

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **李政**

**Student ID 201530612064**

**E-mail li416570803@qq.com**

**Tutor**   **MingkuiTan**

**Date submitted** **2017. 12 . 6**

**1. Topic:** linear regression and linear classification

**2. Time: 17.12.6**

**3. Reporter:李政**

**4. Purposes:**

1) Understand how linear regression and linear classification work

2)And the experiment is done in a small data set

3)experience the process of optimization

**5. Data sets and data analysis:**

Linear regression use data sets Housing\_scale.txt and classification use Australian\_scale.txt

**6. Experimental steps:**

1) read the data set

2) divide data into train sets and test sets

3)model parameter W initialize

4)choose the loss function and get its derivative

5)get the gradient of all samples about this loss function

6)take the negative direction of the gradient, set as D

7)update the parameter using W = W +eta\*D, which eta is learning rate

8) test the model in train sets and validate in test sets

**7. Code:**

(Fill in the contents of 8-12 respectively for linear regression and linear classification)

Linear regression:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import load\_svmlight\_file

# linear regression

X,y = load\_svmlight\_file('housing\_scale.txt')

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size = 0.25,random\_state = 5)

def lossfun(y,predict):

sum = 0

for i in np.arange(len(y)):

cost = (y[i]-predict[i])\*\*2

sum +=cost

sum = sum/(2\*len(y))

return sum

def linear\_regression(train,W):

prediction = []

for i in np.arange(train.shape[0]):

pred = (W.T).dot(train[i])

prediction.append(pred)

return prediction

def delta\_loss(train,actual,W):

loss\_delta = -(train.T).dot(actual) + ((train.T).dot(train)).dot(W)

return loss\_delta

# initial the W

def linear\_model(X\_train,y\_train,X\_test,y\_test,eta,iteration):

W = np.zeros(X\_train.shape[1])

X\_train = X\_train.toarray()

X\_test = X\_test.toarray()

test\_showdata = []

train\_showdata = []

for i in np.arange(iteration):

predict = linear\_regression(X\_train,W)

loss\_sum = lossfun(y\_train,predict)

print('iteration:'+str(i)+'\n'+'loss : '+str(loss\_sum))

test\_pred = linear\_regression(X\_test,W)

test\_loss\_sum = lossfun(y\_test,test\_pred)

print('\n'+'test loss : '+ str(test\_loss\_sum))

D = np.zeros(X\_train.shape[1])

for j in np.arange(X\_train.shape[1]):

all = 0

for k in np.arange(len(y\_train)):

one = (predict[k] - y\_train[k])\*X\_train[k][j]

all +=one

all =eta\*all

D[j] = all

W = W- D

test\_showdata.append(test\_loss\_sum)

train\_showdata.append(loss\_sum)

fig = plt.figure(figsize = (15,7))

ax = fig.add\_subplot(111)

ax.set\_title('loss\_sum')

ax.set\_xlabel('iteration')

ax.set\_ylabel('loss sum')

plt.plot(test\_showdata,c='b',linewidth = 1,label = 'test\_loss')

plt.plot(train\_showdata,c = 'g',linewidth = 1,label = 'train\_loss' )

plt.legend(loc = 'best')

linear\_model(X\_train,y\_train,X\_test,y\_test,0.05,500)

linear classification:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import load\_svmlight\_file

X,y = load\_svmlight\_file('australian\_scale.txt')

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size = 0.25,random\_state = 5)

def loss\_fun(y,predict):

sum = 0

for i in np.arange(len(y)):

cost = max(0,1-y[i]\*predict[i])

sum+= cost

return sum

def linear\_classify(train,W,b):

prediction = np.zeros(train.shape[0])

for i in np.arange(train.shape[0]):

pred = (W.T).dot(train[i])+b

prediction[i] = pred

return prediction

def classify\_model(X\_train,y\_train,X\_test,y\_test,eta,iteration,C,b):

W = np.zeros(X\_train.shape[1])

X\_train =X\_train.toarray()

X\_test = X\_test.toarray()

test\_showdata = []

train\_showdata =[]

for i in np.arange(iteration):

predict = linear\_classify(X\_train,W,b)

prediction = np.zeros(X\_train.shape[0])

for j in np.arange(len(predict)):

if predict[j]>=0:

prediction[j] = 1

else:

prediction[j] = -1

loss\_sum = loss\_fun(y\_train,predict)

wrong = 0

for j in np.arange(len(prediction)):

if prediction[j] !=y\_train[j]:

wrong+=1

#print('wrong rate: ' +str(float(wrong)/len(prediction)))

test\_predict = linear\_classify(X\_test,W,b)

t\_prediction = np.zeros(X\_test.shape[0])

for j in np.arange(len(test\_predict)):

if test\_predict[j]>=0:

t\_prediction[j] = 1

else:

t\_prediction[j] = -1

t\_wrong = 0

for j in np.arange(len(t\_prediction)):

if t\_prediction[j] != y\_test[j]:

t\_wrong+=1

#print('test wrong rate: ' + str(float(t\_wrong)/len(t\_prediction)))

D =np.zeros(X\_train.shape[1])

for j in np.arange(X\_train.shape[0]):

one = np.zeros(X\_train.shape[1])

if (1-y\_train[j]\*( (W.T).dot(X\_train[j])))>=0:

for k in np.arange(len(X\_train[j])):

one[k] = y\_train[j]\*X\_train[j][k]

for k in np.arange(len(D)):

D[k] += -C\*one[k]

D= W +D

W = W-eta\*D

test\_showdata.append(float(t\_wrong)/len(t\_prediction))

train\_showdata.append(float(wrong)/len(prediction))

fig = plt.figure(figsize = (15,7))

ax = fig.add\_subplot(111)

ax.set\_title('wrong\_rate')

ax.set\_xlabel('iteration')

ax.set\_ylabel('wrong\_rate')

plt.plot(test\_showdata,c='b',linewidth = 1,label = 'test\_wrong')

plt.plot(train\_showdata,c = 'g',linewidth = 1,label = 'train\_wrong' )

plt.legend(loc = 'best')

classify\_model(X\_train,y\_train,X\_test,y\_test,0.0003,100,1,0)

**8. Selection of validation (hold-out, cross-validation, k-folds cross-validation, etc.):** hold out

**9. The initialization method of model parameters:**

All zeros

**10. The selected loss function and its derivatives:**

Linear regression : mse

Linear classification: hinge loss

**11. Experimental results and curve:**

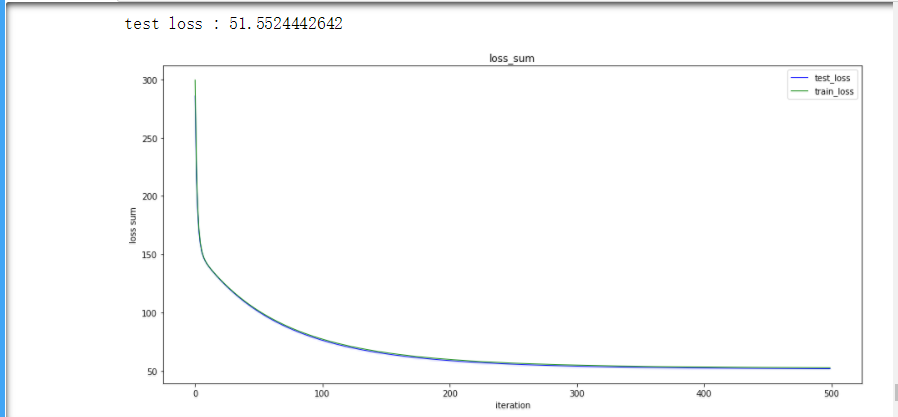
## Hyper-parameter selection (η, epoch, etc.):

## Assessment Results (based on selected validation):

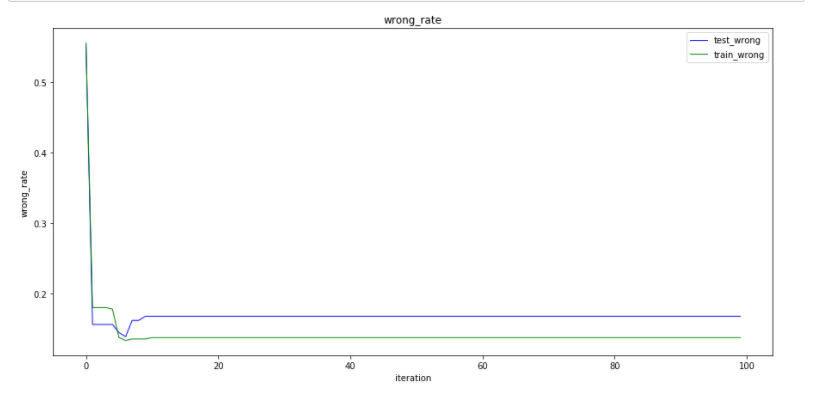
## Predicted Results (Best Results):

## Loss curve:

Linear regression:



Linear classification:



**12. Results analysis:** the result is not bad

**13. Similarities and differences between linear regression and linear classification:** Getting a good result, classification need less iterations than regression

**14. Summary: benefit a lot**