Numerical Analysis and Computing

Lecture Notes #11
— Approximation Theory —
Least Squares Approximation & Orthogonal Polynomials

Joe Mahaffy,
⟨mahaffy@math.sdsu.edu⟩

Department of Mathematics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://www-rohan.sdsu.edu/~jmahaffy

Spring 2010

Outline

Picking Up Where We Left Off... Discrete Least Squares, I

The Idea: Given the data set $(\tilde{\mathbf{x}}, \tilde{\mathbf{f}})$, where $\tilde{\mathbf{x}} = \{x_0, x_1, \ldots, x_n\}^T$ and $\tilde{\mathbf{f}} = \{f_0, f_1, \ldots, f_n\}^T$ we want to fit **a simple model** (usually a low degree polynomial, $p_m(x)$) to this data.

We seek the polynomial, of degree m, which minimizes the residual:

$$r(\tilde{\mathbf{x}}) = \sum_{i=0}^{n} [p_m(x_i) - f(x_i)]^2 .$$

Picking Up Where We Left Off... Discrete Least Squares, II

We find the polynomial by differentiating the sum with respect to the **coefficients** of $p_m(x)$. — If we are fitting a fourth degree polynomial $p_4(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4$, we must compute the partial derivatives wrt. a_0, a_1, a_2, a_3, a_4.

In order to achieve a minimum, we must set all these partial derivatives to zero. — In this case we get 5 equations, for the 5 unknowns; the system is known as the **normal equations**.

The Normal Equations — Second Derivation

Last time we showed that the normal equations can be found with purely a Linear Algebra argument. Given the data points, and the model (here $p_4(x)$), we write down the over-determined system:

$$\left\{\begin{array}{rcl} a_0 + a_1x_0 + a_2x_0^2 + a_3x_0^3 + a_4x_0^4 & = & f_0 \\ a_0 + a_1x_1 + a_2x_1^2 + a_3x_1^3 + a_4x_1^4 & = & f_1 \\ a_0 + a_1x_2 + a_2x_2^2 + a_3x_2^3 + a_4x_2^4 & = & f_2 \\ & \vdots & \\ a_0 + a_1x_n + a_2x_n^2 + a_3x_n^3 + a_4x_n^4 & = & f_n. \end{array}\right.$$

We can write this as a matrix-vector problem:

$$X\tilde{\mathbf{a}} = \tilde{\mathbf{f}},$$

where the **Vandermonde matrix** X is tall and skinny. By multiplying both the left- and right-hand-sides by X^T (the transpose of X), we get a "square" system — we recover the **normal equations:**

$$X^TX\tilde{\mathbf{a}} = X^T\tilde{\mathbf{f}}.$$

Discrete Least Squares: A Simple, Powerful Method.

Given the data set $(\tilde{\mathbf{x}}, \tilde{\mathbf{f}})$, where $\tilde{\mathbf{x}} = \{x_0, x_1, \ldots, x_n\}$ and $\tilde{\mathbf{f}} = \{f_0, f_1, \ldots, f_n\}$, we can quickly find the best polynomial fit for **any** specified polynomial degree!

Notation: Let $\tilde{\mathbf{x}}^j$ be the vector $\{x_0^j, x_1^j, \ldots, x_n^j\}$.

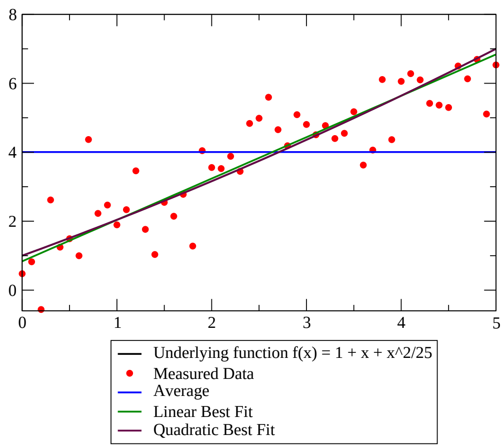
E.g. to compute the best fitting polynomial of degree 4, $p_4(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4$, define:

$$X = \begin{bmatrix} | & | & | & | & | \\ \tilde{\mathbf{1}} & \tilde{\mathbf{x}} & \tilde{\mathbf{x}}^2 & \tilde{\mathbf{x}}^3 & \tilde{\mathbf{x}}^4 \\ | & | & | & | & | \end{bmatrix}, \quad \text{and compute} \quad \tilde{\mathbf{a}} = \underbrace{(X^TX)^{-1}(X^T\tilde{\mathbf{f}})}_{\text{Not like this! See math 543!}}.$$

Example: Fitting $p_i(x)$, $i = 0, 1, 2, 3, 4$ Models.

Figure: We revisit the example from last time; and fit polynomials up to degree four to the given data. The figure shows the best $p_0(x)$, $p_1(x)$, and $p_2(x)$ fits.

Below: the errors give us clues when to stop.



Model	Sum-of-squares-error
$p_0(x)$	205.45
$p_1(x)$	52.38
$p_2(x)$	51.79
$p_3(x)$	51.79
$p_4(x)$	51.79

Table: Clearly in this example there is very little to gain in terms of the least-squares-error by going beyond 1st or 2nd degree models.

Introduction: Defining the Problem.

Up until now: **Discrete Least Squares Approximation** applied to a collection of data.

Now: **Least Squares Approximation of Functions.**

We consider problems of this type: —

Suppose $f \in C[a, b]$ and we have the class $\mathcal{P}_n$ which is the set of all polynomials of degree at most n. Find the $p(x) \in \mathcal{P}_n$ which minimizes

$$\int_a^b [p(x) - f(x)]^2 \, dx.$$

Finding the Normal Equations...

If $p(x) \in \mathcal{P}_n$ we write $p(x) = \sum_{k=0}^{n} a_k x^k$. The sum-of-squares-error, as function of the coefficients, $\tilde{\mathbf{a}} = \{a_0, a_1, \ldots, a_n\}$ is

$$E(\tilde{\mathbf{a}}) = \int_a^b \left[\sum_{k=0}^{n} a_k x^k - f(x) \right]^2 \, dx.$$

Differentiating with respect to a_j ($j = \{0, 1, \ldots, n\}$) gives

$$\frac{\partial E(\tilde{\mathbf{a}})}{\partial a_j} = 2 \sum_{k=0}^{n} a_k \int_a^b x^{j+k} \, dx - 2 \int_a^b x^j f(x) \, dx.$$

At the minimum, we require $\frac{\partial E(\tilde{\mathbf{a}})}{\partial a_j} = 0$, which gives us a system of equations for the coefficients a_k, **the normal equations.**

The Normal Equations.

The $(n+1)$-by-$(n+1)$ system of equations is:

$$\sum_{k=0}^{n} a_k \int_a^b x^{j+k} \, dx = \int_a^b x^j f(x) \, dx, \quad j = 0, 1, \ldots, n.$$

Some **notation**, let:

$$\langle f(x), \, g(x) \rangle = \int_a^b f(x) g(x)^* \, dx,$$

where $g(x)^*$ is the complex conjugate of $g(x)$ (everything we do in this class is real, so it has no effect...)

This is known as an **inner product** on the interval $[a, b]$. (But, if you want, you can think of it as a notational shorthand for the integral...)

The Normal Equations: Inner Product Notation, I

In inner product notation, our normal equations:

$$\sum_{k=0}^{n} a_k \int_a^b x^{j+k} \, dx = \int_a^b x^j f(x) \, dx, \quad j = 0, 1, \ldots, n.$$

become:

$$\sum_{k=0}^{n} a_k \langle x^j, \, x^k \rangle = \langle x^j, \, f(x) \rangle, \quad j = 0, 1, \ldots, n.$$

Recall the Discrete Normal Equations:

$$\sum_{k=0}^{n} \left[a_k \sum_{i=0}^{N} x_i^{j+k} \right] = \sum_{i=0}^{N} x_i^j f_i, \quad j = 0, 1, \ldots, n.$$

Hmmm, looks quite similar!

More Notation, Defining the Discrete Inner Product.

If we have two vectors

$$\tilde{\mathbf{v}} = \{v_0, v_1, \ldots, v_N\}$$
$$\tilde{\mathbf{w}} = \{w_0, w_1, \ldots, w_N\},$$

we can define the discrete inner product

$$[v, \, w] = \sum_{i=0}^{N} v_i w_i^*,$$

where, again w_i^* is the complex conjugate of w_i.

Equipped with this notation, we revisit the Normal Equations...

The Normal Equations: Inner Product Notation, II

Discrete Normal Equations in $\sum$ Notation:

$$\sum_{k=0}^{n}\left[a_k \sum_{i=0}^{n} x_i^{j+k}\right] = \sum_{i=0}^{n} x_i^j f_i, \quad j = 0, 1, \ldots, n.$$

Discrete Normal Equations, in Inner Product Notation:

$$\sum_{k=0}^{n} a_k \left[\tilde{\mathbf{x}}^j, \tilde{\mathbf{x}}^k\right] = \left[\tilde{\mathbf{x}}^j, \tilde{\mathbf{f}}\right], \quad j = 0, 1, \ldots, n.$$

Continuous Normal Equations in Inner Product Notation:

$$\sum_{k=0}^{n} a_k \langle x^j, x^k \rangle = \langle x^j, f(x) \rangle, \quad j = 0, 1, \ldots, n.$$

Hey! It's really the same problem!!! The only thing that changed is the inner product — we went from summation to integration!

Normal Equations for the Continuous Problem: Matrices.

The bottom line is that the polynomial $p(x)$ that minimizes

$$\int_{\alpha}^{\beta} [p(x) - f(x)]^2 \, dx$$

is given by the solution of the linear system $X\vec{a} = \vec{b}$, where

$$X_{i,j} = \langle x^i, x^j \rangle, \quad b_i = \langle x^i, f(x) \rangle.$$

We can compute $\langle x^i, x^j \rangle = \dfrac{\beta^{i+j+1} - \alpha^{i+j+1}}{i+j+1}$ explicitly.

A matrix with these entries is known as a **Hilbert Matrix**; — classical examples for demonstrating **how numerical solutions run into difficulties due to propagation of roundoff errors.**

— *We need some new language, and tools!*

The Condition Number of a Matrix

The **condition number** of a matrix is the ratio of the largest eigenvalue and the smallest eigenvalue:

If A is an $n \times n$ matrix, and its eigenvalues are $|\lambda_1| \leq |\lambda_2| \leq \cdots \leq |\lambda_n|$, then the **condition number** is

$$\mathbf{cond(A)} = \frac{|\lambda_n|}{|\lambda_1|}$$

The condition number is one important factor determining the growth of the numerical (roundoff) error in a computation.

We can interpret the condition number as a **separation of scales**.

If we compute with sixteen digits of precision $\epsilon_{\text{mach}} \approx 10^{-16}$, the best we can expect from our computations (even if we do everything right), is accuracy $\sim \mathbf{cond(A)} \cdot \epsilon_{\text{mach}}$.

The Condition Number for Our Example

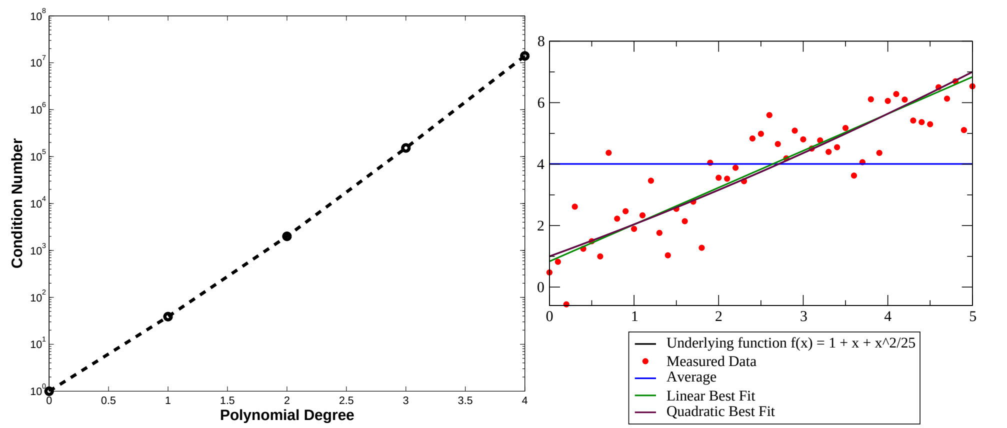


Figure: Ponder, yet again, the example of fitting polynomials to the data (RIGHT). The plot on the left shows the condition numbers for 0th, through 4th degree polynomial problems. Note that for the 5-by-5 system (Hilbert matrix) corresponding to the 4th degree problem the condition number is already $\sim 10^7$.

Linearly Independent Functions.

Definition (Linearly Independent Functions)

The set of functions $\{\Phi_0(x), \Phi_1(x), \ldots, \Phi_n(x)\}$ is said to be **linearly independent** on $[a, b]$ if, whenever

$$\sum_{i=0}^{n} c_i \Phi_i(x) = 0, \quad \forall x \in [a, b],$$

then $c_i = 0$, $\forall i = 0, 1, \ldots, n$. Otherwise the set is said to be **linearly dependent.**

Theorem

If $\Phi_j(x)$ is a polynomial of degree j, then the set $\{\Phi_0(x), \Phi_1(x), \ldots, \Phi_n(x)\}$ is linearly independent on any interval $[a, b]$.

Discrete Least Squares Approximation
Continuous Least Squares Approximation
Orthogonal Polynomials
Linear Independence... Weight Functions... Inner Products
Least Squares, Redux
Orthogonal Functions

Linearly Independent Functions: Polynomials.

Theorem

If $\Phi_j(x)$ is a polynomial of degree j, then the set $\{\Phi_0(x), \Phi_1(x), \ldots, \Phi_n(x)\}$ is linearly independent on any interval $[a, b]$.

Proof.

Suppose $c_i \in \mathbb{R}$, $i = 0, 1, \ldots, n$, and $P(x) = \sum_{i=0}^{n} c_i \Phi_i(x) = 0$ $\forall x \in [a, b]$. Since $P(x)$ vanishes on $[a, b]$ it must be the zero-polynomial, *i.e.* the coefficients of all the powers of x must be zero. In particular, the coefficient of x^n is zero. $\Rightarrow c_n = 0$, hence $P(x) = \sum_{i=0}^{n-1} c_i \Phi_i(x)$. By repeating the same argument, we find $c_i = 0$, $i = 0, 1, \ldots, n$. $\Rightarrow \{\Phi_0(x), \Phi_1(x), \ldots, \Phi_n(x)\}$ is linearly independent. □

Discrete Least Squares Approximation
Continuous Least Squares Approximation
Orthogonal Polynomials
Linear Independence... Weight Functions... Inner Products
Least Squares, Redux
Orthogonal Functions

More Definitions and Notation... Weight Function

Theorem

If $\{\Phi_0(x), \Phi_1(x), \ldots, \Phi_n(x)\}$ is a collection of linearly independent polynomials in $\mathcal{P}_n$, then any $p(x) \in \mathcal{P}_n$ can be written uniquely as a linear combination of $\{\Phi_0(x), \Phi_1(x), \ldots, \Phi_n(x)\}$.

Definition (Weight Function)

An integrable function w is called a weight function on the interval $[a, b]$ if $w(x) \geq 0$ $\forall x \in [a, b]$, but $w(x) \not\equiv 0$ on any subinterval of $[a, b]$.

Discrete Least Squares Approximation
Continuous Least Squares Approximation
Orthogonal Polynomials
Linear Independence... Weight Functions... Inner Products
Least Squares, Redux
Orthogonal Functions

Weight Function... Inner Product

A weight function will allow us to assign different degrees of importance to different parts of the interval. *E.g.* with $w(x) = 1/\sqrt{1 - x^2}$ on $[-1, 1]$ we are assigning more weight away from the center of the interval.

Inner Product, with a weight function:

$$\langle f(x), g(x) \rangle_{w(x)} = \int_a^b f(x) g(x)^* \, w(x) dx.$$

Revisiting Least Squares Approximation with New Notation.

Suppose $\{\Phi_0(x), \Phi_1(x), \ldots, \Phi_n(x)\}$ is a set of linearly independent functions on $[a, b]$, $w(x)$ a weight function on $[a, b]$, and $f(x) \in C[a, b]$.

We are now looking for the linear combination

$$p(x) = \sum_{k=0}^{n} a_k \Phi_k(x)$$

which minimizes the sum-of-squares-error

$$E(\tilde{\mathbf{a}}) = \int_a^b [p(x) - f(x)]^2 \; w(x) dx.$$

When we differentiate with respect to a_k, $w(x)$ is a constant, so the system of normal equations can be written...

The Normal Equations, Revisited for the n^{th} Time.

$$\sum_{k=0}^{n} a_k \langle \Phi_k(x), \, \Phi_j(x) \rangle_{w(x)} = \langle f(x), \, \Phi_j(x) \rangle_{w(x)}, \quad j = 0, 1, \ldots, n.$$

What has changed?

$$\begin{cases} x^k & \to \quad \Phi_k(x) & \text{New basis functions.} \\ \langle \circ, \, \circ \rangle & \to \quad \langle \circ, \, \circ \rangle_{w(x)} & \text{New inner product.} \end{cases}$$

Q: — Is he ever going to get to the point?!? Why are we doing this?

We are going to select the basis functions $\Phi_k(x)$ so that the **normal equations are easy to solve!**

Orthogonal Functions

Definition (Orthogonal Set of Functions)

$\{\Phi_0(x), \Phi_1(x), \ldots, \Phi_n(x)\}$ is said to be **an orthogonal set of functions** on $[a, b]$ with respect to the weight function $w(x)$ if

$$\langle \Phi_i(x), \, \Phi_j(x) \rangle_{w(x)} = \begin{cases} 0, & \text{when } i \neq j, \\ a_i, & \text{when } i = j. \end{cases}$$

If in addition $a_i = 1$, $i = 0, 1, \ldots, n$ the set is said to be **orthonormal.**

The Payoff — No Matrix Inversion Needed.

Theorem

If $\{\Phi_0(x), \Phi_1(x), \ldots, \Phi_n(x)\}$ is a set of orthogonal functions on an interval $[a, b]$, with respect to the weight function $w(x)$, then the least squares approximation to $f(x)$ on $[a, b]$ with respect to $w(x)$ is

$$p(x) = \sum_{k=0}^{n} a_k \Phi_k(x),$$

where, for each $k = 0, 1, \ldots, n$,

$$a_k = \frac{\langle \Phi_k(x), \, f(x) \rangle_{w(x)}}{\langle \Phi_k(x), \, \Phi_k(x) \rangle_{w(x)}}.$$

We can find the coefficients without solving $X^T X \vec{a} = X^T \vec{b}$!!!

Where do we get a set of orthogonal functions???

Building Orthogonal Sets of Functions — The Gram-Schmidt Process

Theorem (Gram-Schmidt Orthogonalization)

The set of polynomials $\{\Phi_0(x), \Phi_1(x), \ldots, \Phi_n(x)\}$ defined in the following way is orthogonal on $[a, b]$ with respect to $w(x)$:

$$\Phi_0(x) = 1, \quad \Phi_1(x) = (x - b_1)\Phi_0,$$

where

$$b_1 = \frac{\langle x\Phi_0(x),\, \Phi_0(x)\rangle_{w(x)}}{\langle \Phi_0(x),\, \Phi_0(x)\rangle_{w(x)}},$$

for $k \geq 2$,

$$\Phi_k(x) = (x - b_k)\Phi_{k-1}(x) - c_k\Phi_{k-2}(x),$$

where

$$b_k = \frac{\langle x\Phi_{k-1}(x),\, \Phi_{k-1}(x)\rangle_{w(x)}}{\langle \Phi_{k-1}(x),\, \Phi_{k-1}(x)\rangle_{w(x)}}, \quad c_k = \frac{\langle x\Phi_{k-1}(x),\, \Phi_{k-2}(x)\rangle_{w(x)}}{\langle \Phi_{k-2}(x),\, \Phi_{k-2}(x)\rangle_{w(x)}}.$$

Example: Legendre Polynomials

The set of Legendre Polynomials $\{P_n(x)\}$ is orthogonal on $[-1, 1]$ with respect to the weight function $w(x) = 1$.

$$P_0(x) = 1, \quad P_1(x) = (x - b_1) \circ 1$$

where

$$b_1 = \frac{\int_{-1}^{1} x\, dx}{\int_{-1}^{1} dx} = 0$$

i.e. $\mathbf{P_1(x) = x}$.

$$b_2 = \frac{\int_{-1}^{1} x^3\, dx}{\int_{-1}^{1} x^2\, dx} = 0, \quad c_2 = \frac{\int_{-1}^{1} x^2\, dx}{\int_{-1}^{1} 1\, dx} = 1/3,$$

i.e. $\mathbf{P_2(x) = x^2 - 1/3}$.

Example: Legendre Polynomials

The first six Legendre Polynomials are

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x \\ P_2(x) &= x^2 - 1/3 \\ P_3(x) &= x^3 - 3x/5 \\ P_4(x) &= x^4 - 6x^2/7 + 3/35 \\ P_5(x) &= x^5 - 10x^3/9 + 5x/21. \end{aligned}$$

We encountered the Legendre polynomials in the context of numerical integration. It turns out that the **roots** of the Legendre polynomials are used as the nodes in Gaussian quadrature.

Now we have the machinery to manufacture Legendre polynomials of any degree.

Example: Laguerre Polynomials

The set of Laguerre Polynomials $\{L_n(x)\}$ is orthogonal on $(0, \infty)$ with respect to the weight function $w(x) = e^{-x}$.

$L_0(x) = 1$,

$$b_1 = \frac{\langle x,\, 1\rangle_{e^{-x}}}{\langle 1,\, 1\rangle_{e^{-x}}} = 1$$

$\mathbf{L_1(x) = x - 1}$,

$$b_2 = \frac{\langle x(x-1),\, x-1\rangle_{e^{-x}}}{\langle x-1,\, x-1\rangle_{e^{-x}}} = 3, \quad c_2 = \frac{\langle x(x-1),\, 1\rangle_{e^{-x}}}{\langle 1,\, 1\rangle_{e^{-x}}} = 1,$$

$\mathbf{L_2(x) = (x-3)(x-1) - 1 = x^2 - 4x + 2}$.