ECGR_4105_HW0

September 20, 2022

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

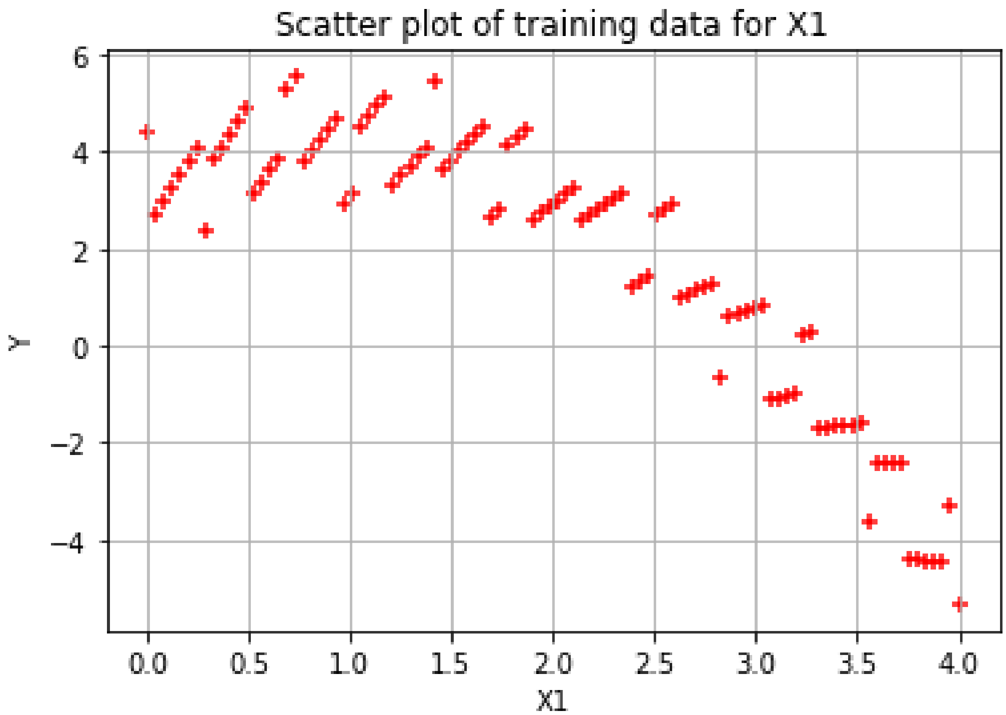
[3]:
```
df = pd.read_csv (r'C:\Users\homer\OneDrive\Documents\School Folder\D3.csv')
print (df)
```

```
          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
..       …         …         …         …
95  3.838384  1.460202  3.046061 -4.440595
96  3.878788  2.155152  3.494545 -4.458663
97  3.919192  2.850101  3.943030 -4.479995
98  3.959596  3.545051  0.391515 -3.304593
99  4.000000  0.240000  0.840000 -5.332455

[100 rows x 4 columns]
```

[4]:
```
x1 = df.values[:,0]
x2 = df.values[:,1]
x3 = df.values[:,2]
Y = df.values[:,3]
m = len(Y)
```

[5]:
```
plt.scatter(x1,Y, color='red',marker= '+')
plt.grid()
plt.rcParams["figure.figsize"] = (10,6)
plt.xlabel('X1')
plt.ylabel('Y')
plt.title('Scatter plot of training data for X1')
```

[5]:
```
Text(0.5, 1.0, 'Scatter plot of training data for X1')
```



[6]:
```
plt.scatter(x2,Y, color='red',marker= '+')
plt.grid()
plt.rcParams["figure.figsize"] = (10,6)
plt.xlabel('X2')
plt.ylabel('Y')
plt.title('Scatter plot of training data for X2')
```

[6]: Text(0.5, 1.0, 'Scatter plot of training data for X2')

[7]:
```
plt.scatter(x3,Y, color='red',marker= '+')
plt.grid()
plt.rcParams["figure.figsize"] = (10,6)
plt.xlabel('X3')
plt.ylabel('Y')
plt.title('Scatter plot of training data for X3')
```

[7]: Text(0.5, 1.0, 'Scatter plot of training data for X3')

[8]:
```python
X_0 = np.ones((m, 1))
```

[9]:
```python
X_1 = x1.reshape(m, 1)
X_2 = x2.reshape(m, 1)
X_3 = x3.reshape(m, 1)
```

[10]:
```python
X1 = np.hstack((X_0, X_1))
X2 = np.hstack((X_0, X_2))
X3 = np.hstack((X_0, X_3))
```

[11]:
```python
theta = np.zeros(2)
```

[12]:
```python
def compute_cost(X1, Y, theta):
    predictions = X1.dot(theta)
    errors = np.subtract(predictions, Y)
    sqrErrors = np.square(errors)
    J = 1 / (2 * m) * np.sum(sqrErrors)
    return J
```

[13]:
```python
cost = compute_cost(X1, Y, theta)
print('The cost for given values of theta_0 and theta_1 =', cost)
```

```
The cost for given values of theta_0 and theta_1 = 5.524438459196242
```

[14]:
```
def compute_cost(X2, Y, theta):
    predictions = X2.dot(theta)
    errors = np.subtract(predictions, Y)
    sqrErrors = np.square(errors)
    J = 1 / (2 * m) * np.sum(sqrErrors)
    return J
```

[15]:
```
cost = compute_cost(X2, Y, theta)
print('The cost for given values of theta_0 and theta_1 =', cost)
```

```
The cost for given values of theta_0 and theta_1 = 5.524438459196242
```

[16]:
```
def compute_cost(X3, Y, theta):
    predictions = X3.dot(theta)
    errors = np.subtract(predictions, Y)
    sqrErrors = np.square(errors)
    J = 1 / (2 * m) * np.sum(sqrErrors)
    return J
```

[17]:
```
cost = compute_cost(X3, Y, theta)
print('The cost for given values of theta_0 and theta_1 =', cost)
```

```
The cost for given values of theta_0 and theta_1 = 5.524438459196242
```

[18]:
```
def gradient_descent(X1, Y, theta, alpha, iterations):
    cost_history = np.zeros(iterations)
    for i in range(iterations):
        predictions = X1.dot(theta)
        errors = np.subtract(predictions, Y)
        sum_delta = (alpha / m) * X1.transpose().dot(errors);
        theta = theta - sum_delta;
        cost_history[i] = compute_cost(X1, Y, theta)

    return theta, cost_history
```

[19]:
```
def gradient_descent(X2, Y, theta, alpha, iterations):
    cost_history = np.zeros(iterations)
    for i in range(iterations):
        predictions = X2.dot(theta)
        errors = np.subtract(predictions, Y)
        sum_delta = (alpha / m) * X2.transpose().dot(errors);
        theta = theta - sum_delta;
        cost_history[i] = compute_cost(X2, Y, theta)

    return theta, cost_history
```

[20]:
```
def gradient_descent(X3, Y, theta, alpha, iterations):
    cost_history = np.zeros(iterations)
```

```
    for i in range(iterations):
        predictions = X3.dot(theta)
        errors = np.subtract(predictions, Y)
        sum_delta = (alpha / m) * X3.transpose().dot(errors);
        theta = theta - sum_delta;
        cost_history[i] = compute_cost(X3, Y, theta)

    return theta, cost_history
```

[21]:

```
theta = [0., 0.]
iterations = 1500;
alpha = 0.01;
```

[22]:

```
theta, cost_history = gradient_descent(X1, Y, theta, alpha, iterations)
print('Final value of theta =', theta)
print('cost_history =', cost_history)
```

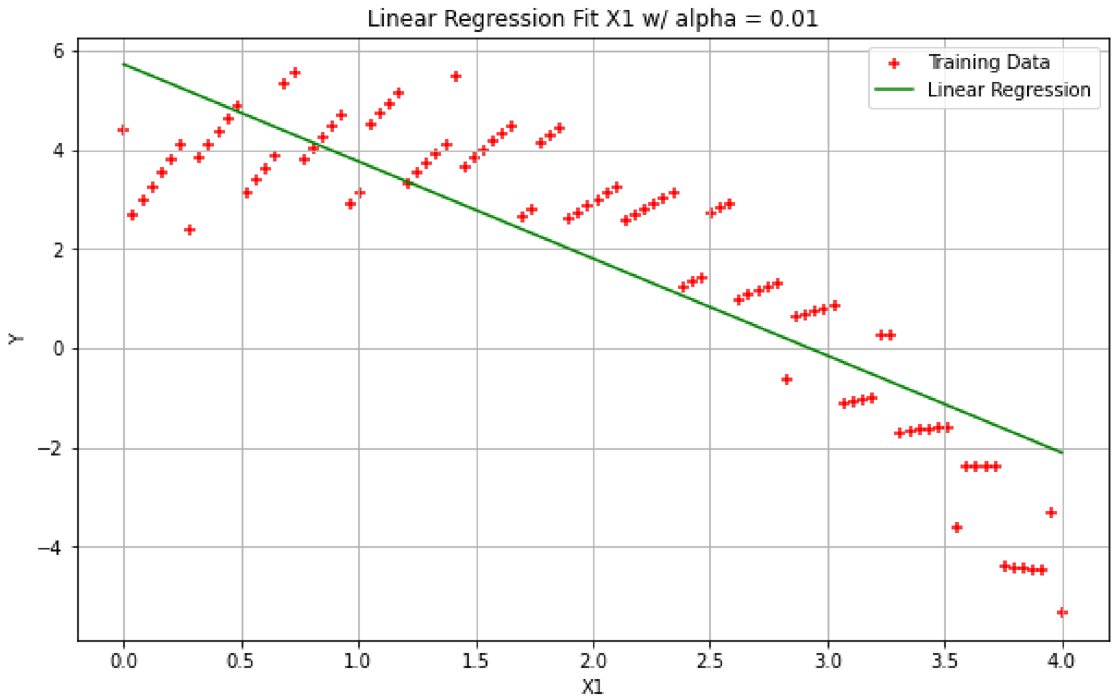
```
Final value of theta = [ 5.71850653 -1.9568206 ]
cost_history = [5.48226715 5.44290965 5.40604087 … 0.99063932 0.99061433
0.99058944]
```

[23]:

```
plt.scatter(X1[:,1], Y, color='red', marker= '+', label= 'Training Data')
plt.plot(X1[:,1],X1.dot(theta), color='green', label='Linear Regression')

plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('X1')
plt.ylabel('Y')
plt.title('Linear Regression Fit X1 w/ alpha = 0.01')
plt.legend()
```

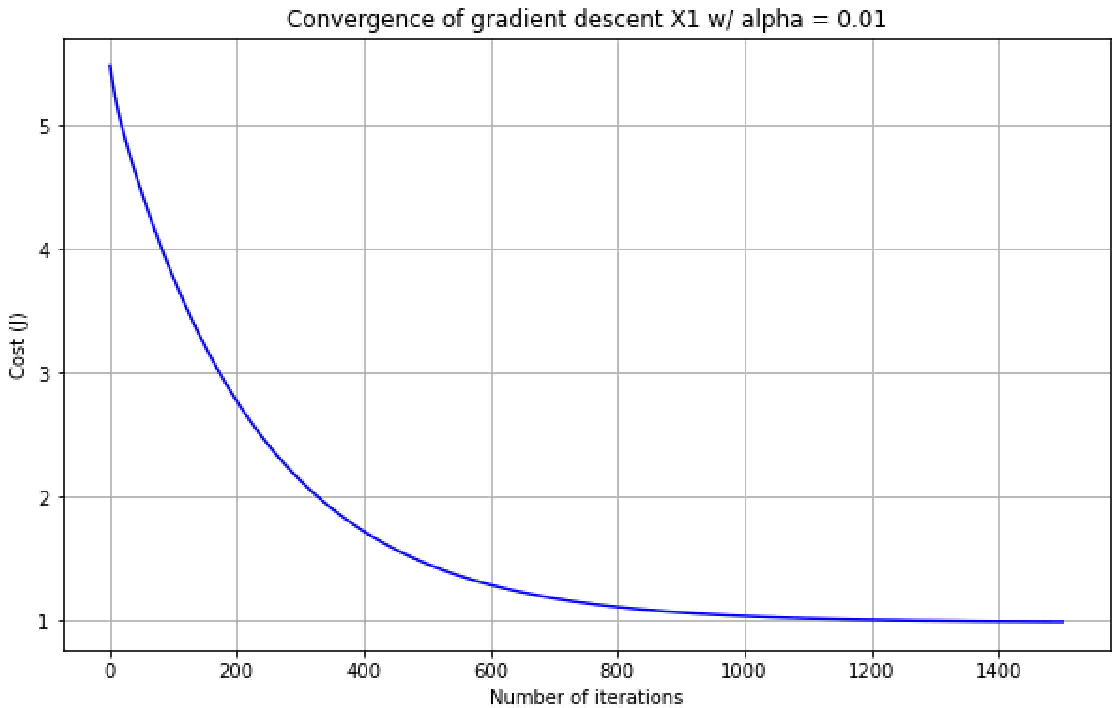
[23]:

```
<matplotlib.legend.Legend at 0x1c7f153f6d0>
```



[24]:
```
plt.plot(range(1, iterations + 1),cost_history, color='blue')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent X1 w/ alpha = 0.01')
```

[24]:
```
Text(0.5, 1.0, 'Convergence of gradient descent X1 w/ alpha = 0.01')
```



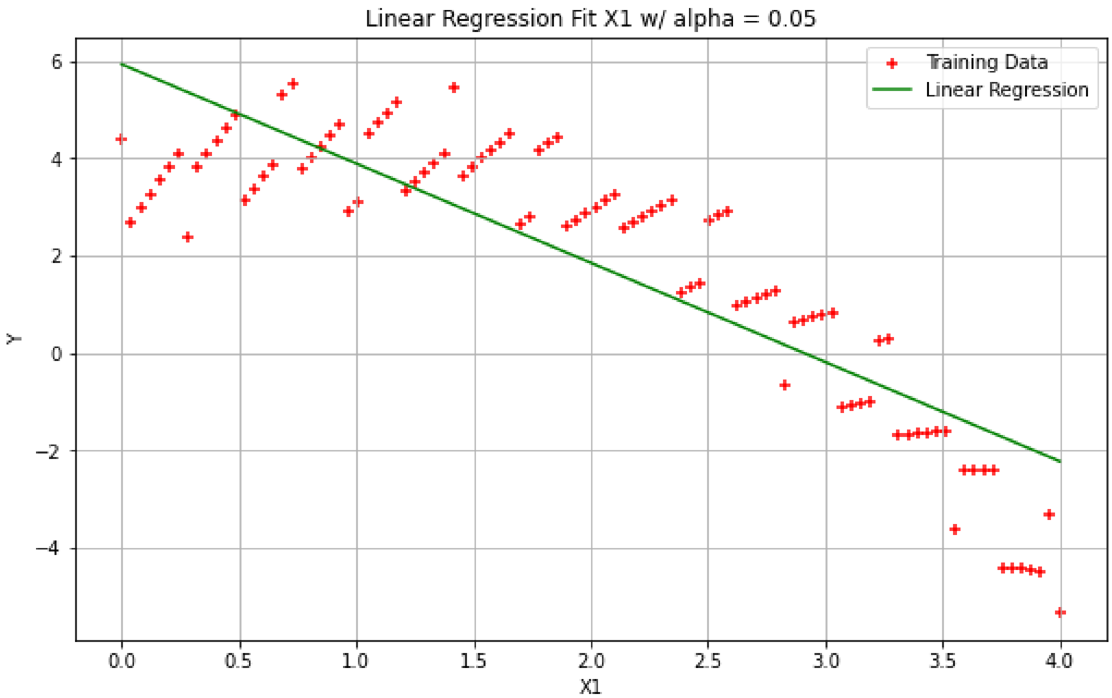
[25]:
```
theta = [0., 0.]
iterations = 1500;
alpha = 0.05;
```

[26]:
```
theta, cost_history = gradient_descent(X1, Y, theta, alpha, iterations)
print('Final value of theta =', theta)
print('cost_history =', cost_history)
```

```
Final value of theta = [ 5.9279486  -2.03833651]
cost_history = [5.32852962 5.18676104 5.07204859 … 0.98499308 0.98499308
0.98499308]
```

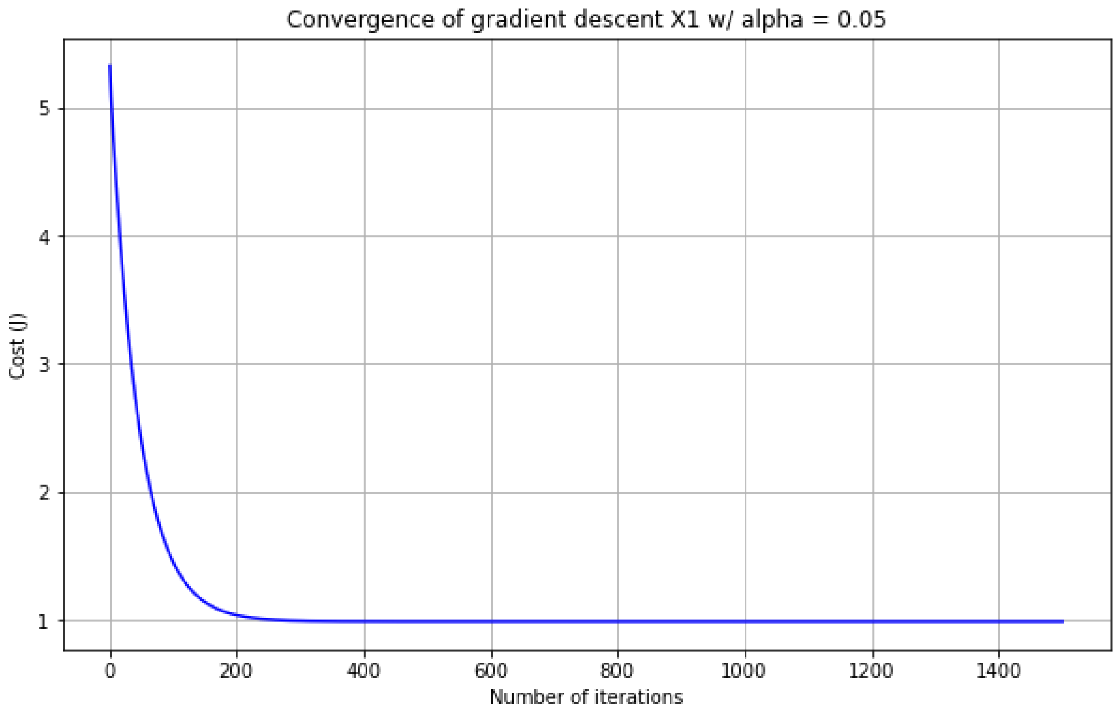
[27]:
```
plt.scatter(X1[:,1], Y, color='red', marker= '+', label= 'Training Data')
plt.plot(X1[:,1],X1.dot(theta), color='green', label='Linear Regression')

plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('X1')
plt.ylabel('Y')
plt.title('Linear Regression Fit X1 w/ alpha = 0.05')
plt.legend()
```

[27]: <matplotlib.legend.Legend at 0x1c7f15af340>



[28]:
```
plt.plot(range(1, iterations + 1),cost_history, color='blue')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent X1 w/ alpha = 0.05')
```

[28]:
```
Text(0.5, 1.0, 'Convergence of gradient descent X1 w/ alpha = 0.05')
```



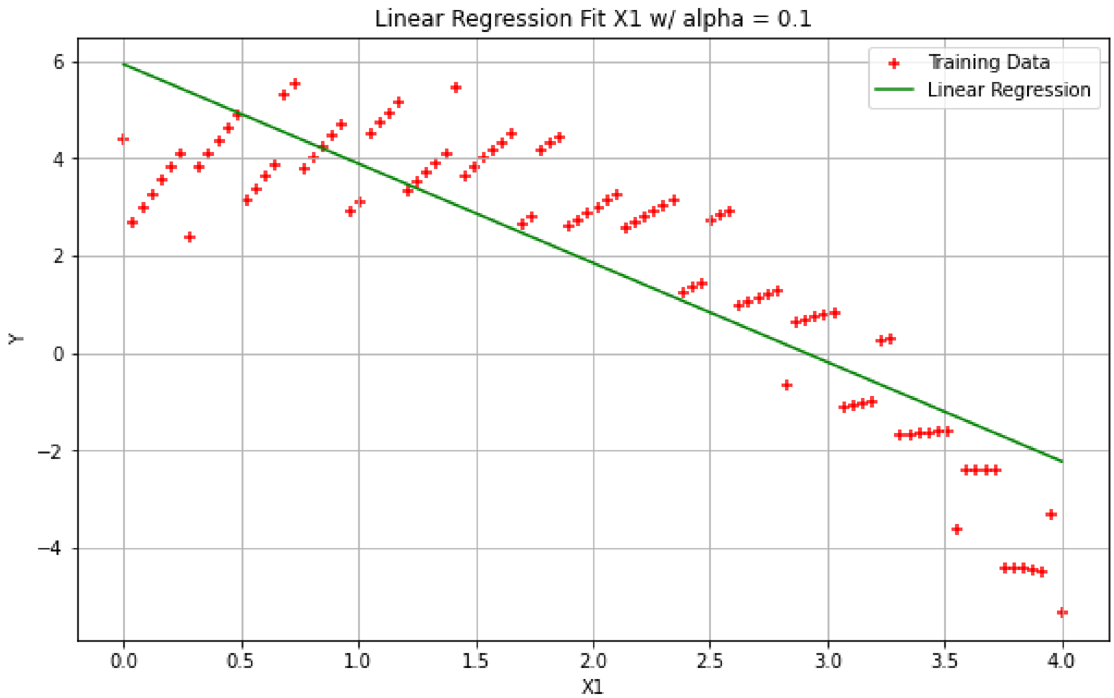

```
[29]: theta = [0., 0.]
      iterations = 1500;
      alpha = 0.1;
```

```
[30]: theta, cost_history = gradient_descent(X1, Y, theta, alpha, iterations)
      print('Final value of theta =', theta)
      print('cost_history =', cost_history)
```

```
Final value of theta = [ 5.92794892 -2.03833663]
cost_history = [5.16999006 4.96338989 4.7855721  … 0.98499308 0.98499308
0.98499308]
```

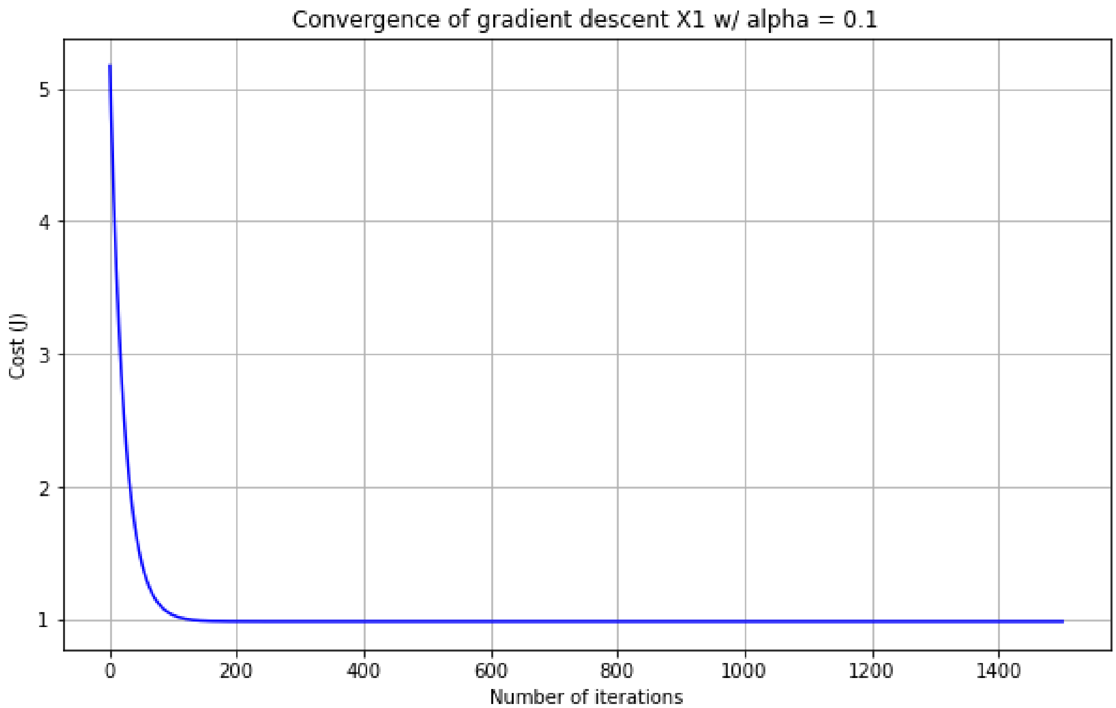
```
[31]: plt.scatter(X1[:,1], Y, color='red', marker= '+', label= 'Training Data')
      plt.plot(X1[:,1],X1.dot(theta), color='green', label='Linear Regression')

      plt.rcParams["figure.figsize"] = (10,6)
      plt.grid()
      plt.xlabel('X1')
      plt.ylabel('Y')
      plt.title('Linear Regression Fit X1 w/ alpha = 0.1')
      plt.legend()
```

```
[31]: <matplotlib.legend.Legend at 0x1c7f16b7d00>
```



[32]:
```
plt.plot(range(1, iterations + 1),cost_history, color='blue')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent X1 w/ alpha = 0.1')
```

[32]:
```
Text(0.5, 1.0, 'Convergence of gradient descent X1 w/ alpha = 0.1')
```



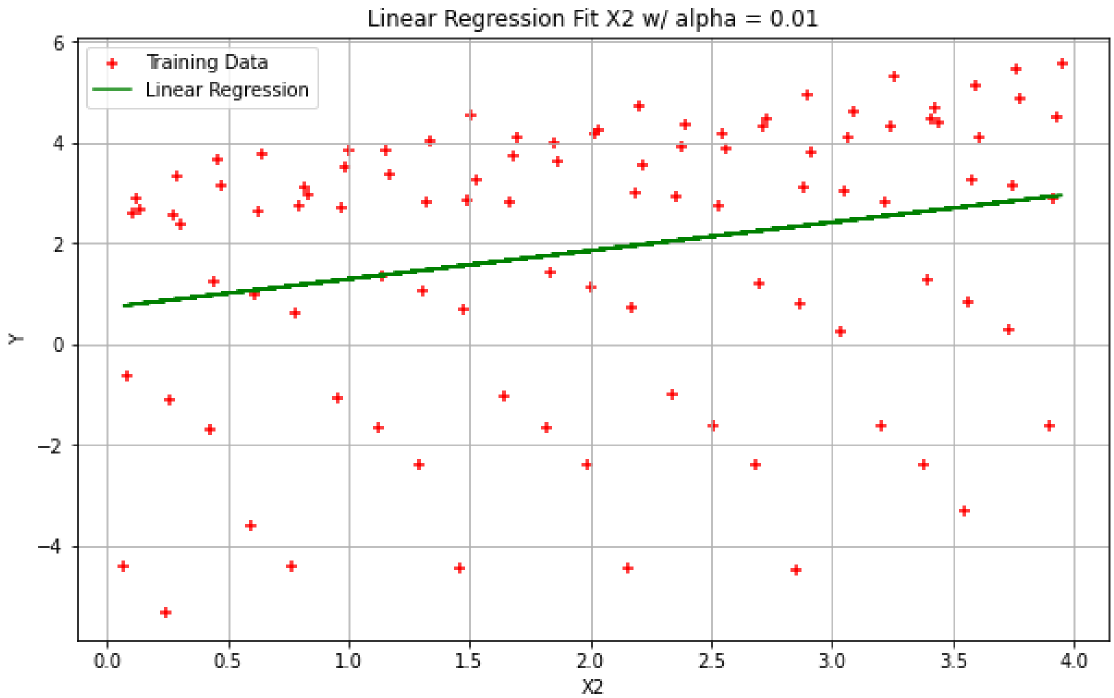
[33]:
```
theta = [0., 0.]
iterations = 1500;
alpha = 0.01;
```

[34]:
```
theta, cost_history = gradient_descent(X2, Y, theta, alpha, iterations)
print('Final value of theta =', theta)
print('cost_history =', cost_history)
```

```
Final value of theta = [0.71988473 0.56390334]
cost_history = [5.29831663 5.09909109 4.92356115 … 3.5993997  3.59939955
3.5993994 ]
```

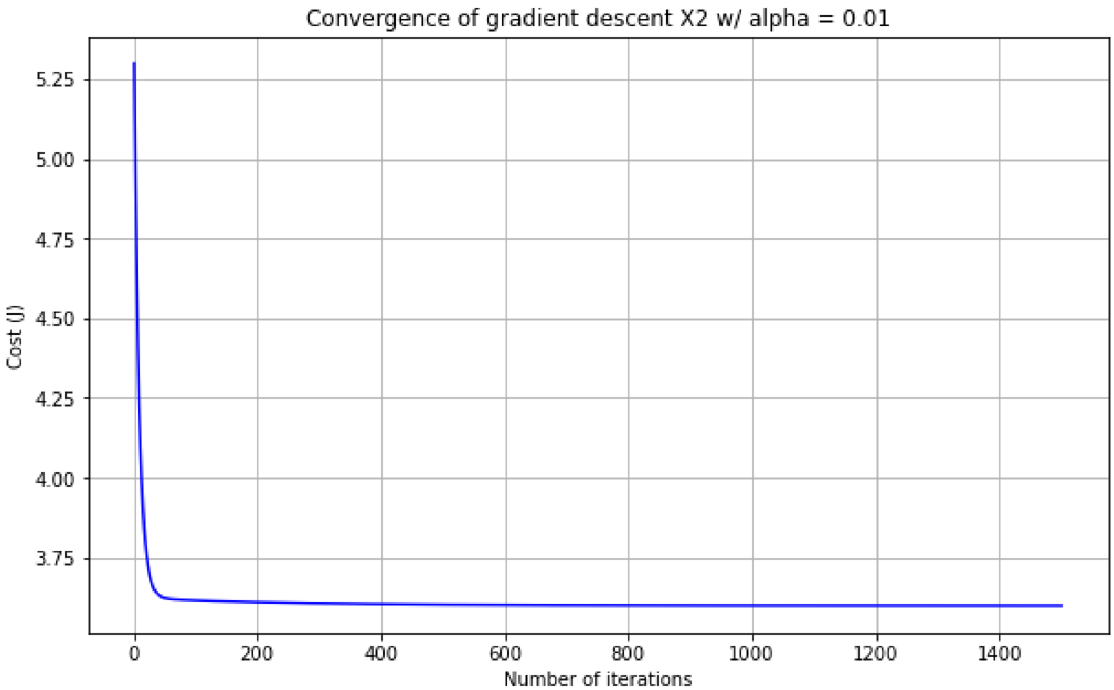
[35]:
```
plt.scatter(X2[:,1], Y, color='red', marker= '+', label= 'Training Data')
plt.plot(X2[:,1],X2.dot(theta), color='green', label='Linear Regression')

plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('X2')
plt.ylabel('Y')
plt.title('Linear Regression Fit X2 w/ alpha = 0.01')
plt.legend()
```

[35]: <matplotlib.legend.Legend at 0x1c7f16d9c10>



[36]:
```python
plt.plot(range(1, iterations + 1),cost_history, color='blue')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent X2 w/ alpha = 0.01')
```

[36]: Text(0.5, 1.0, 'Convergence of gradient descent X2 w/ alpha = 0.01')



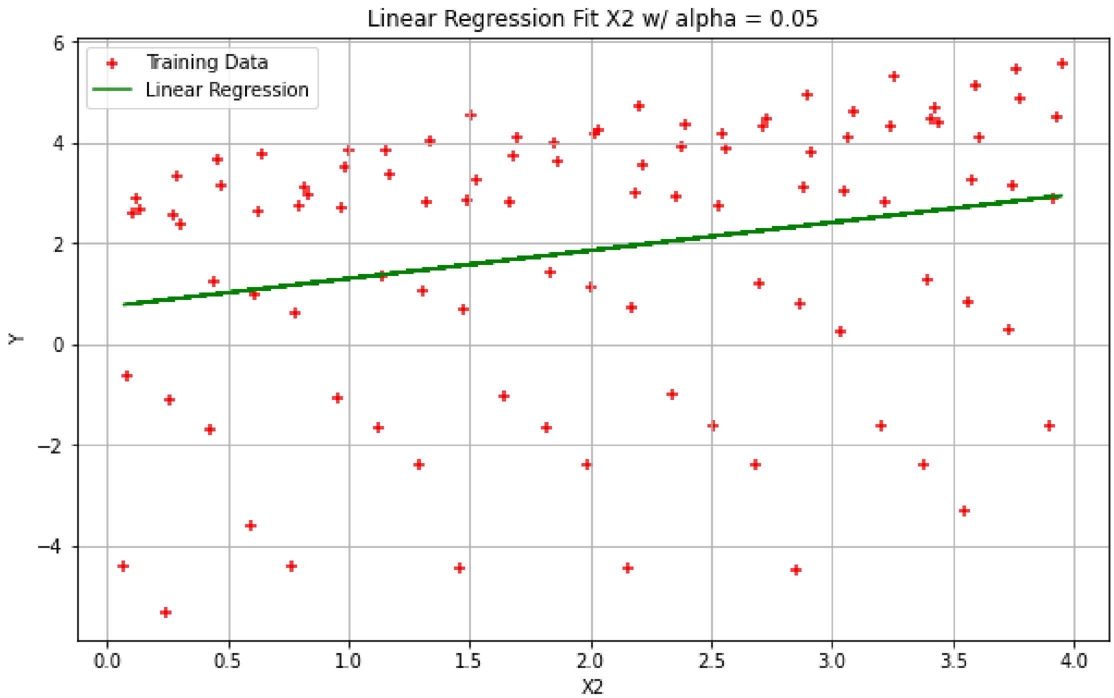
[37]:
```
theta = [0., 0.]
iterations = 1500;
alpha = 0.05;
```

[38]:
```
theta, cost_history = gradient_descent(X2, Y, theta, alpha, iterations)
print('Final value of theta =', theta)
print('cost_history =', cost_history)
```

```
Final value of theta = [0.73606041 0.55760762]
cost_history = [4.5369622  4.06234927 3.83409365 … 3.59936602 3.59936602
3.59936602]
```

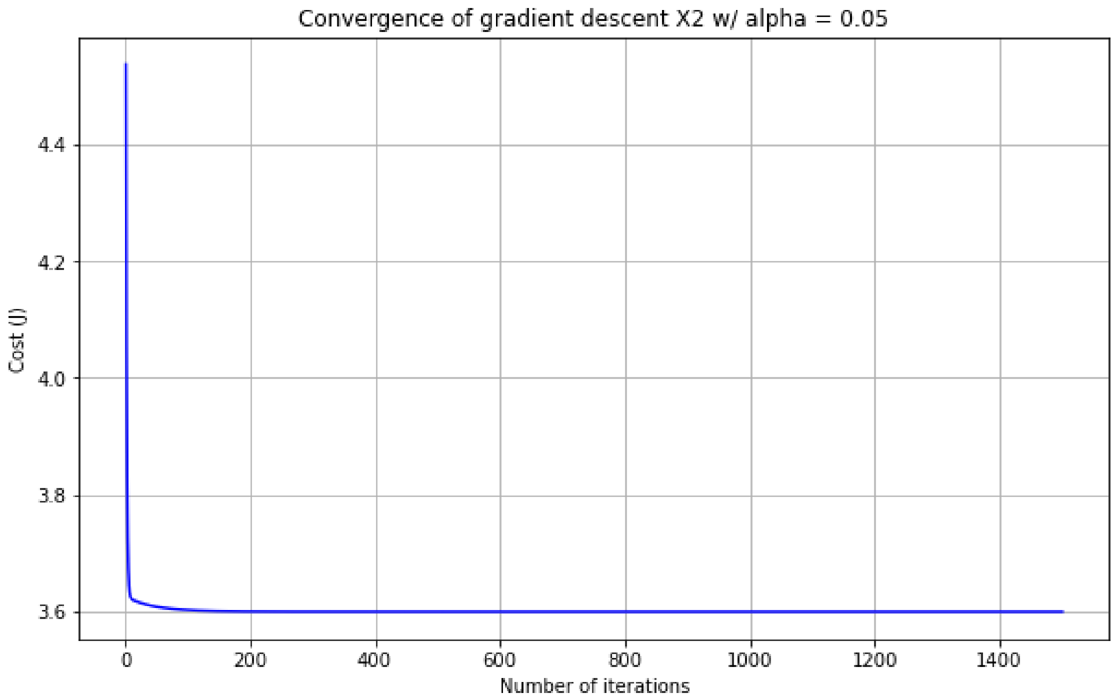
[39]:
```
plt.scatter(X2[:,1], Y, color='red', marker= '+', label= 'Training Data')
plt.plot(X2[:,1],X2.dot(theta), color='green', label='Linear Regression')

plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('X2')
plt.ylabel('Y')
plt.title('Linear Regression Fit X2 w/ alpha = 0.05')
plt.legend()
```

[39]: <matplotlib.legend.Legend at 0x1c7f17d6730>



[40]:
```
plt.plot(range(1, iterations + 1),cost_history, color='blue')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent X2 w/ alpha = 0.05')
```

[40]: Text(0.5, 1.0, 'Convergence of gradient descent X2 w/ alpha = 0.05')



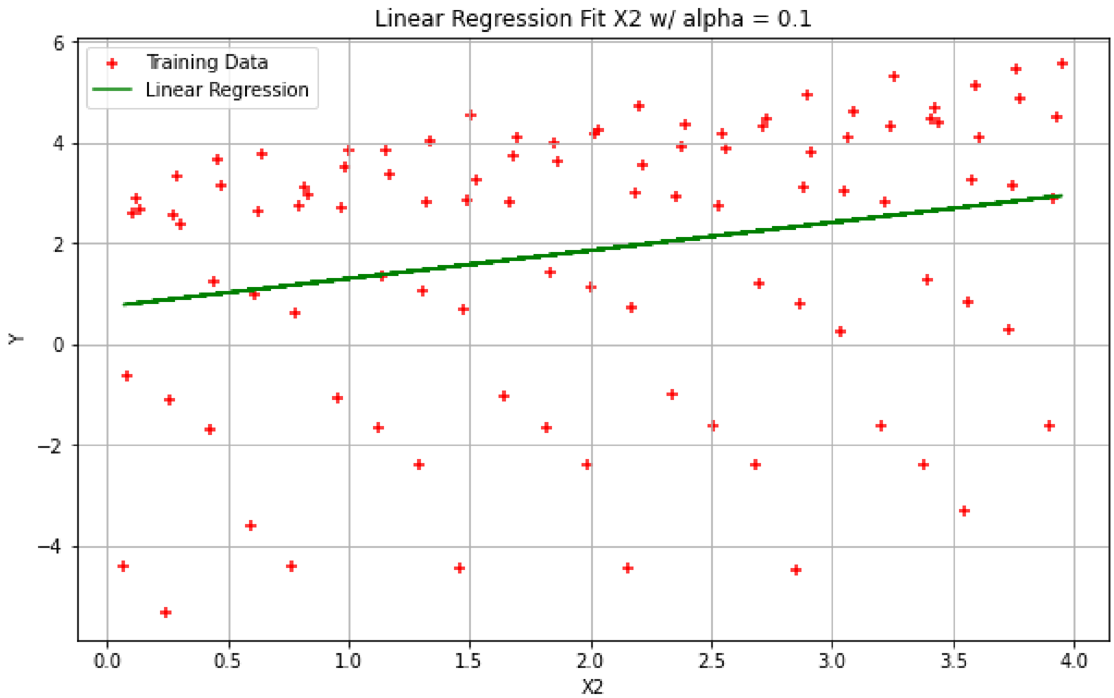
[41]:
```
theta = [0., 0.]
iterations = 1500;
alpha = 0.1;
```

[42]:
```
theta, cost_history = gradient_descent(X2, Y, theta, alpha, iterations)
print('Final value of theta =', theta)
print('cost_history =', cost_history)
```

```
Final value of theta = [0.73606043 0.55760761]
cost_history = [3.90731819 3.66528504 3.62832072 … 3.59936602 3.59936602
3.59936602]
```

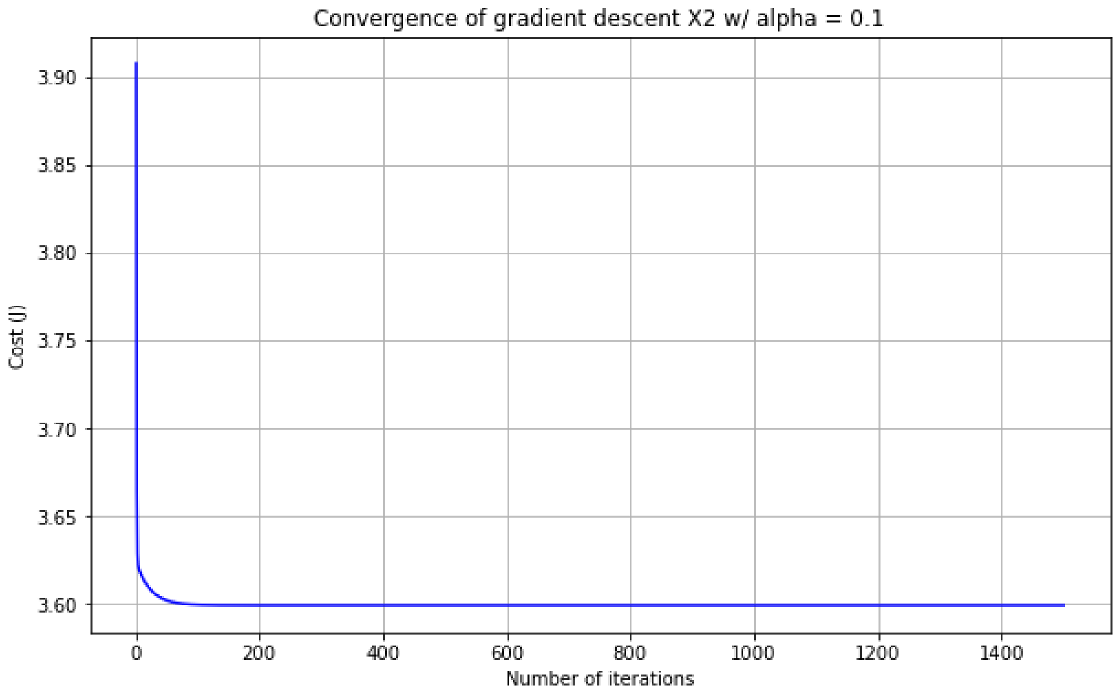
[43]:
```
plt.scatter(X2[:,1], Y, color='red', marker= '+', label= 'Training Data')
plt.plot(X2[:,1],X2.dot(theta), color='green', label='Linear Regression')

plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('X2')
plt.ylabel('Y')
plt.title('Linear Regression Fit X2 w/ alpha = 0.1')
plt.legend()
```

[43]: <matplotlib.legend.Legend at 0x1c7f19be6d0>



[44]:
```
plt.plot(range(1, iterations + 1),cost_history, color='blue')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent X2 w/ alpha = 0.1')
```

[44]: Text(0.5, 1.0, 'Convergence of gradient descent X2 w/ alpha = 0.1')



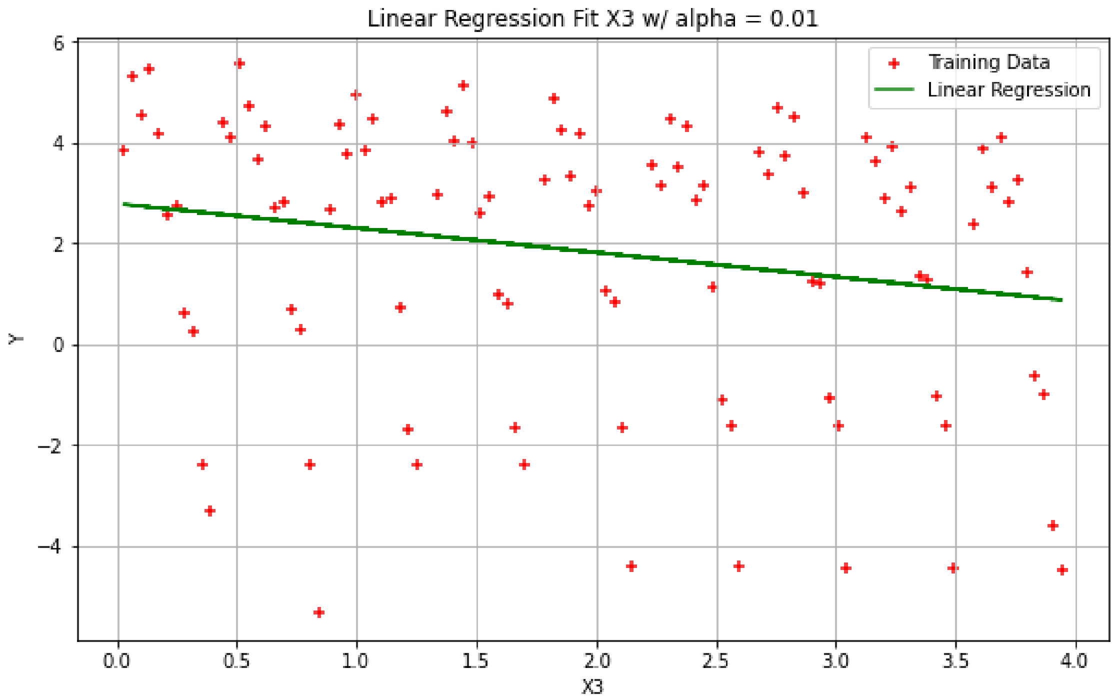
[45]:
```
theta = [0., 0.]
iterations = 1500;
alpha = 0.01;
```

[46]:
```
theta, cost_history = gradient_descent(X3, Y, theta, alpha, iterations)
print('Final value of theta =', theta)
print('cost_history =', cost_history)
```

```
Final value of theta = [ 2.78048129 -0.48451631]
cost_history = [5.40768785 5.30397076 5.21178297 … 3.63053597 3.6305311
3.63052625]
```

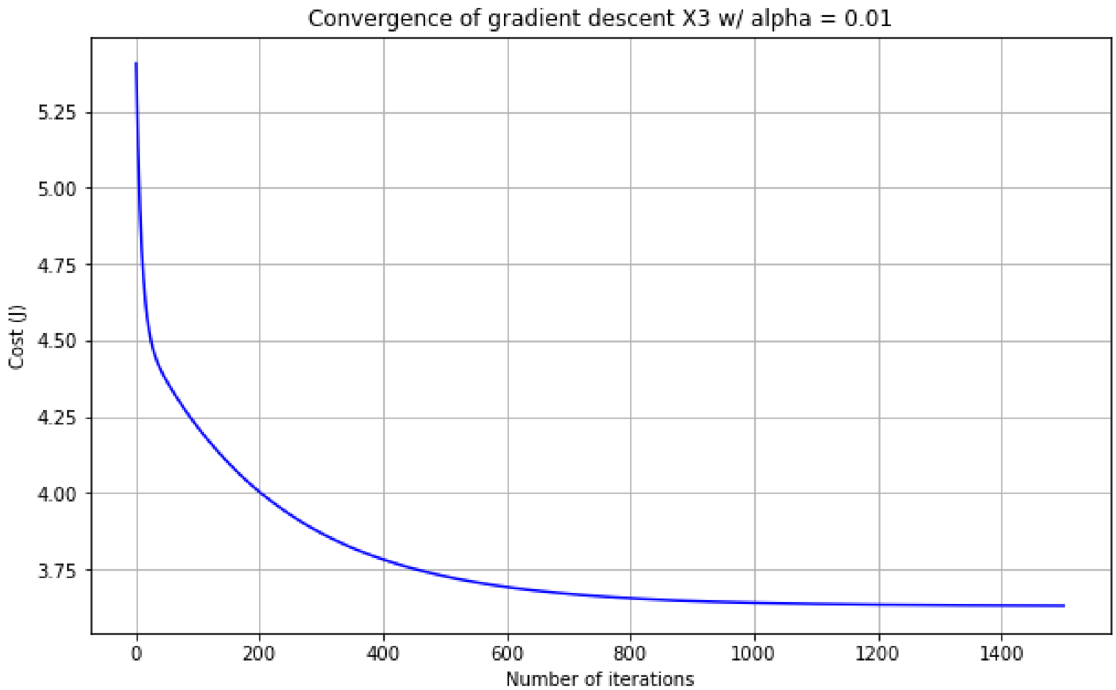
[47]:
```
plt.scatter(X3[:,1], Y, color='red', marker= '+', label= 'Training Data')
plt.plot(X3[:,1],X3.dot(theta), color='green', label='Linear Regression')

plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('X3')
plt.ylabel('Y')
plt.title('Linear Regression Fit X3 w/ alpha = 0.01')
plt.legend()
```

[47]:
```
<matplotlib.legend.Legend at 0x1c7f1a74670>
```



[48]:
```
plt.plot(range(1, iterations + 1),cost_history, color='blue')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent X3 w/ alpha = 0.01')
```

[48]: Text(0.5, 1.0, 'Convergence of gradient descent X3 w/ alpha = 0.01')



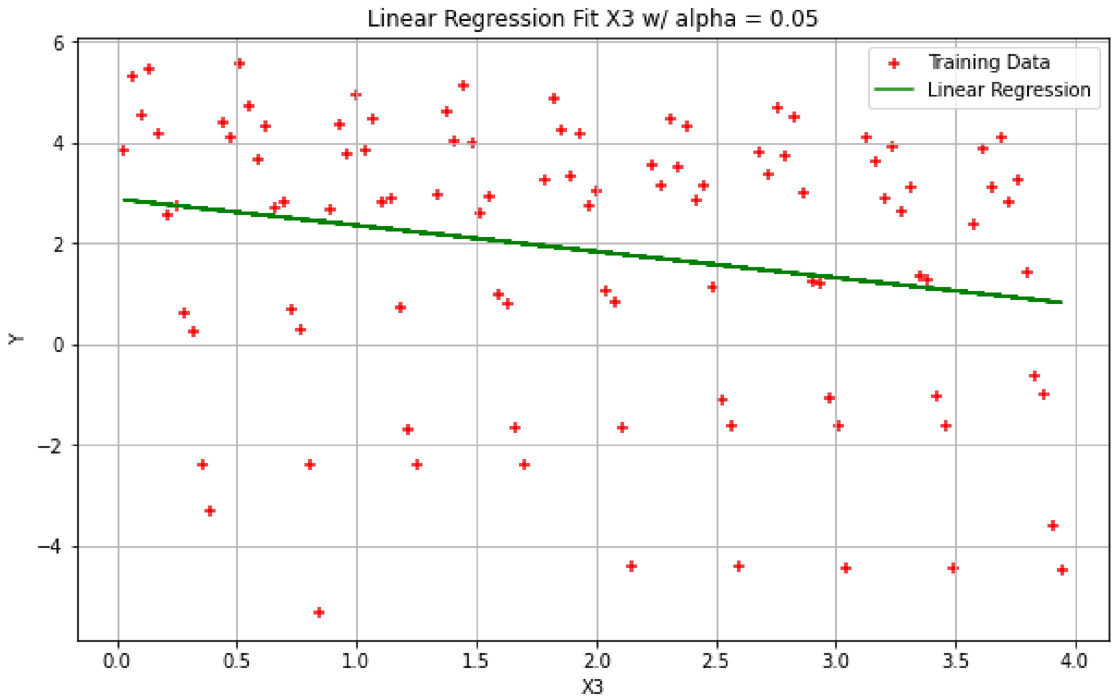
[49]:
```
theta = [0., 0.]
iterations = 1500;
alpha = 0.05;
```

[50]:
```
theta, cost_history = gradient_descent(X3, Y, theta, alpha, iterations)
print('Final value of theta =', theta)
print('cost_history =', cost_history)
```

```
Final value of theta = [ 2.87142199 -0.52048284]
cost_history = [5.00990921 4.74622414 4.60645259 … 3.62945112 3.62945112
3.62945112]
```

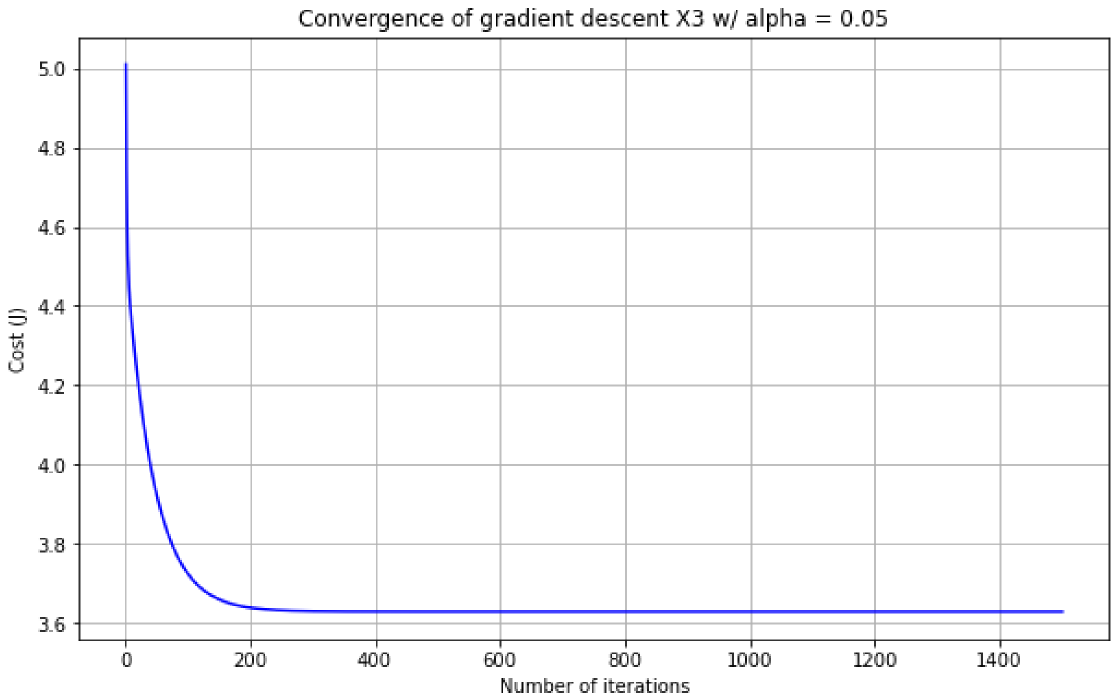
[51]:
```
plt.scatter(X3[:,1], Y, color='red', marker= '+', label= 'Training Data')
plt.plot(X3[:,1],X3.dot(theta), color='green', label='Linear Regression')

plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('X3')
plt.ylabel('Y')
plt.title('Linear Regression Fit X3 w/ alpha = 0.05')
plt.legend()
```

[51]: <matplotlib.legend.Legend at 0x1c7f2b5aa90>



[52]:
```
plt.plot(range(1, iterations + 1),cost_history, color='blue')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent X3 w/ alpha = 0.05')
```

[52]: Text(0.5, 1.0, 'Convergence of gradient descent X3 w/ alpha = 0.05')



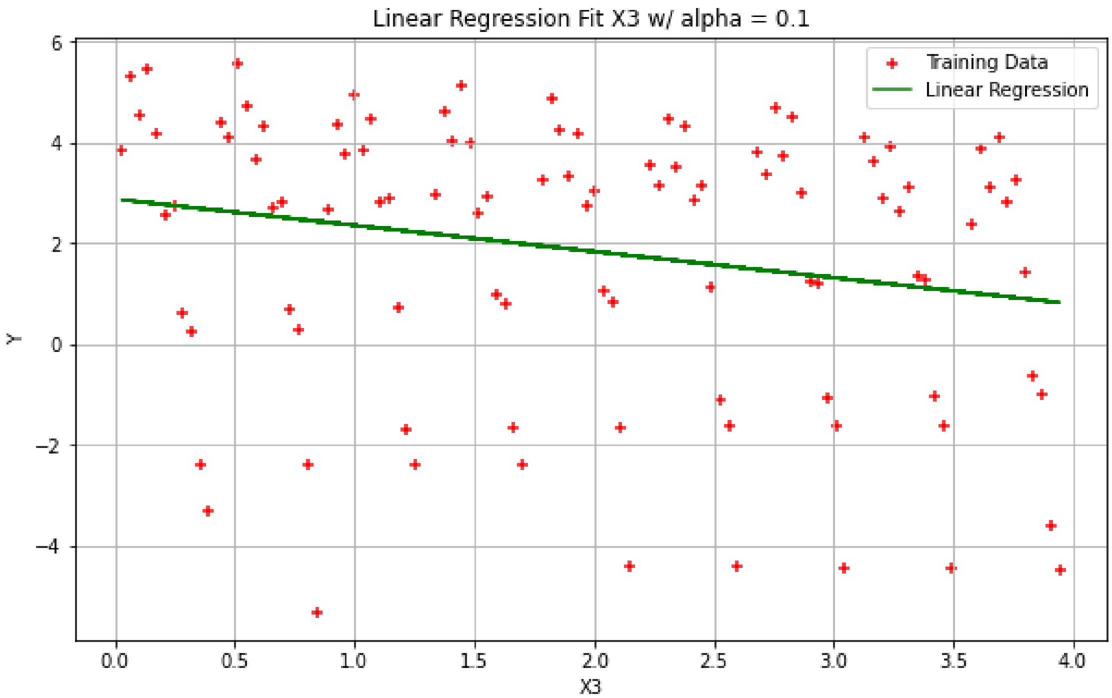
[53]:
```
theta = [0., 0.]
iterations = 1500;
alpha = 0.1;
```

[54]:
```
theta, cost_history = gradient_descent(X3, Y, theta, alpha, iterations)
print('Final value of theta =', theta)
print('cost_history =', cost_history)
```

```
Final value of theta = [ 2.8714221  -0.52048288]
cost_history = [4.66843939 4.49602325 4.43685075 … 3.62945112 3.62945112
3.62945112]
```

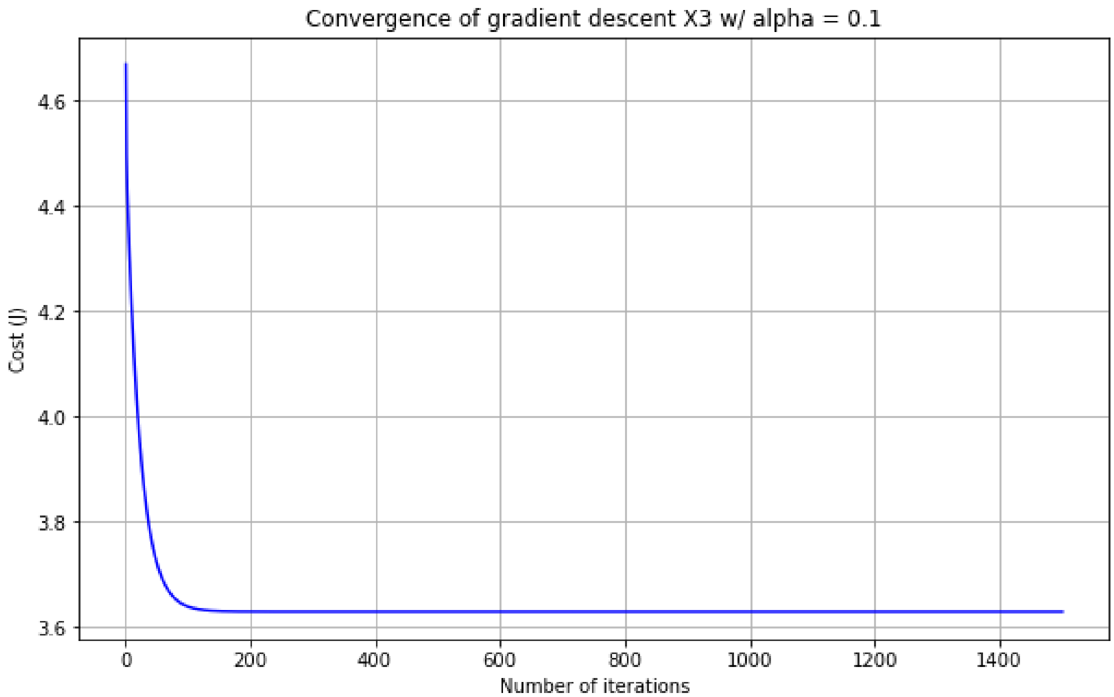
[55]:
```
plt.scatter(X3[:,1], Y, color='red', marker= '+', label= 'Training Data')
plt.plot(X3[:,1],X3.dot(theta), color='green', label='Linear Regression')

plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('X3')
plt.ylabel('Y')
plt.title('Linear Regression Fit X3 w/ alpha = 0.1')
plt.legend()
```

[55]:
```
<matplotlib.legend.Legend at 0x1c7f2bfcac0>
```



[56]:
```
plt.plot(range(1, iterations + 1),cost_history, color='blue')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent X3 w/ alpha = 0.1')
```

[56]:
```
Text(0.5, 1.0, 'Convergence of gradient descent X3 w/ alpha = 0.1')
```


Convergence of gradient descent X3 w/ alpha = 0.1
Cost (J)
4.6
4.4
4.2
4.0
3.8
3.6
0
200
400
600
800
1000
1200
1400
Number of iterations