Alex Rios 801320278

ECGR-4105 Summer 2025

Report for assignment1

GitHub Repository: https://github.com/TheluckyEngineer101/AR-Assignment1-Sum25

---

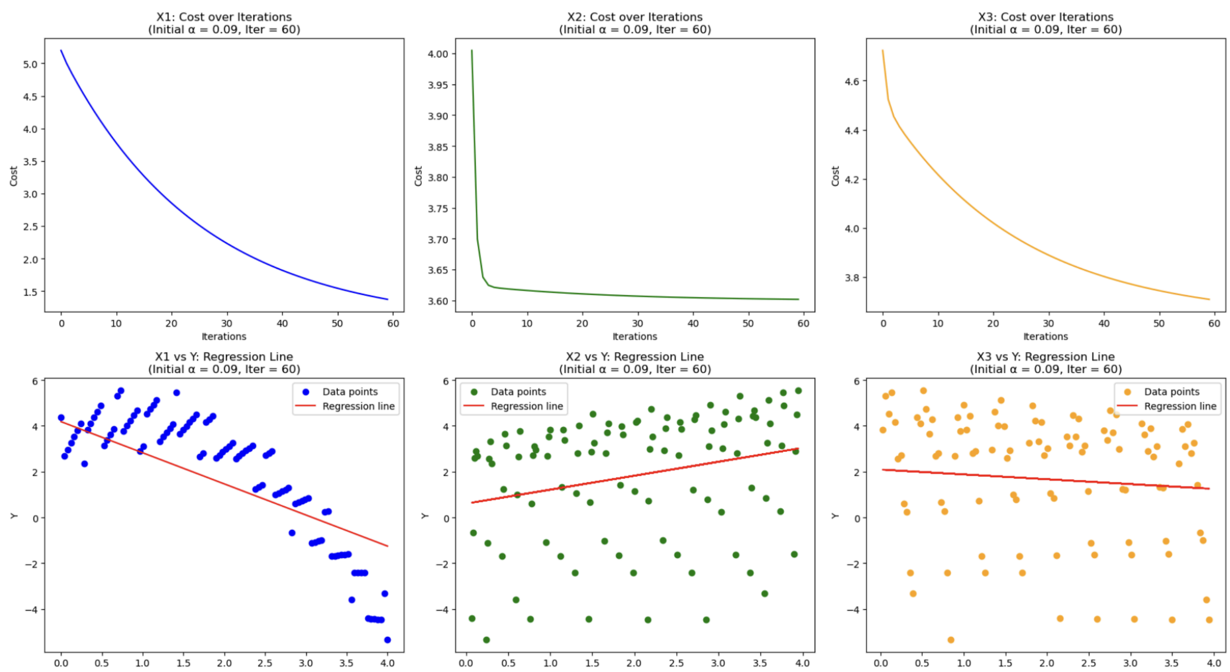
Problem 1:

alpha = 0.09, iterations = 60

For X1:h(x) = 4.1850 + -1.3600x

For X2:h(x) = 0.6015 + 0.6100x

For X3:h(x) = 2.0909 + -0.2118x




**Evaluation of Cost Across Features:**

Following the training of distinct linear regression models for each explanatory variable (X1, X2, and X3), the cost over iterations plots and regression lines shows that X1 had the lowest final cost. This implies that, compared to X2 and X3, X1 has the strongest linear association with the output variable Y. While X2 and X3 level out at higher cost values, X1's cost drops more and converges more successfully in the plotted regression findings.

**Learning Rate and Iteration Impact:**

Based on testing with values ranging from 0.01 to 0.1, I decided to begin my implementation with a learning rate (Alpha) of 0.05 and 40 iterations. This decision prevented the bad lines such as 0.01, it also produced a consistent convergence without exceeding the minimum.

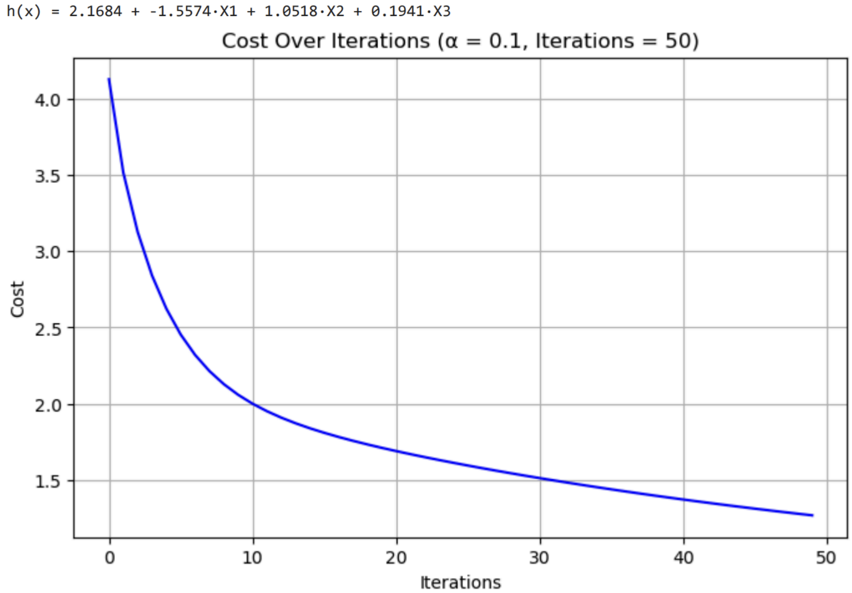
My code's experimental graphs show that a learning rate of 0.05 achieved a nice balance between allowing for adequate convergence within a reasonable number of iterations and maintaining stability across all three features. The general pattern shows that moderate learning rates are safer when training models separately, despite the fact that each variable reacts differently during training.

---

Problem 2:

alpha = 0.1, iterations = 50

```
Final Hypothesis:
h(x) = 2.1684 + -1.5574·X1 + 1.0518·X2 + 0.1941·X3
```




```
Predictions for future points:
(1, 1, 1) → 1.8569
(2, 0, 4) → -0.1699
(3, 2, 1) → -0.2061
```

**Impact of Learning Rate and Iteration Count**

I kept experimenting with learning rates between 0.01 and 0.1 when applying linear regression to all three variables (X1, X2, and X3) at the same time, just like I did in Problem 1. In the end, I decided on a learning rate of 0.05 and 50 iterations as they produced a steady and even cost reduction devoid of volatility or divergence.

Higher learning rates, such as 0.1, did enable the model to lower cost more quickly in the early iterations, but they frequently leveled off at a higher number. However, for the same number of steps, a learning rate of 0.05 provided a smaller final loss of about 1.61 but a significantly slower cost decline. The outcomes were more consistent and broadly applicable, but required more iterations to achieve that minimum.

---

1. Data Loading and Visualization

In [1]:
```
# Alex Rios 801320278
# Assignment1 Summer 2025

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd

# go ahead load and find the path for my dataset
df = pd.read_csv('../Datasets/HW1.csv')
df.head() # To get first n rows from the dataset default value of n is 5
M=len(df)
M
print(f"Loaded {M} samples.") # rather than a random "100" I will add addtional info on what 100 means.
print(df.head()) # from the df.head() I went ahead and added print to make it show our first 4 rows
```

```
Loaded 100 samples.
         X1        X2        X3         Y
0  0.000000  3.440000  0.440000  4.387545
1  0.040404  0.134949  0.888485  2.679650
2  0.080808  0.829899  1.336970  2.968490
3  0.121212  1.524848  1.785455  3.254065
4  0.161616  2.219798  2.233939  3.536375
```

Additional data loading and statistics

Approach: Used df.describe() this function is used for Generating descriptive stats for each column such as [count, mean, std, min/max and percentiles]

Result:

```
[2]: df.describe()
```

[2]:

| | X1 | X2 | X3 | Y |
|---|---|---|---|---|
| **count** | 100.000000 | 100.000000 | 100.000000 | 100.000000 |
| **mean** | 2.000000 | 2.000000 | 1.960000 | 1.851276 |
| **std** | 1.172181 | 1.172154 | 1.163005 | 2.774643 |
| **min** | 0.000000 | 0.070303 | 0.027879 | -5.332455 |
| **25%** | 1.000000 | 0.979394 | 0.952121 | 0.527533 |
| **50%** | 2.000000 | 2.009697 | 1.949091 | 2.879003 |
| **75%** | 3.000000 | 3.040000 | 2.946061 | 3.925389 |
| **max** | 4.000000 | 3.949091 | 3.943030 | 5.545892 |

```python
# Gradient Descent Function
def GradientDescent(X, Y, Theta, Alpha, Iterations):
    M = len(X)  # number of training examples
    CostHistory = []  # Track cost at each iteration

    for i in range(Iterations):
        Predictions = X @ Theta
        Errors = Predictions - Y
        Gradient = (Alpha / M) * (X.T @ Errors)
        Theta = Theta - Gradient
        CostHistory.append(ComputeCost(X, Y, Theta))

    return Theta, CostHistory

# Cost Function
def ComputeCost(X, Y, Theta):
    M = len(Y)
    Predictions = X @ Theta
    Errors = Predictions - Y
    SquaredErrors = Errors ** 2
    return (1 / (2 * M)) * np.sum(SquaredErrors)
```

Had to find the mean squared error cost function J(θ).

Approach: we used compute_cost(X, y, theta) that computes .

```python
#  Cost Over Iterations
plt.subplot(2, 3, Idx)
plt.plot(range(Iterations), CostLog, color=Color)
plt.xlabel("Iterations")
plt.ylabel("Cost")
plt.title(f"{Col}: Cost over Iterations\n(Initial α = {Alpha}, Iter = {Iterations})")

#  Regression Line
plt.subplot(2, 3, Idx + 3)
plt.scatter(Xvals[:, 1], Y, color=Color, label="Data points")
plt.plot(Xvals[:, 1], Xvals @ ThetaParams, color="red", label="Regression line")
plt.xlabel(Col)
plt.ylabel("Y")
plt.title(f"{Col} vs Y: Regression Line\n(Initial α = {Alpha}, Iter = {Iterations})")
plt.legend()
```

---

Assumptions & Notes

- All code, results, and plots available in the GitHub repository above.

DISCLAIMER

This work involved the use of ChatGPT by OpenAI as a supportive tool for brainstorming, formatting assistance, and refining explanations. All logic, analysis, and conclusions along with the overall development of this submission was completed independently by me, based on both my own reasoning and the material provided for this assignment. I also conducted my own research and referenced publicly available resources and open source examples to ensure correct implementation of functions and concepts. The model served solely to enhance clarity and presentation, not as a replacement for my own understanding or decision making.