ECE 4803, Fall 2020

Homework #2

Due Thursday, September 3, at 9:30am

1. Prepare a one paragraph summary of what we talked about in class in the last week. I do not want just a bulleted list of topics, I want you to use complete sentences and establish context (Why is what we have learned relevant? How does it connect with other classes?). The more insight you give, the better.

2. Suppose we have a signal $f(t)$ on $[0, 1]$ which is "bandlimited" in that it only has $N = 2B+1$ Fourier series coefficients which are non-zero:

$$f(t) = \sum_{k=-B}^{B} \alpha_k e^{j2\pi kt}, \tag{1}$$

for some set of expansion coefficients

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_{-B} \\ \vdots \\ \alpha_0 \\ \vdots \\ \alpha_B \end{bmatrix} \in \mathbb{C}^N.$$

We observe samples M samples of $f(t)$ at locations $t_1, t_2, \ldots, t_M$ which are **not** necessarily uniformly spaced,

$$y[m] = f(t_m), \quad m = 1, \ldots, M. \tag{2}$$

   (a) Write a Python script that takes a vector of length M containing the sample locations and a dimension $N = 2B + 1$ (which you can assume is odd), and computes an $M \times N$ matrix $\boldsymbol{A}$ such that when $\boldsymbol{A}$ is applied to a vector of Fourier series coefficients (as in (1)), it returns the sample values in (2).

   (b) The file `samptimes.csv` contains $M = 259$ sample times t_m and the file `y.csv` contains the corresponding sample values $f(t_m)$. Find the signal of bandwidth $N = 21$ (so $B = 10$) that best explains these samples in the least squares sense. Plot your **synthesized estimate** $\hat{f}(t)$ as a function of time.

3. The standard least squares objective is to minimize

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 = \sum_{m=1}^{M} (\boldsymbol{a}_m^T \boldsymbol{x} - y_m)^2,$$

where $\boldsymbol{a}_m^T$ represents the m^{th} row of $\boldsymbol{A}$. This inherently assumes that finding an explanation for each y_m is equally important. There are often situations where you might have more confidence in some measurements over others. For instance, you may have different sensors which yield measurements that have been corrupted by different levels of noise. If you know

how much noise has been added to each y_m, you can compensate for this by introducing weights w_m and consider the modified objective function

$$\sum_{m=1}^{M} (\boldsymbol{a}_m^T \boldsymbol{x} - y_m)^2 = \sum_{m=1}^{M} w_m (\boldsymbol{a}_m^T \boldsymbol{x} - y_m)^2. \tag{3}$$

One could set the weights w_m to be higher for the y_m which are known to be reliable, and lower for the y_m that one suspects are particularly noisy.

(a) Show that the so-called *weighted least squares* objective function in (3) can be written as

$$\|\boldsymbol{W}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})\|_2^2$$

for a particular choice of $\boldsymbol{W}$.

(b) Derive a system of equations that the $\boldsymbol{x}$ that minimizes (3) must satisfy by taking the gradient with respect to $\boldsymbol{x}$ and setting this to zero (see page 14 of the notes for the analogous derivation in the unweighted case).

(c) A careful review of the data in problem 2(b) leads you to believe that noise in your measurement process led to a high level of noise on every other sample. Compute a weighted least squares estimate using the same data, but now using weights given by:

$$w_m = \begin{cases} 1 & \text{if } m \text{ is odd} \\ 100 & \text{if } m \text{ is even.} \end{cases}$$

Plot the results.

4. Suppose that we are given a set of points $(x_1, y_1), \ldots, (x_M, y_M)$ and wish to interpolate between these points using a polynomial of degree $N-1$, i.e., we would like to find a function of the form

$$p(x) = a_{N-1}x^{N-1} + \cdots + a_1 x + a_0,$$

such that $p(x_m) = y_m$ for $m = 1, \ldots, M$.

(a) The problem of finding such an interpolating polynomial can be cast as finding the solution to a system of equations of the form $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{a}$, where $\boldsymbol{y}$ is an $M \times 1$ matrix and $\boldsymbol{X}$ is an $M \times N$ matrix, both of which are determined by our observations, and $\boldsymbol{a}$ is an $N \times 1$ vector that represents the parameters of the polynomial. Write down what the entries of $\boldsymbol{y}$, $\boldsymbol{X}$ and $\boldsymbol{a}$ are.

(b) It is a fact that if you have M samples of a function, you can always perfectly interpolate those M samples using a polynomial of degree $M-1$ (i.e., a polynomial with M parameters). Given a function $f(x)$, write a Python script that will

- Form a dataset of size M by computing M equally spaced samples $x_1, \ldots, x_M$ in the interval $[-1, 1]$ and setting $y_m = f(x_m)$.
- Form the system $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{a}$.
- Solve for $\boldsymbol{a}$ by inverting $\boldsymbol{X}$.
- Plot the original $f(x)$ along with your interpolating polynomial.

Test your code on the function $f(x) = 1 - x^2$ (verifying that with $M \geq 3$ your interpolation identifies $f(x)$ perfectly.)

(c) Now test your code on the function

$$f(x) = \frac{1}{1 + 25x^2}.$$

Note that this is *not* a polynomial, but it is smooth and can be well-approximated by a polynomial. One way to find a polynomial that might be a good approximation would be to sample the function and then interpolate. Do this for $M = 3, 5, 7, \ldots, 21$. (Since this is an even function, we expect the polynomial to only have non-zero coefficients for terms involving the even powers of x.) Decide which value of M yields the best result.[1] Submit plots showing $f(x)$ and the interpolating polynomials for both the best choice of M as well as for $M = 21$.

(d) Up to now we have focused only on the interpolation problem, in which we use M samples to fit a polynomial of degree $M-1$. Now suppose we wish to fit a polynomial of degree $N-1$ where $N < M$. Argue that we can cast this as a least squares optimization problem – write down both the optimization problem and an analytical formula for the solution. Modify your code from part (b) to handle the general case where $N \leq M$ (you should only need to change one line) and use this to find a polynomial of degree 20 that approximates the function from part (c) by setting $M > 20$. You should experiment with M and can set it as large as you like. Comment on your results and compare to what you obtained in part (c).

[1] To do this automatically, I would use some kind of numerical integration here to estimate $\int_{-1}^{1}(f(x) - p_M(x))^2 dx$ where $p_M(x)$ is the interpolating polynomial of degree $M-1$. This can be a very simple method. Recall the Riemann approximation of an integral.