Ordinary least squares

A system of linear equations is considered *overdetermined* if there are more equations than unknown variables. If all equations of an overdetermined system are linearly independent, the system has no exact solution.

A *linear least-squares problem* is the problem of finding an approximate solution to an overdetermined system. It often arises in applications where a theoretical model is fitted to experimental data.

Linear least-squares problem

Consider a linear system

$$A\mathbf{c} = \mathbf{b} \ , \tag{1}$$

where A is an $n \times m$ matrix, $\mathbf{c}$ is an m-component vector of unknown variables and $\mathbf{b}$ is an n-component vector of the right-hand side terms. If the number of equations n is larger than the number of unknowns m, the system is overdetermined and generally has no solution.

However, it is still possible to find an approximate solution—the one where $A\mathbf{c}$ is only approximately equal $\mathbf{b}$—in the sence that the Euclidean norm of the difference between $A\mathbf{c}$ and $\mathbf{b}$ is minimized,

$$\min_{\mathbf{c}} \|A\mathbf{c} - \mathbf{b}\|^2 \ . \tag{2}$$

The problem (2) is called the ordinary least-squares problem and the vector $\mathbf{c}$ that minimizes $\|A\mathbf{c} - \mathbf{b}\|^2$ is called the *least-squares solution*.

Solution via QR-decomposition

The linear least-squares problem can be solved by QR-decomposition. The matrix A is factorized as $A = QR$, where Q is $n \times m$ matrix with orthogonal columns, $Q^T Q = 1$, and R is an $m \times m$ upper triangular matrix. The Euclidean norm $\|A\mathbf{c} - \mathbf{b}\|^2$ can then be rewritten as

$$\|A\mathbf{c} - \mathbf{b}\|^2 = \|QR\mathbf{c} - \mathbf{b}\|^2 = \|R\mathbf{c} - Q^T\mathbf{b}\|^2 + \|(1 - QQ^T)\mathbf{b}\|^2 \geq \|(1 - QQ^T)\mathbf{b}\|^2 \ . \tag{3}$$

The term $\|(1 - QQ^T)\mathbf{b}\|^2$ is independent of the variables $\mathbf{c}$ and can not be reduced by their variations. However, the term $\|R\mathbf{c} - Q^T\mathbf{b}\|^2$ can be reduced down to zero by solving the $m \times m$ system of linear equations

$$R\mathbf{c} - Q^T\mathbf{b} = 0 \ . \tag{4}$$

The system is right-triangular and can be readily solved by back-substitution. Thus the solution to the ordinary least-squares problem (2) is given by the solution of the triangular system (4).

Ordinary least-squares curve fitting

Ordinary least-squares curve fitting is a problem of fitting n (experimental) data points $\{x_i, y_i \pm \Delta y_i\}$, where Δy_i are experimental errors, by a linear combination of m functions $\{f_k(x) \mid k = 1, \ldots, m\}$,

$$F(x) = \sum_{k=1}^{m} c_k f_k(x) \ . \tag{5}$$

The objective of the least-squares fit is to minimize the square deviation, called χ^2, between the fitting function and the experimental data,

$$\chi^2 = \sum_{i=1}^{n} \left(\frac{F(x_i) - y_i}{\Delta y_i} \right)^2 \ . \tag{6}$$

Individual deviations from experimental points are weighted with their inverse errors in order to promote contributions from the more precise measurements.

Minimization of χ^2 with respect to the coefficiendt c_k in (5) is apparently equivalent to the least-squares problem (2) where

$$A_{ik} = \frac{f_k(x_i)}{\Delta y_i}\ ,\ b_i = \frac{y_i}{\Delta y_i}\ . \tag{7}$$

If $QR = A$ is the QR-decomposition of the matrix A, the formal least-squares solution to the fitting problem is

$$\mathbf{c} = R^{-1}Q^T\mathbf{b}\ . \tag{8}$$

However in practice one has too back-substitute the system $R\mathbf{c} = Q^T\mathbf{b}$.

Variances and correlations of fitting parameters

Suppose δy_i is a (small) deviation of the measured value of the physical observable from its exact value. The corresponding deviation δc_k of the fitting coefficient is then given as

$$\delta c_k = \sum_i \frac{\partial c_k}{\partial y_i}\delta y_i\ . \tag{9}$$

In a good experiment the deviations δy_i are statistically independent and distributed normally with the standard deviations Δy_i. The deviations (9) are then also distributed normally with *variances*

$$\langle \delta c_k \delta c_k \rangle = \sum_i \left(\frac{\partial c_k}{\partial y_i}\Delta y_i\right)^2 = \sum_i \left(\frac{\partial c_k}{\partial b_i}\right)^2\ . \tag{10}$$

The standard errors in the fitting coefficients are then given as the square roots of variances,

$$\Delta c_k = \sqrt{\langle \delta c_k \delta c_k \rangle} = \sqrt{\sum_i \left(\frac{\partial c_k}{\partial b_i}\right)^2}\ . \tag{11}$$

The variances are diagonal elements of the *covariance matrix*, Σ, made of *covariances*,

$$\Sigma_{kq} \equiv \langle \delta c_k \delta c_q \rangle = \sum_i \frac{\partial c_k}{\partial b_i}\frac{\partial c_q}{\partial b_i}\ . \tag{12}$$

Covariances $\langle \delta c_k \delta c_q \rangle$ are measures of to what extent the coefficients c_k and c_q change together if the measured values y_i are varied. The normalized covariances,

$$\frac{\langle \delta c_k \delta c_q \rangle}{\sqrt{\langle \delta c_k \delta c_k \rangle \langle \delta c_q \delta c_q \rangle}} \tag{13}$$

are called *correlations*.

Using (12) and (8) the covariance matrix can be calculated as

$$\Sigma = \left(\frac{\partial \mathbf{c}}{\partial \mathbf{b}}\right)\left(\frac{\partial \mathbf{c}}{\partial \mathbf{b}}\right)^T = R^{-1}(R^{-1})^T = (R^TR)^{-1} = (A^TA)^{-1}\ . \tag{14}$$

The square roots of the diagonal elements of this matrix provide the estimates of the errors of the fitting coefficients and the (normalized) off-diagonal elements are the estimates of their correlations.

C++ implementation with armadillo matrices

Table 1: Least squares fit using QR method

```
#include<vector>
#include<functional>
#include<armadillo>
using namespace arma;

void qrdec(mat& A, mat& R);
void qrbak(mat& Q, mat& R, vec& b, vec& x);
void inverse(mat& A,mat& Ainverse);

void lsfit(
        const vec & x, const vec & y, const vec & dy,
        const std::vector<std::function<double(double)>> & funs,
        vec & c, mat & S)
{
int n = x.size(), m=funs.size();
mat A(n,m), R(m,m);
vec b = y/dy;
for(int i=0;i<n;i++)for(int k=0;k<m;k++)
        A(i,k)=funs[k](x[i])/dy[i];
qrdec(A,R);
qrbak(A,R,b,c);
mat Ri(m,m);
inverse(R,Ri);
S = Ri*Ri.t();
}
```