04.04 least squares and the normal equations

least squares as an idea has been around since gauss and legendre, depending on how you define "as an idea". and "L2" is all over, underpinning this modern world.

rootfinding has been cases where solutions exist, exact or approximate; what about when they dont exist? interpolation has focused on models that intersect all data; is that always the best model?

1 inconsistent systems of equations

$$\begin{aligned} x_1 + x_2 &= 2 \\ x_1 - x_2 &= 1 \\ x_1 + x_2 &= 3. \end{aligned}$$

who do you uninvite to the party? lets not have math be exclusionist; lets compromise with a solution that comes close. interpret "close" in terms of euclidean distance as method of the week.

for $m \times n$ system $Ax = b$, C

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

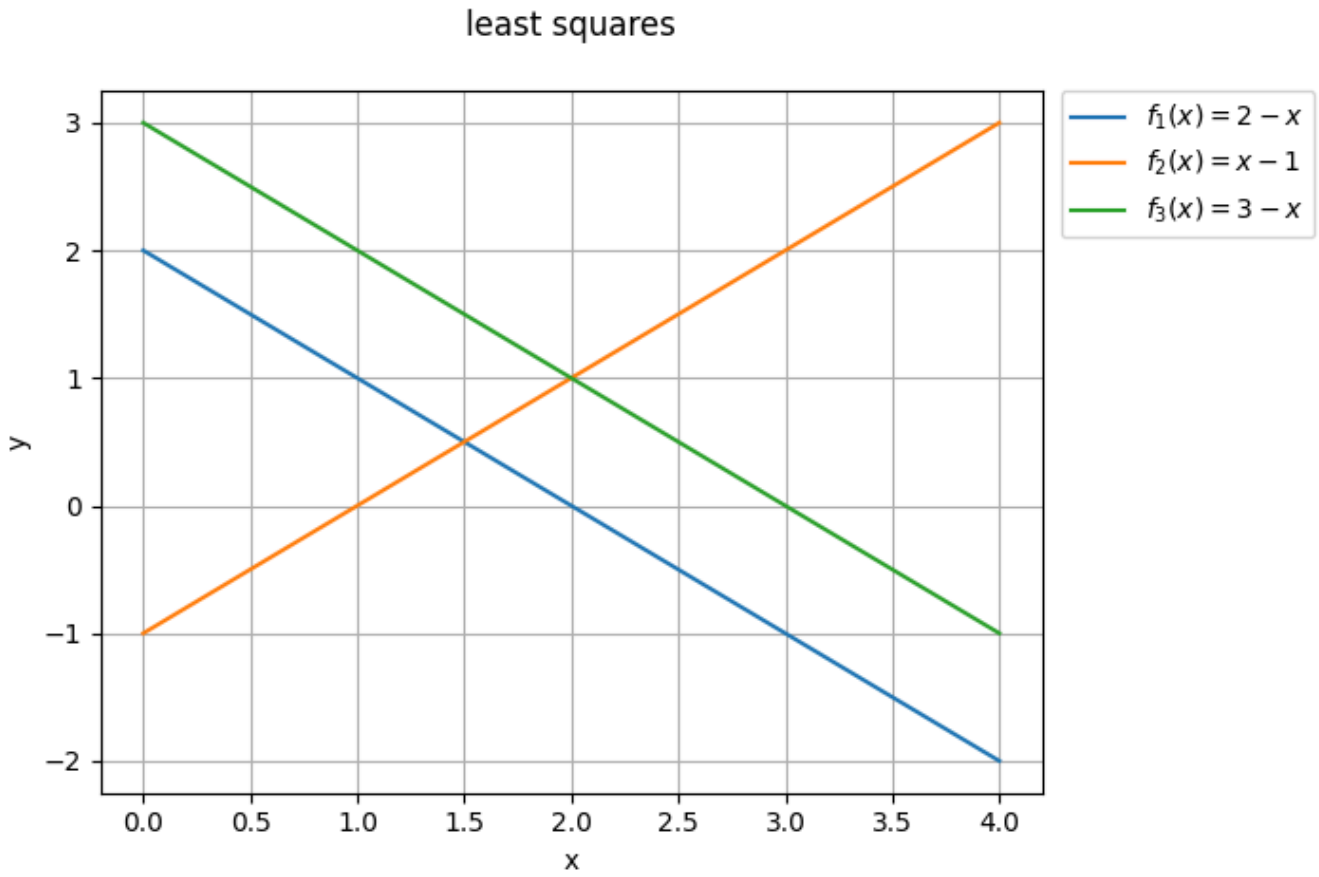
$$\Downarrow \quad \text{or}$$

$$x_1 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + x_2 \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} \quad \Rightarrow \quad x_1 v_1 + x_2 v_2 + \cdots + x_n v_n = b.$$

ie, hit target 3d b with two 3d vectors, x_1, x_2. "closest" to b is in the plane $Ax = x_1 + x_2$ in $\mathbb{R}^3$.

code, visual

```
# this is a plain jane, no special method script to plot the previous system

import matplotlib.pyplot as plt
import numpy as np

def main():

  y1 = lambda x: 2 - x
  y2 = lambda x: x - 1
```



usw

there is no x_1, x_2 that solves the problem but there is a point in the plane Ax of all possible candidates that is closest to b. also, $\vec{r} = b - A\bar{x}$ is perpendicular to $\{Ax | x \in \mathbb{R}^n\}$.

recall: $A^T_{m \times n}$ is $n \times m$ matrix; $(A + B)^T = A^T + B^T$; $(AB)^T = B^T A^T$.

$$\vec{r} = b - A\bar{x} \perp \{Ax | x \in \mathbb{R}^n\}$$

$$\Downarrow \quad \vec{r} \neq 0 \text{ but imagine it for a moment}$$

$$(Ax)^T(b - A\bar{x}) = 0 \quad \forall\, x \in \mathrm{r}^n$$

$$\Downarrow$$

$$x^T A^T(b - A\bar{x}) = 0 \quad \forall\, x \in \mathrm{r}^n$$

$$\Downarrow \quad A^T(b - A\bar{x}) \perp \text{ all } x \in \mathbb{R}^n$$

$$A^T(b - A\bar{x}) = 0$$

$$\Downarrow \quad \text{least squares solution}$$

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

normal equations for least squares

given inconsistent system $Ax = b$, solve $A^T A\bar{x} = A^T b$ where $\bar{x}$ that minimizes the euclidean length of the residual $r = b - Ax$.

▾ example 01

use normal equations to find least squares solution of inconsistent system

$$A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix}, b = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}.$$

$$A^T A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix},$$

$$A^T b = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}.$$

$$\Downarrow \quad \text{normal equations}$$

$$\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}$$

$$\Downarrow \quad \text{gaussian elimination}$$

$$\left[\begin{array}{cc|c} 3 & 1 & 6 \\ 1 & 3 & 4 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 3 & 1 & 6 \\ 0 & \frac{8}{3} & 2 \end{array}\right] \quad \Rightarrow \quad \bar{x} = \begin{bmatrix} \bar{x_1} \\ \bar{x_2} \end{bmatrix} = \begin{bmatrix} \frac{7}{4} \\ \frac{3}{4} \end{bmatrix}.$$

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{7}{4} \\ \frac{3}{4} \end{bmatrix} = \begin{bmatrix} 2.5 \\ 1 \\ 2.5 \end{bmatrix} \neq \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} \qquad \text{remember youve solved exactly for a system wh}$$

$$\Downarrow \quad \text{and}$$

$$r = b - A\bar{x} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 1 \\ 2.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.0 \\ 0.5 \end{bmatrix}$$

if r is zero, the solution is exact. if not, the euclidean length of the residual vector is a backward error measure of how far $\bar{x}$ is from being a solution. the size of the residual is expressed in more than one way:

$$\text{euclidean 2-norm, } ||r||_2 = \sqrt{r_1^2 + \cdots + r_m^2}$$

$$\text{squared error, } SE = r_1^2 + \cdots + r_m^2$$

$$\text{root mean squared error, } RMSE = \sqrt{(r_1^2 + \cdots + r_m^2)/m}.$$

$$\Rightarrow \quad RMSE = \sqrt{SE/m} = \frac{||r||_2}{\sqrt{m}}$$

▾ example 02

solve least squares problem

$$\begin{bmatrix} 1 & -4 \\ 2 & 3 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -3 \\ 15 \\ 9 \end{bmatrix}.$$

$$A^T A x = A^T b$$

$$\Downarrow$$

$$\begin{bmatrix} 9 & 6 \\ 6 & 29 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 45 \\ 75 \end{bmatrix}$$

$$\Downarrow \quad \text{gauss elimination or numpy.linalg.solve(), usw}$$

$$\bar{x} = \begin{bmatrix} \bar{x_1} \\ \bar{x_2} \end{bmatrix} = \begin{bmatrix} 3.8 \\ 1.8 \end{bmatrix},$$

$$r = b - A\bar{x}$$

$$= \begin{bmatrix} -3 \\ 15 \\ 9 \end{bmatrix} - \begin{bmatrix} 1 & -4 \\ 2 & 3 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 3.8 \\ 1.8 \end{bmatrix}$$

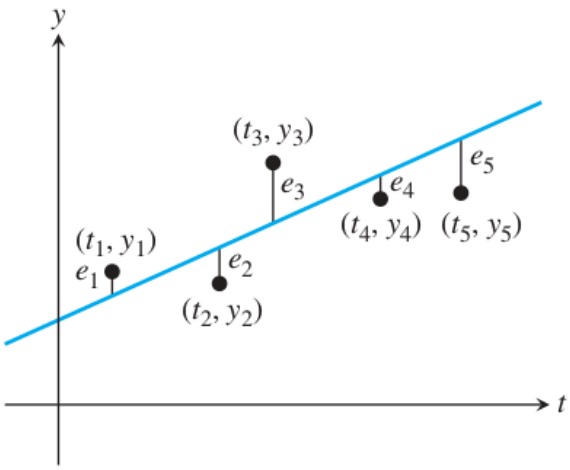
$$= \begin{bmatrix} -3 \\ 15 \\ 9 \end{bmatrix} - \begin{bmatrix} -3.4 \\ 13 \\ 11.2 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 2 \\ -2.2 \end{bmatrix},$$

$$\Downarrow$$

$$||e||_2 = \sqrt{(0.4)^2 + 2^2 + (-2.2)^2} = 3.$$

2 fitting models to data

let $(t_1, y_1), \ldots, (t_m, y_m)$ be a set of points in the plane. given a fixed class of models, such as all lines $y = c_1 + c_2 t$, seek to locate the specific instance of the model that best fits the data points in the 2-norm. the core of the least squares idea consists of measuring the residual of the fit by the squared errors of the model at the data points and finding the model parameters that minimize this quantity.



▾ example 03

continues example 01.

find best fit line for $(t, y) = (1, 2), (-1, 1), (1, 3)$.

find best c_1, c_2 for model $y = c_1 + c_2 t$.

$$
\begin{aligned}
c_1 + c_2(1) &= 2 \\
c_1 + c_2(-1) &= 1 \\
c_1 + c_2(1) &= 3
\end{aligned}
$$

$$\Downarrow$$

$$
\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}
$$

$$\Downarrow$$

$$
c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \frac{7}{4} \\ \frac{3}{4} \end{bmatrix}.
$$

t	y	line	error
1	2	2.5	-0.5
-1	1	1.0	0.0
1	3	2.5	0.5

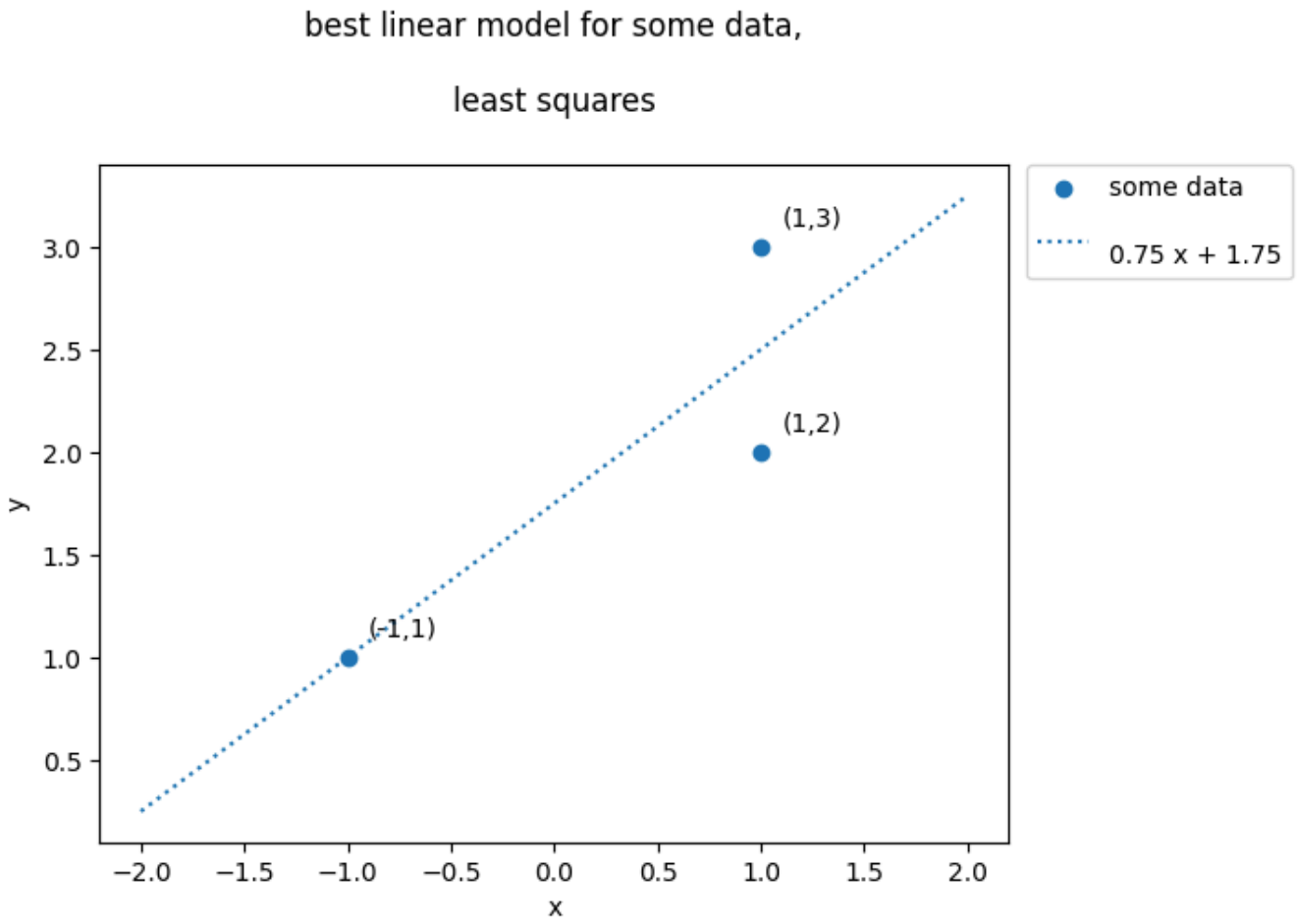
$$\Rightarrow \quad RMSE = \frac{1}{\sqrt{6}}.$$

code

```
 1 # example 03: uses polyfit,poly1d
 2
 3 import matplotlib.pyplot as plt
 4 import numpy as np
 5 import pandas as pd
 6
 7 pd.set_option('display.max_rows',10)
 8
 9 def main():
10   xs = np.array([1,-1,1])
11   ys = np.array([2,1,3])
12
```

```
  SE: [0.5]

RMSE: [0.40824829]
```

	t	y	model	error
0	1.0	2.0	2.5	-0.5000
1	-1.0	1.0	1.0	-0.0000
2	1.0	3.0	2.5	0.5000



algorithm

fitting data by least squares

```
given data (t,y),
```

```
01 choose parameterized model - eg, f(t) = c[1] + c[2]*t;

02 force model to fit data - ie, y_hat = f(t);

03 solve with normal equations.
```

(they can look like this, too.)

▾ example 04

find best fit line for $(t, y) = (-1, 1), (0, 0), (1, 0), (2, -2)$.

1. for model $y = c_1 + c_2 t$.

$$
\begin{aligned}
c_1 + c_2(-1) &= 1 \\
c_1 + c_2(0) &= 0 \\
c_1 + c_2(1) &= 0 \\
c_1 + c_2(2) &= -2
\end{aligned}
$$

$$\Downarrow$$

$$
\begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}
\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} =
\begin{bmatrix} 1 \\ 0 \\ 0 \\ -2 \end{bmatrix}
$$

$$\Downarrow$$

$$
\begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix}
\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} =
\begin{bmatrix} -1 \\ -5 \end{bmatrix}
$$

$$\Downarrow$$

$$
c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} =
\begin{bmatrix} 0.2 \\ -0.9 \end{bmatrix}
$$

t	y	line	error
-1	1	1.1	-0.1
0	0	0.2	-0.2
1	0	-0.7	0.7
2	-2	-1.6	-0.4

$$\Rightarrow \quad SE = (-0.1)^2 + (-0.2)^2 + (0.7)^2 + (-0.4)^2 = 0.7, \quad RMSE = \sqrt{\tfrac{0.7}{4}} = 0.418.$$

2. for model $y = c_1 + c_2 t + c_3 t^2$.

$$\begin{aligned} c_1 + c_2(-1) + c_3(-1)^2 &= 1 \\ c_1 + c_2(0) + c_3(0)^2 &= 0 \\ c_1 + c_2(1) + c_3(1)^2 &= 0 \\ c_1 + c_2(2) + c_3(2)^2 &= -2 \end{aligned}$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -2 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 4 & 2 & 6 \\ 2 & 6 & 8 \\ 6 & 8 & 18 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} -1 \\ -5 \\ -7 \end{bmatrix}$$

$$\Downarrow$$

$$c = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0.45 \\ -0.65 \\ -0.25 \end{bmatrix}$$

t	y	parabola	error
-1	1	0.85	0.15
0	0	0.45	-0.45

	t	y	parabola	error
1	0		-0.45	0.45
2	-2		-1.85	-0.15

$$\Rightarrow \quad SE = (0.15)^2 + (-0.45)^2 + (0.45)^2 + (-0.15)^2 = 0.45, \quad RMSE = \sqrt{\tfrac{0.45}{4}} = 0.335.$$

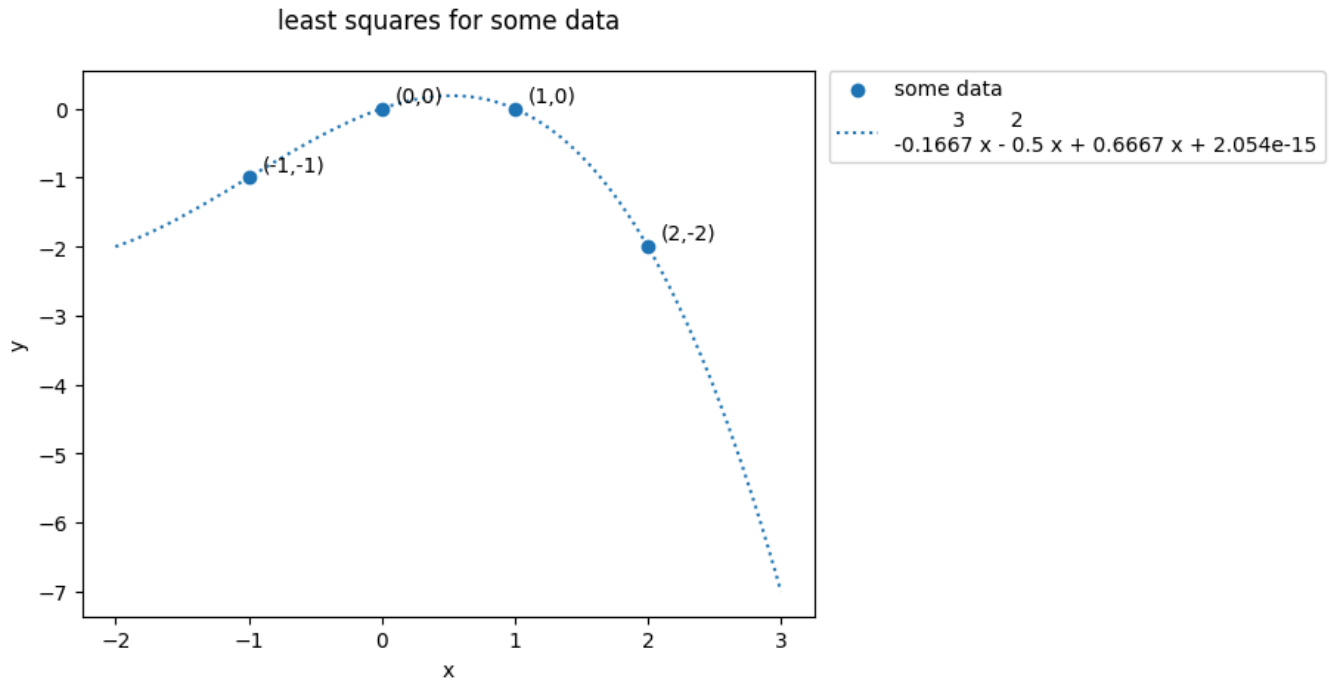
code

```
# during lecture, ok?

#  # data for example 04
#  xs = np.array([-1,0,1,2])
#  ys = np.array([-1,0,0,-2])

# 01 copy-pasted previous code cell for example 03
# 02 renamed pasted code cell to example 04
# 03 replaced lines 10,11 with xs,ys above
# 04 updated line 15 argument "deg" to 2 from 1 (and later to 3 for kicks)
#
# nothing else needs to change - no labels, no anything
```

```
# example 04: uses polyfit,poly1d # try with different degrees of fit

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

pd.set_option('display.max_rows',10)

def main():
  xs = np.array([-1,0,1,2])
  ys = np.array([-1,0,0,-2])

```

```
  SE: []
RMSE: []
```

	t	y	model	error
0	-1.0	-1.0	-1.000000e+00	-0.0000
1	0.0	0.0	2.053913e-15	-0.0000
2	1.0	0.0	1.942890e-15	-0.0000
3	2.0	-2.0	-2.000000e+00	-0.0000



3 conditioning of least squares

how accurately can least squares solution $\bar{x}$ be determined? consider a system where the true root is known.

example 05 van der monde matrix

let $x_1 = 2.0, x_2 = 2.2, \ldots, x_{11} = 4.0$ and set $y_i = 1 + x_i + x_i^2 + x_i^3 + x_i^4 + x_i^5 + x_i^6 + x_i^7$ for $1 < i < 11$. use normal equations to find least squares polynomial