Polynomial Approximation in L_2

A. Doner, Spring 2021

So far we talked about interpolation as a way to approximate a non polynomial function $f(x)$ on some interval $[a, b]$. The advantages are:

1) All we needed were values of the function at the $(n+1)$ interpolation nodes, we didn't even need to know the function $f(x)$ ["black box" mode]

2) Easy to evaluate interpolant using barycentric interpolation.

The main disadvantages are:

1) The choice of nodes really matters and equi-spaced is bad

2) Even if $p_n(x_k) = f(x_k)$ at the nodes, this does not guarantee anything about $|f(x) - p(x)|$ for x in-between the nodes (recall Runge function)

We improved on these issues by using piecewise polynomial interpolation like splines, but that was not very accurate

so to get 16 digits in $f(x)$ we would need thousands of nodes/points.

Is there another way to approximate $f(x)$ by $p_n(x)$ on $[a,b]$?

Yes!

$$p_n^* = \arg\min_{p_n \in \mathcal{P}^n} \|f - p_n\|_2$$

"Least squares" polynomial approx.

This is just like doing least squares fitting to data, but now we use the function L_2 norm not vector one.

E.g. Approximate $f(x) = -2x^2$ with a constant function ($n=0$!) on $[-1,1]$:

$$\|f - p_0\|_2^2 = \int_{-1}^{1} (-2x^2 - a_0)^2 \, dx$$

$$= 2a_0^2 + \frac{8a_0}{3} + \frac{8}{5}$$

$$\boxed{p_0(x) = a_0}$$

$$a_0^* = \arg\min_{a_0} \left(2a_0^2 + \frac{8a_0}{3} + \frac{8}{5}\right)$$

$$\frac{d}{da_0}\left(2a_0^2 + \frac{8a_0}{3} + \frac{8}{5}\right)\Big|_{a_0 = a^*} = 0 \Rightarrow$$

$$a_0^* = -\frac{2}{3} = p_0^*(x)$$

Better & more instructive is to do this for a general $f(x)$ [OFTEN SIMPLER!!!] ④

$$a_0^* = \arg\min \int (f(x) - a_0)^2 dx$$

$$= \arg\min \int f^2(x)\,dx - 2a_0 \int f(x)\,dx + \int a_0^2\,dx$$

Differentiate w.r.t a_0:

$$2a_0^* \int dx - 2\int f(x)\,dx = 0$$

$$\Rightarrow a_0^* = \frac{\int f(x)\,dx}{\int dx} = \frac{\int f(x)\,dx}{b-a}$$

which is simply the mean (average) of $f(x)$ over $[a, b]$.

If $f(x) = -2x^2$, mean $= \frac{-2\int_{-1}^{1} x^2\,dx}{2} = -\frac{2}{3}$

Let's now repeat this for an arbitrary degree polynomial.

First, choose a basis for $\mathcal{P}_n$:

$$\text{basis} = \{ p_0(x), p_1(x), p_2(x), \ldots, p_n(x) \}$$

$$p_n^*(x) = \sum_{j=0}^{n} a_j p_j(x) = \text{best } L_2 \text{ approximation}$$

How do we find the $(n+1)$ coefficients $\vec{a}$? (drop $*$)

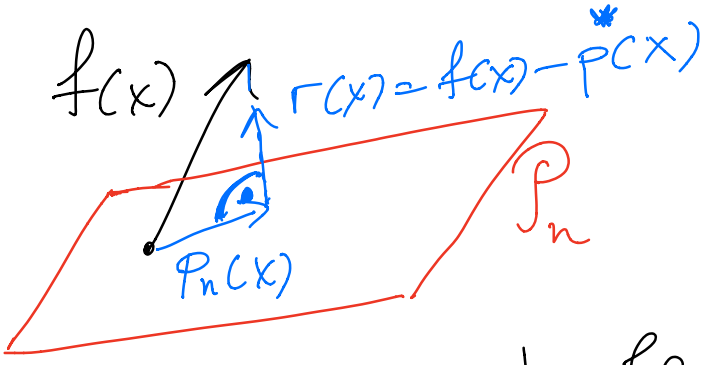
Standard approach in textbooks:

$$F(\vec{a}) = \int_a^b \left(f(x) - \sum c_j p_j(x) \right)^2 dx$$

$$\frac{\partial F}{\partial a_k} = 0, \quad k = 0, \ldots, n+1$$

$\uparrow (n+1)$ equations

Easy to do by expanding the square. But let's instead follow a more linear algebra "geometric" approach:



$r(x) \perp p_n$

Residual must be orthogonal to all polynomials of degree n, i.e., it must be orthogonal to $p_k(x)$, $k = 0, \dots, n$!

$$(r, p_k) = 0 \quad \forall k$$

$$(f - p_n^*, p_k) = 0 \quad \forall k$$

$\Rightarrow$

$$(f - p_n^*, p_k) = 0 \quad \forall k$$

$$\left(f - \sum_{j=0}^{n} a_j p_j, p_k\right) = 0$$

$$\left(\sum_{j=0}^{n} a_j p_j, p_k\right) = (f, p_k)$$

(properties of inner product)

$$\sum_{j=0}^{n} a_j (p_j, p_k) = (f, p_k)$$

$$k = 0, \dots, n+1$$

$n+1$ equations for $n+1$ unknowns

In matrix form

$$\overleftrightarrow{V}\,\vec{a} = \vec{f}$$

$$V_{ij} = (p_i, p_j) = \int_a^b p_i(x)\,p_j(x)\,dx$$

(A "Vandermonde" matrix)

$$f_k = (f, p_k) = \int_a^b f(x)\,p_k(x)\,dx$$

Here we assumed a real-valued function.

Once again, like for interpolation, it boils down to solving a linear system!

Expensive? Ill-conditioned?

Let's take monomials as basis:

$$p_k(x) = x^k \quad \text{on } [0,1)$$

$$V_{ij} = \int_0^1 x^i x^j \, dx = \frac{x^{i+j+1}}{i+j+1}\Big|_0^1$$

$$V_{ij} = \frac{1}{i+j+1} \leftarrow \text{Hilbert matrix}$$

You learned in worksheets that the Hilbert matrix is very ill conditioned, just like the Vandermonde matrix.

So using a monomial basis is not a good idea.

Instead, we need something akin to Lagrange polynomials for interpolation but for L_2 approximation.

What this means is that we want to choose basis such that $V_{ij} = \delta_{ij}$, i.e., V is the identity matrix, because then we have

$$a_k = (f, p_k) = \int f(x) p_k(x) dx$$

if $$V_{ij} = \int p_i(x) p_j(x) dx = \delta_{ij}$$

$\Rightarrow$ (!!) $\int p_i(x) p_j(x) dx = 0$ if $i \neq j$ or 1 if $i = j$

This means that we want to use as basis a set of $(n+1)$ orthogonal polynomials.

How do we find an orthonormal basis for $\mathcal{P}_n$?

I.E. how do we find an orthonormal basis for the space spanned by $\{x^0, x^1, \ldots, x^n\}$?

In the case of vectors, we did this using QR factorization, which was really simply doing the Gram-Schmidt orthogonalization process.