

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**  严淑敏

**Student ID 201530613337**

**E-mail 1987516254@qq.com**

**Tutor**  谭明奎

**Date submitted** **2017.12.7**

**1. Topic:**

Linear Regression, Linear Classification and Gradient Descent

**2. Time:**

2017-12-02 9:00-12:00 AM B7-138

**3. Reporter:**

严淑敏

**4. Purposes:**

Further understand of linear regression and gradient descent.

Conduct some experiments under small scale dataset.

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. I use the scaled edition. I divide it into training set, validation set. The training set has 339 samples and the validation has 167 samples.  
 Linear classification uses australian in LIBSVM Data, including 690 samples and each sample has 14 features. I use the scaled edition. I divide it into training set, validation set. The training set has 462 samples and the validation has 228 samples.

**6. Experimental steps:**

**Linear Regression and Gradient Descent:**

1. Use use load\_svmlight\_file function in sklearn library to Load the experiment data.
2. Divide dataset into training set and validation using train\_test\_split function.
3. Initialize linear model parameters. I set all parameter into zero.
4. Choose loss function and derivation.
5. Calculate gradient  G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D .
7. Update model: . lamda is learning rate, a hyper-parameter that we can adjust.
8. Get the loss Ltrain under the training set and Lvalidation by validating under validation set.
9. Repeate step 5 to 8 for several times, and drawing graph Ltrain of  as well as  Lvalidation with the number of iterations.

**Linear Classification and Gradient Descent:**

1. Load the experiment data.
2. Divide dataset into training set and validation set.
3. Initialize SVM model parameters. I set all parameter into zero.
4. Choose loss function and derivation.
5. Calculate gradient G  toward loss function from all samples.
6. Denote the opposite direction of gradient G as D .
7. Update model: . .lamda 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  Ltrain under the trainin set and Lvalidation by validating under validation set.
9. Repeate step 5 to 8 for several times, and drawing graph of   Ltrain as well as  Lvalidation with the number of iterations.

**7. Code:**

**Linear Regression and Gradient Descent:**

from sklearn.datasets import load\_svmlight\_file

from numpy import \*

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

data=load\_svmlight\_file(r'D:\housing\_scale.txt')#读取数据

X=data[0]

y=data[1]

S=zeros((506,13))

S=mat(S)

X=X+S

X=X.getA()

c=ones(X.shape[0])

X=insert(X, 0, values=c, axis=1)#添加一列为1的值，那么后面的W0就表示b

X=mat(X)

X\_train, X\_validation, y\_train, y\_validation = train\_test\_split(X, y, test\_size=0.33, random\_state=42)#划分训练集和验证集

y\_train=y\_train.reshape(339,1)

y\_train=mat(y\_train)

y\_validation=y\_validation.reshape(167,1)

y\_validation=mat(y\_validation)

W=array([0,0,0,0,0,0,0,0,0,0,0,0,0,0])#将参数全部初始化为0

W=W.reshape(14,1)

W=mat(W)

lamda=0.001#lamda为学习率

k=0

Loss\_train=[]#存储在训练集上的损失

Loss\_validation=[]#存储在验证集上的损失

i=[]#存储迭代次数

while 1:

i.append(k)

Ltrain=0.5\*(1/339)\*(y\_train-X\_train\*W).transpose()\*(y\_train-X\_train\*W)#训练集的loss函数

Lvalidation=0.5\*(1/167)\*(y\_validation-X\_validation\*W).transpose()\*(y\_validation-X\_validation\*W)#测试集的loss函数

Lvalidation=Lvalidation.getA()

Ltrain=Ltrain.getA()

#print(Ltrain[0][0])

Loss\_train.append(Ltrain[0][0])

Loss\_validation.append(Lvalidation[0][0])

G=(1/339)\*(-X\_train.transpose()\*y\_train+X\_train.transpose()\*X\_train\*W)#G为loss函数的W偏导

D=-G#D为G的负方向

if Loss\_train[k]<0.01 or k>1000:#loss小于0.01或者迭代次数大于1000时停止

break

else:

W=W+lamda\*D#更新模型参数

k=k+1

%matplotlib inline

import matplotlib.pyplot as plt

plt.plot(i,Loss\_train,'b-',lw=1.5,label='Loss\_train')

plt.plot(i,Loss\_validation,'r-',lw=1.5,label='loss\_validation')

plt.legend(loc='upper right')

plt.xlabel('iteration')

plt.ylabel('loss')

**Linear Classification and Gradient Descent:**

from sklearn.datasets import load\_svmlight\_file

from numpy import \*

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

data=load\_svmlight\_file(r'D:\australian\_scale.txt')#读取数据集

X=data[0]

y=data[1]

S=zeros((690,14))

S=mat(S)

X=X+S

X=X.getA()

c=ones(X.shape[0])#插入一列1，这样参数b就相当与W0

X=insert(X, 0, values=c, axis=1)

X=mat(X)

X\_train, X\_validation, y\_train, y\_validation = train\_test\_split(X, y, test\_size=0.33, random\_state=42)#切分数据集

y\_train=y\_train.reshape(462,1)

y\_train=mat(y\_train)

y\_validation=y\_validation.reshape(228,1)

y\_validation=mat(y\_validation)

W=array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0])#模型参数初始化

W=W.reshape(1,15)

W=mat(W)

lamda=0.00001#lamda为学习率

i=0

S=array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0])#存储G

S=S.reshape(1,15)

S=mat(S)

k=0

m=0

loss\_train=[]#存储训练集的loss

loss\_validation=[]#存储验证集的loss

ix1=[]#存储迭代次数

while 1:

ix1.append(k)

loss=0

lossv=0

#计算训练集的loss值

for j in range(462):

if y\_train[j]\*(X\_train[j]\*W.transpose())<1:

loss=loss+1

loss\_train.append(loss)

#计算验证集的loss值

for j in range(228):

if y\_validation[j]\*(X\_validation[j]\*W.transpose())<1:

lossv=lossv+1

loss\_validation.append(lossv)

for i in range(462):

fx=1-y\_train[i]\*(X\_train[i]\*W.transpose())

if fx>=0:

G=-0.1\*y\_train[i]\*X\_train[i]

else:

G=0

S=S+G

S=(S+W)#S是求出的梯度

D=-S#D取梯度的反方向

W=W+lamda\*D#更新模型参数

if loss<0.01 or k>500:#若loss值小于0.01或者迭代次数大于500则停止

break

k=k+1

%matplotlib inline

import matplotlib.pyplot as plt

plt.plot(ix1,loss\_train,'b-',lw=1.5,label='loss\_train')

plt.plot(ix1,loss\_validation,'r-',lw=1.5,label='loss\_validation')

plt.legend(loc='upper right')

plt.xlabel('iteration')

plt.ylabel('loss')

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

**Linear Regression and Gradient Descent:**

hold-out

**Linear Classification and Gradient Descent:**

hold-out

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

**Linear Regression and Gradient Descent:**

set all parameter into zero.

**Linear Classification and Gradient Descent:**

set all parameter into zero.

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

**Linear Regression and Gradient Descent:**

Loss function:

Derivative:

**Linear Classification and Gradient Descent:**

Loss function:

Optimization:

Derivative:

**11. Experimental results and curve:**

**Linear Regression and Gradient Descent:**

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

η=0.001

epoch=1000

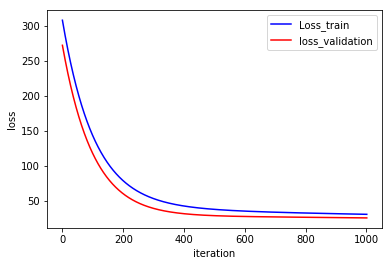
## Assessment Results (based on selected validation):

Loss\_train=30.32

## Predicted Results (Best Results):

Loss\_validation=25.18

## Loss curve:



**Linear Classification and Gradient Descent:**

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

η= 0.00001

epoch=500

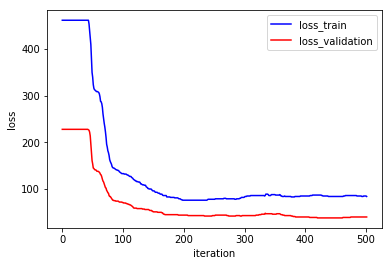
## Assessment Results (based on selected validation):

Loss\_train=84

## Predicted Results (Best Results):

Loss\_validation=40

## Loss curve:



**12. Results analysis:**

**Linear Regression and Gradient Descent:**

At the beginning, the value of the loss of the training set is very big. With the increase of the number of iterations, loss value is more and more small, until finally no longer change. loss of the validation set also showed the same trend as the training set. The regression curve is more and more close to the real.

**Linear Classification and Gradient Descent:**

At the beginning, the value of the loss of the trainning set is very big and the classification accuracy rate is very low. With the increase of the number of iterations, loss value is more and more small, until finally no longer change. The loss of validation set also showed the same trend as the training set. The classification accuracy is higher and higher.

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

Linear regression and linear classification are all about finding f(x)=WX+ b to predict the real value, and their steps are similar.

The linear regression’s prediction y corresponds to a lot of values, but the prediction y of the linear classification corresponds to only -1 and +1.

**14. Summary:**

Through this experiment, I learned a lot and applied the knowledge in the textbook to practice. I am familiar with the principle of linear regression and linear classification, familiar with the process of gradient descent, and experience the process of optimizing and tuning parameters.