Linear Regression Univariate

April 25, 2019

```
In [1]: import numpy as np

In [2]: import matplotlib.pyplot as plt

In [4]: data=np.genfromtxt("ex1data1.txt",delimiter=",")

In [116]: X=data[:,0]

In [6]: y=data[:,1]

In [7]: X

Out[7]: array([ 6.1101,  5.5277,  8.5186,  7.0032,  5.8598,  8.3829,  7.4764,
                8.5781,  6.4862,  5.0546,  5.7107, 14.164 ,  5.734 ,  8.4084,
                5.6407,  5.3794,  6.3654,  5.1301,  6.4296,  7.0708,  6.1891,
               20.27  ,  5.4901,  6.3261,  5.5649, 18.945 , 12.828 , 10.957 ,
               13.176 , 22.203 ,  5.2524,  6.5894,  9.2482,  5.8918,  8.2111,
                7.9334,  8.0959,  5.6063, 12.836 ,  6.3534,  5.4069,  6.8825,
               11.708 ,  5.7737,  7.8247,  7.0931,  5.0702,  5.8014, 11.7   ,
                5.5416,  7.5402,  5.3077,  7.4239,  7.6031,  6.3328,  6.3589,
                6.2742,  5.6397,  9.3102,  9.4536,  8.8254,  5.1793, 21.279 ,
               14.908 , 18.959 ,  7.2182,  8.2951, 10.236 ,  5.4994, 20.341 ,
               10.136 ,  7.3345,  6.0062,  7.2259,  5.0269,  6.5479,  7.5386,
                5.0365, 10.274 ,  5.1077,  5.7292,  5.1884,  6.3557,  9.7687,
                6.5159,  8.5172,  9.1802,  6.002 ,  5.5204,  5.0594,  5.7077,
                7.6366,  5.8707,  5.3054,  8.2934, 13.394 ,  5.4369])

In [8]: #import os
        #data = np.loadtxt(os.path.join( 'ex1data1.txt'), delimiter=',')
        #X, y = data[:, 0], data[:, 1]

In [181]: theta=np.zeros([2,1])#dtype=int

In [182]: theta

Out[182]: array([[0.],
                 [0.]])

In [183]: m=len(y)
In [184]: X=X.reshape(m,1)
          y=y.reshape(m,1)
```

```
In [185]: def computeCost(X,y,theta,m):
              ons=np.ones(([m,1]),dtype=int)
              #print(ons)
              x=np.hstack((ons,X))
              #print(x)
              prediction=np.matmul(x,theta)
              #print (prediction)
              #print(prediction.shape)
              #print((prediction-y).shape)
              sqerror=np.power((prediction-y),2)
              print(sqerror.shape)
              print(sqerror.sum)
              J=(1/(2*m))*(sqerror.sum())
              return J
```

```
In [186]: computeCost(X,y,theta,m)
```

```
(97, 1)
<built-in method sum of numpy.ndarray object at 0x0000026D5450A1C0>
```

```
Out[186]: 32.072733877455676
```

```
In [187]: alpha=0.01
          iterations=1500;
```

```
In [188]: def gradientDescent(X,y,theta,alpha,iterations,m):
              ons=np.ones([m,1])
              x=np.hstack((ons,X))
              #prediction=np.matmul(x,theta)
              #print(prediction)
              for i in range(iterations):
                  t1=theta[0][0]-(1/m)*alpha*((np.matmul(x,theta)-y).sum())
                  t2=theta[1][0]-(1/m)*alpha*(((np.matmul(x,theta)-y)*X).sum())
                  theta[0][0]=t1
                  theta[1][0]=t2
                 # print(theta)
              return theta
```

```
In [206]: ons=np.ones([m,1])
          x=np.hstack((ons,X))
```

```
In [190]: theta=gradientDescent(X,y,theta,alpha,iterations,m)
```

```
In [191]: theta
```

```
Out[191]: array([[-3.63029144],
                 [ 1.16636235]])
```

```
In [192]: ons=np.ones([m,1])
          x=np.hstack([ons,X])
```

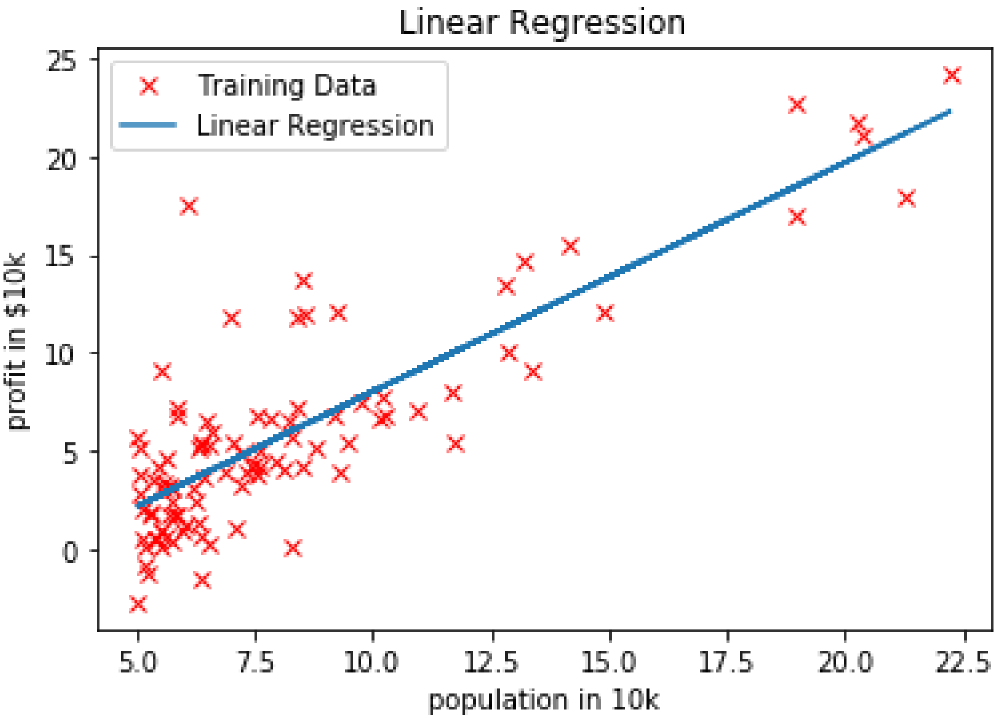
```
In [193]: hold on
```

```
          File "<ipython-input-193-31699b0708a0>", line 1
        hold on
              ^
    SyntaxError: invalid syntax
```

```
In [ ]:
```

```
In [201]: plt.plot(X,y,'rx',X,np.matmul(x,theta),'-')
          plt.legend(['Training Data','Linear Regression'])
          plt.xlabel('population in 10k')
          plt.ylabel('profit in $10k')
          plt.title('Linear Regression')
```

```
Out[201]: Text(0.5, 1.0, 'Linear Regression')
```



```
In [195]: plt.show()
In [196]: #for 35 thousand

          pred=np.dot([1,3.5],theta)

In [197]: pred

Out[197]: array([0.45197679])

In [198]: pred*10000

Out[198]: array([4519.7678677])

In [199]: print(pred*10000)

[4519.7678677]
```

0.1 Using Normal Equations

```
In [202]: theta=np.zeros([2,1])

In [203]: #theta=inv(X'*X)X'y
          ons=np.ones([m,1])
          x=np.hstack([ons,X])

In [204]: theta=np.dot(np.linalg.inv(np.dot(x.T,x)),np.dot(x.T,y))

In [205]: theta

Out[205]: array([[-3.89578088],
                 [ 1.19303364]])

In [ ]:
```