assignment_1_Tri_Ninh

February 15, 2018

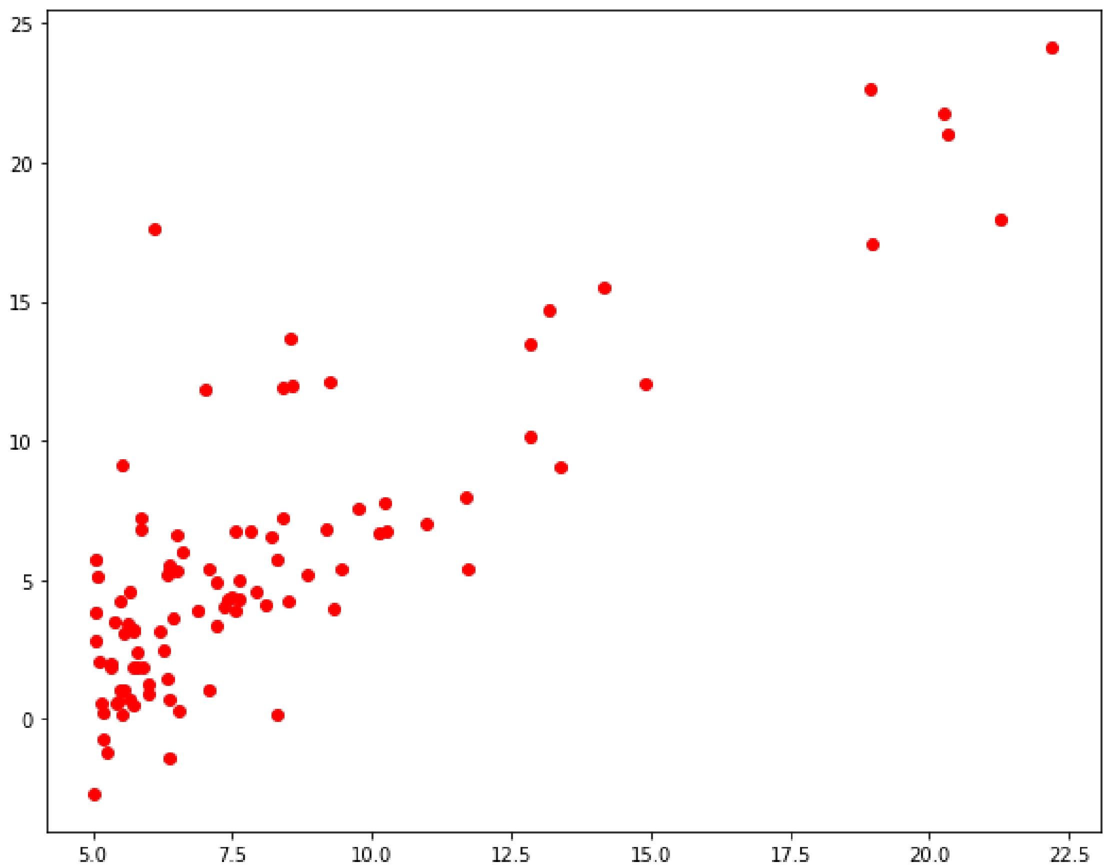
```
In [1]: import pandas as pd
        import matplotlib as mp
        from matplotlib import pyplot as plt
        import numpy as np
        %matplotlib inline
```

0.0.1 1. Linear Regression with one variable

```
In [2]: df0 = pd.read_csv('ex1data1.csv', header=None)
```

```
In [3]: df0.columns=['Population', 'Profit']
        #df0.plot(kind='scatter', x='', y='1', figsize=(12,8))
        df0.insert(0, 'w0', 1)
```

```
In [4]: x=df0['Population'].values
        y=df0['Profit'].values
        plt.figure(figsize=(10,8))
        plt.scatter(x, y,  c='red')
        plt.show()
```




```
In [5]: x = df0['Population'].values
        x

Out[5]: 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 [6]: def cost_function(x, y, w1, w0):
            total_cost = 0
            for i in range(len(x)):
                total_cost += np.power(((w1 * x[i] + w0) - y[i]),2)
            return total_cost / len(x)
```

```
In [7]: oldCost = cost_function(x, y, 1, 1)
        oldCost
```

Out[7]: 20.533040982767009

```
In [8]: def gradient_descent(x, y, w1, w0, alpha, iterations):
            inner_w1_deriv = 0
            inner_w0_deriv = 0
            #temp = 1
            cost = 0
            for _ in range(iterations):
                #cost
                inner_w1_deriv = 0
                inner_w0_deriv = 0
                for i in range(len(x)):
                    # Calculate partial derivatives
                    # -2x(y - (mx + b))
                    inner_w1_deriv += (x[i] * ((w1*x[i]) - y[i]))
                    inner_w0_deriv += ((w1*x[i]) - y[i])
                # We subtract because the derivatives point in direction of steepest ascent
                temp1 = w1 - (2 * (inner_w1_deriv / len(x)) * alpha)
                temp0 = w0 - (2 * (inner_w0_deriv / len(x)) * alpha)
                w1 = temp1
                w0 = temp0
                #temp = cost
                cost= cost_function(x, y, w1, w0)
        #         if j % 10 == 0:
        #             print "iter: "+str(j) + " cost: "+str(cost)
            return w1, w0, cost
```

```
In [9]: alpha = 0.01
        iterations = 100
        w1, w0, cost = gradient_descent(x, y, 1, 1, alpha, iterations)
```

```
In [10]: cost
```

Out[10]: 11.287549032521689

```
In [11]: print 'RMSE %f' % np.sqrt(cost)
```

RMSE 3.359695

```
In [12]: w1
```

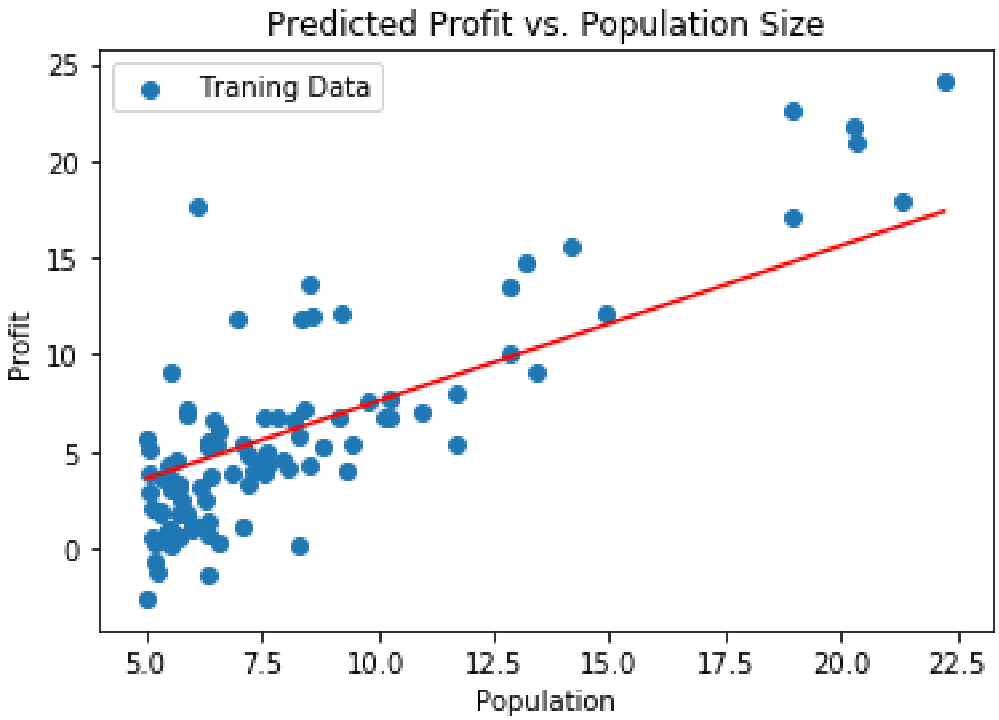
Out[12]: 0.80252684920135675

In [13]: w0

Out[13]: -0.4384414554300905

```
In [14]: x1 = np.linspace(df0.Population.min(), df0.Population.max(), 100)
         f = w1 * x1 + w0
         fig, ax = plt.subplots()
         ax.plot(x1, f, 'r')
         ax.scatter(df0.Population, df0.Profit, label='Traning Data')
         ax.legend(loc=2)
         ax.set_xlabel('Population')
         ax.set_ylabel('Profit')
         ax.set_title('Predicted Profit vs. Population Size')
         fig.show()
```

```
/Users/trininh/anaconda/lib/python2.7/site-packages/matplotlib/figure.py:403: UserWarning: mat
  "matplotlib is currently using a non-GUI backend, "
```



0.0.2 2-1

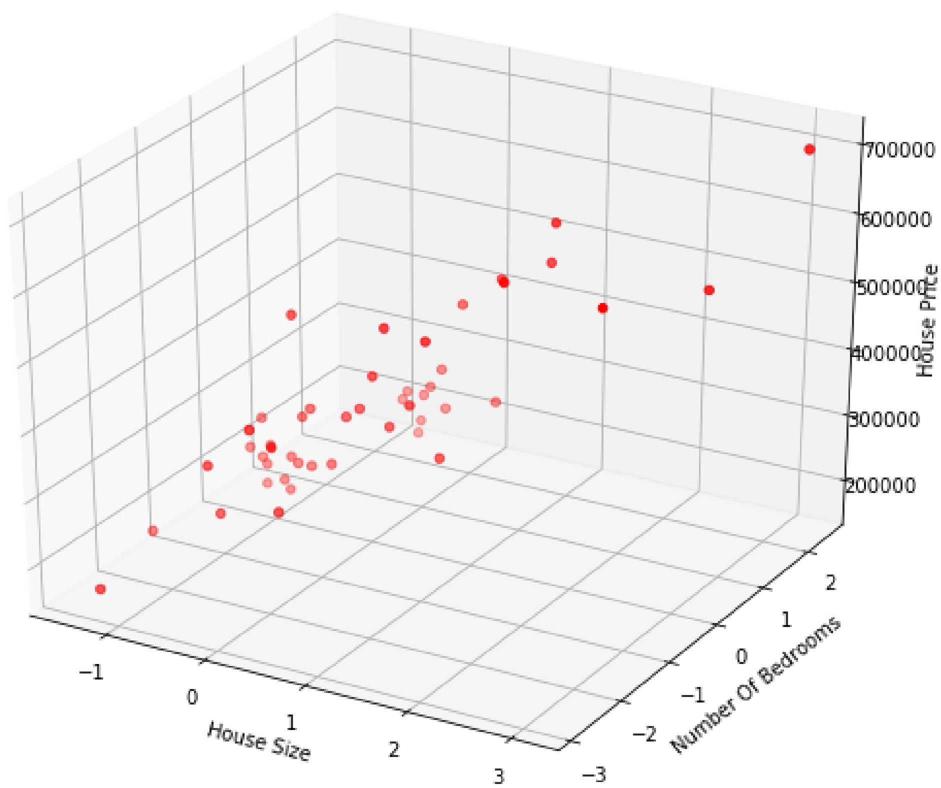
```
In [15]: df1 = pd.read_csv('ex1data2.csv', header=None)
```

```
In [16]: df1.columns=['HouseSize', 'NumberOfBedrooms', 'HousePrice']

In [17]: df1['HouseSize'] = (df1['HouseSize'] - df1['HouseSize'].mean()) / df1['HouseSize'].std
         df1['NumberOfBedrooms'] = (df1['NumberOfBedrooms'] - df1['NumberOfBedrooms'].mean()) /

In [18]: df1.insert(0, 'Ones', 1)

In [19]: from mpl_toolkits.mplot3d import Axes3D
         fig = plt.figure(figsize=(10,8))
         ax = fig.add_subplot(111, projection='3d')
         X = df1['HouseSize']
         Y = df1['NumberOfBedrooms']
         Z = df1['HousePrice']
         ax.scatter(X, Y, Z, c = 'r')
         ax.set_xlabel('House Size')
         ax.set_ylabel('Number Of Bedrooms')
         ax.set_zlabel('House Price')
         plt.show()
```




```
In [20]: x1 = np.matrix(df1.iloc[:, 0:3].values)
         y1 = np.matrix(df1.iloc[:, 3:4].values)
```

```
In [31]: # def cost_function_matrix(x, y, w):
         #     total_cost = ((np.matmul(x,w) - y).T)*(np.matmul(x,w) - y)
         # #     total_cost = (np.matmul(x, w) - y)**2
         #     m = len(x)
         #     cost = total_cost / m
         #     return cost
             #return np.sum(np.square(np.matmul(x, w) - y)) / (2 * len(y))
         m = y1.size
         def h(w,x): #Linear hypothesis function
             return np.dot(x,w)

         def cost_function_matrix(w,x,y): #Cost function
             """
             theta_start is an n- dimensional vector of initial theta guess
             X is matrix with n- columns and m- rows
             y is a matrix with m- rows and 1 column
             """
             #note to self: *.shape is (rows, columns)
             return float((1./(2*m)) * np.dot((h(w,x)-y).T,(h(w,x)-y)))

In [33]: w = np.zeros((3,1))
         #w.reshape((2,1))
         cost_function_matrix(w, x1, y1)

Out[33]: 65591585744.68085

In [23]: x1.shape, y1.shape, w.shape

Out[23]: ((47, 3), (47, 1), (3, 1))

In [67]: m

Out[67]: 47

In [99]: # w.reshape((2,1))
         # parameters = int(w.ravel().shape[1])
         # parameters
         # def gradient_descent_matrix1(x, y, w, alpha, iterations):
         #     temp = np.ones((3, 1))
         #     w = np.ones((3,1))
         #     m = len(x)
         #     costs = np.zeros(iterations)
         #     for i in range(iterations):
         #         deriv = (2.0/m) * np.matmul((np.matmul(x, temp) - y).T, x)
         #         #deriv = (2.0/m) * np.matmul(x.T, np.matmul(x, w) - y)
         #         #print deriv.shape
         #         temp = temp - (alpha * deriv).T
         #         costs[i] = cost_function_matrix(x, y, temp)
         #     w = temp
```

```python
#     return w, costs
def descendGradient(X, y, init_w):
    """
    theta_start is an n- dimensional vector of initial theta guess
    X is matrix with n- columns and m- rows
    """
    w = init_w
    costs = [] #Used to plot cost as function of iteration
    w_history = [] #Used to visualize the minimization path later on
    for i in xrange(iterations):
        temp = w
        costs.append(cost_function_matrix(w,X,y))
        # Buggy line
        #thetahistory.append(list(tmptheta))
        # Fixed line
        w_history.append(list(w[:,0]))
        #error = (h(w, X) - y)
        #Simultaneously updating theta values
        for j in xrange(len(temp)):
            temp[j] = w[j] - (alpha/m)*np.sum((h(w, X) - y)*np.array(X[:,j]).T)
        w = temp
    return w, w_history, costs
```

In [73]:
```python
iterations = 1000
alpha = 0.001
init_w = np.zeros((x1.shape[1],1))
w1, w1_history, costs1 = descendGradient(x1, y1, init_w)
```

In [74]:
```python
w1
```

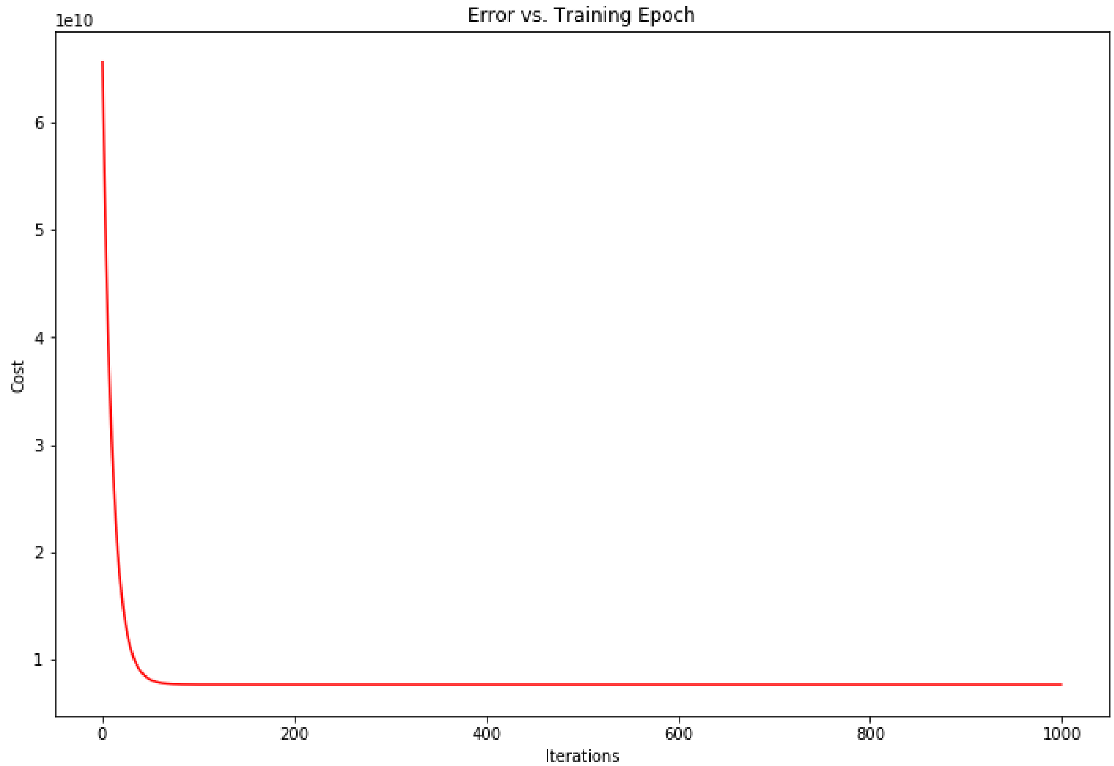
Out[74]:
```
array([[  3.40412766e+05],
       [  9.27978730e-12],
       [  8.47965644e-11]])
```

In [75]:
```python
print 'RMSE %f' % np.sqrt(cost_function_matrix(w1, x1, y1))
```

```
RMSE 87470.910200
```

In [76]:
```python
fig, ax = plt.subplots(figsize=(12,8))
ax.plot(np.arange(iterations), costs1, 'r')
ax.set_xlabel('Iterations')
ax.set_ylabel('Cost')
ax.set_title('Error vs. Training Epoch')
```

Out[76]:
```
<matplotlib.text.Text at 0x11d674290>
```



0.0.3 2-2

```
In [184]: from numpy.linalg import inv
          def normal_equation(x, y):
              return inv(x.T.dot(x)).dot(x.T).dot(y)
          w = normal_equation(x1, y1)
          cost=cost_function_matrix(w, x1, y1)
          print w
          print cost
          #print cost
```

```
[[ 340412.76595745]
 [ 110631.01899357]
 [  -6649.39536298]]
2043280477.39
```

```
In [185]: print 'RMSE %f' % np.sqrt(cost)
```

```
RMSE 45202.660070
```

0.0.4 3-1

In [100]:
```
df2 = pd.read_csv('ex1data3.csv', index_col=0)
```

In [101]:
```
X = df2.iloc[:,0:8].values
Y = df2.iloc[:, 8:9].values
```

In [102]:
```
X2 = np.matrix(X)
Y2 = np.matrix(Y)
w3 = np.ones(X2.shape[1])
```

In [103]:
```
X2
```

Out[103]:
```
matrix([[   8.3252    ,   41.        ,    6.98412698, ...,    2.55555556,
          37.88      , -122.23      ],
        [   8.3014    ,   21.        ,    6.23813708, ...,    2.10984183,
          37.86      , -122.22      ],
        [   7.2574    ,   52.        ,    8.28813559, ...,    2.80225989,
          37.85      , -122.24      ],
        ...,
        [   1.7       ,   17.        ,    5.20554273, ...,    2.3256351 ,
          39.43      , -121.22      ],
        [   1.8672    ,   18.        ,    5.32951289, ...,    2.12320917,
          39.43      , -121.32      ],
        [   2.3886    ,   16.        ,    5.25471698, ...,    2.61698113,
          39.37      , -121.24      ]])
```

In [104]:
```
def feature_normalize(x):
    numberOfFeatures = x.shape[1]
    means = np.array([np.mean(x[:,i]) for i in range(numberOfFeatures)])
    stds = np.array([np.std(x[:,i]) for i in range(numberOfFeatures)])
    normalized = (x - means) / stds

    return normalized
```

In [113]:
```
#X_norm = data.as_matrix(X)
X_norm = feature_normalize(df2.iloc[:,0:8].values)
```

In [114]:
```
#X_norm1 = np.concatenate(np.ones(shape=(20640,1)), X_norm, axis=1)
X_norm1 = np.insert(X_norm, 0, 1, axis=1)
```

In [115]:
```
X_norm1.shape
```

Out[115]:
```
(20640, 9)
```

In [118]:
```
matrix_x = np.matrix(X_norm1)
```

In [175]:
```
# def gradient_descent_matrix2(x, y, w, alpha, iterations):
#     w = np.zeros((x.shape[1], 1))
#     #w = w.reshape((x.shape[1],1))
```

```
#     m = len(X)
#     costs = np.zeros(iterations)
#     for i in range(iterations):
#         deriv = (2.0/m) * np.matmul(x.T, np.matmul(x, w)- y)
#         w = w - alpha * deriv
#         costs[i] = cost_function_matrix(x, y, w)
#     return w, costs
def gradientDescent2(X, y, theta, alpha, iters):
    temp = np.matrix(np.zeros((theta.shape[0],1)))
    parameters = int(theta.ravel().shape[0])
    cost = np.zeros(iters)

    for i in range(iters):
        error = (X * theta) - y

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

        theta = temp
        cost[i] = cost_function_matrix(theta, X, y)

    return theta, cost
```

In [176]:
```
alpha = 0.01
iterations = 1000
init_w = np.zeros((X_norm1.shape[1], 1))
w5, costs5 = gradientDescent2(matrix_x, Y2, init_w, alpha, iterations)
```

In [177]:
```
temp = np.matrix(np.zeros((init_w.shape[0],1)))
temp[2]
```

Out[177]:
```
matrix([[ 0.]])
```

In [178]:
```
w5
```

Out[178]:
```
matrix([[ 2.06846887],
        [ 0.81659877],
        [ 0.17689017],
        [-0.12729893],
        [ 0.14127008],
        [ 0.0166395 ],
        [-0.04392099],
        [-0.48604502],
        [-0.44967077]])
```

In [172]:
```
costs5
```

Out[172]:
```
array([ 1210.11655077,  1188.75282301,  1167.81874126,  1147.30555168,
        1127.20468216,  1107.50773849,  1088.20650045,  1069.29291818,
```

```
1050.7591085 ,  1032.59735139,  1014.80008644,   997.35990956,
 980.26956954,   963.52196491,   947.11014069,   931.0272853 ,
 915.26672757,   899.82193369,   884.68650436,   869.85417195,
 855.3187977 ,   841.07436902,   827.11499681,   813.4349129 ,
 800.02846746,   786.89012657,   774.01446974,   761.39618754,
 749.03007929,   736.9110508 ,   725.03411207,   713.39437519,
 701.98705217,   690.80745287,   679.85098294,   669.11314184,
 658.58952088,   648.2758013 ,   638.16775243,   628.26122983,
 618.5521735 ,   609.03660615,   599.71063146,   590.57043241,
 581.61226967,   572.83247992,   564.22747435,   555.79373707,
 547.52782364,   539.42635957,   531.48603888,   523.70362269,
 516.07593785,   508.59987557,   501.27239009,   494.09049739,
 487.05127394,   480.15185541,   473.38943552,   466.76126477,
 460.26464934,   453.8969499 ,   447.65558055,   441.53800765,
 435.54174881,   429.66437181,   423.9034936 ,   418.25677926,
 412.72194102,   407.29673734,   401.97897192,   396.76649277,
 391.65719136,   386.64900167,   381.73989937,   376.92790094,
 372.21106285,   367.58748075,   363.05528867,   358.61265821,
 354.25779783,   349.98895206,   345.80440078,   341.70245848,
 337.6814736 ,   333.73982781,   329.87593532,   326.08824226,
 322.375226  ,   318.73539455,   315.16728588,   311.66946737,
 308.24053519,   304.87911373,   301.58385499,   298.35343808,
 295.1865686 ,   292.08197818,   289.0384239 ,   286.05468779,
 283.12957633,   280.26191994,   277.45057253,   274.69441098,
 271.99233472,   269.34326524,   266.74614565,   264.19994028,
 261.70363419,   259.2562328 ,   256.85676148,   254.5042651 ,
 252.19780768,   249.936472  ,   247.71935921,   245.54558846,
 243.41429653,   241.32463752,   239.27578242,   237.26691885,
 235.29725068,   233.36599771,   231.47239537,   229.61569436,
 227.79516041,   226.0100739 ,   224.25972964,   222.54343651,
 220.86051724,   219.2103081 ,   217.59215861,   216.00543131,
 214.44950149,   212.92375691,   211.42759758,   209.96043549,
 208.52169439,   207.11080953,   205.72722747,   204.37040578,
 203.03981292,   201.73492792,   200.45524023,   199.20024951,
 197.9694654 ,   196.76240731,   195.57860428,   194.41759472,
 193.27892628,   192.16215562,   191.06684825,   189.99257835,
 188.9389286 ,   187.90549002,   186.89186175,   185.89765097,
 184.92247267,   183.96594952,   183.02771172,   182.10739685,
 181.20464972,   180.31912219,   179.4504731 ,   178.59836806,
 177.76247938,   176.94248587,   176.13807276,   175.34893154,
 174.57475985,   173.81526137,   173.07014566,   172.3391281 ,
 171.6219297 ,   170.91827705,   170.22790219,   169.55054248,
 168.88594053,   168.23384407,   167.59400583,   166.9661835 ,
 166.35013957,   165.74564127,   165.15246045,   164.57037353,
 163.99916134,   163.4386091 ,   162.8885063 ,   162.34864661,
 161.81882782,   161.29885174,   160.7885241 ,   160.28765453,
 159.79605642,   159.31354689,   158.83994669,   158.37508012,
 157.91877501,   157.47086258,   157.03117742,   156.59955741,
```

```
156.17584365,   155.7598804 ,   155.35151501,   154.95059788,
154.55698237,   154.17052475,   153.79108417,   153.41852255,
153.05270457,   152.6934976 ,   152.34077164,   151.99439927,
151.6542556 ,   151.32021823,   150.99216719,   150.66998489,
150.35355607,   150.04276776,   149.73750923,   149.43767198,
149.14314962,   148.85383789,   148.5696346 ,   148.2904396 ,
148.01615471,   147.74668371,   147.48193228,   147.22180797,
146.96622019,   146.7150801 ,   146.46830068,   146.22579659,
145.9874842 ,   145.75328154,   145.52310826,   145.29688562,
145.07453642,   144.85598501,   144.64115721,   144.42998034,
144.22238316,   144.01829582,   143.81764987,   143.62037822,
143.4264151 ,   143.23569605,   143.04815788,   142.86373865,
142.68237767,   142.50401542,   142.32859358,   142.15605499,
141.98634359,   141.81940447,   141.65518377,   141.49362874,
141.33468763,   141.17830974,   141.02444537,   140.87304579,
140.72406326,   140.57745097,   140.43316302,   140.29115445,
140.15138116,   140.01379995,   139.87836846,   139.74504516,
139.61378935,   139.48456114,   139.35732142,   139.23203186,
139.10865488,   138.98715365,   138.86749205,   138.74963471,
138.63354693,   138.5191947 ,   138.40654469,   138.29556421,
138.18622124,   138.07848438,   137.97232285,   137.86770648,
137.76460569,   137.66299149,   137.56283547,   137.46410976,
137.36678705,   137.27084058,   137.1762441 ,   137.08297189,
136.99099873,   136.90029991,   136.81085119,   136.72262881,
136.63560949,   136.54977041,   136.46508918,   136.38154387,
136.29911298,   136.21777542,   136.13751053,   136.05829806,
135.98011814,   135.9029513 ,   135.82677847,   135.75158092,
135.67734033,   135.60403871,   135.53165843,   135.46018222,
135.38959313,   135.31987456,   135.25101024,   135.18298419,
135.11578077,   135.04938465,   134.98378079,   134.91895445,
134.85489117,   134.79157678,   134.7289974 ,   134.66713942,
134.60598947,   134.54553447,   134.4857616 ,   134.42665826,
134.36821214,   134.31041113,   134.25324339,   134.1966973 ,
134.14076145,   134.08542469,   134.03067606,   133.97650482,
133.92290045,   133.86985263,   133.81735124,   133.76538636,
133.71394826,   133.66302741,   133.61261446,   133.56270025,
133.51327578,   133.46433225,   133.41586102,   133.36785361,
133.32030173,   133.27319723,   133.22653212,   133.18029859,
133.13448895,   133.08909568,   133.0441114 ,   132.99952889,
132.95534103,   132.91154089,   132.86812163,   132.82507658,
132.78239918,   132.740083  ,   132.69812173,   132.65650919,
132.61523934,   132.57430621,   132.53370398,   132.49342695,
132.45346951,   132.41382616,   132.37449151,   132.33546029,
132.29672732,   132.25828751,   132.22013588,   132.18226755,
132.14467773,   132.10736172,   132.07031492,   132.0335328 ,
131.99701095,   131.96074501,   131.92473072,   131.88896392,
131.8534405 ,   131.81815644,   131.78310781,   131.74829073,
131.71370143,   131.67933619,   131.64519135,   131.61126335,
```

```
131.57754869,   131.54404391,   131.51074566,   131.47765062,
131.44475556,   131.41205728,   131.37955268,   131.34723868,
131.3151123 ,   131.28317058,   131.25141064,   131.21982965,
131.18842483,   131.15719345,   131.12613285,   131.0952404 ,
131.06451353,   131.03394972,   131.00354649,   130.97330143,
130.94321214,   130.9132763 ,   130.88349161,   130.85385583,
130.82436675,   130.79502221,   130.7658201 ,   130.73675832,
130.70783485,   130.67904767,   130.65039482,   130.62187438,
130.59348445,   130.56522318,   130.53708875,   130.50907938,
130.4811933 ,   130.45342881,   130.42578422,   130.39825788,
130.37084815,   130.34355345,   130.31637221,   130.2893029 ,
130.26234402,   130.23549409,   130.20875166,   130.18211531,
130.15558365,   130.1291553 ,   130.10282892,   130.07660319,
130.05047682,   130.02444854,   129.99851711,   129.9726813 ,
129.94693991,   129.92129176,   129.89573569,   129.87027058,
129.84489531,   129.81960879,   129.79440993,   129.7692977 ,
129.74427105,   129.71932897,   129.69447046,   129.66969456,
129.64500029,   129.62038671,   129.59585291,   129.57139797,
129.547021  ,   129.52272112,   129.49849748,   129.47434923,
129.45027555,   129.42627562,   129.40234864,   129.37849382,
129.35471041,   129.33099764,   129.30735476,   129.28378106,
129.26027581,   129.23683831,   129.21346788,   129.19016382,
129.16692548,   129.14375221,   129.12064335,   129.09759829,
129.07461639,   129.05169705,   129.02883967,   129.00604367,
128.98330846,   128.96063349,   128.93801818,   128.915462  ,
128.89296441,   128.87052488,   128.84814289,   128.82581793,
128.80354949,   128.78133709,   128.75918025,   128.73707848,
128.71503132,   128.69303831,   128.67109899,   128.64921293,
128.62737969,   128.60559884,   128.58386995,   128.56219262,
128.54056644,   128.51899099,   128.49746591,   128.47599078,
128.45456524,   128.43318892,   128.41186143,   128.39058242,
128.36935154,   128.34816843,   128.32703276,   128.30594417,
128.28490235,   128.26390696,   128.24295767,   128.22205418,
128.20119618,   128.18038334,   128.15961538,   128.138892  ,
128.1182129 ,   128.0975778 ,   128.07698641,   128.05643846,
128.03593368,   128.01547178,   127.99505252,   127.97467562,
127.95434083,   127.93404789,   127.91379657,   127.89358661,
127.87341776,   127.85328981,   127.8332025 ,   127.81315561,
127.79314892,   127.77318219,   127.75325522,   127.73336777,
127.71351965,   127.69371063,   127.67394052,   127.6542091 ,
127.63451618,   127.61486156,   127.59524504,   127.57566643,
127.55612554,   127.53662219,   127.51715619,   127.49772735,
127.47833551,   127.45898048,   127.43966209,   127.42038017,
127.40113454,   127.38192505,   127.36275153,   127.34361381,
127.32451175,   127.30544517,   127.28641392,   127.26741785,
127.24845682,   127.22953066,   127.21063923,   127.19178239,
127.17296   ,   127.15417191,   127.13541798,   127.11669807,
127.09801206,   127.07935979,   127.06074115,   127.042156  ,
```

```
127.02360422,   127.00508566,   126.98660022,   126.96814776,
126.94972816,   126.9313413 ,   126.91298705,   126.89466531,
126.87637596,   126.85811887,   126.83989393,   126.82170104,
126.80354008,   126.78541093,   126.7673135 ,   126.74924767,
126.73121333,   126.71321039,   126.69523873,   126.67729826,
126.65938887,   126.64151046,   126.62366294,   126.60584619,
126.58806014,   126.57030467,   126.5525797 ,   126.53488513,
126.51722086,   126.49958682,   126.48198289,   126.464409  ,
126.44686505,   126.42935096,   126.41186663,   126.39441199,
126.37698694,   126.3595914 ,   126.34222529,   126.32488852,
126.30758101,   126.29030267,   126.27305343,   126.25583321,
126.23864193,   126.22147951,   126.20434586,   126.18724092,
126.1701646 ,   126.15311684,   126.13609755,   126.11910666,
126.10214409,   126.08520978,   126.06830365,   126.05142563,
126.03457564,   126.01775363,   126.0009595 ,   125.98419321,
125.96745467,   125.95074382,   125.93406058,   125.91740491,
125.90077672,   125.88417594,   125.86760253,   125.8510564 ,
125.83453749,   125.81804574,   125.80158109,   125.78514348,
125.76873283,   125.75234908,   125.73599218,   125.71966207,
125.70335867,   125.68708194,   125.67083181,   125.65460822,
125.63841111,   125.62224042,   125.60609609,   125.58997808,
125.57388631,   125.55782072,   125.54178128,   125.5257679 ,
125.50978055,   125.49381916,   125.47788368,   125.46197405,
125.44609022,   125.43023214,   125.41439974,   125.39859297,
125.38281179,   125.36705613,   125.35132595,   125.33562119,
125.3199418 ,   125.30428773,   125.28865892,   125.27305533,
125.25747689,   125.24192357,   125.22639531,   125.21089206,
125.19541377,   125.17996039,   125.16453187,   125.14912815,
125.13374921,   125.11839497,   125.1030654 ,   125.08776044,
125.07248005,   125.05722418,   125.04199278,   125.02678581,
125.01160321,   124.99644494,   124.98131095,   124.9662012 ,
124.95111564,   124.93605423,   124.92101691,   124.90600364,
124.89101438,   124.87604909,   124.8611077 ,   124.84619019,
124.8312965 ,   124.8164266 ,   124.80158043,   124.78675796,
124.77195913,   124.75718391,   124.74243225,   124.7277041 ,
124.71299943,   124.69831819,   124.68366034,   124.66902584,
124.65441464,   124.63982669,   124.62526197,   124.61072042,
124.596202  ,   124.58170668,   124.5672344 ,   124.55278514,
124.53835884,   124.52395546,   124.50957498,   124.49521733,
124.48088249,   124.4665704 ,   124.45228104,   124.43801436,
124.42377033,   124.40954889,   124.39535001,   124.38117365,
124.36701978,   124.35288834,   124.33877931,   124.32469264,
124.31062829,   124.29658623,   124.28256641,   124.2685688 ,
124.25459335,   124.24064004,   124.22670881,   124.21279964,
124.19891248,   124.18504729,   124.17120405,   124.1573827 ,
124.14358321,   124.12980555,   124.11604967,   124.10231555,
124.08860313,   124.07491239,   124.06124329,   124.04759578,
124.03396984,   124.02036542,   124.0067825 ,   123.99322102,
```

```
123.97968097,   123.96616229,   123.95266495,   123.93918892,
123.92573417,   123.91230064,   123.89888832,   123.88549716,
123.87212712,   123.85877818,   123.84545029,   123.83214342,
123.81885754,   123.80559261,   123.79234859,   123.77912545,
123.76592315,   123.75274166,   123.73958095,   123.72644097,
123.7133217 ,   123.7002231 ,   123.68714513,   123.67408776,
123.66105096,   123.6480347 ,   123.63503893,   123.62206362,
123.60910874,   123.59617426,   123.58326013,   123.57036634,
123.55749284,   123.5446396 ,   123.53180658,   123.51899376,
123.5062011 ,   123.49342857,   123.48067612,   123.46794374,
123.45523138,   123.44253901,   123.42986661,   123.41721413,
123.40458155,   123.39196883,   123.37937593,   123.36680283,
123.3542495 ,   123.3417159 ,   123.32920199,   123.31670775,
123.30423315,   123.29177814,   123.27934271,   123.26692681,
123.25453042,   123.2421535 ,   123.22979602,   123.21745795,
123.20513926,   123.19283991,   123.18055988,   123.16829913,
123.15605763,   123.14383535,   123.13163226,   123.11944833,
123.10728352,   123.09513781,   123.08301115,   123.07090353,
123.05881492,   123.04674527,   123.03469456,   123.02266276,
123.01064983,   122.99865575,   122.98668049,   122.97472402,
122.9627863 ,   122.9508673 ,   122.938967  ,   122.92708536,
122.91522235,   122.90337795,   122.89155212,   122.87974483,
122.86795606,   122.85618577,   122.84443393,   122.83270051,
122.82098549,   122.80928883,   122.7976105 ,   122.78595048,
122.77430873,   122.76268522,   122.75107993,   122.73949282,
122.72792387,   122.71637305,   122.70484033,   122.69332567,
122.68182905,   122.67035044,   122.65888981,   122.64744714,
122.63602238,   122.62461552,   122.61322653,   122.60185537,
122.59050202,   122.57916644,   122.56784862,   122.55654852,
122.54526611,   122.53400136,   122.52275425,   122.51152475,
122.50031283,   122.48911846,   122.47794161,   122.46678225,
122.45564036,   122.44451591,   122.43340886,   122.4223192 ,
122.4112469 ,   122.40019191,   122.38915423,   122.37813382,
122.36713065,   122.35614469,   122.34517593,   122.33422432,
122.32328984,   122.31237247,   122.30147217,   122.29058893,
122.27972271,   122.26887348,   122.25804122,   122.2472259 ,
122.23642749,   122.22564597,   122.2148813 ,   122.20413347,
122.19340245,   122.1826882 ,   122.17199071,   122.16130993,
122.15064586,   122.13999845,   122.12936769,   122.11875355,
122.108156  ,   122.09757501,   122.08701056,   122.07646262,
122.06593116,   122.05541617,   122.0449176 ,   122.03443544,
122.02396966,   122.01352023,   122.00308713,   121.99267033,
121.98226981,   121.97188553,   121.96151748,   121.95116562,
121.94082993,   121.93051039,   121.92020697,   121.90991964,
121.89964838,   121.88939316,   121.87915395,   121.86893074,
121.85872349,   121.84853219,   121.83835679,   121.82819728,
121.81805364,   121.80792584,   121.79781385,   121.78771764,
121.7776372 ,   121.76757249,   121.7575235 ,   121.74749019,
```

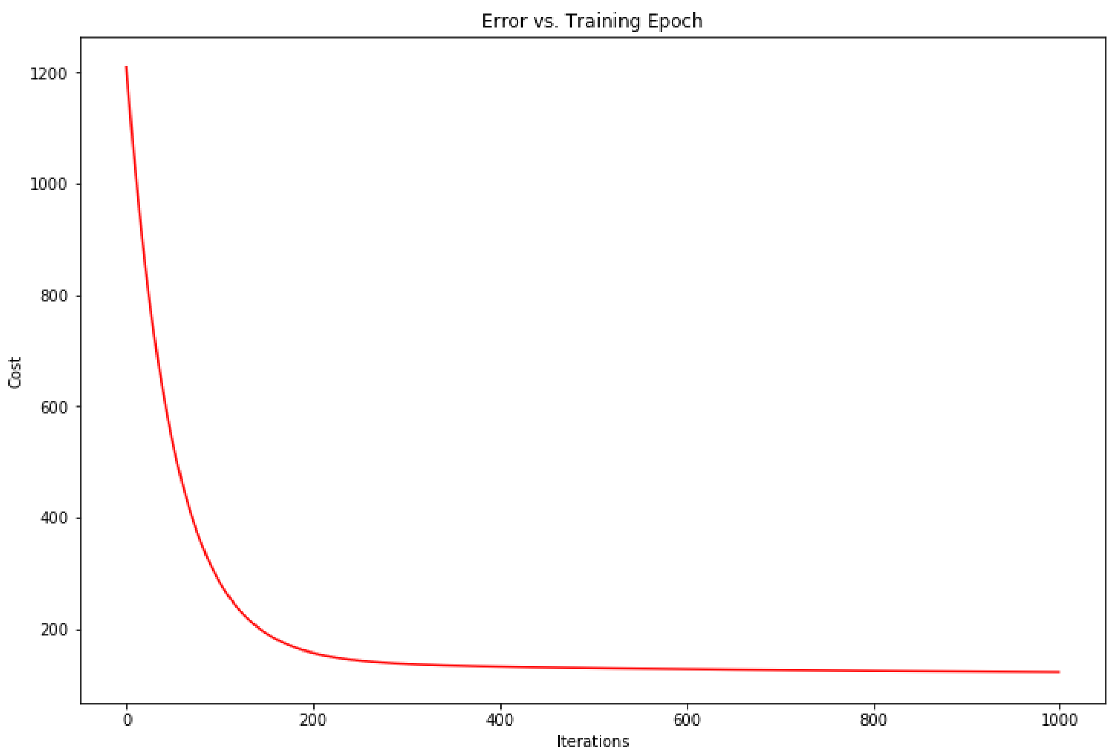
```
                   121.73747255,   121.72747054,   121.71748414,   121.70751333,
                   121.69755808,   121.68761837,   121.67769417,   121.66778546,
                   121.65789221,   121.6480144 ,   121.638152  ,   121.62830499,
                   121.61847335,   121.60865705,   121.59885606,   121.58907036,
                   121.57929993,   121.56954474,   121.55980477,   121.55007999,
                   121.54037038,   121.53067592,   121.52099657,   121.51133233,
                   121.50168316,   121.49204903,   121.48242993,   121.47282584,
                   121.46323671,   121.45366255,   121.44410331,   121.43455897])
```

In [179]:
```
print 'RMSE %f' % float(np.sqrt(cost_function_matrix(w5, X_norm1, Y2)))
```

```
RMSE 11.019735
```

In [180]:
```
fig, ax = plt.subplots(figsize=(12,8))
ax.plot(np.arange(iterations), costs5, 'r')
ax.set_xlabel('Iterations')
ax.set_ylabel('Cost')
ax.set_title('Error vs. Training Epoch')
```

Out[180]: <matplotlib.text.Text at 0x11e2fcbd0>



0.0.5 3-2

```
In [187]: w6 = normal_equation(matrix_x, Y2)
          cost6=cost_function_matrix(w6, matrix_x, Y2)

In [188]: w6

Out[188]: matrix([[ 2.06855817],
                  [ 0.8296193 ],
                  [ 0.11875165],
                  [-0.26552688],
                  [ 0.30569623],
                  [-0.004503  ],
                  [-0.03932627],
                  [-0.89988565],
                  [-0.870541  ]])

In [190]: print 'RMSE %f' % np.sqrt(cost6)

RMSE 10.729748
```

0.0.6 3-3

```
In [191]: from sklearn import linear_model
          reg = linear_model.LinearRegression()
          reg.fit(X_norm, Y2)

Out[191]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [195]: print 'Weights:'
          print reg.coef_

Weights:
[[ 0.8296193   0.11875165 -0.26552688  0.30569623 -0.004503   -0.03932627
  -0.89988565 -0.870541  ]]

In [194]: from sklearn.metrics import mean_squared_error
          print 'RMSE: %f' % np.sqrt(mean_squared_error(reg.predict(X_norm), Y2))

RMSE: 0.724100
```

0.0.7 3-4

```
In [2176]: import tensorflow as tf
```

In [2177]:
```
m = X_norm.shape[0]
n = X_norm.shape[1]
X_train = tf.placeholder(tf.float32, [m,n])
Y_train = tf.placeholder(tf.float32, [m,1])
weights = tf.Variable(tf.zeros([n,1], dtype=np.float32), name='weight')
bias = tf.Variable(tf.zeros([1], dtype=np.float32), name='bias')
```

In [2178]:
```
learning_rate = 0.1
iterations = 5000
cost_history = np.empty(shape=[1],dtype=float)
init = tf.initialize_all_variables()
```

```
WARNING:tensorflow:From <ipython-input-2178-b9a007e7fbf2>:4: initialize_all_variables (from ter
Instructions for updating:
Use `tf.global_variables_initializer` instead.
```

In [2179]:
```
y_hat = tf.add(tf.matmul(X_train, weights), bias)
cost = tf.reduce_mean(tf.square(y_hat - Y_train))
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
```

In [2180]:
```
#costs = np.empty(shape=[1], dtype=float)
#costs = np.zeros(iterations)

sess = tf.Session()
sess.run(init)

for i in range(1, iterations):
    sess.run(optimizer, feed_dict={
        X_train: X_norm,
        Y_train: Y2
    })
    cost_history = np.append(cost_history,sess.run(cost,feed_dict={X_train: X_norm,Y

print('bias:', sess.run(bias))
print('weights:', sess.run(weights))
```

```
('bias:', array([ 2.06855774], dtype=float32))
('weights:', array([[ 0.82962048],
       [ 0.11875186],
       [-0.26552892],
       [ 0.30569792],
       [-0.00450296],
       [-0.03932633],
       [-0.89988297],
       [-0.87053841]], dtype=float32))
```

In [2182]:
```
pred_y = sess.run(y_hat, feed_dict={X_train: X_norm})
mse = tf.reduce_mean(tf.square(pred_y - Y2))
print "RMSE: %.4f" % np.sqrt(sess.run(mse))
```

```
RMSE: 0.7241
```

```
In [2183]: sess.close()
```