17-Least-Squares

January 6, 2017

```
In [17]: %matplotlib inline
         import numpy as np
         import matplotlib.pyplot as plt
         from latools import *
         from sympy import *
         init_printing(use_latex=True)
```

1 Fitting a linear function to data

Suppose that we have a set of data:

i	α_i	β_i
1	2	3
2	1	1
3	4	6
4	3	3
5	7	8

Scientists suspect that there is a linear relationship relating the variables α and β:

$$\beta_i = \alpha_i m + d$$

where m and d are contants to be determined. We can set up the problem of finding m and d as a linear system:

$$\begin{aligned} 2m + d &= 3 \\ 1m + d &= 1 \\ 4m + d &= 6 \\ 3m + d &= 3 \\ 7m + d &= 8 \end{aligned}$$

We can formulate this in matrix form:

$$\begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 4 & 1 \\ 3 & 1 \\ 7 & 1 \end{bmatrix} \begin{bmatrix} m \\ d \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 6 \\ 3 \\ 8 \end{bmatrix}$$

Let's try to solve it using our standard methods. Define the augmented matrix:

```
In [18]: A = matrix_to_rational([[2,1],[1,1],[4,1],[3,1],[7,1]])
         A
```

Out[18]:

$$\begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 4 & 1 \\ 3 & 1 \\ 7 & 1 \end{bmatrix}$$

```
In [19]: b = matrix_to_rational([[3],[1],[6],[3],[8]])
         b
```

Out[19]:

$$\begin{bmatrix} 3 \\ 1 \\ 6 \\ 3 \\ 8 \end{bmatrix}$$

```
In [20]: M = Matrix.hstack(A,b)
         M
```

Out[20]:

$$\begin{bmatrix} 2 & 1 & 3 \\ 1 & 1 & 1 \\ 4 & 1 & 6 \\ 3 & 1 & 3 \\ 7 & 1 & 8 \end{bmatrix}$$

```
In [21]: R = reduced_row_echelon_form(M)
         R
```

Out[21]:

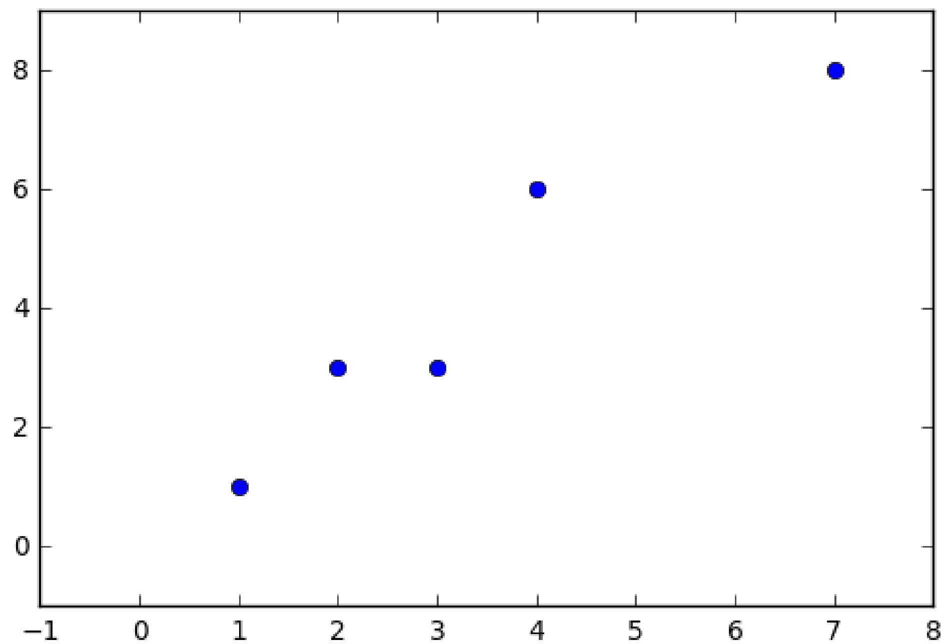
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Thus, the system is equivalent to:

$$\begin{aligned} m &= 0 \\ d &= 0 \\ 0 &= 1 \end{aligned}$$

Due to the last equation, this system is inconsistent. This happens because *there is no straight line that goes through all the points*:

```
In [22]: plt.plot(A[:,0], b[:], 'o')
         plt.axis([-1,8,-1,9])
         None
```




```
In [23]: A[:,0]
```

```
Out[23]:
```

$$\begin{bmatrix} 2 \\ 1 \\ 4 \\ 3 \\ 7 \end{bmatrix}$$

The fact that the data does not perfectly fit a straight line is not surprising. Two factors may be at play:

- The data contains measurement errors. Even if the "actual" values fall on a straight line, the measured values will not.
- The straight line model is not completely accurate. It may be valid as a first approximation, but it will be necessary to adjust it with a more refined model.

This is a situation where we can find a *Least Squares Solution*. To do this we first compute $A_1 = A^T A$ and $\mathbf{b}_1 = A.T * b$:

```
In [24]: A1 = A.T * A
         A1
```

```
Out[24]:
```

$$\begin{bmatrix}79 & 17\\17 & 5\end{bmatrix}$$

```
In [25]: b1 = A.T * b
         b1
```

```
Out[25]:
```

$$\begin{bmatrix}96\\21\end{bmatrix}$$

```
In [26]: M = Matrix.hstack(A1, b1)
         M
```

```
Out[26]:
```

$$\begin{bmatrix}79 & 17 & 96\\17 & 5 & 21\end{bmatrix}$$

```
In [27]: R = reduced_row_echelon_form(M)
         R
```

```
Out[27]:
```

$$\begin{bmatrix}1 & 0 & \frac{123}{106}\\0 & 1 & \frac{27}{106}\end{bmatrix}$$

So, we get the solution:

$$\begin{bmatrix}m\\d\end{bmatrix} = \begin{bmatrix}\frac{123}{106}\\\frac{27}{106}\end{bmatrix}$$

That is, the line that best fits our data (in the least squares sense) is:

$$\beta = \frac{123}{106}\alpha + \frac{27}{106}$$

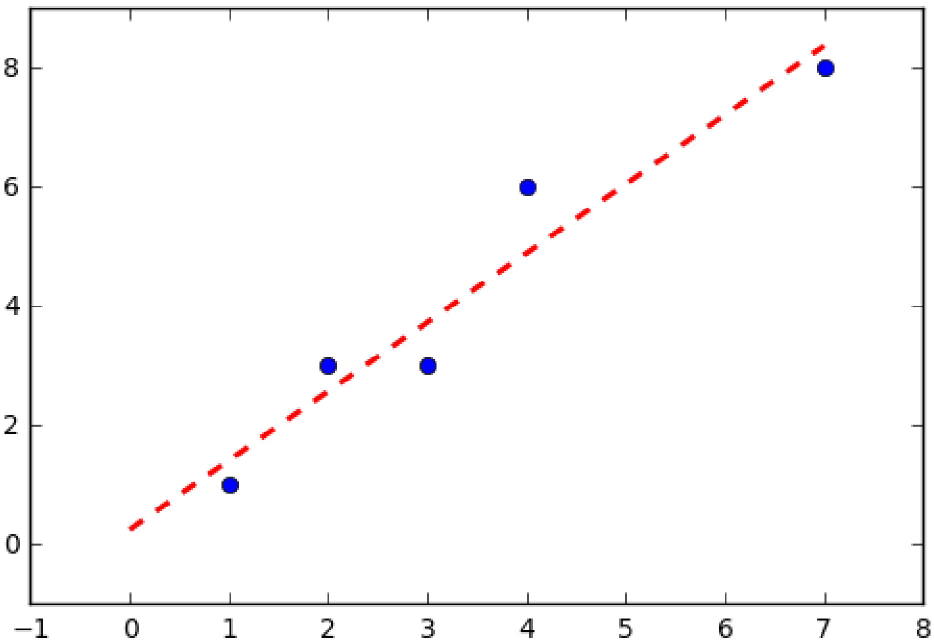
Let's now display the data points again, together with the linear approximation:

```
In [28]: xlss = R[:,2]
         m, d = xlss
         m, d
```

```
Out[28]:
```

$$\left(\frac{123}{106}, \quad \frac{27}{106}\right)$$

```
In [29]: plt.plot(A[:,0], b[:], 'o')
         xvalues = np.linspace(0,7,300)
         yvalues = m * xvalues + d
         plt.plot(xvalues, yvalues, '--', color='red', lw=2)
         plt.axis([-1,8,-1,9])
         None
```



To estimate the error in the approximation, we can compute the residuals:

```
In [30]: r = (b - A * xlss)
         r
```

Out[30]:

$$\begin{bmatrix} \frac{45}{106} \\ -\frac{22}{53} \\ \frac{117}{106} \\ -\frac{39}{53} \\ -\frac{20}{53} \end{bmatrix}$$

```
In [31]: [float(vv) for vv in r]
```

Out[31]:

[0.42452830188679247, −0.41509433962264153, 1.1037735849056605, −0.7358490566037735, −0.37735849

The squared length of the residuals is a measure of how good the linear model is:

```
In [32]: float(r.norm()**2)
```

Out[32]:

$$2.2547169811320753$$

The interpretation of this number is the following: any other pair (m, d) would yield a larger value of $||\mathbf{r}||^2$ for this data set.