Place csv in the same folder as .pynb notebook

In [1]:
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from math import sqrt
import tensorflow as tf
```

1 Linear regression with one variable from scratch

In [2]:
```
data1 = pd.read_csv("ex1data1.csv",header=None)
X_1= data1.as_matrix(columns=data1.columns[0:1])
Y_1 = data1.as_matrix(columns=data1.columns[1:])
X_new_1 = np.c_[np.ones((data1.shape[0], 1)), X_1]
```

In [18]:
```
from __future__ import division
eta = 0.01  # learning rate
n_iterations = 10000
m1 = data1.shape[0]
theta_0 = np.random.randn()
theta_1 = np.random.randn()  # random initialization
gradients_X = 0
gradients_b = 0
for iteration in range(n_iterations):
    for x in range(m1):
        gradients_b = gradients_b+2*((X_new_1[x,1]*theta_1 + theta_0)-Y_
1[x])
        gradients_X = gradients_X+2*X_new_1[x,1]*((X_new_1[x,1]*theta_1
+ theta_0)-Y_1[x])
    gradients_b = gradients_b/m1
    gradients_X = gradients_X/m1
    theta_1 = theta_1 - eta * gradients_X
    theta_0 = theta_0 - eta * gradients_b
print (theta_1,theta_0)
```

```
[1.19303364] [-3.89578088]
```

In [6]:
```
## TO Verify

theta_normal1 = np.linalg.inv(X_new_1.T.dot(X_new_1)).dot(X_new_1.T).dot
(Y_1)
print (theta_normal1)
```

```
[[-3.89578088]
 [ 1.19303364]]
```

1 RMSE Linear regression with one variable from scratch

In [21]:
```
errorsquare = 0
for x in range(m1):
    predict = X_new_1[x,1]*theta_1+theta_0
    errorsquare  = errorsquare+ np.square(predict-Y_1[x])
rms_1 = np.sqrt(errorsquare/m1)
print (rms_1)
```

```
[2.99231395]
```

2 -Linear regression with two variables from scratch

In [23]:
```
data2 = pd.read_csv("ex1data2.csv",header=None)
scaler = StandardScaler()
X = data2.as_matrix(columns=data2.columns[0:2])
Y = data2.as_matrix(columns=data2.columns[2:])
X_new = scaler.fit_transform(X)
X_new = np.c_[np.ones((data2.shape[0], 1)), X_new]
```

```
/anaconda2/envs/carnd-term1/lib/python3.5/site-packages/sklearn/utils/v
alidation.py:475: DataConversionWarning: Data with input dtype int64 wa
s converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

In [28]:
```
eta = 0.01  # learning rate
n_iterations = 1000
m1_2 = data2.shape[0]
theta_2_0 = np.random.randn()
theta_2_1 = np.random.randn()
theta_2_2 = np.random.randn()# random initialization
gradients_X1 = 0
gradients_X2 = 0
gradients_b = 0
for iteration in range(n_iterations):
    for x in range(m1_2):
        gradients_b = gradients_b+2*((X_new[x,1]*theta_2_1 + X_new[x,1]*
theta_2_2 + theta_2_0)-Y[x])
        gradients_X1 = gradients_X1+2*X_new[x,1]*((X_new[x,1]*theta_2_1
+X_new[x,2]*theta_2_2+ theta_2_0)-Y[x])
        gradients_X2 = gradients_X2+2*X_new[x,2]*((X_new[x,1]*theta_2_1
+X_new[x,2]*theta_2_2+ theta_2_0)-Y[x])
    gradients_b = gradients_b/m1_2
    gradients_X1 = gradients_X1/m1_2
    gradients_X2 = gradients_X2/m1_2
    theta_2_0 = theta_2_0 - eta * gradients_b
    theta_2_1 = theta_2_1 - eta * gradients_X1
    theta_2_2 = theta_2_2 - eta * gradients_X2
print (theta_2_1,theta_2_2,theta_2_0)
```

```
[109440.847039] [-6571.3583103] [340412.76559315]
```

2 -Linear regression with two variables from scratch - RMSE

In [29]:
```
errorsquare2 = 0
for x in range(m1_2):
    predict = X_new[x,1]*theta_2_1+X_new[x,2]*theta_2_2+theta_2_0
    errorsquare2  = errorsquare2+ np.square(predict-Y[x])
rms_2 = np.sqrt(errorsquare2/m1_2)
print (rms_2)
```

```
[63926.21525564]
```

2-1 Linear regression with two variables using matrix

In [31]:
```
eta = 0.01  # learning rate
n_iterations = 10000
theta = np.random.randn(data2.shape[1],1)  # random initialization
for iteration in range(n_iterations):
    #print (iteration)
    gradients = 2/m1_2 * X_new.T.dot(X_new.dot(theta)- Y)
    theta = theta - eta * gradients
print (theta)
```

```
[[340412.76595745]
 [109447.76551898]
 [ -6578.27679028]]
```

2-1 RMSE Linear regression with two variables using matrix

In [32]:
```
predict_y = X_new.dot(theta)
rms = sqrt(mean_squared_error(Y, predict_y))
print (rms)
```

```
63926.21492615641
```

2-2. Linear regression with two variables using Normal equation

In [33]:
```
theta_normal = np.linalg.inv(X_new.T.dot(X_new)).dot(X_new.T).dot(Y)
print (theta_normal)
```

```
[[340412.76595745]
 [109447.76551898]
 [ -6578.27679028]]
```

2-2 RMSE Linear regression with two variables using matrix

In [34]:
```
predict_y2 = X_new.dot(theta_normal)
rms_2_normal = sqrt(mean_squared_error(Y, predict_y2))
print (rms_2_normal)
```

```
63926.2149261564
```

3 Linear regression with multiple variables

3-1. Linear regression with multiple variables using matrix

In [35]:
```
data3 = pd.read_csv("ex1data3.csv")
del data3['Unnamed: 0']
```

In [36]:
```
X_3 = data3.as_matrix(columns=data3.columns[0:8])
Y_3 = data3.as_matrix(columns=data3.columns[8:])
X_3_new = scaler.fit_transform(X_3)
X_3_new = np.c_[np.ones((data3.shape[0], 1)), X_3_new]
```

In [37]:
```
eta = 0.01  # learning rate
n_iterations = 10000
m3 = data3.shape[0]

theta_3 = np.random.randn(data3.shape[1],1)  # random initialization
for iteration in range(n_iterations):
    #print (iteration)
    gradients = 2/m3 * X_3_new.T.dot(X_3_new.dot(theta_3)- Y_3)
    theta_3 = theta_3 - eta * gradients
print (theta_3)
```

```
[[ 2.06855817]
 [ 0.82961554]
 [ 0.11875097]
 [-0.26551973]
 [ 0.3056903 ]
 [-0.0045032 ]
 [-0.03932613]
 [-0.89989418]
 [-0.87054909]]
```

3-1. RMSE - Linear regression with multiple variables using matrix

In [20]:
```
predict_y3 = X_3_new.dot(theta_3)
rms_3_matrix = sqrt(mean_squared_error(Y_3, predict_y3))
print (rms_3_matrix)
```

```
0.7241001216741928
```

3-2. Linear regression with multiple variables using Normal equation

In [39]:
```
theta_3_normal = np.linalg.inv(X_3_new.T.dot(X_3_new)).dot(X_3_new.T).do
t(Y_3)
print (theta_3_normal)
```

```
[[ 2.06855817]
 [ 0.8296193 ]
 [ 0.11875165]
 [-0.26552688]
 [ 0.30569623]
 [-0.004503  ]
 [-0.03932627]
 [-0.89988565]
 [-0.870541  ]]
```

3-2. RMSE - Linear regression with multiple variables using normal equation

In [40]:
```
predict_y3_normal = X_3_new.dot(theta_3_normal)
rms_3_normal = sqrt(mean_squared_error(Y_3, predict_y3_normal))
print (rms_3_normal)
```

```
0.7241001216576544
```

3-3. Linear regression with multiple variables using scikit-learn linear regression model

In [41]:
```
import sklearn
regr_model = sklearn.linear_model.LinearRegression()
X_3_regr_model = scaler.fit_transform(X_3)
regr_model.fit(X_3_regr_model, Y_3)
Y_predict_model = regr_model.predict(X_3_regr_model)
```

3-3. RMSE - Linear regression with multiple variables using scikit-learn linear regression model

In [42]:
```
rms_3_model = sqrt(mean_squared_error(Y_3, Y_predict_model))
print (rms_3_model)
```

```
0.7241001216576544
```

3-4. Linear regression with multiple variables using TensorFlow with RMSE

In [43]:
```
n_epochs = 10000
learning_rate = 0.01
tf.reset_default_graph()
X_4 = tf.constant(X_3_new, dtype=tf.float32, name="X")
y = tf.constant(Y_3, dtype=tf.float32, name="y")
theta = tf.Variable(tf.random_uniform([data3.shape[1], 1], -1.0, 1.0), n
ame="theta")
y_pred_4 = tf.matmul(X_4, theta, name="predictions")
error = y_pred_4 - y
mse = tf.sqrt(tf.reduce_mean(tf.square(error), name="mse"))
gradients = 2/m3 * tf.matmul(tf.transpose(X_4), error)
training_op = tf.assign(theta, theta - learning_rate * gradients)

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)

    for epoch in range(n_epochs):
        sess.run(training_op)
    best_RMSE = mse.eval()
    best_theta = theta.eval()
    print (best_RMSE)
```

```
0.7241
```