

Assignment 3: Maximum Likelihood Estimation

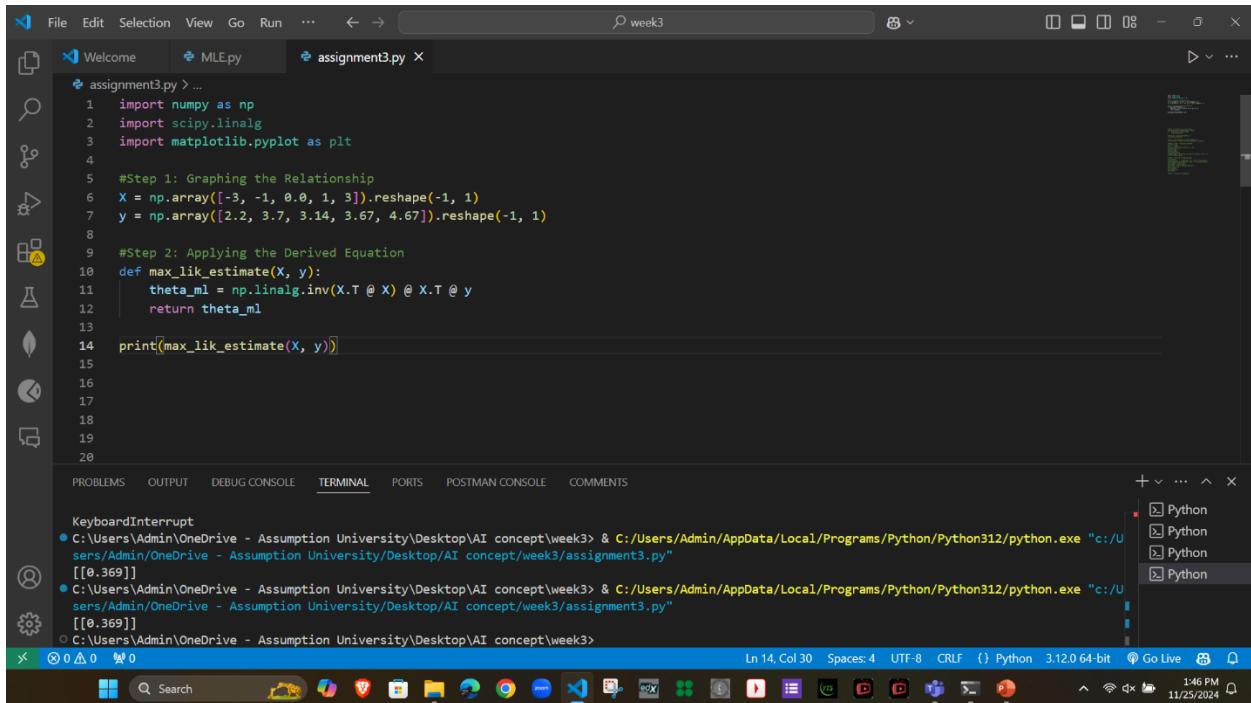
Step 1: Graphing the Relationship

- Python code and relationship between x and y graph

The screenshot shows a Jupyter Notebook environment with the following details:

- Code Cell:** The code cell contains Python code to generate a scatter plot of training data points. The X-axis ranges from -5 to 5, and the Y-axis ranges from 2.5 to 4.5. The data points are represented by blue plus signs.
- Output Cell:** The output cell displays the generated scatter plot titled "Plot of Training Data Set". The plot has "X-axis (x)" on the horizontal axis and "y-axis (y)" on the vertical axis. The data points are approximately at (-3.5, 2.2), (-1.5, 3.7), (0.5, 3.14), (1.5, 3.67), and (3.5, 4.67).
- Terminal:** The terminal window shows the command run: `C:\Users\Admin\OneDrive - Assumption University\Desktop\AI concept\week3 & C:/Users/Admin/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Admin/OneDrive - Assumption University/Desktop/AI concept/week3/assignment3.py"`.
- Status Bar:** The status bar at the bottom indicates the current line (Ln 17), column (Col 11), spaces (Spaces: 4), encoding (UTF-8), and Python version (Python 3.12.0 64-bit). It also shows the date and time (11/25/2024, 1:43 PM).

Step 2: Applying the Derived Equation



The screenshot shows a code editor window titled "assignment3.py". The code is as follows:

```
assignment3.py > ...
1 import numpy as np
2 import scipy.linalg
3 import matplotlib.pyplot as plt
4
5 #Step 1: Graphing the Relationship
6 X = np.array([-3, -1, 0.0, 1, 3]).reshape(-1, 1)
7 y = np.array([2.2, 3.7, 3.14, 3.67, 4.67]).reshape(-1, 1)
8
9 #Step 2: Applying the Derived Equation
10 def max_likelihood(X, y):
11     theta_ml = np.linalg.inv(X.T @ X) @ X.T @ y
12     return theta_ml
13
14 print(max_likelihood(X, y))
15
16
17
18
19
20
```

The code uses NumPy and SciPy libraries to define a function `max_likelihood` that calculates the maximum likelihood estimate for a linear regression model. The function takes `X` (input features) and `y` (target values) as arguments and returns the estimated parameters `theta_ml`. The code then prints the result of calling this function with the provided data points.

Theta value is 0.3689

Step 3: Plotting the Initial Model

- Applying Theta obtained from Step2 in Step3

The screenshot shows a Jupyter Notebook environment with the following details:

- Left Panel (Code Cell):** The code cell contains Python code for linear regression. It includes importing numpy, defining functions for maximum likelihood estimation, prediction, and plotting, and finally plotting the data points and the initial model line.
- Right Panel (Plot):** A figure titled "Figure 1" displays a scatter plot of the training data set. The x-axis is labeled "X-axis (x)" and ranges from -4 to 4. The y-axis is labeled "y-axis (y)" and ranges from -2 to 4. Blue '+' markers represent the "Data points". A solid orange line represents the initial model fit, labeled as $y = 0.3689999999999994x$.
- Bottom Status Bar:** Shows the current line (Ln 15), column (Col 41), and other terminal settings like spaces (4), encoding (UTF-8), and Python version (3.12.0 64-bit).

Step 4: Improving the Model

The screenshot shows a code editor window titled "week3" with several tabs open. The active tab is "assignment3.py". The code in the editor is as follows:

```
20     print(theta_ml[0][0])
21
22 Xtest = np.linspace(-5, 5, 100).reshape(-1,1)
23 ml_prediction = predict_with_estimate(Xtest, theta_ml)
24
25 #Step 4: Step 4: Improving the Model
26 print("=*10)
27 N, D = X.shape
28 X_aug = np.hstack([np.ones((N, 1)), X])
29 print("X_aug = ")
30 print(X_aug)
31 print("=*10)
32 # Recompute Theta
33 theta_ml_bias = np.linalg.inv(X_aug.T @ X_aug) @ X_aug.T @ y
34 print(theta_ml_bias)
```

The output console below the editor shows the results of the execution:

```
0.3689999999999994
=====
X_aug =
[[ 1. -3.]
 [ 1. -1.]
 [ 1.  0.]
 [ 1.  1.]
 [ 1.  3.]]
=====
[[3.476]
 [0.369]]
```

The status bar at the bottom indicates the file path "C:\Users\Admin\OneDrive - Assumption University\Desktop\AI concept\week3\assignment3.py", line 45, column 1, spaces: 4, UTF-8, CRLF, Python 3.12.0 64-bit, and the current time "1:51 PM 11/25/2024".

Theta values are 3.476 and 0.369

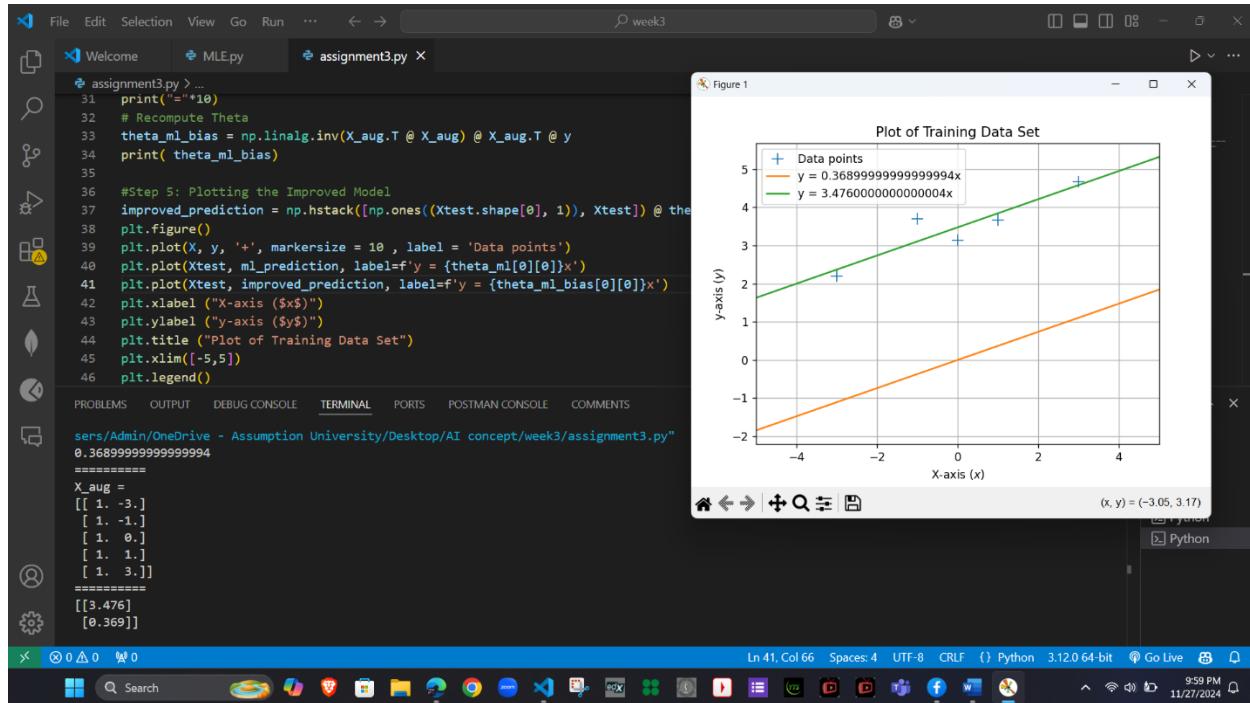
Explanation

In Step 2, the model assumes the line passes through the origin, so there is no intercept term. The parameter θ_{ml} is solely the slope, which might not fully capture the relationship if the line does not pass through the origin.

In Step 4, by adding a bias term, the model accounts for a possible offset in the data. The resulting θ_{ml} -bias includes both the intercept and slope, potentially providing a more accurate fit to the data.

Step 5: Plotting the Improved Model

- Use new Theta values from step4 in linear equation and plot the graph



Step 6: Analytical Explanation

- **Step 2 Limitation:** Assumes the line passes through the origin ($y=\theta x$), which may not fit real-world data where y -intercept isn't zero.
- **Bias Term Role:** Adding ones enables the model to estimate a non-zero intercept ($y=\theta_0+\theta_1 x$), making it more flexible.
- **Improved Fit:** Accounts for vertical shifts, reducing residual errors and better aligning the line with the data.
- **Broader Model Capability:** Supports relationships with both slope and intercept, fitting a wider range of linear patterns.
- **Mathematical Impact:** Minimizes errors by recalculating parameters (θ_0 and θ_1) for a closer match to data.