# gradient descent

September 20, 2022

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
df = pd.read_csv('D3.csv')
df.head()

m = len(df)
num_vars = np.shape(df)[1] - 1

print(np.shape(df))
print(df)
```

```
(100, 4)
          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]
```

[3]:
```
X = df.values[:, 0:3]  # get x values
Y = df.values[:, 3]  # get y values

# stack 1s on left side of X
X = np.hstack( (np.ones((m, 1)) , X ) )

print(f"X=\n{str(X)}\n")
print(f"Y=\n{str(Y)}\n")
```

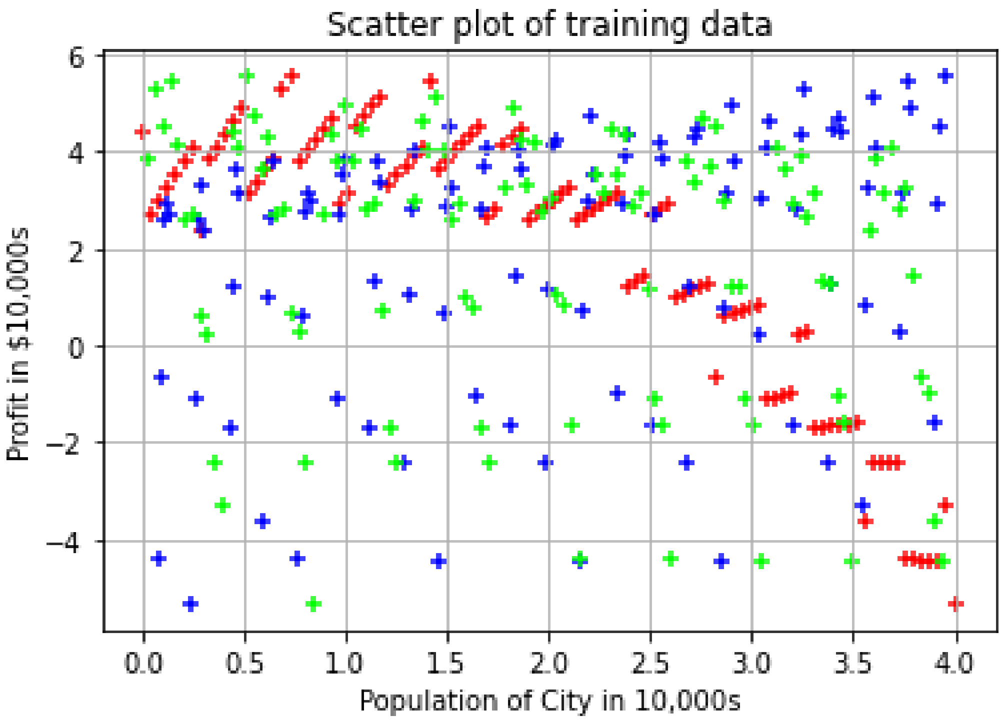
```
X=
[[1.         0.         3.44       0.44      ]
 [1.         0.04040404 0.1349495  0.88848485]
 [1.         0.08080808 0.82989899 1.3369697 ]
 [1.         0.12121212 1.52484848 1.78545454]
 [1.         0.16161616 2.21979798 2.23393939]
 [1.         0.2020202  2.91474747 2.68242424]
 [1.         0.24242424 3.60969697 3.13090909]
 [1.         0.28282828 0.30464646 3.57939394]
 [1.         0.32323232 0.99959596 0.02787879]
 [1.         0.36363636 1.69454546 0.47636364]
 [1.         0.4040404  2.38949495 0.92484849]
 [1.         0.44444444 3.08444444 1.37333333]
 [1.         0.48484848 3.77939394 1.82181818]
 [1.         0.52525252 0.47434343 2.27030303]
 [1.         0.56565657 1.16929293 2.71878788]
 [1.         0.60606061 1.86424242 3.16727273]
 [1.         0.64646465 2.55919192 3.61575758]
 [1.         0.68686869 3.25414141 0.06424242]
 [1.         0.72727273 3.94909091 0.51272727]
 [1.         0.76767677 0.6440404  0.96121212]
 [1.         0.80808081 1.3389899  1.40969697]
 [1.         0.84848485 2.03393939 1.85818182]
 [1.         0.88888889 2.72888889 2.30666667]
 [1.         0.92929293 3.42383838 2.75515152]
 [1.         0.96969697 0.11878788 3.20363636]
 [1.         1.01010101 0.81373737 3.65212121]
 [1.         1.05050505 1.50868687 0.10060606]
 [1.         1.09090909 2.20363636 0.54909091]
 [1.         1.13131313 2.89858586 0.99757576]
 [1.         1.17171717 3.59353535 1.44606061]
 [1.         1.21212121 0.28848485 1.89454546]
 [1.         1.25252525 0.98343434 2.3430303 ]
 [1.         1.29292929 1.67838384 2.79151515]
 [1.         1.33333333 2.37333333 3.24      ]
 [1.         1.37373737 3.06828283 3.68848485]
 [1.         1.41414141 3.76323232 0.1369697 ]
 [1.         1.45454546 0.45818182 0.58545455]
 [1.         1.49494949 1.15313131 1.03393939]
 [1.         1.53535354 1.84808081 1.48242424]
 [1.         1.57575758 2.5430303  1.93090909]
 [1.         1.61616162 3.2379798  2.37939394]
 [1.         1.65656566 3.93292929 2.82787879]
 [1.         1.6969697  0.62787879 3.27636364]
 [1.         1.73737374 1.32282828 3.72484848]
 [1.         1.77777778 2.01777778 0.17333333]
 [1.         1.81818182 2.71272727 0.62181818]
 [1.         1.85858586 3.40767677 1.07030303]
```

```
[1.         1.8989899  0.10262626 1.51878788]
[1.         1.93939394 0.79757576 1.96727273]
[1.         1.97979798 1.49252525 2.41575758]
[1.         2.02020202 2.18747475 2.86424242]
[1.         2.06060606 2.88242424 3.31272727]
[1.         2.1010101  3.57737374 3.76121212]
[1.         2.14141414 0.27232323 0.20969697]
[1.         2.18181818 0.96727273 0.65818182]
[1.         2.22222222 1.66222222 1.10666667]
[1.         2.26262626 2.35717172 1.55515151]
[1.         2.3030303  3.05212121 2.00363636]
[1.         2.34343434 3.74707071 2.45212121]
[1.         2.38383838 0.4420202  2.90060606]
[1.         2.42424242 1.1369697  3.34909091]
[1.         2.46464646 1.83191919 3.79757576]
[1.         2.5050505  2.52686869 0.24606061]
[1.         2.54545455 3.22181818 0.69454545]
[1.         2.58585859 3.91676768 1.1430303 ]
[1.         2.62626263 0.61171717 1.59151515]
[1.         2.66666667 1.30666667 2.04      ]
[1.         2.70707071 2.00161616 2.48848485]
[1.         2.74747475 2.69656566 2.9369697 ]
[1.         2.78787879 3.39151515 3.38545454]
[1.         2.82828283 0.08646465 3.83393939]
[1.         2.86868687 0.78141414 0.28242424]
[1.         2.90909091 1.47636364 0.73090909]
[1.         2.94949495 2.17131313 1.17939394]
[1.         2.98989899 2.86626263 1.62787879]
[1.         3.03030303 3.56121212 2.07636364]
[1.         3.07070707 0.25616162 2.52484849]
[1.         3.11111111 0.95111111 2.97333333]
[1.         3.15151515 1.64606061 3.42181818]
[1.         3.19191919 2.3410101  3.87030303]
[1.         3.23232323 3.0359596  0.31878788]
[1.         3.27272727 3.73090909 0.76727273]
[1.         3.31313131 0.42585859 1.21575758]
[1.         3.35353535 1.12080808 1.66424242]
[1.         3.39393939 1.81575758 2.11272727]
[1.         3.43434343 2.51070707 2.56121212]
[1.         3.47474748 3.20565657 3.00969697]
[1.         3.51515151 3.90060606 3.45818182]
[1.         3.55555556 0.59555556 3.90666667]
[1.         3.5959596  1.29050505 0.35515151]
[1.         3.63636364 1.98545455 0.80363636]
[1.         3.67676768 2.68040404 1.25212121]
[1.         3.71717172 3.37535353 1.70060606]
[1.         3.75757576 0.07030303 2.14909091]
[1.         3.7979798  0.76525252 2.59757576]
```

```
 [1.         3.83838384 1.46020202 3.04606061]
 [1.         3.87878788 2.15515152 3.49454545]
 [1.         3.91919192 2.85010101 3.9430303 ]
 [1.         3.95959596 3.5450505  0.39151515]
 [1.         4.         0.24       0.84      ]]

Y=
[ 4.38754501  2.6796499   2.96848981  3.25406475  3.53637472  3.81541972
  4.09119974  2.36371479  3.83296487  4.09894997  4.3616701   4.62112526
  4.87731544  3.13024065  3.37990089  3.62629616  3.86942645  5.30929177
  5.54589212  3.77922749  4.00929789  4.23610332  4.45964378  4.67991926
  2.89692977  3.1106753   4.52115587  4.72837146  4.93232208  5.13300772
  3.33042839  3.52458409  3.71547481  3.90310057  4.08746135  5.46855715
  3.64638799  3.82095385  3.99225473  4.16029065  4.32506159  4.48656756
  2.64480856  2.79978458  4.15149563  4.29994171  4.44512281  2.58703894
  2.7256901   2.86107628  2.99319749  3.12205374  3.247645    2.56997129
  2.68903261  2.80482896  2.91736034  3.02662674  3.13262817  1.23536462
  1.3348361   1.43104261  2.72398415  2.81366071  2.9000723   0.98321892
  1.06310057  1.13971724  1.21306894  1.28315566 -0.65002258  0.6135342
  0.673826    0.73085284  0.7846147   0.83511159 -1.1176565  -1.07368956
 -1.03298759 -0.99555059  0.23862143  0.26952848 -1.70282944 -1.67845234
 -1.6573402  -1.63949305 -1.62491086 -1.61359365 -3.60554141 -2.40075414
 -2.39923185 -2.40097453 -2.40598218 -4.4142548  -4.4257924  -4.44059497
 -4.45866252 -4.47999504 -3.30459253 -5.33245499]
```

[4]:
```python
plt.scatter(X[:,1], Y, color='red'  , marker= '+')
plt.scatter(X[:,2], Y, color='blue' , marker= '+')
plt.scatter(X[:,3], Y, color='lime' , marker= '+')
plt.grid()
plt.rcParams["figure.figsize"] = (10,6)
plt.xlabel('Population of City in 10,000s')
plt.ylabel('Profit in $10,000s')
plt.title('Scatter plot of training data')

# Was for fun with the data despite how stupid it looks
```

[4]:
```
Text(0.5, 1.0, 'Scatter plot of training data')
```




[5]:
```python
theta = np.zeros(num_vars + 1)
print(theta)
```

```
[0. 0. 0. 0.]
```

[6]:
```python
def compute_cost(x, y, theta, debug:bool=False):
    """
    Compute cost for linear regression.

    Input Parameters
    ----------------
    X : 2D array where each row represent the training example and each column␣
↪represent
      m= number of training examples
      n= number of features (including X_0 column of ones)
    y : 1D array of labels/target value for each traing example. dimension(1 x␣
↪m)

    theta : 1D array of fitting parameters or weights. Dimension (1 x n)

    Output Parameters
    -----------------
    J : Scalar value.
```

```
    """
    if(debug):
        print("x shape = ", np.shape(x))
        print("y shape = ", np.shape(y))
        print("theta shape = ", np.shape(theta))

    predictions = x.dot(theta)
    errors = np.subtract(predictions, y)
    sqrErrors = np.square(errors)
    J = 1 / (2 * m) * np.sum( sqrErrors )

    if(debug):
        print(sqrErrors)
        print(J)

    return J
```

[7]:
```
# Lets compute the cost for theta values
cost = compute_cost(X, Y, theta, debug=True)
print('The cost for given values of theta =', cost)
```

```
x shape =  (100, 4)
y shape =  (100,)
theta shape =  (4,)
[19.25055122  7.18052358  8.81193177 10.58893742 12.50594617 14.55742762
 16.73791531  5.58714761 14.69161966 16.80139086 19.02416606 21.35479865
 23.78820593  9.79840655 11.42373005 13.15002383 14.97246107 28.18857912
 30.75691938 14.28256044 16.07446959 17.94457135 19.88842261 21.90164427
  8.39220208  9.67630085 20.44085037 22.35749664 24.32780105 26.34776824
 11.09175326 12.422693   13.80475309 15.23419404 16.70734026 29.90511734
 13.29614534 14.5996883  15.93809787 17.30801828 18.70615777 20.12928847
  6.9950123   7.83879368 17.23491595 18.48949867 19.75911679  6.69277048
  7.42938651  8.1857575   8.95923124  9.74721952 10.54719805  6.60475245
  7.2308964   7.86706551  8.51099133  9.16046941  9.81335923  1.52612575
  1.78178742  2.04788296  7.42008965  7.91668661  8.41041937  0.96671945
  1.13018282  1.29895538  1.47153625  1.64648846  0.42252936  0.37642421
  0.45404148  0.53414587  0.61562022  0.69741136  1.24915605  1.15280926
  1.06706336  0.99112098  0.05694019  0.0726456   2.89962811  2.81720224
  2.74677656  2.68793745  2.64033531  2.60368446 12.99992885  5.76362045
  5.75631346  5.76467868  5.78875025 19.48564548 19.5876384  19.71888373
 19.87967146 20.07035552 10.92033176 28.43507621]
5.524438459196242
The cost for given values of theta = 5.524438459196242
```

[8]:
```
# Test compute with subsection of values
cost = compute_cost(X[:,1:2], Y, theta[1:2], debug=True)
print('The cost for given values of theta =', cost)
```

```
x shape =  (100, 1)
y shape =  (100,)
theta shape =  (1,)
[19.25055122  7.18052358  8.81193177 10.58893742 12.50594617 14.55742762
 16.73791531  5.58714761 14.69161966 16.80139086 19.02416606 21.35479865
 23.78820593  9.79840655 11.42373005 13.15002383 14.97246107 28.18857912
 30.75691938 14.28256044 16.07446959 17.94457135 19.88842261 21.90164427
  8.39220208  9.67630085 20.44085037 22.35749664 24.32780105 26.34776824
 11.09175326 12.422693   13.80475309 15.23419404 16.70734026 29.90511734
 13.29614534 14.5996883  15.93809787 17.30801828 18.70615777 20.12928847
  6.9950123   7.83879368 17.23491595 18.48949867 19.75911679  6.69277048
  7.42938651  8.1857575   8.95923124  9.74721952 10.54719805  6.60475245
  7.2308964   7.86706551  8.51099133  9.16046941  9.81335923  1.52612575
  1.78178742  2.04788296  7.42008965  7.91668661  8.41041937  0.96671945
  1.13018282  1.29895538  1.47153625  1.64648846  0.42252936  0.37642421
  0.45404148  0.53414587  0.61562022  0.69741136  1.24915605  1.15280926
  1.06706336  0.99112098  0.05694019  0.0726456   2.89962811  2.81720224
  2.74677656  2.68793745  2.64033531  2.60368446 12.99992885  5.76362045
  5.75631346  5.76467868  5.78875025 19.48564548 19.5876384  19.71888373
 19.87967146 20.07035552 10.92033176 28.43507621]
5.524438459196242
The cost for given values of theta = 5.524438459196242
```

[9]:
```python
# Test computing with a different theta
cost = compute_cost(X, Y, np.array([1, 4, 6, 8]), debug=True)
print('The cost for given values of theta =', cost)
```

```
x shape =  (100, 4)
y shape =  (100,)
theta shape =  (4,)
[ 431.49488627   40.95413801  196.83792641  469.30770885  858.51302749
 1364.60348633 1987.72875596  913.89090821   21.90774011  152.12206258
  399.61005868  764.52171728 1247.00709232  440.12911056  820.75293796
 1319.25736438 1935.7926999   341.39845187  684.32270642  140.31395062
  381.59827227  741.30555272 1219.5864856  1816.59182589  802.23063903
 1298.0976808   111.04594623  333.05557829  674.18390807 1134.58207556
  376.56680082  736.05672701 1215.12663727 1813.92792681 2532.61205674
  618.17672603  112.4558678   336.69120161  681.05540224 1145.70025067
 1730.77758923 2436.43932764 1233.50234542 1839.16336112  304.60542237
  636.23477235 1088.84856661  352.49992154  705.19574986 1179.18943121
 1774.63344713 2491.6803477  3330.48274192  106.23838054  327.87543667
  671.36575263 1136.86219541 1724.51769668 2434.4852505  1236.01173906
 1846.99494182 2580.60590955  646.47714545 1105.92363765 1688.24104423
  724.915552   1208.5055282  1815.2832804  2545.40244424 3399.01671751
 1949.52775834  353.7819442   711.18180675 1192.33176019 1797.38582438
 2526.49807996 1305.82801183 1937.00925663 2692.56766735 3572.65757992
 1187.26638729 1793.10046236  797.30370561 1305.47673849 1938.59347302
 2696.80863009 3580.27699796 4589.15342375 2878.79789135  804.79079653
```

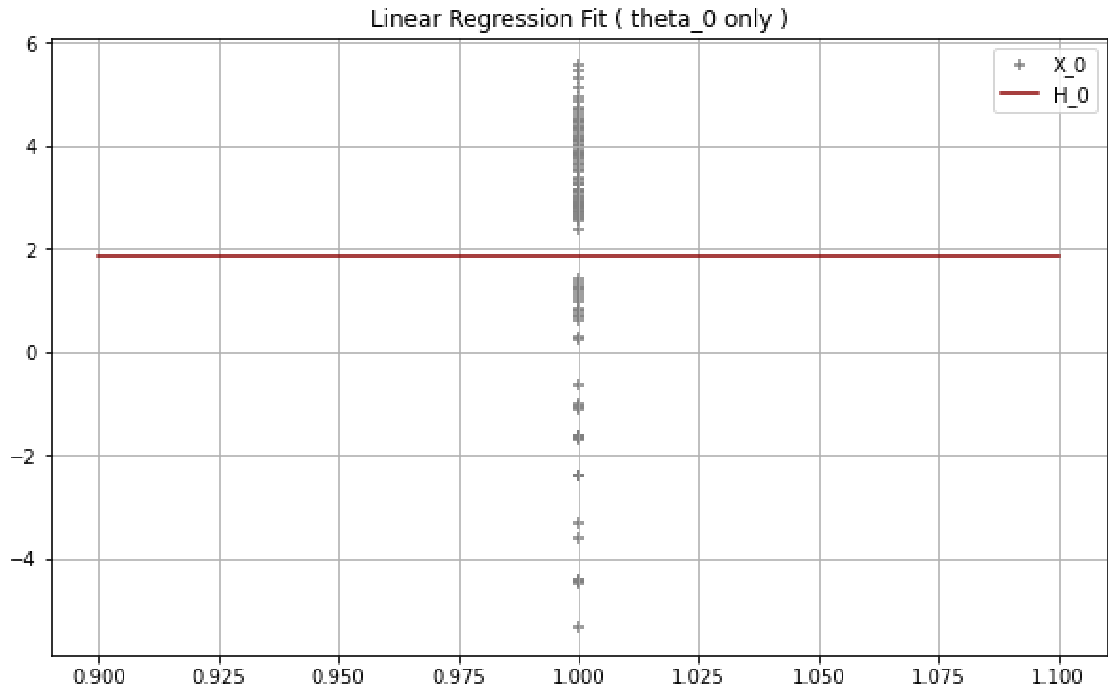
```
 1316.7104142  1954.29767815 2717.70775834 1448.49534316 2115.06472189
 2907.77915014 3826.79405575 4872.26492556 1984.29271568  929.78981126]
702.983123692495
The cost for given values of theta = 702.983123692495
```

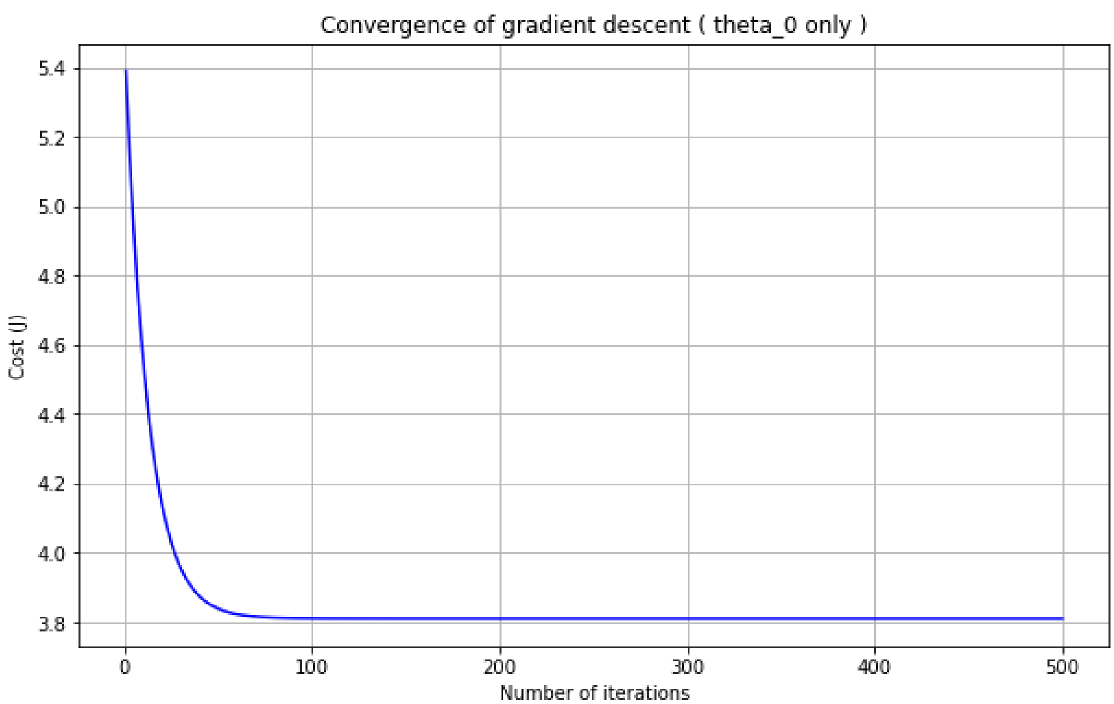
[10]:
```python
def gradient_descent_single(x, y, theta, alpha, iterations):
    cost_history = np.zeros(iterations)

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

    return theta, cost_history
```

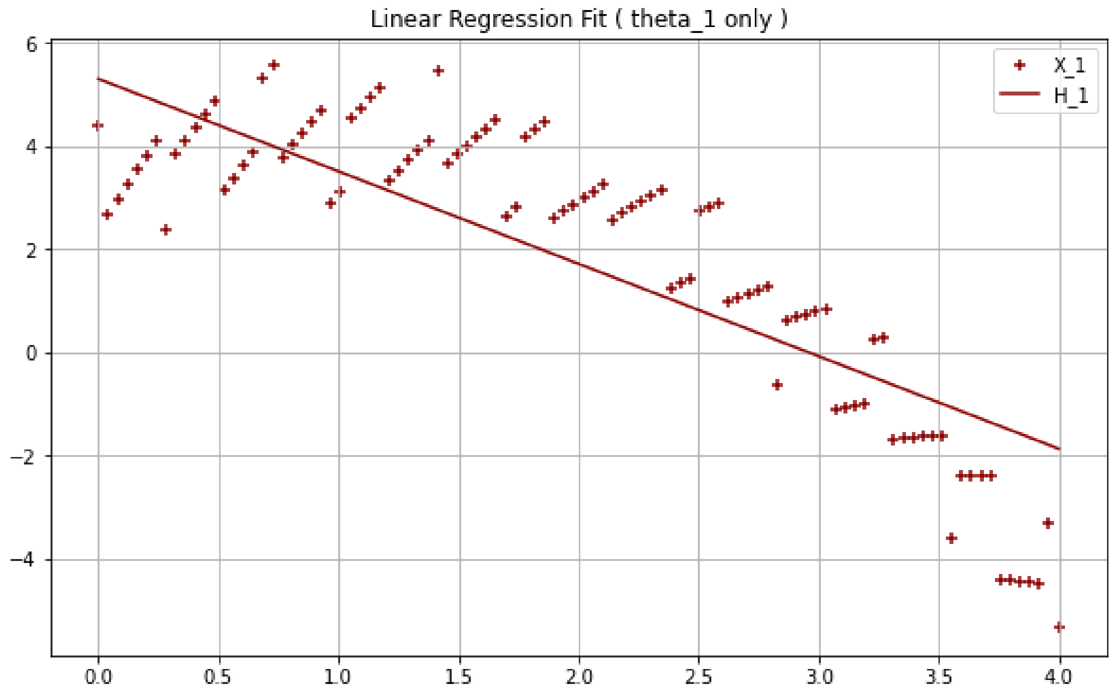
[11]:
```python
iterations = 500;
alpha = 0.02;

color = {
    0: "gray",
    1: "DarkRed",
    2: "DarkBlue",
    3: "Green"
}
tolerance = 0.1

for i in range(0,np.shape(X)[1]):
    theta = np.array([0,0]);

    trimmed_x = X[:,i].reshape(len(X), 1)
    x_0 = np.ones(trimmed_x.shape)
    trimmed_x = np.hstack(( x_0 , trimmed_x ))

    # print(trimmed_x.shape)
    # print(theta.shape)
    # print(trimmed_x)

    print(f'Theta {i}:')
    theta, cost_history = gradient_descent_single(trimmed_x, Y, theta, alpha,␣
 ↪iterations)
    print('    theta = ', theta)
    print('    cost  = ', cost_history[-1])

    x_vals = trimmed_x[:,1]
    if i == 0:
```

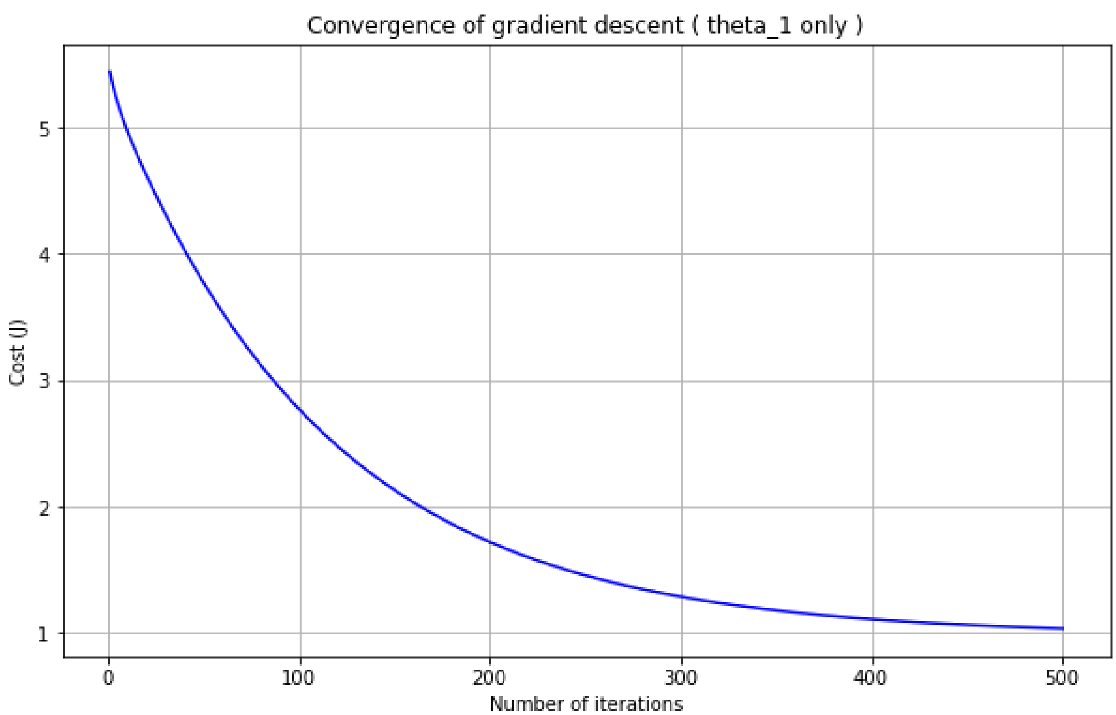
```
        center = trimmed_x[:,1].min()
        x_vals = np.linspace(center - tolerance, center + tolerance,␣
↪len(x_vals))

    plt.figure(30+i)
    # Since X is list of list (feature matrix) lets take values of column of␣
↪index 1 only
    plt.scatter(X[:,i], Y, color=color[i]  , marker= '+', label= f'X_{i}')
    plt.plot(x_vals ,trimmed_x.dot(theta), color='darkRed', label=f'H_{i}')

    plt.rcParams["figure.figsize"] = (10,6)
    plt.grid()
    plt.xlabel('')
    plt.ylabel('')
    plt.title(f'Linear Regression Fit ( theta_{i} only )')
    plt.legend()


    plt.figure(40+i)
    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(f'Convergence of gradient descent ( theta_{i} only )')
```

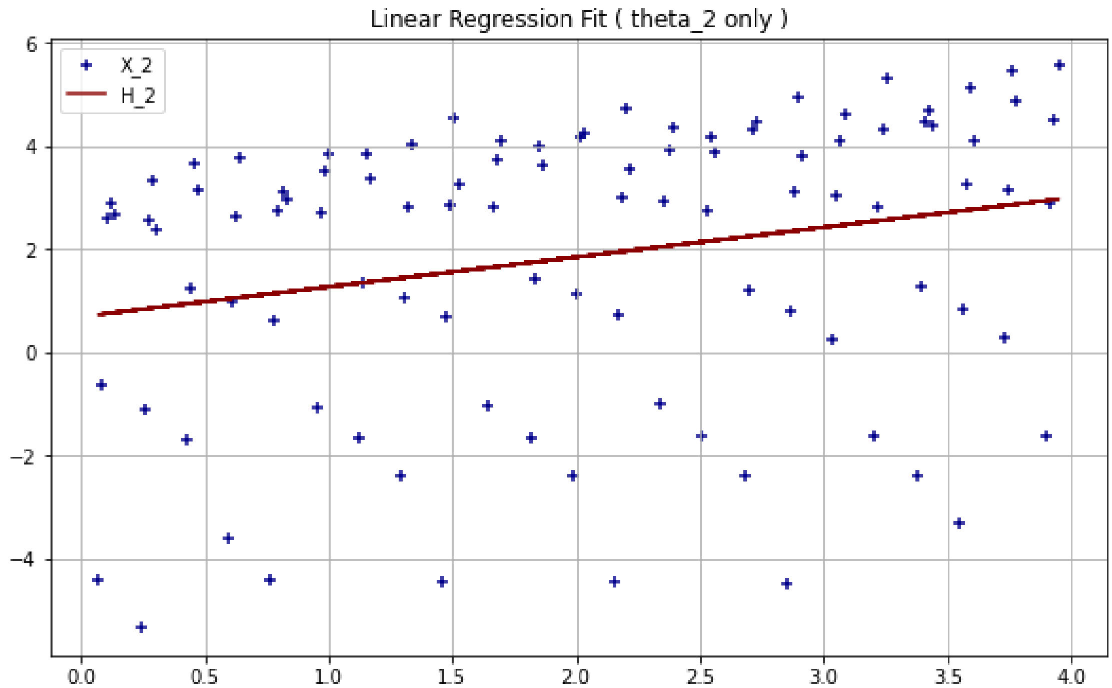
```
(100, 2)
(2,)
Theta 0:
    theta =  [0.92563782 0.92563782]
    cost  =  3.81082769164781
(100, 2)
(2,)
Theta 1:
    theta =  [ 5.29451128 -1.79179951]
    cost  =  1.0361829849497886
(100, 2)
(2,)
Theta 2:
    theta =  [0.68714072 0.57664762]
    cost  =  3.5996713196049464
(100, 2)
(2,)
Theta 3:
    theta =  [ 2.59189722 -0.40993238]
    cost  =  3.6396084981267816
```

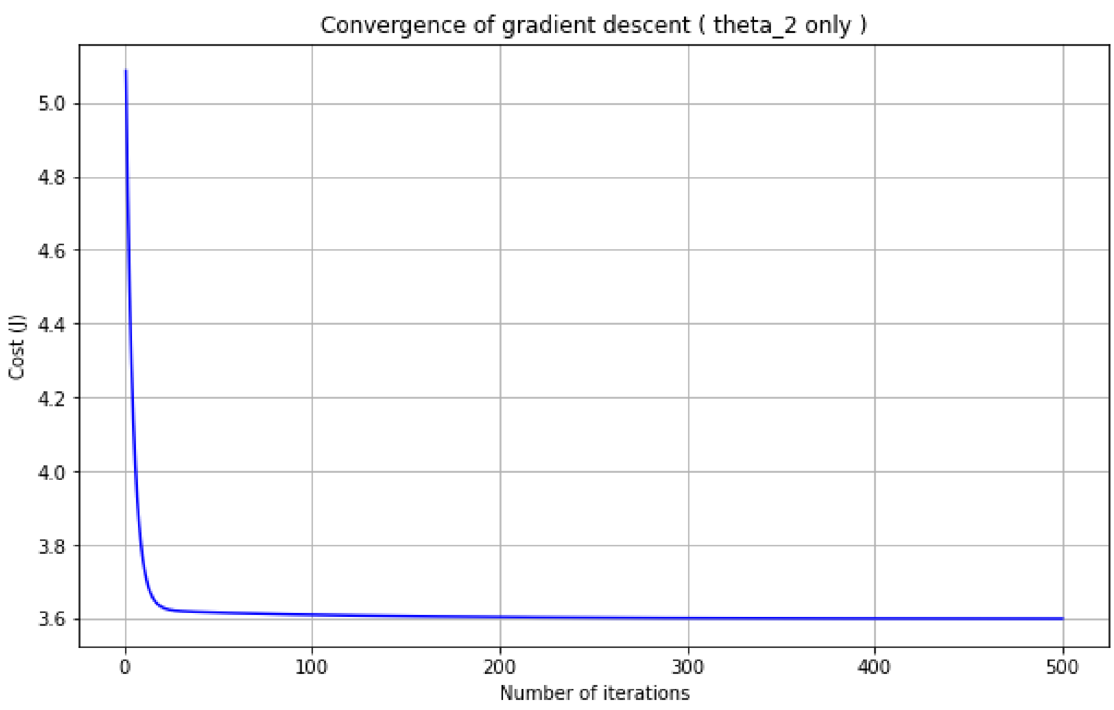

Linear Regression Fit ( theta_0 only )
X_0
H_0

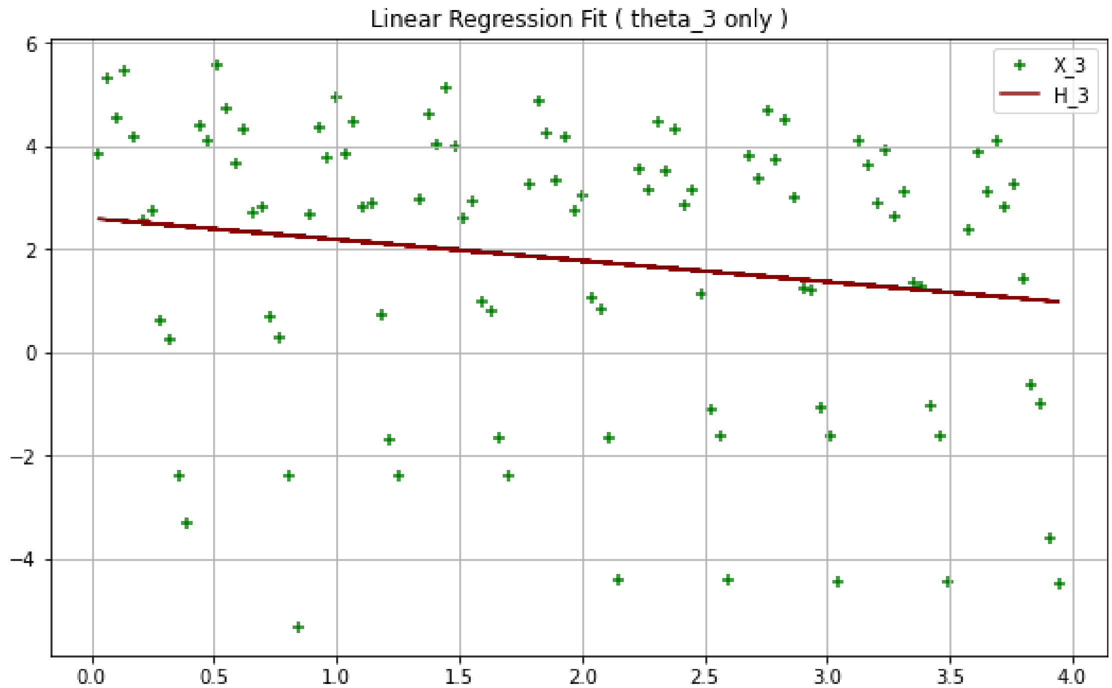

Convergence of gradient descent ( theta_0 only )
Cost (J)
Number of iterations

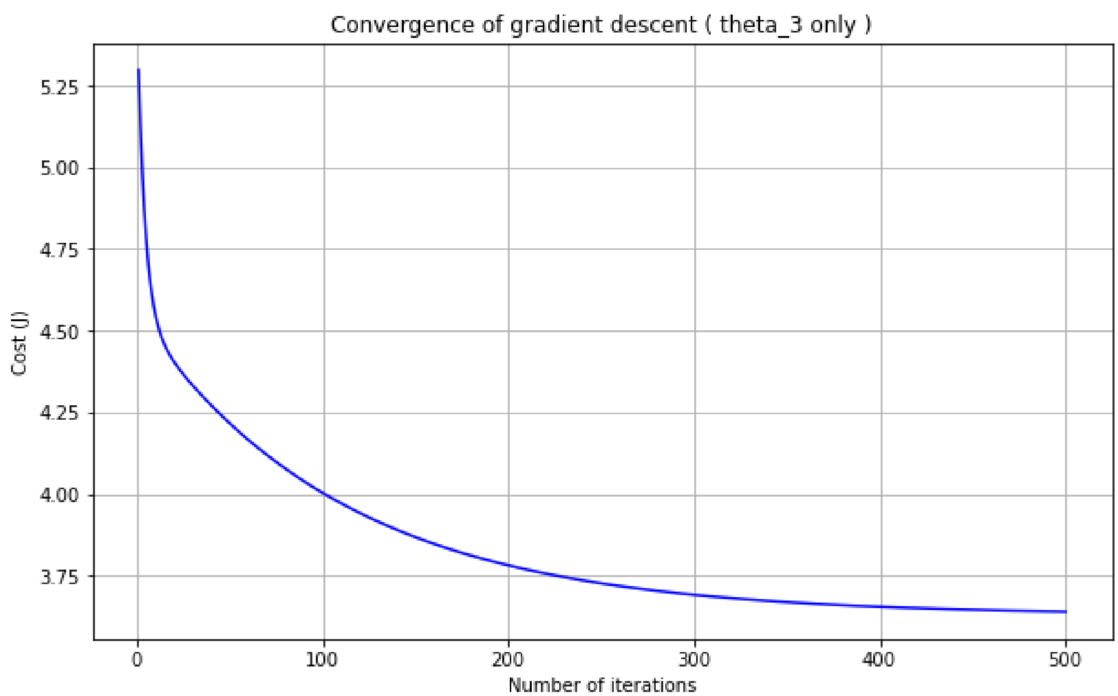

Linear Regression Fit ( theta_1 only )
X_1
H_1
6
4
2
0
−2
−4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Convergence of gradient descent ( theta_1 only )
Cost (J)
5
4
3
2
1
0
100
200
300
400
500
Number of iterations


Linear Regression Fit ( theta_2 only )
X_2
H_2
6
4
2
0
−2
−4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Convergence of gradient descent ( theta_2 only )
Cost (J)
5.0
4.8
4.6
4.4
4.2
4.0
3.8
3.6
0
100
200
300
400
500
Number of iterations




[12]:
```
def gradient_descent(x, y, theta, alpha, iterations):
    """
```

```
    Compute cost for linear regression.

    Input Parameters
    ----------------
    X : 2D array where each row represent the training example and each column␣
↪represent
      m= number of training examples
      n= number of features (including X_0 column of ones)
    y : 1D array of labels/target value for each traing example. dimension(m x␣
↪1)
    theta : 1D array of fitting parameters or weights. Dimension (1 x n)
    alpha : Learning rate. Scalar value
    iterations: No of iterations. Scalar value.

    Output Parameters
    -----------------
    theta : Final Value. 1D array of fitting parameters or weights. Dimension␣
↪(1 x n)
    cost_history: Conatins value of cost for each iteration. 1D array.␣
↪Dimansion(m x 1) """
    cost_history = np.zeros(iterations)

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

    return theta, cost_history
```

[13]:
```
iterations = 1000;
alpha = 0.08;
theta = [0,0,0,0]

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

```
Final value of theta = [ 5.31243718 -2.00347488  0.53284921 -0.26534828]
cost =
[4.10782641 3.67807605 3.34391141 3.07366878 2.85411759 2.67496473
 2.5280389  2.40684689 2.30622613 2.22206972 2.15110871 2.09073955
 2.03888743 1.99389779 1.95445041 1.91949103 1.88817721 1.85983525
 1.83392592 1.81001724 1.78776284 1.76688464 1.74715919 1.72840667
 1.71048214 1.69326862 1.6766715  1.66061408 1.64503406 1.62988065
```

| | | | | | |
|---|---|---|---|---|---|
| 1.61511236 | 1.60069513 | 1.58660089 | 1.5728064 | 1.5592923 | 1.54604238 |
| 1.53304294 | 1.52028236 | 1.50775067 | 1.49543923 | 1.48334051 | 1.47144788 |
| 1.45975543 | 1.44825784 | 1.43695029 | 1.42582837 | 1.414888 | 1.40412538 |
| 1.39353693 | 1.38311929 | 1.37286926 | 1.36278378 | 1.35285991 | 1.34309483 |
| 1.3334858 | 1.32403016 | 1.31472535 | 1.30556884 | 1.2965582 | 1.28769103 |
| 1.27896499 | 1.27037779 | 1.26192718 | 1.25361096 | 1.24542697 | 1.23737308 |
| 1.22944721 | 1.22164732 | 1.21397138 | 1.20641743 | 1.19898351 | 1.19166771 |
| 1.18446815 | 1.17738298 | 1.17041038 | 1.16354855 | 1.15679574 | 1.15015021 |
| 1.14361026 | 1.13717421 | 1.1308404 | 1.12460721 | 1.11847304 | 1.11243632 |
| 1.1064955 | 1.10064906 | 1.09489549 | 1.08923333 | 1.08366111 | 1.07817741 |
| 1.07278083 | 1.06746997 | 1.06224348 | 1.05710002 | 1.05203826 | 1.04705692 |
| 1.0421547 | 1.03733037 | 1.03258267 | 1.02791039 | 1.02331233 | 1.01878732 |
| 1.01433419 | 1.0099518 | 1.00563903 | 1.00139477 | 0.99721793 | 0.99310745 |
| 0.98906226 | 0.98508134 | 0.98116365 | 0.9773082 | 0.97351399 | 0.96978006 |
| 0.96610545 | 0.96248921 | 0.95893041 | 0.95542815 | 0.95198153 | 0.94858966 |
| 0.94525167 | 0.9419667 | 0.93873392 | 0.9355525 | 0.93242161 | 0.92934046 |
| 0.92630826 | 0.92332423 | 0.92038759 | 0.91749761 | 0.91465354 | 0.91185465 |
| 0.90910022 | 0.90638955 | 0.90372194 | 0.9010967 | 0.89851317 | 0.89597068 |
| 0.89346858 | 0.89100623 | 0.88858299 | 0.88619825 | 0.88385139 | 0.88154181 |
| 0.87926893 | 0.87703214 | 0.8748309 | 0.87266462 | 0.87053275 | 0.86843475 |
| 0.86637007 | 0.8643382 | 0.8623386 | 0.86037077 | 0.8584342 | 0.85652839 |
| 0.85465286 | 0.85280712 | 0.85099071 | 0.84920314 | 0.84744398 | 0.84571276 |
| 0.84400904 | 0.84233239 | 0.84068237 | 0.83905856 | 0.83746055 | 0.83588792 |
| 0.83434028 | 0.83281722 | 0.83131836 | 0.8298433 | 0.82839168 | 0.82696312 |
| 0.82555726 | 0.82417372 | 0.82281217 | 0.82147224 | 0.8201536 | 0.81885591 |
| 0.81757883 | 0.81632204 | 0.81508521 | 0.81386803 | 0.81267019 | 0.81149138 |
| 0.81033129 | 0.80918963 | 0.80806611 | 0.80696043 | 0.80587232 | 0.8048015 |
| 0.80374769 | 0.80271061 | 0.80169001 | 0.80068563 | 0.7996972 | 0.79872447 |
| 0.79776719 | 0.79682512 | 0.79589802 | 0.79498564 | 0.79408776 | 0.79320414 |
| 0.79233456 | 0.79147879 | 0.79063662 | 0.78980782 | 0.7889922 | 0.78818952 |
| 0.7873996 | 0.78662223 | 0.78585721 | 0.78510434 | 0.78436342 | 0.78363428 |
| 0.78291673 | 0.78221057 | 0.78151563 | 0.78083172 | 0.78015869 | 0.77949634 |
| 0.77884452 | 0.77820305 | 0.77757177 | 0.77695052 | 0.77633914 | 0.77573747 |
| 0.77514536 | 0.77456265 | 0.7739892 | 0.77342486 | 0.77286949 | 0.77232294 |
| 0.77178507 | 0.77125574 | 0.77073483 | 0.77022219 | 0.76971769 | 0.76922121 |
| 0.76873261 | 0.76825178 | 0.76777858 | 0.7673129 | 0.76685462 | 0.76640362 |
| 0.76595978 | 0.765523 | 0.76509315 | 0.76467013 | 0.76425383 | 0.76384414 |
| 0.76344097 | 0.76304419 | 0.76265372 | 0.76226946 | 0.76189129 | 0.76151914 |
| 0.7611529 | 0.76079247 | 0.76043777 | 0.76008871 | 0.75974519 | 0.75940712 |
| 0.75907443 | 0.75874702 | 0.75842482 | 0.75810773 | 0.75779568 | 0.75748859 |
| 0.75718637 | 0.75688896 | 0.75659627 | 0.75630823 | 0.75602476 | 0.7557458 |
| 0.75547127 | 0.7552011 | 0.75493523 | 0.75467357 | 0.75441608 | 0.75416267 |
| 0.75391329 | 0.75366787 | 0.75342635 | 0.75318867 | 0.75295476 | 0.75272457 |
| 0.75249803 | 0.75227509 | 0.7520557 | 0.75183979 | 0.75162731 | 0.75141821 |
| 0.75121242 | 0.75100991 | 0.75081061 | 0.75061448 | 0.75042147 | 0.75023152 |
| 0.75004459 | 0.74986062 | 0.74967958 | 0.74950142 | 0.74932609 | 0.74915354 |
| 0.74898373 | 0.74881662 | 0.74865217 | 0.74849033 | 0.74833105 | 0.74817431 |
| 0.74802006 | 0.74786826 | 0.74771887 | 0.74757185 | 0.74742717 | 0.74728479 |

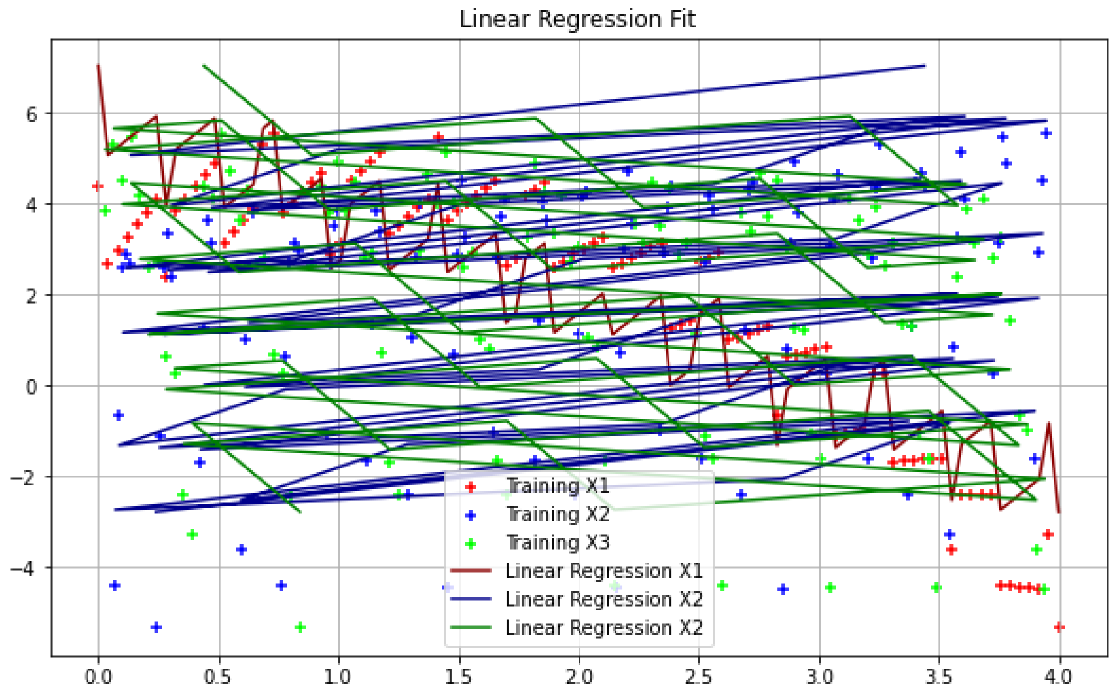
| | | | | | |
|---|---|---|---|---|---|
| 0.74714467 | 0.74700678 | 0.74687107 | 0.74673752 | 0.7466061 | 0.74647676 |
| 0.74634947 | 0.74622421 | 0.74610094 | 0.74597962 | 0.74586024 | 0.74574275 |
| 0.74562712 | 0.74551333 | 0.74540135 | 0.74529115 | 0.7451827 | 0.74507598 |
| 0.74497094 | 0.74486758 | 0.74476586 | 0.74466575 | 0.74456724 | 0.74447029 |
| 0.74437488 | 0.74428098 | 0.74418858 | 0.74409764 | 0.74400815 | 0.74392008 |
| 0.74383341 | 0.74374812 | 0.74366418 | 0.74358158 | 0.74350029 | 0.74342029 |
| 0.74334155 | 0.74326408 | 0.74318783 | 0.74311279 | 0.74303894 | 0.74296627 |
| 0.74289475 | 0.74282437 | 0.74275511 | 0.74268694 | 0.74261986 | 0.74255385 |
| 0.74248888 | 0.74242495 | 0.74236203 | 0.74230011 | 0.74223918 | 0.74217921 |
| 0.74212019 | 0.74206212 | 0.74200496 | 0.74194871 | 0.74189336 | 0.74183889 |
| 0.74178528 | 0.74173252 | 0.7416806 | 0.74162951 | 0.74157923 | 0.74152974 |
| 0.74148104 | 0.74143312 | 0.74138596 | 0.74133954 | 0.74129387 | 0.74124892 |
| 0.74120468 | 0.74116115 | 0.7411183 | 0.74107614 | 0.74103465 | 0.74099382 |
| 0.74095363 | 0.74091409 | 0.74087517 | 0.74083687 | 0.74079918 | 0.74076209 |
| 0.74072559 | 0.74068966 | 0.74065431 | 0.74061952 | 0.74058528 | 0.74055159 |
| 0.74051843 | 0.7404858 | 0.74045368 | 0.74042208 | 0.74039098 | 0.74036037 |
| 0.74033025 | 0.74030061 | 0.74027143 | 0.74024272 | 0.74021447 | 0.74018667 |
| 0.74015931 | 0.74013238 | 0.74010588 | 0.7400798 | 0.74005414 | 0.74002888 |
| 0.74000402 | 0.73997956 | 0.73995549 | 0.7399318 | 0.73990849 | 0.73988555 |
| 0.73986297 | 0.73984075 | 0.73981888 | 0.73979736 | 0.73977618 | 0.73975534 |
| 0.73973483 | 0.73971465 | 0.73969479 | 0.73967524 | 0.739656 | 0.73963707 |
| 0.73961844 | 0.7396001 | 0.73958206 | 0.7395643 | 0.73954683 | 0.73952963 |
| 0.7395127 | 0.73949605 | 0.73947966 | 0.73946353 | 0.73944765 | 0.73943203 |
| 0.73941666 | 0.73940153 | 0.73938664 | 0.73937198 | 0.73935756 | 0.73934337 |
| 0.73932941 | 0.73931566 | 0.73930214 | 0.73928883 | 0.73927573 | 0.73926284 |
| 0.73925015 | 0.73923767 | 0.73922538 | 0.73921329 | 0.73920139 | 0.73918968 |
| 0.73917815 | 0.73916681 | 0.73915565 | 0.73914467 | 0.73913386 | 0.73912322 |
| 0.73911275 | 0.73910245 | 0.73909231 | 0.73908234 | 0.73907252 | 0.73906286 |
| 0.73905335 | 0.73904399 | 0.73903478 | 0.73902571 | 0.73901679 | 0.73900802 |
| 0.73899938 | 0.73899088 | 0.73898251 | 0.73897428 | 0.73896618 | 0.7389582 |
| 0.73895036 | 0.73894263 | 0.73893503 | 0.73892756 | 0.7389202 | 0.73891295 |
| 0.73890582 | 0.73889881 | 0.73889191 | 0.73888511 | 0.73887843 | 0.73887185 |
| 0.73886537 | 0.738859 | 0.73885273 | 0.73884656 | 0.73884048 | 0.73883451 |
| 0.73882862 | 0.73882284 | 0.73881714 | 0.73881153 | 0.73880602 | 0.73880059 |
| 0.73879524 | 0.73878999 | 0.73878481 | 0.73877972 | 0.73877471 | 0.73876978 |
| 0.73876492 | 0.73876015 | 0.73875544 | 0.73875082 | 0.73874627 | 0.73874179 |
| 0.73873738 | 0.73873304 | 0.73872877 | 0.73872457 | 0.73872043 | 0.73871636 |
| 0.73871236 | 0.73870841 | 0.73870454 | 0.73870072 | 0.73869696 | 0.73869326 |
| 0.73868963 | 0.73868605 | 0.73868252 | 0.73867905 | 0.73867564 | 0.73867228 |
| 0.73866898 | 0.73866573 | 0.73866253 | 0.73865938 | 0.73865628 | 0.73865323 |
| 0.73865022 | 0.73864727 | 0.73864436 | 0.7386415 | 0.73863868 | 0.73863591 |
| 0.73863319 | 0.7386305 | 0.73862786 | 0.73862526 | 0.7386227 | 0.73862019 |
| 0.73861771 | 0.73861527 | 0.73861287 | 0.73861051 | 0.73860819 | 0.7386059 |
| 0.73860365 | 0.73860144 | 0.73859926 | 0.73859711 | 0.738595 | 0.73859292 |
| 0.73859088 | 0.73858887 | 0.73858689 | 0.73858494 | 0.73858302 | 0.73858114 |
| 0.73857928 | 0.73857745 | 0.73857565 | 0.73857388 | 0.73857214 | 0.73857043 |
| 0.73856874 | 0.73856708 | 0.73856545 | 0.73856384 | 0.73856226 | 0.7385607 |
| 0.73855917 | 0.73855766 | 0.73855618 | 0.73855471 | 0.73855328 | 0.73855186 |

| | | | | | |
|---|---|---|---|---|---|
| 0.73855047 | 0.7385491 | 0.73854775 | 0.73854643 | 0.73854512 | 0.73854384 |
| 0.73854257 | 0.73854133 | 0.7385401 | 0.7385389 | 0.73853771 | 0.73853654 |
| 0.7385354 | 0.73853427 | 0.73853315 | 0.73853206 | 0.73853098 | 0.73852992 |
| 0.73852888 | 0.73852785 | 0.73852684 | 0.73852585 | 0.73852487 | 0.7385239 |
| 0.73852296 | 0.73852202 | 0.73852111 | 0.7385202 | 0.73851931 | 0.73851844 |
| 0.73851758 | 0.73851673 | 0.7385159 | 0.73851508 | 0.73851427 | 0.73851347 |
| 0.73851269 | 0.73851192 | 0.73851116 | 0.73851042 | 0.73850969 | 0.73850896 |
| 0.73850825 | 0.73850755 | 0.73850687 | 0.73850619 | 0.73850552 | 0.73850487 |
| 0.73850422 | 0.73850359 | 0.73850296 | 0.73850235 | 0.73850174 | 0.73850115 |
| 0.73850056 | 0.73849998 | 0.73849941 | 0.73849886 | 0.73849831 | 0.73849776 |
| 0.73849723 | 0.73849671 | 0.73849619 | 0.73849568 | 0.73849519 | 0.73849469 |
| 0.73849421 | 0.73849373 | 0.73849327 | 0.7384928 | 0.73849235 | 0.7384919 |
| 0.73849146 | 0.73849103 | 0.73849061 | 0.73849019 | 0.73848978 | 0.73848937 |
| 0.73848897 | 0.73848858 | 0.73848819 | 0.73848781 | 0.73848744 | 0.73848707 |
| 0.73848671 | 0.73848635 | 0.738486 | 0.73848565 | 0.73848531 | 0.73848498 |
| 0.73848465 | 0.73848432 | 0.738484 | 0.73848369 | 0.73848338 | 0.73848308 |
| 0.73848278 | 0.73848248 | 0.73848219 | 0.73848191 | 0.73848163 | 0.73848135 |
| 0.73848108 | 0.73848081 | 0.73848055 | 0.73848029 | 0.73848004 | 0.73847978 |
| 0.73847954 | 0.73847929 | 0.73847906 | 0.73847882 | 0.73847859 | 0.73847836 |
| 0.73847814 | 0.73847792 | 0.7384777 | 0.73847748 | 0.73847727 | 0.73847707 |
| 0.73847686 | 0.73847666 | 0.73847647 | 0.73847627 | 0.73847608 | 0.73847589 |
| 0.73847571 | 0.73847552 | 0.73847535 | 0.73847517 | 0.738475 | 0.73847482 |
| 0.73847466 | 0.73847449 | 0.73847433 | 0.73847417 | 0.73847401 | 0.73847386 |
| 0.7384737 | 0.73847355 | 0.7384734 | 0.73847326 | 0.73847312 | 0.73847297 |
| 0.73847284 | 0.7384727 | 0.73847257 | 0.73847243 | 0.7384723 | 0.73847217 |
| 0.73847205 | 0.73847192 | 0.7384718 | 0.73847168 | 0.73847156 | 0.73847145 |
| 0.73847133 | 0.73847122 | 0.73847111 | 0.738471 | 0.73847089 | 0.73847079 |
| 0.73847068 | 0.73847058 | 0.73847048 | 0.73847038 | 0.73847028 | 0.73847019 |
| 0.73847009 | 0.73847 | 0.73846991 | 0.73846982 | 0.73846973 | 0.73846964 |
| 0.73846956 | 0.73846947 | 0.73846939 | 0.73846931 | 0.73846923 | 0.73846915 |
| 0.73846907 | 0.73846899 | 0.73846892 | 0.73846884 | 0.73846877 | 0.7384687 |
| 0.73846863 | 0.73846856 | 0.73846849 | 0.73846842 | 0.73846836 | 0.73846829 |
| 0.73846823 | 0.73846816 | 0.7384681 | 0.73846804 | 0.73846798 | 0.73846792 |
| 0.73846786 | 0.7384678 | 0.73846775 | 0.73846769 | 0.73846764 | 0.73846758 |
| 0.73846753 | 0.73846748 | 0.73846743 | 0.73846738 | 0.73846733 | 0.73846728 |
| 0.73846723 | 0.73846718 | 0.73846713 | 0.73846709 | 0.73846704 | 0.738467 |
| 0.73846695 | 0.73846691 | 0.73846687 | 0.73846683 | 0.73846679 | 0.73846675 |
| 0.73846671 | 0.73846667 | 0.73846663 | 0.73846659 | 0.73846655 | 0.73846652 |
| 0.73846648 | 0.73846644 | 0.73846641 | 0.73846638 | 0.73846634 | 0.73846631 |
| 0.73846628 | 0.73846624 | 0.73846621 | 0.73846618 | 0.73846615 | 0.73846612 |
| 0.73846609 | 0.73846606 | 0.73846603 | 0.738466 | 0.73846597 | 0.73846595 |
| 0.73846592 | 0.73846589 | 0.73846587 | 0.73846584 | 0.73846582 | 0.73846579 |
| 0.73846577 | 0.73846574 | 0.73846572 | 0.73846569 | 0.73846567 | 0.73846565 |
| 0.73846563 | 0.7384656 | 0.73846558 | 0.73846556 | 0.73846554 | 0.73846552 |
| 0.7384655 | 0.73846548 | 0.73846546 | 0.73846544 | 0.73846542 | 0.7384654 |
| 0.73846538 | 0.73846537 | 0.73846535 | 0.73846533 | 0.73846531 | 0.7384653 |
| 0.73846528 | 0.73846526 | 0.73846525 | 0.73846523 | 0.73846522 | 0.7384652 |
| 0.73846518 | 0.73846517 | 0.73846515 | 0.73846514 | 0.73846513 | 0.73846511 |

```
0.7384651  0.73846508 0.73846507 0.73846506 0.73846505 0.73846503
0.73846502 0.73846501 0.738465   0.73846498 0.73846497 0.73846496
0.73846495 0.73846494 0.73846493 0.73846492 0.7384649  0.73846489
0.73846488 0.73846487 0.73846486 0.73846485 0.73846484 0.73846483
0.73846482 0.73846482 0.73846481 0.7384648  0.73846479 0.73846478
0.73846477 0.73846476 0.73846475 0.73846475 0.73846474 0.73846473
0.73846472 0.73846472 0.73846471 0.7384647  0.73846469 0.73846469
0.73846468 0.73846467 0.73846466 0.73846466 0.73846465 0.73846465
0.73846464 0.73846463 0.73846463 0.73846462 0.73846461 0.73846461
0.7384646  0.7384646  0.73846459 0.73846459 0.73846458 0.73846457
0.73846457 0.73846456 0.73846456 0.73846455 0.73846455 0.73846454
0.73846454 0.73846453 0.73846453 0.73846453 0.73846452 0.73846452
0.73846451 0.73846451 0.7384645  0.7384645  0.7384645  0.73846449
0.73846449 0.73846448 0.73846448 0.73846448 0.73846447 0.73846447
0.73846446 0.73846446 0.73846446 0.73846445 0.73846445 0.73846445
0.73846444 0.73846444 0.73846444 0.73846443 0.73846443 0.73846443
0.73846443 0.73846442 0.73846442 0.73846442 0.73846441 0.73846441
0.73846441 0.73846441 0.7384644  0.7384644 ]
```

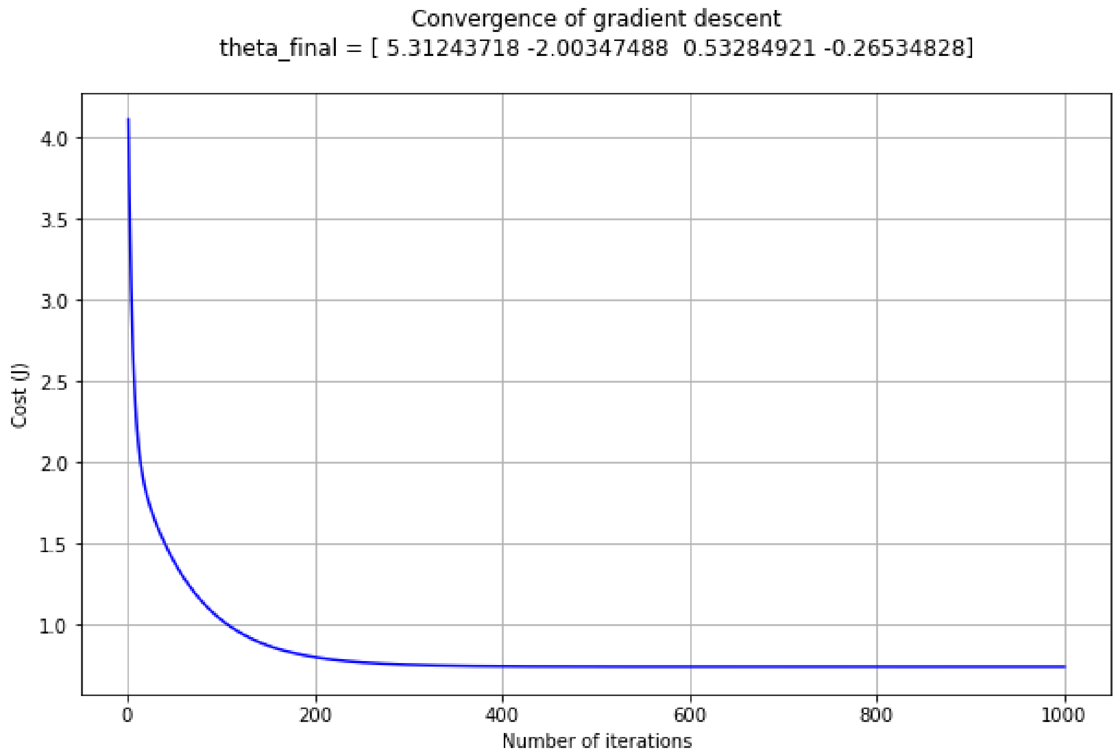
[14]:
```
# Since X is list of list (feature matrix) lets take values of column of index␣
↪1 only
plt.scatter(X[:,1], Y, color='red'  , marker= '+', label= 'Training X1')
plt.scatter(X[:,2], Y, color='blue' , marker= '+', label= 'Training X2')
plt.scatter(X[:,3], Y, color='lime' , marker= '+', label= 'Training X3')
plt.plot(X[:,1],X.dot(theta), color='darkRed', label='Linear Regression X1')
# plt.plot(X[:,2],X.dot(theta), color='darkBlue', label='Linear Regression X2')
# plt.plot(X[:,3],X.dot(theta), color='Green', label='Linear Regression X2')

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

[14]:
```
<matplotlib.legend.Legend at 0x7fdfb3d6ff70>
```




[15]:
```
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(f'Convergence of gradient descent\ntheta_final = {theta}\n')
```

[15]:
```
Text(0.5, 1.0, 'Convergence of gradient descent\ntheta_final = [ 5.31243718
-2.00347488  0.53284921 -0.26534828]\n')
```




[16]:
```
# testing it low to not overfit like above
theta = [0,0,0,0]
iterations = 500;
alpha = 0.01;

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

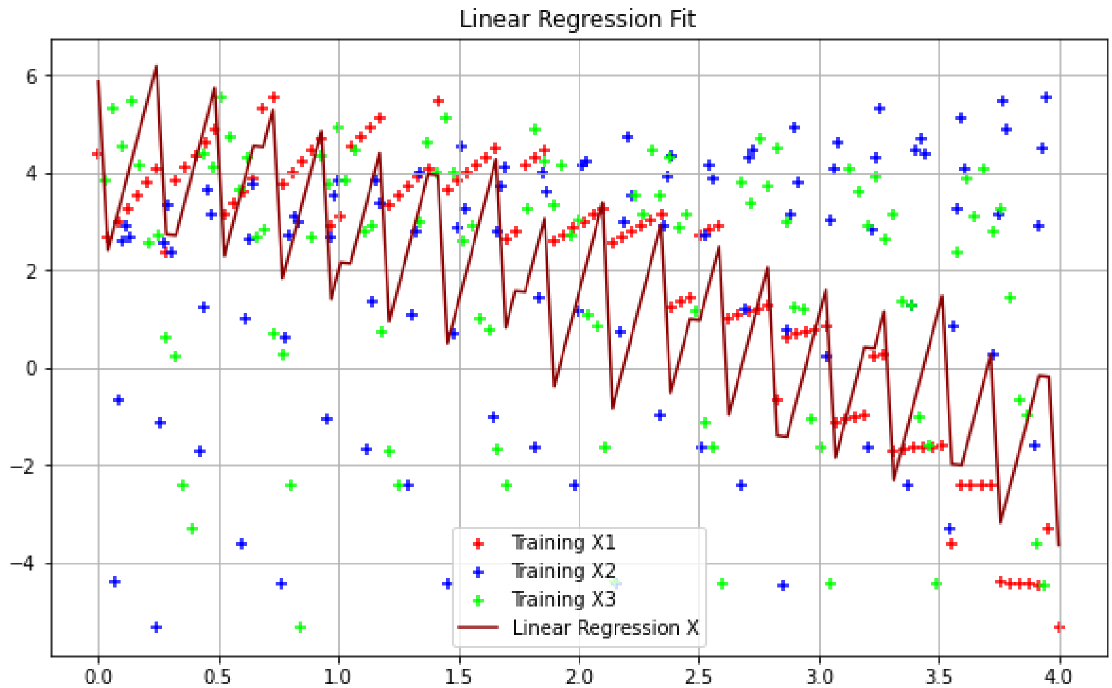
```
Final value of theta = [ 2.16125586 -1.55553751  1.05260964  0.19466187]
cost_history = [5.21542243 4.97171977 4.7765543  4.61755306 4.48558801
4.37392249
 4.27758231 4.19289222 4.11713446 4.04829701 3.98488821 3.92580007
 3.87020751 3.81749422 3.76719793 3.71897006 3.67254609 3.62772359
 3.58434612 3.54229132 3.50146212 3.46178029 3.42318165 3.38561257
 3.34902735 3.31338631 3.27865434 3.2447999  3.2117942  3.17961066
 3.14822444 3.11761213 3.08775154 3.05862146 3.0302016  3.00247242
 2.97541508 2.94901137 2.92324366 2.89809486 2.87354841 2.84958821
 2.82619863 2.80336448 2.781071   2.75930383 2.73804901 2.71729296
 2.69702247 2.67722469 2.65788713 2.63899761 2.62054429 2.60251568
 2.58490056 2.56768802 2.55086746 2.53442856 2.51836126 2.50265579
 2.48730265 2.47229256 2.45761652 2.44326576 2.42923175 2.41550619
 2.402081   2.38894831 2.37610046 2.36353002 2.35122973 2.33919253
```

| | | | | | |
|---|---|---|---|---|---|
| 2.32741157 | 2.31588015 | 2.30459179 | 2.29354014 | 2.28271905 | 2.27212252 |
| 2.26174472 | 2.25157997 | 2.24162274 | 2.23186766 | 2.22230948 | 2.21294311 |
| 2.20376359 | 2.19476609 | 2.18594592 | 2.1772985 | 2.16881937 | 2.16050421 |
| 2.1523488 | 2.14434902 | 2.13650089 | 2.12880052 | 2.12124411 | 2.11382799 |
| 2.10654856 | 2.09940234 | 2.09238592 | 2.08549599 | 2.07872933 | 2.07208282 |
| 2.06555338 | 2.05913806 | 2.05283396 | 2.04663826 | 2.04054823 | 2.03456119 |
| 2.02867454 | 2.02288577 | 2.01719239 | 2.01159202 | 2.00608233 | 2.00066104 |
| 1.99532593 | 1.99007486 | 1.98490573 | 1.9798165 | 1.97480517 | 1.96986983 |
| 1.96500857 | 1.96021957 | 1.95550104 | 1.95085124 | 1.94626848 | 1.94175112 |
| 1.93729753 | 1.93290618 | 1.92857552 | 1.92430408 | 1.92009041 | 1.91593312 |
| 1.91183083 | 1.90778222 | 1.90378597 | 1.89984084 | 1.89594559 | 1.89209903 |
| 1.88829998 | 1.88454731 | 1.88083991 | 1.87717672 | 1.87355667 | 1.86997875 |
| 1.86644196 | 1.86294533 | 1.85948793 | 1.85606882 | 1.85268712 | 1.84934196 |
| 1.84603247 | 1.84275785 | 1.83951728 | 1.83630998 | 1.83313518 | 1.82999214 |
| 1.82688013 | 1.82379846 | 1.82074642 | 1.81772336 | 1.81472862 | 1.81176155 |
| 1.80882156 | 1.80590802 | 1.80302036 | 1.800158 | 1.79732038 | 1.79450696 |
| 1.79171722 | 1.78895062 | 1.78620669 | 1.78348491 | 1.78078483 | 1.77810597 |
| 1.77544788 | 1.77281011 | 1.77019225 | 1.76759387 | 1.76501457 | 1.76245394 |
| 1.7599116 | 1.75738717 | 1.75488029 | 1.7523906 | 1.74991774 | 1.74746138 |
| 1.74502118 | 1.74259683 | 1.740188 | 1.7377944 | 1.73541571 | 1.73305166 |
| 1.73070196 | 1.72836632 | 1.72604448 | 1.72373618 | 1.72144116 | 1.71915917 |
| 1.71688997 | 1.71463332 | 1.71238898 | 1.71015674 | 1.70793637 | 1.70572766 |
| 1.70353039 | 1.70134437 | 1.6991694 | 1.69700528 | 1.69485182 | 1.69270884 |
| 1.69057616 | 1.68845361 | 1.686341 | 1.68423819 | 1.682145 | 1.68006128 |
| 1.67798686 | 1.67592161 | 1.67386538 | 1.67181801 | 1.66977937 | 1.66774933 |
| 1.66572775 | 1.6637145 | 1.66170946 | 1.65971249 | 1.65772349 | 1.65574233 |
| 1.6537689 | 1.65180309 | 1.64984478 | 1.64789387 | 1.64595027 | 1.64401385 |
| 1.64208453 | 1.64016222 | 1.6382468 | 1.6363382 | 1.63443632 | 1.63254108 |
| 1.63065239 | 1.62877016 | 1.62689432 | 1.62502478 | 1.62316147 | 1.62130431 |
| 1.61945323 | 1.61760815 | 1.615769 | 1.61393572 | 1.61210823 | 1.61028648 |
| 1.60847039 | 1.6066599 | 1.60485495 | 1.60305549 | 1.60126145 | 1.59947277 |
| 1.5976894 | 1.59591129 | 1.59413837 | 1.59237061 | 1.59060794 | 1.58885032 |
| 1.5870977 | 1.58535003 | 1.58360726 | 1.58186935 | 1.58013625 | 1.57840792 |
| 1.57668432 | 1.5749654 | 1.57325113 | 1.57154146 | 1.56983636 | 1.56813578 |
| 1.56643969 | 1.56474805 | 1.56306083 | 1.56137799 | 1.55969949 | 1.55802531 |
| 1.55635541 | 1.55468976 | 1.55302832 | 1.55137107 | 1.54971797 | 1.54806899 |
| 1.54642411 | 1.5447833 | 1.54314653 | 1.54151377 | 1.53988499 | 1.53826017 |
| 1.53663929 | 1.53502231 | 1.53340921 | 1.53179998 | 1.53019457 | 1.52859298 |
| 1.52699518 | 1.52540114 | 1.52381084 | 1.52222427 | 1.52064139 | 1.5190622 |
| 1.51748666 | 1.51591476 | 1.51434649 | 1.51278181 | 1.51122071 | 1.50966317 |
| 1.50810918 | 1.50655871 | 1.50501176 | 1.50346829 | 1.50192829 | 1.50039175 |
| 1.49885865 | 1.49732897 | 1.4958027 | 1.49427982 | 1.49276032 | 1.49124417 |
| 1.48973138 | 1.48822191 | 1.48671576 | 1.48521291 | 1.48371335 | 1.48221706 |
| 1.48072404 | 1.47923426 | 1.47774771 | 1.47626439 | 1.47478427 | 1.47330735 |
| 1.47183361 | 1.47036304 | 1.46889563 | 1.46743137 | 1.46597024 | 1.46451223 |
| 1.46305733 | 1.46160554 | 1.46015683 | 1.4587112 | 1.45726864 | 1.45582913 |
| 1.45439267 | 1.45295924 | 1.45152884 | 1.45010145 | 1.44867707 | 1.44725568 |
| 1.44583728 | 1.44442185 | 1.44300939 | 1.44159988 | 1.44019332 | 1.43878969 |

```
1.43738899 1.43599122 1.43459635 1.43320438 1.4318153  1.43042911
1.4290458  1.42766534 1.42628775 1.42491301 1.42354111 1.42217204
1.42080579 1.41944237 1.41808175 1.41672393 1.41536891 1.41401667
1.41266721 1.41132052 1.40997659 1.40863542 1.40729699 1.40596131
1.40462836 1.40329814 1.40197063 1.40064584 1.39932375 1.39800436
1.39668766 1.39537364 1.3940623  1.39275364 1.39144763 1.39014428
1.38884358 1.38754553 1.38625011 1.38495733 1.38366717 1.38237962
1.38109469 1.37981236 1.37853263 1.3772555  1.37598094 1.37470897
1.37343958 1.37217275 1.37090848 1.36964677 1.36838761 1.367131
1.36587692 1.36462537 1.36337635 1.36212985 1.36088586 1.35964439
1.35840541 1.35716894 1.35593495 1.35470345 1.35347444 1.35224789
1.35102382 1.34980221 1.34858306 1.34736636 1.34615211 1.3449403
1.34373092 1.34252398 1.34131946 1.34011737 1.33891768 1.33772041
1.33652554 1.33533308 1.334143   1.33295532 1.33177001 1.33058709
1.32940654 1.32822836 1.32705254 1.32587908 1.32470797 1.32353921
1.32237279 1.32120871 1.32004697 1.31888755 1.31773045 1.31657567
1.31542321 1.31427305 1.3131252  1.31197964 1.31083638 1.30969541
1.30855672 1.30742031 1.30628617 1.30515431 1.30402471 1.30289736
1.30177228 1.30064944 1.29952885 1.29841051 1.29729439 1.29618051
1.29506886 1.29395943 1.29285222 1.29174722 1.29064443 1.28954385
1.28844546 1.28734927 1.28625527 1.28516346 1.28407383 1.28298637
1.28190109 1.28081798 1.27973703 1.27865824 1.2775816  1.27650711
1.27543477 1.27436458 1.27329652 1.27223059 1.27116679 1.27010511
1.26904556 1.26798812]
```

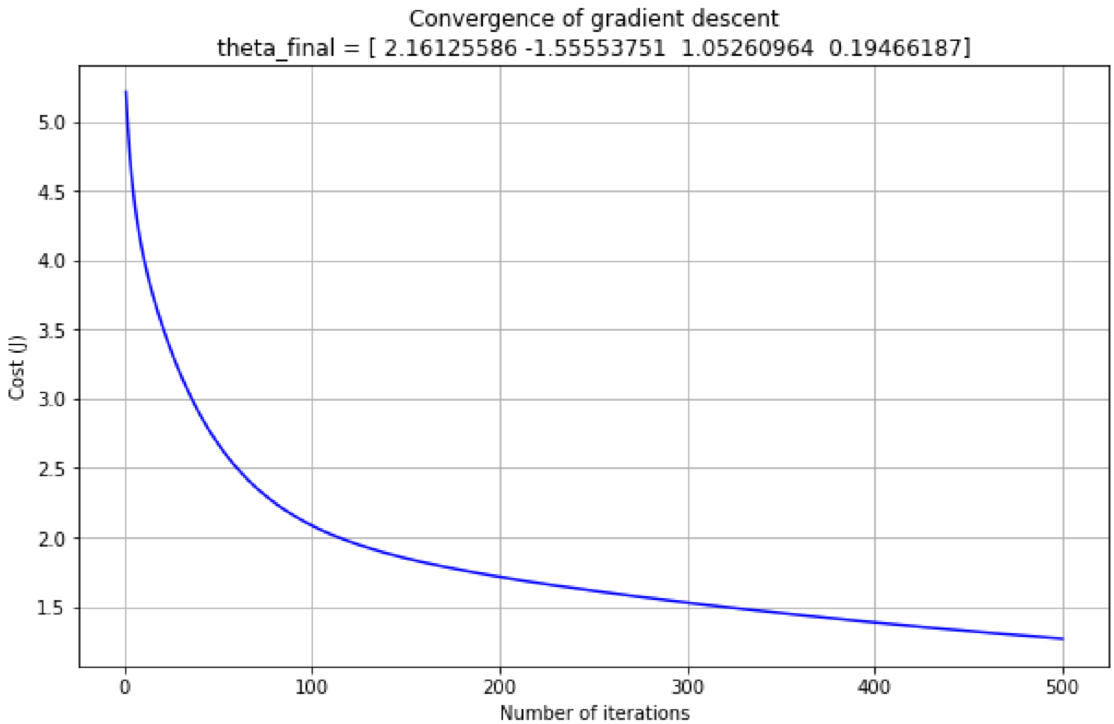
[17]:
```
# Since X is list of list (feature matrix) lets take values of column of index␣
 ↪1 only
plt.scatter(X[:,1], Y, color='red'  , marker= '+', label= 'Training X1')
plt.scatter(X[:,2], Y, color='blue' , marker= '+', label= 'Training X2')
plt.scatter(X[:,3], Y, color='lime' , marker= '+', label= 'Training X3')
plt.plot(X[:,1],X.dot(theta), color='darkRed', label='Linear Regression X')
# plt.plot(X[:,2],X.dot(theta), color='darkBlue', label='Linear Regression X2')
# plt.plot(X[:,3],X.dot(theta), color='Green', label='Linear Regression X2')

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

[17]:
```
<matplotlib.legend.Legend at 0x7fdfb2053970>
```




[18]:
```
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(f'Convergence of gradient descent\ntheta_final = {theta}')
```

[18]:
```
Text(0.5, 1.0, 'Convergence of gradient descent\ntheta_final = [ 2.16125586
-1.55553751  1.05260964  0.19466187]')
```




[19]:

```
theta_sets = np.array([ [ 0, 0, 0, 0] ,
                        [ 7, -1, 3, 1] ,
                        [-3, 2, 5, -1] ,
                        [ 2, 3, 4.6, -0.5] ,
                        [ 5.2, -2, 0.6, 0.2]
                      ]) # I used several starting thetas
iteration_set = [100, 500, 1000, 2000, 5000, 10000]
rate_set = [0.1, 0.05, 0.01, 0.001, 0.0001]
color = {0.1 : "red",
         0.05 : "purple",
         0.01 : "lime",
         0.001 : "green",
         0.0001 : "blue"
        }

counter = 0
spacing = "     "

best_theta = None
best_cost = 10000000

for theta_counter in range(0,len(theta_sets)):
    for iterations in iteration_set:
```

```
        print(f"iteration set {counter}")
        print(f"{spacing}theta = {theta_sets[theta_counter,:]} \n    iterations␣
↪= {iterations} ")

        plt.figure(counter)
        plt.rcParams["figure.figsize"] = (10,6)
        plt.grid()
        plt.xlabel('Number of iterations')
        plt.ylabel('Cost (J)')
        plt.title(f'Convergence of gradient descent: set {counter}')

        for learning_rate in rate_set:
            # print(theta_sets[theta_counter,:]) # Ensures that theta sets isnt␣
↪being changed through the runs
            theta, cost_history = gradient_descent(X, Y,␣
↪theta_sets[theta_counter,:], learning_rate, iterations)

            print(  spacing + 'rate =',learning_rate)
            print(2*spacing + 'Final value of theta =', theta)
            print(2*spacing + 'Final cost =', cost_history[-1])

            plt.plot(range(1, iterations + 1),cost_history,␣
↪color=color[learning_rate], label=f"{learning_rate}")

            # pl\nFinal Cost = {cost_history[-1]}\ntheta_final = {theta}\n')

            if cost_history[-1] < best_cost:
                best_cost = cost_history[-1]
                best_theta = theta
                best_set = counter
                best_learning_rate = learning_rate

        plt.legend(title="Learning Rate")
        counter = counter + 1
```

```
iteration set 0
    theta = [0. 0. 0. 0.]
    iterations = 100
    rate = 0.1
        Final value of theta = [ 3.40790315 -1.73442481  0.84756123  0.01381493]
        Final cost = 0.9320318861072047
    rate = 0.05
        Final value of theta = [ 2.16440604 -1.55638764  1.05226697  0.19443589]
        Final cost = 1.2669306628949162
    rate = 0.01
        Final value of theta = [ 0.60762297 -0.92952628  1.02749653  0.29002083]
        Final cost = 2.0854959880783266
```

```
    rate = 0.001
        Final value of theta = [ 0.12479902 -0.02922729  0.30290823  0.16264479]
        Final cost = 4.061011260435958
    rate = 0.0001
        Final value of theta = [0.01764344 0.00748598 0.04258771 0.02743028]
        Final cost = 5.230877369186605
iteration set 1
    theta = [0. 0. 0. 0.]
    iterations = 500
    rate = 0.1
        Final value of theta = [ 5.27950776 -1.99882305  0.53829059 -0.2605215 ]
        Final cost = 0.7385282311148745
    rate = 0.05
        Final value of theta = [ 4.88732076 -1.94342015  0.60309706 -0.20303486]
        Final cost = 0.7481695634087793
    rate = 0.01
        Final value of theta = [ 2.16125586 -1.55553751  1.05260964  0.19466187]
        Final cost = 1.2679881169698313
    rate = 0.001
        Final value of theta = [ 0.35922042 -0.52685201  0.76749911  0.25282456]
        Final cost = 2.6824046208649457
    rate = 0.0001
        Final value of theta = [0.07416997 0.00861981 0.17987562 0.10673197]
        Final cost = 4.510662934484395
iteration set 2
    theta = [0. 0. 0. 0.]
    iterations = 1000
    rate = 0.1
        Final value of theta = [ 5.31393577 -2.00368658  0.53260157 -0.26556795]
        Final cost = 0.7384642444206544
    rate = 0.05
        Final value of theta = [ 5.27907019 -1.99876124  0.53836289 -0.26045736]
        Final cost = 0.7385298570293829
    rate = 0.01
        Final value of theta = [ 3.39929705 -1.73320582  0.84898275  0.0150737 ]
        Final cost = 0.9337836009440804
    rate = 0.001
        Final value of theta = [ 0.60734448 -0.92648414  1.02489963  0.28964961]
        Final cost = 2.088281558476974
    rate = 0.0001
        Final value of theta = [ 0.12459386 -0.02957951  0.30237153  0.1622064 ]
        Final cost = 4.062288494204468
iteration set 3
    theta = [0. 0. 0. 0.]
    iterations = 2000
    rate = 0.1
        Final value of theta = [ 5.31416716 -2.00371927  0.53256334 -0.26560186]
        Final cost = 0.738464241568294
```

```
    rate = 0.05
        Final value of theta = [ 5.31392989 -2.00368575  0.53260255 -0.26556708]
        Final cost = 0.7384642445674475
    rate = 0.01
        Final value of theta = [ 4.60784132 -1.90393905  0.64927931 -0.16206885]
        Final cost = 0.7650394625052138
    rate = 0.001
        Final value of theta = [ 1.05916465 -1.28537887  1.16711429  0.30708225]
        Final cost = 1.7151921467481035
    rate = 0.0001
        Final value of theta = [ 0.1958606  -0.15472615  0.46842416  0.21010646]
        Final cost = 3.54852153011445
iteration set 4
    theta = [0. 0. 0. 0.]
    iterations = 5000
    rate = 0.1
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.05
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682943
    rate = 0.01
        Final value of theta = [ 5.27871829 -1.99871153  0.53842104 -0.26040578]
        Final cost = 0.738531179403324
    rate = 0.001
        Final value of theta = [ 2.16054884 -1.55534128  1.05268315  0.19471046]
        Final cost = 1.268225629312937
    rate = 0.0001
        Final value of theta = [ 0.35919919 -0.52657219  0.76722468  0.25279995]
        Final cost = 2.682920394509304
iteration set 5
    theta = [0. 0. 0. 0.]
    iterations = 10000
    rate = 0.1
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.05
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.01
        Final value of theta = [ 5.31392511 -2.00368507  0.53260334 -0.26556638]
        Final cost = 0.7384642446895517
    rate = 0.001
        Final value of theta = [ 3.39843999 -1.73308432  0.84912428  0.01519897]
        Final cost = 0.9339584820956784
    rate = 0.0001
        Final value of theta = [ 0.60731667 -0.92618125  1.02464108  0.28961266]
        Final cost = 2.0885601259364504
```

```
iteration set 6
    theta = [ 7. -1.  3.  1.]
    iterations = 100
    rate = 0.1
        Final value of theta = [ 5.64084826 -2.0498691   0.4785815  -0.31348613]
        Final cost = 0.7441490344715822
    rate = 0.05
        Final value of theta = [ 5.8539578  -2.08064407  0.44388109 -0.34455634]
        Final cost = 0.7539848392611772
    rate = 0.01
        Final value of theta = [ 6.12411049 -2.2733088   0.61044716 -0.43883507]
        Final cost = 0.8241221097762303
    rate = 0.001
        Final value of theta = [ 6.39873757 -2.24120624  1.5807388  -0.24141437]
        Final cost = 5.315259832332308
    rate = 0.0001
        Final value of theta = [ 6.89627435 -1.22114862  2.76131061  0.78103729]
        Final cost = 51.87887456527105
iteration set 7
    theta = [ 7. -1.  3.  1.]
    iterations = 500
    rate = 0.1
        Final value of theta = [ 5.32010684 -2.00455835  0.53158184 -0.2664725 ]
        Final cost = 0.7384661208458
    rate = 0.05
        Final value of theta = [ 5.38731688 -2.01405288  0.5204758  -0.27632413]
        Final cost = 0.7387492723897018
    rate = 0.01
        Final value of theta = [ 5.85449968 -2.08084709  0.44391703 -0.34463271]
        Final cost = 0.7540160598257616
    rate = 0.001
        Final value of theta = [ 6.17091936 -2.43029095  0.83296724 -0.51937563]
        Final cost = 0.9946160157220293
    rate = 0.0001
        Final value of theta = [ 6.60015833 -1.84116277  2.06986209  0.16368481]
        Final cost = 17.73623554373306
iteration set 8
    theta = [ 7. -1.  3.  1.]
    iterations = 1000
    rate = 0.1
        Final value of theta = [ 5.31420683 -2.00372487  0.53255678 -0.26560768]
        Final cost = 0.7384642416520637
    rate = 0.05
        Final value of theta = [ 5.32018183 -2.00456894  0.53156945 -0.26648349]
        Final cost = 0.7384661685964834
    rate = 0.01
        Final value of theta = [ 5.64232312 -2.05007824  0.47833811 -0.31370182]
        Final cost = 0.744200479723384
```

```
    rate = 0.001
        Final value of theta = [ 6.12419057 -2.27460871  0.61245615 -0.43956352]
        Final cost = 0.8250845679437001
    rate = 0.0001
        Final value of theta = [ 6.40044767 -2.2374287   1.58457398 -0.23771467]
        Final cost = 5.38618799169076
iteration set 9
    theta = [ 7. -1.  3.  1.]
    iterations = 2000
    rate = 0.1
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.05
        Final value of theta = [ 5.31420784 -2.00372501  0.53255662 -0.26560783]
        Final cost = 0.7384642416563747
    rate = 0.01
        Final value of theta = [ 5.43521195 -2.02081885  0.51256144 -0.28334457]
        Final cost = 0.7392447161550453
    rate = 0.001
        Final value of theta = [ 6.04416298 -2.14798657  0.4623284  -0.38126404]
        Final cost = 0.7697888919843413
    rate = 0.0001
        Final value of theta = [ 6.24682867 -2.48793392  1.1665632  -0.50937583]
        Final cost = 1.525210611157076
iteration set 10
    theta = [ 7. -1.  3.  1.]
    iterations = 5000
    rate = 0.1
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.05
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.01
        Final value of theta = [ 5.32024213 -2.00457746  0.53155949 -0.26649233]
        Final cost = 0.7384662074326327
    rate = 0.001
        Final value of theta = [ 5.85462134 -2.08089463  0.44392761 -0.34465038]
        Final cost = 0.7540230804830845
    rate = 0.0001
        Final value of theta = [ 6.17095469 -2.43035439  0.8332376  -0.51939202]
        Final cost = 0.9948223850687109
iteration set 11
    theta = [ 7. -1.  3.  1.]
    iterations = 10000
    rate = 0.1
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
```

```
    rate = 0.05
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682943
    rate = 0.01
        Final value of theta = [ 5.31420866 -2.00372513  0.53255648 -0.26560795]
        Final cost = 0.7384642416599609
    rate = 0.001
        Final value of theta = [ 5.64246999 -2.0500991   0.47831389 -0.31372329]
        Final cost = 0.7442056157220115
    rate = 0.0001
        Final value of theta = [ 6.12419857 -2.27473809  0.6126562  -0.43963601]
        Final cost = 0.8251808874165668
iteration set 12
    theta = [-3.  2.  5. -1.]
    iterations = 100
    rate = 0.1
        Final value of theta = [ 2.0465592  -1.54210847  1.072515    0.2133561 ]
        Final cost = 1.3072209794582035
    rate = 0.05
        Final value of theta = [-0.08494447 -1.23644088  1.42534184  0.52050632]
        Final cost = 2.2912623492791337
    rate = 0.01
        Final value of theta = [-2.73761301e+00 -3.60779438e-01  2.25409855e+00
2.19490968e-03]
        Final cost = 4.972796342067925
    rate = 0.001
        Final value of theta = [-3.34619018  0.87536034  3.79413459 -1.56208384]
        Final cost = 14.85394533847354
    rate = 0.0001
        Final value of theta = [-3.06650194  1.81550165  4.80754585 -1.12356524]
        Final cost = 41.85236865913163
iteration set 13
    theta = [-3.  2.  5. -1.]
    iterations = 500
    rate = 0.1
        Final value of theta = [ 5.25475601 -1.99532646  0.54238066 -0.2568934 ]
        Final cost = 0.7386522610374414
    rate = 0.05
        Final value of theta = [ 4.58249165 -1.90035799  0.65346819 -0.1583531 ]
        Final cost = 0.7669812347002913
    rate = 0.01
        Final value of theta = [-0.09033623 -1.23498694  1.42642428  0.52039406]
        Final cost = 2.2943741859013143
    rate = 0.001
        Final value of theta = [-3.14503418  0.06397706  2.76351113 -0.75703406]
        Final cost = 7.382796392512553
    rate = 0.0001
        Final value of theta = [-3.24562805  1.27348798  4.23285556 -1.43330425]
```

```
        Final cost = 22.58984020697077
iteration set 14
    theta = [-3.  2.  5. -1.]
    iterations = 1000
    rate = 0.1
        Final value of theta = [ 5.31377051 -2.00366323  0.53262888 -0.26554372]
        Final cost = 0.738464249949339
    rate = 0.05
        Final value of theta = [ 5.25400595 -1.9952205   0.54250461 -0.25678345]
        Final cost = 0.7386570384362362
    rate = 0.01
        Final value of theta = [ 2.03180712 -1.54001741  1.07495263  0.21551126]
        Final cost = 1.3123680154480897
    rate = 0.001
        Final value of theta = [-2.73793084 -0.35809467  2.25810936 -0.00445455]
        Final cost = 4.984630207200435
    rate = 0.0001
        Final value of theta = [-3.34494179  0.87822065  3.79703706 -1.559567  ]
        Final cost = 14.893094750485993
iteration set 15
    theta = [-3.  2.  5. -1.]
    iterations = 2000
    rate = 0.1
        Final value of theta = [ 5.31416715 -2.00371927  0.53256334 -0.26560186]
        Final cost = 0.738464241568294
    rate = 0.05
        Final value of theta = [ 5.31376044 -2.00366181  0.53263055 -0.26554225]
        Final cost = 0.7384642503806589
    rate = 0.01
        Final value of theta = [ 4.10342409 -1.83268177  0.73263114 -0.08813154]
        Final cost = 0.8165497914753693
    rate = 0.001
        Final value of theta = [-1.9758887  -0.80565624  1.82535608  0.54165941]
        Final cost = 3.6337933333131054
    rate = 0.0001
        Final value of theta = [-3.3642142   0.50002685  3.34534585 -1.4545243 ]
        Final cost = 11.03173213240137
iteration set 16
    theta = [-3.  2.  5. -1.]
    iterations = 5000
    rate = 0.1
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.05
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.01
        Final value of theta = [ 5.25340275 -1.99513529  0.54260428 -0.25669504]
```

```
        Final cost = 0.7386609239466002
    rate = 0.001
        Final value of theta = [-0.09154616 -1.23465328  1.4266717   0.52035674]
        Final cost = 2.2950732882268583
    rate = 0.0001
        Final value of theta = [-3.14502697  0.06424096  2.76397235 -0.75762113]
        Final cost = 7.384923882355383
iteration set 17
    theta = [-3.  2.  5. -1.]
    iterations = 10000
    rate = 0.1
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.05
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.01
        Final value of theta = [ 5.31375224 -2.00366065  0.5326319  -0.26554104]
        Final cost = 0.7384642507394361
    rate = 0.001
        Final value of theta = [ 2.030338   -1.53980894  1.0751954   0.21572565]
        Final cost = 1.3128818659759969
    rate = 0.0001
        Final value of theta = [-2.73796261 -0.35782731  2.25850874 -0.00511655]
        Final cost = 4.985813355938602
iteration set 18
    theta = [ 2.   3.   4.6 -0.5]
    iterations = 100
    rate = 0.1
        Final value of theta = [ 3.71347704 -1.7775873   0.79706697 -0.03098092]
        Final cost = 0.8749480620323901
    rate = 0.05
        Final value of theta = [ 2.66929842 -1.62477346  0.96980698  0.11643869]
        Final cost = 1.1111160180476878
    rate = 0.01
        Final value of theta = [ 1.36701311 -0.78884539  1.41142625 -0.58919007]
        Final cost = 2.3572460504747528
    rate = 0.001
        Final value of theta = [ 1.24870161  0.99985072  2.69804474 -1.86901446]
        Final cost = 16.038713700790964
    rate = 0.0001
        Final value of theta = [ 1.86621818  2.66763677  4.27962416 -0.76287155]
        Final cost = 101.90298695526357
iteration set 19
    theta = [ 2.   3.   4.6 -0.5]
    iterations = 500
    rate = 0.1
        Final value of theta = [ 5.28506366 -1.99960792  0.53737251 -0.26133588]
```

```
        Final cost = 0.7385093603576643
    rate = 0.05
        Final value of theta = [ 4.9557442  -1.95308609  0.59179051 -0.21306434]
        Final cost = 0.7453074276086923
    rate = 0.01
        Final value of theta = [ 2.66665307 -1.62357271  0.97033656  0.11588561]
        Final cost = 1.1118728129797173
    rate = 0.001
        Final value of theta = [ 1.16606113 -0.24889625  1.71069643 -1.34636206]
        Final cost = 4.385592089616808
    rate = 0.0001
        Final value of theta = [ 1.49090682  1.69831575  3.35239828 -1.47020773]
        Final cost = 39.332552329745404
iteration set 20
    theta = [ 2.   3.   4.6 -0.5]
    iterations = 1000
    rate = 0.1
        Final value of theta = [ 5.31397286 -2.00369182  0.53259544 -0.26557338]
        Final cost = 0.7384642435794827
    rate = 0.05
        Final value of theta = [ 5.28469623 -1.99955601  0.53743323 -0.26128202]
        Final cost = 0.7385105067838877
```

```
/tmp/ipykernel_36108/3765187480.py:27: RuntimeWarning: More than 20 figures have
been opened. Figures created through the pyplot interface
(`matplotlib.pyplot.figure`) are retained until explicitly closed and may
consume too much memory. (To control this warning, see the rcParam
`figure.max_open_warning`).
  plt.figure(counter)
```

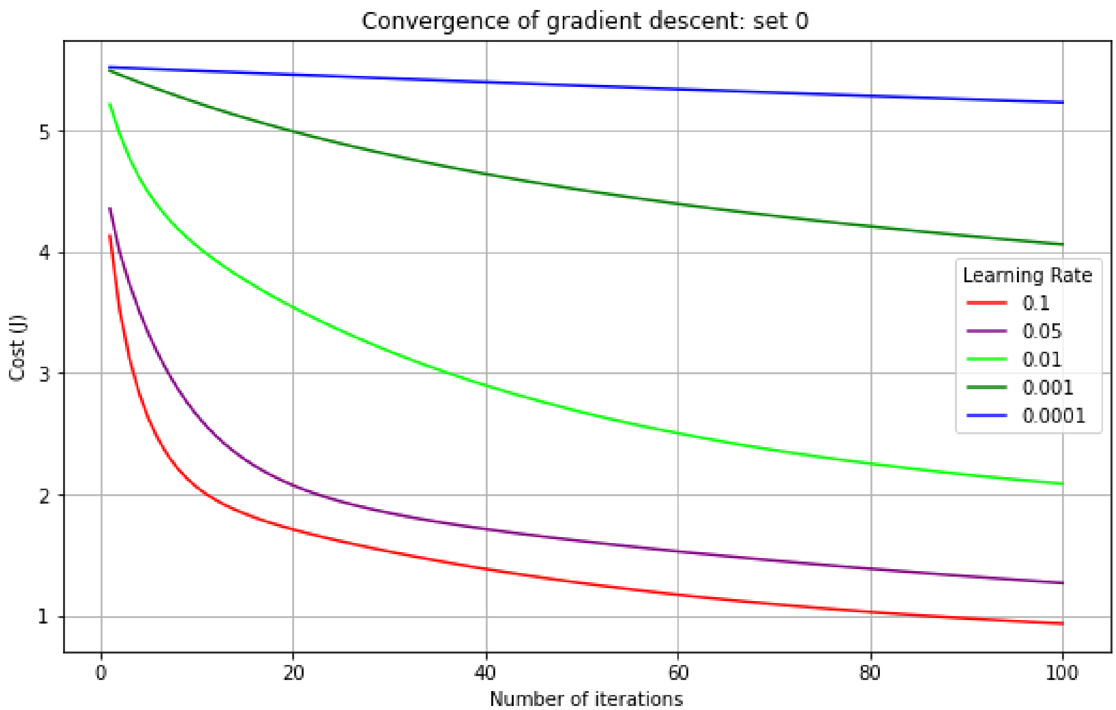
```
    rate = 0.01
        Final value of theta = [ 3.70625046 -1.7765584   0.79826078 -0.02992954]
        Final cost = 0.8761831897979573
    rate = 0.001
        Final value of theta = [ 1.36684386 -0.78498163  1.41385209 -0.59551588]
        Final cost = 2.368163493833769
    rate = 0.0001
        Final value of theta = [ 1.25099976  1.00509817  2.70320001 -1.86419085]
        Final cost = 16.16803672752888
iteration set 21
    theta = [ 2.   3.   4.6 -0.5]
    iterations = 2000
    rate = 0.1
        Final value of theta = [ 5.31416716 -2.00371927  0.53256334 -0.26560186]
        Final cost = 0.7384642415682943
    rate = 0.05
        Final value of theta = [ 5.31396793 -2.00369112  0.53259626 -0.26557266]
        Final cost = 0.7384642436829859
    rate = 0.01
```

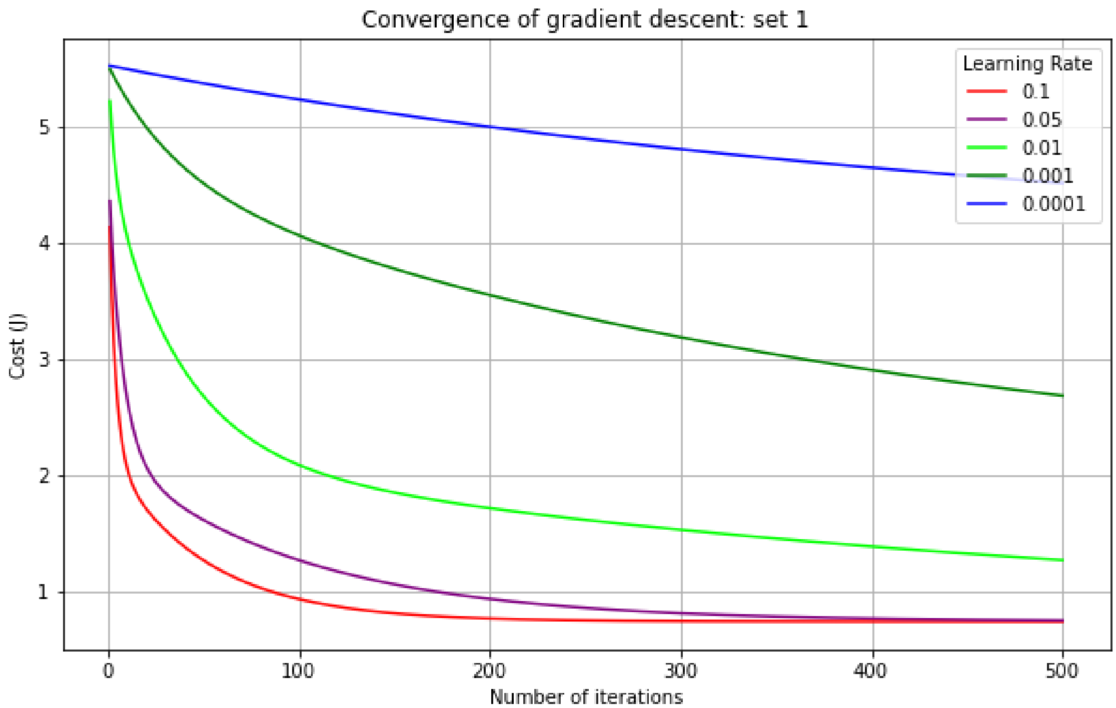
```
        Final value of theta = [ 4.72106529 -1.91993381  0.63056975 -0.17866518]
        Final cost = 0.7572023302092515
    rate = 0.001
        Final value of theta = [ 1.74204112 -1.28928092  1.17166586 -0.00627824]
        Final cost = 1.4879885638018455
    rate = 0.0001
        Final value of theta = [ 1.10142566  0.3748325   2.14741656 -1.96277399]
        Final cost = 7.856272270874399
iteration set 22
    theta = [ 2.   3.   4.6 -0.5]
    iterations = 5000
    rate = 0.1
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.05
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.738464241568294
    rate = 0.01
        Final value of theta = [ 5.28440075 -1.99951427  0.53748205 -0.26123871]
        Final cost = 0.7385114391847956
    rate = 0.001
        Final value of theta = [ 2.66605941 -1.6232939   0.97045795  0.11574934]
        Final cost = 1.1120431736876832
    rate = 0.0001
        Final value of theta = [ 1.16609222 -0.24847667  1.71103582 -1.34686934]
        Final cost = 4.387477410557966
iteration set 23
    theta = [ 2.   3.   4.6 -0.5]
    iterations = 10000
    rate = 0.1
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.05
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.01
        Final value of theta = [ 5.31396391 -2.00369055  0.53259692 -0.26557207]
        Final cost = 0.7384642437690814
    rate = 0.001
        Final value of theta = [ 3.70553078 -1.77645567  0.79837967 -0.0298251 ]
        Final cost = 0.8763064978791245
    rate = 0.0001
        Final value of theta = [ 1.36682697 -0.78459688  1.41409367 -0.59614565]
        Final cost = 2.369254648051613
iteration set 24
    theta = [ 5.2 -2.   0.6  0.2]
    iterations = 100
    rate = 0.1
```

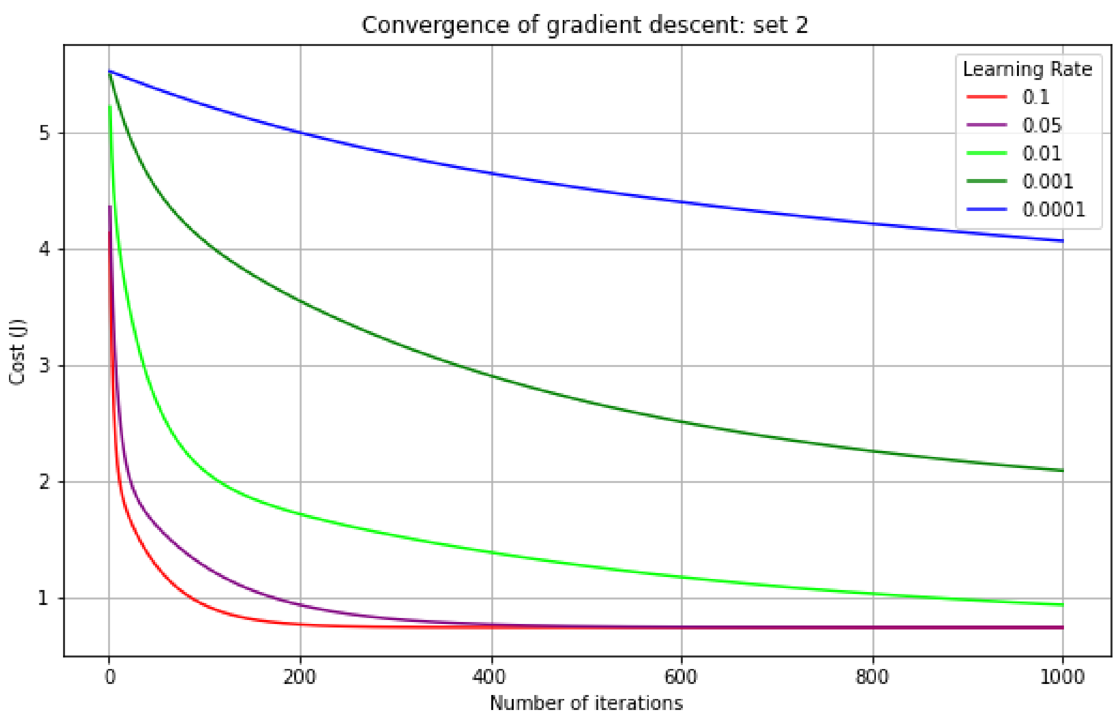
```
        Final value of theta = [ 5.24748284 -1.99429971  0.54358255 -0.25582661]
        Final cost = 0.7387011139370278
    rate = 0.05
        Final value of theta = [ 5.2039878  -1.9886222   0.55074227 -0.24894485]
        Final cost = 0.7391112139617002
    rate = 0.01
        Final value of theta = [ 5.14987858 -2.03695771  0.53516948 -0.15891936]
        Final cost = 0.7464064598918929
    rate = 0.001
        Final value of theta = [ 5.15039337 -2.09732834  0.50184817  0.05288834]
        Final cost = 0.8312094173309614
    rate = 0.0001
        Final value of theta = [ 5.19123526 -2.01816017  0.58190926  0.17678184]
        Final cost = 1.2004399482128383
iteration set 25
    theta = [ 5.2 -2.   0.6  0.2]
    iterations = 500
    rate = 0.1
        Final value of theta = [ 5.31295473 -2.00354799  0.53276369 -0.26542415]
        Final cost = 0.738464319873506
    rate = 0.05
        Final value of theta = [ 5.29923536 -2.0016099   0.53503073 -0.26341317]
        Final cost = 0.7384761181532145
    rate = 0.01
        Final value of theta = [ 5.20387826 -1.98867906  0.55074961 -0.24884379]
        Final cost = 0.739112605598407
    rate = 0.001
        Final value of theta = [ 5.14142645 -2.07879271  0.50843951 -0.08482635]
        Final cost = 0.7627911507363951
    rate = 0.0001
        Final value of theta = [ 5.16654209 -2.06779964  0.53213971  0.1070025 ]
        Final cost = 0.933383954710693
iteration set 26
    theta = [ 5.2 -2.   0.6  0.2]
    iterations = 1000
    rate = 0.1
        Final value of theta = [ 5.31415908 -2.00371813  0.53256467 -0.26560068]
        Final cost = 0.7384642415717848
    rate = 0.05
        Final value of theta = [ 5.31293942 -2.00354583  0.53276621 -0.2654219 ]
        Final cost = 0.7384643218631683
    rate = 0.01
        Final value of theta = [ 5.24718179 -1.99425789  0.54363232 -0.25578178]
        Final cost = 0.7387032575451623
    rate = 0.001
        Final value of theta = [ 5.14986979 -2.03727875  0.53491561 -0.15833007]
        Final cost = 0.7464974267561909
    rate = 0.0001
```

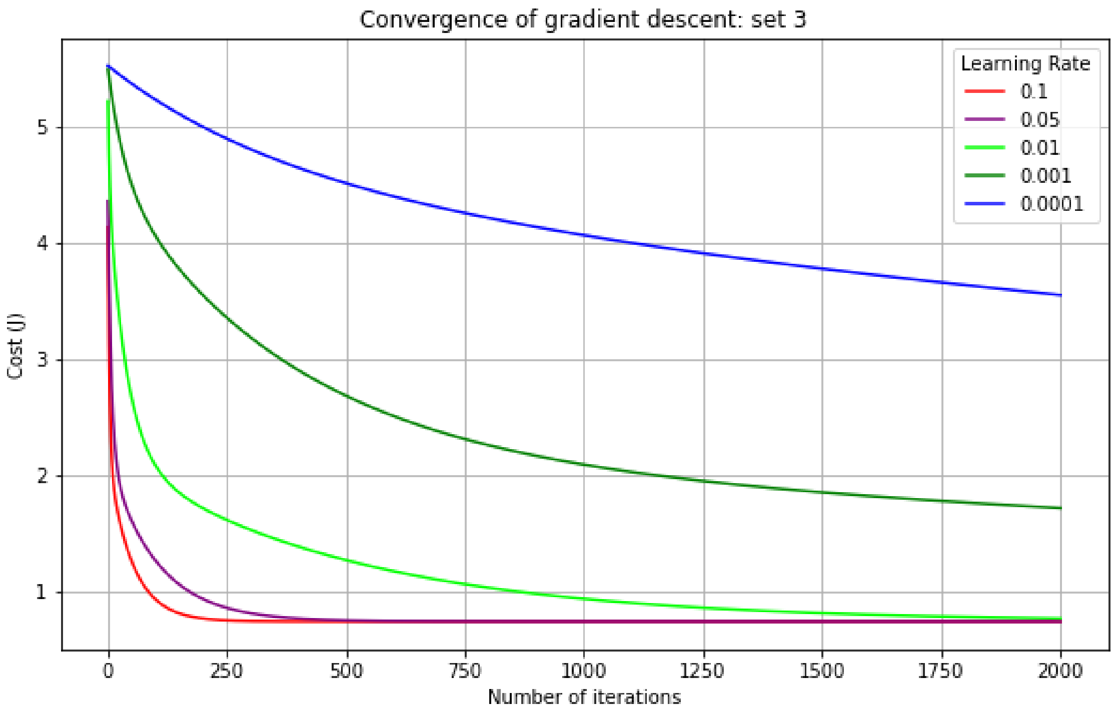
```
        Final value of theta = [ 5.15054238 -2.09700291  0.502167    0.05323244]
        Final cost = 0.831758187727815
iteration set 27
    theta = [ 5.2 -2.   0.6  0.2]
    iterations = 2000
    rate = 0.1
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682943
    rate = 0.05
        Final value of theta = [ 5.31415887 -2.0037181   0.53256471 -0.26560065]
        Final cost = 0.7384642415719641
    rate = 0.01
        Final value of theta = [ 5.2894587  -2.00022879  0.53664626 -0.26198011]
        Final cost = 0.7384967621661084
    rate = 0.001
        Final value of theta = [ 5.1654581  -2.00089152  0.55175712 -0.21974711]
        Final cost = 0.7402122518238244
    rate = 0.0001
        Final value of theta = [ 5.13995409e+00 -2.10880894e+00  4.87858812e-01
-4.49826073e-03]
        Final cost = 0.789946410145446
iteration set 28
    theta = [ 5.2 -2.   0.6  0.2]
    iterations = 5000
    rate = 0.1
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.05
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682943
    rate = 0.01
        Final value of theta = [ 5.31292711 -2.00354409  0.53276825 -0.2654201 ]
        Final cost = 0.7384643234813822
    rate = 0.001
        Final value of theta = [ 5.20385368 -1.98869263  0.55075098 -0.24882001]
        Final cost = 0.7391129215090445
    rate = 0.0001
        Final value of theta = [ 5.14142839 -2.07881446  0.50841842 -0.08476555]
        Final cost = 0.7628071035721903
iteration set 29
    theta = [ 5.2 -2.   0.6  0.2]
    iterations = 10000
    rate = 0.1
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
    rate = 0.05
        Final value of theta = [ 5.31416717 -2.00371927  0.53256334 -0.26560187]
        Final cost = 0.7384642415682942
```

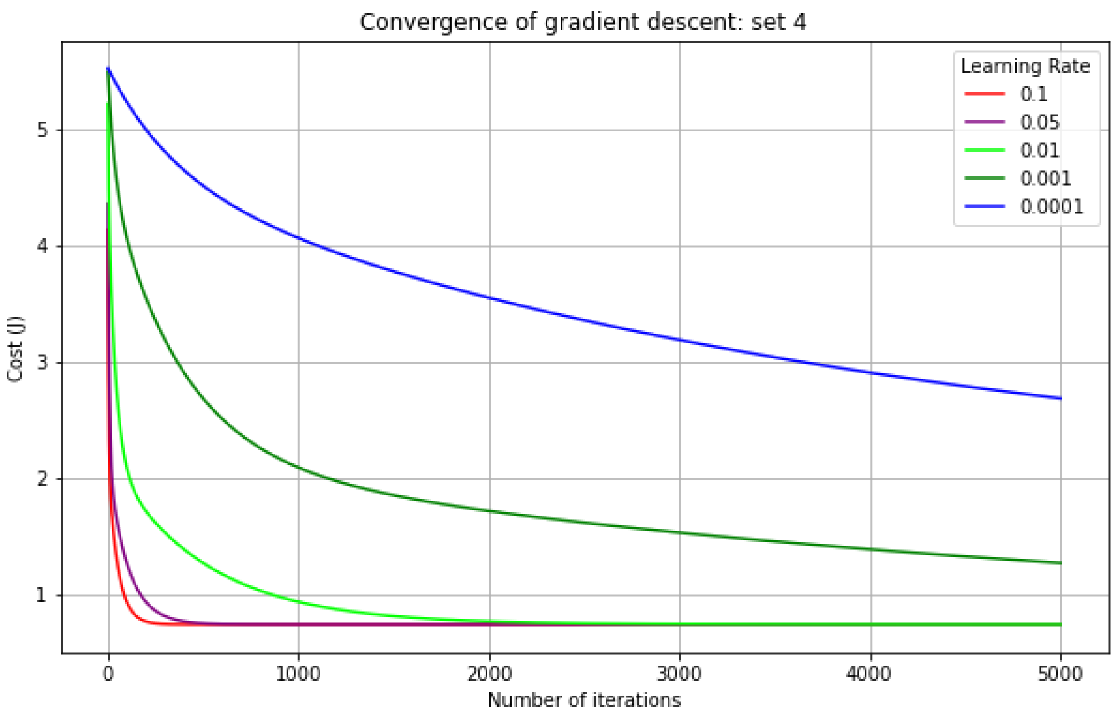
```
rate = 0.01
    Final value of theta = [ 5.3141587  -2.00371807  0.53256474 -0.26560062]
    Final cost = 0.7384642415721138
rate = 0.001
    Final value of theta = [ 5.24715181 -1.99425375  0.54363728 -0.25577729]
    Final cost = 0.7387034715508343
rate = 0.0001
    Final value of theta = [ 5.14986892 -2.03731071  0.53489033 -0.1582714 ]
    Final cost = 0.7465065192787477
```

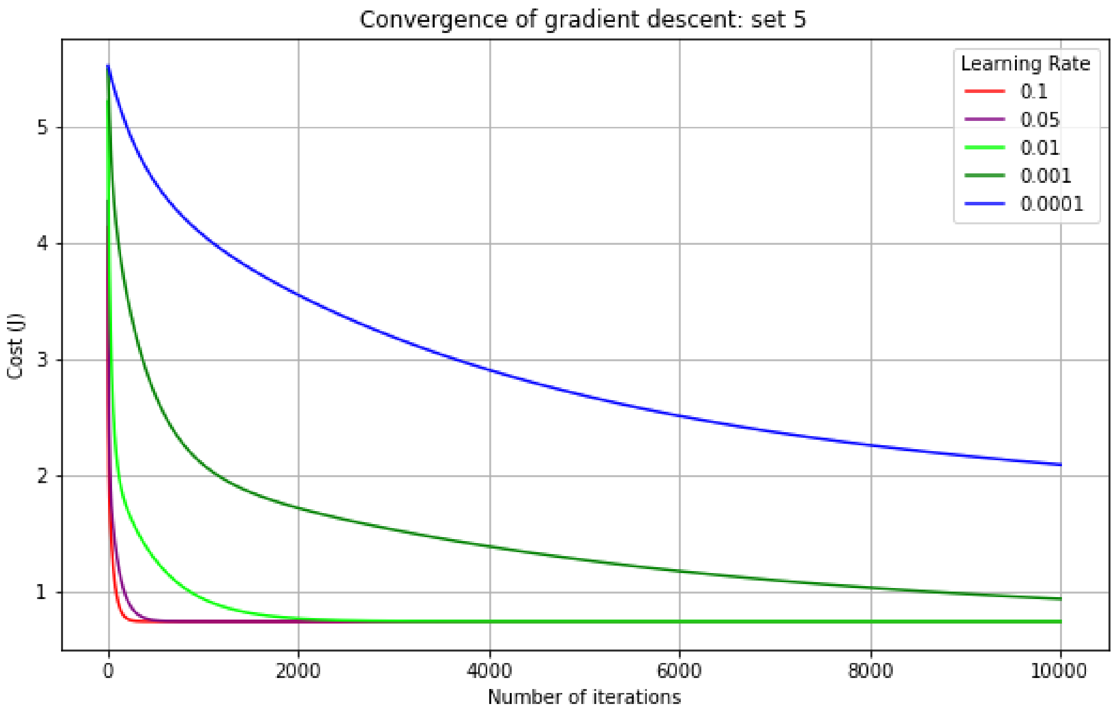


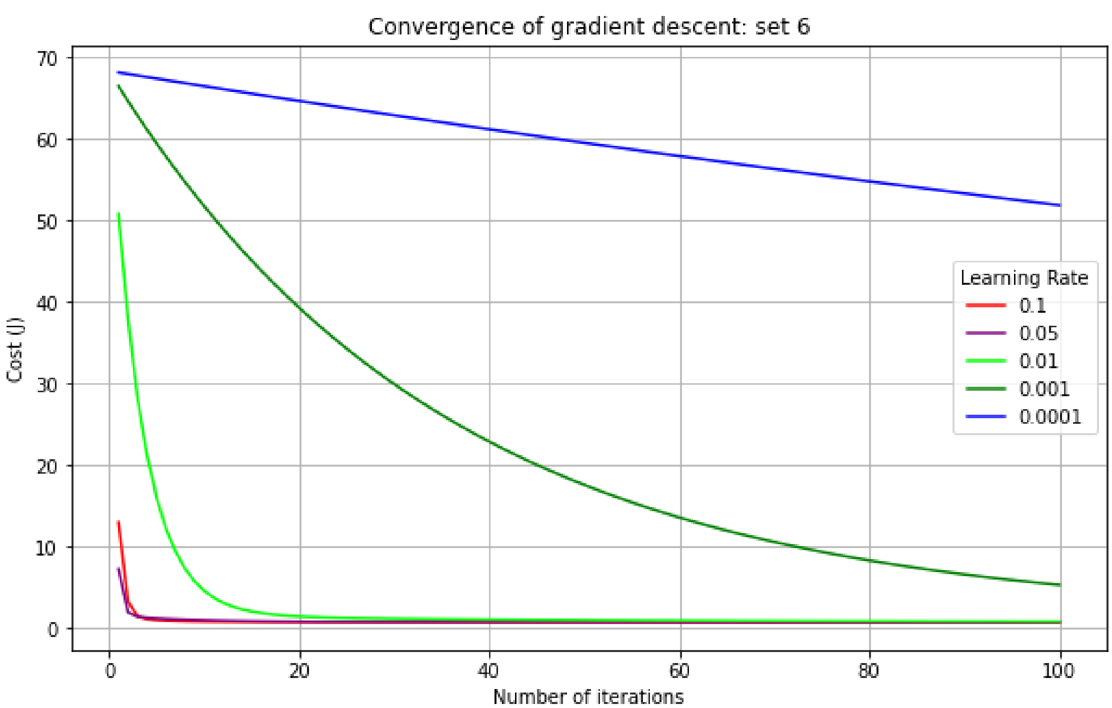

Convergence of gradient descent: set 1
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
100
200
300
400
500
1
2
3
4
5

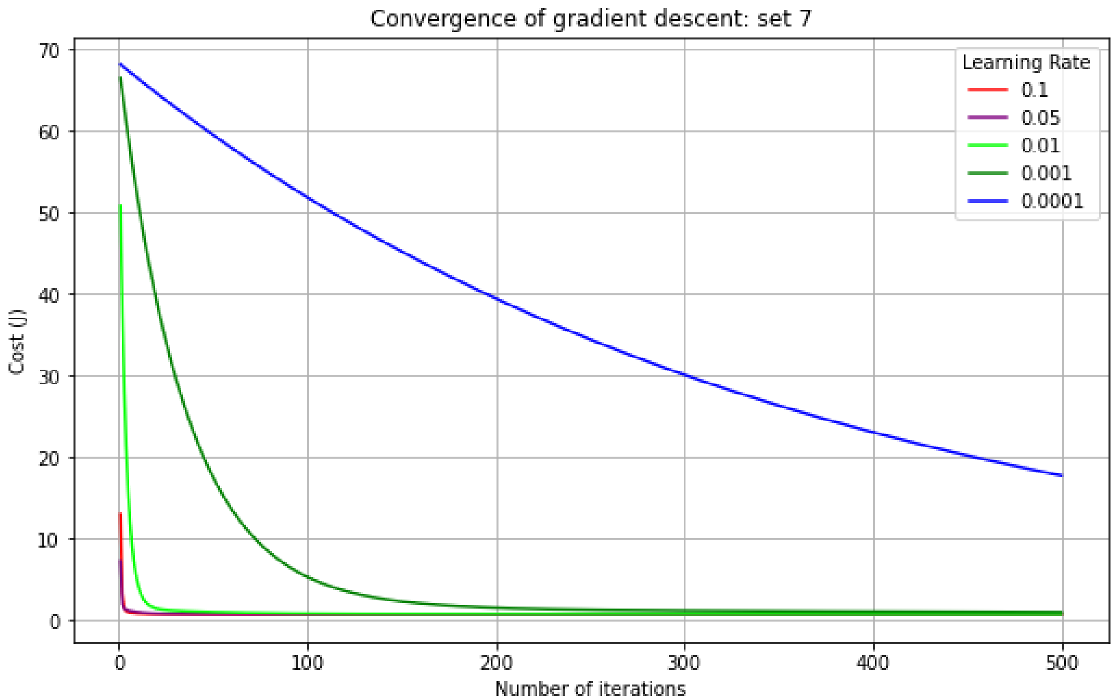

Convergence of gradient descent: set 2
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
200
400
600
800
1000
1
2
3
4
5


Convergence of gradient descent: set 3
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
250
500
750
1000
1250
1500
1750
2000
1
2
3
4
5

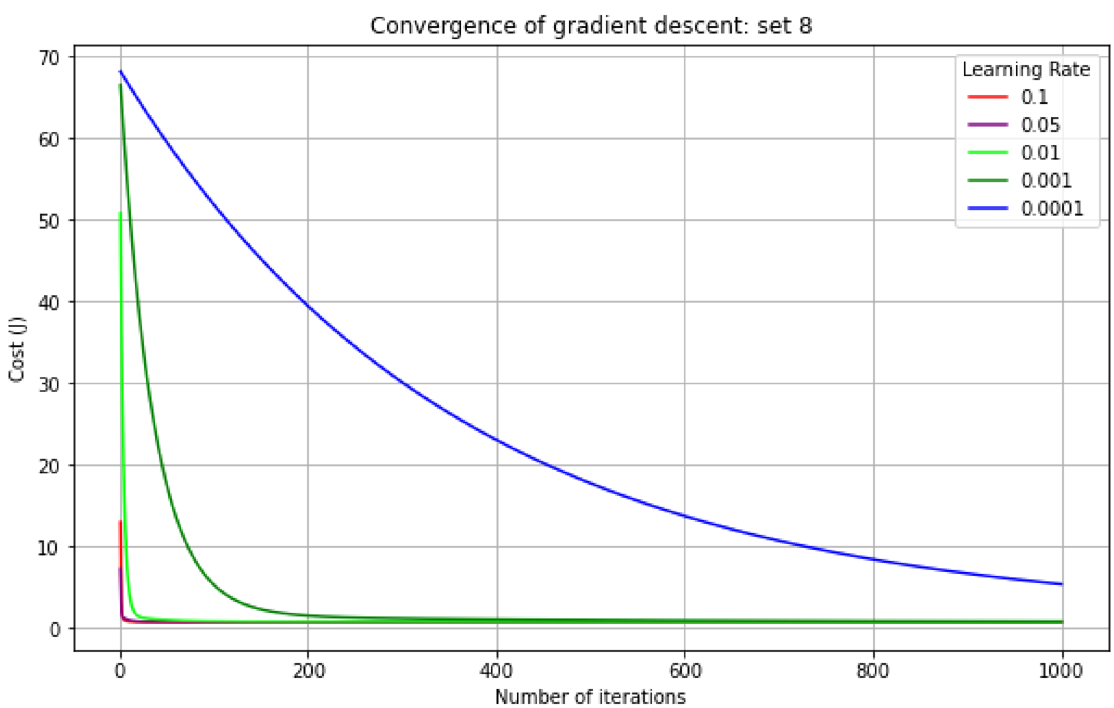

Convergence of gradient descent: set 4
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
1000
2000
3000
4000
5000
1
2
3
4
5


Convergence of gradient descent: set 5
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
2000
4000
6000
8000
10000
1
2
3
4
5

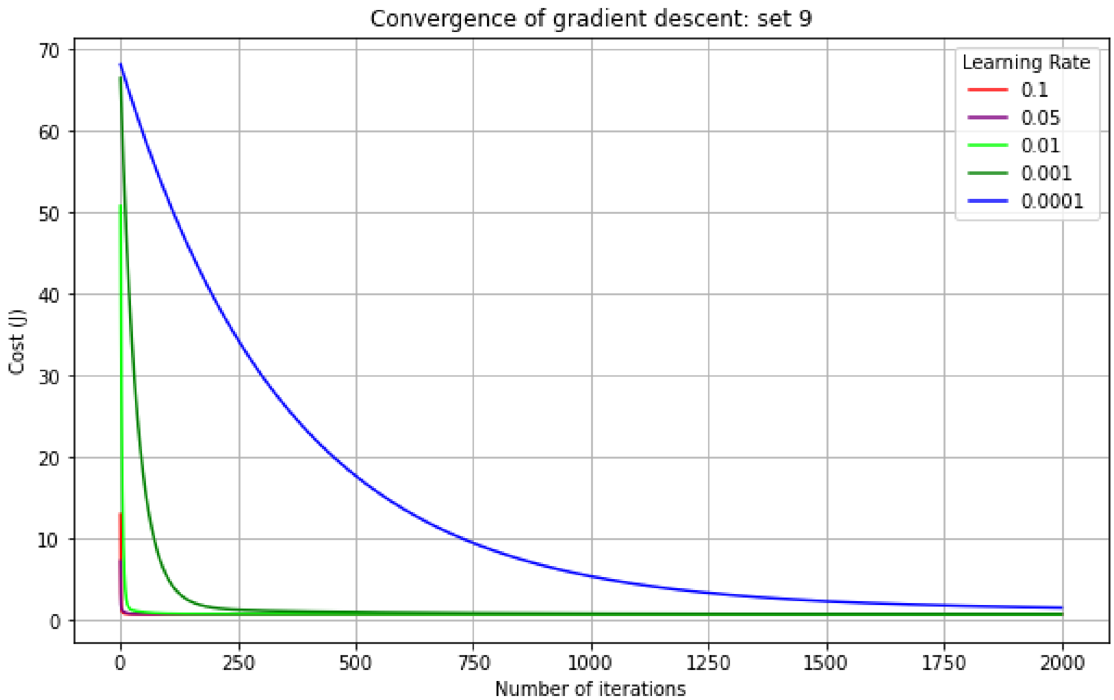

Convergence of gradient descent: set 6
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
20
40
60
80
100
10
30
50
70

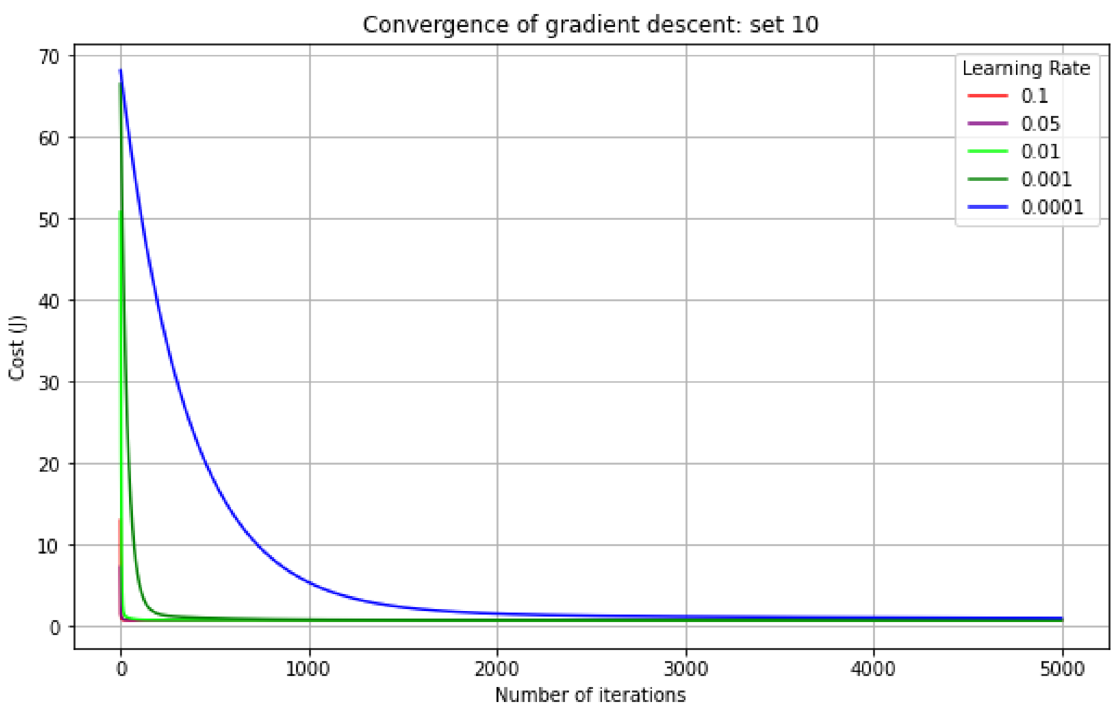


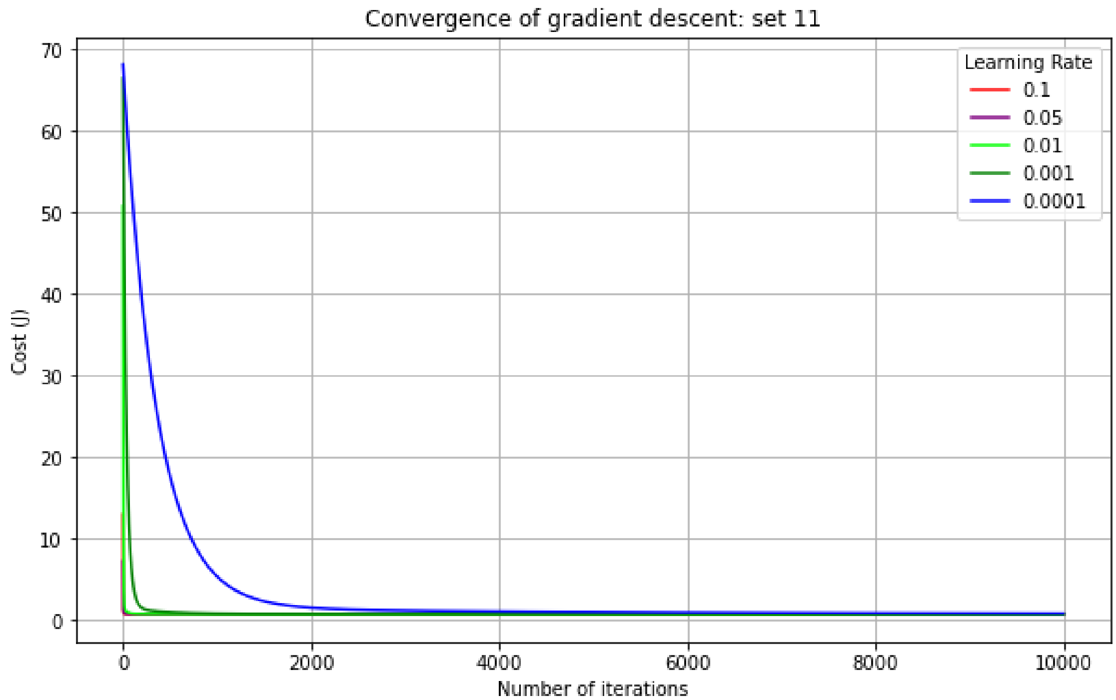
Convergence of gradient descent: set 7
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations

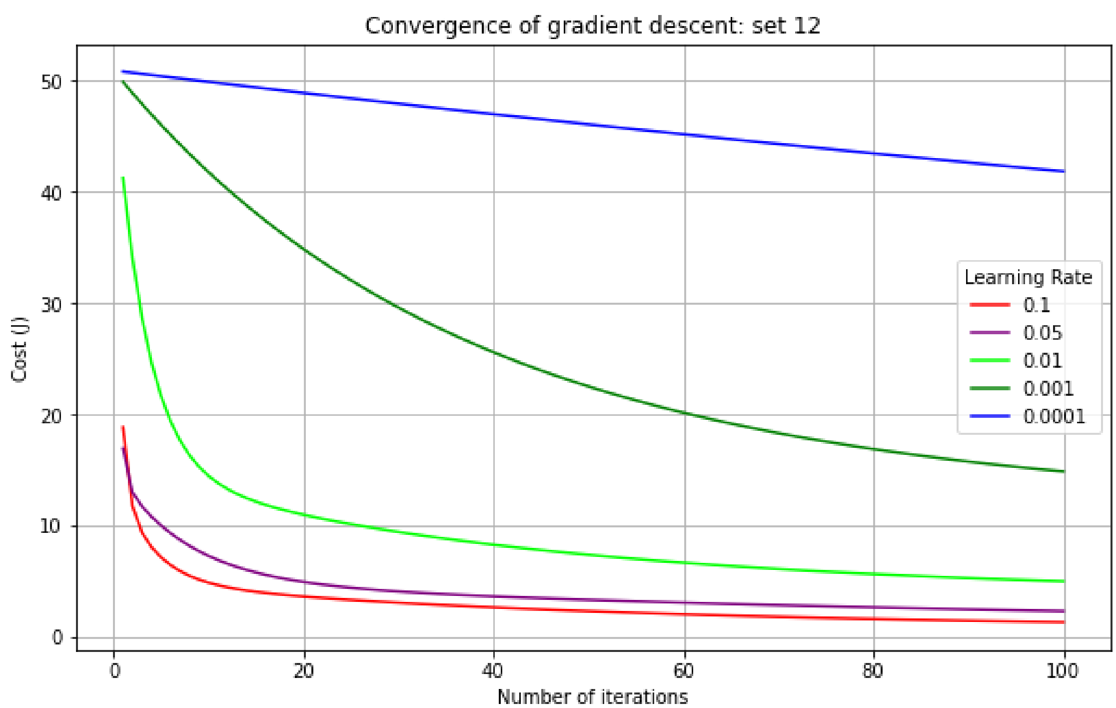



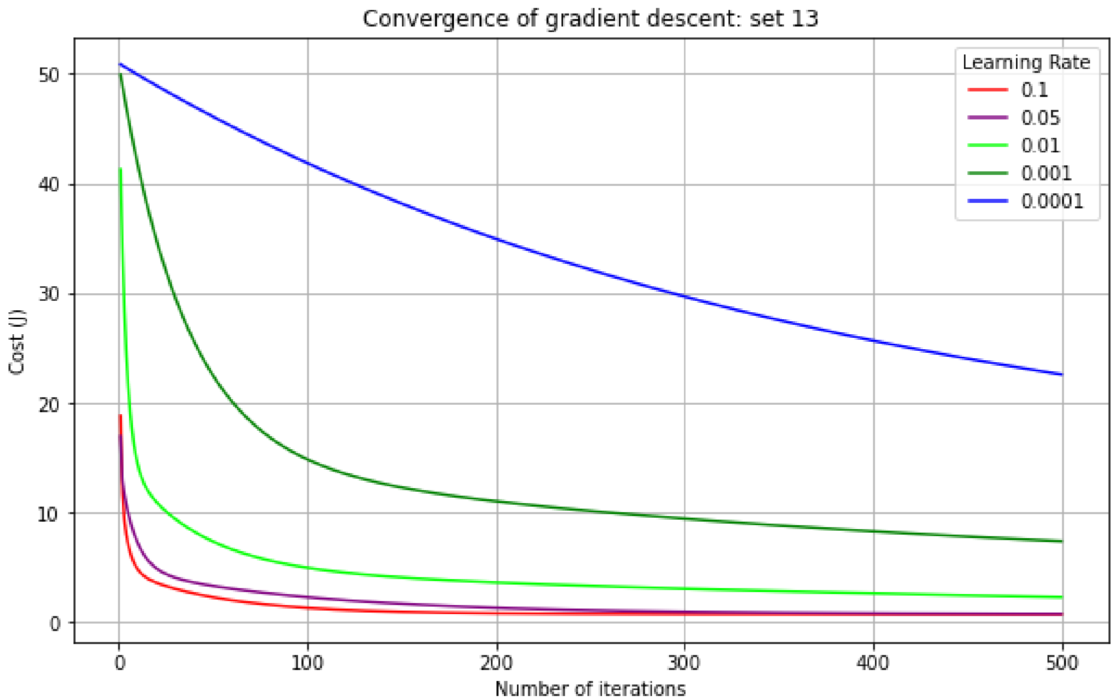
Convergence of gradient descent: set 8
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations

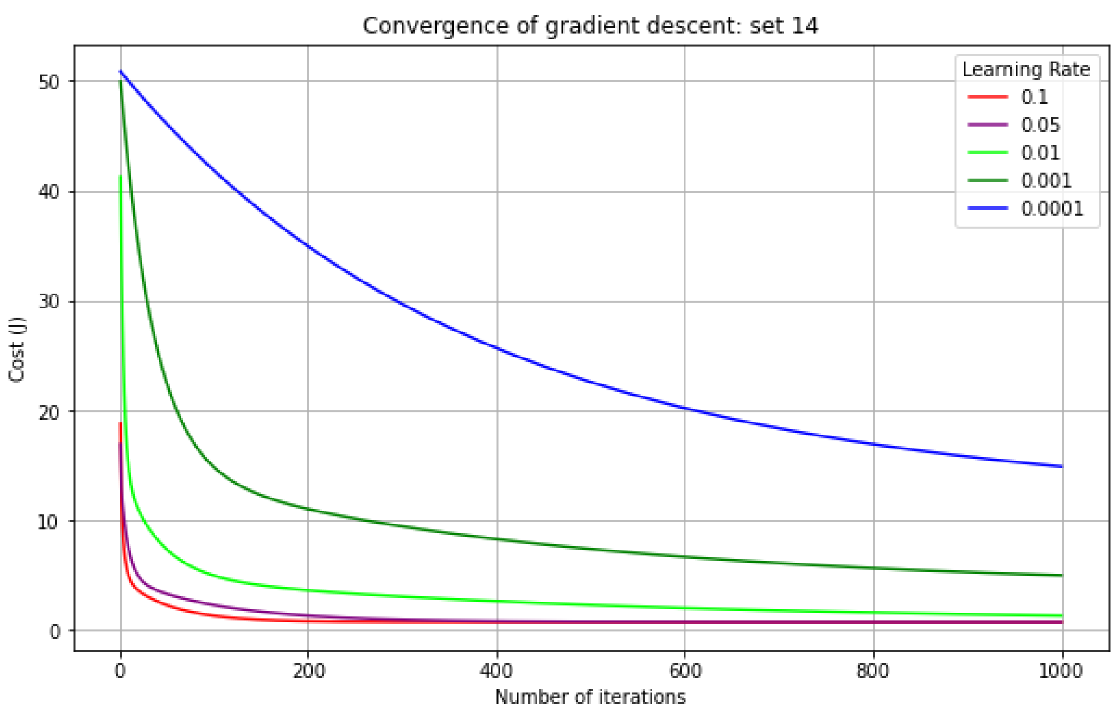

Convergence of gradient descent: set 9
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
10
20
30
40
50
60
70
0
250
500
750
1000
1250
1500
1750
2000

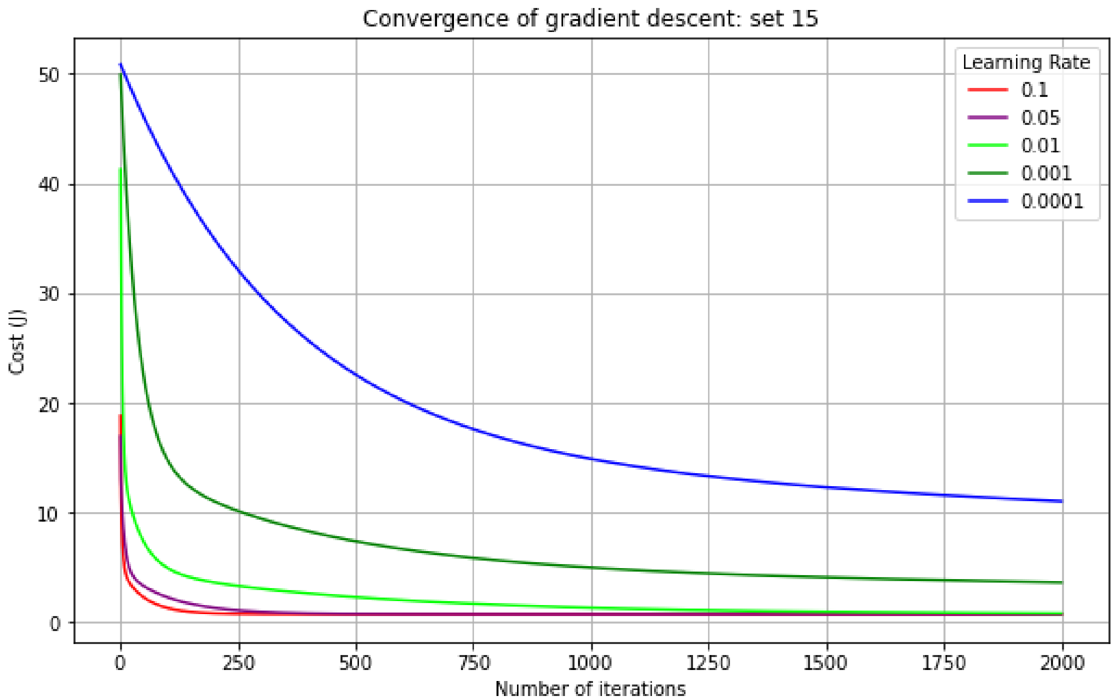

Convergence of gradient descent: set 10
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
10
20
30
40
50
60
70
0
1000
2000
3000
4000
5000

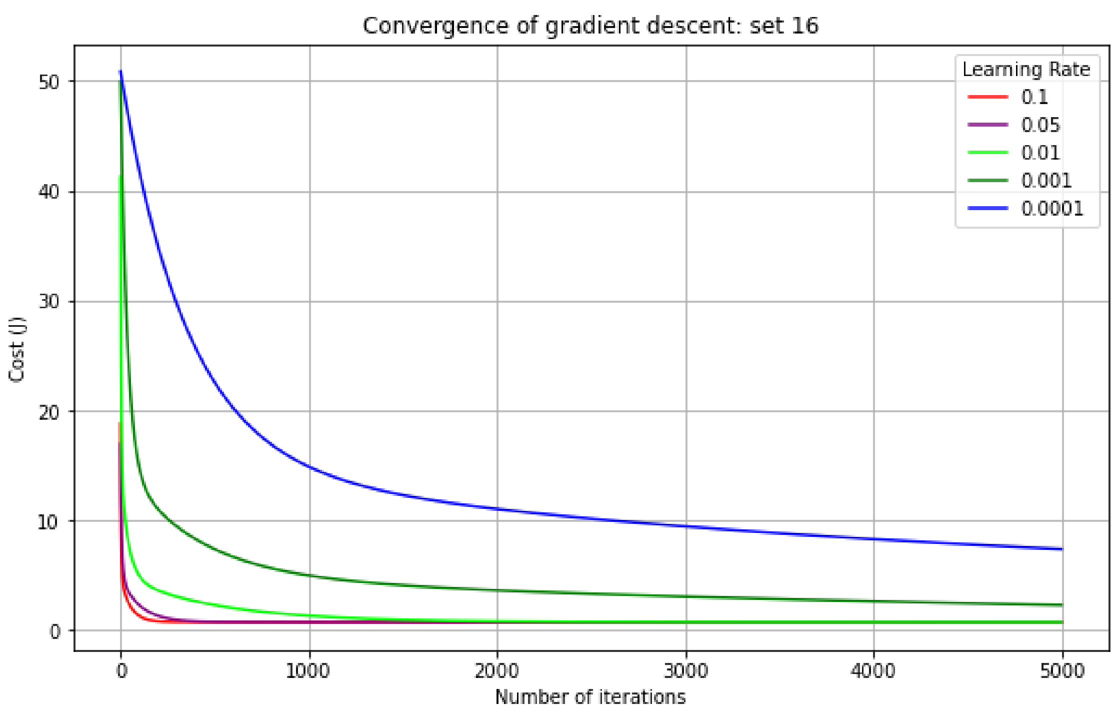

Convergence of gradient descent: set 11
Cost (J)
Number of iterations
Learning Rate
0.1
0.05
0.01
0.001
0.0001
0
10
20
30
40
50
60
70
0
2000
4000
6000
8000
10000

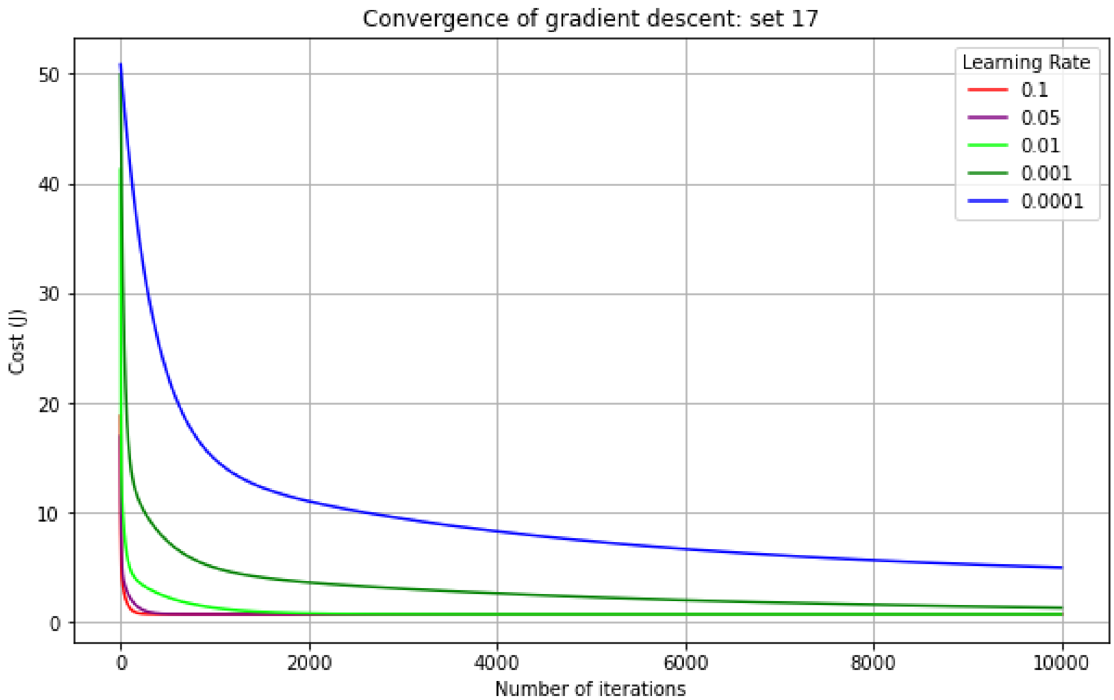

Convergence of gradient descent: set 12
Cost (J)
Number of iterations
Learning Rate
0.1
0.05
0.01
0.001
0.0001
0
10
20
30
40
50
0
20
40
60
80
100

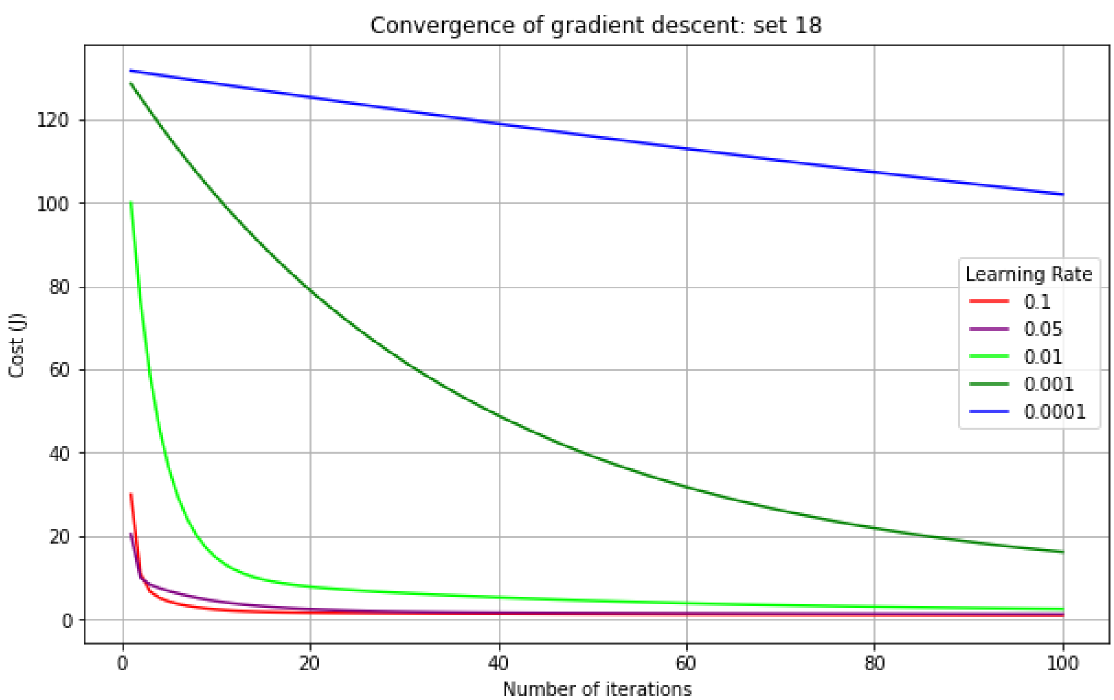

Convergence of gradient descent: set 13
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
10
20
30
40
50
0
100
200
300
400
500

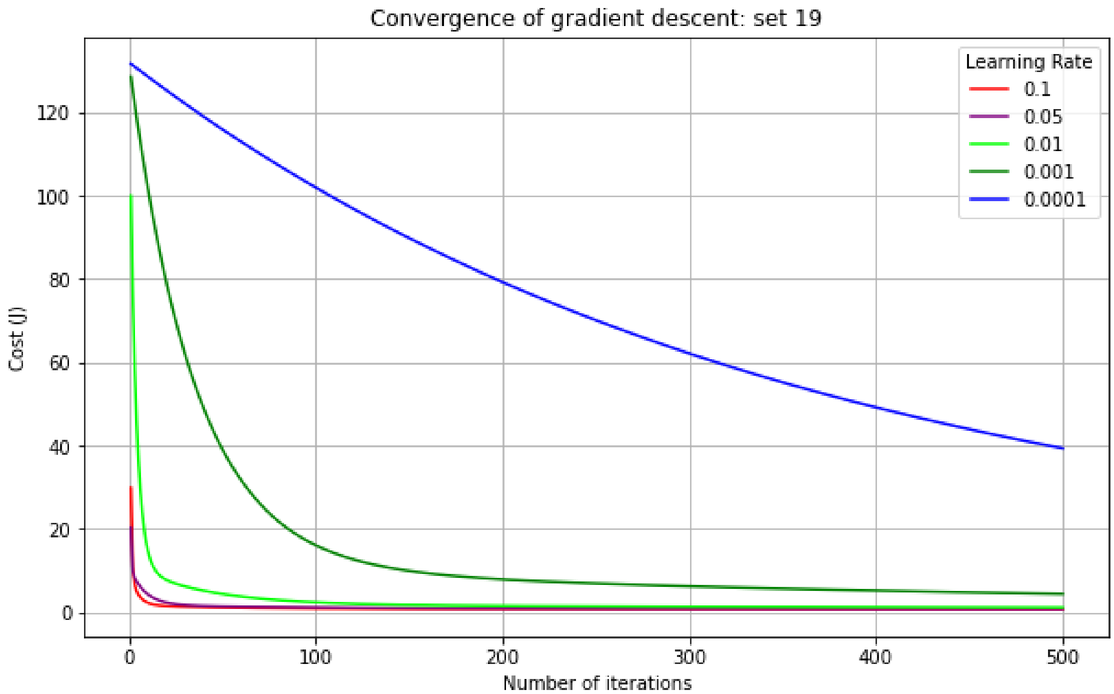

Convergence of gradient descent: set 14
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
10
20
30
40
50
0
200
400
600
800
1000

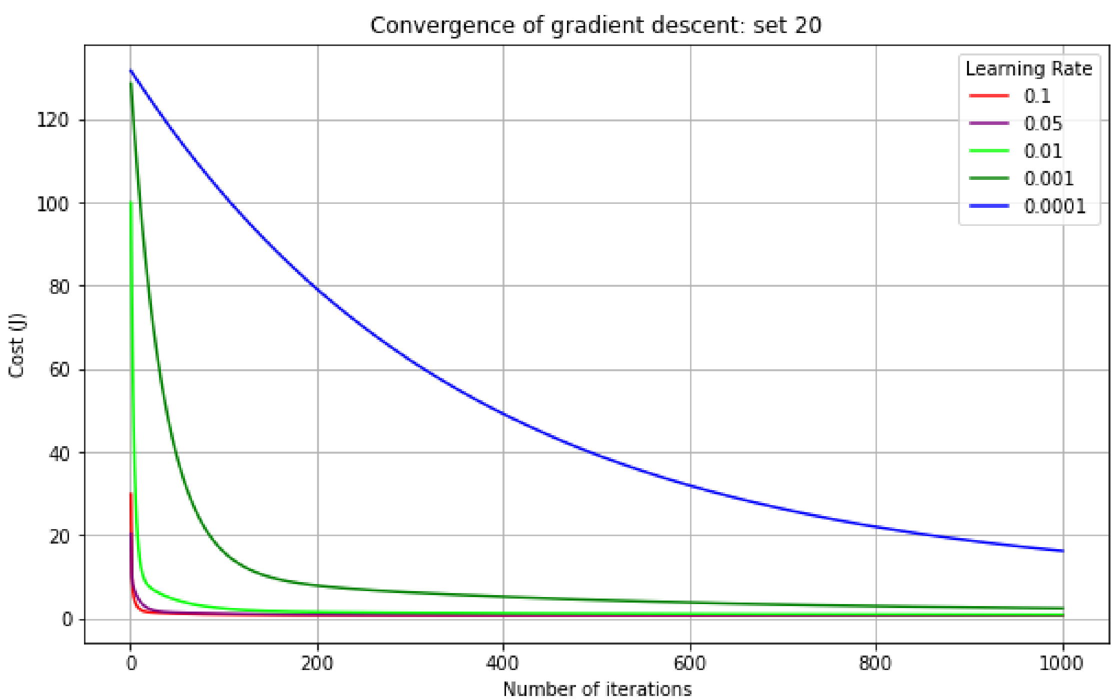

Convergence of gradient descent: set 15
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
10
20
30
40
50
0
250
500
750
1000
1250
1500
1750
2000

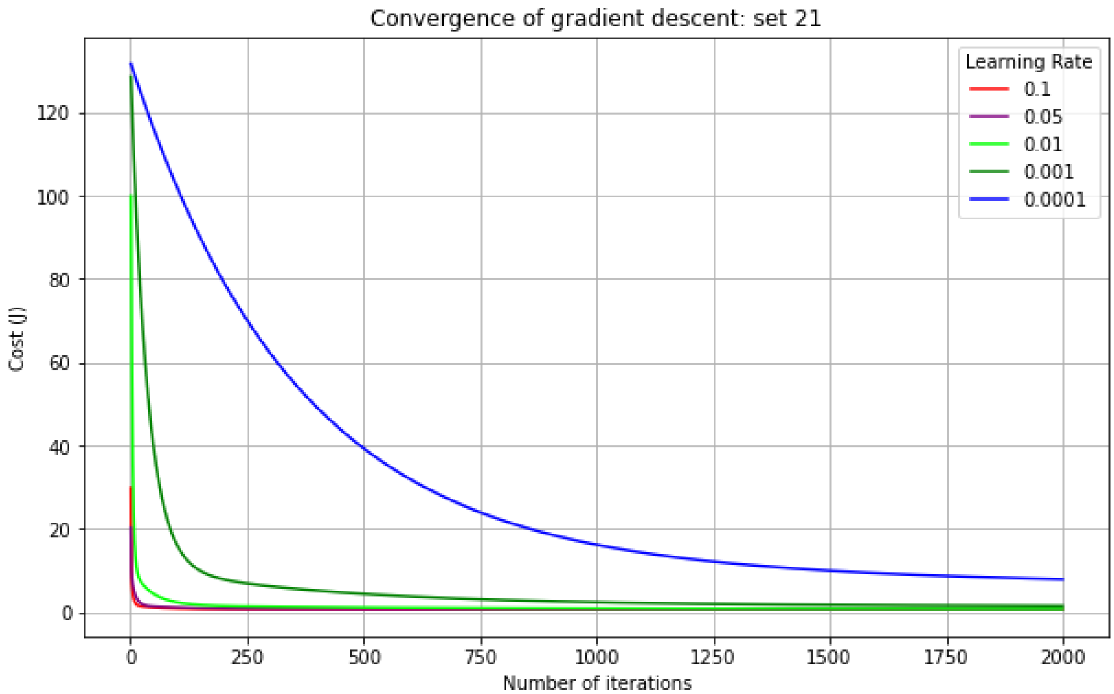

Convergence of gradient descent: set 16
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
10
20
30
40
50
0
1000
2000
3000
4000
5000

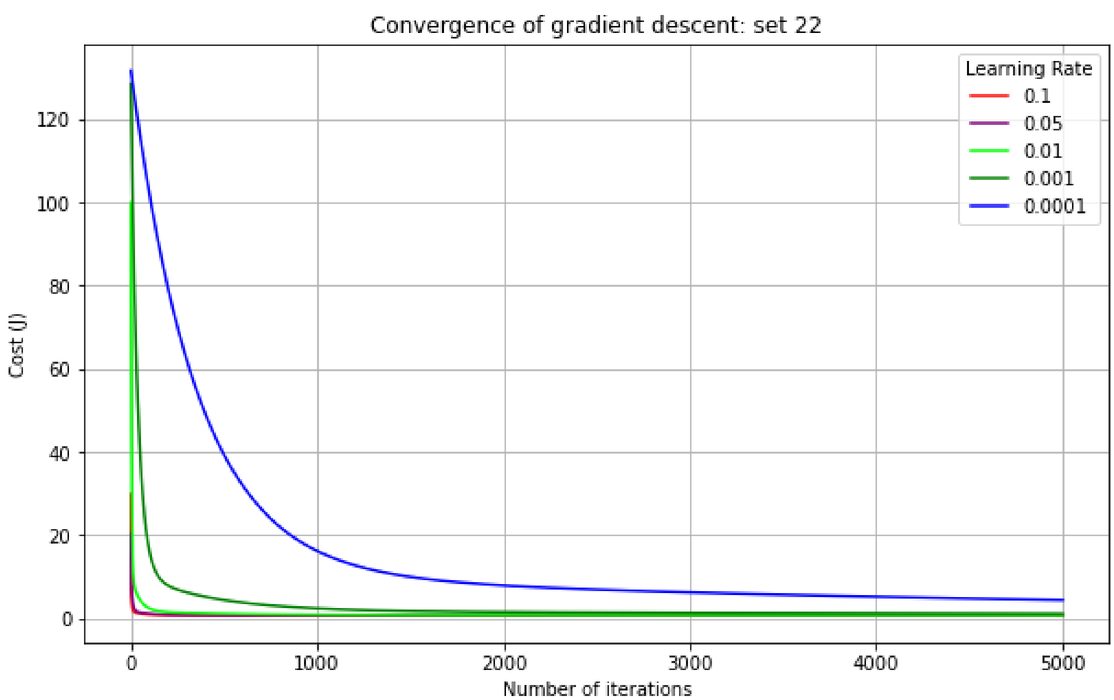

Convergence of gradient descent: set 17
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
10
20
30
40
50
0
2000
4000
6000
8000
10000

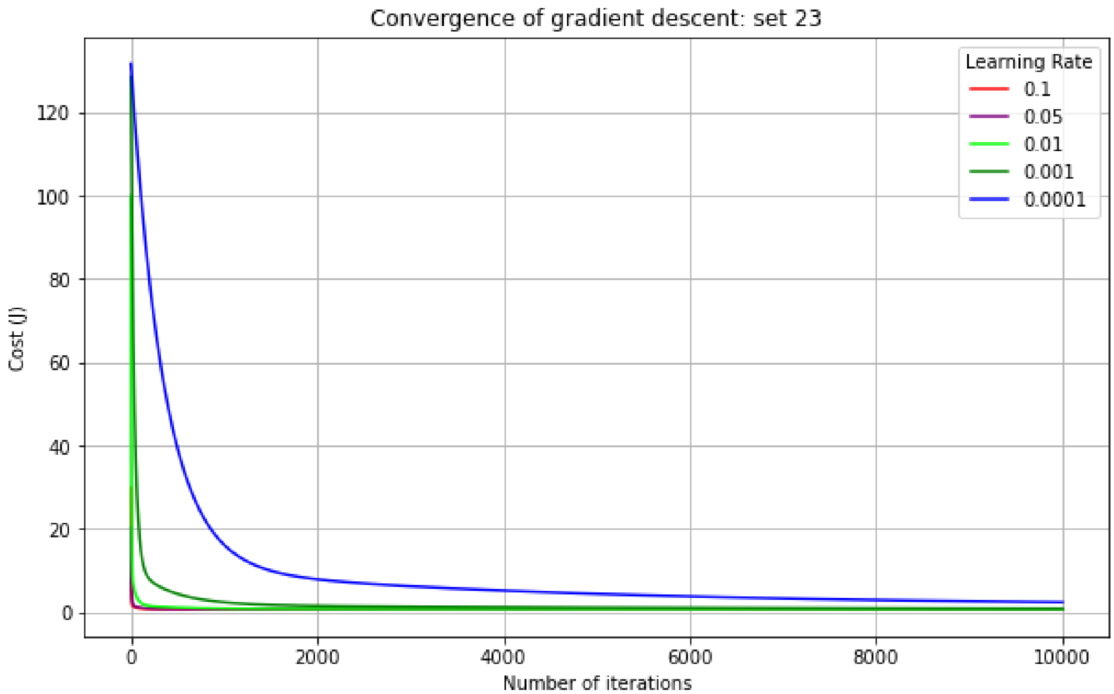

Convergence of gradient descent: set 18
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
20
40
60
80
100
120
0
20
40
60
80
100

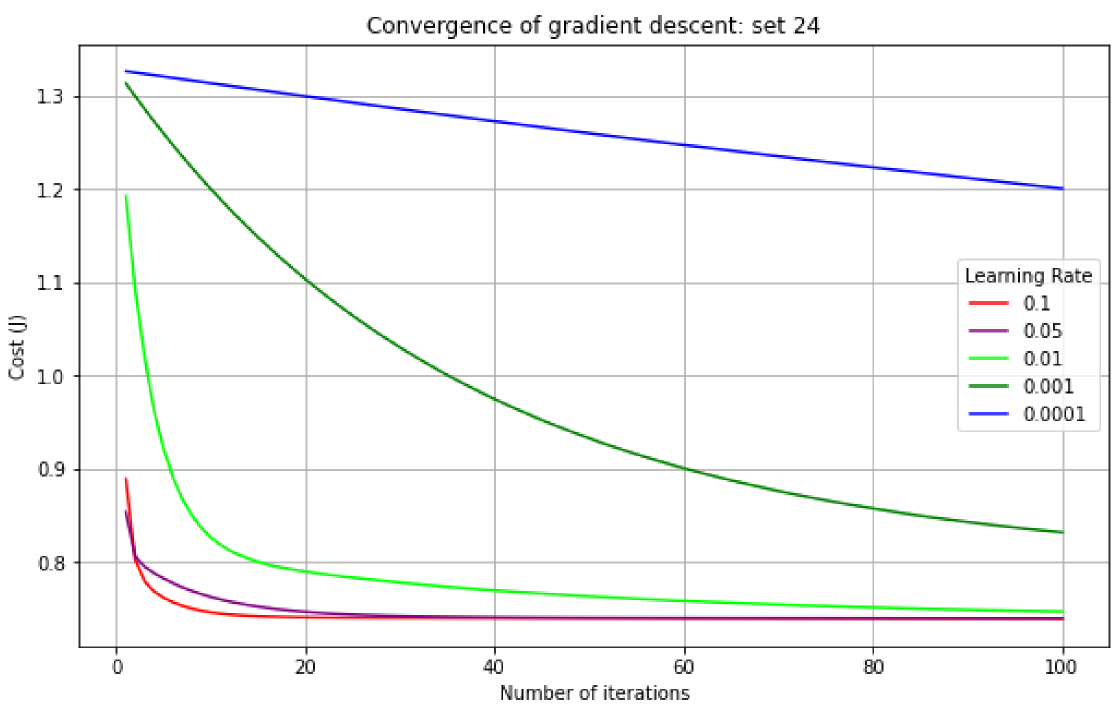

Convergence of gradient descent: set 19
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
120
100
80
60
40
20
0
0
100
200
300
400
500
Number of iterations

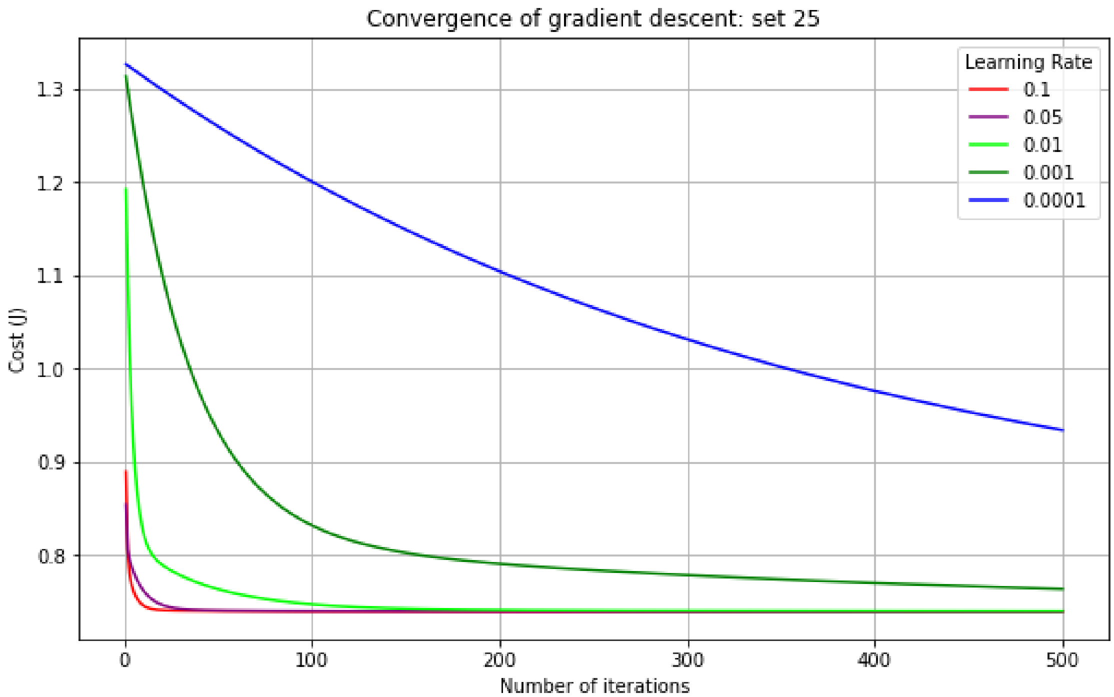

Convergence of gradient descent: set 20
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
120
100
80
60
40
20
0
0
200
400
600
800
1000
Number of iterations

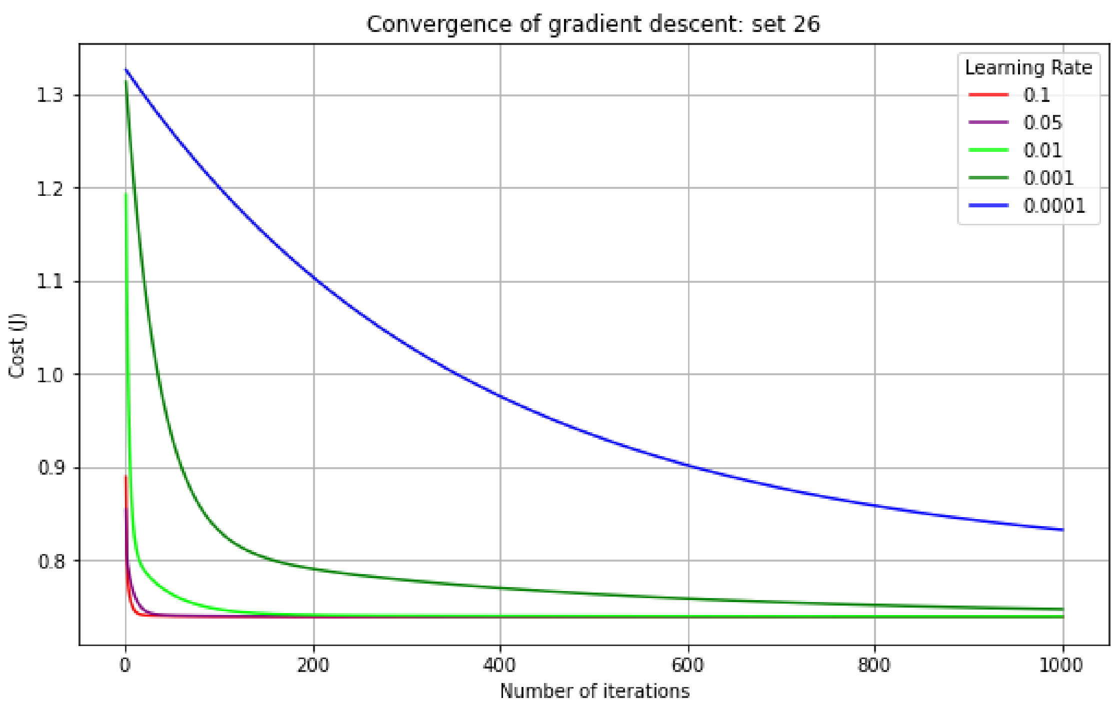

Convergence of gradient descent: set 21
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
0
20
40
60
80
100
120
0
250
500
750
1000
1250
1500
1750
2000
Number of iterations

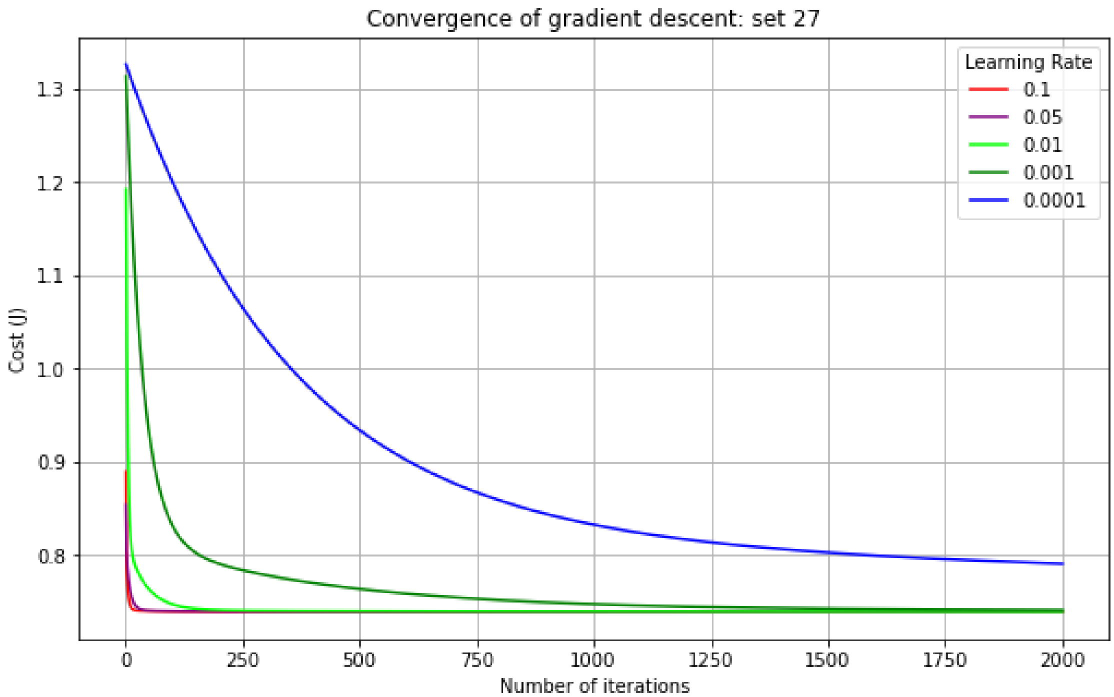

Convergence of gradient descent: set 22
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
0
20
40
60
80
100
120
0
1000
2000
3000
4000
5000
Number of iterations

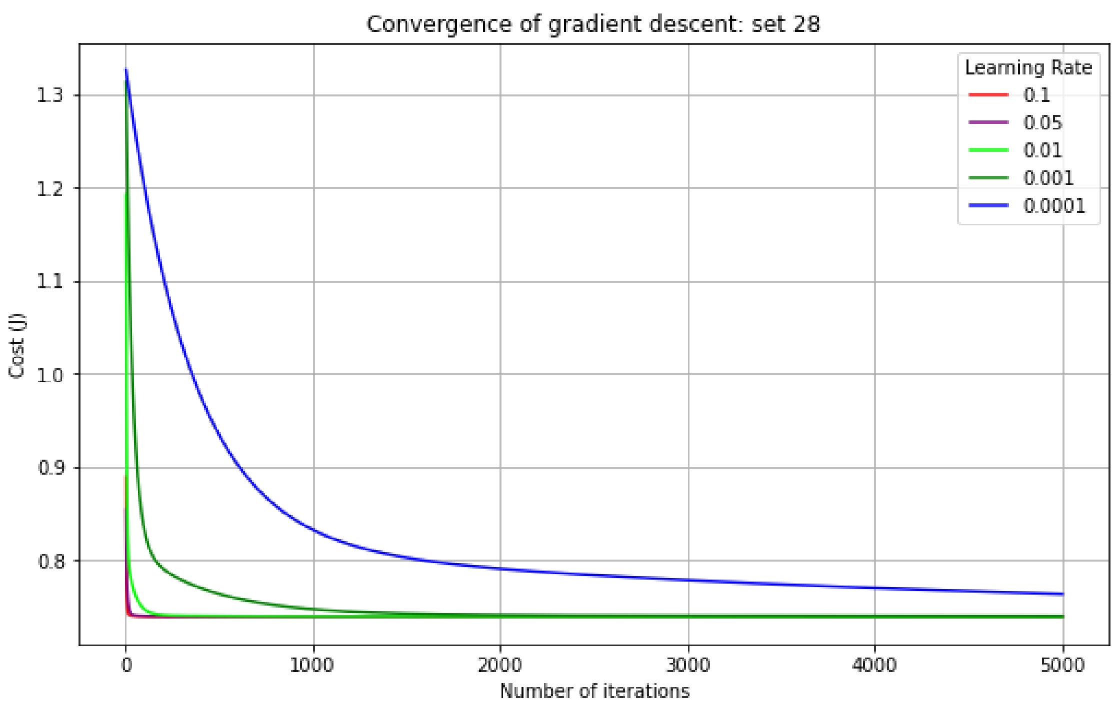

Convergence of gradient descent: set 23
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0
20
40
60
80
100
120
0
2000
4000
6000
8000
10000

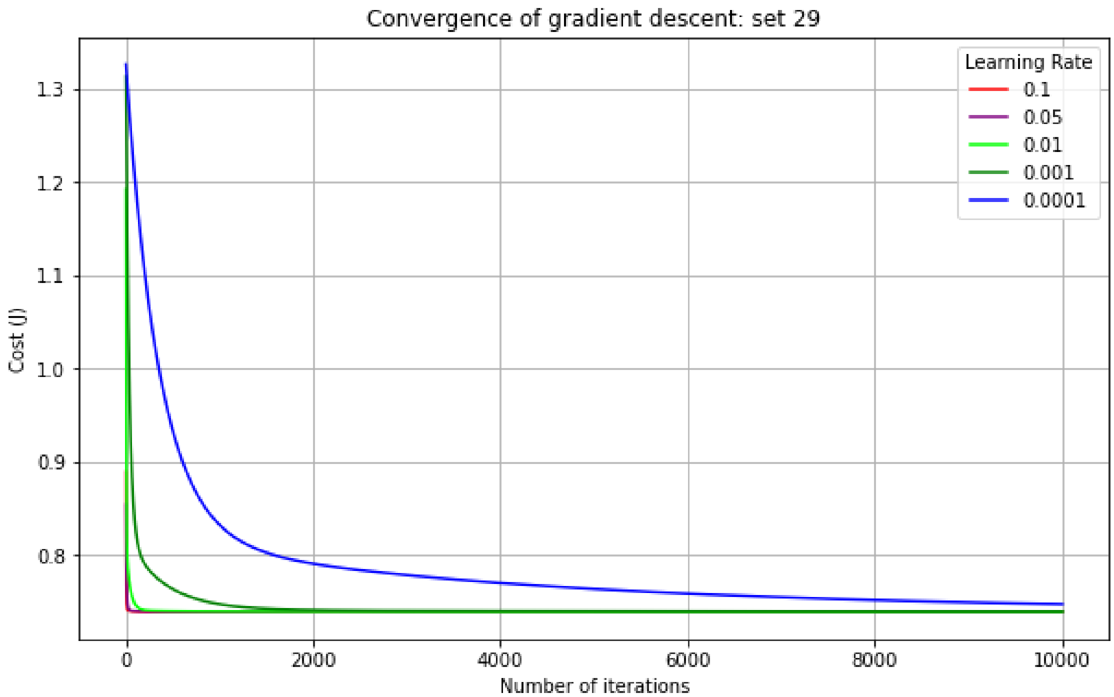

Convergence of gradient descent: set 24
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
Number of iterations
0.8
0.9
1.0
1.1
1.2
1.3
0
20
40
60
80
100


Convergence of gradient descent: set 25
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
1.3
1.2
1.1
1.0
0.9
0.8
0
100
200
300
400
500
Number of iterations


Convergence of gradient descent: set 26
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
1.3
1.2
1.1
1.0
0.9
0.8
0
200
400
600
800
1000
Number of iterations


Convergence of gradient descent: set 27
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
1.3
1.2
1.1
1.0
0.9
0.8
0
250
500
750
1000
1250
1500
1750
2000
Number of iterations


Convergence of gradient descent: set 28
Learning Rate
0.1
0.05
0.01
0.001
0.0001
Cost (J)
1.3
1.2
1.1
1.0
0.9
0.8
0
1000
2000
3000
4000
5000
Number of iterations




[20]:
```
print(f"From set {best_set}\n\tbest_cost = {best_cost}")
print(f"\tbest_theta = {best_theta}")
print(f"\tbest_learning_rate = {best_learning_rate}")

print("\nTheres a high probability its overfitted but thats not the point of␣
↪the assignment")
```

```
From set 3
	best_cost = 0.738464241568294
	best_theta = [ 5.31416716 -2.00371927  0.53256334 -0.26560186]
	best_learning_rate = 0.1

Theres a high probability its overfitted but thats not the point of the
assignment
```

[21]:
```
explanatory_vars = [1, 1, 1] # input question
# puts the one in front of the matrix necesary for theta 0
in_mat = np.matrix([1, *explanatory_vars])
# outputs nicely
print(f"Y({str(explanatory_vars)}) = {float(in_mat.dot(best_theta))}")


explanatory_vars = [2, 0, 4]
in_mat = np.matrix([1, *explanatory_vars])
```

```
print(f"Y({str(explanatory_vars)}) = {float(in_mat.dot(best_theta))}")


explanatory_vars = [3, 2, 1]
in_mat = np.matrix([1, *explanatory_vars])
print(f"Y({str(explanatory_vars)}) = {float(in_mat.dot(best_theta))}")
```

```
Y([1, 1, 1]) = 3.577409368656757
Y([2, 0, 4]) = 0.2443211714832545
Y([3, 2, 1]) = 0.10253417186972857
```

[ ]:

```

```