Least Squares Approximation

Michael S. Floater

November 7, 2011

Abstract

For some kinds of data it is better to approximate than to interpolate. In these notes we look at how to approximate data using least squares methods.

1 Univariate data

We begin by considering univariate data: a sequence of increasing real values $x_1 < \cdots < x_m$, together with arbitrary real values $z_1, \ldots, z_m$. We wish to approximate this data by a function

$$s(x) = \sum_{i=1}^{n} B_i(x) c_i, \qquad c_i \in \mathbb{R}.$$

The functions $B_1, \ldots, B_n$ are defined over some interval $[a, b]$ containing the x_k and could for example be B-splines, Bernstein polynomials, monomials, radial functions etc. Loosely speaking we say that s *approximates* the data if $s(x_k) \approx z_k$. One reason to approximate the data rather than interpolate it is that it could be noisy: there could be noise in the values z_k and the points x_k. Another is that m might be very large and by chooising n to be much smaller than m, we end up reducing the amount of data: replacing the m values z_k (and the x_k) by the n values c_i.

The basic *least squares* method consists of finding a coefficient vector $c = (c_1, \ldots, c_n)^T$ which minimizes the sum of squared errors

$$\sum_{k=1}^{m} (s(x_k) - z_k)^2, \tag{1}$$

but sometimes there is no unique solution and it is usual to add a smoothing term such as

$$J(c) = \int_a^b (s''(x))^2 dx \tag{2}$$

(assuming the functions $B_1, \ldots, B_n$ are C^2). This also helps to smooth out the data when it contains noise. There are many possible smoothing terms used in practice but usually they can be expressed as

$$J(c) = c^T E c$$

for some $n \times n$ symmetric positive semidefinite matrix E. This is the case for the smoothing term in (2), which can be rewritten as

$$J(c) = \int_a^b \big(\sum_{i=1}^n B_i''(x)c_i\big)\big(\sum_{j=1}^n B_j''(x)c_j\big)dx = \sum_{i=1}^n \sum_{j=1}^n E_{ij}c_i c_j$$

where

$$E_{ij} = \int_a^b B_i''(x)B_j''(x)dx. \tag{3}$$

It is clear that $E_{ij} = E_{ji}$, and $c^T E c \geq 0$ for any $c \in \mathbb{R}^n$ since $J(c) \geq 0$.

So we will concentrate on minimizing the more general functional

$$F(c) = \sum_{k=1}^m (s(x_k) - z_k)^2 + \lambda c^T E c, \tag{4}$$

for some real $\lambda > 0$.

A minimum of $F(c)$ occurs at a point c where all partial derivatives are zero, a *critical point*. The equations $\partial F/\partial c_i = 0$ are called the *normal equations* of the least squares problem. By differentiating F in (4) explicitly and rearranging the subsequent expression, the normal equations can be rewritten as the single matrix equation

$$(B^T B + \lambda E)c = B^T z, \tag{5}$$

where $z = (z_1, \ldots, z_m)^T$ and B is the $m \times n$ matrix

$$B = (B_j(x_i))_{i=1,\ldots,m,j=1,\ldots,n} = \begin{bmatrix} B_1(x_1) & \ldots & B_n(x_1) \\ \vdots & & \vdots \\ B_1(x_m) & \ldots & B_n(x_m) \end{bmatrix}.$$

Then the solution to the least squares problem is the solution c to (5). The $n \times n$ matrix $G = B^T B$, whose ij-th element is

$$G_{ij} = \sum_{k=1}^{m} B_i(x_k) B_j(x_k),$$

is easily seen to be symmetric and because

$$c^T G c = \|Bc\|_2^2 \geq 0,$$

it is also positive semidefinite. Therefore, assuming that E is also symmetric and positive semidefinite, as is the case when E is given by (3), the combined matrix,

$$A = B^T B + \lambda E,$$

is symmetric and positive semidefinite.

If A is also positive definite, i.e., that $c^T A c = 0$ implies $c = 0$, then the normal equations have a unique solution. This will be the case for example if $m \geq n$ and the matrix B has full rank n. i.e., n of its rows are linearly independent, because then G is positive definite. However, we do not need to check this if we use the smoothing term (2), for there is then a very simple criterion for positive-definiteness, namely that the *number of points in the data set is at least two*, i.e., $m \geq 2$. To see this suppose that $c^T A c = 0$. Then both $c^T G c = 0$ and $cEc = 0$. Since $c^T G c = 0$ we have $Bc = 0$ and therefore $s(x_k) = 0$ for all $k = 1, \ldots, m$. Since $c^T E c = 0$ we have $J(c) = 0$ and therefore $s''(x) = 0$. We conclude that s is a linear polynomial, $s(x) = \alpha + \beta x$, such that $s(x_k) = 0$. If $m \geq 2$, this is only possible if $s = 0$, i.e., $c = 0$.

2 B-splines

Let us now suppose that the functions $B_1, \ldots, B_n$ in (1) are B-splines in which case the function s in (1) is a spline. In this case the matrix B in (5) is sparse.

We write $B_{i,K} = B_i$ to indicate that the B-splines have order K and they are defined over some knot sequence

$$t_1, t_2, \ldots, t_{n+K}.$$

We make the usual assumptions that $t_i \leq t_{i+1}$ and $t_i < t_{i+K}$. One way of defining the B-splines is recursively by the Cox-de Boor algorithm. We define

$$B_{i,1}(x) = \begin{cases} 1 & t_i \leq x < t_{i+1}; \\ 0 & \text{otherwise,} \end{cases}$$

and, for $K \geq 2$,

$$B_{i,K}(x) = \frac{x - t_i}{t_{i+K-1} - t_i} B_{i,K-1}(x) + \frac{t_{i+K} - x}{t_{i+K} - t_{i+1}} B_{i+1,K-1}(x). \tag{6}$$

The piecewise polynomial $B_{i,K}$ has support $[t_i, t_{i+K}]$ and is (strictly) positive in the open interval (t_i, t_{i+K}). If we are applying the smoothing integral in (2) to the interval $[a, b]$, it is natural to let

$$t_1 = \cdots = t_K = a, \qquad t_{n+1} = \cdots t_{n+K} = b.$$

2.1 Solving the linear system

The size of the matrix

$$A = G + \lambda E$$

in (5) is $n \times n$ (and independent of m) and so we should take care about the structure of A when n is large. With the smoothing term given by (2), A is *sparse* because the product $B_i B_j$ is zero if the interiors of the supports of B_i and B_j are disjoint, which occurs when $|j - i| \geq K$. Therefore all elements A_{ij} of A are zero outside a diagonal band of width $2K - 1$. This fact can be exploited by the equation solver. One could for example write a tailored Gauss elimination or Cholesky decomposition.

2.2 Constructing A

The computational time required to *construct* the matrix A may also be considerable when m is large. In fact, if m is, for example, of the order of 10^6 while n is of the order of 10, it could take considerably more CPU time to compute the elements A_{ij} than to solve equation (5). The bottleneck in this case is the construction of G and so we should try to minimize the computational cost of constructing G. An obvious way of computing the elements of G is to compute each G_{ij} in turn, but it is much more efficient to process each x_k in turn, compute all the B-splines whose supports contain it, applying the Cox de Boor algorithm (6) just once, and then add the contribution $B_i(x_k)B_j(x_k)$ to the current value of G_{ij}.

3 Bivariate data

Since the method does not depend on any particular ordering of the points $x_1, \ldots, x_m$, it is quite easy to generalize it to the approximation of bivariate 'scattered data'. Suppose we have distinct points $\mathbf{x}_1, \ldots, \mathbf{x}_m$ in $\mathbb{R}^2$, where $\mathbf{x}_k = (x_k, y_k)$, and associated values $z_1, \ldots, z_m$. We could approximate this data in a least squares sense by a tensor-product spline

$$s(x, y) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} B_i(x) C_j(y) c_{ij}, \tag{7}$$

where $B_1, \ldots, B_{n_1}$ are B-splines defined on an interval $[a_1, b_1]$ contained the x_k and $C_1, \ldots, C_{n_2}$ are B-splines over an interval $[a_2, b_2]$ containing the y_k.

Defining

$$c = (c_{1,1}, \ldots, c_{n_1,1}, c_{1,2}, \ldots, c_{n_1,n_2})^T,$$

the task is to find the coefficient vector c, of length $n = n_1 n_2$, which minimizes

$$F(c) = \sum_{k=1}^{m} (s(x_k, y_k) - z_k)^2 + \lambda c^T E c,$$

for some symmetric, positive semi-definite $n \times n$ matrix E. Similar to the univariate case, a minimum of F occurs when c is the solution of the normal equations (5) where now B is the $m \times n$ matrix

$$B = \begin{bmatrix} B_1(x_1)C_1(y_1) & B_2(x_1)C_1(y_1) & \ldots & B_{n_1}(x_1)C_{n_2}(y_1) \\ \vdots & \vdots & & \vdots \\ B_1(x_m)C_1(y_m) & B_2(x_m)C_1(y_m) & \ldots & B_{n_1}(x_m)C_{n_2}(y_m) \end{bmatrix},$$

so

$$G_{(j-1)n_1+i,(s-1)n_1+r} = \sum_{k=1}^{m} B_i(x_k) C_j(y_k) B_r(x_k) C_s(y_k)$$

for $i, r = 1, \ldots, n_1$ and $j, s = 1, \ldots, n_2$.

For the smoothing term we could take the thin plate spline energy

$$J(c) = \int_{a_1}^{b_1} \int_{a_2}^{b_2} (s_{xx}^2 + 2s_{xy}^2 + s_{yy}^2) dy\, dx, \tag{8}$$

which, after substitution into the definition of s in (7) can be expressed as

$$\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{r=1}^{n_1}\sum_{s=1}^{n_2} E_{ijrs}c_{ij}c_{rs},$$

where

$$E_{ijrs} = A_{ijrs} + 2B_{ijrs} + C_{ijrs}$$

and

$$A_{ijrs} = \int_{a_1}^{b_1} B_i''(x)B_r''(x)dx \int_{a_2}^{b_2} C_j(y)C_s(y)dy,$$

$$B_{ijrs} = \int_{a_1}^{b_1} B_i'(x)B_r'(x)dx \int_{a_2}^{b_2} C_j'(y)C_s'(y)dy,$$

$$C_{ijrs} = \int_{a_1}^{b_1} B_i(x)B_r(x)dx \int_{a_2}^{b_2} C_j''(y)C_s''(y)dy.$$

Similar to the univariate case, the resulting matrix A is by definition symmetric and positive semi-definite. An argument similar to the univariate case shows that A is also positive definite if the data set contains any *three* points $\mathbf{x}_k$ that are *not collinear* (because if $J = 0$, s must be linear). Therefore, there will be a unique minimizer for F for any 'reasonable' set of scattered data: for any set of points $\mathbf{x}_k$ that do not all lie on a straight line.

If the functions B_i and C_j are B-splines of orders K and L respectively then as in the univariate case, the matrices G and E are sparse and therefore A is sparse. In fact in this case

$$A_{(j-1)n_1+i,(s-1)n_1+r} = 0$$

if either $|i - r| \geq K$ or $|j - s| \geq L$. Therefore, there are at most $(2K - 1) \times (2L - 1)$ non-zero elements in each row of A. The non-zero elements of A are shown in Figure 1, for the values $n_1 = n_2 = 20$ when the spline $s(x, y)$ is bicubic ($K = L = 4$). An iterative method like the conjugate gradient method works well with a sparse matrix such as A.

4 A numerical example

A data set of 10,000 points $\mathbf{z}_k = (x_k, y_k, z_k)$ is shown in Figure 2. These points were parameterized, giving corresponding parameter points $\mathbf{u}_k = (u_k, v_k)$.

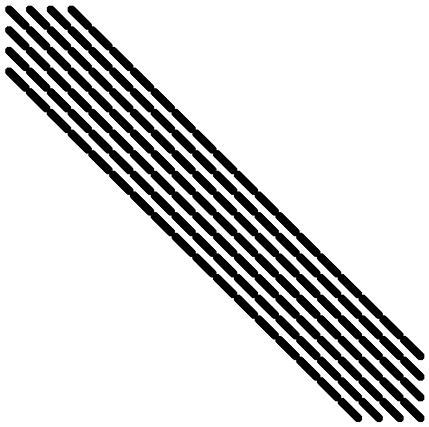

Figure 1: Structure of matrix A when $K = L = 4$ and $n_1 = n_2 = 20$.

The least squares method was then applied to the parametrized points. Specifically, the method was used to find

$$s_r(u,v) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} B_i(u) C_j(v) c_{ij}^r, \qquad r = 1,2,3,$$

such that

$$s_1(\mathbf{u}_k) \approx x_k, \qquad s_2(\mathbf{u}_k) \approx y_k, \qquad s_3(\mathbf{u}_k) \approx z_k,$$

yielding a parametric spline surface

$$\mathbf{s}(\mathbf{u}) = (s_1(\mathbf{u}), s_2(\mathbf{u}), s_3(\mathbf{u}))$$

such that $\mathbf{s}(\mathbf{u}_k) \approx \mathbf{z}_k$.

The spline surface was chosen to be bicubic, i.e. $K = L = 4$, and to have uniform knot vectors with $n_1 = n_2 = 100$. The resulting surface is shown shaded in Figure 3 and via isocurves in Figure 4. The conjugate gradient method was used to solve the three linear systems.

The parameter λ was chosen to be

$$\lambda = \|B^T B\|_F / \|E\|_F,$$

where the $\|M\|_F$ denotes the Frobenius norm of a matrix M, $\sqrt{(\sum_{ij} M_{ij}^2)}$. The effect of this heuristic is roughly speaking to ensure that the two contributions $B^T B$ and λE to the matrix A have equal weight. This choice seems to perform well in examples.

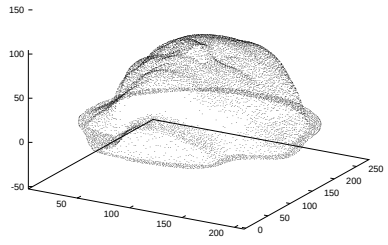

Figure 2: Data set of $m = 10,000$ points.

Figure 3: Approximate surface $\mathbf{s}(\mathbf{u})$.

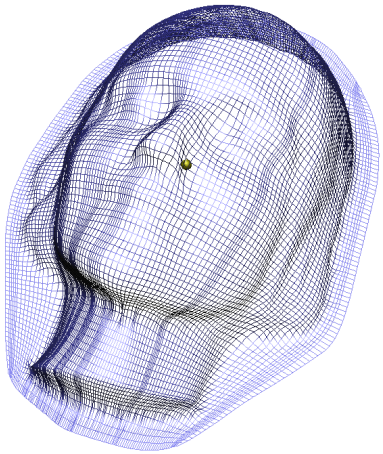

Figure 4: Isocurves of the surface $\mathbf{s}(\mathbf{u})$.