# Gradient descent and linear regression

. karn arora(j078)

## importing libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

## importing the datasets

```
dataset = pd.read_csv('ex1data2.txt')
```

```
dataset.columns = ["size","rooms","price"]
```

```
x = dataset.iloc[:,:-1].values
y = dataset.iloc[:,-1].values
```

```
x
```

```
array([[1600,    3],
       [2400,    3],
       [1416,    2],
       [3000,    4],
       [1985,    4],
       [1534,    3],
       [1427,    3],
       [1380,    3],
       [1494,    3],
       [1940,    4],
       [2000,    3],
       [1890,    3],
       [4478,    5],
       [1268,    3],
       [2300,    4],
       [1320,    2],
       [1236,    3],
       [2609,    4],
       [3031,    4],
       [1767,    3],
       [1888,    2],
       [1604,    3],
       [1962,    4],
       [3890,    3],
       [1100,    3],
```

```
       [1458,    3],
       [2526,    3],
       [2200,    3],
       [2637,    3],
       [1839,    2],
       [1000,    1],
       [2040,    4],
       [3137,    3],
       [1811,    4],
       [1437,    3],
       [1239,    3],
       [2132,    4],
       [4215,    4],
       [2162,    4],
       [1664,    2],
       [2238,    3],
       [2567,    4],
       [1200,    3],
       [ 852,    2],
       [1852,    4],
       [1203,    3]])
```

```
dataset
```

| | size | rooms | price |
|---|---|---|---|
| **0** | 1600 | 3 | 329900 |
| **1** | 2400 | 3 | 369000 |
| **2** | 1416 | 2 | 232000 |
| **3** | 3000 | 4 | 539900 |
| **4** | 1985 | 4 | 299900 |
| **5** | 1534 | 3 | 314900 |
| **6** | 1427 | 3 | 198999 |
| **7** | 1380 | 3 | 212000 |
| **8** | 1494 | 3 | 242500 |
| **9** | 1940 | 4 | 239999 |
| **10** | 2000 | 3 | 347000 |
| **11** | 1890 | 3 | 329999 |
| **12** | 4478 | 5 | 699900 |
| **13** | 1268 | 3 | 259900 |
| **14** | 2300 | 4 | 449900 |
| **15** | 1320 | 2 | 299900 |
| **16** | 1236 | 3 | 199900 |
| **17** | 2609 | 4 | 499998 |
| **18** | 3031 | 4 | 599000 |
| **19** | 1767 | 3 | 252900 |
| **20** | 1888 | 2 | 255000 |
| **21** | 1604 | 3 | 242900 |
| **22** | 1962 | 4 | 259900 |
| **23** | 3890 | 3 | 573900 |
| **24** | 1100 | 3 | 249900 |
| **25** | 1458 | 3 | 464500 |
| **26** | 2526 | 3 | 469000 |

```
y
```

```
array([329900, 369000, 232000, 539900, 299900, 314900, 198999, 212000,
       242500, 239999, 347000, 329999, 699900, 259900, 449900, 299900,
       199900, 499998, 599000, 252900, 255000, 242900, 259900, 573900,
       249900, 464500, 469000, 475000, 299900, 349900, 169900, 314900,
       579900, 285900, 249900, 229900, 345000, 549000, 287000, 368500,
       329900, 314000, 299000, 179900, 299900, 239500])
```

32 3137 3 579900

# training the model

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x,y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

# predicting using our model

41 2567 4 314000

```
y_pred = regressor.predict(x)
```

43 852 2 179900

```
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y.reshape(len(y),1)),1))
```

```
[[285258.01 329900.  ]
 [396262.71 369000.  ]
 [267913.31 232000.  ]
 [471329.85 539900.  ]
 [330492.64 299900.  ]
 [276100.12 314900.  ]
 [261253.24 198999.  ]
 [254731.71 212000.  ]
 [270549.88 242500.  ]
 [324248.62 239999.  ]
 [340760.36 347000.  ]
 [325497.21 329999.  ]
 [668224.66 699900.  ]
 [239191.06 259900.  ]
 [374200.74 449900.  ]
 [254592.74 299900.  ]
 [234750.87 199900.  ]
 [417076.31 499998.  ]
 [475631.29 599000.  ]
 [308430.24 252900.  ]
 [333406.08 255000.  ]
 [285813.03 242900.  ]
 [327301.25 259900.  ]
 [603008.97 573900.  ]
 [215880.07 249900.  ]
 [265554.67 464500.  ]
 [413745.95 469000.  ]
 [368511.53 475000.  ]
 [429147.85 299900.  ]
 [326607.04 349900.  ]
 [218377.25 169900.  ]
 [338124.21 314900.  ]
 [498525.79 579900.  ]
 [306349.11 285900.  ]
 [262640.8  249900.  ]
 [235167.13 229900.  ]
 [350889.75 345000.  ]
```

```
 [639918.25 549000.  ]
 [355052.43 287000.  ]
 [302324.77 368500.  ]
 [373784.26 329900.  ]
 [411248.56 314000.  ]
 [229755.66 299000.  ]
 [189654.99 179900.  ]
 [312038.11 299900.  ]
 [230171.92 239500.  ]]
```

# Gradient descent

```
x.shape
```

```
(46, 2)
```

```
y.shape
```

```
(46,)
```

```
y = y.reshape(y.shape[0],1)
```

```
y.shape
```

```
(46, 1)
```

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
sc_y = StandardScaler()
x = sc_x.fit_transform(x)
y = sc_y.fit_transform(y)
```

addings 1s into the matrix

```
x = np.c_[np.ones(x.shape[0]),x]
```

```
x
```

```
array([[ 1.00e+00, -5.01e-01, -2.29e-01],
       [ 1.00e+00,  5.05e-01, -2.29e-01],
       [ 1.00e+00, -7.33e-01, -1.54e+00],
       [ 1.00e+00,  1.26e+00,  1.09e+00],
       [ 1.00e+00, -1.69e-02,  1.09e+00],
       [ 1.00e+00, -5.85e-01, -2.29e-01],
       [ 1.00e+00, -7.19e-01, -2.29e-01],
       [ 1.00e+00, -7.78e-01, -2.29e-01],
       [ 1.00e+00, -6.35e-01, -2.29e-01],
       [ 1.00e+00, -7.35e-02,  1.09e+00],
       [ 1.00e+00,  1.97e-03, -2.29e-01],
```

```
       [ 1.00e+00, -1.36e-01, -2.29e-01],
       [ 1.00e+00,  3.12e+00,  2.40e+00],
       [ 1.00e+00, -9.19e-01, -2.29e-01],
       [ 1.00e+00,  3.80e-01,  1.09e+00],
       [ 1.00e+00, -8.54e-01, -1.54e+00],
       [ 1.00e+00, -9.60e-01, -2.29e-01],
       [ 1.00e+00,  7.68e-01,  1.09e+00],
       [ 1.00e+00,  1.30e+00,  1.09e+00],
       [ 1.00e+00, -2.91e-01, -2.29e-01],
       [ 1.00e+00, -1.39e-01, -1.54e+00],
       [ 1.00e+00, -4.96e-01, -2.29e-01],
       [ 1.00e+00, -4.59e-02,  1.09e+00],
       [ 1.00e+00,  2.38e+00, -2.29e-01],
       [ 1.00e+00, -1.13e+00, -2.29e-01],
       [ 1.00e+00, -6.80e-01, -2.29e-01],
       [ 1.00e+00,  6.64e-01, -2.29e-01],
       [ 1.00e+00,  2.54e-01, -2.29e-01],
       [ 1.00e+00,  8.04e-01, -2.29e-01],
       [ 1.00e+00, -2.01e-01, -1.54e+00],
       [ 1.00e+00, -1.26e+00, -2.86e+00],
       [ 1.00e+00,  5.23e-02,  1.09e+00],
       [ 1.00e+00,  1.43e+00, -2.29e-01],
       [ 1.00e+00, -2.36e-01,  1.09e+00],
       [ 1.00e+00, -7.07e-01, -2.29e-01],
       [ 1.00e+00, -9.56e-01, -2.29e-01],
       [ 1.00e+00,  1.68e-01,  1.09e+00],
       [ 1.00e+00,  2.79e+00,  1.09e+00],
       [ 1.00e+00,  2.06e-01,  1.09e+00],
       [ 1.00e+00, -4.21e-01, -1.54e+00],
       [ 1.00e+00,  3.02e-01, -2.29e-01],
       [ 1.00e+00,  7.16e-01,  1.09e+00],
       [ 1.00e+00, -1.00e+00, -2.29e-01],
       [ 1.00e+00, -1.44e+00, -1.54e+00],
       [ 1.00e+00, -1.84e-01,  1.09e+00],
       [ 1.00e+00, -1.00e+00, -2.29e-01]])
```

## initialising theta matrix

```
np.random.seed(123)
theta = np.array([[0],[0],[0]])
```

```
theta
```

```
array([[0],
       [0],
       [0]])
```

## forward propogation

```
m = y.size
```

```
m
```

```
46
```

```
h_theta = np.dot(x,theta)
```

```
h_theta
```

```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

```
error = h_theta-y
```

```
error = h_theta-y
```

```
error
```

```
array([[ 0.07],
       [-0.24],
       [ 0.86],
       [-1.61],
       [ 0.31],
       [ 0.19],
       [ 1.12],
       [ 1.02],
       [ 0.77],
       [ 0.79],
       [-0.06],
       [ 0.07],
       [-2.89],
       [ 0.64],
       [-0.89],
       [ 0.31],
       [ 1.12],
       [-1.29],
       [-2.08],
       [ 0.69],
       [ 0.67],
       [ 0.77],
       [ 0.64],
       [-1.88],
       [ 0.72],
       [-1.01],
       [-1.04],
       [-1.09],
       [ 0.31],
       [-0.09],
       [ 1.36],
       [ 0.19],
       [-1.93],
       [ 0.43],
       [ 0.72],
       [ 0.88],
       [-0.05],
       [-1.68],
       [ 0.42],
       [-0.24],
       [ 0.07],
       [ 0.2 ],
       [ 0.32],
       [ 1.28],
       [ 0.31],
       [ 0.8 ]])
```

```
cost = (1/2)*(1/m)*np.dot(error.T,error)
```

```
cost
```

```
array([[0.5]])
```

## backward propogation

```
alpha = 0.01
```

```
theta = theta -(alpha*(1/m)*np.dot(x.T,error))
```

```
theta
```

```
array([[-4.34e-19],
       [ 8.56e-03],
       [ 4.46e-03]])
```

```
new_h_theta = np.dot(x,theta)
```

```
new_error = new_h_theta - y
```

```
new_cost = (1/2)*(1/m)*np.dot(new_error.T,new_error)
```

```
new_cost
```

```
array([[0.49]])
```

## backwards propogation with epoch

```
epoch = 10000
```

```
def GD(x,y,epoch,alpha,theta):
  past_cost = []
  past_theta = [theta]


  for i in range(epoch):
    h_theta = np.dot(x,theta)
    error = h_theta - y
    cost = (1/2)*(1/m)*np.dot(error.T,error)
    past_cost.append(cost)
    theta = theta - alpha*(1/m)*np.dot(x.T,error)
    past_theta.append(theta)


    try:
      if past_cost[-1]==past_cost[-2]:
        break
      else:
        continue
    except:
      pass
  return past_cost,past_theta
```

```
past_cost,past_theta = GD(x,y,epoch,alpha,theta)
```

```
past_cost[-1]
```

```
array([[0.13]])
```

```
c = np.asarray((past_cost))
```

```
c.shape[0]
```

```
3414
```

```
c = c.reshape(c.size,1)
```

```
c.shape
```

```
(3414, 1)
```

```
plt.plot(c)
plt.xlabel("epochs")
plt.ylabel("cost")
plt.title("cost vs epochs")
plt.show()
```