Programming Assignment 3: Hilbert Projection in Practice

Due October 20, 2022

The purpose of this assignment is to concretize Hilbert Projection. You will use data to exactly reconstruct polynomials (of low degree) as well as use Hilbert Projection to estimate data with polynomials, and compare to Taylor approximations of various degress. In this assignment we are in the regression setting $f : \mathcal{X} \to \mathcal{Y}$ where both $\mathcal{X} = \mathcal{Y} = \mathbb{R}$.

1 Pointwise Exact Reconstruction

Let $S := \{(x_0, y_0), \ldots, (x_m, y_m)\} \subset \mathcal{X} \times \mathcal{Y} = \mathbb{R}^2$ and suppose you would like to fit polynomial $\hat{y}_m(t)$ of degree m exactly to this data, i.e. so that $p_m(x_i) = y_i$ for $i = 0, \ldots, m$, i.e. define

$$\hat{y}_m(t) := \sum_{i=0}^{m} \prod_{j \neq i} \left(\frac{t - x_j}{x_i - x_j} \right) y_i. \tag{1}$$

(You should amend this definition for when $t \in \{x_0, \ldots, x_m\}$ and convince yourself that $\hat{y}_m(x_i) = y_i$.)

Implement a method which, given finite labeled data set $S \subset \mathcal{X} \times \mathcal{Y}$, and input $t \in \mathbb{R}$, returns $\hat{y}_m(t)$ as in (1). Implement another method which returns the coefficients of this polynomial $\hat{y}_m$. You will need to solve the system

$$\begin{pmatrix} t_0^0 & \cdots & t_0^m \\ \vdots & \ddots & \vdots \\ t_m^0 & \cdots & t_m^m \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ \vdots \\ v_m \end{pmatrix} = \begin{pmatrix} \hat{y}_m(t_0) \\ \vdots \\ \hat{y}_m(t_m) \end{pmatrix} \tag{2}$$

for distinct values $t_0, \ldots, t_m$ (c.f. Vandermonde matrix) Since you'll need to solve matrix equations anyway, it may be worthwhile to implement a generic method.

2 Estimation (Regression)

In the previous worksheet, you worked out the steps, using Hilbert Projection, for finding optimal polynomial approximation

$$y_m^* \in \mathcal{H}_m := \left\{ \sum_{j=0}^{m} a_j t^j : a_j \in \mathbb{R} \right\}$$

of $\pi_{\mathcal{Y}} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ mapping $(x, y) \mapsto y$, and how, given data S, to approximate y_m^*. Write methods which, given data S, find quasi-optimal coefficients (we may label them as) $a_0^*, \ldots, a_k^*$ for some degree $k < m$. (When $m \leq k$, you may note that estimation (should) returns the same result as exact reconstruction. Why might it not?)

3 Best Estimate: Taylor Polynomial

Recall from calculus that $T_{f,n}(t) := \sum_{j=0}^{n} \frac{f^{(j)}(0)}{j!} t^j$ is the degree-n estimate of n-differentiable function $f : \mathbb{R} \to \mathbb{R}$ about zero with error term $o(x^n)$. Write a method which returns coefficients of Taylor-about-zero function f given specified degree n. (This method will not depend on data.)

Because you will be feeding in various coefficients and numpy arrays of input data, you may want to write a polynomial method which takes input t (as single value or array) and coefficients array c and returns the degree n polynomial (values) defined by coefficients (note that degree is specified by length of c).

4 Deliverables

Submit code that approximates $f = \sin(x)$ about 0 using data $x \sim \mathcal{N}(0, \sigma^2)$. Your code should reliably accomplish the tasks specified above. Generage independent input data x as well as output data $y = \sin(x) + \eta$ for white noise $\eta \sim \mathcal{N}(0, \tau^2)$. Output a plot illustrating clean data (x, y), as well as exact and regression (Hilbert) estimate on both clean and noisy data, as illustrated in figure below, for some reasonably nontrivial degree (say between 5 and 10). Also run an experiment over polynomial degree, up to deg $= 20$ and plot bias $\mathbb{E}((y_{\mathcal{H}}^* - y)^2)$ and variance $\mathbb{E}((y_{\mathcal{H}}^* - \tilde{y})^2)$ against m: you will see bias decrease and variance (eventually) increase. Notice that bias is lower bounded by some function of τ, can you explain why?

Submit a plot of bias/variance/total as a function of degree. In principle, the bias plot is not supposed to depend on data, but there is a problem: you are not analytically computing the various moments $\mathbb{E}(x^k)$ or $\mathbb{E}(x^k y)$. Therefore, you may "pretend" that Hilbert Projection, using a very large data set (say $m > 2e^6$) on clean data is for all intents and purposes a "good enough"

approximation of the optimal-in-$\mathcal{H}_m$. If this feels awkward, it should; you're working with what you have. On small degrees, you'll notice an error in bias term decrease and settle (stabilize) around some value related to τ^2. The variance term, by contrast, should start to grow (not necessarily monotonically) with degree. You should use the total loss (sum of bias and variance terms) to determine an optimal degree for estimation, and report it.

Collecting exact requirements, submit two plots with your code:

1. a plot of various the functional approximations—including Taylor approximation, exact reconstruction (on both clean and noisy data) of specified degree selecting said degree number of data points, and the Hilbert Projection estimation of same degree, also on clean and noisy data, and
2. a plot of total loss as a function of degree, accompanied with bias and variance plots (so three curves overlayed) for your Hilbert estimation on noisy data, with an indication on plot of optimal (w.r.t. total mse) degree.