Updated on October 7, 2021

Lab: Overdetermined Linear Systems

(Hints for Homework 6)

Linear Least Squares: General Fitting Functions

> **Objective.** Given data points $\{(x_i, y_i) \mid i \in \mathbb{N}[1, m]\}$, pick a form for the "fitting" function $f(x)$ and minimize its total error in representing the data.

In lecture, we used a polynomial fitting function

$$p(x) = c_1 + c_2 x + \cdots + c_n x^{n-1} \quad \text{with } n < m, \tag{1}$$

and looked for the coefficients $c_1, \ldots, c_n$ which minimize the 2-norm of the error[1] $\mathbf{r} = \mathbf{y} - p(\mathbf{x})$:

$$\|\mathbf{r}\|_2 = \sqrt{\sum_{i=1}^{m} r_i^2} = \sqrt{\sum_{i=1}^{m} \big(y_i - p(x_i)\big)^2}. \tag{2}$$

The coefficients are found by solving the overdetermined system

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{y}} \text{"="} \underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^{n-1} \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}}_{\mathbf{c}}. \tag{3}$$

The solution $\mathbf{c}$ of $\mathbf{y}$ "=" $V\mathbf{c}$ turns out to be the solution of the *normal equation*

$$V^{\mathrm{T}} V \mathbf{c} = V^{\mathrm{T}} \mathbf{y}\,.$$

If the x and y data points are saved in MATLAB as column vectors `xdp` and `ydp`, respectively, one can solve the normal equation conveniently by

```
V = xdp.^(0:n-1)
c = V\ydp;
```

In the most general terms, the fitting function takes the form

$$f(x) = c_1 f_1(x) + \cdots + c_n f_n(x) \quad \text{with } n < m, \tag{4}$$

[1] This difference is often called the *residual*.

where $f_1, \dots, f_n$ are known functions while $c_1, \dots, c_n$ are to be determined so that the 2-norm of the residual $\mathbf{r} = \mathbf{y} - f(\mathbf{x})$ is minimized. This gives rise to an overdetermined system analogous to (3):

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{y}} \text{``}=\text{''} \underbrace{\begin{bmatrix} f_1(x_1) & f_2(x_1) & f_3(x_1) & \cdots & f_n(x_1) \\ f_1(x_2) & f_2(x_2) & f_3(x_2) & \cdots & f_n(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_1(x_m) & f_2(x_m) & f_3(x_m) & \cdots & f_n(x_m) \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}}_{\mathbf{c}}. \tag{5}$$

As in the case of polynomial fitting, this system is equivalent to the normal equation $V^{\mathrm{T}}V\mathbf{c} = V^{\mathrm{T}}\mathbf{y}$, which can be solved in MATLAB by

```
V = [f1(xdp) f2(xdp) ... fn(xdp)];
c = V\ydp;
```

where `f1`, `f2`, ..., `fn` are anonymous functions corresponding to $f_1, f_2, \dots, f_n$.

Exercise 1 (**FNC** 3.1.4). Define the following data in MATLAB:

```
t = (0:.5:10)';  y = tanh(t);
```

(a) Fit the data to a cubic polynomial and plot the data together with the polynomial fit.

(b) Fit the data to the function $c_1 + c_2 z + c_3 z^2 + c_4 z^3$, where $z = t^2/(1+t^2)$. Plot the data together with the fit. What feature of z makes this fit much better than the original cubic?

Gram-Schmidt and Thin QR Factorization

Consider the subspace $S = \mathrm{Span}\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\} \subset \mathbb{R}^4$ where

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \ \mathbf{a}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \ \mathbf{a}_3 = \begin{bmatrix} 4 \\ 2 \\ -2 \\ 1 \end{bmatrix}.$$

By the *Gram-Schmidt procedure*, we can obtain an orthonormal basis for S:

$$\mathbf{q}_1 = \frac{\mathbf{a}_1}{\|\mathbf{a}_1\|_2} = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

$$\mathbf{q}_2 = \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|_2} = \frac{1}{2}\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}, \quad \text{where } \mathbf{v}_2 = \mathbf{a}_2 - (\mathbf{a}_2^{\mathrm{T}}\mathbf{q}_1)\mathbf{q}_1,$$

$$\mathbf{q}_3 = \frac{\mathbf{v}_3}{\|\mathbf{v}_3\|_2} = \frac{1}{5\sqrt{2}}\begin{bmatrix} 3 \\ 4 \\ -4 \\ -3 \end{bmatrix}, \quad \text{where } \mathbf{v}_3 = \mathbf{a}_3 - (\mathbf{a}_3^{\mathrm{T}}\mathbf{q}_1)\mathbf{q}_1 - (\mathbf{a}_3^{\mathrm{T}}\mathbf{q}_2)\mathbf{q}_2.$$

The equations above can be re-arranged as

$$\begin{aligned}
\mathbf{a}_1 &= \|\mathbf{a}_1\|_2\, \mathbf{q}_1 \\
\mathbf{a}_2 &= (\mathbf{a}_1^{\mathrm{T}}\mathbf{q}_1)\mathbf{q}_1 + \|\mathbf{v}_2\|_2\, \mathbf{q}_2 \\
\mathbf{a}_3 &= (\mathbf{a}_3^{\mathrm{T}}\mathbf{q}_1)\mathbf{q}_1 + (\mathbf{a}_3^{\mathrm{T}}\mathbf{q}_2)\mathbf{q}_2 + \|\mathbf{v}_3\|_2\, \mathbf{q}_3,
\end{aligned}$$

which, in matrix form, is written as

$$\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 \end{bmatrix} \begin{bmatrix} \|\mathbf{a}_1\|_2 & \mathbf{a}_2^{\mathrm{T}}\mathbf{q}_1 & \mathbf{a}_3^{\mathrm{T}}\mathbf{q}_1 \\ 0 & \|\mathbf{v}_1\|_2 & \mathbf{a}_3^{\mathrm{T}}\mathbf{q}_2 \\ 0 & 0 & \|\mathbf{v}_3\|_2 \end{bmatrix},$$

or simply $A = \widehat{Q}\widehat{R}$ (thin QR factorization).

The general Gram-Schmidt procedure can be summarized as

$$\begin{aligned}
\mathbf{q}_1 &= \frac{\mathbf{a}_1}{r_{11}}, \\
\mathbf{q}_2 &= \frac{\mathbf{a}_2 - r_{12}\mathbf{q}_1}{r_{22}}, \\
\mathbf{q}_3 &= \frac{\mathbf{a}_3 - r_{13}\mathbf{q}_1 - r_{23}\mathbf{q}_2}{r_{33}}, \qquad \text{where} \quad r_{ij} = \begin{cases} \mathbf{q}_i^{\mathrm{T}}\mathbf{a}_j, & \text{if } i \neq j \\ \pm\left\| \mathbf{a}_j - \sum_{k=1}^{j-1} r_{kj}\mathbf{q}_k \right\|_2, & \text{if } i = j. \end{cases} \\
&\ \vdots \\
\mathbf{q}_n &= \frac{\mathbf{a}_n - \sum_{i=1}^{n-1} r_{in}\mathbf{q}_i}{r_{nn}},
\end{aligned}$$

To turn these formulas into compute program, we first write down the logics in plain terms, not worrying about programming syntax; this is called a *pseudocode*.

```
Store the dimensions of A as m and n
Initialize Q by making a copy of A. (This will be overwritten below.)
Initialize R as an n × n zero matrix.
for j = 1 to n
    for i = 1 to j − 1
        r_ij = q_i^T a_j
        q_j = q_j − r_ij q_i
    r_jj = ||q_j||_2
    q_j = q_j / r_jj
```

Even though the logic has been written using a nested for-loop, the inner loop can be vectorized resulting in a single loop for j. Either version will be okay for this homework.