

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  黄景超

**学 号 201530611722**

**邮 箱 1620265690@qq.com**

**指导教师**  **吴庆耀**

**提交日期** **2017年 12 月 8 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 2 日

## 3. 报告人:黄景超

## 4. 实验目的:

## a.进一步理解线性回归和梯度下降的原理。

## b.在小规模数据集上实践。

## c.体会优化和调参的过程。

## 数据集以及数据分析：

线性回归使用的是LIBSVM Data中的Housing数据，包含506个样本，每个样本有13个属性。下载scaled版本，并将其切分为训练集，验证集。

线性分类使用的是LIBSVM Data中的australian数据，包含690个样本，每个样本有14 个属性。下载scaled版本，并将其切分为训练集，验证集。

## 实验步骤:



## （线性回归）代码内容:

#读取数据

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

import numpy as np

def get\_data():

data = load\_svmlight\_file("housing\_scale")

return data[0],data[1]

data\_x,data\_y = get\_data()

#切分数据

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

x\_train,x\_valid,y\_train,y\_valid = train\_test\_split(data\_x,data\_y,test\_size=0.33,random\_state=42)

y\_train = np.mat(y\_train).T

y\_valid = np.mat(y\_valid).T

#初始化线性模型参数

w = np.mat(np.random.random([14,1]))

from scipy import sparse

import matplotlib.pyplot as plt

m\_train = x\_train.shape[0] #训练数量

m\_valid = x\_valid.shape[0] #验证数量

x\_1\_train = np.ones((m\_train,1),dtype='float64') #创造 [1,1,……,1]^T

X\_train = sparse.hstack((x\_train,x\_1\_train)) #归并 x\_train 和 [1,1,……,1]^T

x\_1\_valid = np.ones((x\_valid.shape[0],1),dtype='float64') #创造 [1,1,……,1]^T

X\_valid = sparse.hstack((x\_valid,x\_1\_valid)) #归并 x\_valid 和 [1,1,……,1]^T

Loss\_train = [] #Loss\_train-axis

Loss\_valid = [] #Loss\_valid-axis

x = [] #x-axis:迭代次数

for n in range(0,200): #重复200次

#求得所有样本对Loss函数的梯度G

G = (1/m\_train)\*((-X\_train.T \* y\_train) + (X\_train.T \* X\_train \* w))

D = -G #取梯度G的负方向，记为D

learn\_rate = 0.05 #Learning rate

w = w + learn\_rate\*D #更新模型参数

x.append(n) #更新 x-axis

#在训练集上测试并得到Loss函数值

Loss\_train.append( ( 1/(2\*m\_train) \* (y\_train-X\_train\*w).T \* (y\_train-X\_train\*w) )[0,0])

Loss\_valid.append( ( 1/(2\*m\_valid) \* (y\_valid-X\_valid\*w).T \* (y\_valid-X\_valid\*w) )[0,0])

#画出Loss\_train 和Loss\_valid随迭代次数的变化图

plt.plot(x, Loss\_train)

plt.plot(x, Loss\_valid)

#最后一次迭代后的Loss

print("Loss of Training set after the last interation: "+str(Loss\_train[199])

+"\nLoss of Validation set after the last interation: "+str(Loss\_valid[199]))

from sklearn.linear\_model import LinearRegression

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

from scipy import sparse

x\_train,x\_valid,y\_train,y\_valid = train\_test\_split(data\_x,data\_y,test\_size=0.33,random\_state=42)

linreg = LinearRegression()

model=linreg.fit(x\_train, y\_train)

print (model)

linreg.intercept\_ = np.mat(linreg.intercept\_)

linreg.coef\_ = np.mat(linreg.coef\_)

w = sparse.vstack((linreg.coef\_.T,linreg.intercept\_))

y\_train = np.mat(y\_train).T

y\_valid = np.mat(y\_valid).T

x\_1\_train = np.ones((x\_train.shape[0],1),dtype='float64')

X\_train = sparse.hstack((x\_train,x\_1\_train))

x\_1\_valid = np.ones((x\_valid.shape[0],1),dtype='float64')

X\_valid = sparse.hstack((x\_valid,x\_1\_valid))

Loss\_train = 1/(2\*m\_train) \* np.mat((y\_train- X\_train\*w).T) \* (y\_train- X\_train\*w)

Loss\_valid =1/(2\*m\_valid) \* np.mat((y\_valid- X\_valid\*w).T) \* (y\_valid- X\_valid\*w)

print(Loss\_train)

print(Loss\_valid)

print(w)

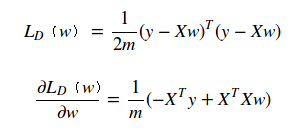
## （线性回归）选择的评估方法（留出法，交叉验证，k折交叉验证等）:

留出法

## 9. （线性回归）模型参数的初始化方法:

w = np.mat(np.random.random([14,1]))

## （线性回归）选择的loss函数及其导数:



## 11.（线性回归）实验结果和曲线图:

## 超参数选择（η,epoch等）：

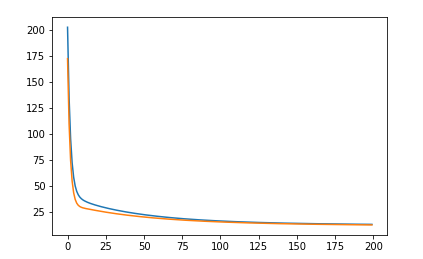
η=0.05

epoch=200

## 评估结果（根据选择的评估方法）：11.4909405

## 预测结果（最佳结果）：11.4909405

## loss曲线图：



## （线性回归）实验结果分析:

随着超参数的不断优化和调整，实验结果也越来越优化。

## （线性分类）代码内容:

#读取数据

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

import numpy as np

def get\_data():

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

return data[0],data[1]

data\_x,data\_y = get\_data()

#切分数据

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

x\_train,x\_test,y\_train,y\_test = train\_test\_split(data\_x,data\_y,test\_size=0.33,random\_state=42)

y\_train = np.mat(y\_train).T

y\_test = np.mat(y\_test).T

#初始化

w = np.mat(np.random.random([x\_train.shape[1],1]))

b = np.random.random()

import matplotlib.pyplot as plt

m\_train = x\_train.shape[0]

m\_test = x\_test.shape[0]

Loss\_train = []

Loss\_test = []

x = []

for n in range(1,50):

x.append(n)

G\_w = np.zeros([x\_train.shape[1],1])

G\_b = 0

C =100000

for n in range(0,m\_train):

result = 1 - y\_train[n]\*(w.T\*x\_train[n].T+b)

if(result>=0):

G\_w = G\_w + (w-C\*x\_train[n].T\*y\_train[n])

G\_b = G\_b + (-C\*y\_train[n])

elif(result<0):

G\_w = G\_w + (w)

G\_b = G\_b

D\_w = -G\_w

D\_b = -G\_b

learn\_rate = 0.00000001

w = w + learn\_rate\*D\_w

b = b + learn\_rate\*D\_b

yu = 0

Loss\_test\_sum = 0

fore = []

for n in range(0,m\_test):

result = w.T\*x\_test[n].T+b

if(result>=0):

fore.append(1)

elif(result<0):

fore.append(-1)

r = 0

for n in range(1,m\_test):

if(fore[n]==y\_test[n]):

r = r+1

else:

r = r

Loss\_test.append(r/m\_test)

x = x

y = Loss\_test

print(w,w.T\*w)

plt.plot(x, y)

plt.show()

## （线性分类）选择的评估方法（留出法，交叉验证，k折交叉验证等）:

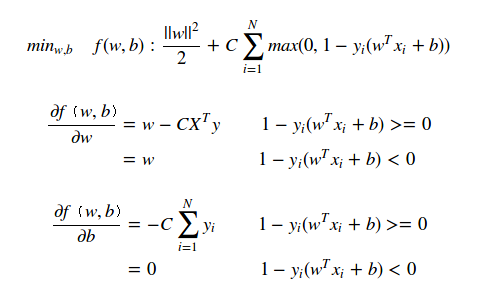
留出法

## 15. （线性分类）模型参数的初始化方法:

w = np.mat(np.random.random([x\_train.shape[1],1]))

b = np.random.random()

## 16.（线性分类）选择的loss函数及其导数:



## 17.（线性分类）实验结果和曲线图:

## 超参数选择（η,epoch等）：

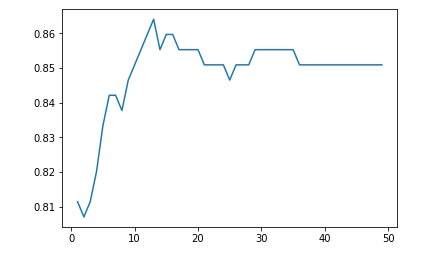
η=0.00000001

epoch=50

## 评估结果（根据选择的评估方法）：-0.03263794

## 预测结果（最佳结果）：-0.03263794

## loss曲线图：



## （线性分类）实验结果分析:

随着超参数的不断优化和调整，实验结果也越来越优化。

## 19.对比线性回归和线性分类的异同点：

对于这两类问题都有以下几个步骤：

1.如何选取一个合理的模型；

2.制造一个"美好"的误差函数；

3.采取一切可能的技术求出最好的模型参数。

总的来说两个问题本质上都是一致的，就是模型的拟合（匹配）。但是分类问题的y值(也称为label), 更离散化一些。而且，同一个y值可能对应着一大批的x, 这些x是具有一定范围的。 所以分类问题更多的是 (一定区域的一些x) 对应着 (一个y).。而回归问题的模型更倾向于(很小区域内的x，或者一般是一个x)对应着(一个y)。

## 20.实验总结：

通过本次实验，我们在小规模数据集上实践，进一步理解线性回归和梯度下降的原理，体会优化和调参的过程。对于这两类问题都有3个步骤：1.如何选取一个合理的模型；2.制造一个"美好"的误差函数；3.采取一切可能的技术求出最好的模型参数。