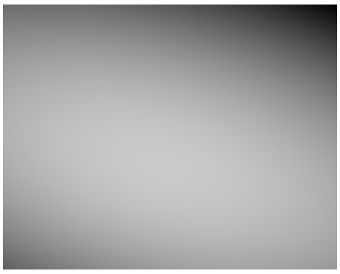
c) Prepare the design matrix $\mathbf{H}$ like in the lectures and solve the LS coefficients $\mathbf{c}$.

d) Compute the model prediction as `z_pred = np.dot(H, c)` and resize the vector to the original size.

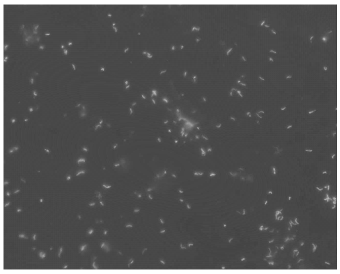
e) Subtract the model prediction from the original image and show the result on screen.



-



=



5. **python** *Estimate sinusoidal parameters.*

a) Generate a 100-sample long synthetic test signal from the model:

$$x[n] = \sin(2\pi f_0 n) + w[n], \qquad n = 0, 1, \ldots, 99$$

with $f_0 = 0.015$ and $w[n] \sim \mathcal{N}(0, 0.3)$. Note that $w[n]$ is generated by `w = numpy.sqrt(0.3) * numpy.random.randn(100)`. Plot the result.

b) Implement code from estimating the frequency of `x` using the maximum likelihood estimator:

$$\hat{f}_0 = \text{value of } f \text{ that maximizes } \left| \sum_{n=0}^{N-1} x(n) e^{-2\pi i f n} \right|.$$

Implementation is straightforward by noting that the sum expression is in fact a dot product:

$$\hat{f}_0 = \text{value of } f \text{ that maximizes } |\mathbf{x} \cdot \mathbf{e}|,$$

with $\mathbf{x} = (x_0, x_1, \ldots, x_{N-1})$ and $\mathbf{e} = (e^{-2\pi i f \cdot 0}, e^{-2\pi i f \cdot 1}, \ldots, e^{-2\pi i f \cdot (N-1)})$.

Use the following template and fill in the blanks.

[1]Note: There are several other ways of doing this, such as: `matplotlib.image.imread`, `scipy.ndimage.imread`, `PIL.Image.open` or `cv2.imread`

```
scores = []
frequencies = []

for f in numpy.linspace(0, 0.5, 1000):

    # Create vector e. Assume data is in x.
    n = numpy.arange(100)
    z = # <compute -2*pi*i*f*n. Imaginary unit is 1j>
    e = numpy.exp(z)

    score = # <compute abs of dot product of x and e>
    scores.append(score)
    frequencies.append(f)

fHat = frequencies[np.argmax(scores)]
```

c) Run parts (a) and (b) a few times. Are the results close to true $f_0 = 0.015$?