DATA.ML.200 Pattern Recognition and Machine Learning

Exercise Set 2: Estimation theory

Be prepared for the exercise sessions (watch the demo lecture). You may ask TAs to help if you cannot make your program to work, but don't expect them to show you how to start from the scratch.

1. **python** *LS and ML estimators. (10 pts)*

   In this exercise you use the loss functions you defined in Homework 1 to estimate the sinusoid frequency f_0. Below is the code to generate a noisy sinusoid.

```
# Form a sinusoidal signal
N = 160
n = np.arange(N)
f0 = 0.06752728319488948

x0 = A * np.cos(2 * np.pi * f0 * n+phi)

# Add noise to the signal
sigmaSq = 0.0 # 1.2
phi = 0.6090665392794814
A = 0.6669548209299414

x = x0 + sigmaSq * np.random.randn(x0.size)

# Estimation parameters

A_hat = A*1.0
phi_hat = phi*1.0
fRange = np.linspace(0, 0.5, 100)
```

   In this exercise we find f_0 using *brute force* and the two loss functions (SSE and ML). Brute force in the sense that we test all values in the variable "fRange", compute SSE and ML, and then select the frequency for which SSE loss is minimized or ML loss is maximized.

   a) Write code that tests all values of "fRange" and then plots the two losses, the true signal, noisy measurement points and finally the estimated sinusoid (Figure 1).

   b) You may notice that at the end of the sinusoid the signals do not quite match. This is due to the fact that the samples do not quite match the true frequency. Keep the same search region, $[0.0, 0.5]$, but add more samples (100 in the above code), until the match is good enough.

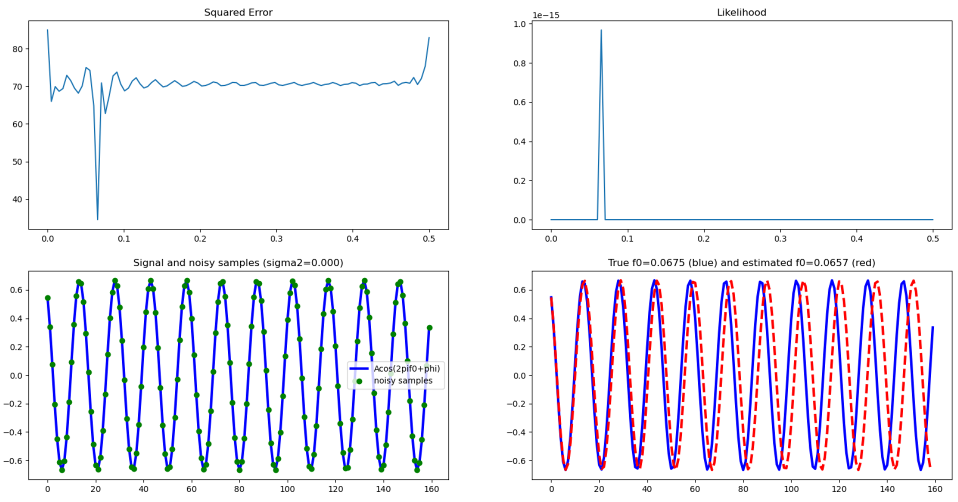

Figure 1: Example of what your code should produce.

c) Test how much noise you can add (“sigmaSq”) until your method fails to estimate the frequency f_0.

d) Another weakness of this approach is that we assume that we know the exact values of the amplitude A and phase ϕ. Test that if you add error to them, e.g., 10% (“phi_hat = phi*1.10”), then what happens to the maximum noise level where f_0 can still be estimated.