Approximation Theory

Discrete Least Squares Approximation

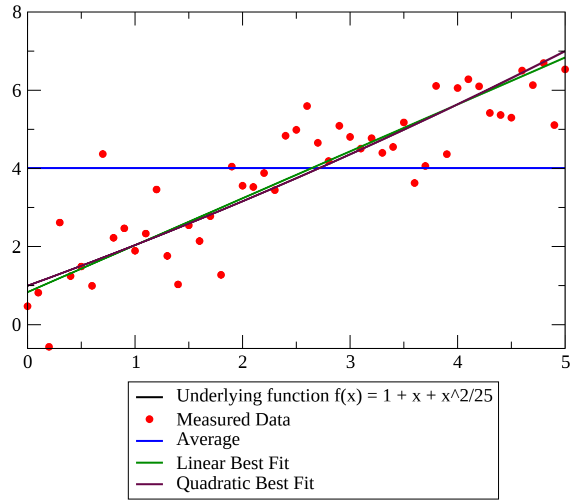
Lecture Notes #10

Joe Mahaffy
Department of Mathematics
San Diego State University
San Diego, CA 92182-7720
mahaffy@math.sdsu.edu
http://www-rohan.sdsu.edu/∼jmahaffy

$Id: lecture.tex,v 1.10 2007/11/08 18:49:01 mahaffy Exp $

Introduction: Matching a Few Parameters to a Lot of Data.

Sometimes we get a lot of data, ***many observations***, and want to fit it to a simple model.



PDF-link: code.

Why a Low Dimensional Model?

Low dimensional models (*e.g.* low degree polynomials) are **easy to work with**, and are quite **well behaved** (high degree polynomials can be quite oscillatory.)

All measurements are **noisy**, to some degree. Often, we want to use a large number of measurements in order to "average out" random noise.

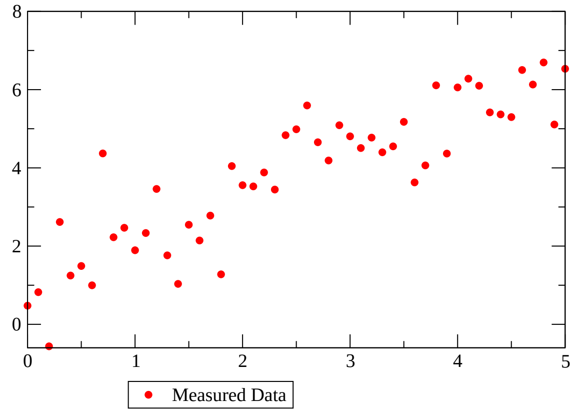
Approximation Theory looks at two problems:

[1] Given a data set, find the best fit for a model (*i.e.* in a class of functions, find the one that best represents the data.)

[2] Find a simpler model approximating a given function.

Interpolation: A Bad Idea?

We can probably agree that trying to interpolate this data set:



with a 50th degree polynomial is not the best idea in the world... Even fitting a cubic spline to this data gives wild oscillations!
[***I tried, and it was not pretty!***]

Defining "Best Fit" — the Residual.

We are going to ***relax*** the requirement that the approximating function must pass through all the data points.

Now we need a measurement of how well our approximation fits the data. — A definition of "best fit."

If $f(x_i)$ are the measured function values, and $a(x_i)$ are the values of our approximating functions, we can define a function, $r(x_i) = f(x_i) - a(x_i)$ which measures the deviation **(residual)** at x_i. Notice that $\vec{\mathbf{r}} = \{r(x_0), r(x_1), \dots, r(x_n)\}^T$ is a vector.

Notation: From now on, $f_i = f(x_i)$, $a_i = a(x_i)$, and $r_i = r(x_i)$. Further, $\vec{\mathbf{f}} = \{f_0, f_1, \dots, f_n\}^T$, $\vec{\mathbf{a}} = \{a_0, a_1, \dots, a_n\}^T$, and $\vec{\mathbf{r}} = \{r_0, r_1, \dots, r_n\}^T$.

What is the Size of the Residual?

There are many possible choices, *e.g.*

- The abs-sum of the deviations:

$$E_1 = \sum_{i=0}^{n} |r_i| \quad \Leftrightarrow \quad E_1 = \|\vec{\mathbf{r}}\|_1$$

- The sum-of-the-squares of the deviations:

$$E_2 = \sqrt{\sum_{i=0}^{n} |r_i|^2} \quad \Leftrightarrow \quad E_2 = \|\vec{\mathbf{r}}\|_2$$

- The largest of the deviations:

$$E_\infty = \max_{0\le i\le n} |r_i| \quad \Leftrightarrow \quad E_\infty = \|\vec{\mathbf{r}}\|_\infty$$

In most cases, the **sum-of-the-squares** version is the easiest to work with. (From now on we will focus on this choice...)

Discrete Least Squares Approximation.

We have chosen the sum-of-squares measurement for errors. Lets find the **constant** that best fits the data, *minimize*

$$E(C) = \sum_{i=0}^{n} (f_i - C)^2.$$

If C^* is a minimizer, then $E'(C^*) = 0$ [the derivative at a max/min is zero]

$$E'(C) = -\sum_{i=0}^{n} 2(f_i - C) = \underbrace{-2\sum_{i=0}^{n} f_i + 2(n+1)C,}_{\text{Set =0, and solve for C}} \quad E''(C) = \underbrace{2(n+1)}_{\text{Positive}}$$

hence

$$\mathbf{C^* = \frac{1}{n+1}\sum_{i=0}^{n} f_i}, \quad \text{it is a min since } E''(C^*) = 2(n+1) > 0.$$

is the constant that best the fits the data. (***Note:*** C^* is the average.)

Discrete Least Squares: Linear Approximation.

The form of Least Squares you are most likely to see: ***Find the Linear Function,*** $p_1(x) = a_0 + a_1 x$ ***that best fits the data.*** The error $E(a_0, a_1)$ we need to minimize is:

$$E(a_0, a_1) = \sum_{i=0}^{n} \left[(a_0 + a_1 x_i) - f_i\right]^2.$$

The first partial derivatives with respect to a_0 and a_1 better be zero at the minimum:

$$\begin{aligned} \frac{\partial}{\partial a_0} E(a_0, a_1) &= 2\sum_{i=0}^{n} \left[(a_0 + a_1 x_i) - f_i\right] &= 0 \\ \frac{\partial}{\partial a_1} E(a_0, a_1) &= 2\sum_{i=0}^{n} x_i \left[(a_0 + a_1 x_i) - f_i\right] &= 0. \end{aligned}$$

We "massage" these expressions to get the **Normal Equations**...

Linear Approximation: The Normal Equations $p_1(x)$

$$\begin{cases} \mathbf{a_0}(n+1) + \mathbf{a_1}\sum_{i=0}^{n} x_i = \sum_{i=0}^{n} f_i \\ \mathbf{a_0}\sum_{i=0}^{n} x_i + \mathbf{a_1}\sum_{i=0}^{n} x_i^2 = \sum_{i=0}^{n} x_i f_i. \end{cases}$$

Since **everything except $\mathbf{a_0}$ and $\mathbf{a_1}$** is known, this is a 2-by-2 system of equations.

$$\begin{bmatrix} (n+1) & \sum_{i=0}^{n} x_i \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} \mathbf{a_0} \\ \mathbf{a_1} \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} f_i \\ \sum_{i=0}^{n} x_i f_i \end{bmatrix}.$$

Quadratic Model, $p_2(x)$

For the quadratic polynomial $p_2(x) = a_0 + a_1 x + a_2 x^2$, the error is given by

$$E(a_0, a_1, a_2) = \sum_{i=0}^{n} \left[a_0 + a_1 x_i + a_2 x_i^2 - f_i\right]^2$$

At the minimum (best model) we must have

$$\begin{aligned} \frac{\partial}{\partial a_0} E(a_0, a_1, a_2) &= 2\sum_{i=0}^{n} \left[(a_0 + a_1 x_i + a_2 x_i^2) - f_i\right] &= 0 \\ \frac{\partial}{\partial a_1} E(a_0, a_1, a_2) &= 2\sum_{i=0}^{n} x_i \left[(a_0 + a_1 x_i + a_2 x_i^2) - f_i\right] &= 0 \\ \frac{\partial}{\partial a_2} E(a_0, a_1, a_2) &= 2\sum_{i=0}^{n} x_i^2 \left[(a_0 + a_1 x_i + a_2 x_i^2) - f_i\right] &= 0. \end{aligned}$$

Quadratic Model: The Normal Equations $p_2(x)$

Similarly for the quadratic polynomial $p_2(x) = a_0 + a_1 x + a_2 x^2$, the normal equations are:

$$\begin{cases} \mathbf{a_0}(n+1) + \mathbf{a_1}\sum_{i=0}^{n} x_i + \mathbf{a_2}\sum_{i=0}^{n} x_i^2 = \sum_{i=0}^{n} f_i \\ \mathbf{a_0}\sum_{i=0}^{n} x_i + \mathbf{a_1}\sum_{i=0}^{n} x_i^2 + \mathbf{a_2}\sum_{i=0}^{n} x_i^3 = \sum_{i=0}^{n} x_i f_i. \\ \mathbf{a_0}\sum_{i=0}^{n} x_i^2 + \mathbf{a_1}\sum_{i=0}^{n} x_i^3 + \mathbf{a_2}\sum_{i=0}^{n} x_i^4 = \sum_{i=0}^{n} x_i^2 f_i. \end{cases}$$

Note: Even though the model is quadratic, the resulting (normal) equations are **linear**. — The model is linear in its parameters, a_0, a_1, and a_2.

The Normal Equations — As Matrix Equations.

We rewrite the Normal Equations as:

$$\begin{bmatrix} (n+1) & \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i^3 \\ \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i^3 & \sum_{i=0}^{n} x_i^4 \end{bmatrix} \begin{bmatrix} \mathbf{a_0} \\ \mathbf{a_1} \\ \mathbf{a_2} \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} f_i \\ \sum_{i=0}^{n} x_i f_i. \\ \sum_{i=0}^{n} x_i^2 f_i. \end{bmatrix}.$$

It is not immediately obvious, but this expression can be written in the form $\mathbf{A^T A \vec{a}} = \mathbf{A^T \vec{f}}$. Where the matrix A is very easy to write in terms of x_i. [Jump Forward].

The Polynomial Equations in Matrix Form

We can express the mth degree polynomial, $p_m(x)$, evaluated at the points x_i:

$$a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_m x_i^m = f_i, \quad i = 0, \ldots, n$$

as a product of an $(n+1)$-by-$(m+1)$ matrix, A and the $(m+1)$-by-1 vector $\vec{\mathbf{a}}$ and the result is the $(n+1)$-by-1 vector $\vec{\mathbf{f}}$, usually $n \gg m$:

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^m \\ 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ 1 & x_3 & x_3^2 & \cdots & x_3^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}}_{\vec{\mathbf{a}}} = \underbrace{\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_n \end{bmatrix}}_{\vec{\mathbf{f}}}.$$

Building a Solvable System from $A\vec{a} = \vec{\mathbf{f}}$

We cannot immediately solve the linear system

$$A\vec{\mathbf{a}} = \vec{\mathbf{f}}$$

when A is a rectangular matrix $(n+1)$-by-$(m+1)$, $m \neq n$.

We can generate a solvable system by multiplying both the left- and right-hand-side by A^T, *i.e.*

$$\mathbf{A^T A}\vec{\mathbf{a}} = \mathbf{A^T}\vec{\mathbf{f}}$$

Now, the matrix $A^T A$ is a square $(m+1)$-by-$(m+1)$ matrix, and $A^T\vec{\mathbf{f}}$ an $(m+1)$-by-1 vector.

Let's take a closer look at $A^T A$, and $A^T\vec{\mathbf{f}}$...

Computing $A^T A$.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ x_0 & x_1 & x_2 & x_3 & \ldots & x_n \\ x_0^2 & x_1^2 & x_2^2 & x_3^2 & \ldots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_0^m & x_1^m & x_2^m & x_3^m & \ldots & x_n^m \end{bmatrix} \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^m \\ 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ 1 & x_3 & x_3^2 & \cdots & x_3^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix}$$

$$= \begin{bmatrix} n+1 & \sum_{i=0}^n x_i^1 & \cdots & \sum_{i=0}^n x_i^m \\ \sum_{i=0}^n x_i^1 & \sum_{i=0}^n x_i^2 & \cdots & \sum_{i=0}^n x_i^{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^n x_i^m & \sum_{i=0}^n x_i^{m+1} & \cdots & \sum_{i=0}^n x_i^{2m} \end{bmatrix}.$$

Computing $A^T f$.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ x_0 & x_1 & x_2 & x_3 & \ldots & x_n \\ x_0^2 & x_1^2 & x_2^2 & x_3^2 & \ldots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_0^m & x_1^m & x_2^m & x_3^m & \ldots & x_n^m \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^n f_i \\ \sum_{i=0}^n x_i f_i \\ \sum_{i=0}^n x_i^2 f_i \\ \vdots \\ \sum_{i=0}^n x_i^m f_i. \end{bmatrix}$$

We have recovered the Normal Equations...

[Jump Back].

Discrete Least Squares: A Simple, Powerful Method.

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

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

E.g. to compute the best fitting polynomial of degree 3, $p_3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$, define:

$$A = \begin{bmatrix} | & | & | & | \\ \vec{\mathbf{1}} & \vec{\mathbf{x}} & \vec{\mathbf{x}}^2 & \vec{\mathbf{x}}^3 \\ | & | & | & | \end{bmatrix}, \quad \text{and compute} \quad \vec{\mathbf{a}} = (A^T A)^{-1}(A^T \vec{\mathbf{f}}).$$

Discrete Least Squares: Matlab Example.

I used this code to generate the data for the plots on slide 2.

```
x  = (0:0.1:5)';                 % The x-vector
f  = 1+x+x.∧2/25;                % The underlying function
n  = randn(size(x));             % Random perturbations
fn = f+n;                        % Add randomness
A  = [x ones(size(x))];          % Build A for linear fit
%a  = (A'*A)\(A'*f);             % Solve
a  = A\f;                        % Better, Equivalent, Solve
p1 = polyval(a,x);               % Evaluate
A  = [x.∧2 x ones(size(x))];     % A for quadratic fit
%a  = (A'*A)\(A'*f);             % Solve
a  = A\f;                        % Better, Equivalent, Solve
p2 = polyval(a,x);               % Evaluate
```

But... I do not want to fit a polynomial!!!

Fitting an **exponential model** $g(x) = be^{cx}$ to the given data $\vec{\mathbf{d}}$, is quite straight-forward.

First, re-cast the problem as a set of linear equations. We have:

$$be^{cx_i} = d_i, \quad i = 0, \ldots, n$$

compute the natural logarithm on both sides:

$$\underbrace{\ln b}_{a_0} + \underbrace{c}_{a_1} x_i = \underbrace{\ln d_i}_{f_i}.$$

Now, we can apply a polynomial least squares fit to the problem, and once we have (a_0, a_1), $b = e^{a_0}$ and $c = a_1$.

Note: This does **not** give the least squares fit to the original problem!!! (It gives us a pretty good estimate.)

But... That is not a True Least Squares Fit!

Note: Fitting the modified problem does **not** give the least squares fit to the original problem!!!

In order to find the true least squares fit we need to know how to find roots and/or minima/maxima of non-linear systems of equations.

Feel free to sneak a peek at Burden-Faires chapter 10. Unfortunately we do not have the time to talk about this here...

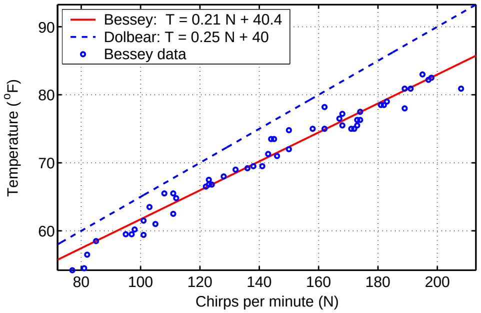
What we need: **`Math 693a` — Numerical Optimization Techniques.**

Some of this stuff may show up in a different context in: **`Math 562` — Mathematical Methods of Operations Research.**

Cricket Thermometer Application

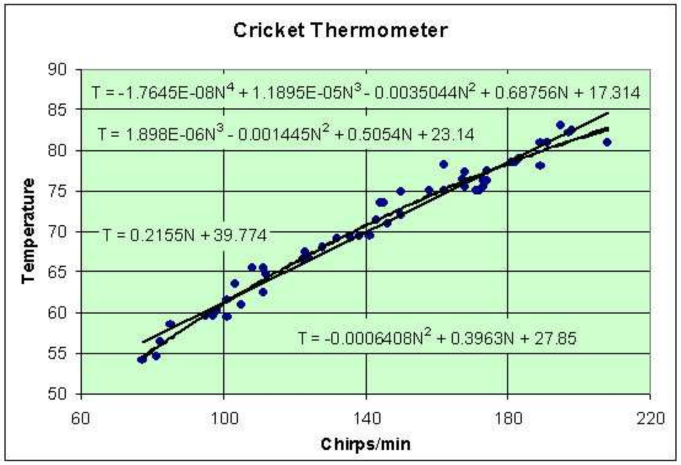
There is a folk method of approximating the temperature (in Fahrenheit). This entered the scientific literature in 1896 by Dolbear with data collected by the Bessey brothers in 1898.

The temperature is approximated from the rate of crickets chirping by taking the number of chirps/min dividing by 4 and adding 40.



Polynomial Fits to the Data

Excel's Trendline was used to fit linear, quadratic, cubic, and quartic polynomials.



Cricket Data Analysis

C. A. Bessey and E. A. Bessey collected data on eight different crickets that they observed in Lincoln, Nebraska during August and September, 1897. The number of chirps/min was N and the temperature was T.

Create matrices

$$A_1 = \begin{pmatrix} 1 & N_1 \\ 1 & N_2 \\ \vdots & \vdots \end{pmatrix} \qquad A_2 = \begin{pmatrix} 1 & N_1 & N_1^2 \\ 1 & N_2 & N_2^2 \\ \vdots & \vdots & \vdots \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 1 & N_1 & N_1^2 & N_1^3 \\ 1 & N_2 & N_2^2 & N_2^3 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Cricket Linear Model

Compute the matrix

$$A_1^T A_1 = \begin{pmatrix} 52 & 7447 \\ 7447 & 1133259 \end{pmatrix}$$

This has eigenvalues

$$\lambda_1 = 3.0633 \quad \text{and} \quad \lambda_2 = 1,133,308$$

which gives the condition number

$$cond(A_1^T A_1) = 3.6996 \times 10^5.$$

With MatLab's linsolve solving for coefficients a,

$$A_1^T A_1 a = A_1^T T,$$

we obtain the best linear model

$$T = 0.21548\,N + 39.744.$$

Cricket Quadratic Model

Similarly, compute the matrix

$$A_2^T A_2 = \begin{pmatrix} 52 & 7447 & 1133259 \\ 7447 & 1133259 & 1.8113 \times 10^8 \\ 1133259 & 1.8113 \times 10^8 & 3.0084 \times 10^10 \end{pmatrix}$$

This has eigenvalues

$$\lambda_1 = 0.1957 \quad \lambda_2 = 42,706 \quad \lambda_3 = 3.00853 \times 10^{10}$$

which gives the condition number

$$cond(A_2^T A_2) = 1.5371 \times 10^{11}.$$

With MatLab's linsolve, we obtain the best quadratic model

$$T = -0.00064076\, N^2 + 0.39625\, N + 27.849.$$

Cricket Cubic and Quartic Models

The condition numbers for the cubic and quartic rapidly get larger with

$$cond(A_3^T A_3) = 6.3648 \times 10^{16} \quad \text{and} \quad cond(A_4^T A_4) = 1.1218 \times 10^{23}$$

These last two condition numbers suggest that any coefficients obtained are highly suspect.

The best cubic and quartic models are given by

$$\begin{aligned} T &= 0.0000018977\, N^3 - 0.001445\, N^2 + 0.50540\, N + 23.138 \\ T &= -0.00000001765\, N^4 + 0.00001190\, N^3 - 0.003504\, N^2 \\ &= +0.6876\, N + 17.314 \end{aligned}$$

Best Cricket Model

So how does one select the best model?

Visually, one can see that the linear model does a very good job, and one only obtains a slight improvement with a quadratic. Is it worth the added complication for the slight improvement.

It is clear that the sum of square errors (SSE) will improve as the number of parameters increase.

From statistics, it is hotly debated how much penalty one should pay for adding parameters.

Best Cricket Model - Analysis

Bayesian Information Criterion: Let n be the number of data points, SSE be the sum of square errors, and let k be the number of parameters in the model.

$$BIC = n \ln(SSE/n) + k \ln(n).$$

Akaike Information Criterion:

$$AIC = 2k + n(\ln(2\pi SSE/n) + 1).$$

Best Cricket Model - Analysis Continued

The table below shows the by the Akaike information criterion that we should take a quadratic model, while using a Bayesian Information Criterion we should use a cubic model.

	Linear	Quadratic	Cubic	Quartic
SSE	108.8	79.08	78.74	78.70
BIC	46.3	33.65	33.43	37.35
AIC	189.97	175.37	177.14	179.12

Returning to the original statement, we do fairly well by using the folk formula, despite the rest of this analysis!