SGN-41006 Signal Interpretation

Exercise Set 2: January 20 - 21, 2016

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`** *Find a maximum likelihood estimator for one sample.*

   Suppose we measure one sample x for which we assume the model

   $$p(x, \lambda) = \begin{cases} \lambda \exp(-\lambda x) & x \geq 0 \\ 0, & x < 0 \end{cases}$$

   a) Write and simplify the expression for $\ln p(x; \lambda)$. For simplicity, you can omit the case $x < 0$.

   b) Compute the derivative $\frac{\partial \ln p(x;\lambda)}{\partial \lambda}$.

   c) Solve $\frac{\partial \ln p(x;\lambda)}{\partial \lambda} = 0$ with respect to λ. This is the formula for maximum likelihood estimate of λ.

2. **`pen&paper`** *Find a maximum likelihood estimator for many samples.*

   Suppose we measure samples $\mathbf{x} = (x[0], x[1], \ldots, x[N-1])$ for which we assume the model

   $$p(x[n], \lambda) = \begin{cases} \lambda \exp(-\lambda x[n]) & x[n] \geq 0 \\ 0, & x[n] < 0 \end{cases}$$

   a) Write and simplify the expression for $\ln p(\mathbf{x}; \lambda)$. You can assume the samples are independent and $p(\mathbf{x}; \lambda)$ is simply the product of individual probabilities: $p(\mathbf{x}; \lambda) = \prod p(x[n], \lambda)$.

   b) Compute the derivative $\frac{\partial \ln p(\mathbf{x};\lambda)}{\partial \lambda}$.

   c) Solve $\frac{\partial \ln p(\mathbf{x};\lambda)}{\partial \lambda} = 0$ with respect to λ. This is the formula for maximum likelihood estimate of λ.

3. **`python`** *Same as last week Question 1, but without numpy.*

   Download the following file and extract the contents:

   `http://www.cs.tut.fi/courses/SGN-41006/exercises/locationData.zip`

   a) Read the file into memory one line at a time (in a for loop). See similar example at the end of lecture slide set 1.

b) Load the same data into another variable using `numpy.loadtxt`. Check that the contents of the two arrays are equal using `numpy.all` or `numpy.any`.

4. $\boxed{\texttt{python}}$ *Implement two utility functions we need later.*

a) Implement the following function:

```
p = gaussian(x, mu, sigma)
```

which returns the Gaussian density with parameters `mu` and `sigma` at point `x`. In mathematical notation, the function is:

$$p(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x-\mu)^2\right]$$

b) Implement the function:

```
p = log_gaussian(x, mu, sigma)
```

that returns $\ln p(x; \mu, \sigma)$. Do not use the the previous function, because the straightforward solution `p = numpy.log(gaussian(x, mu, sigma))` would be numerically inaccurate. Instead, first manually simplify the expression

$$\ln p(x; \mu, \sigma) = \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x-\mu)^2\right]\right)$$

and write the corresponding code.

5. $\boxed{\texttt{python}}$ *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 $f_0 = 0.017$ 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)`.

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.

```python
scores = []
frequencies = []

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

    # Create vector e. Assume data is in x.

    e = numpy.exp(-2 * numpy.pi * 1j * f * n)
    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 $f_0 = 0.017$?