

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]

①

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

(2)

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.

(3)

E.g. Approximate $f(x) = -2x^2$
with a constant function ($n=0!$)

on $[-1, 1]$:

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

$$\boxed{P_0(x) = a_0} = 2a_0^2 + \frac{8a_0}{3} + \frac{8}{5}$$

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

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

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

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

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

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


 $+ \int a_0^2 dx$

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 mean
 (average) of $f(x)$ over $[a, b]$.

$$\text{If } f(x) = -2x^2, \text{ mean} = \frac{-2 \int_{-1}^1 x^2 dx}{2} = -\frac{2}{3} \quad (5)$$

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

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

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

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

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

Standard approach in textbooks:

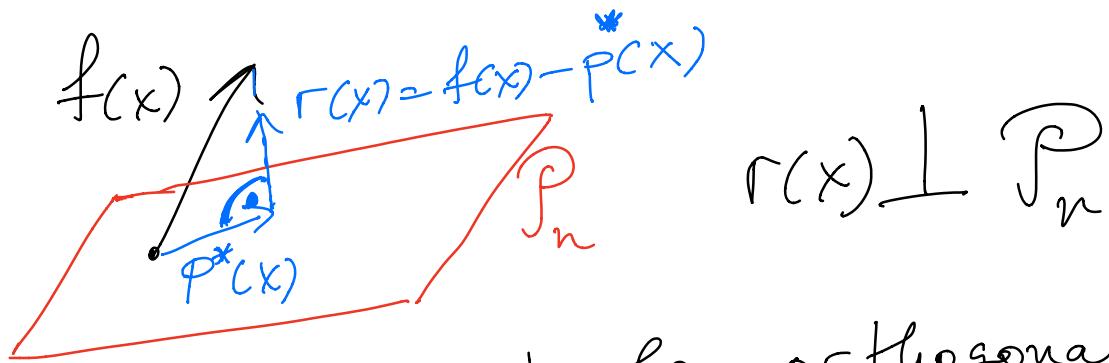
$$F(\vec{a}) = \int_a^b \left(f(x) - \sum c_j p_j(x) \right)^2 dx$$

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

$\nwarrow_{(n+1) \text{ equations}}$

6

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

(7)

$$(f - p^*, \varphi_k) = 0$$

$$(f - \sum_{j=0}^n a_j \varphi_j, \varphi_k) = 0$$

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

 (properties of inner product)

$$\sum_{j=0}^n a_j (\varphi_j, \varphi_k) = (f, \varphi_k)$$

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

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

⑧

In matrix form

$$\begin{matrix} \leftarrow \\ V \end{matrix} \vec{a} = \vec{f}$$

$$V_{ij} = (P_i, P_j) = \int_a^b P_i(x) P_j(x) dx$$

(A "Vandermonde" matrix)

$$\vec{f}_k = (f, P_k) = \int_a^b f(x) P_k(x) dx$$

Here we assumed a real-valued function.

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

Expensive? Ill-conditioned?

⑨

Let's take monomials as

basis :

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

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

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

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.

(10)

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.,

\vec{V} is the identity matrix, because then we have

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

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

$$\Rightarrow \int p_i(x) p_j(x) dx = 0$$

(11) 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 P_n ?

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

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

(12)

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 P_2 (think of QR factorization of the matrix)

(13)

Gram-Schmidt process

$$P_0(x) = 1$$

$$P_2(x) = x^2$$

$$P_1(x) = x$$

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

$$\varphi_0(x) = P_0(x) = 1$$

$$\varphi_1 = P_1 - \text{Proj}_{\{\varphi_0\}} P_1$$

$$= P_1 - \frac{(\varphi_0, P_1)}{(\varphi_0, \varphi_0)} \varphi_0 =$$

$$= x - \frac{\int_{-1}^1 x \cdot 1 \cdot dx}{\int_{-1}^1 1 \cdot 1 \cdot dx} \cdot 1 = x = P_1$$

(19)

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

Now

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

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

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

(But  $\int_{-1}^1 x^2 dx = \frac{x^3}{3} \Big|_{-1}^1 = \frac{2}{3}$ ,  $\int_{-1}^1 1 dx = 2$   
 $\int_{-1}^1 x^3 dx = \frac{x^4}{4} \Big|_{-1}^1 = 0$ )

$$= x^2 - \frac{1}{3} - 0 = x^2 - \frac{1}{3}$$
(15)

So we obtain that

$$\varphi_0(x) = 1$$

$$\varphi_1(x) = x$$

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

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

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:

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

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

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

$$\text{if } V_{ij} = \int P_i(x) P_j(x) dx = \delta_{ij}$$

Now we have polynomials s.t.

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

(orthogonal Legendre polynomials)

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

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

(we could have normalized them!)

So now we have

$$f(x) \approx P_n^*(x) = \sum_{k=0}^n a_k x^k \quad (\text{best } L_2 \text{ approx.})$$

of  $f$  in  $P_n$

$$a_k = \frac{(f, \varphi_k)}{(\varphi_k, \varphi_k)} \leftarrow \begin{array}{l} \text{explicit} \\ \text{formula} \\ \text{by integral} \end{array}$$

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

(19)

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

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

$$\varphi_0(t) = 1$$

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

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

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

$$a_k = \frac{1}{\int_0^\pi \varphi_k^2(t) dt} \int_0^\pi \sin(t) \varphi_k(t) dt$$

$$k = 0, 1, 2$$

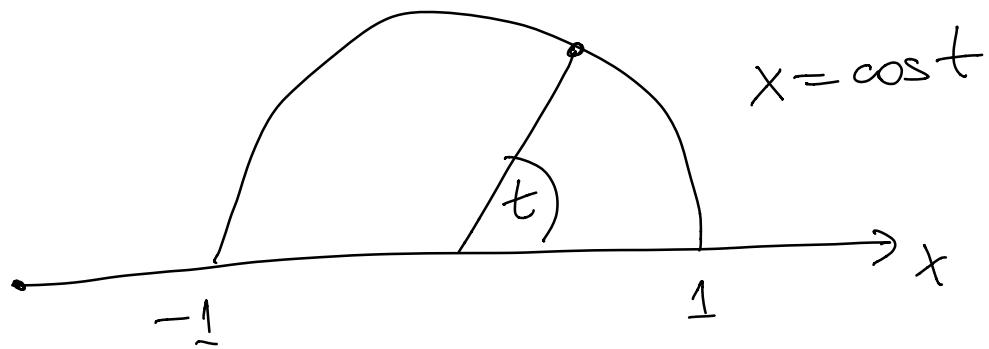
Homework: Compute  $\varphi_2^*(x)$  and plot it & compare to worksheet 7

## Chebyshev polynomials

It is not hard to show that

$$\int_0^{\pi} \cos(m t) \cos(n t) dt = 0$$

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



$$t = a \cos X \Rightarrow$$

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

$$\Rightarrow \int_{-1}^1 \cos(m \cdot \cos(x)) \cdot \cos(n \cdot \cos(x)) \cdot \frac{dx}{\sqrt{1-x^2}} = 0 \quad \dots \quad (\ast)$$

Define

$$T_k(x) = \cos(k \cdot \cos(x))$$

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

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

The Chebyshev polynomials  $T_k(x)$  are orthogonal on  $[-1, 1]$  with respect to the weighted

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

(22)

Homework: Use Gram-Schmidt  
to obtain explicit formulas  
for

$$\begin{cases} T_0(x) = 1 \\ T_1(x) = x \\ T_2(x) = 2x^2 - 1 \end{cases}$$

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

...

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

(23)

## 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 orthonormal 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 text book 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 not the best  $L_2$  approximant, in practice it is just as good & easy to compute.

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

$$T_k(x) = \cos(k \cdot \alpha \cos(x))$$

Let's find its roots :

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

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

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

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

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

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

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

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

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

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