

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**  **郑孟丹**

**Student ID 201530613795**

**E-mail 1228804246@qq.com**

**Tutor**   **谭明奎**

**Date submitted** **2017. 12 .5**

**1. Topic:** Linear Regression, Linear Classification and Gradient Descent

**2. Time:** 2017-12-02 9:00-12:00 AM

**3. Reporter:** 郑孟丹

**4. 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.

Linear classification uses australian in LIBSVM Data, including 690 samples and each sample has 14 features.

**6. Experimental steps:** Linear Regression and Gradient Descent

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

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

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

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

⑤Calculate gradient G toward loss function from all samples.

⑥Denote the opposite direction of gradient G as D

⑦Update model:w\_t=w\_t-1 - η \* D ,η is learning rate, a hyper-parameter that we can adjust.

⑧Get the loss L\_train under the training set and L\_validation by validating under validation set.

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

Linear Classification and Gradient Descent

①Load the experiment data.

②Divide dataset into training set and validation set.

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

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

⑤Calculate gradient G toward loss function from all samples.

⑥Denote the opposite direction of gradient G as D

⑦Update model: w\_t=w\_t-1 -η\*D, η is learning rate, a hyper-parameter that we can adjust.

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

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

**7. Code:  
1** Linear Regression and Gradient Descent  
  
*# coding: utf-8  
  
# In[1]:***from** sklearn.externals.joblib **import** Memory  
**from** sklearn.datasets **import** load\_svmlight\_file  
**from** sklearn **import** datasets  
**from** sklearn **import** model\_selection  
**import** numpy **as** np  
**import** matplotlib **as** mp  
**import** jupyter  
**import** matplotlib.pyplot **as** plt  
get\_ipython().magic(**'matplotlib inline'**)  
  
  
*# In[2]:*mem = Memory(**"./mycache"**)  
  
@mem.cache  
**def** get\_data(mysvmlightfile):  
 data = load\_svmlight\_file(mysvmlightfile)  
 **return** data[0], data[1]  
  
  
*# In[3]:  
  
#读取数据集*x\_data,y\_data=get\_data(**'housing\_scale.txt'**)  
  
  
*# In[4]:  
  
#扩展x矩阵一列全为1 使线性模型满足yi=w.T\*xi*x\_data=x\_data.todense()  
one=np.ones(x\_data.shape[0])  
x\_data=np.column\_stack((x\_data,one))  
  
  
*# In[5]:  
  
#划分训练集和测试集，20%的数据作为测试集*x\_train,x\_test,y\_train,y\_test=model\_selection.train\_test\_split(x\_data,y\_data,train\_size=0.8,test\_size=0.2,random\_state=0)  
  
  
*# In[6]:  
  
#调整数据维度*y\_data=np.mat(y\_data).T  
y\_train=np.mat(y\_train).T  
y\_test=np.mat(y\_test).T  
  
  
*# In[7]:  
  
#参数全零初始化*w=np.zeros(x\_train.shape[1])  
w=np.mat(w)  
w=w.T  
  
*#学习速率*rate=0.00029  
  
  
*# In[8]:  
  
#输出数据维度*print(x\_data.shape,y\_data.shape,w.shape,)  
print(x\_train.shape,y\_train.shape)  
print(x\_test.shape,y\_test.shape)  
  
  
*# In[9]:  
  
#循环次数*epoch=0  
  
  
*# In[10]:  
  
#存储循环次数，L\_train,L\_validation数组来画图*epoch\_array=[]  
L\_train\_array=[]  
L\_validation\_array=[]  
  
  
*# 线性模型为 yi=w.T\*xi 则损失函数为 LOSS=1/2\*(Y-X\*W).T\*(Y-X\*W),对w求梯度可得，梯度grad=X.T\*(X\*W-Y)  
  
# In[11]:  
  
#梯度下降，循环400次***while** epoch<400:  
 epoch=epoch+1  
   
 *#由梯度更新w，w=w-rate\*grad* w = w-rate \* (x\_train.T \* (x\_train \* w - y\_train))  
   
 *#计算L\_train，L\_validation* L\_train=((y\_train-x\_train\*w).T\*(y\_train-x\_train\*w))/2  
 L\_validation=((y\_test-x\_test\*w).T\*(y\_test-x\_test\*w))/2  
 L\_train = L\_train/x\_train.shape[0]  
 L\_validation =L\_validation/x\_test.shape[0]  
   
 *#输出当前的循环次数，L\_train,L\_validation* print(**"epoch:"**,epoch)  
 print(**"train\_loss:"**,L\_train)  
 print(**"validation loss:"**,L\_validation)  
   
 *#将数据存到数组中* epoch\_array.append(epoch)  
 L\_train\_array.append(np.array(L\_train)[0][0])  
 L\_validation\_array.append(np.array(L\_validation)[0][0])  
  
  
*# In[12]:  
  
#画图*plt.xlabel(**'epoch'**)  
plt.ylabel(**'loss'**)  
line1=plt.plot(epoch\_array,L\_train\_array)  
line2=plt.plot(epoch\_array,L\_validation\_array)  
label = [**"L\_train"**, **"L\_validation"**]  
plt.legend(label, loc = 0, ncol = 2)  
plt.show()

**8\_1. Selection of validation (hold-out, cross-validation, k-folds cross-validation, etc.):**hold-out

**9\_1. The initialization method of model parameters:**set all parameter into zero

**10\_1. The selected loss function and its derivatives:**Loss function =1/2\*(Y-X\*W).T\*(Y-X\*W),  
its derivatives : grad\_w=X.T\*(X\*W-Y)

**11\_1. Experimental results and curve:** Hyper-parameter selection (η, epoch, etc.):η=0.00029, epoch=400 Assessment Results (based on selected validation):  
①If the training data is larger, the model trained by the training data is similar to the data set training model, but the test data is too small and the evaluation result is accidental and inaccurate.

②If the training data is small and the test data is large, then the training data and dataset have different training models, and the evaluation of the test data has no significance.  
 Predicted Results (Best Results):  
L\_train, L\_validation curves tend to be stable, L\_train is less than L\_validation

## Loss curve:

**12\_1. Results analysis:**The actual result is approximate to the predict result

Linear Classification and Gradient Descent

*# coding: utf-8  
  
# In[36]:***from** sklearn.externals.joblib **import** Memory  
**from** sklearn.datasets **import** load\_svmlight\_file  
**from** sklearn **import** datasets  
**from** sklearn **import** model\_selection  
**import** numpy **as** np  
**import** matplotlib **as** mp  
**import** jupyter  
**import** matplotlib.pyplot **as** plt  
get\_ipython().magic(**'matplotlib inline'**)  
  
  
*# In[37]:*mem = Memory(**"./mycache"**)  
  
@mem.cache  
**def** get\_data(mysvmlightfile):  
 data = load\_svmlight\_file(mysvmlightfile)  
 **return** data[0], data[1]  
  
  
*# In[38]:  
  
#读取数据集*x\_data,y\_data=datasets.load\_svmlight\_file(**'australian\_scale.txt'**)  
  
  
*# In[39]:*x\_data=x\_data.todense()  
  
  
*# In[40]:  
  
#划分训练集和测试集，20%的数据作为测试集*x\_train,x\_test,y\_train,y\_test=model\_selection.train\_test\_split(x\_data,y\_data,train\_size=0.8,test\_size=0.2,random\_state=0)  
  
  
*# In[41]:  
  
#调整数据维度*y\_data=np.mat(y\_data).T  
y\_train=np.mat(y\_train).T  
y\_test=np.mat(y\_test).T  
  
  
*# In[42]:  
  
#参数全零初始化*w=np.zeros(x\_train.shape[1])  
w=np.mat(w)  
w=w.T  
b=0  
  
*#学习速率*rate=0.0005  
*#C*c=0.1  
  
  
*# In[43]:  
  
#输出数据维度*print(x\_data.shape,y\_data.shape,w.shape,)  
print(x\_train.shape,y\_train.shape)  
print(x\_test.shape,y\_test.shape)  
  
  
*# In[44]:  
  
#存储循环次数，L\_train,L\_validation数组来画图*epoch\_array=[]  
L\_train\_array=[]  
L\_validation\_array=[]  
  
  
*# In[45]:  
  
#循环次数*epoch=0  
  
  
*# 选用svm线性模型yi=w.T\*xi+b,loss function=1/2 \* w.T \* w + c\*max(0,1-yi\*(w.T\*xi+b))。  
# 对w求梯度:当1-yi\*(w.T\*xi+b)>=0时，grad\_w = w-c\*(yi\*xi).T,  
# 当1-yi\*(w.T\*xi+b)< 0时,grad\_w=w.  
# 对b求梯度，当1-yi\*(w.T\*xi+b)>=0时，grad\_b = -c\*yi,  
# 当1-yi\*(w.T\*xi+b)< 0时,grad\_b=0.  
  
# In[46]:***while** epoch<100:  
 epoch =epoch + 1  
 grad\_w =0  
 grad\_b=0  
   
 L\_train =0  
 L\_test =0  
   
 *#计算grad\_W , grad\_b* **for** i **in** range(x\_train.shape[0]):  
 **if**(1-(y\_train[i]\*(x\_train[i]\*w+b))>0):  
 grad\_w=grad\_w+w-c\*(y\_train[i]\*x\_train[i]).T  
 grad\_b=grad\_b-c\*y\_train[i]  
 **else**:  
 grad\_w=grad\_w+w  
   
 *#更新 w，b* w=w-rate\*grad\_w  
 b=b-rate\*grad\_b  
   
 *#计算L\_train,L\_validation* **for** i **in** range(x\_train.shape[0]):  
 **if**(1-y\_train[i]\*(x\_train[i]\*w+b)>0):  
 L\_train=L\_train+1-(y\_train[i]\*(x\_train[i]\*w+b))  
 **for** i **in** range(x\_test.shape[0]):  
 **if**(1-y\_test[i]\*(x\_test[i]\*w+b)>0):  
 L\_test=L\_test+(1-y\_test[i]\*(x\_test[i]\*w))  
 L\_train=0.5\*w.T\*w+c\*L\_train  
 L\_train=L\_train/x\_train.shape[0]  
 L\_test=0.5\*w.T\*w+c\*L\_test  
 L\_test=L\_test/x\_test.shape[0]  
   
 *#输出每一次迭代的loss值* print(**"epoch:"**,epoch)  
 print(**"train\_error:"**,L\_train)  
 print(**"validation error:"**,L\_test)  
   
 *#将数据存到数组中* epoch\_array.append(epoch)  
 L\_train\_array.append(np.array(L\_train)[0][0])  
 L\_validation\_array.append(np.array(L\_test)[0][0])  
  
  
*# In[47]:  
  
#画图*plt.xlabel(**'epoch'**)  
plt.ylabel(**'loss'**)  
line1=plt.plot(epoch\_array,L\_train\_array)  
line2=plt.plot(epoch\_array,L\_validation\_array)  
label = [**"L\_train"**, **"L\_validation"**]  
plt.legend(label, loc = 0, ncol = 2)  
plt.show()

**8\_2. Selection of validation (hold-out, cross-validation, k-folds cross-validation, etc.):**hold-out

**9\_2. The initialization method of model parameters:**set all parameter into zero

**10\_2. The selected loss function and its derivatives:**loss function=1/2 w.T w + cmax(0,1-yi(w.Txi+b))。   
for **its derivatives of w**:  
if 1-yi(w.Txi+b)>=0，grad\_w = w-c(yixi).T,   
if 1-yi(w.Txi+b)< 0， grad\_w=w.   
**its derivatives of b**，  
if 1-yi(w.Txi+b)>=0，grad\_b = -cyi,   
if 1-yi(w.T\*xi+b)< 0，grad\_b=0.

**11\_2. Experimental results and curve:** Hyper-parameter selection (η, epoch, etc.):η=0.0005, epoch=100 Assessment Results (based on selected validation):  
①If the training data is larger, the model trained by the training data is similar to the data set training model, but the test data is too small and the evaluation result is accidental and inaccurate.

②If the training data is small and the test data is large, then the training data and dataset have different training models, and the evaluation of the test data has no significance.  
 Predicted Results (Best Results):  
L\_train, L\_validation curves tend to be stable, L\_train is less than L\_validation

## Loss curve:

**12\_2. Results analysis:**

The actual result is approximate to the predict result

**13. Similarities and differences between linear regression and linear classification:** Overall, the two issues are essentially the same, that is, the fitting of the model. However, the y value (also known as label) of the classification problem is more discretized, and the same y value may correspond to a large number of x, which is of a certain range. The y value of the classification problem is usually a discrete point, while the y value of the regression problem is continuous.

Therefore, the classification problem is more that some x in a certain region corresponds to a y, while the model of regression problem is more inclined to x in a very small region, or generally x corresponds to a y.

**14. Summary:** I learned a lot from this experiment.such as calculate the loss function, find the gradient and how to achieve the code. This part contains how to read data, complement the data，using numpy to do matrix operations, etc., in particular,I has a deeper understanding of svm.