LEAST SQUARE DATA FITTING

Consider that you are given a set of data points

$$(t_i, b_i), \quad i = 1, ..., m$$

with m a reasonably large number. We want to find a function $p(t)$, perhaps a polynomial, whose graph goes "as close as possible" to these data points. How do we choose $p(t)$?

For example, suppose $p(t)$ is a cubic polynomial:

$$p(t) = x_1 + x_2 t + x_3 t^2 + x_4 t^3$$

How do we choose the coefficients $\left\{x_1, ..., x_4\right\}$? What we would like to have be true is that

$$p(t_i) = b_i, \quad i = 1, ..., m$$

In our case with $p(t)$ a cubic polynomial, this says

$$\sum_{j=1}^{4} x_j t_i^{j-1} = b_i, \quad i = 1, ..., m$$

This is an overdetermined system of linear equations; and in general, it does not have an exact solution. Therefore, how do we choose an approximate solution?

In general, the fitting function $p(t) \equiv p(t; x)$ is going to depend on a set of parameters $\left\{x_j\right\}$ which we need to determine. We can ask to find:

$$(a) \quad \min_{\{x_j\}} \quad \max_i |p(t_i) - b_i|$$

$$(b) \quad \min_{\{x_j\}} \quad \sum_{m=1}^{m} |p(t_i) - b_i|$$

$$(c) \quad \min_{\{x_j\}} \quad \sum_{m=1}^{m} [p(t_i) - b_i]^2$$

For reasons coming from statistics (regression analysis), it makes sense to look at the option (c). The resulting approximation $p(t)$ is called the *least squares approximation* to the given data $\{(t_i, b_i) \mid 1 \leq i \leq m\}$. It is very widely used in applications, and finding it is an important problem.

The approximation arising from (*a*) is called the Chebyshev approximation; and that arising from (*b*) is called the best ℓ^1-approximation.

Introduce the *root mean square error*

$$E(x) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left[b_i - p\left(t_i; x\right)\right]^2}$$

for the approximation of the data $\{(t_i, b_i)\}$ by the approximant $p(t; x)$. The *least squares approximation* minimizes this quantity:

$$E(x^*) = \min_{x\in\mathbb{R}^n} E(x)$$

For our linear system

$$\sum_{j=1}^{4} x_j t_i^{j-1} = b_i, \quad i = 1, ..., m$$

we want to minimize

$$\sum_{i=1}^{m}\left[b_i - \sum_{j=1}^{4} x_j t_i^{j-1}\right]^2$$

THE GENERAL MODEL

For our model $p(t)$, we assume it is of the form

$$p(t) \equiv p(t; x) = \sum_{j=1}^{n} x_j \varphi_j(t)$$

where $\{\varphi_1(t), ..., \varphi_n(t)\}$ are independent functions over an interval containing the nodes $\{t_i\}$. We want to have

$$\sum_{j=1}^{n} x_j \varphi_j(t_i) = b_i, \quad i = 1, ..., m$$

Write this linear system as

$$Ax = b$$

In general, it does not have an exact solution.

The quantity $E(x)$ can be written as

$$E(x) = \frac{1}{\text{sqrt}(m)} \|Ax - b\|_2$$

and thus we wish to find a vector $x \in \mathbb{R}^n$ for which $\|Ax - b\|_2$ is minimized.

THEORETICAL SOLUTION

Let A be an $m \times n$ matrix with $m > n$. We want to minimize

$$\|Ax - b\|_2$$

We begin by looking at the singular value decomposition of A:

$$A = VFU^T$$

with U and V orthogonal matrices, and

$$F = \begin{bmatrix} \mu_1 & 0 & & \cdots & & 0 \\ 0 & \ddots & & & & \\ \vdots & & \mu_r & & & \vdots \\ & & & 0 & & \\ & & & & \ddots & \\ 0 & & \cdots & & & 0 \\ \vdots & & & & & \vdots \\ 0 & & \cdots & & & 0 \end{bmatrix}$$

The numbers μ_i are all real, with

$$\mu_1 \geq \mu_2 \geq \cdots \geq \mu_r > 0$$

The number r is the rank of A, and $r \leq n$.

Using $A = VFU^T$,

$$\begin{aligned} \|Ax - b\|_2 &= \left\|VFU^T x - b\right\|_2 \\ &= \left\|V\left(FU^T x - V^T b\right)\right\|_2 \end{aligned}$$

Recall that for the Euclidean norm, with V an $m \times m$ orthogonal matrix and $w \in \mathbb{R}^m$,

$$\begin{aligned} \|Vw\|_2 &= \text{sqrt}\left[(Vw)^T (Vw)\right] \\ &= \text{sqrt}\left[w^T V^T V w\right] \\ &= \text{sqrt}\left[w^T w\right] = \|w\|_2 \end{aligned}$$

Therefore,

$$\begin{aligned} \|Ax - b\|_2 &= \left\|FU^T x - V^T b\right\|_2 \\ &= \|Fz - c\|_2 \end{aligned}$$

with $z = U^T x$ and $c = V^T b$. If we look at the special form of F, we obtain

$$\|Ax - b\|_2 = \sqrt{\sum_{j=1}^{r} \left(\mu_j z_j - c_j\right)^2 + \sum_{j=r+1}^{m} c_j^2}$$

Thus the minimum of this is obtained by letting

$$\mu_j z_j - c_j = 0, \quad j = 1, ..., r$$

The solution is given by

$$z_j^* = \frac{c_j}{\mu_j}, \quad j = 1, ..., r$$

If $r < n$, there is no restriction on $z_{j+1}, ..., z_n$. Choose $z_{j+1}^*, ..., z_n^*$ arbitrarily, and then we have a minimizer z^*.

Then $x^* = Uz^*$ is the minimizer of the original expression $\|Ax - b\|_2$. Also,

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2 = \|Ax^* - b\|_2 = \sqrt{\sum_{j=r+1}^{m} c_j^2}$$

We often choose the solution of minimum Euclidean norm. Since

$$\|x^*\|_2 = \|Uz^*\|_2 = \|z^*\|_2 = \text{sqrt}\left[\sum_{j=1}^{n} \left|z_j^*\right|^2\right]$$

we minimize $\|x^*\|_2$ by letting $z_{j+1}^*, ..., z_n^* = 0$,

$$\|x^*\|_2 = \text{sqrt}\left[\sum_{j=1}^{r} \left(\frac{c_j}{\mu_j}\right)^2\right]$$

Then our minimizer is

$$x^* = Uz^* = U\left[\frac{c_1}{\mu_1}, \cdots, \frac{c_r}{\mu_r}, 0, ..., 0\right]^T$$

THE GENERALIZED INVERSE

Introduce

$$F^{\dagger} = \begin{bmatrix} \mu_1^{-1} & 0 & \cdots & 0 & \cdots & & 0 \\ 0 & \ddots & & \vdots & & & \\ & & \mu_r^{-1} & & & & \vdots \\ \vdots & & & 0 & & & \\ & & & & \ddots & & \\ 0 & & \cdots & & & 0 & 0 \end{bmatrix}$$

and

$$A^{\dagger} = UF^{\dagger}V^T$$

Then

$$x^* = A^{\dagger}b$$

To see this

$$\begin{aligned} A^{\dagger}b &= UF^{\dagger}V^Tb \\ &= UF^{\dagger}c \\ &= U\left[\frac{c_1}{\mu 1}, \cdots, \frac{c_r}{\mu r}, 0, ..., 0\right]^T \\ &= Uz^* = x^* \end{aligned}$$

The matrix $A^\dagger$ is called the *generalized inverse* or *pseudoinverse* of A; and when A is a square non-singular matrix, $A^\dagger = A^{-1}$. To see the motivation for the definition of $A^\dagger$, use $A = VFU^T$ to obtain

$$A^{-1} = \left(VFU^T\right)^{-1} = UF^{-1}V^T$$

provided F^{-1} exists. Also note that $V^{-1} = V^T$ and $U^{-1} = U^T$.

The vector $x = A^\dagger b$ is the minimizer of $\|Ax - b\|_2$ of minimum norm. Perhaps surprisingly, it is a linear function of b.

Properties of $A^\dagger$ are given in problems 27-29 at the end of the chapter. For example,

$$AA^\dagger A = A \qquad \left(AA^\dagger\right)^T = AA^\dagger$$
$$A^\dagger AA^\dagger = A^\dagger \qquad \left(A^\dagger A\right)^T = A^\dagger A$$

$$\lim_{\alpha \to 0+} \left(\alpha I + A^T A\right)^{-1} A^T = A^\dagger$$

This last property is tied into the least squares solution process, to which we now return.

THE LEAST SQUARES MODEL

Recall that for our model $p(t)$, we assume it is of the form

$$p(t) \equiv p(t; x) = \sum_{j=1}^{n} x_j \varphi_j(t)$$

where $\{\varphi_1(t), ..., \varphi_n(t)\}$ are independent functions over an interval containing the nodes $\{t_i\}$. We want to have

$$\sum_{j=1}^{n} x_j \varphi_j(t_i) = b_i, \quad i = 1, ..., m$$

and we write this linear system as

$$Ax = b$$

with

$$A_{i,j} = \varphi_j(t_i), \quad 1 \le i \le m, \; 1 \le j \le n$$

Then

$$\|Ax - b\|_2^2 = \sum_{i=1}^{m} \left[\sum_{j=1}^{n} x_j \varphi_j(t_i) - b_i \right]^2$$

To minimize this, we can form the partial derivatives with respect to each of $x_1, ..., x_n$ and set them to zero:

$$\frac{\partial}{\partial x_k} \sum_{i=1}^{m} \left[\sum_{j=1}^{n} x_j \varphi_j(t_i) - b_i \right]^2 = 0$$

$$2 \sum_{i=1}^{m} \left[\sum_{j=1}^{n} x_j \varphi_j(t_i) - b_i \right] \varphi_k(t_i) = 0$$

$$\sum_{i=1}^{m} \left[\sum_{j=1}^{n} x_j \varphi_j(t_i) \right] \varphi_k(t_i) = \sum_{i=1}^{m} b_i \varphi_k(t_i)$$

for $k = 1, ..., n$. If you look at this carefully, you will see this is exactly the same as

$$A^T A x = A^T b$$

These are called the *normal equations* for solving $Ax = b$, and historically they have been the means by which the least squares problem has been solved.

Recall fitting the data $\{(t_i, b_i)\}$ with

$$p(t) = \sum_{j=1}^{n} x_j t^{j-1}$$

The system $Ax = b$ to be solved is

$$\sum_{j=1}^{n} x_j t_i^{j-1} = b_i, \quad i = 1, ..., m$$

The normal equations become

$$\sum_{j=1}^{n} x_j \left(\sum_{i=1}^{m} t_i^{j+k-2} \right) = \sum_{i=1}^{m} t_i^{k-1} b_i, \quad k = 1, ..., n$$

To see this is ill-conditioned, imagine the nodes $\{t_i\}$ as being uniformly distributed in an interval, say $[0, 1]$. Then

$$\left(\sum_{i=1}^{m} t_i^{j+k-2} \right) \approx m \int_0^1 t^{j+k-2} dt = \frac{m}{j+k-1}$$

for $j, k = 1, ..., n$. Then the coefficient matrix of the above, namely $A^T A$, looks like mH_n, a multiple of the Hilbert matrix of order n.

EXAMPLE

From the data in Table 9.3 on page 639, with $m = 21$ data points,

$$A^T A = \begin{bmatrix} 21 & 10.5 & 7.175 & 5.125 \\ 10.5 & 7.175 & 5.125 & 4.51666 \\ 7.175 & 5.125 & 4.51666 & 3.85416 \\ 5.125 & 4.51666 & 3.85416 & 3.38212 \end{bmatrix}$$

This is approximately $21H_4$; and

$$\text{cond}(A^T A)_2 = 12105$$

The linear system $A^T Ax = A^T b$ has the solution

$$x^* = [.5747,\ 4.7259,\ -11.1282,\ 7.6687]^T$$

Now perturb $A^T b$ by

$$[.01,\ -.01,\ .01,\ -.01]^T$$

which is consistent with experimental error in the data b. The new solution is

$$\widehat{x}^* = [.7408,\ 2.6825,\ -6.1538,\ 4.4550]^T$$

In choosing a set of basis functions $\left\{\varphi_j\right\}$ for our model

$$p(t) = \sum_{j=1}^{n} x_j \varphi_j(t)$$

consider the matrix elements of $A^T A$:

$$\left(A^T A\right)_{j,k} = \sum_{i=1}^{m} \varphi_j(t_i)\varphi_k(t_i) \approx \frac{m}{b-a} \int_a^b \varphi_j(t)\varphi_k(t)\, dt$$

if the nodes are uniformly distributed in $[a, b]$. Then one way to make the matrix well-conditioned is to choose functions $\left\{\varphi_j\right\}$ that are orthogonal or nearly orthogonal on $[a, b]$. This leads to using as a basis the Legendre polynomials with respect to $[a, b]$. Thus $\varphi_j(t) = P_{j-1}(t)$, the Legendre polynomial of degree $j - 1$.

The above example of fitting the data with a cubic polynomial is repeated with the Legendre polynomials as the basis functions, on page 642. In this case, the condition number of $A^T A$ is 1.58, indicating a much better conditioned coefficient matrix.

If you do not want to worry about choosing a good basis, then you need to use the singular value decomposition and the generalized inverse as the means of solving $Ax = b$ approximately,

$$x^* = A^\dagger b$$

$$\|Ax^* - b\|_2 = \sqrt{\sum_{j=r+1}^{m} c_j^2}$$

There is often a question as to the degree we should be using. Generally, we would look at $\|Ax^* - b\|_2$, and when it became sufficiently small, we would know we had found approximately the correct degree. Usually it will drop gradually until a "right degree" is found, at which point it will decrease steeply to something much smaller.

In Matlab, use $svd(A)$ and $pinv(A)$ for computing the singular value decomposition and the pseudoinverse of a matrix A, respectively.

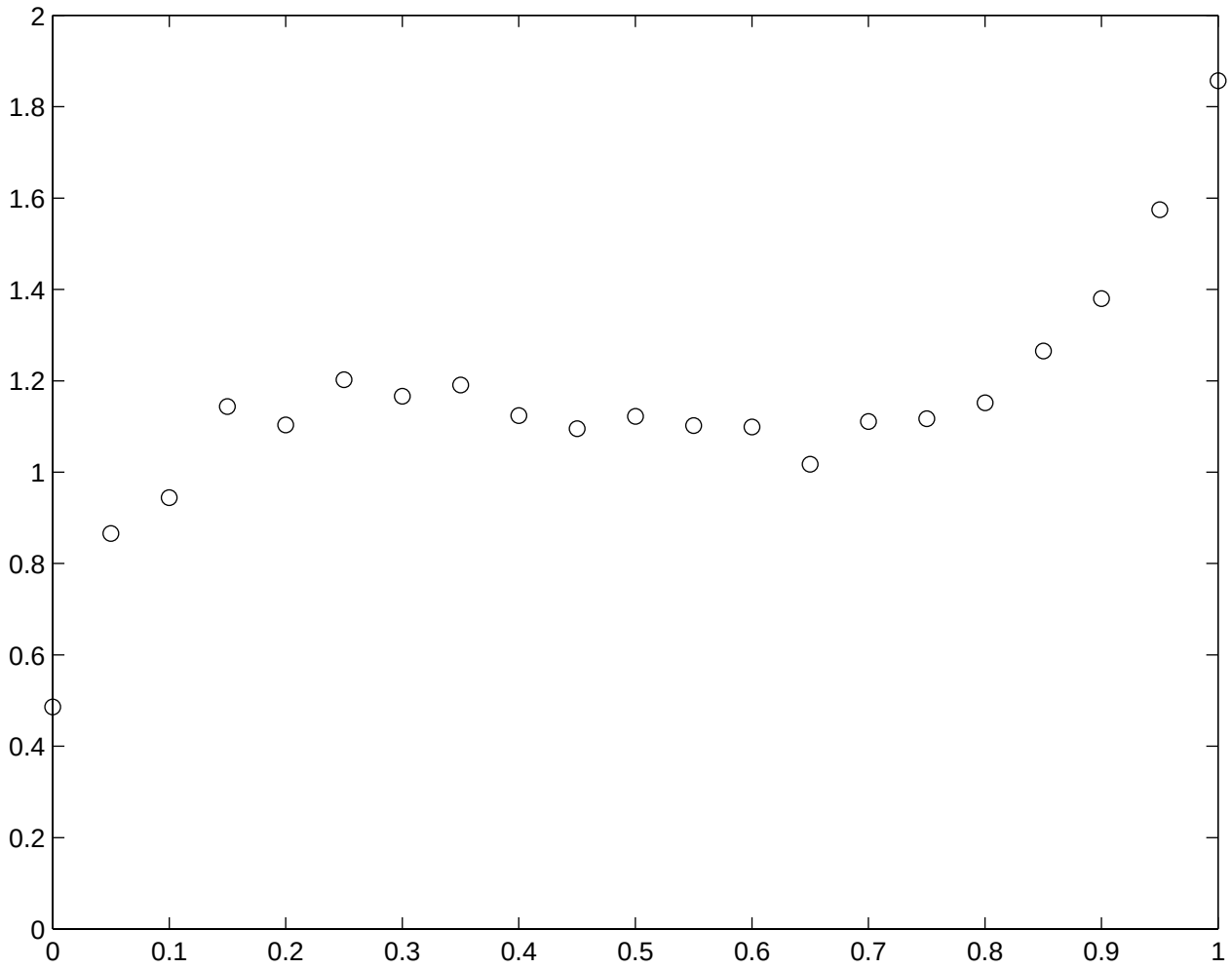

Figure 1: Data points to be fit by a cubic polynomial