EE 6973: MACHINE LEARNING WITH BIG DATA ANALYTICS

ASSIGNMNET 1

By

MITHA ANN PHILIP

Jrb468

1. **Closed Form:**

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


X = []
Y = []
#Load dataset
for line in open('Assign1_input.txt'):
   x, y = line.split(',')
   X.append(float(x))
   Y.append(float(y))


print "Input X:",X, "and Y: ",Y
>>Input X: [0.0, 1.0, 2.0, 3.0, 4.0] and Y:  [1.0, 3.0, 7.0, 13.0, 21.0
]

#Plot for input data
X = np.array(X)
Y = np.array(Y)
fig, plot = plt.subplots(figsize=(15,8))
plot.scatter(X,Y)
plot.set_xlabel('X')
plot.set_ylabel('Y')
plot.set_title('Input Data')
plt.show()
```

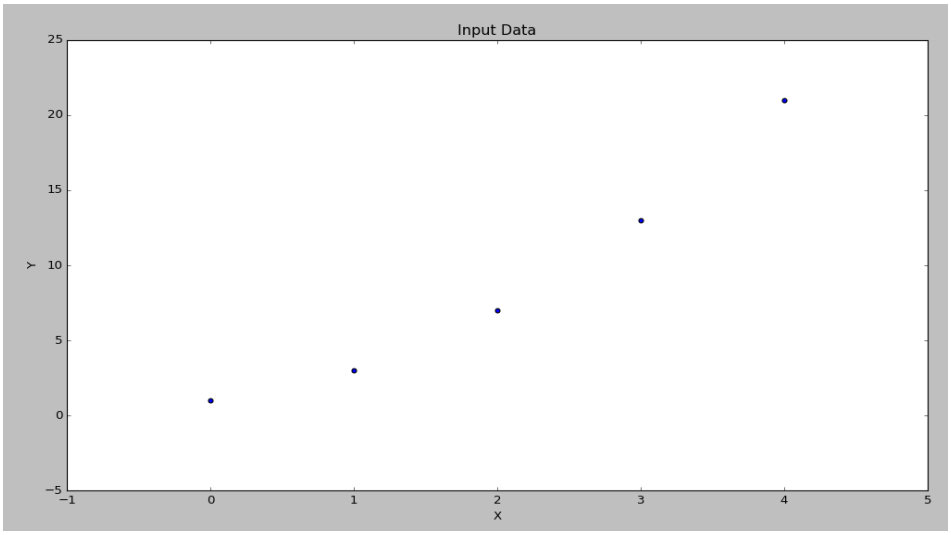

Fig 1: Plot with Input data

#Computing Regression

denominator = X.dot(X) - X.mean() * X.sum()

a = (X.dot(Y) - Y.mean()*X.sum()) / denominator

b = (Y.mean() * X.dot(X) - X.mean() * X.dot(Y)) / denominator

Yhat = a*X + b

print "Slope: ",a,"and Intercept is: ",b

```
>>Slope:  5.0 and Intercept is:  -1.0
```

plt.scatter(X, Y)

plt.plot(X, Yhat)

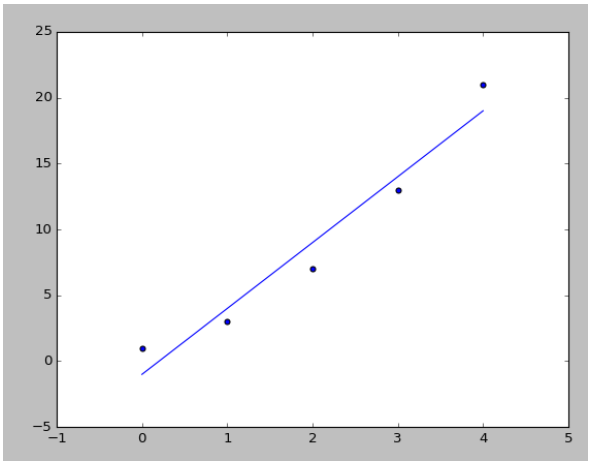

Fig 2: Plot X vs Yhat, with Best Fitting line

1) Gradient Descent

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
get_ipython().magic(u'matplotlib inline')

#os.getcwd represents a string with current working directory
path = os.getcwd() + '/' + 'Assign1_input.txt'
data = pd.read_csv(path, header=None, names=['X', 'Y'])  # directs table data values to data
data
```

	X	Y
0	0	1
1	1	3
2	2	7
3	3	13
4	4	21

```
def computeCost(X, y, theta):
    inner = np.power(((X * theta.T) - y), 2)
    return np.sum(inner) / (2 * len(X))
```

```
# In[15]:


# append a ones column to the front of the data set
data.insert(0, 'Ones', 1)


# set X (training data) and y (target variable)
cols = data.shape[1]
X = data.iloc[:,0:cols-1]
y = data.iloc[:,cols-1:cols]


# convert from data frames to numpy matrices
X = np.matrix(X.values)
y = np.matrix(y.values)
theta = np.matrix(np.array([0,0]))


def gradientDescent(X, y, theta, alpha, iters):
    temp = np.matrix(np.zeros(theta.shape))
    parameters = int(theta.ravel().shape[1])
    cost = np.zeros(iters)
    Wtheta = np.zeros(shape=(iters,2))


    for i in range(iters):
        error = (X * theta.T) - y
        #if ((iters%1000)==0):
            #alpha = alpha/10


        for j in range(parameters):
            term = np.multiply(error, X[:,j])
            temp[0,j] = theta[0,j] - ((alpha / len(X)) * np.sum(term))


        theta = temp
```

```
        cost[i] = computeCost(X, y, theta)
        Wtheta [i,0] = theta[0,0]
        Wtheta [i,1] = theta[0,1]
    return Wtheta, cost
# initialize variables for learning rate and iterations
alpha = 0.01
iters = 10000
# perform gradient descent to "fit" the model parameters
W, cost = gradientDescent(X, y, theta, alpha, iters)
print W
[[ 0.09      0.28    ]
 [ 0.1735    0.5414  ]
 [ 0.250937  0.785446]
 ...,
 [-1.        5.      ]
 [-1.        5.      ]
 [-1.        5.      ]]
print cost
[ 58.53965     51.2615588   44.92553487 ...,   1.4          1.4
1.4       ]

x = np.linspace(data.X.min(), data.X.max(), 10)
fig, ax = plt.subplots(figsize=(12,8))
for i in range(iters):
    f = W[i, 0] + (W[i, 1] * x)
    ax.plot(x, f, 'c')
f = W[iters-1, 0] + (W[iters-1, 1] * x)
ax.plot(x, f,'r')
print "Intercept:",W[iters-1, 0], "Slope: ",W[iters-1, 1]
ax.scatter(data.X, data.Y, label='Traning Data')
ax.legend(loc=2)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_title('Predicted Y vs. X Size')
```

```
Intercept: -1.0 SLope:  5.0
Out[42]: <matplotlib.text.Text at 0x51759a20>
```

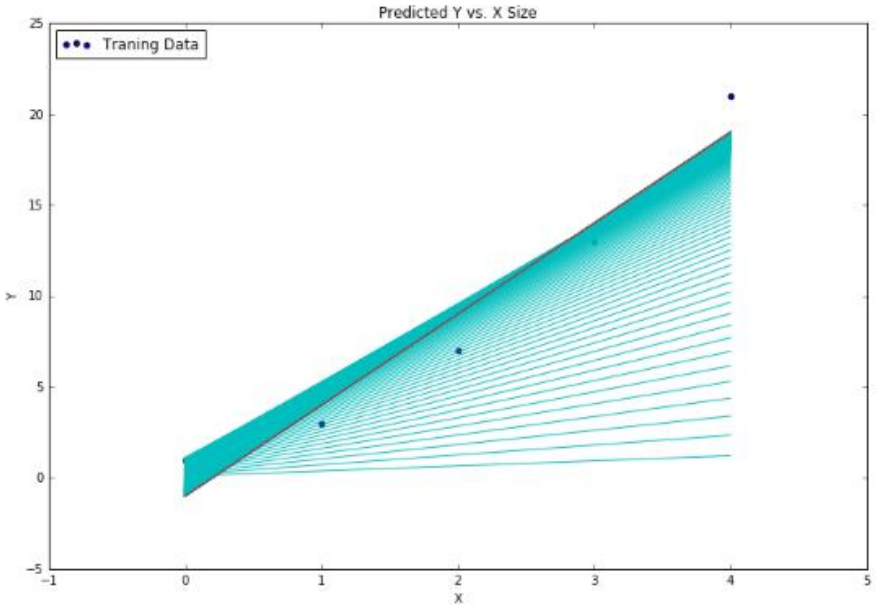

Fig 3: Plot with Predicted Y vs X

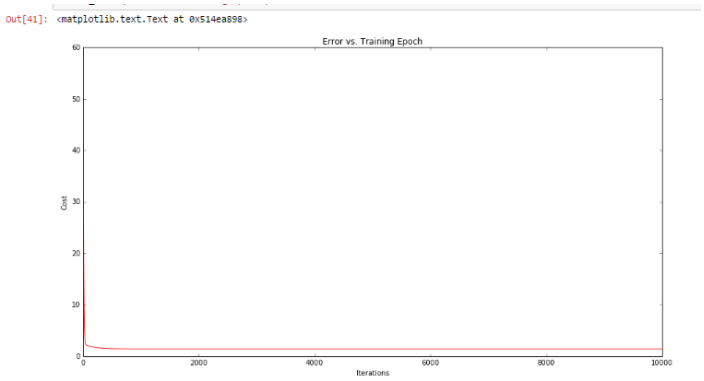

Fig 4: Plot with Error Vs Training Epoch

```
fig, ax = plt.subplots(figsize=(15,8))

ax.plot(np.arange(iters), cost, 'r')

ax.set_xlabel('Iterations')

ax.set_ylabel('Cost') ax.set_title('Error vs. Training Epoch')
```