2.3 Nonlinear regression: polynomials and linear combinations

Let's now consider the problem of finding a polynomial of degree m that gives the best least-squares fit. As before, let $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ be given data points. We assume that $m < n-1$, for otherwise, we can use polynomial interpolation to fit the points exactly. The function we will fit to is a polynomial of the form

$$f_m(x) = \sum_{j=0}^{m} a_j x^j = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m,$$

so the coefficients to find are a_0, a_1, a_2, …, a_m.

As in the linear case, the residuals are defined similarly for each $i \leqslant n$:

$$r_i = y_i - f_m(x_i) = y_i - \sum_{j=0}^{m} a_j x^j,$$

and we wish to minimise the sum of squared residuals. The objective function is

$$F(\mathbf{a}) = \sum_{i=1}^{n} (f_m(x_i) - y_i)^2 = \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{m} a_j x_i^j \right)^2 .$$

Differentiating this function with respect to a_k (for each $k \in \{0, 1, \dots, m+1\}$) yields

$$\frac{\partial F}{\partial a_k} = \sum_{i=1}^{n} -2x_i^k \left(y_i - \sum_{j=0}^{m} a_j x_i^j \right) .$$

The FONC implies each of these partial derivatives should equal zero. So, for each $k \in \{0, 1, \dots, m+1\}$, we have:

$$\begin{aligned}
\sum_{i=1}^{n} -2x_i^k \left(y_i - \sum_{j=0}^{m} a_j x_i^j \right) = 0 \iff & \sum_{i=1}^{n} x_i^k \left(y_i - \sum_{j=0}^{m} a_j x_i^j \right) = 0 \\
\iff & \sum_{i=1}^{n} y_i x_i^k - \sum_{i=1}^{n} x_i^k \sum_{j=0}^{m} a_j x_i^j = 0 \\
\iff & \sum_{i=1}^{n} y_i x_i^k = \sum_{i=1}^{n} \sum_{j=0}^{m} a_j x_i^{j+k} \\
\iff & \sum_{i=1}^{n} y_i x_i^k = \sum_{j=0}^{m} \left(a_j \sum_{i=1}^{n} x_i^{j+k} \right) .
\end{aligned}$$

This yields a system of $m+1$ equations, linear in a_0, a_1, …, a_m:

$$\sum_{j=0}^{m} \left(a_j \sum_{i=1}^{n} x_i^{j+k} \right) = \sum_{i=1}^{n} y_i x_i^k, \quad k \in \{0, \dots, m\},$$

which are the **normal equations** for the polynomial. Note that the case $m = 1$ corresponds to the normal equations used for linear regression in Section 2.1. Solving the normal equations will give the coefficients a_k of the polynomial $f_m(x)$.

Example: normal equations for a quadratic

Here we will explicitly write the normal equations for the case of a quadratic fit, so that $m = 2$. In other words, suppose we have n data points and we are fitting a function of the form

$$f(x) = a_0 + a_1 x + a_2 x^2.$$

There will be $m + 1 = 3$ normal equations. For $k = 0$, we get:

$$\begin{aligned}
\sum_{j=0}^{2}\left(a_j \sum_{i=1}^{n} x_i^{j+0}\right) = \sum_{i=1}^{n} y_i x_i^0 &\iff \sum_{j=0}^{2}\left(a_j \sum_{i=1}^{n} x_i^{j}\right) = \sum_{i=1}^{n} y_i \\
&\iff a_0 \sum_{i=1}^{n} x_i^0 + a_1 \sum_{i=1}^{n} x_i^1 + a_2 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i \\
&\iff n a_0 + a_1 \sum_{i=1}^{n} x_i + a_2 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i
\end{aligned}$$

For $k = 1$, we get:

$$\begin{aligned}
\sum_{j=0}^{2}\left(a_j \sum_{i=1}^{n} x_i^{j+1}\right) = \sum_{i=1}^{n} y_i x_i^1 &\iff a_0 \sum_{i=1}^{n} x_i^{0+1} + a_1 \sum_{i=1}^{n} x_i^{1+1} + a_2 \sum_{i=1}^{n} x_i^{2+1} = \sum_{i=1}^{n} y_i x_i \\
&\iff a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + a_2 \sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} y_i x_i
\end{aligned}$$

For $k = 2$, we get:

$$\begin{aligned}
\sum_{j=0}^{2}\left(j \sum_{i=1}^{n} x_i^{j+2}\right) = \sum_{i=1}^{n} y_i x_i^2 &\iff a_0 \sum_{i=1}^{n} x_i^{0+2} + a_1 \sum_{i=1}^{n} x_i^{1+2} + a_2 \sum_{i=1}^{n} x_i^{2+2} = \sum_{i=1}^{n} y_i x_i^2 \\
&\iff a_0 \sum_{i=1}^{n} x_i^2 + a_1 \sum_{i=1}^{n} x_i^3 + a_2 \sum_{i=1}^{n} x_i^4 = \sum_{i=1}^{n} y_i x_i^2
\end{aligned}$$

Notice that the coefficients of each a_k involve sums of powers of the x_i, while the right-hand side of each equation involves multiplying the y_i by increasing powers of x_i.

The previous example shows that an explicit construction of the normal equations is a little unwieldy. However, as in the linear case, the normal equations can be written in matrix-vector form as follows:

$$\mathbf{A}^T\mathbf{A}\mathbf{a} = \mathbf{A}^T\mathbf{y},$$

where

$$\mathbf{A} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{pmatrix}, \quad \mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

Thus, assuming $\mathbf{A}^T\mathbf{A}$ is invertible, the coefficient vector $\mathbf{a}$ can be found using a single matrix product:

$$\mathbf{a} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$$

Example: fitting a quadratic function

We wish to fit the quadratic function $f(x) = a_0 + a_1 x + a_2 x^2$ to the data below:

i	1	2	3	4	5	6	7
x_i	0.464	0.698	0.968	2.179	2.987	3.038	3.565
y_i	1.946	2.086	3.674	3.979	4.606	5.611	6.367

The required matrices are

$$\mathbf{A} = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{pmatrix}, \quad \mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_7 \end{pmatrix},$$

Calculating the coefficients by hand is not a worthwhile task, so using the following MATLAB code we determine $\mathbf{a} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$

```
% Store x and y as column matrices
x = [0.464 0.698 0.968 2.179 2.987 3.038 3.565]';
y = [1.946 2.086 3.674 3.979 4.606 5.611 6.367]';
% Initialise the matrix A (as a list of columns)
A = [ones([7,1]) x x.^2];
% Compute the coefficients
a = inv(A'*A)*A'*y
```

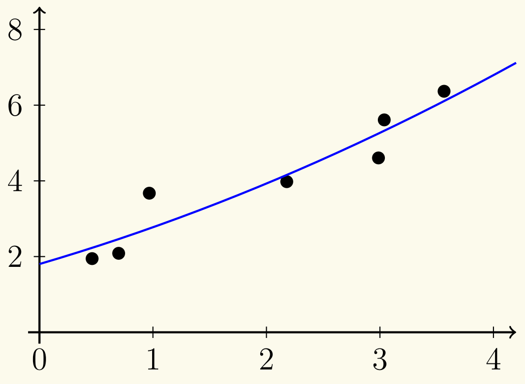
To four decimal places, this gives

$$\mathbf{a} = \begin{pmatrix} 1.8026 \\ 0.8798 \\ 0.0919 \end{pmatrix},$$

so the quadratic function we obtain is

$$f(x) = 1.8026 + 0.8798x + 0.0919x^2.$$

This is plotted below.



The method above can be employed when the function we seek to fit is a linear combination of arbitrary functions; i.e., a function of the form

$$y = f(x) = a_0\phi_0(x) + a_1\phi_1(x) + \cdots a_m\phi_m(x),$$

where the $\phi_k(x)$, $0 \leqslant k \leqslant m$, are given functions of x.

The normal equations can be used to compute the coefficients of the linear combination of functions that best fits the data in the least-squares sense, provided that these functions are linearly independent. This generalises the polynomial fit from above, by taking $\phi_k(x) = x^k$, $0 \leqslant k \leqslant m$. The entries of the $n \times m$ matrix $\mathbf{A}$ are given by

$$\mathbf{A} = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_m(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_m(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \cdots & \phi_m(x_n) \end{pmatrix}, \quad \text{and } \mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

Example: a linear combination of functions

Consider the same set of data as earlier:

i	1	2	3	4	5	6	7
x_i	0.464	0.698	0.968	2.179	2.987	3.038	3.565
y_i	1.946	2.086	3.674	3.979	4.606	5.611	6.367

Suppose we had reason to believe the data in the table would be best approximated by a function of the form

$$f(x) = a_0 e^x + a_1 \sin(x) + a_2 x^3.$$

The matrix $\mathbf{A}$ is constructed by applying each function elementwise to each x_i. Using MATLAB:

```
% Store x and y as column matrices
x = [0.464 0.698 0.968 2.179 2.987 3.038 3.565]';
y = [1.946 2.086 3.674 3.979 4.606 5.611 6.367]';
% Initialise the matrix A (as a list of columns)
A = [exp(x) sin(x) x.^3];
% Compute the coefficients
a = inv(A'*A)*A'*y
```

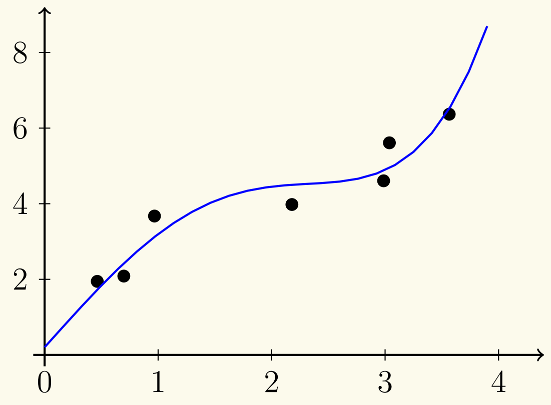
We obtain, to four decimal places,

$$\mathbf{a} = \begin{pmatrix} 0.2035 \\ 3.1240 \\ 0.0134 \end{pmatrix},$$

so the function we obtain is

$$f(x) = 0.2035 e^x + 3.1240 \sin(x) + 0.0134 x^3.$$

This is plotted below.



2.4 Linearisation

In some cases, least-squares fitting can also be used to fit data with functions that are not linear combinations of functions. Suppose we believe that some given data points can best be fitted by an exponential function of the form

$$y = ae^{bx},$$

where the constants a and b are to be determined. The box below demonstrates that a direct approach using the previous methods is insufficient.

Example: an attempt to determine the coefficients

To see why the method from earlier fails, we will try to fit the function $f(x) = ae^{bx}$ to just three data points: $(1, 4)$, $(2, 5)$ and $(3, 6)$. The residuals are then

$$\begin{aligned} r_1 &= 4 - ae^{b}, \\ r_2 &= 5 - ae^{2b}, \\ r_3 &= 6 - ae^{3b}, \end{aligned}$$

and so the objective function to minimise is

$$F(a, b) = (4 - ae^{b})^2 + (5 - ae^{2b})^2 + (6 - ae^{3b})^2.$$

The partial derivatives are

$$\begin{aligned} \frac{\partial F}{\partial a} &= -2e^{b}(4 - ae^{b}) - 2e^{2b}(5 - ae^{2b}) - 2e^{3b}(6 - ae^{3b}) \\ &= 2ae^{2b} + 2ae^{4b} + 2ae^{6b} - 8e^{b} - 10e^{2b} - 12e^{3b} \\ &= 2e^{b}\left((a-5)e^{b} + ae^{3b} + ae^{5b} - 6e^{2b} - 4\right), \\ \frac{\partial F}{\partial b} &= -2ae^{b}(4 - ae^{b}) - 4ae^{2b}(5 - ae^{2b}) - 6ae^{3b}(6 - ae^{3b}) \\ &= 2a^2e^{2b} + 4a^2e^{4b} + 6a^2e^{6b} - 8ae^{b} - 20ae^{2b} - 36ae^{3b} \\ &= 2ae^{b}\left((a-10)e^{b} + 2ae^{3b} + 3ae^{5b} - 18e^{2b} - 4\right). \end{aligned}$$

Setting these equal to zero now amounts to solving the system of equations

$$\begin{aligned} (a-5)e^{b} + ae^{3b} + ae^{5b} - 6e^{2b} &= 4, \\ (a-10)e^{b} + 2ae^{3b} + 3ae^{5b} - 18e^{2b} &= 4. \end{aligned}$$

Note that these are not linear in terms of the unknown coefficients a and b (due to the appearance of e^{b}, e^{2b}, etc.). Since they are non-linear, the matrix methods of the previous section cannot be applied.

We will not attempt to solve this algebraically; the next chapter deals with iterative methods to approximate these solutions (which can be used to show that $a \approx 3.307$ and $b \approx 0.200$).

The linearisation method works by transforming the expression $y = ae^{bx}$ into an expression that *is* a linear combination of functions. In this case, by taking the natural logarithm of both sides of this equation we get

$$\ln(y) = \ln(a) + bx.$$

If we define $z = \ln(y)$ and $c = \ln(a)$, this expression is

$$z = c + bx,$$

which now transforms the original problem into the problem of fitting the data points (x_i, z_i), where $z_i = \ln(y_i)$, with a linear function $f(x) = c + ax$.

Determining the original coefficients a and b can then be done algebraically.

Example: linearisation for exponential functions

Consider the data points

i	1	2	3	4	5	6	7
x_i	0.464	0.698	0.968	2.179	2.987	3.038	3.565
y_i	1.946	2.086	3.674	3.979	4.606	5.611	6.367

We wish to fit a function of the form $y = ae^{bx}$.

Linearising gives $\ln(y) = \ln(a) + bx$. So, with $z_i = \ln(y_i)$ and $c = \ln(a)$, we are fitting the points (x_i, z_i) to $f(x) = c + bx$.

Computing $z_i = \ln(y_i)$ is simple in MATLAB:

```
% Store x and y as column matrices
x = [0.464 0.698 0.968 2.179 2.987 3.038 3.565]';
y = [1.946 2.086 3.674 3.979 4.606 5.611 6.367]';
% Compute z = ln(y)
z = log(y)
```

This gives us

i	1	2	3	4	5	6	7
x_i	0.464	0.698	0.968	2.179	2.987	3.038	3.565
z_i	0.6658	0.7352	1.3013	1.3810	1.5274	1.7247	1.8511

We can now compute the coefficients exactly the same as we have done for the linear case:

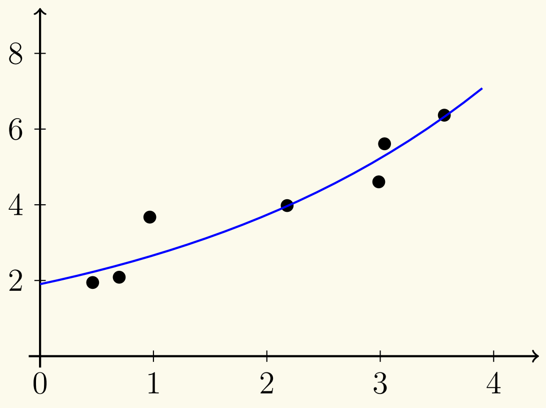
```
% Initialise the matrix A (as a list of columns)
A = [ones([7,1]) x]
% Compute the coefficients
a = inv(A'*A)*A'*z
```

To four decimal places, this gives $c = 0.6434$ and $b = 0.3369$.

Since $c = \ln(a)$, we get $a = e^c = 1.9029$, so the function we obtain is

$$f(x) = 1.9029e^{0.3369x}.$$

This is plotted below.



Similarly, suppose the given data is believed to approximately conform to a function of the form

$$y = ax^b.$$

Contrast this with the polynomial method from Section 2.3: in that case, the degree of the polynomial was decided in advance. In this case, the power is unknown, and we wish to optimise for it.

Taking the natural logarithm of both sides of this equation yields

$$\ln(y) = \ln(a) + b\ln(x).$$

In this case, we let $u_i = \ln(x_i)$ and $z_i = \ln(y_i)$, so that the new problem is to fit the data (u_i, z_i) to the function $f(u) = c + bu$.

Example: linearisation for power functions

Consider the data points

i	1	2	3	4	5	6	7
x_i	0.464	0.698	0.968	2.179	2.987	3.038	3.565
y_i	1.946	2.086	3.674	3.979	4.606	5.611	6.367

We wish to fit a function of the form $y = ax^b$.

Linearising gives $\ln(y) = \ln(a) + b\ln(x)$. So, with $u_i = \ln(x_i)$, $z_i = \ln(y_i)$ and $c = \ln(a)$, we are fitting the points (u_i, z_i) to $f(u) = c + bu$.

We compute the coefficients using MATLAB:

```
% Store x and y as column matrices
x = [0.464 0.698 0.968 2.179 2.987 3.038 3.565]';
y = [1.946 2.086 3.674 3.979 4.606 5.611 6.367]';
% Compute u = ln(x) and z = ln(y)
u = log(x)
z = log(y)
% Initialise the matrix A (as a list of columns)
A = [ones([7,1]) u]
% Compute the coefficients
a = inv(A'*A)*A'*z
```

To four decimal places, this gives $c = 1.0765$ and $b = 0.5333$.

Since $c = \ln(a)$, we get $a = e^c = 2.9344$, so the function we obtain is

$$f(x) = 2.9345x^{0.5333}.$$

This is plotted below.