Linear Regression Multivariate

April 25, 2019

```
In [120]: import numpy as np

In [121]: import matplotlib.pyplot as plt

In [122]: data=np.genfromtxt("ex1data2.txt",delimiter=",")

In [123]: X=data[:,0:2];
          X.reshape(47,2);

In [124]: y=data[:,2]

In [125]: mu=np.zeros([1,2])

In [126]: theta=np.zeros([3,1])

In [127]: m=len(y)

In [128]: np.shape(X)
          y=y.reshape(m,1)

In [129]: def featureNormalize(X):
              mu=[np.mean(X[:,0]),np.mean(X[:,1])]
              sigma=[np.std(X[:,0]),np.std(X[:,1])]
              X=(X-mu)/sigma
              return X,mu,sigma

In [130]: [X,mu,sigma]=featureNormalize(X)
          mu

Out[130]: [2000.6808510638298, 3.1702127659574466]

In [131]: def computeCost(X,y,theta,m):
              ons=np.ones(([m,1]),dtype=int)
              x=np.hstack((ons,X))
              pred=(np.matmul(x,theta)-y)
              sqerror=np.power((pred),2)
              J=(1/(2*m))*(sqerror.sum())
              return J
```

In [ ]:

In [132]:
```
J=computeCost(X,y,theta,m)
J
```

Out[132]:
```
65591548106.457443
```

Difference between sqerror.sum() and sum(sqerror)

In [133]:
```
alpha=0.01
iterations=400
```

In [134]:
```
def gradientDescent(X,y,theta,m,alpha,iterations):
    x=np.hstack((np.ones([m,1]),X))
    a=(x[:,1]).reshape(m,1)
    b=(x[:,2]).reshape(m,1)

    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)*a).sum())
        t3=theta[2][0]-(1/m)*alpha*(((np.matmul(x,theta)-y)*b).sum())
        theta[0][0]=t1
        theta[1][0]=t2
        theta[2][0]=t3
    return theta
```

In [ ]:

In [88]:
```
x=np.hstack((np.ones([m,1]),X))
a=(x[:,1]).reshape(m,1)
t2=theta[1][0]-(1/m)*alpha*(((np.matmul(x,theta)-y)*a).sum())
t2
```

Out[88]:
```
1057.6413349281561
```

In [135]:
```
theta=gradientDescent(X,y,theta,m,alpha,iterations)
```

In [136]:
```
theta
```

Out[136]:
```
array([[ 334302.06399328],
       [  99411.44947359],
       [   3267.01285407]])
```

In [137]:
```
mu[1]
```

Out[137]:
```
3.1702127659574466
```

```
In [138]: z=[1,3031,4]
          mu

          z[1]=(z[1]-mu[0])/sigma[0]
          z[2]=(z[2]-mu[1])/sigma[1]
          z

Out[138]: [1, 1.3105007848783414, 1.1022051669412321]

In [119]: mu

Out[119]: [9.4487065925545235e-18, 2.7105977037390789e-16]

In [139]: price=np.matmul(z,theta)

In [140]: price

Out[140]: array([ 468181.76500252])

In [ ]:
```