hw6_q4e

March 11, 2021

[4]:
```
# Please do not edit this part
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(41)
# Please do not edit this part
```

0.0.1 Important Note

For ease of submission, we suggest downloading a PDF of your notebook and merging it with a PDF of your written/typed work, using platforms such as smallpdf, sodapdf, combinedpdf, etc. and then assigning only the part of the PDF containing the plot for question 4e ie. please **don't** assign the entire notebook PDF for question 4e. In addition to the figure from the Make a Plot, answer the following 2 questions listed in the main assignment and include your answers in your submission for Question 4e: - In your plot, does the simple linear regression model without an intercept term have the same slope as the model with an intercept term? - Describe one shortcoming for a simple linear regression model without an intercept term.

0.1 Question 4e

Now that you have done some analysis on whether some of the properties still hold for a simple linear regression model without an intercept term, let's actually run a simulation to confirm our findings.

Complete the following the functions and statements to plot a graph of the simple linear regression model with an intercept term v.s. one without an intercept term.

The slope ($\hat{\gamma}$) of your fitted line (without the intercept term) should be defined based on what you have shown in question 3:

$$\hat{\gamma} = \frac{\sum x_i y_i}{\sum x_i^2}$$

[5]:
```
# This function helps generate a synthesized dataset based on a given gamma␣
↪value
def generate_dataset(gamma, std=1, num_samples=100, with_intercept=True):
    X = np.random.random_sample(num_samples)
    e = np.random.randn(num_samples) * std
    intercept = -int(with_intercept) * 2
    Y = gamma * X + intercept + e
    return X, Y
```

Run the cell below to generate the synthesized dataset.

[6]:
```
dataset = generate_dataset(gamma=10)
X, Y = dataset
```

Complete the following function `calculate_gamma` that computes the value of the slope for your model based on whether or not an intercept term is specified within the model.

Hint: To calculate the slope for your simple linear regression model **when it has an intercept term**, check out the function np.corrcoef.

[18]:
```
import matplotlib.pyplot as plt

x = X[:,np.newaxis]
a, _, _, _ = np.linalg.lstsq(x, Y)
a[0]
```

```
<ipython-input-18-7f39414de375>:4: FutureWarning: `rcond` parameter will change
to the default of machine precision times ``max(M, N)`` where M and N are the
input matrix dimensions.
To use the future default and silence this warning we advise to pass
`rcond=None`, to keep using the old, explicitly pass `rcond=-1`.
  a, _, _, _ = np.linalg.lstsq(x, Y)
```

[18]:
```
6.811190340683249
```

[17]:
```
from scipy.stats import linregress
```

[17]:
```
9.895032383790364
```

[19]:
```
def calculate_gamma(X, Y, with_intercept):
    if with_intercept:
        gamma_hat = linregress(X, Y)[0]
    else:
        x = X[:,np.newaxis]
        a, _, _, _ = np.linalg.lstsq(x, Y)
        gamma_hat = a[0]
    return gamma_hat
```

Complete the following function `linear_model` that outputs a predicted value from your simple linear regression model based on whether or not an intercept is specified within the model.

[20]:
```
def linear_model(x, gamma, intercept, with_intercept):
    if with_intercept:
        Y_hat = intercept + x * gamma
    else:
        Y_hat = x * gamma
    return Y_hat
```

[21]:
```
# Compute the estimated slope with & without an intercept
gamma_with_intercept = calculate_gamma(X, Y, with_intercept=True)
gamma_without_intercept = calculate_gamma(X, Y, with_intercept=False)

# Compute the estimtaed intercept
intercept_hat = np.mean(Y) - gamma_with_intercept * np.mean(X)

# Compute the predicted y values using a simple linear regression model with &␣
 ↪without an intercept
Y_hat_with_intercept = linear_model(X, gamma_with_intercept, intercept_hat,␣
 ↪True)
Y_hat_without_intercept = linear_model(X, gamma_without_intercept,␣
 ↪intercept_hat, False)

# Compute the residual vector for all of the predicted y values from the model␣
 ↪without an intercept
residual_without_intercept = Y - Y_hat_without_intercept
```

```
<ipython-input-19-875a878ac4c3>:6: FutureWarning: `rcond` parameter will change
to the default of machine precision times ``max(M, N)`` where M and N are the
input matrix dimensions.
To use the future default and silence this warning we advise to pass
`rcond=None`, to keep using the old, explicitly pass `rcond=-1`.
  a, _, _, _ = np.linalg.lstsq(x, Y)
```

0.1.1 A Quick Numerical Check

In parts 4a, 4b, and 4c, you have answered a few True/False questions about whether certain properties still hold for the simplified linear regression model without an intercept term. Now would be a great opportunity for you to actually numerically check if the synthesized dataset empirically agrees with your answers to the earlier parts of this question.

Run through the following few cells and see if it matches what you originally expected.

Proposed statement from 4a

$$\sum_{i=1}^{n} e_i = 0$$

[22]:
```
np.isclose(np.sum(residual_without_intercept), 0)
```

[22]:
```
False
```

Proposed statement from 4b The column vector $\vec{x}$ and the residual vector e are orthogonal.

[23]:
```
np.isclose(np.sum(X * residual_without_intercept), 0)
```

[23]:
```
True
```

Proposed statement from 4c The predicted response vector $\hat{\mathbb{Y}}$ and the residual vector e are orthogonal.

[24]:
```
np.isclose(np.sum(Y_hat_without_intercept * residual_without_intercept), 0)
```

[24]:
```
True
```

0.1.2 Make a Plot

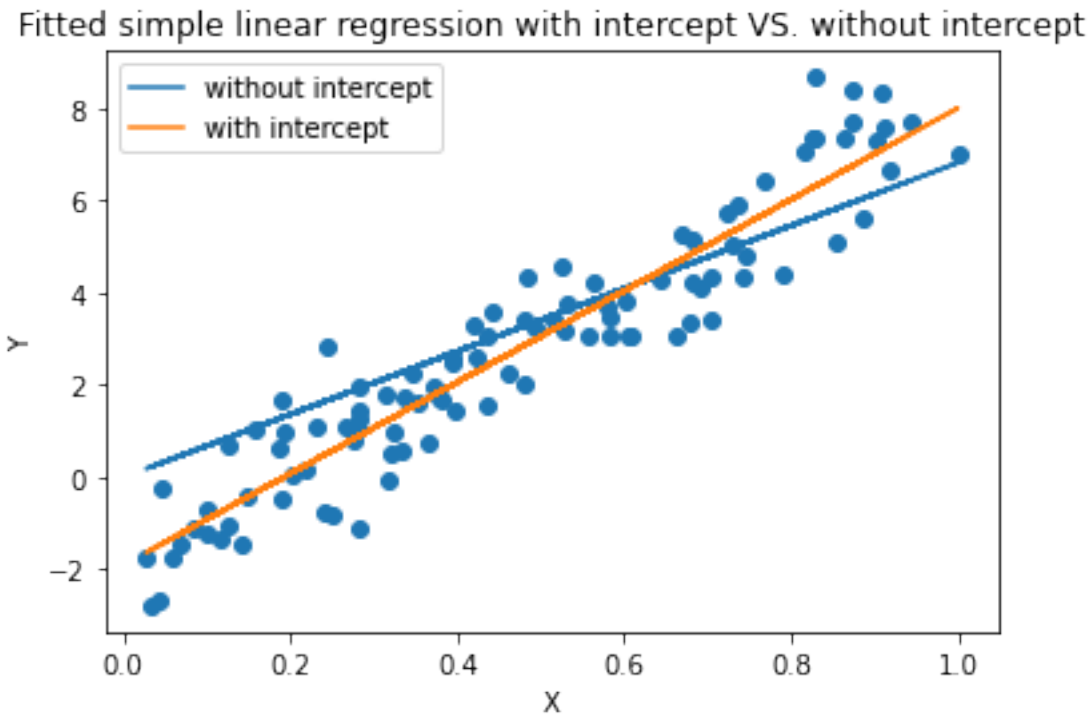
Finally, let's create a plot below comparing our fitted simple linear regression model on the observed data for when it includes an intercept v.s. when it does not include an intercept. Your plot should include at least the following: - A scatter plot of all the observed data - A line plot for the simple linear regression model without an intercept - A line plot for the simple linear regression model with an intercept - An appropriate title, a legend labeling which line contains an intercept, and labels for both axes

You should also mark the point $(\bar{x}, \bar{y})$ in your plot with a different marker style, color, or size.

Note: As a reminder, $\bar{x}$ represents the average of all of the x values in our observed data. The same goes for $\bar{y}$.

[29]:
```
plt.scatter(X, Y)
plt.plot(X, Y_hat_without_intercept,label="without intercept")
plt.plot(X, Y_hat_with_intercept,label="with intercept")
plt.title("Fitted simple linear regression with intercept VS. without␣
 ↪intercept")
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc="upper left")
```

[29]:
```
<matplotlib.legend.Legend at 0x7f6f18551f40>
```



- In your plot, does the simple linear regression model without an intercept term have the same slope as the model with an intercept term?
- Describe one shortcoming for a simple linear regression model without an intercept term.

- The model without intercept term has the slop coefficient of 6.811; the model with intercept term has the slop coeefficient term of 9.895
- Dropping the intercept in a regression model forces the regression line to go through the origin, just because we're forcing the line through the origin, it's not because it fits the data better. We might be creating that steepness artificially, therefore a more statistically significant model isn't better if it's inaccurate.

[ ]: