

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  曾浩

**学 号 201530613559**

**邮 箱 137315734@qq.com**

**指导教师**  **吴庆耀**

**提交日期** **2017年 12 月 6 日**

## 1. 实验题目: 线性回归、线性分类与梯度下降

## 2. 实验时间：2017年12 月 7 日

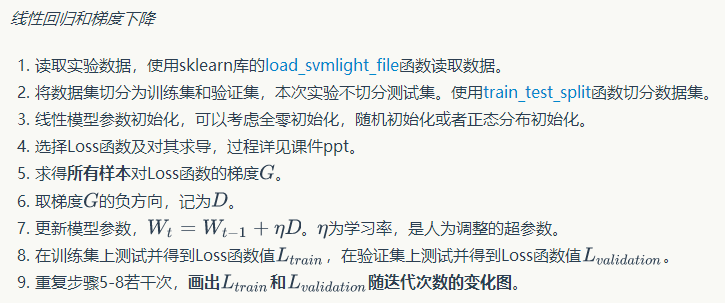
## 3. 报告人: 曾浩

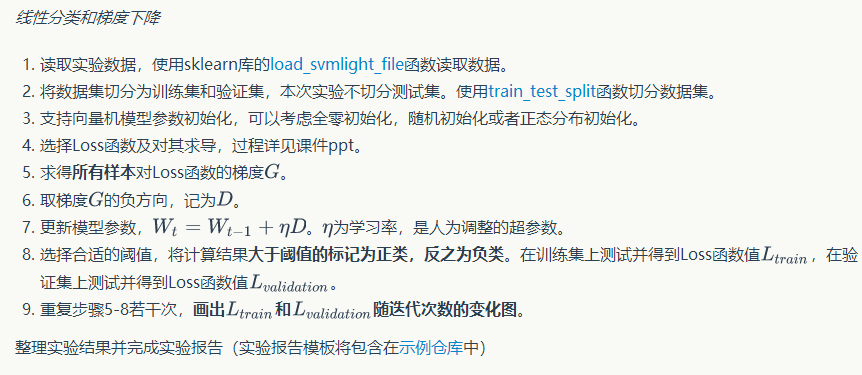
## 4. 实验目的:

1. 进一步理解线性回归和梯度下降的原理。
2. 在小规模数据集上实践。
3. 体会优化和调参的过程。

## 5. 数据集以及数据分析：

## 6. 实验步骤:





## 7. 代码内容:

线性回归代码：

import matplotlib.pylab as plt

from matplotlib import \*

import numpy as np

from numpy import \*

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

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

mem = Memory("./mycache")

@mem.cache

def get\_data():

data = load\_svmlight\_file("housing.txt")

return data[0], data[1]

#导入数据

X, y = get\_data()

# 修改y格式：数组变为矩阵

Y=mat(y).T

# 划分训练集以及验证集

X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split( X, Y, test\_size=0.22, random\_state=42)

# 修改X\_train格式

X1\_train=mat(np.column\_stack((X\_train.todense(),ones(X1\_train.shape[0]))))

# 修改X\_validation格式

X1\_validation=mat(np.column\_stack((X\_validation.todense(),ones(X1\_validation.shape[0]))))

# 选择Loss\_train 函数

def computeLoss(X,y,w):

L=array((X\*w-y))

sum=0

for j in range(X.shape[0]):

sum=sum+L[j]\*\*2

return sum/X.shape[0]

#参数全零初始化

w=mat(zeros((14,1)))

L\_train=[]

count=[]

L\_validation=[]

#学习率

eta=0.0004

#循环次数

times=100

# 梯度下降 公式为 参数w=w-学习率\*（2\*X.T\*（X\*w-Y））

for i in range(times):

w=w-theta\*(2\*(X1\_train.T)\*(X1\_train\*w-Y\_train))

L\_train.append(computeLoss(X1\_train,Y\_train,w)[0])

L\_validation.append(computeLoss(X1\_validation,Y\_validation,w)[0])

for i in range(10,times):

count.append(i)

#画图部分

x = count

L1\_train=[]

L1\_validation=[]

for i in range(10,times):

L1\_train.append(L\_train[i])

for i in range(10,times):

L1\_validation.append(L\_validation[i])

y1 = L1\_train

y2=L1\_validation

plt.plot(x,y1,"b-",linewidth=2,label='Ltrain')

plt.plot(x,y2,"r-",linewidth=2,label='Lvalidation')

plt . ylabel (" Value" )

plt . xlabel ("Repeat times" )

plt . title ('Figure of L\_train and L\_validation with changes in the number of iterations')

plt.legend()

plt.show()

线性分类代码：

import numpy as np

from numpy import \*

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

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

import matplotlib.pylab as plt

from matplotlib import \*

#导入数据

mem = Memory("./mycache1")

@mem.cache

def get\_data():

data = load\_svmlight\_file("australian\_scale.txt")

return data[0], data[1]

X, y = get\_data()

# 划分训练集以及验证集

Y=mat(y).T

X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split( X, Y, test\_size=0.23, random\_state=42)

#更改训练集，验证集格式

X1\_train=transpose(X\_train.todense()).T

X1\_validation=transpose(X\_validation.todense()).T

def Gradent\_w(X,y,w,b,epoch):

G=[0 for i in range(X.shape[1])]

for i in range(y.size):

if(1-(y[i,0])\*((X\*w)[i,0])-b>=0):

for j in range(X.shape[1]):

G[j]=G[j]-y[i,0]\*X[i,j]

return mat(G).T

def computeLoss(X,y,w,b,epoch):

L=[0 for i in range(y.size)]

l=0

for i in range(y.size):

if(1-(y[i,0])\*((X\*w)[i,0])-b>=0):

L[i]=1-(y[i,0])\*((X\*w)[i,0])-b

for i in range(y.size):

l=l+L[i]

l=l\*epoch

for i in range(X.shape[1]):

l=l+(w[i,0]\*\*2)/2

return l/y.size

def Gradent\_b(X,y,w,b):

G=0

for i in range(y.size):

if(1-(y[i,0])\*((X\*w)[i,0])-b>=0):

G=G-y[i,0]

return G

#初始化参数

w=mat(zeros((14,1))) #全零初始化

C=0.1 #eooch

eta=0.001 #学习率

times=100 #学习次数

count=[]

L\_train=[]

L\_validation=[]

b=0

for i in range(times):

w=w-eta\*(w+C\*Gradent\_w(X1\_train,Y\_train,w,b,C))

b=b-eta\*(C\*Gradent\_b(X1\_train,Y\_train,w,b))

L\_train.append(computeLoss(X1\_train,Y\_train,w,b,C))

L\_validation.append(computeLoss(X1\_validation,Y\_validation,w,b,C))

for i in range(times):

count.append(i)

#画图部分

x = count

y1 = L\_train

y2 = L\_validation

plt.plot(x,y1,"b-",linewidth=2,label='Ltrain')

plt.plot(x,y2,"r-",linewidth=2,label='Lvalidation')

plt . ylabel (" Value" )

plt . xlabel ("Repeat times" )

plt . title ('Figure of L\_train and L\_validation with changes in the number of iterations')

plt.legend()

plt.show()

## 8. 选择的评估方法（留出法，交叉验证，k折交叉验证等）:

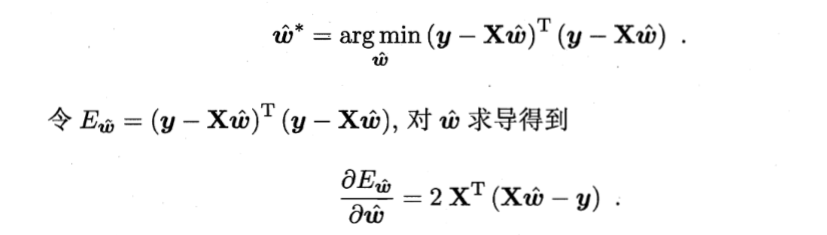
## 9. 模型参数的初始化方法:

线性回归与线性分类参数均使用全零初始化

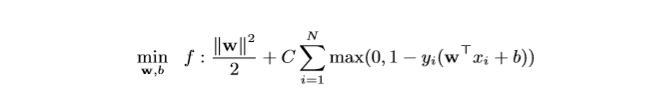
## 10.选择的loss函数及其导数:

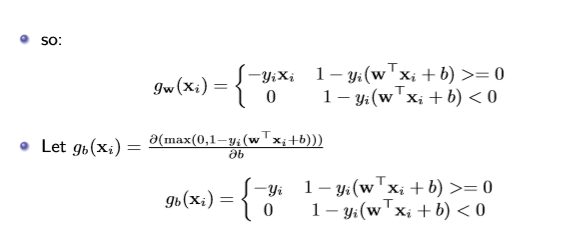
C:\Users\hao\AppData\Roaming\Tencent\Users\137315734\QQ\WinTemp\RichOle\S7ITIJ@7EXHA@%])K%VLMXY.png

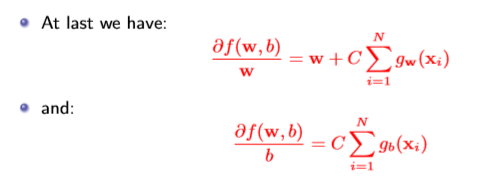
线性回归：



线性分类：







## 11.实验结果和曲线图:

## 超参数选择（η,epoch等）：

线性回归：

学习率η=0.0004,

学习次数times=100

训练集：验证集=78:22（39:11）

线性分类：

学习率η=0.001,

学习次数times=100

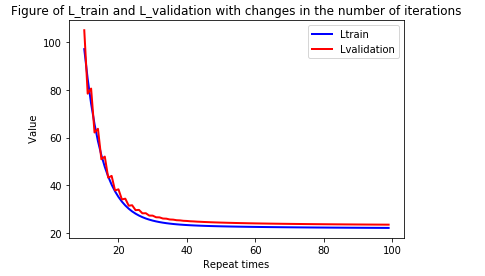
训练集：验证集=77:23

epoch=0.1

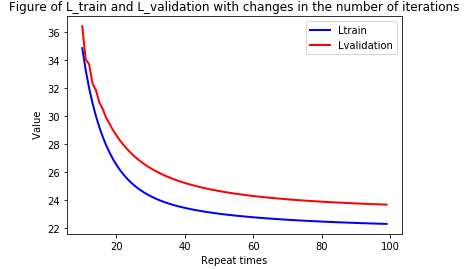
## 评估结果（根据选择的评估方法）：

线性回归：

* η=0.0005,times=100

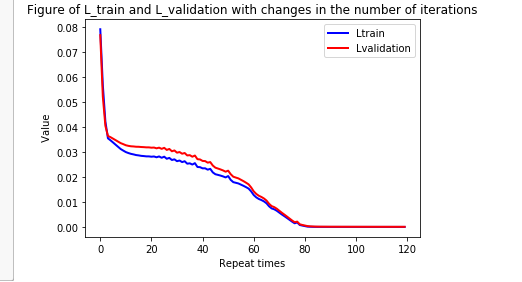


* η=0.00045,times=100

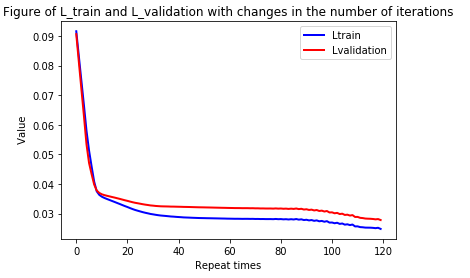


线性分类：

* η=0.005,times=120

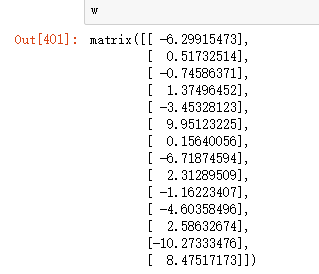


* η=0.002,times=120

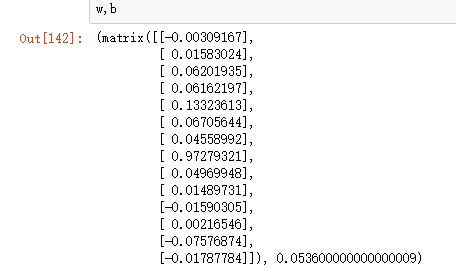


## 预测结果（最佳结果）：

线性回归：η=0.0004,times=100

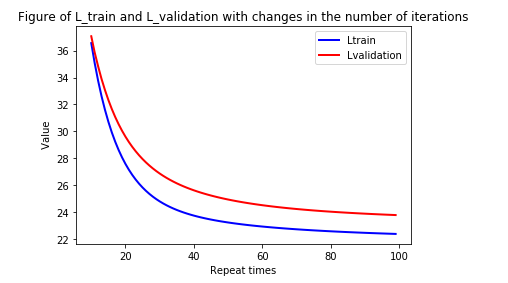


线性分类：η=0.001,times=100

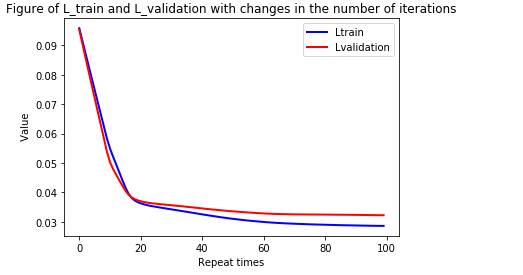


## loss曲线图：

线性回归：



线性分类：



## 12.实验结果分析:

线性回归：

线性分类：

## 13.对比线性回归和线性分类的异同点：

异：线性回归是为了找到最优拟合，线性分类则是为了寻找决策边界，此外，线性回归输出的是连续数据，线性分类输出的是离散的数据

同：过程及模型是类似的

## 14.实验总结：

第一次实验在磕磕碰碰中完成，虽然难度不算大，但由于之前对python接触不多，语句掌握不熟练以及对知识点理解不深刻，绕了许多弯路,而且如今仍有一些问题尚未解决。但通过这次实验，我切实体会到了机器学习的强大及重要之处，我需要更熟练的掌握和运用这些知识。