week12-01–least-squares

April 22, 2024

1 George McNinch Math 87 - Spring 2024

2 Week 12

3 Least squares

4 Linear Least Squares

We are going to begin our discussion of "least squares" approximation with an example.

4.1 Example

Consider a stretch of highway with four distinct reference points **A**, **B**, **C**, and **D**:

[**A**] —-x1—– [**B**] —x2—— [**C**] —x3——[**D**]

Write `x1` = **AB** for the distance from **A** to **B**, `x2` = **BC**, `x3` = **CD**.

We take some measurements – which potentially reflect errors – , and we seek the *best approximation* to the distances `x1, x2, x3`.

The measurements taken are as follows:

segment	**AD**	**AC**	**BD**	**AB**	**CD**
length	89 m	67 m	53 m	35 m	20 m

Thus the observations suggest the following equations:

```
(1)  x1 + x2 + x3 = 89
(2)  x1 + x2      = 67
(3)       x2 + x3 = 53
(4)  x1           = 35
(5)            x3 = 20
```

These equations aren't compatible, though. Note e.g. that equations `(3) -- (5)` indicate the following:

```
x1 = 35, x3 = 20, x2 = 53-20 = 33
```

but then we find that

```
x1 + x2 + x3 = 35 + 33 + 20 = 88
```

which is incompatible with `(1)`.

And we find that

```
x1 + x2 = 35 + 33 = 68
```

which is incompatible with `(2)`.

Let's formulate these equalities in matrix form.

Thus let

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 89 \\ 67 \\ 53 \\ 35 \\ 20 \end{pmatrix}.$$

With these notations, the above equations suggest that $A\mathbf{x}$ should be equal to $\mathbf{b}$.

Our observation(s) in the preceding slides show, however, that the system of equations $A\mathbf{x} = \mathbf{b}$ is *inconsistent* (i.e. there is no vector $\mathbf{x}$ which makes the equation true).

4.2 Residual

In general, given an $m \times n$ matrix A, a column vector $\mathbf{b} \in \mathbb{R}^m$ and an equation $A\mathbf{x} = \mathbf{b}$, we instead look at the so-called *residual*

$$\mathbf{r} = \mathbf{b} - \mathbf{Ax},$$

and *minimize* this residual.

More precisely, we want to minimize the *magnitude* (or *length*) of this vector.

Thus if $\mathbf{r} = \begin{pmatrix} r_1 & \cdots & r_m \end{pmatrix}^T$, we must minimize the quantity

$$\|\mathbf{r}\| = \left(\sum_{i=1}^{m} r_i^2 \right)^{1/2}$$

Here $\|\mathbf{r}\|$ is the magnitude, also called the Euclidean norm, of the vector $\mathbf{r}$.

In fact, because $f(x) = \sqrt{x}$ is an increasing function of x, we instead minimize the *square* of the magnitude of $\mathbf{r}$::

$$\|\mathbf{r}\|^2 = \sum_{i=1}^{m} r_i^2$$

Thus, we wish to find

$$\min_{\mathbf{x}} \|\mathbf{r}\|^2 = \min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|^2 = \min_{x_1, x_2, \ldots, x_n} \sum_{i=1}^{m} \left(b_i - \sum_{j=1}^{n} A_{ij} x_j \right)^2$$

The idea behind this minimization is to first compute for $1 \leq k \leq n$ the partial derivatives $\dfrac{\partial F}{\partial x_k}$ of the function

$$F(x_1, x_2, \dots, x_n) = \sum_{i=1}^{m} \left(b_i - \sum_{j=1}^{n} A_{ij} x_j \right)^2$$

Critical points - and thus possible minima - for F occur at points $\mathbf{x}$ for which all $\dfrac{\partial F}{\partial x_k}(\mathbf{x}) = 0$.

Now,

$$\frac{\partial F}{\partial x_k} = \sum_{i=1}^{m} 2 \left(b_i - \sum_{j=1}^{n} A_{ij} x_j \right) (-A_{ik}) = 2 \left(\sum_{i=1}^{m} (-A_{ik} b_i) + \sum_{i=1}^{m} A_{ik} \sum_{j=1}^{n} A_{ij} x_j \right)$$

and this expression is equal to the k-th coefficient of the vector

$$2\left(-A^T \mathbf{b} + A^T A \mathbf{x}\right)$$

Thus, the condition $\dfrac{\partial F}{\partial x_k} = 0$ for all k is equivalent to the so-called *normal equations*:

$$(\diamondsuit) \quad A^T A \mathbf{x} = A^T \mathbf{b}.$$

Thus the solutions $\mathbf{x}$ to the normal equations $(\diamondsuit)$ are precisely the critical points of the function F.

Recall that $A \in \mathbb{R}^{m \times n}$. Thus, $A^T \in \mathbb{R}^{n \times m}$ so that the matrix $A^T A$ is $n \times n$; in particular, $A^T A$ is always a square matrix.

Moreover, $A^T A$ is *symmetric*, since

$$(A^T A)^T = A^T (A^T)^T = A^T A.$$

We are interested here in the case of *overdetermined systems* – i.e. in the case where A has more rows than columns ("more equations than variables"). Thus $m \geq n$.

We also are interested in the case where A has rank n – i.e.A has n linearly independent columns – since otherwise we don't expect to have enough information to find $\mathbf{x}$.

4.3 Proposition

Let $A \in \mathbb{R}^{m \times n}$, suppose that $m \geq n$ and that A has rank n. Then $A^T \cdot A$ is invertible.

Proof:

Since $A^T \cdot A$ is an $n \times n$ square matrix, the proposition will follow if we argue that the null space $\text{Null}(A^T A)$ is zero. So: suppose that $\mathbf{v} \in \text{Null}(A^T A)$.

Thus $A^T A \mathbf{v} = 0$ and thus also $\mathbf{v}^T A^T \cdot A \mathbf{v} = 0$.

Now,

$$0 = \mathbf{v}^T A^T \cdot A \mathbf{v} = (A\mathbf{v})^T (A\mathbf{v})$$

and of course for any vector $\mathbf{w}$, we know that

$$0 = \mathbf{w}^T\mathbf{w} \implies \mathbf{w} = \mathbf{0}.$$

We now conclude that $A\mathbf{v} = 0$, so $\mathbf{v} \in \mathrm{Null}(A)$. Since A has rank n, the Null space of A is equal to zero, and we condlue that $\mathbf{v} = \mathbf{0}$.

We have now proved that $\mathrm{Null}(A^T A)$ is zero, as required.

Remark: What we have really showed is that the symmetric matrix $A^T A$ is *definite*: $\mathbf{v}^T A^T A\mathbf{v} = 0 \implies \mathbf{v} = \mathbf{0}$.

We finally claim that this solution must minimize the magnitude of the residual.

This depends on a "second derivative test" argument for which I'm not going to give full details. The main point is that the "second derivative" in this context – known as the Hessian – concides with the matrix $2A^T A$. Now, under our assumptions the matrix $A^T A$ is postive definite, and it follows that $\mathbf{x}_0$ is a global minimum for the magnitude of the residual!

4.4 Return to the example

Recall that

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 89 \\ 67 \\ 53 \\ 35 \\ 20 \end{pmatrix}.$$

So to minimize the magnitude of the residual, we must solve the normal equations:

$$A^T A\mathbf{x} = A^T\mathbf{b}.$$

Now

$$A^T A = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 3 \end{pmatrix}$$

and

$$A^T\mathbf{b} = \begin{pmatrix} 191 \\ 209 \\ 162 \end{pmatrix}$$

So we need to solve the equation

$$\begin{pmatrix} 3 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 3 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 191 \\ 209 \\ 162 \end{pmatrix}$$

[1]:
```python
import numpy as np
import numpy.linalg as la

A= np.array([[1,1,1],[1,1,0],[0,1,1],[1,0,0],[0,0,1]])
b = np.array([89,67,53,35,20])


x0=la.solve(A.T @ A, A.T @ b)

def residual(x):
    return b - A @ x

def magnitude(x):
    return np.sqrt(x@x)

[x0,magnitude(residual(x0))]
```

[1]:
```
[array([35.125, 32.5  , 20.625]), 1.1726039399558574]
```

Thus the *least squares solution* is

$$\mathbf{x}_0 = \begin{pmatrix} 35.125 \\ 32.5 \\ 20.625 \end{pmatrix}$$

and

$$\|\mathbf{b} - A\mathbf{x}_0\| \approx 1.1726$$

Recall that our "first guess" for a solution (based on some of the measurements) was

$$\mathbf{x}_1 = \begin{pmatrix} 35 \\ 33 \\ 20 \end{pmatrix}$$

The residual is indeed larger for x_1:

[2]:
```python
x1 = np.array([35,33,20])

magnitude(residual(x1))
```

[2]:
```
1.4142135623730951
```

Let's note that `numpy` already implements this least-squares functionality:

– you can read more about it here

[3]:
```python
res=la.lstsq(A,b,rcond=None)
res[0]
```

[3]:
```
array([35.125, 32.5  , 20.625])
```

5 Example recapitulated

Consider towns in a certain area labeled `[a,b,c,d,e,f,g,h,i,j]`. You have hired someone to estimate the population of these towns, but they got confused and have reported the populations of *pairs* of towns.

Thus you have the following data:

[4]:
```python
p_est ={('a', 'b'): 21.21,
 ('a', 'c'): 24.35,
 ('a', 'd'): 34.57,
 ('a', 'e'): 32.72,
 ('a', 'f'): 36.14,
 ('a', 'g'): 17.21,
 ('a', 'h'): 26.41,
 ('a', 'i'): 29.92,
 ('a', 'j'): 32.84,
 ('b', 'c'): 29.09,
 ('b', 'd'): 33.78,
 ('b', 'e'): 40.63,
 ('b', 'f'): 44.46,
 ('b', 'g'): 17.03,
 ('b', 'h'): 30.58,
 ('b', 'i'): 32.7,
 ('b', 'j'): 31.19,
 ('c', 'd'): 30.08,
 ('c', 'e'): 37.3,
 ('c', 'f'): 39.4,
 ('c', 'g'): 20.44,
 ('c', 'h'): 28.34,
 ('c', 'i'): 31.07,
 ('c', 'j'): 36.78,
 ('d', 'e'): 41.44,
 ('d', 'f'): 47.97,
 ('d', 'g'): 28.51,
 ('d', 'h'): 38.0,
 ('d', 'i'): 38.71,
 ('d', 'j'): 39.24,
 ('e', 'f'): 59.61,
 ('e', 'g'): 29.19,
 ('e', 'h'): 35.09,
 ('e', 'i'): 42.18,
```

```
 ('e', 'j'): 46.8,
 ('f', 'g'): 34.14,
 ('f', 'h'): 46.71,
 ('f', 'i'): 48.07,
 ('f', 'j'): 49.53,
 ('g', 'h'): 23.46,
 ('g', 'i'): 22.13,
 ('g', 'j'): 24.21,
 ('h', 'i'): 29.03,
 ('h', 'j'): 32.51,
 ('i', 'j'): 38.17}

p_est
```

[4]:
```
{('a', 'b'): 21.21,
 ('a', 'c'): 24.35,
 ('a', 'd'): 34.57,
 ('a', 'e'): 32.72,
 ('a', 'f'): 36.14,
 ('a', 'g'): 17.21,
 ('a', 'h'): 26.41,
 ('a', 'i'): 29.92,
 ('a', 'j'): 32.84,
 ('b', 'c'): 29.09,
 ('b', 'd'): 33.78,
 ('b', 'e'): 40.63,
 ('b', 'f'): 44.46,
 ('b', 'g'): 17.03,
 ('b', 'h'): 30.58,
 ('b', 'i'): 32.7,
 ('b', 'j'): 31.19,
 ('c', 'd'): 30.08,
 ('c', 'e'): 37.3,
 ('c', 'f'): 39.4,
 ('c', 'g'): 20.44,
 ('c', 'h'): 28.34,
 ('c', 'i'): 31.07,
 ('c', 'j'): 36.78,
 ('d', 'e'): 41.44,
 ('d', 'f'): 47.97,
 ('d', 'g'): 28.51,
 ('d', 'h'): 38.0,
 ('d', 'i'): 38.71,
 ('d', 'j'): 39.24,
 ('e', 'f'): 59.61,
 ('e', 'g'): 29.19,
 ('e', 'h'): 35.09,
```

```
 ('e', 'i'): 42.18,
 ('e', 'j'): 46.8,
 ('f', 'g'): 34.14,
 ('f', 'h'): 46.71,
 ('f', 'i'): 48.07,
 ('f', 'j'): 49.53,
 ('g', 'h'): 23.46,
 ('g', 'i'): 22.13,
 ('g', 'j'): 24.21,
 ('h', 'i'): 29.03,
 ('h', 'j'): 32.51,
 ('i', 'j'): 38.17}
```

The value `mdist[('a','b')] == 21.21` means that the sum of the populations of towns `a` and `b` is 21.21 thousand people.

Let's form the matrix `M` and vector `b` so that `M @ x == b` expresses these distance relationships.

[7]:
```python
towns = [ 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j' ]
pairs = list(p_est.keys())
```

[8]:
```python
def sbv(i,n):
    # return the ith standard basis vector of length n
    return np.array([1 if j == i else 0 for j in range(n)])

def sbv_list(elem,ls):
    # return the standard basis vector determined by the position of `elem` in␣
 ↪the list `ls`
    return sbv(list(ls).index(elem),len(ls))

M = np.array([sbv_list(x,towns) + sbv_list(y,towns) for (x,y) in p_est.keys()])

b = np.array([p_est[x] for x in p_est.keys()])

(M,b)
```

[8]:
```
(array([[1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 1, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 0, 1, 0, 0, 0, 0, 0, 0],
        [1, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [1, 0, 0, 0, 0, 1, 0, 0, 0, 0],
        [1, 0, 0, 0, 0, 0, 1, 0, 0, 0],
        [1, 0, 0, 0, 0, 0, 0, 1, 0, 0],
        [1, 0, 0, 0, 0, 0, 0, 0, 1, 0],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
        [0, 1, 1, 0, 0, 0, 0, 0, 0, 0],
        [0, 1, 0, 1, 0, 0, 0, 0, 0, 0],
        [0, 1, 0, 0, 1, 0, 0, 0, 0, 0],
```

```
       [0, 1, 0, 0, 0, 1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 1, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 1, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 1, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 1, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 1, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0, 1, 0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 1, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 1, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 1, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 1, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 0, 1, 1, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 1],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 1]]),
 array([21.21, 24.35, 34.57, 32.72, 36.14, 17.21, 26.41, 29.92, 32.84,
        29.09, 33.78, 40.63, 44.46, 17.03, 30.58, 32.7 , 31.19, 30.08,
        37.3 , 39.4 , 20.44, 28.34, 31.07, 36.78, 41.44, 47.97, 28.51,
        38.  , 38.71, 39.24, 59.61, 29.19, 35.09, 42.18, 46.8 , 34.14,
        46.71, 48.07, 49.53, 23.46, 22.13, 24.21, 29.03, 32.51, 38.17]))
```

Let's use `least squares` to find the best solution to `M @ x == b`

[9]:
```
x = la.lstsq(M,b,rcond=None)
x[0]
```

[9]:
```
array([10.63041667, 13.79291667, 13.31541667, 20.24666667, 24.32916667,
       29.46291667,  5.74916667, 14.97541667, 17.70666667, 20.11791667])
```

[10]:
```
(M @ x[0],b - M @ x[0])
```

[10]:
```
(array([24.42333333, 23.94583333, 30.87708333, 34.95958333, 40.09333333,
        16.37958333, 25.60583333, 28.33708333, 30.74833333, 27.10833333,
        34.03958333, 38.12208333, 43.25583333, 19.54208333, 28.76833333,
        31.49958333, 33.91083333, 33.56208333, 37.64458333, 42.77833333,
        19.06458333, 28.29083333, 31.02208333, 33.43333333, 44.57583333,
        49.70958333, 25.99583333, 35.22208333, 37.95333333, 40.36458333,
        53.79208333, 30.07833333, 39.30458333, 42.03583333, 44.44708333,
        35.21208333, 44.43833333, 47.16958333, 49.58083333, 20.72458333,
        23.45583333, 25.86708333, 32.68208333, 35.09333333, 37.82458333]),
 array([-3.21333333,  0.40416667,  3.69291667, -2.23958333, -3.95333333,
         0.83041667,  0.80416667,  1.58291667,  2.09166667,  1.98166667,
        -0.25958333,  2.50791667,  1.20416667, -2.51208333,  1.81166667,
         1.20041667, -2.72083333, -3.48208333, -0.34458333, -3.37833333,
         1.37541667,  0.04916667,  0.04791667,  3.34666667, -3.13583333,
        -1.73958333,  2.51416667,  2.77791667,  0.75666667, -1.12458333,
         5.81791667, -0.88833333, -4.21458333,  0.14416667,  2.35291667,
        -1.07208333,  2.27166667,  0.90041667, -0.05083333,  2.73541667,
        -1.32583333, -1.65708333, -3.65208333, -2.58333333,  0.34541667]))
```

[ ]: