Eli Weissler

Math189R SU17
Homework 1
Wednesday, May 17, 2017

Feel free to work with other students, but make sure you write up the homework and code on your own (no copying homework *or* code; no pair programming). Feel free to ask students or instructors for help debugging code or whatever else, though. The starter code for problem 2 part c and d can be found under the Resource tab on course website.

*Note:* You need to create a Github account for submission of the coding part of the homework. Please create a repository on Github to hold all your code and include your Github account username as part of the answer to problem 2.

**1 (Linear Transformation)** Let $\mathbf{y} = A\mathbf{x} + \mathbf{b}$ be a random vector. show that expectation is linear:

$$\mathbb{E}[\mathbf{y}] = \mathbb{E}[A\mathbf{x} + \mathbf{b}] = A\mathbb{E}[\mathbf{x}] + \mathbf{b}.$$

Also show that

$$\mathrm{cov}[\mathbf{y}] = \mathrm{cov}[A\mathbf{x} + \mathbf{b}] = A\mathrm{cov}[\mathbf{x}]A^\top = A\Sigma A^\top.$$

First show for expectation value

$$E[Ax+b] = \int (Ax+b) f(x)dx = \int A x f(x) dx + b\int f(x)dx = AE[x] + b$$

where $f(x)$ is the probability density function for $x$ and $b$ is a constant vector

Next show for covariance

$$\mathrm{cov}[Ax+b] = E[(Ax+b-E[Ax+b])(Ax+b-E[Ax+b])^T]$$

$$= E[(Ax+b-AE[x]-b)(Ax+b-AE[x]-b)^T]$$

$$= AE[(x-E[x])(x-E[x])^T]A^T$$

$$= A\Sigma A^T$$

where $\Sigma = \mathrm{cov}[x]$ and $A$ is a constant matrix ∎

**2** Given the dataset $\mathcal{D} = \{(x, y)\} = \{(0,1), (2,3), (3,6), (4,8)\}$

(a) Find the least squares estimate $y = \boldsymbol{\theta}^\top \mathbf{x}$ by hand using Cramer's Rule.

(b) Use the normal equations to find the same solution and verify it is the same as part (a).

(c) Plot the data and the optimal linear fit you found.

(d) Find randomly generate 100 points near the line with white Gaussian noise and then compute the least squares estimate (using a computer). Verify that this new line is close to the original and plot the new dataset, the old line, and the new line.

a) First let's calculate the individual components

$\Sigma x_i = 0+2+3+4 = 9$

$\Sigma y_i = 1+3+6+8 = 18$

$\Sigma x_i^2 = 0+4+9+16 = 29$

$(\Sigma x_i)^2 = 9^2 = 81$

$\Sigma x_i y_i = 0\cdot1 + 2\cdot3 + 3\cdot6 + 4\cdot8 = 0 + 6 + 18 + 32 = 56$

$m = \dfrac{4\cdot56 - 18\cdot9}{4\cdot29 - 81} = \dfrac{62}{35}$

$b = \dfrac{29\cdot18 - 9\cdot56}{4\cdot29 - 81} = \dfrac{18}{35}$

b-d) see attached code

# hw1pr2

June 9, 2020

b) Perform the linear regression using the normal equation

[9]:
```
import numpy as np

X = np.array([[0,1],
              [2,1],
              [3,1],
              [4,1]])
Y = np.array([1,3,6,8])
```

[16]:
```
theta = np.linalg.inv(X.T@X)@X.T@Y
print(theta)
```

```
[1.77142857 0.51428571]
```

Now let's verify that's the same as on paper

[18]:
```
62/35
```

[18]:
```
1.7714285714285714
```

[19]:
```
18/35
```

[19]:
```
0.5142857142857142
```

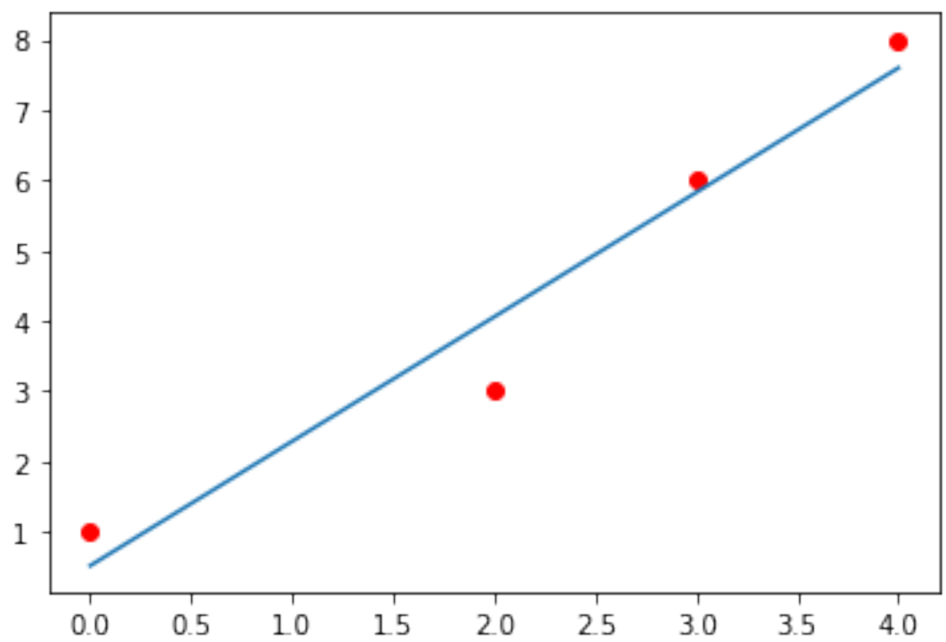
yayyyyy!

c) Plot the data and the linear fit

[20]:
```
import matplotlib.pyplot as plt
```

[26]:
```
plt.scatter(X[:,0],Y,c='r')
x = np.linspace(0,4,100)
y = 62*x/35 + 18/35
plt.plot(x,y)
```

[26]:
```
[<matplotlib.lines.Line2D at 0x1121b0320>]
```



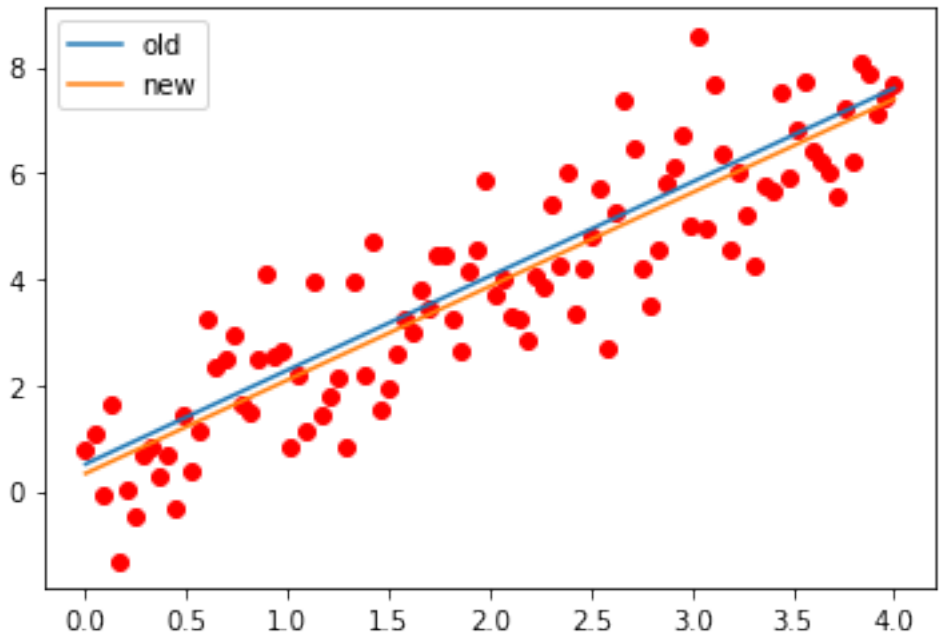
Looks decent to me.

d) generate 100 points with white gaussian noise and get a new least squares estimate. I'm using a variance of 1

[36]:
```
noise = np.random.normal(0,1,100)
X_noise = np.hstack((x.reshape(100,1),np.ones(100).reshape(100,1)))
Y_noise = y+noise
theta_noise = np.linalg.inv(X_noise.T@X_noise)@X_noise.T@Y_noise
```

[39]:
```
plt.scatter(x,Y_noise,c='r')
y_pred_noise = x*theta_noise[0]+theta_noise[1]
plt.plot(x,y)
plt.plot(x,y_pred_noise)
plt.legend(('old','new'))
```

[39]: <matplotlib.legend.Legend at 0x112114518>




Indeed it does look close to the original fit