Linear Least Squares, General case

Our fitting function in general case is:

$$F(x) = a_1 f_1(x) + a_2 f_2(x) + ... + a_n f_n(x)$$

Note that the function itself does not have to be linear for the problem to be linear. The fit should be linear in the fitting parameters.

Linear Least Squares, General case

Thus we have: vectors x, y and a:

$$x = \begin{pmatrix} x_1 \\ \dots \\ x_N \end{pmatrix} \text{ where the data was taken points}, \quad y = \begin{pmatrix} y_1 \\ \dots \\ y_N \end{pmatrix} \text{ the data},$$

$$a = \begin{pmatrix} a_1 \\ \dots \\ a_n \end{pmatrix} \text{ fitting parameters}$$

and functions $f_1(x), f_2(x), \dots, f_n(x)$.

Linear Least Squares, General case

The problem now looks like:

$y_i = F(x_i) + e_i,$ where e_i is a residual: mismatch between the measured value and the one predicted by the fit.

Let's intorduce vector e:

$$e = \begin{pmatrix} e_1 \\ \ldots \\ e_N \end{pmatrix}$$

Linear Least Squares, General case

Let us express the problem in matrix notation:

$$Z=\begin{bmatrix} f_1(x_1) & f_2(x_1) & \dots & f_n(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_n(x_2) \\ \dots & \dots & \dots & \dots \\ f_1(x_N) & f_2(x_N) & \dots & f_n(x_N) \end{bmatrix}$$

Overall we have now:

$$y = Z \cdot a + e$$

Fitting problem in matrix notation.

$$\text{Look for } \min\left(\sum_{i=1}^{N} e_i^{\,2}\right) = \min(e^T e)$$

Linear Least Squares, General case

Look for $\min\left(\sum_{i=1}^{N} e_i^{\,2}\right) = \min\left(\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{n} z_{ij} a_j\right)^2\right) =$

$$\min\left(\left(y - z \cdot a\right)^T \cdot \left(y - z \cdot a\right)\right)$$

$$\frac{\partial\left(e^T e\right)}{\partial a_k} = 0 \quad for\ 1 \le k \le n$$

$$\left(\frac{\partial\left(y - z \cdot a\right)}{\partial a_k}\right)^T \cdot \left(y - z \cdot a\right) = 0$$

Linear Least Squares, General case

$$\left(-z\cdot\frac{\partial(a)}{\partial a_k}\right)^T\cdot(y-z\cdot a)=0$$

$$\left(z\cdot(0\,0\ldots1\ldots0)\right)^T\cdot(y-z\cdot a)=0$$

$$\left(z_{1k}\ z_{2k}\ldots z_{Nk}\right)^T\cdot(y-z\cdot a)=0\quad for\ 1\le k\le n$$

Using Matlab colon notation:

$$\left(\mathrm{z}_{:,\mathrm{k}}\right)^T\cdot(z\cdot a)=\left(z_{:,k}\right)^T\cdot y$$

Or after putting all n equations together:

$$\boxed{z^T\cdot z\cdot a=z^T\cdot y}$$

Linear Least Squares, General case

In general case linear lest squares problem can be formulated as a set of linear equations.

Ways to solve:

1. Gaussian elimination.
2. To calculate the matrix inverse:

$$a = \left(z^T \cdot z\right)^{-1} \cdot z^T \cdot y$$

Suitable for Matlab, see homework 9.

Nonlinear Regression (Least Squares)

What if the fitting function is not linear in fitting parameters?

We get a nonlinear equation (system of equations).

Example:

$$f(x) = a_1(1 - e^{-a_2 x}) + e$$

$$y_i = f(x_i; a_1, a_1, \ldots, a_m) + e_i \text{ or just } y_i = f(x_i) + e_i$$

Again look for the minimum of $\sum_{i=1}^{N} e_i^2$ with respect to the fitting parameters.

Matlab Function FMINSEARCH.

Accepts as input parameters:

1. Name of the function (FUN) to be minimized
2. Vector with initial guess X0 for the fitting parameters

Returns: Vector X of fitting parameters providing the local minimum of FUN.

Function FUN accepts vector X and returns the scalar value dependent on X.

In our case (hw10) FUN should calculate $\sum_{i=1}^{N} e_i^2$ dependent on the fitting parameters b, m, A_1, A_2, …

Matlab Function FMINSEARCH.

Syntax: x = FMINSEARCH(FUN,X0) or
x = FMINSEARCH(FUN,X0,OPTIONS)
See OPTIMSET for the detail on OPTIONS.

x = FMINSEARCH(FUN,X0,OPTIONS,P1,P2,..)
in case you want to pass extra parameters to FMINSEARCH

If no options are set use OPTIONS = [] as a place holder.

Use "@" to specify the FUN:

```
x = fminsearch(@myfun,X0)
```

Gauss-Newton method for nonlinear regression

$y_i = f(x_i; a_1, a_1, ..., a_m) + e_i$ or just $y_i = f(x_i) + e_i$

Look for the minimum of $\sum_{i=1}^{N} e_i^2$ with respect to a_i.

1. Make an initial guess for a: a0.
2. Linearize the equations (use Taylor expansion about a0).
3. Solve for Δa - correction to a0 $\rightarrow$ a1=a0+Δa - improved a-s and our new initial guess.
4. Back to (1).
5. Repeat until $\left| a_{k,j+1} - a_{k,j} \right| < \varepsilon$ for any k.

Gauss-Newton method for nonlinear regression

Linearization by Taylor expansion:

$$y_i = f(x_i) + e_i \approx f(x_i, a0) + \sum_{j=1}^{n} \frac{\partial f(x_i, a0)}{\partial a_n} + e_i$$

$$y_i - f(x_i, a0) = \sum_{j=1}^{n} \frac{\partial f(x_i, a0)}{\partial a_n} + e_i \quad \text{for } i = 1,2,\ldots,N$$

or in matrix form:

$$D = Z \cdot \Delta a + e, \text{ where}$$

$$D = \begin{pmatrix} y_1 - f(x_1, a0) \\ \ldots. \\ y_N - f(x_N, a0) \end{pmatrix}, \quad \text{and} \quad Z = \begin{bmatrix} \frac{\partial f(x_1, a0)}{\partial a_1} & \ldots & \frac{\partial f(x_1, a0)}{\partial a_n} \\ \ldots & \ldots & \ldots \\ \frac{\partial f(x_N, a0)}{\partial a_1} & \ldots & \frac{\partial f(x_N, a0)}{\partial a_n} \end{bmatrix}$$

Gauss-Newton method for nonlinear regression

Linear regression: $y = Z \cdot a + e$

Now, nonlinear regression : $D = Z \cdot \Delta a + e.$

Old good linear equations with Δa in plce of a,
D in place of y and Z with partial derivatives
in place of Z with values of functions.
Solve it for Δa, use $a1 = a0 + \Delta a$ as the new initial
guess and repeat the procedure untill the
convergence criteria are met.....