

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **Israr Ahmad**

**Student ID 201722800140**

**E-mail israrshmad@yahoo.com**

**Tutor**   **tangmingkui**

**Date submitted** **2017.12 .8**

1. **Topic:**

Linear Regression, Linear Classification and Gradient Descent

1. **Time:**

**12/8/2017**

1. **Reporter:**

**Israr Ahmad**

1. **Purposes:**

1、Further understand of linear regression and gradient descent.

2、Conduct some experiments under small scale dataset.

3、Realize the process of optimization and adjusting parameters.

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

Linear Regression uses Housing in LIBSVM Data, including 506 samples and each sample has 13 features. You are expected to download scaled edition. After downloading, you are supposed to divide it into training set, validation set.

Linear classification uses australian in LIBSVM Data, including 690 samples and each sample has 14 features. You are expected to download scaled edition. After downloading, you are supposed to divide it into training set, validation set.

1. **Experimental steps:**

The experimental code and drawing are completed on jupyter.

（1）Linear Regression and Gradient Descent

1、Load the experiment data. You can use load\_svmlight\_file function in sklearn library.

2、Devide dataset. You should divide dataset into training set and validation set using train\_test\_split function. Test set is not required in this experiment.

3、Initialize linear model parameters. You can choose to set all parameter into zero, initialize it randomly or with normal distribution.

4、Choose loss function and derivation: Find more detail in PPT.

5、Calculate gradient toward loss function from all samples.

6、Denote the opposite direction of gradient as .

7、Update model: . is learning rate, a hyper-parameter that we can adjust.

8、Get the loss under the training set and by validating under validation set.

9、Repeate step 5 to 8 for several times, and drawing graph of as well as with the number of iterations.

（2）Linear Classification and Gradient Descent

1、Load the experiment data.

2、Divide dataset into training set and validation set.

3、Initialize SVM model parameters. You can choose to set all parameter into zero, initialize it randomly or with normal distribution.

4、Choose loss function and derivation: Find more detail in PPT.

5、Calculate gradient toward loss function from all samples.

6、Denote the opposite direction of gradient as .

7、Update model: . is learning rate, a hyper-parameter that we can adjust.

8、Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Get the loss under the trainin set and by validating under validation set.

9、Repeate step 5 to 8 for several times, and drawing graph of as well as with the number of iterations.

Finishing experiment report according to result: The template of report can be found in example repository.

**7. Code:**

（1）linear regression

**from** sklearn.datasets **import** load\_svmlight\_file  
**from** sklearn.model\_selection **import** train\_test\_split  
**import** matplotlib.pyplot **as** plt  
  
alpha = 0.001  
iteration = 100  
accuracy = 0.001  
  
m = 506  
m\_train = 203  
d\_test = 203  
feature = 13  
theta = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  
  
iter\_num = [1] \* iteration  
loss\_train = [1] \* iteration  
loss\_test = [1] \* iteration  
  
*#read Dataset***def** get\_data():  
 data = load\_svmlight\_file(**r"C:\Users\israr\Desktop\hotmodel\DATA\housing\_scale.txt"**, n\_features=13)  
 **return** data[0], data[1]  
  
X, y = get\_data()  
X = X.toarray()  
*#divide dataset into train dataset and test dataset*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=43)  
print(X\_train)  
print(y\_train)  
  
*#hypothesis***def** hypothesis(x):  
 result = theta[0]  
 **for** i **in** range(0, feature):  
 result = result + theta[i + 1] \* x[i]  
 **return** result  
  
*#lossfunction***def** loss(m, X, y):  
 sum = 0  
 **for** i **in** range(0, m):  
 sum = sum + (hypothesis(X[i]) - y[i]) \*\* 2  
 sum = sum / (2 \* m)  
 **return** sum  
  
*#gradient***def** gradient(j, m, X, y):  
 sum = 0  
 **if** (j == 0):  
 **for** i **in** range(0, m):  
 sum = sum + (hypothesis(X[i]) - y[i])  
 **else**:  
 **for** i **in** range(0, m):  
 sum = sum + (hypothesis(X[i]) - y[i]) \* X[i][j - 1]  
 sum = sum / m  
 **return** sum  
  
*#train regression model***def** train():  
 **for** i **in** range(0, iteration):  
 **for** j **in** range(0, feature + 1):  
 theta[j] = theta[j] - alpha \* gradient(j, m\_train, X\_train, y\_train)  
 iter\_num[i] = i;  
 loss\_train[i] = loss(m\_train, X\_train, y\_train);  
 loss\_test[i] = loss(d\_test, X\_test, y\_test);  
  
train()  
  
fig, ax = plt.subplots()  
ax.plot(iter\_num, loss\_train, color=**'m'**, label=**'loss of train'**)  
ax.plot(iter\_num, loss\_test, color=**'c'**, label=**'loss of test'**)  
  
ax.set\_xlabel(**'Iteration times'**)  
ax.set\_ylabel(**'loss'**)  
plt.xticks(iter\_num, rotation=0)  
plt.show()

（2）linear classification

**from** sklearn.datasets **import** load\_svmlight\_file  
**from** sklearn.cross\_validation **import** train\_test\_split  
**from** numpy **import** \*  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
  
filename = **"australian\_scale.txt"**learning\_rate = 1 *#learning rate*num\_iter = 100 *#iteration*k=50 *#batch  
  
#train model***def** optimizer(data1,label1,data2,label2,starting\_w):  
  
 w = starting\_w  
 Loss\_train = []  
 Loss\_valid = []  
  
 **for** i **in** range(num\_iter):  
 eta = 1.0 / (learning\_rate \* (i + 1)) *# step* w = compute\_gradient( w, data1, label1,eta,i)  
 **if** i%10==0:  
 loss = compute\_error(w, data1, label1)  
 loss\_valid=compute\_error(w,data2,label2)  
 print(**'iter {0}:train\_error={1}, valid\_error={2}'**.format \  
 (i, float(loss), float(loss\_valid)))  
 Loss\_train.append(float(loss))  
 Loss\_valid.append(float(loss\_valid))  
  
 loss\_final = compute\_error(w, data1, label1)  
 loss\_final2 = compute\_error(w, data2, label2)  
 Loss\_train.append(loss\_final)  
 Loss\_valid.append(loss\_final2)  
 **return** [w,Loss\_train,Loss\_valid]  
  
*#gradient***def** compute\_gradient(w\_current,data,label, eta, iter\_count):  
  
 N,M = shape(data)  
 w\_gradient=(np.zeros((1, M)))  
 dataIndex = list(range(N))  
 random.shuffle(dataIndex) *#check* **for** j **in** range(N):  
 i=dataIndex[j]  
 v = data[i, :].T  
 p=predict(w\_current, v)  
 **if** label[i]\*p<1:  
 w\_gradient+=label[i]\*v.T  
 *#else:  
 # w\_gradient+=* new\_w = (1.0-1/(iter\_count+1))\*w\_current + (eta/k) \* w\_gradient  
 **return** new\_w  
  
**def** predict(w,v):  
  
 **return** w\*v  
  
*#loss function***def** compute\_error(w, data, label):  
  
 totalError = 0  
 w=w.A  
 count = data.shape[0]  
 **for** i **in** range(0, count):  
 v = data[i, :].T  
 p = predict(w, v)  
 **if** label[i]\*p<1:  
 totalError=totalError+(1-p)  
 **else**:  
 totalError+=0  
  
 error=learning\_rate\*0.5\*sum(w \*\* 2)+totalError/float(count)  
 **return** error  
  
*#divide into train dataset and test dataset***def** get\_data(filename):  
  
 *#read data* data = load\_svmlight\_file(filename)  
 X, y = data[0], data[1]  
 count=X.shape[0]  
 X=X.A  
 X=np.column\_stack((X, np.ones((count, 1))))  
 X=mat(X)  
 *#divide into train dataset and test dataset* train\_X, valid\_X, train\_y, valid\_y = train\_test\_split(X, y, train\_size=0.8, random\_state=0)  
  
 **return** (train\_X, valid\_X, train\_y, valid\_y)  
  
**def** plot\_loss(y1,y2):  
  
 plt.plot(range(11), array(y1))  
 plt.figure  
 plt.plot(range(11), array(y2))  
 plt.show()  
  
**def** linear\_classification():  
  
 *# parameter initialization* w=mat(np.zeros((1, 15)))  
  
 train\_X, valid\_X, train\_y, valid\_y = get\_data(filename)  
 [w, loss\_train, loss\_valid] = optimizer(train\_X, train\_y, valid\_X, valid\_y, w)  
 print(w)  
 plot\_loss(loss\_train, loss\_valid)  
  
**if** \_\_name\_\_ ==**'\_\_main\_\_'**:  
  
 linear\_classification()  
 print(**"end"**)

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

Using sklearn [load\_svmlight\_file](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html" \t "https://www.zybuluo.com/chenyaofo/note/_blank) read experimental training set and verification set.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=43)

1. **The initialization method of model parameters:**
2. linear regression

set all parameter into zero, initialize it randomly or with normal distribution.

1. linear classification

set all parameter into zero, initialize it randomly or with normal distribution.

1. **The selected loss function and its derivatives:**
2. linear regression

*#lossfunction***def** loss(m, X, y):  
 sum = 0  
 **for** i **in** range(0, m):  
 sum = sum + (hypothesis(X[i]) - y[i]) \*\* 2  
 sum = sum / (2 \* m)  
 **return** sum

1. linear classification

*#loss function***def** loss(w, data, label):  
 totalError = 0  
 w=w.A  
 count = data.shape[0]  
 **for** i **in** range(0, count):  
 v = data[i, :].T  
 p = predict(w, v)  
 **if** label[i]\*p<1:  
 totalError=totalError+(1-p)  
 **else**:  
 totalError+=0  
  
 error=learning\_rate\*0.5\*sum(w \*\* 2)+totalError/float(count)  
 **return** error

**11. Experimental results and curve:**

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

(1)linear regression

alpha = 0.001

(2)linear classification

alpha = 0.001

## Assessment Results (based on selected validation):

(1)linear regression

(2)linear classification

iter 0:train\_error=57.893330593537925, valid\_error=58.105194360321725

iter 10:train\_error=1.4714407698168488, valid\_error=1.425896564488078

iter 20:train\_error=0.7764015933804379, valid\_error=0.7686723116397204

iter 30:train\_error=0.7688194139672568, valid\_error=0.8052417632758357

iter 40:train\_error=0.7886324294374275, valid\_error=0.8150043828352052

iter 50:train\_error=0.7308634704535156, valid\_error=0.7829090940180339

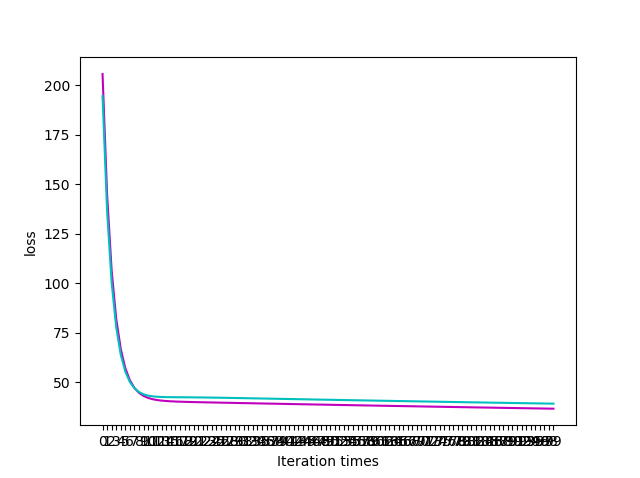
iter 60:train\_error=0.7033959064600617, valid\_error=0.7364732618166564

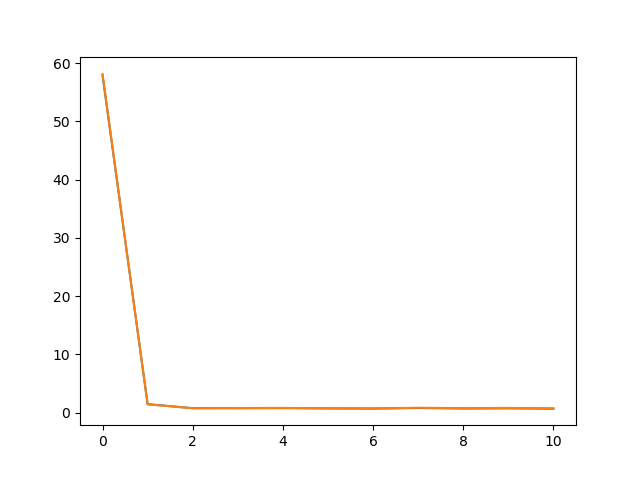
iter 70:train\_error=0.8098681467421865, valid\_error=0.8451277105086832

iter 80:train\_error=0.7194892422945491, valid\_error=0.7825685183238287

iter 90:train\_error=0.7488184073943814, valid\_error=0.8171066271025944

## Loss curve:





**12. Results analysis:**

**13. Similarities and differences between linear regression and linear classification:**

The loss function and derivation are different.

linear classification need to select the appropriate threshold, will verify the centralized calculation results greater than the threshold marked as positive, otherwise negative .

1. **Summary:**

I get familiar with linear regression and linear classification very well