assignment_vishal_kumar_chaudhary

August 29, 2017

```
In [16]: ###########################################################
         ##########     PART 1     #################################
         ###########################################################

         import numpy as np
         import numpy as np
         def LeastSquare(x, Y):
             x= np.matrix(x)
             inverse_ = np.linalg.inv(np.matmul(np.transpose(x),x))
             #print(inverse_)
             z =  np.matmul(inverse_,np.transpose(x))
             q = np.matmul(z,np.transpose(y))
             return q

         data =[[3,1],[1,2]]
         y = [9,8]
         print(LeastSquare(data , y))




         #####################################################################################
```

```
[[ 2.  3.]]
```

```
In [10]: import numpy as np
         def ridgeRegression(x , y ,lamb):
             #print(np.shape(x))
             xtx = np.matmul(np.transpose(x),x)
             #print('""""""""""""""""""""""""""""""""""""""""""""""""""')
             #print(xtx)
             #print("lambda os **********************************************",lamb)
             #print(y)
             #print('""""""""""""""""""""""""""""""""""""""""""""""""""')
             n , m = np.shape(xtx)
             lamb_I = np.multiply(lamb,np.identity(n))
```

```
    xtx_add_lamb_I =np.add(xtx ,lamb * np.identity(n))
    inverse_ = np.linalg.inv(xtx_add_lamb_I)

    #print(np.shape(inverse_))
    #print(np.shape(np.transpose(x)))
    #print(np.shape(y))
    z=  np.matmul(np.matmul(inverse_,np.transpose(x)),y)
    #print(z)

   # print("*******************")
    return z
```

```
In [21]: #######################################################################################
         ###################   gradient descent ################################################

         import numpy as np
         x_data_ = np.genfromtxt("x.txt",delimiter=',')
         y_data_ = np.genfromtxt("y.txt",delimiter=',')

         stepsize=0.06
         w_initial=[0  for i in range(11)]
         #print(w_initial)
         eps=0.00001
         dif=10
         print("hello")
         matrix = []
         for each_x in x_data_ :
             matrix.append(feature_matrix(each_x))
         while(dif > eps):


             delta =[0  for i in range(11)]
             count=0

             for i in matrix :
                 delta =delta + i*(np.subtract(np.matmul(np.transpose(w_initial),i),y_data_[coun
                 count = count+1
             decent = 0
             for  i in delta :
                 decent =decent + i**2

             dif =decent

             #print(decent)
             w_initial = w_initial - stepsize * delta

         print("w is" ,w_initial)
             #print(delta)
             #break
```

```
hello
w is [ 0.01608493  0.06556429  0.0035093   0.55101668 -0.2588549   0.49423214
 -0.17545708  0.11544585  0.07729444 -0.26122431  0.35033915]
```

```
In [22]: def feature_matrix(x):
             vector = []
             k =1
             vector.append(k)
             for i in range(10):
                 k=k*x
                 vector.append(k)
             vector = np.array(vector)
             #print("vector is====>>>>",vector)
             return vector
```

```
In [23]: def find_error(param , x ,y ) :

             #print("************************",x, y  )

             x_matrix = []
             for element in x :
                     x_matrix.append(feature_matrix(element))
             z = np.subtract(np.matmul(x_matrix,np.transpose(param)),y)
             error = 0
             for i in z :
                 error += i**2

             #print (error )
             return error/len(x)
```

```
In [24]: import numpy as np
         def optimal_hyperparam(x_data_,y_data_  ):

             ###################################################################################
             #################
             #x_data_  =[1,2,3,4,5]
             #y_data_  =[1,2,1,2,1]

             #################
             ###################################################################################
             #numpy.random.shuffle(x_data_)

             list_=[]
```

```
    list_x = np.array_split(x_data_,5)
    #print(list_x)
    list_y = np.array_split(y_data_,5)
    global_error = 100000000000000000
    lambda_ =0

    for i in range(-12,11) :
        local_error_1_fold= 0

        for j in range(5) :
            list_x_train = []
            list_y_train = []
            for k in range(5) :
                for l in range(5) :
                    if(k!=l):
                         #print(type(list_x[3]))
                        list_x_train = np.concatenate([list_x_train , list_x[l] ])
                        list_y_train = np.concatenate([list_y_train ,list_y[l]])
                x_matrix =[]
                #print(list_x_train)
                #print(np.shape(list_x_train))
                for element in list_x_train :
                    x_matrix.append(feature_matrix(element))
                #print("***********",x_matrix)
                    #x_matrix = np.append( feature_matrix(element))

                parameter = ridgeRegression(x_matrix , list_y_train,2**i)
                #print(list_x[k],list_y[k])
                local_error_1_fold +=  find_error(parameter ,list_x[k] , list_y[k])
        #print(local_error_1_fold)

        if(local_error_1_fold <= global_error ):
            global_error = local_error_1_fold
            lambda_ = i

    return  2**lambda_




print("hello hi there , optimum hyperparameter ")
x_data_ = np.genfromtxt("x.txt",delimiter=',')
y_data_ = np.genfromtxt("y.txt",delimiter=',')

print("************>>>",optimal_hyperparam(x_data_,y_data_  ))
```

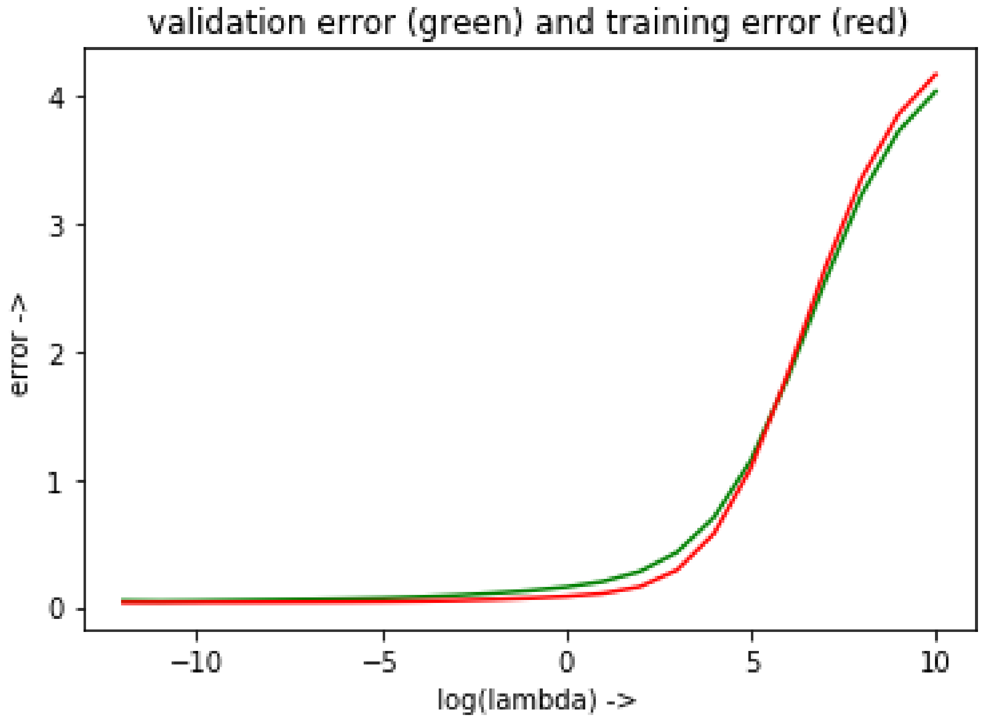
```
hello hi there , optimum hyperparameter
************>>> 0.00048828125
```

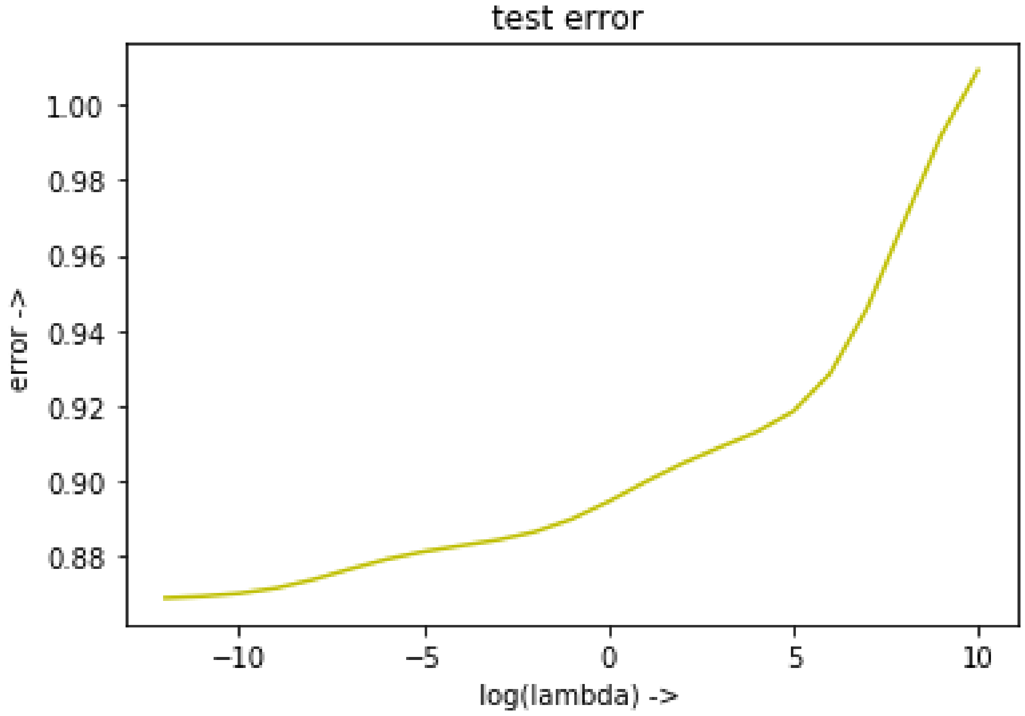
```
In [25]: import matplotlib.pyplot as plt
         import numpy as np
         def error_in_regress(x_data_,y_data_ , lambda_ ):

             ###################################################################################
             #################
             #x_data_   =[1,2,3,4,5]
             #y_data_   =[1,2,1,2,1]

             #################
             ###################################################################################
             #numpy.random.shuffle(x_data_)

             list_=[]
             list_x = np.array_split(x_data_,5)
             #print(list_x)
             list_y = np.array_split(y_data_,5)




             validation_error = 0
             training_error = 0
             for j in range(5) :
                 list_x_train = []
                 list_y_train = []
                 for k in range(5) :
                     for l in range(5) :
                         if(k!=l):
                              #print(type(list_x[3]))
                             list_x_train = np.concatenate([list_x_train , list_x[l] ])
                             list_y_train = np.concatenate([list_y_train ,list_y[l]])
                     x_matrix =[]
                     #print(list_x_train)
                     #print(np.shape(list_x_train))
                     for element in list_x_train :
                         x_matrix.append(feature_matrix(element))
                     #print("***********",x_matrix)
                         #x_matrix = np.append( feature_matrix(element))

                     parameter = ridgeRegression(x_matrix , list_y_train,lambda_)
                     #print(list_x[k],list_y[k])


                     validation_error +=  find_error(parameter ,list_x[k] , list_y[k])
                     training_error +=  find_error(parameter ,list_x_train , list_y_train)
```

```
    #print(training_error)


    return  (validation_error ,training_error )



x_data_ = np.genfromtxt("x.txt",delimiter=',')
y_data_ = np.genfromtxt("y.txt",delimiter=',')
print("hello hi there , optimum hyperparameter ",optimal_hyperparam(x_data_,y_data_  ))

#print("***********>>>",error_in_regress(x_data_,y_data_  ))


validation_error =[]
training_error = []
log_lambda = []
#error.append(error_in_regress(x_data_ ,y_data_,10 ))
for i in range(-12,11):
    (x,y) = error_in_regress(x_data_ ,y_data_,2**i)

    validation_error.append(x )
    training_error.append(y )
    log_lambda.append(i)


plt.xlabel("  log(lambda) ->")
plt.ylabel("error ->")
plt.title("validation error (green) and training error (red)")



plt.plot(log_lambda,validation_error,'g')
plt.plot(log_lambda,training_error , 'r')
plt.show()


#####################################################################3

############ finding test error with changing hyper parameter

#######################################################################


x_data_ = np.genfromtxt("xts.txt",delimiter=',')
y_data_ = np.genfromtxt("yts.txt",delimiter=',')
```

```
x_matrix=[]
test_error =[]

#generating feature matrix
for element in  x_data_ :
    x_matrix.append(feature_matrix(element))

#for every lambda finding error for test_data
for i in range(-12 , 11):
    param = ridgeRegression(x_matrix , y_data_ ,2**i)
    test_error.append(find_error(param ,x_data_ , y_data_ ))


plt.xlabel("  log(lambda) ->")
plt.ylabel("error ->")
plt.title("test error ")

plt.plot(log_lambda,test_error , 'y')

plt.show()
```

```
hello hi there , optimum hyperparameter  0.00048828125
```


validation error (green) and training error (red)
error ->
0
1
2
3
4
−10
−5
0
5
10
log(lambda) ->


test error
error ->
0.88
0.90
0.92
0.94
0.96
0.98
1.00
−10
−5
0
5
10
log(lambda) ->