

SGN-41007 Pattern Recognition and Machine Learning

Exercise Set 2: January 14.1–18.1.2019

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 text `python` and Pen&paper questions by text `pen&paper`

1. `pen&paper` *Least squares fit.* Two measurements $x(n)$ and $y(n)$ depend on each other in a linear manner, and there are the following measurements:

n	0	1	2
$x(n)$	7	9	2
$y(n)$	11.6	14.8	3.5

We want to model the relationship between the two variables using the model:

$$y(n) = ax(n) + b.$$

Find the least squares estimates \hat{a} and \hat{b} that minimize the squared error.

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

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

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 $\mathbf{x} = (x[0], \dots, x[N-1])$, 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 it as a constant (it will be part of the end result).

3. `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, \dots, 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.

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

4. **python** *Estimate sinusoidal parameters.*

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

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

with $f_0 = 0.017$ and $w[n] \sim \mathcal{N}(0, 0.25)$. Note that $w[n]$ is generated by `w = numpy.sqrt(0.25) * numpy.random.randn(100)`. Plot the result.

- Implement code from estimating the frequency of \mathbf{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 $\mathbf{x} = (x_0, x_1, \dots, x_{N-1})$ and $\mathbf{e} = (e^{-2\pi i f \cdot 0}, e^{-2\pi i f \cdot 1}, \dots, e^{-2\pi i f \cdot (N-1)})$.

Use the following template and fill in the blanks.

```
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)]
```

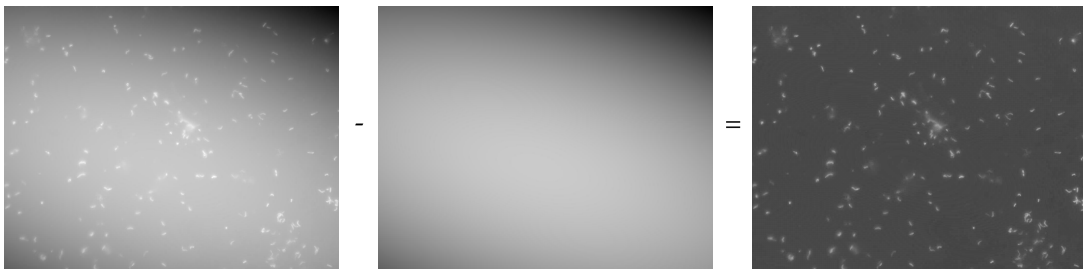
- c) Run parts (a) and (b) a few times. Are the results close to true $f_0 = 0.017$?
5. **python** *Remove the uneven illumination from a microscope image using least squares fitting like we did at the lecture.*

You may use the Python template as a basis:

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

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.

- Create the explanatory variables (all x,y -coordinates in a matrix):
`X, Y = np.meshgrid(range(1300), range(1030))`
- Vectorize the matrices X, Y, Z using `ravel`, e.g., `z = Z.ravel()`.
- Prepare the design matrix H like in the lectures and solve the LS coefficients c .
- Compute the model prediction as `z_pred = np.dot(H, c)` and resize the vector to the original size.
- Subtract the model prediction from the original image and show the result on screen.



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