Tracy Sun
COEN 129
Sprint 2017

HW 3 Report

Source Code:

```
'''
   File: regression.py
   Desciption: Perform linear regression and ridge regression analysis on training and test data set
   Author: Tracy Sun
'''

import math
import numpy as np
import re

#number of test & training data
nTraining = 1595
nTest = 399
nXi = 95    #num of features

# read txt file into data
def read_file(file_name):
    data = []
    # read every float number into data sets
    with open(file_name,'r') as f:
        for line in f:
            line = re.findall(r'[-+]?\d+\.\d+', line)
            if not line:
                continue
            else:
                data.append(line)
        f.close()

    data = [[float(float(num)) for num in row] for row in data]
    data = np.array(data)
    return data

# calculate w
def calc_w(x,x_tran,y,lamda):
    iden = np.identity(nXi+1)
    a = np.linalg.inv(np.dot(x_tran,x)+(lamda*iden))
    b = np.dot(x_tran,y)
    w = np.dot(a,b)
    return w

# calculate RMSE
def calc_rmse(y1,y2,m):
    sum = 0.0
    diff = np.abs(y1-y2)
    for num in diff:
        sum += math.pow(num,2)
    rmse = math.sqrt(sum/m)
    return rmse
```

```
# gradient decent
def gradient_decent(w_t,l_rate,x,x_trans,y,ep,lam):

    while 1:
        w_t1 = w_t + l_rate * (np.dot(x_trans,(y - np.dot(x,w_t))) - lam * w_t)
        sum_error = 0
        for i in range(96):
            sum_error += float(abs(w_t[i] - w_t1[i]))
        sum_error /= 96
        if (sum_error <= ep):
            break
        w_t = w_t1
    return w_t


if __name__ == '__main__':

    train_file = "crime-train.txt"
    test_file = "crime-test.txt"

    # training and test data set
    train_data = read_file(train_file)
    test_data = read_file(test_file)

    y_train = train_data[:,0]
    y_train = np.reshape(y_train,(nTraining,1))

    y_test = test_data[:,0]
    y_test = np.reshape(y_test,(nTest,1))

    # insert column of 1s at the last column of X
    x_train = train_data[:,1:]
    x_test = test_data[:,1:]
    x_train = np.insert(x_train,nXi,1,axis=1)
    x_test = np.insert(x_test,nXi,1,axis=1)

    x_traintrans = x_train.transpose()
    x_testrans = x_test.transpose()

    # perform linear regression close form and calculate w
    w = calc_w(x_train,x_traintrans,y_train,0)

    #calculate Yi
    yi_train = np.dot(x_train,w)
    yi_test = np.dot(x_test,w)

    # RMSE for both training and test data
    train_rmse = calc_rmse(yi_train,y_train,nTraining)
    test_rmse = calc_rmse(yi_test,y_test,nTest)

    # ridge regression using k fold cross validation
    lam = 400
    best_lambda = lam
    k = 5
    n = nTraining/5    #k fold data size = 319
    min_rmse = 1
```

```
x_ridge = np.reshape(x_train,(5,n,nXi+1))
y_ridge = np.reshape(y_train,(5,n,1))

# ridge regression test data sets
x1 = x_ridge[1:]
x2 = x_ridge[[0,2,3,4]]
x3 = x_ridge[[0,1,3,4]]
x4 = x_ridge[[0,1,2,4]]
x5 = x_ridge[:-1]

y1 = y_ridge[1:]
y2 = y_ridge[[0,2,3,4]]
y3 = y_ridge[[0,1,3,4]]
y4 = y_ridge[[0,1,2,4]]
y5 = y_ridge[:-1]

# cut lambda value by factor of 2 and find optimal lambda with minimum rmse rate
for attemp in range(10):

    for k in range(5):
        ridge_rmse = 0

        validation = x_ridge[k]
        validation = np.reshape(validation,(n,nXi+1))   #validation set

        y_validation = y_ridge[k]
        y_validation = np.reshape(y_validation,(n,1))   #corresponing y

        if k == 0:
            x = x1
            y = y1
        elif k==1:
            x = x2
            y = y2
        elif k==2:
            x = x3
            y = y3
        elif k==3:
            x = x4
            y = y4
        else:
            x = x5
            y = y5

        y = np.reshape(y,(nTraining-n,1))
        x = np.reshape(x,(nTraining-n,nXi+1))

        y_train = np.reshape(y_train,(nTraining,1))
        xtrans = x.transpose()

        w_ridge = calc_w(x,xtrans,y,lam)
        yi_ridge = np.dot(validation,w_ridge)

        # sum of all rmse for each lambda value
        ridge_rmse += calc_rmse(yi_ridge,y_validation,n)
```

```
    ridge_rmse/=5
    #print ridge_rmse

    #find the smallest rmse that leads to the best lambda value
    if ridge_rmse < min_rmse:
        min_rmse = ridge_rmse
        best_lambda = lam

    lam = float(lam/2)

# train the entire training data with the most optimal lambda
w_ridge = calc_w(x_train,x_traintrans,y_train,best_lambda)
yi_testridge = np.dot(x_test,w_ridge)
ridge_testrmse = calc_rmse(yi_testridge,y_test,nTest)

# gradient decent algorithm for linear regression
l_rate = 0.00001    #learning rate
ep = math.pow(10,-6) #converge criteria

w_t = np.zeros((96,1))
w_t = gradient_decent(w_t,l_rate,x_train,x_traintrans,y_train,ep,0)

#compute RMSE for training and test data
lrgd_trainyi = np.dot(x_train,w_t)
lrgd_testyi = np.dot(x_test,w_t)
lrgd_trainrmse = calc_rmse(lrgd_trainyi,y_train,nTraining)
lrgd_testrmse = calc_rmse(lrgd_testyi,y_test,nTest)

# compute RMSE for test data on ridge regression
ridge_wt = np.zeros((96,1))
ridge_wt = gradient_decent(ridge_wt,l_rate,x_test,x_testrans,y_test,ep,best_lambda)
rrgd_testyi = np.dot(x_test,ridge_wt)
rrgd_testrmse = calc_rmse(rrgd_testyi,y_test,nTest)

#print all important info for LR and RR analysis
print("\nx_training data\n**************************************")
print x_train

print("\nx_test data\n**************************************")
print x_test

print("\ncalculated w\n**************************************")
print w

print("\nLinear Regression RMSE\n**************************************")
print "Linear Regression RMSE train = ",train_rmse
print "Linear Regression RMSE test = ",test_rmse

print("\nK Fold Analysis on Ridge regression\n**************************************")
print "most optimal lambda = ",best_lambda
print "Ridge RMSE for test data: ",ridge_testrmse

print("\nGradient Decent linear Regression\n**************************************")
print "norm value of w_t vs w: ",np.linalg.norm(w_t - w)
print "linear regression gradient decent RMSE Train: ",lrgd_trainrmse
print "linear regression gradient decent RMSE Test: ",lrgd_testrmse
```

```
print("\nGradient Decent Ridge Regression\n**********************************")
print "norm value of w_t vs w: ",np.linalg.norm(ridge_wt - w)
print "ridge regression gradient decent RMSE Test: ",rrgd_testrmse
```

```
x_training data
**********************************
[[-0.45 -1.85 -1.06 ...,  1.26 -0.39  1.  ]
 [-0.45 -0.27 -0.22 ..., -0.62 -0.39  1.  ]
 [-0.14  1.87  0.55 ...,  0.52 -0.39  1.  ]
 ...,
 [ 0.81 -0.57 -0.48 ...,  0.08  3.4   1.  ]
 [ 0.18  0.28  1.   ...,  0.73  0.52  1.  ]
 [ 1.12  1.93  0.49 ..., -0.49  3.77  1.  ]]

x_test data
**********************************
[[-0.14  0.35 -0.41 ...,  0.65 -0.39  1.  ]
 [ 0.02 -0.45 -0.22 ..., -0.66 -0.39  1.  ]
 [-0.45  0.28 -0.16 ..., -0.66 -0.39  1.  ]
 ...,
 [-0.38  1.99  1.07 ..., -0.57 -0.39  1.  ]
 [-0.38  0.04 -0.22 ..., -0.27 -0.39  1.  ]
 [ 0.02 -0.57 -0.48 ..., -0.14 -0.39  1.  ]]
```

```
calculated w
**********************************
[[ -1.54906138e-02]
 [  3.35459838e-03]
 [  1.27547102e-02]
 [ -2.99835603e-02]
 [ -2.49633935e-02]
 [  2.42430772e-02]
 [ -1.26473288e-02]
 [  2.03432879e-02]
 [ -3.20098178e-02]
 [ -3.13299628e-02]
 [  6.01490622e-03]
 [ -3.97817625e-02]
 [  4.88968626e-03]
 [ -2.59111673e-03]
 [ -1.25557009e-02]
 [  5.74520977e-02]
 [ -5.57639378e-02]
 [  2.37527507e-03]
 [ -3.68181254e-04]
 [ -4.24706174e-03]
 [  4.31644481e-03]
 [  1.27018447e-02]
 [  2.54547492e-02]
 [ -3.32700396e-02]
 [ -1.60002475e-02]
 [  3.54599959e-03]
 [  1.54036062e-02]
 [  4.24360129e-03]
 [  4.25444378e-02]
 [ -1.03008495e-02]
 [  3.77526976e-03]
 [  1.37145404e-02]
 [  1.56911065e-02]
 [  6.22095883e-02]
 [  3.59684662e-02]
 [  1.04463505e-02]
 [ -6.95109507e-02]
 [  1.25702313e-02]
 [  4.52392413e-02]
 [ -1.27783374e-01]
 [ -3.11803197e-03]
 [ -1.58893093e-04]
 [  8.76703186e-03]
 [ -2.98193646e-02]
```

```
Linear Regression RMSE
**********************************
Linear Regression RMSE train =  0.127689674218
Linear Regression RMSE test =  0.145834644909

K Fold Analysis on Ridge regression
**********************************
most optimal lambda =  3.125
minimum RSME =  0.180718999106
Ridge RMSE for test data:  0.147656984685

Gradient Decent linear Regression
**********************************
norm value of w_t vs w:  0.202242440576
linear regression gradient decent RMSE Train:  0.128257141313
linear regression gradient decent RMSE Test:  0.145804233988

Gradient Decent Ridge Regression
**********************************
norm value of w_t vs w:  0.319443900262
ridge regression gradient decent RMSE Test:  0.145344019013
Tracys-MacBook-Pro:Desktop trace$
```

Conclusions:

Linear regression assumes a linear relationship between the input X and the output Y variable. More specifically, the output Y can be calculated from a linear combination of the input. In this assignment, we can use data set on the training data to estimate the coefficients ie. weights, required by the model to make predictions on the new data.

After performing closed form solution for linear regression using calculated $W = (X*X^T)^{-1} X^T Y$ to predict Yi:

RMSE for training data = 0.128
RMSE for test data = 0.146

Performing ridge regression by using K fold analysis when k = 5. Starting lambda=400 and continuing decrease lambda by factor of 2. For each lambda value, we calculate the average of all 5 RMSE values for each validation data set, then conclude the most optimal lambda value for the minimum average RMSE. The algorithm gives the most optimal lambda = 3.125 when its average RMSE is about 0.18.
Then we train the entire test data set with the most optimal lambda value by performing ridge regression closed-form solution to calculate $w\_ridge = (X*X^T + lambda*I)^{-1} X^T Y$. Finally, we use w_ridge to get

RMSE for test data = 0.148

Using the gradient descent algorithm on linear regression with the converging criteria that w(t+1) - w(t) is less than 10^-5. After find w(t) that converges and if it's the correct result, RMSE for training data and test data should be very close or equal to RMSE calculated in part 1 because w_t should be close or equal to w. In order to make the results even more precise, I changed the converging criteria to be 10^-6. From the output above, we can conclude that both RMSEs are very close to part1 RMSE results.

Same reasoning applies to the gradient descent on ridge regression except involving an additional factor, the most optimal lambda value calculated in part2, into the equation. And we can also see that RMSE for the test data is very close to RMSE test in part 2.