MFDS_A5_Q4

November 10, 2024

1 Shubharthak Sangharasha

2 Student ID: a1944839

3 MATHS7027 Mathematical Foundations of Data Science

3.1 Assignment 5 - Question 4

Consider a system with a predictor variable x and a response variable y. Suppose we collect data as pairs (x, y), and observe the following data:

$$(2,5),\ (-1,3),\ (0,2)$$

[1]:
```python
#First I will import the necessary libraries and add the path to the␣
↪site-packages directory [IGNORE THIS LINE]
import sys
sys.path.append('/home/shubharthak/miniconda3/lib/python3.12/site-packages')␣
↪#IGNORE THIS LINE

#Now I will import the necessary libraries
import numpy as np
import matplotlib.pyplot as plt
```

3.2 4(a)

Suppose we wish to fit a linear regression model to describe this data. That is, we want to find the equation of the line of best fit:

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

where - x_i is the i^{th} observation of the predictor variable; - y_i is the i^{th} observation of the response variable; - $\hat{\beta}_0$ is our best estimate for the intercept; and - $\hat{\beta}_1$ is our best estimate for the slope.

3.2.1 (i)

Enter the design matrix X and the response vector y for this linear regression.

[2]:
```python
#First I will create the design matrix X and the response vector y
X = np.array([
    [1, 2],
    [1, -1],
    [1, 0]
])
y = np.array([5, 3, 2])

#Now I will print the design matrix X and the response vector y
print("Design matrix X:")
print(X)
print("Response vector y:")
print(y)
```

```
Design matrix X:
[[ 1  2]
 [ 1 -1]
 [ 1  0]]
Response vector y:
[5 3 2]
```

3.2.2 (ii)

Use matrix operations in Python to calculate the vector $\hat{\beta} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix}$. Print your results.

Hint: You may find the material in Practical 4 (Week 7) helpful.

[6]:
```python
#First I will calculate the transpose of the design matrix X
X_transpose = X.T

#Now I will calculate the vector beta_hat
beta_hat = np.linalg.inv(X_transpose @ X) @ X_transpose @ y

#Now I will print the vector beta_hat
print("Vector beta_hat:")
print(beta_hat)
```

```
Vector beta_hat:
[3.07142857 0.78571429]
```

3.3 4(b)

Suppose we now wish to fit a quadratic regression model to the same data. That is, y now depends on both x and x^2:

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{\beta}_2 x_i^2$$

Use matrix operations in Python to find the values of $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$. Print your results.

```
[7]: #First I will create the design matrix X for the quadratic regression model
X_quad = np.array([
    [1, 2, 2**2],
    [1, -1, (-1) ** 2],
    [1, 0, 0 ** 2]
])

#Now I will print the design matrix X_quad
print("Design matrix X_quad:")
print(X_quad)

#Now I will calculate the vector beta_hat_quad
X_quad_transpose = X_quad.T
beta_hat_quad = np.linalg.inv(X_quad_transpose @ X_quad) @ X_quad_transpose @ y

#Now I will print the Estimated coefficients (beta_hat)
print("Estimated coefficients for the quadratic regression model (beta_hat):")
print(beta_hat_quad)
```

```
Design matrix X_quad:
[[ 1  2  4]
 [ 1 -1  1]
 [ 1  0  0]]
Estimated coefficients for the quadratic regression model (beta_hat):
[ 2.         -0.16666667  0.83333333]
```

3.4 4(c)

Produce a plot showing: - the data, x and y; - your linear regression model; and - your quadratic regression model.

Hint: You may want to create functions to represent your linear and quadratic regression models. The material from Practical 2 (Week 3) may be helpful.

```
[10]: #original data points
x = np.array([2, -1, 0])
y = np.array([5, 3, 2])

#linear regression coefficients
beta0_linear, beta1_linear = beta_hat

#quadratic regression coefficients
beta0_quadratic, beta1_quadratic, beta2_quadratic = beta_hat_quad

def linear_model(x: np.ndarray) -> np.ndarray:
    return beta0_linear + beta1_linear * x

def quadratic_model(x: np.ndarray) -> np.ndarray:
```

```
    return beta0_quadratic + beta1_quadratic * x + beta2_quadratic * x**2

#now i will generate x values for the plot
x_plot = np.linspace(min(x) - 1, max(x) + 1, 100)

#now i will calculate corresponding y values for the linear and quadratic models
y_linear = linear_model(x_plot)
y_quadratic = quadratic_model(x_plot)

#now i will plot the data, the linear regression model, and the quadratic␣
 ↪regression model
plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='blue', label='Data Points')
plt.plot(x_plot, y_linear, color='red', label='Linear Regression')
plt.plot(x_plot, y_quadratic, color='green', label='Quadratic Regression')
plt.legend()
plt.xlabel('Predictor Variable x')
plt.ylabel('Response Variable y')
plt.title('Linear and Quadratic Regression Models by Shubharthak')
plt.grid(True)
plt.show()
```