

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**   **林兆桃**

**学 号 201530612309**

**邮 箱 821575189@qq.com**

**指导教师**  **谭明奎**

**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:林兆桃

## 4. 实验目的

1）进一步理解线性回归和梯度下降的原理。

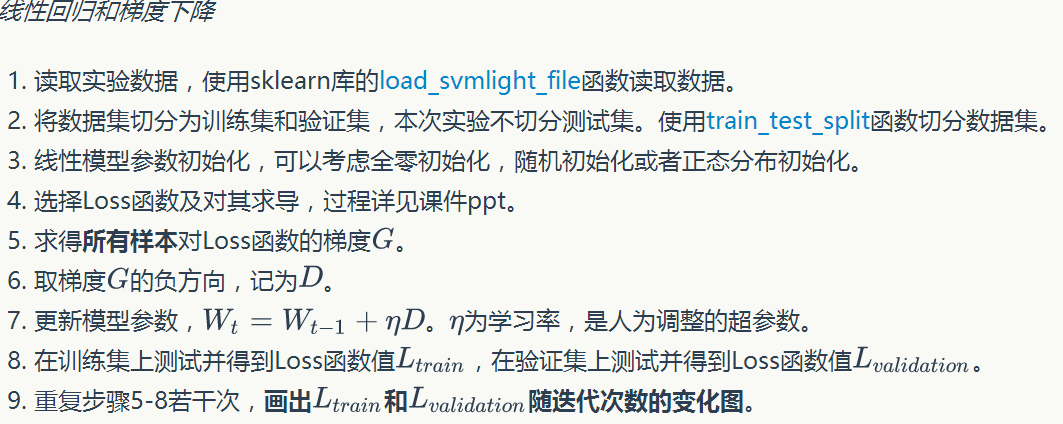
2）在小规模数据集上实践。

3）体会优化和调参的过程。

## 5. 数据集以及数据分析：

线性回归使用的是[LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/)中的[Housing](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html#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](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/)中的[australian](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#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。

## 6. 实验步骤:





## 7. 代码内容:

线性回归：

from numpy import \*

import numpy as np

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

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

import matplotlib.pyplot as plt

import time

import warnings

warnings.filterwarnings("ignore")

mem = Memory("./mycache")

@mem.cache

def get\_data():

data = load\_svmlight\_file("h")

return data[0], data[1]

def model(weight,x): #线性模型

return x.dot(weight)

def loss(weight,x,y,l): #Loss函数

sum = 0.

idex = 0

y = mat (y)

y = y.T

for line in y:

y\_pred = model(weight,x[idex ,:])

y\_pred = y\_pred - y[idex ,:]

y\_pred = y\_pred\*\*2

sum = sum + y\_pred

idex = idex + 1

return sum/2/l

def grad(weight,idex\_weight,x,y,rate,l): #求得所有样本对Loss函数的梯度G并取负方向D

idex = 0

sum = 0.

for line in y:

y\_pred = model(weight,x[idex ,:]) - y[idex]

y\_pred = y\_pred \* x[idex,idex\_weight]

idex = idex +1

sum = sum + y\_pred

return -sum/l

def updateW(weight,x,y,rate,l): #更新模型参数

idex\_weight = 0

for line1 in weight:

weight[idex\_weight]= weight[idex\_weight] + rate \* grad(weight,idex\_weight,x,y,rate,l)

idex\_weight = idex\_weight+1

return weight

def linearRegression(x,y): #线性回归函数

start = time.clock()

weight = np.array([0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.]) #线性模型参数全零初始化

iter = 0 #迭代器

iter = int(iter)

iterations = []

loss\_rec = [] #记录loss函数值

l = len(y)

l = int(l)

rate = 0.1 #学习率

loss\_val = loss(weight,x,y,l) #选择Loss函数及对其求导

loss\_rec.append(loss\_val)

iterations.append(iter)

while iter < 100:

weight = updateW(weight,x,y,rate,l) #计算梯度

loss\_updata = loss(weight,x,y,l) #计算Loss

iter = iter + 1

loss\_val = loss\_updata

loss\_rec.append(loss\_val)

iterations.append(iter)

end = time.clock()

return loss\_rec,iterations

def showLoss(X\_train,y\_train,X\_test,y\_test): # 显示loss随迭代次数的变化图

plt.xlabel('iteration')

plt.ylabel('loss')

plt.plot(X\_train,y\_train,'g.')

plt.plot(X\_test,y\_test,'r.')

plt.show()

def add\_bias(X):

temp\_y=X.shape[1]+1

temp=ones((X.shape[0],temp\_y))

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

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

temp[i,j]=X[i,j]

return temp

X, y = get\_data()

X=add\_bias(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

print("开始100次迭代:绿色曲线为Ltrain，红色曲线为Lvalidation")

Ltrain,iter\_train= linearRegression(X\_train,y\_train)

Lvalidation,iter\_validation= linearRegression(X\_test,y\_test)

showLoss(mat(np.array(iter\_train)),mat(np.array(Ltrain)),mat(np.array(iter\_validation)),mat(np.array(Lvalidation)))

print("完成")

线性分类：

from numpy import \*

import numpy as np

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

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

import matplotlib.pyplot as plt

import time

mem = Memory("./mycache")

@mem.cache

def get\_data():

data = load\_svmlight\_file("a")

return data[0], data[1]

def loss(X, y, weight, C):

loss\_ = 0.

idex = 0

y = mat (y)

y = y.T

for line in y:

s = (y[idex,:] \* (np.dot(X[idex,:], weight))).sum()

loss\_ = (weight \*\* 2).sum() + C \* max(0,1-s)

idex=idex+1

return loss\_

def grad(X, y, weight, C):

temp = np.zeros(len(weight))

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

if y[i] \* np.dot(X[i,:], weight) < 1:

temp += -y[i] \* X[i,:]

return weight + C \* temp

X, y = get\_data()

X = X.toarray()

X = np.c\_[np.ones(len(X)), X]

X\_train,X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)

print("开始1000次迭代:绿色曲线为Ltrain，红色曲线为Lvalidation")

weight = np.zeros(X\_train.shape[1])

C = 5

Ltrain = []

Lvalidation = []

iteration\_num = 1000

for i in range(iteration\_num):

G = grad(X\_train, y\_train, weight, C)

rate = 0.00005

D = -G

weight = weight + rate \* D

Ltrain.append(loss(X\_train, y\_train, weight, C))

Lvalidation.append(loss(X\_test, y\_test, weight, C))

plt.plot(range(iteration\_num),Ltrain,'g')

plt.plot(range(iteration\_num),Lvalidation,'r')

plt.xlabel('iteration')

plt.ylabel('loss')

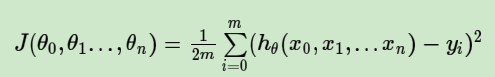
plt.show()（针对线性回归和线性分类分别填写8-12内容）

## 8. 选择的评估方法（留出法，交叉验证，k折交叉验证等）:留出法，测试集与训练集的比例为0.33：0.67

## 9. 模型参数的初始化方法:全零法

## 10.选择的loss函数及其导数:

线性回归：

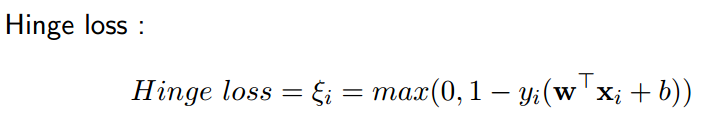
loss函数：

loss函数导数：

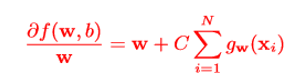


线性分类：

loss函数：



loss函数导数：



## 11.实验结果和曲线图:

## 超参数选择（η,epoch等）：

## 线性回归η

## 线性分类η、C

## 评估结果（根据选择的评估方法）：

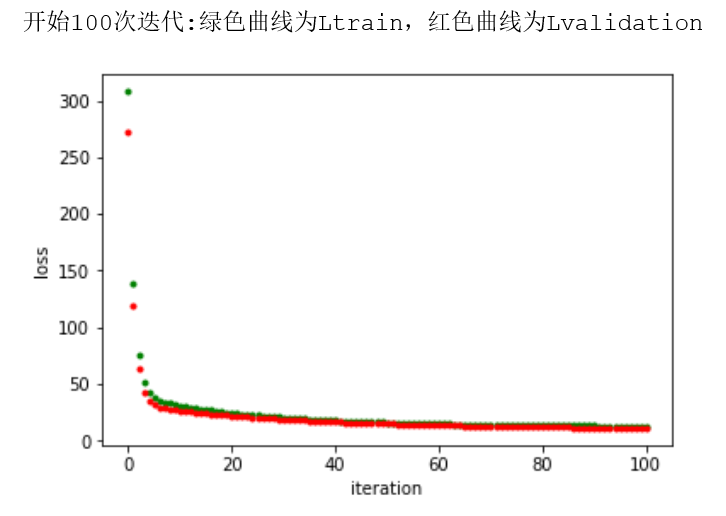
## 线性回归η=0.1

**线性分类**η**=0.00005 C=5**

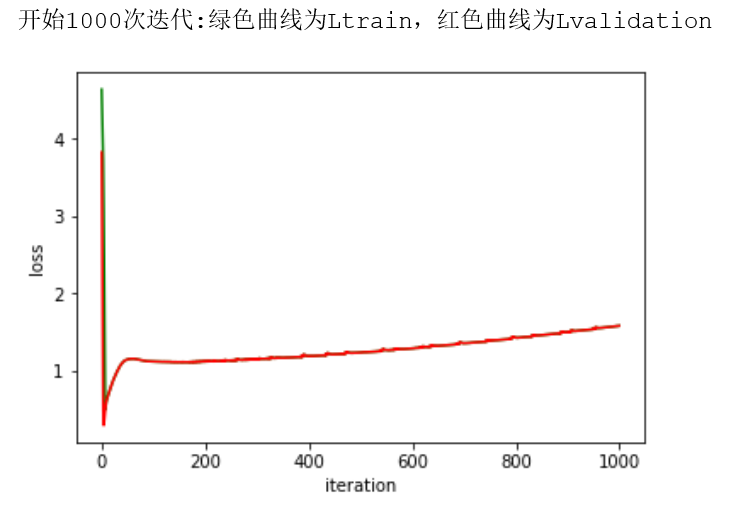
## 预测结果（最佳结果）：

## loss曲线图：

线性回归：



线性分类：



## 12.实验结果分析:

## 线性回归：由Ltrain和Lvalidation随迭代次数的变化图，loss函数呈下降趋势，由此可知拟合的模型较好。

## 线性分类：由Ltrain和Lvalidation随迭代次数的变化图，loss函数值稳定在一个较低的范围内，由此可知拟合的模型较好。

## 13.对比线性回归和线性分类的异同点：

相同点：两者解决问题的思路、使用的模型相似

不同点：使用的loss函数不同

分类问题的函数值是离散的，回归问题的函数值是连续的

## 14.实验总结：

通过本次实验，我复习到了线性回归和线性分类的概念，也对相关公式有了进一步的理解，并且能够借鉴资料来编写完整代码。实验过程中遇到了很多难题，但最终还是一一解决了。

1：对需要用到的一些python库里的函数使用不清楚，对list、array、matrix概念区分不清，导致实验卡在一些小细节上。因此我先查阅资料学了一些基础概念和numpy、matplotlib.pyplot等库的常用函数，对后续实验带来了很大帮助。

2.梯度下降的含义理解不到位，我上网找了一些线性回归和线性分类的相关视频，受益良多。

3.实验完成后，与同学进行交流讨论，发现自己的代码还是有很多不足，例如不会用sum()来简化运算。通过讨论我开拓了自己的思维，学习了解决问题的新思路。

在接下来的实验，我会继续探索和学习。