

# Data Fitting

Com S 477/577

Nov 4, 2008

## 1 Introduction

Given a set of observations, one often wants to condense and summarize the data by fitting it to a “model” that depends on adjustable parameters. Sometimes the model is simply a class of functions, such as polynomials or trigonometric functions, and the fit determines the appropriate coefficients. Other times, the model’s parameters come from underlying theory that the data are supposed to satisfy. Modeling can also be used as a kind of constrained interpolation, where you want to extend a few data points into a continuous function, but with some underlying idea of what the function should look like.

Imagine that we have taken measurements of some unknown function  $f$  at points  $x_1, \dots, x_n$ . We would like to “reconstruct” this function to our best. Let us refer to the  $n$  measurements as  $f_1, \dots, f_n$ . If these measurements were perfect, then  $f_i = f(x_i)$  would hold for  $i = 1, \dots, n$ . Generally, they are subject to measurement errors (for instance, noise in the context of signal processing) so that

$$f_i = f(x_i) + \epsilon_i,$$

where  $\epsilon_i$  is some unknown error. Nevertheless, we would like to try to recover  $f(x)$ . Were it not for the measurement errors, we might consider using interpolation. But it does not make sense to force an interpolating polynomial to pass through points  $(x_i, f_i)$  that are errant. The polynomial may just wiggle around a lot to pass through all the points  $(x_i, f_i)$  and thus have an order much higher than  $f(x)$  really has.

EXAMPLE 1. Suppose we have taken measurements of the function  $f(x) = (x - 1)^2$  at 7 points, evenly spaced over the interval  $[-1, 2]$ . The following table lists the points  $x_i$ , the true values  $f(x_i)$ , and the (errant) measurements  $f_i$ .

$x_i$	$f(x_i)$	$f_i$
-1	4	4.1
$-\frac{1}{2}$	2.25	2.3
0	1	1.05
$\frac{1}{2}$	.25	.20
1	0	.05
$\frac{3}{2}$	.25	.26
2	1	.90

The function  $f(x)$  is a quadratic. If we interpolated at the supporting points  $\{(x_i, f_i)\}$  we would get a 6th order polynomial. No doubt that this polynomial would contain some twists and turns that  $f(x)$  does not. In the general case, the interpolating polynomial matches  $f(x)$  well at the supporting points, but its derivatives may not.

Of course, there are many functions  $\{g(x)\}$  that could produce the measured values  $f_i$  above. How could we possibly hope to recover the correct function? Without further information, we cannot expect to recover the correct one.

Suppose, however, that we know that the underlying function is quadratic. Then we could pick the following as *basis functions*<sup>1</sup>

$$1, \quad x, \quad x^2$$

and seek to find coefficients  $a, b, c$  such that

$$f(x) = a \cdot 1 + b \cdot x + c \cdot x^2.$$

If our measurements  $f_i$  were perfect, then the system of equations

$$f_i = a + bx_i + cx_i^2, \quad i = 1, \dots, 7, \tag{1}$$

in the variables  $a, b, c$  would have a unique solution.

If the  $f_i$  are imperfect, then the system (1) is generally overconstrained. In this case, we can still obtain a least-squares solution, using for example, the technique of singular value decomposition.

In the general version of data fitting, we are given  $n$  measurements  $(x_i, f_i)$ . And we would like to reconstruct the function  $f(x)$ . What we have is some parameterized family of functions

$$F(x) = F(x; c_1, \dots, c_k).$$

We then choose the parameters  $c_1, \dots, c_k$  based on the observations  $\{(x_i, f_i)\}$  in such a way that  $F(x)$  is “close to”  $f(x)$ . Often, for mathematical simplicity we write  $F(x)$  as a linear combination of some  $k$  basis functions:

$$F(x) = c_1\phi_1(x) + \dots + c_k\phi_k(x).$$

Here  $k$  should be large enough so that the information about  $f(x)$  can be represented by the choice of  $c_1, \dots, c_k$  while in the meantime it should be small enough to avoid reproduction of noise. Our objective is to choose the coefficients  $c_1, \dots, c_k$  well.

Why do we do things this way? The point is that  $k$  will in general be much smaller than  $n$ . So rather than retain all  $n$  pieces of data (as with interpolation), most of which really contains little information, we try to extract the important or useful information. That information is encoded in the  $k$  basis functions.

## 2 Least Squares

How are the coefficients  $\{c_i\}$  chosen? Ideally, we would like to minimize the difference between  $f(x)$  and  $F(x)$ . Unfortunately, we do not know  $f(x)$ , so instead we simply minimize the difference

---

<sup>1</sup>Other basis functions are possible and indeed sometimes desirable, as long as they are independent and span the vector space of quadratic functions. Later we will also look at orthogonal bases.

between the two functions at the data points  $x_1, \dots, x_n$ . There are a variety of norms by which to measure the error:

$$\begin{aligned} \infty\text{-norm:} \quad \|f - F\|_\infty &= \max_{1 \leq i \leq n} |f_i - F(x_i)|, \\ 1\text{-norm:} \quad \|f - F\|_1 &= \sum_{i=1}^n |f_i - F(x_i)|, \\ p\text{-norm:} \quad \|f - F\|_p &= \sqrt[p]{\sum_{i=1}^n |f_i - F(x_i)|^p}. \end{aligned}$$

Recall that the parameters  $\{c_j\}$  are hidden in this notation. In fact,  $F(x_i) = F(x_i; c_1, \dots, c_k)$ . For each norm, the goal would be to choose these parameters to minimize the error. Such a minimization process would lead to solution of nonlinear equations in  $c_1, \dots, c_k$ , even when  $F(x)$  has the simple linear form  $F(x) = c_1\phi_1(x) + \dots + c_k\phi_k(x)$ . However, as we shall see, in the case of a 2-norm with  $F(x)$  having the linear form, error minimization leads to linear equations that determine  $c_1, \dots, c_k$ . For this practical reason, 2-norms, that is, *least squares*, are so popular.

More formally, we want to choose  $\{c_j\}$  to minimize the quantity

$$\|f - F\|_2 = \sqrt{\sum_{i=1}^n |f_i - F(x_i; c_1, \dots, c_k)|^2}.$$

Equivalently, we are choosing  $\mathbf{c} = (c_1, \dots, c_k)^T$  to minimize the function

$$E(\mathbf{c}) = \sum_{i=1}^n (f_i - F(x_i; \mathbf{c}))^2.$$

At a minimum of  $E(\mathbf{c})$ , each of the partial derivatives vanishes, that is,

$$\frac{\partial}{\partial c_j} E(\mathbf{c}) = 0, \quad j = 1, \dots, k.$$

Below we determine these partial derivatives:

$$\begin{aligned} \frac{\partial}{\partial c_j} E(\mathbf{c}) &= -2 \sum_{i=1}^n (f_i - F(x_i; \mathbf{c})) \frac{\partial}{\partial c_j} F(x_i; \mathbf{c}) \\ &= -2 \sum_{i=1}^n (f_i - F(x_i; \mathbf{c})) \phi_j(x_i), \quad \text{since } F(x; \mathbf{c}) = \sum_{j=1}^k c_j \phi_j(x). \end{aligned}$$

Namely, that  $E(\mathbf{c})$  is a minimum only if

$$\sum_{i=1}^n (f_i - F(x_i; \mathbf{c})) \phi_j(x_i) = 0, \quad j = 1, \dots, k. \quad (2)$$

This system of equations is called the system of *normal equations*. The vector

$$\mathbf{e} = \begin{pmatrix} f_1 - F(x_1; \mathbf{c}) \\ \vdots \\ f_n - F(x_n; \mathbf{c}) \end{pmatrix}$$

measures the error at the  $n$  data points. The vectors

$$\phi_j = \begin{pmatrix} \phi_j(x_1) \\ \vdots \\ \phi_j(x_n) \end{pmatrix}, \quad j = 1, \dots, k,$$

encode the value of the basis function  $\phi_j$  at the  $n$  data points.

Equations (2) can be written compactly as

$$\mathbf{e}^T \phi_j = 0, \quad j = 1, \dots, k. \quad (3)$$

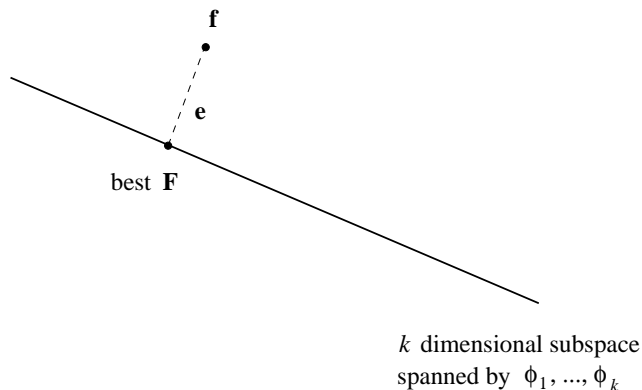
In other words, at the minimizing  $\mathbf{c}$ , the error vector  $\mathbf{e}$  is perpendicular to each of the vectors  $\phi_j$ .

All these vectors reside in  $\mathbb{R}^n$ , the space of possible data values at the points  $x_1, \dots, x_n$ .

This is classic least squares. Write

$$\mathbf{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \quad \text{and} \quad \mathbf{F} = \begin{pmatrix} F(x_1; \mathbf{c}) \\ \vdots \\ F(x_n; \mathbf{c}) \end{pmatrix}.$$

Hence  $\mathbf{e} = \mathbf{f} - \mathbf{F}$ . Then we have the following picture (drawn in  $\mathbb{R}^n$ ).



We often cannot get the exact solution. Instead we get as close as possible, that is, the error vector is perpendicular to the space of possible solutions. The obtained solution  $\mathbf{F} = c_1 \phi_1 + \dots + c_k \phi_k$  is thus the orthogonal projection of  $\mathbf{f}$  (the data vector) onto the hyperplane spanned by  $\phi_1, \dots, \phi_k$  (the space of permissible function vectors). This is precisely what the normal equations (2) says. Practically, we hope that the effect of this orthogonal projection is to remove noise and reconstruct the function  $f(x)$ .

To solve the normal equations (2), we expand them using

$$F(x_i; \mathbf{c}) = c_1 \phi_1(x_i) + \dots + c_k \phi_k(x_i).$$

This yields a linear system of equations

$$\begin{pmatrix} \sum_i \phi_1(x_i) \phi_1(x_i) & \cdots & \sum_i \phi_k(x_i) \phi_1(x_i) \\ \vdots & \ddots & \vdots \\ \sum_i \phi_1(x_i) \phi_k(x_i) & \cdots & \sum_i \phi_k(x_i) \phi_k(x_i) \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_k \end{pmatrix} = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_1(x_n) \\ \vdots & \ddots & \vdots \\ \phi_k(x_1) & \cdots & \phi_k(x_n) \end{pmatrix} \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix},$$

or in short,

$$P\mathbf{c} = \mathbf{q}, \tag{4}$$

where  $P$  is the  $k \times k$  matrix with entries

$$p_{lj} = \boldsymbol{\phi}_l^T \boldsymbol{\phi}_j = \sum_{i=1}^n \phi_l(x_i) \phi_j(x_i),$$

and  $\mathbf{q}$  is the  $k \times 1$  matrix with entries

$$q_l = \mathbf{f}^T \boldsymbol{\phi}_l = \sum_{i=1}^n f_i \phi_l(x_i),$$

and  $\mathbf{c} = (c_1, \dots, c_k)^T$ . It turns out that equation (4) always has a solution. However, the matrix  $P$  is often very ill-conditioned, leading to unreliable results.

But, recall that the normal equations (2) tell us to perform an orthogonal projection in  $\mathbb{R}^n$ . Consider the  $n \times k$  matrix

$$A = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_k(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_n) & \cdots & \phi_k(x_n) \end{pmatrix}.$$

In other words, the  $j$ th column of  $A$  is the vector  $\boldsymbol{\phi}_j$ . Hence

$$\mathbf{F} = A\mathbf{c}.$$

Now consider the overconstrained system

$$A\mathbf{c} = \mathbf{f}.$$

If we solve this system using SVD, we will indeed be projecting  $\mathbf{f}$  orthogonally onto the space spanned by  $\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_k$ . The resulting vector  $\mathbf{c}$  is then our best approximation.

EXAMPLE 2. Let us return to Example 1. The basis functions are

$$\phi_1(x) = 1, \quad \phi_2(x) = x, \quad \text{and} \quad \phi_3(x) = x^2.$$

We seek coefficients  $c_1, c_2, c_3$  such that  $F(x) = c_1 + c_2x + c_3x^2$  is the best least squares approximation to the data  $(x_i, f_i)$ . We have

$$\mathbf{f} = \begin{pmatrix} 4.1 \\ 2.3 \\ 1.05 \\ 0.2 \\ 0.05 \\ 0.26 \\ 0.9 \end{pmatrix}, \quad \boldsymbol{\phi}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad \boldsymbol{\phi}_2 = \begin{pmatrix} -1.0 \\ -0.5 \\ 0 \\ 0.5 \\ 1.0 \\ 1.5 \\ 2.0 \end{pmatrix}, \quad \boldsymbol{\phi}_3 = \begin{pmatrix} 1.0 \\ 0.25 \\ 0 \\ 0.25 \\ 1.0 \\ 2.25 \\ 4.0 \end{pmatrix}.$$

If we use the method  $P\mathbf{c} = \mathbf{q}$ , we would solve

$$\begin{pmatrix} 7 & 3.5 & 8.75 \\ 3.5 & 8.75 & 11.375 \\ 8.75 & 11.375 & 23.1875 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 8.86 \\ -2.91 \\ 8.96 \end{pmatrix}.$$

If we use the SVD method,  $Ac = \mathbf{f}$ , we would solve

$$\begin{pmatrix} 1 & -1.0 & 1.0 \\ 1 & -0.5 & 0.25 \\ 1 & 0 & 0 \\ 1 & 0.5 & 0.25 \\ 1 & 1.0 & 1.0 \\ 1 & 1.5 & 2.25 \\ 1 & 2.0 & 4.0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 4.1 \\ 2.3 \\ 1.05 \\ 0.2 \\ 0.05 \\ 0.26 \\ 0.9 \end{pmatrix}.$$

By either means we find that

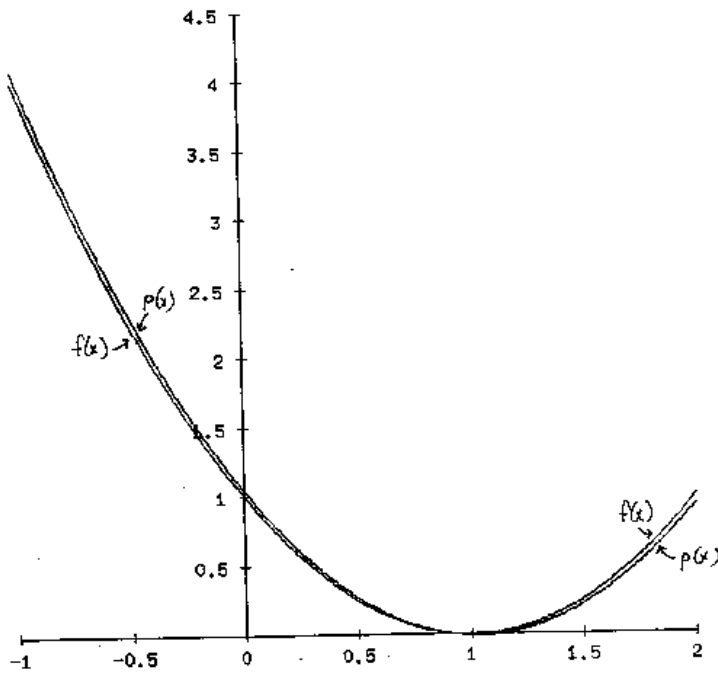
$$c_1 \approx 1.044, \quad c_2 \approx -2.044 \quad c_3 = 0.995.$$

Therefore our estimated polynomial is

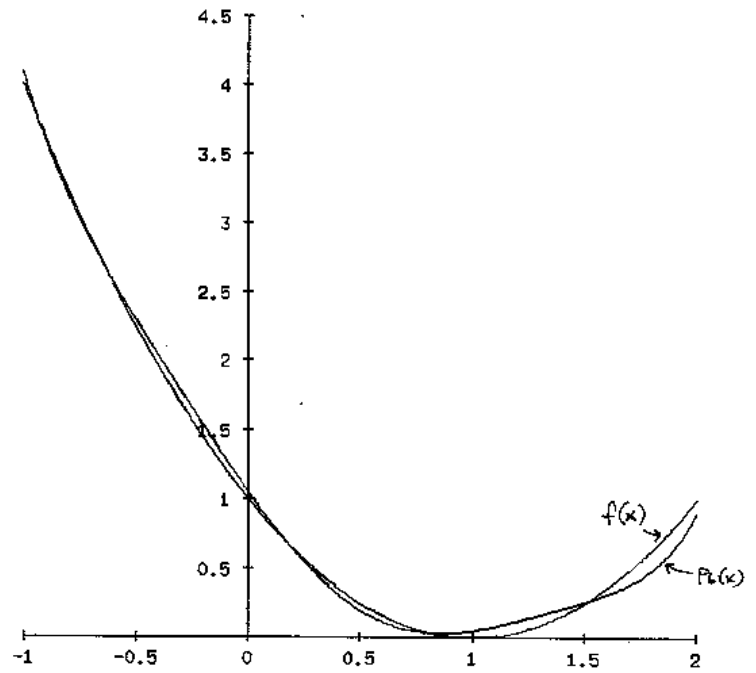
$$p(x) = 1.044 - 2.044x + 0.995x^2,$$

which is not too far from the true function

$$f(x) = 1 - 2x + x^2.$$



(a)



(b)

The diagram<sup>2</sup> (a) above compares the graphs of  $p(x)$  and  $f(x)$ . Notice that there is a small persistent error between  $p(x)$  and  $f(x)$ . However, the shape of  $p$  and  $f$  agree well. This is because we restricted ourselves to a quadratic subspace of the space of functions.

<sup>2</sup>Both diagrams are courtesy of Mike Erdmann.

In contrast, diagram (b) shows the interpolating polynomial  $p_6(x)$  that matches the  $f_i$  values at the  $x_i$ ,  $i = 1, \dots, 7$ . While we get agreement between  $p_6(x)$  and  $f(x)$  more often, the maximum error is substantially larger than it was for  $p(x)$  and  $f(x)$ . Furthermore, being of higher order and by matching the spurious data  $f_i$  exactly,  $p_6(x)$  wiggles around a lot more than  $p(x)$ . Thus the general “shape” is not as good.

### 3 Statistical Rationale for Least Squares

The problem of data fitting can be formally phrased below:

*Given observations  $\{(x_i, f_i)\}$  and a family of functions  $F(x) = \sum_{j=1}^k c_j \phi_j(x)$ , what is the likelihood of a particular set of parameters  $c_1, \dots, c_k$ ?*

Instead of trying to answer the above question directly, we seek to answer the following question:

*Given  $c_1, \dots, c_k$ , what is the probability that  $\{(x_i, f_i)\}$  could have occurred?*

Essentially, we identify the probability given the parameters as the likelihood of the parameters given the data.

Let us treat each  $f_i$  as an independent “random variable” with normal distribution around the “true” value  $F(x_i)$ , that is, the “mean” of  $f_i$ . This yields the probability

$$e^{-\frac{(f_i - F(x_i))^2}{2\sigma^2}} df_i,$$

where  $\sigma$  is the standard deviation of all  $f_i$ . But the above probability is mathematically zero, so we replace  $df_i$  with  $\Delta f_i = \Delta f$ :

$$e^{-\frac{(f_i - F(x_i))^2}{2\sigma^2}} \Delta f.$$

Therefore the probability of the data set  $\{(x_i, f_i)\}$  is

$$p = \prod_{i=1}^n \left( e^{-\frac{(f_i - F(x_i))^2}{2\sigma^2}} \Delta f \right).$$

Maximizing  $p$  is equivalent to minimizing

$$-\ln p = \sum_{i=1}^n \frac{(f_i - F(x_i))^2}{2\sigma^2} - n \ln(\Delta f).$$

Since  $n$ ,  $\sigma$ , and  $\Delta f$  are constants, we end up with the familiar least squares problem:

$$\min \sum_{i=1}^n (f_i - F(x_i))^2.$$

## 4 Curve and Surface Fitting

In computer vision, robotics, and geometric modeling, it is often important to derive some implicit representation of an object from the raw shape data obtained with an imaging or touch sensing device. Such an implicit representation often takes the form of the zero set of a polynomial of even degree.<sup>3</sup> In practice, quartics or superquartics have often been the choices for describing surfaces of moderate geometric complexities.

An implicit representation in polynomial form has several advantages. Firstly, it gives us a succinct shape description with few parameters so storage of a large amount of raw data is no longer necessary. Secondly, using the polynomial we can compute some algebraic invariants (which are expressions in terms of the polynomial's coefficients) to recognize the object from a set of known models. Thirdly, we would be able to differentiate the shape to obtain local geometry indices such as curvature and torsion. Fourthly, evaluating the polynomial at a point would easily tell us whether the point is inside, outside, or on the surface. Finally, a polynomial fit handles noisy data effectively in the sense that most shape irregularities due to noise are filtered out.

In surface fitting, a family of quartic polynomials with stably bounded zero sets is often considered. Choice of the family takes into account factors such as the effectiveness of representation and the computational efficiency of fitting. Polynomial coefficients are the parameters to be determined by, say, a least-squares method. For example, we may minimize the sum of squares of the distances from individual data points to the polynomial surface. We refer the reader to [3, 4, 5] for good instances of surface fitting techniques and experiments.

## References

- [1] M. Erdmann. Lecture notes for *16-811 Mathematical Fundamentals for Robotics*. The Robotics Institute, Carnegie Mellon University, 1998.
- [2] W. H. Press, *et al.* *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 2002.
- [3] M. M. Blane, Z. Lei, H. Civi, D. B. Cooper. The 3L algorithm for fitting implicit polynomial curves and surfaces to data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):298–313, 2000.
- [4] D. Keren and D. Cooper. Describing complicated objects by implicit polynomials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):38–53, 1994.
- [5] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D. J. Kriegman. Parameterized families of polynomials for bounded algebraic curve and surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):287–303, 1994.

---

<sup>3</sup>It is known that the zero sets of polynomials of odd degrees always define unbounded curves and surfaces.