Polynomial Approximation

in L_2

A. Donev, 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 <u>piecewise</u> 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 <u>constant</u> function ($n=0$!) on $[-1,1]$:

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

$$\boxed{p_0(x) = a_0} \quad = 2a_0^2 + \frac{8a_0}{3} + \frac{8}{5}$$

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

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

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

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

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

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

Differentiate w.r.t a_0:

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

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

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

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

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

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

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

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

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

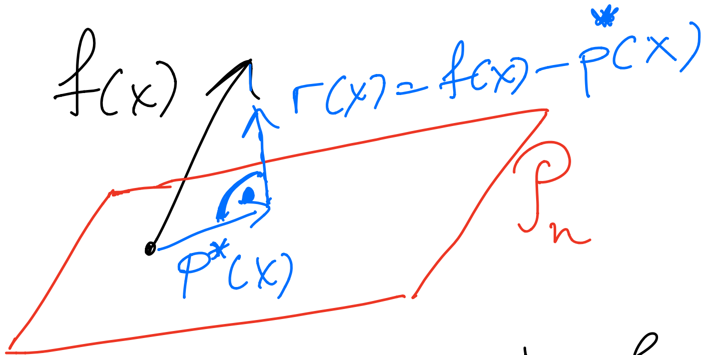
Standard approach in textbooks:

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

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

$(n+1)$ 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 $P_k(x)$, $k = 0, \dots, n$

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

$$(f - p^*, P_k) = 0 \quad \forall k$$

$\Rightarrow$

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

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

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

(properties of inner product)

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

$$k = 0, \ldots, n$$

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 <u>monomials</u> as basis :

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

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

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

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

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

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.

Let's illustrate this on a very important example:

Find the first 3 orthogonal polynomials on $[-1, 1]$ in the standard L_2 inner product

$$(f, g)_2 = \int_{-1}^{1} f(x)\, g(x)\, dx$$

Assume real-valued functions

Start with the 3 functions

$$\{1, x, x^2\}$$

(think of a matrix with 3 columns) and then find an orthogonal basis for $\mathcal{P}_2$ (think of QR factorization of the matrix)

Gram- Schmidt process

$p_0(x) = 1$ $\quad p_2(x) = x^2$

$p_1(x) = x$

We want $\varphi_0(x), \varphi_1(x), \varphi_2(x)$ that are orthogonal to each other

$\varphi_0(x) = p_0(x) = 1$

$$\varphi_1 = p_1 - \mathrm{Proj}_{\{\varphi_0\}} p_1$$

$$= p_1 - \frac{(\varphi_0, p_1)}{(\varphi_0, \varphi_0)} \varphi_0 =$$

$$= x - \frac{\int_{-1}^{1} x \cdot 1 \cdot dx}{\int_{-1}^{1} 1 \cdot 1 \cdot dx} \cdot 1 = x = p_1$$

Since p_1 and p_0 are already orthogonal on $[-1, 1]$.

Now

$$\varphi_2 = p_2 - \mathrm{Proj}_{\{\varphi_0, \varphi_1\}} p_2 =$$

$$= p_2 - \frac{(\varphi_0, p_2)}{(\varphi_0, \varphi_0)} \varphi_0 - \frac{(\varphi_1, p_2)}{(\varphi_1, \varphi_1)} \varphi_1$$

$$= x^2 - \frac{\int_{-1}^{1} x^2\, dx}{\int_{-1}^{1} 1 \cdot dx} - \frac{\int_{-1}^{1} x^2 \cdot x \cdot dx}{\int_{-1}^{1} x^2\, dx} \cdot x$$

(but $\int_{-1}^{1} x^2\, dx = \frac{x^3}{3}\Big|_{-1}^{1} = \frac{2}{3}$, $\int_{-1}^{1} dx = 2$

$\int_{-1}^{1} x^3\, dx = \frac{x^4}{4}\Big|_{-1}^{1} = 0$

$$= x^2 - \frac{1}{3} - 0 = x^2 - 1/3$$

So we obtain that

$$\varphi_0(x) = 1$$

$$\varphi_1(x) = x$$

$$\varphi_2(x) = x^2 - \frac{1}{3}$$

$$\varphi_n(x) = x^n - \sum_{j=0}^{n-1} \left(\frac{\int_{-1}^{1} x^n \varphi_j(x)\,dx}{\int \varphi_j^2(x)\,dx} \right) \varphi_j(x)$$

are orthogonal polynomials in L_2 on $[-1, 1]$

They are known as Legendre polynomials and are very important for polynomial approximation

It is easy to see by change of coordinate

$$t = \left(\frac{x+1}{2}\right)(b-a) + a \in [a,b]$$

that if $j \neq k$

$$0 = \int_a^b \varphi_j(t)\,\varphi_k(t)\,dt = \int_{-1}^{1} \varphi_j(x)\,\varphi_k(x)\,dx$$

which means $\{\varphi_k(t)\}$ are orthogonal on $[a,b]$ in L_2.

So we can easily generalize to another interval $[a,b]$, though $[-1,1]$ is standard.

Now go back to our goal of approximating functions:

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

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

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

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

Now we have polynomials s.t.

$$V_{ij} = \int_{-1}^{1} \varphi_i(x) \varphi_j(x) \, dx = 0 \quad \text{if } i \neq j$$

(orthogonal Legendre polynomials)

They are not ortho<u>normal</u>

because we did not normalize them, i.e., in general

$$\int \varphi_j^2(x)\,dx \neq 1$$

(we <u>could have</u> normalized them!)

So now we have

$$f(x) \simeq P_n^*(x) = \sum_{k=0}^{n} a_k \varphi_k(x)$$ (best L_2 approx. of f in $\mathcal{P}_n$)

$$a_k = \frac{(f, \varphi_k)}{(\varphi_k, \varphi_k)}$$ ← <u>explicit</u> formula by integral

This works for <u>any</u> choice of the inner product, interval $[a, b]$ — what changes are the $\varphi_k(x)$.

E.g. Find the best quadratic approximation to $\sin(x)$ on $[0,\pi]$ in standard L_2 norm.

Solution: $t = \frac{x+1}{2}\pi \Rightarrow x = \frac{2t}{\pi} - 1$

Demo code : ApproxL2.m

$$\varphi_0(t) = 1$$

$$\varphi_1(t) = x = \frac{2t}{\pi} - 1$$

$$\varphi_2(t) = x^2 - \frac{1}{3} = \left(\frac{2t}{\pi} - 1\right)^2 - \frac{1}{3}$$

$$= \frac{4}{\pi}\left(\frac{t^2}{\pi} - t + \frac{\pi}{6}\right)$$

$$a_k = \int_0^{\pi} \sin(t)\,\varphi_k(t)\,dt \Big/ \int \varphi_k^2(t)\,dt$$

$k = 0, 1, 2$

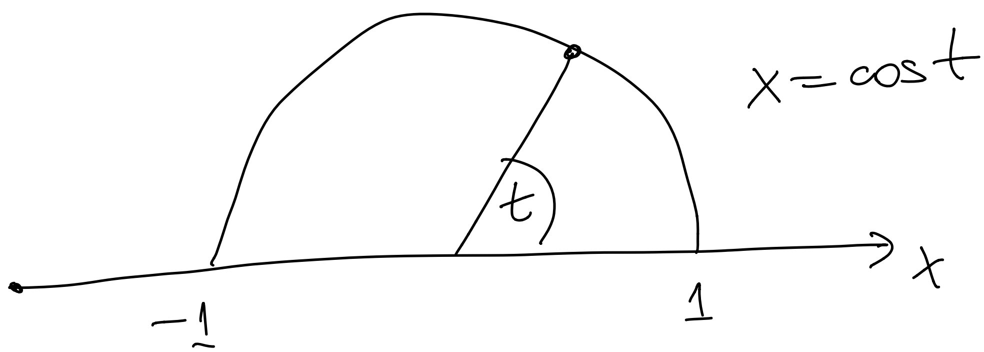
Homework: Compute $p_2^*(x)$ and plot it & compare to worksheet 7

Chebyshev polynomials

It is not hard to show that

$$\int_0^{\pi} \cos(mt)\cos(nt)\,dt = 0$$

So $\cos(kt)$ are orthogonal functions (called trigonometric polynomials) on $(0, \pi)$



$$t = a\cos x \Rightarrow$$

$$dt = \frac{1}{\sqrt{1-x^2}}\,dx$$

$$\Rightarrow \int_{-1}^{1} \cos(m \cdot \operatorname{acos}(x)) \cdot \cos(n \cdot \operatorname{acos}(x)) \cdot \frac{dx}{\sqrt{1-x^2}} = 0 \quad \ldots\ldots (*)$$

Define

$$T_k(x) = \cos(k \cdot \operatorname{acos}(x))$$

$$k = 0, 1, 2, \ldots$$

It turns out that $T_k(x)$ is actually a <u>polynomial</u> of x of degree k!

The Chebyshev polynomials $T_k(x)$ are orthogonal on $[-1, 1]$ with respect to the <u>weighted</u> inner product

$$(f, g) = \int_{-1}^{1} f g \frac{dx}{\sqrt{1-x^2}}$$

Homework: Use Gram-Schmidt to obtain explicit formulas for

$$\left[\begin{array}{l} T_0(x) = 1 \\ T_1(x) = x \\ T_2(x) = 2x^2 - 1 \end{array}\right.$$

$$T_3(x) = 4x^3 - 3x$$

$$\dots$$

One can use <u>either</u> Legendre <u>or</u> Chebyshev polynomials, they are similar & which one is used depends on the context & choice by the numerical analyst.

Orthogonal polynomials & computation

We saw that to compute orthogonal polynomials or L_2 function approximants, we need to be able to numerically compute integrals – we will study this next & see that orthogonal polynomials will make a re-appearance!

In practice, we rarely use optimal L_2 approximation because it is hard to compute.

But, it turns out that orthogonal polynomials are very useful for polynomial interpolation, which is easy to do numerically.

The following two properties/theorems are key:

① The zeros of the orthogonal polynomial $\varphi_{j\geq 1}$ are real & distinct and in (a,b)

(see Theorem 9.4 in theory textbook for proof)

2 The roots of $\varphi_j(x)$ in (a,b) can be used as nodes for polynomial interpolation that is accurate & stable, i.e., does not suffer from Runge's phenomenon. This is because the nodal polynomial is "well-behaved."

This is the key lesson. In practice, we use the roots of orthogonal polynomials as interpolation nodes for global polynomial interpolation.

While the resulting interpolating polynomial is <u>not</u> the best L_2 approximant, in practice it is just as good & easy to compute.

Take the Chebyshev polynomials of <u>first</u> kind (difference is basically cos vs. sine)

$$T_k(x) = \cos(k \cdot \operatorname{acos}(x))$$

Let's find its roots:

$$T_k(x) = 0 \implies$$

$$k \cdot \operatorname{acos}(x) = (2n+1)\frac{\pi}{2}$$

$$n \in \mathbb{Z}$$

$$\Rightarrow \quad t = \operatorname{acos}(x) = \frac{(2n+1)}{k}\frac{\pi}{2}$$

$$t \in [0, \pi]$$

$$x \in [-1, 1]$$

$$0 \le \frac{2n+1}{k} \le 2$$

$$\Rightarrow \qquad 0 \le n \le \frac{(2k-1)}{2}$$

$$\Rightarrow \begin{cases} x_n = \cos\left(\frac{n+1/2}{k}\pi\right) \\ 0 \le n < k \end{cases}$$

which are called the Chebyshev nodes of the first kind (second kind are $x_n = \cos\left(\frac{n}{k+1}\pi\right)$) and you used them in Worksheet 6 for interpolation