EX.4.1.9.a, Sauer3

Find the best parabola through each data point set in EX.4.1.8.a, and compare the RMSE with the best-line fit. The points are

$$(0,0), \quad (1,3), \quad (2,3), \quad (5,6).$$

Instructions:

Please handwrite the matrix A and b, and then use python to solve the linear least squares problem by giving x to 3 digits after the dot. Then write the best-fit polynomial (with 3 digits after the dot coefficients). Then compute the RMSE with Python and please compare the RMSE of this problem with the RMSE of EX.4.1.8.a.

So to repeat, I need: (1) A, b, (2) x, (3) parabola, (4) RMSE, and (5) comparison of RMSEs.

Note: I give answers in fractional form and in decimal form. You only need to give decimal form.

EX.4.1.9.a, Sauer3, solution, Langou

Colab: `https://colab.research.google.com/drive/1_rkglFPEyktaBG3zKfu6q0lIKwVdZTo8`

Setting

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 5 & 25 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 0 \\ 3 \\ 3 \\ 6 \end{bmatrix},$$

we form the linear least squares problem

$$\min_{x \in \mathbb{R}^3} \| \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 5 & 25 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 0 \\ 3 \\ 3 \\ 6 \end{bmatrix} \|_2.$$

Solving with python, we find

$$x = \frac{1}{362} \begin{bmatrix} 126 \\ 705 \\ -60 \end{bmatrix} \approx \begin{bmatrix} 0.348 \\ 1.948 \\ -0.166 \end{bmatrix}$$

We find that the best-fit parabola in the least squares sense that fits our data points is

$$y = -\frac{60}{362}x^2 + \frac{705}{362}x + \frac{126}{362}.$$

or

$$y = -0.166x^2 + 1.948x + 0.348.$$

Using python, we compute the residual

$$r = \frac{1}{362} \begin{bmatrix} -126 \\ 315 \\ -210 \\ 21 \end{bmatrix}$$

The 2-norm error is

$$\|r\|_2 = \frac{1}{362}\sqrt{(-126)^2 + (315)^2 + (-210)^2 + (21)^2} = \frac{1}{362}\sqrt{159642} = \frac{21}{\sqrt{362}} \approx 1.104.$$

The root mean squared error (RMSE) is

$$\text{RMSE} = \frac{\|r\|_2}{\sqrt{m}}\frac{21}{2\sqrt{362}} \approx 0.552.$$

With a line fit, see EX.4.1.8.a, we found an RMSE of 0.694. With With a parabola fit, see this exercise, EX.4.1.9.a, we found an RMSE of 0.552. We see that the RMSE is lower with a parabola fit (this exercise, EX.4.1.9.a) than with a line fit (EX.4.1.8.a). It must be so because the set of all parabolae includes the lines and therefore, in EX.4.1.9.a, we minimize over a strictly larger set, so we get a solution that is at least as good, if not better.

The RMSE with a parabola is smaller at the 4 nodes than the RMSE with a line, however it is not clear that a parabola fit is always "better" than a line fit. In general we look how well the fit is "overall". "For data points that we do not have". So be very careful before drawing conclusions. A line fit is, for some applications, better than a parabola fit. The problem with a parabola is that it might "overfit" the data.

What is below is not needed.

Check:

$$\begin{aligned} A^T r &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 5 \\ 0 & 1 & 4 & 25 \end{bmatrix} \frac{1}{362} \begin{bmatrix} -126 \\ 315 \\ -210 \\ 21 \end{bmatrix} = \frac{1}{362} \begin{bmatrix} (1)(-126) + (1)(315) + (1)(-210) + (1)(21) \\ (0)(-126) + (1)(315) + (2)(-210) + (5)(21) \\ (0)(-126) + (1)(315) + (4)(-210) + (25)(21) \end{bmatrix} \\ &= \frac{1}{362} \begin{bmatrix} -126 + 315 - 210 + 21 \\ 315 - 420 + 105 \\ 315 - 840 + 525 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \qquad \checkmark \end{aligned}$$

What is below is not needed.

Python helper notebook:

```
import numpy as np
from math import sqrt
import matplotlib.pyplot as plt

data = np.array([
        [0.,0.],
        [1.,3.],
        [2.,3.],
        [5.,6.] ])

xx = data[:,0]
yy = data[:,1]

A = np.array([ np.ones( xx.shape ), xx ]).T
b = np.array([ yy ]).T
print("A =\n",A)
```

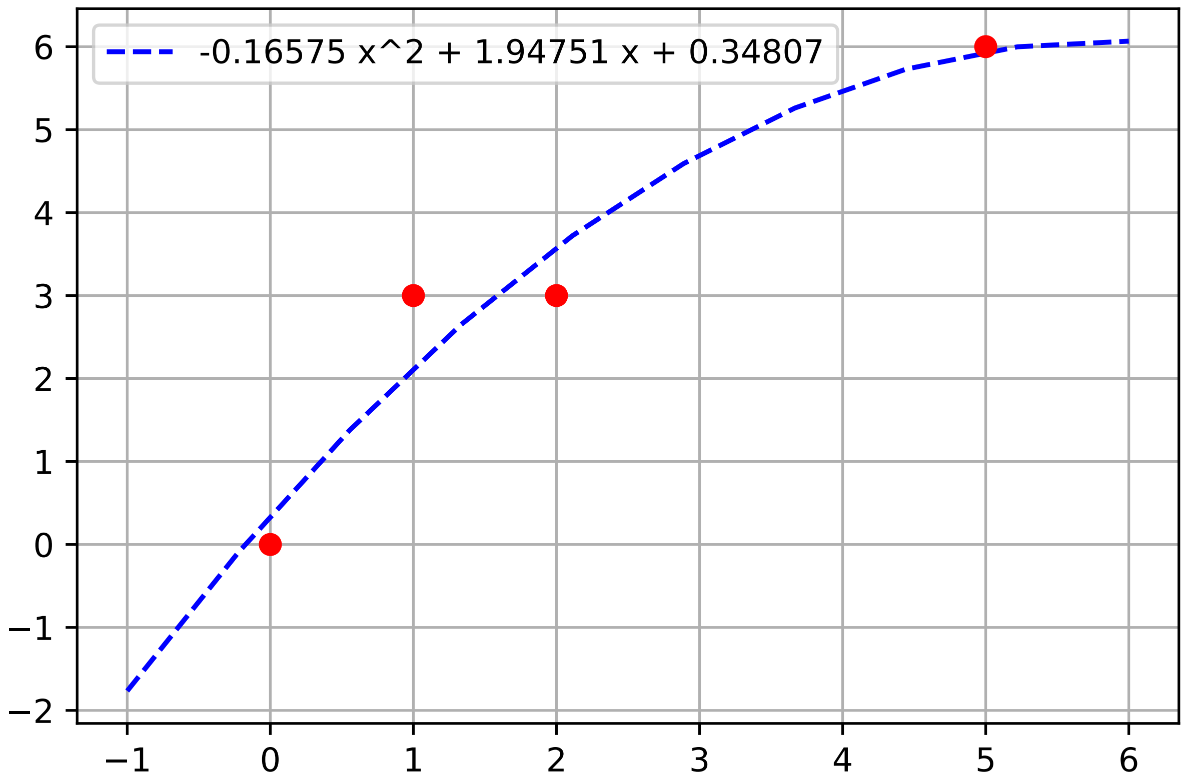
```python
print("b =\n",b)

# compute x either using np.linalg.lstsq or normal equation methods
# x = np.linalg.lstsq(A,b,rcond=None)[0]
x = np.linalg.solve( A.T@A, A.T@b )

print("x =\n",x)
print("[ Ax, b ]=\n",np.concatenate((A@x,b),axis=1))
r = b - A@x
print("r =\n",r)
print("|| b - Ax ||_2 = ", f"{np.linalg.norm( r, 2 ):.4f}")
print("RMSE = ", f"{np.linalg.norm( r, 2 ) / sqrt( A.shape[0] ) :.4f}")
print("|| A^T ( b - Ax ) ||_oo =",
      f"{np.linalg.norm( A.T@( A@x - b), np.infty ):.1e}" )

# plot
xxx = np.linspace( -1., 6., 10)
yyy = x[0] + x[1] * xxx
label_ = f'{x[1,0]:7.5f}'+' x + '+f'{x[0,0]:7.5f}'
plt.plot(xxx, yyy, '--b',label=label_)
plt.plot( xx, yy, 'ro')
plt.legend()
plt.grid()
plt.show()
```

```
A =
 [[ 1.  0.  0.]
 [ 1.  1.  1.]
 [ 1.  2.  4.]
 [ 1.  5. 25.]]
b =
 [[0.]
 [3.]
 [3.]
 [6.]]
x =
 [[ 0.3480663 ]
 [ 1.94751381]
 [-0.16574586]]
[ Ax, b ]=
 [[0.3480663  0.        ]
 [2.12983425 3.        ]
 [3.5801105  3.        ]
 [5.94198895 6.        ]]
r =
 [[-0.3480663 ]
 [ 0.87016575]
 [-0.5801105 ]
 [ 0.05801105]]
|| b - Ax ||_2 =  1.1037
RMSE =  0.5519
|| A^T ( b - Ax ) ||_oo = 1.2e-14
```



-0.16575 x^2 + 1.94751 x + 0.34807
−2
−1
0
1
2
3
4
5
6
−1
0
1
2
3
4
5
6