

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  吴甲科

**学 号 201530613054**

**邮 箱 347129540@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:吴甲科

## 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。

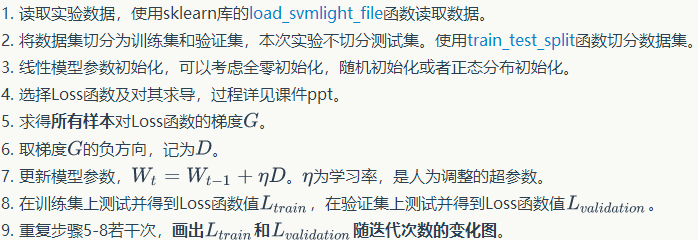
## 5. 数据集以及数据分析：

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

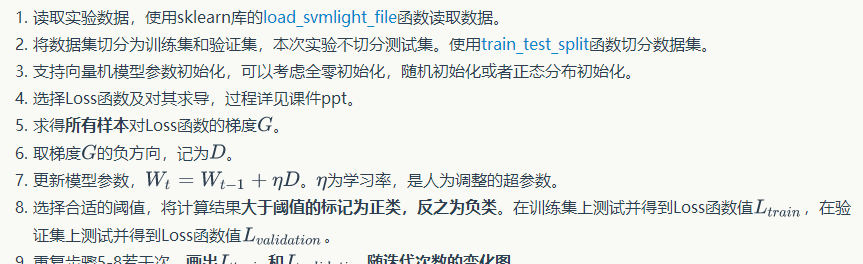
线性分类使用的是LIBSVM Data中的australian数据，包含690个样本，每个样本有14 个属性。请自行下载scaled版本，并将其切分为训练集，验证集。

## 6. 实验步骤:

线性回归和梯度下降



线性分类和梯度下降



## 7. 代码内容:

线性回归：import sklearn

import numpy as np

import matplotlib.pyplot as plt

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

mem = Memory("./mycache")

@mem.cache

def get\_data():

data = load\_svmlight\_file("C:/Users/her/Desktop/ho.txt")

return data[0], data[1]

X, y = get\_data()

X = X.todense()

#划分训练集和测试集

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

from numpy import \*

w = mat(zeros((13,1)))

b = np.random.normal()

b1 = mat(b)

w1 = vstack((w,b1))

#定义loss函数

def loss(x,y):

he = mat(ones((len(x),1)))

xx = hstack((x, he))

yy = mat(y).T

z = yy -xx\*w1

return (z.T)\*z

#定义梯度下降

def gradient(x,y):

he = mat(ones((len(x), 1)))

xx = hstack((x, he))

xxT = xx.T

yy = mat(y).T

z = xx \* w1

return 2 \* xxT \* (( z - yy ))

learnRate = 0.0005 #学习率

listx = []

listy = []

listz = []

for i in range(2000):

w1 = w1- learnRate \* gradient(X\_train,y\_train)

listx.append(i)

aaa = loss(X\_train,y\_train)

aaa2 = array(aaa)

listy.append(aaa2[0][0])

bbb = loss(X\_test, y\_test)

bbb2 = array(bbb)

listz.append(bbb2[0][0])

print(loss(X\_train,y\_train)/len(X\_train))

print(loss(X\_test,y\_test)/len(X\_test))

plt.figure('model')

plt.plot(listx ,listy, label="$Ltrain$",color="yellow")

plt.plot(listx ,listz, label="$Lvalidation$",color="red")

plt.legend()

plt.show()

线性分类：

import sklearn

import numpy as np

import matplotlib.pyplot as plt

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

mem = Memory("./mycache")

@mem.cache

def get\_data():

data = load\_svmlight\_file("C:/Users/her/Desktop/au.txt")

return data[0], data[1]

X, y = get\_data()

X = X.todense()

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

from numpy import \*

w = mat(random.rand(14,1))

b=np.random.normal()

c = 0.1

def loss(x,y):

b1 = mat(b)

w1 = vstack((w, b1))

w2=(w.T)\*w/2

a = len(x)

k = mat(ones((a, 1)))

x1 = hstack((x, k))

t1 = x1\*w1

t2 = t1

for i in range(a):

t1[i,0]=1-y[i]\*t1[i,0]

for i in range(a):

t2[i,0]=max(0,t1[i,0])

w3 = 0

for i in range(a):

w3+ = t2[i,0]

return w2+c\*w3

def gw(x,y):

b1 = mat(b)

w1 = vstack((w, b1))

a = len(x)

k = mat(ones((a, 1)))

x1 = hstack((x, k))

t1 = x1 \* w1

x2 = x

x3 = x

for i in range(a):

t1[i, 0] = 1 - y[i] \* t1[i, 0]

if(t1[i,0]>=0):

x2[i]=-y[i]\*x[i]

else:

x2[i]=0\*x[i]

x3[0]=0\*x3[0]

for i in range(a):

x3[0]+=x2[i]

return x3[0]

def gb(x,y):

b1 = mat(b)

w1 = vstack((w, b1))

a = len(x)

k = mat(ones((a, 1)))

x1 = hstack((x, k))

t1 = x1 \* w1

y2=y

for i in range(a):

t1[i, 0] = 1 - y[i] \* t1[i, 0]

if (t1[i, 0] >= 0):

y2[i]=-y[i]

else:

y2[i]=0

y1=0

for i in range(a):

y1+=y2[i]

return y1

def gradient\_w(x,y):

return w.T+c\*gw(x,y)

def gradient\_b(x,y):

return c\*gb(x,y)

rate=0.001

listx=[]

listy=[]

listz=[]

for i in range(300):

w\_t = w.T

w\_t = w\_t-rate\*gradient\_w(X\_train,y\_train)

b = b-rate\*gradient\_b(X\_train,y\_train)

w = w\_t.T

listx.append(i)

a1 = loss(X\_train, y\_train)

a2 = array(a1)

listy.append(a2[0][0])

b = loss(X\_test, y\_test)

b2 = array(b)

listz.append(b2[0][0])

print(loss(X\_train,y\_train)/len(X\_train))

print(loss(X\_test,y\_test)/len(X\_test))

plt.figure('model')

plt.plot(listx, listy, label="$Ltrain$", color="yellow")

# plt.figure('losstest')

plt.plot(listx, listz, label="$Lvalidation$", color="red")

plt.title(["rate=", rate])

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）:留出法

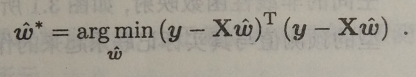
## 9. 模型参数的初始化方法:

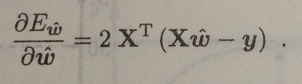
线性回归：全零初始化和随机初始化

线性分类：随机初始化

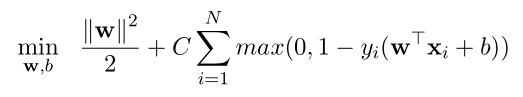
## 10.选择的loss函数及其导数:

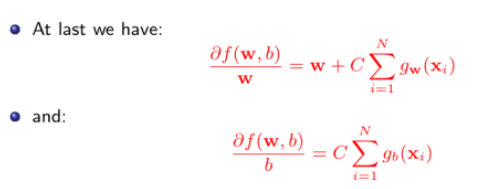
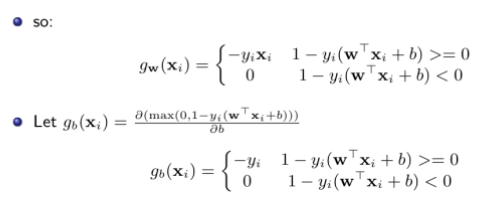
**线性回归：**

Loss函数：

导数：

**线性分类：**

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：**学习率：0.0005 迭代：2000次

**线性分类：**学习率：0.001 迭代：300次

## 评估结果（根据选择的评估方法）：

线性回归：：19.32646828

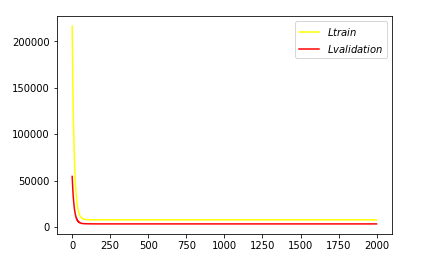
线性分类：：0.1029105

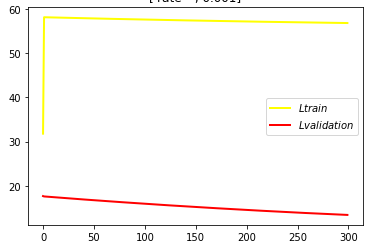
## 预测结果（最佳结果）：

线性回归：：33.4490231

线性分类：：0.09706383

## loss曲线图：

线性回归：

线性分类：

## 12.实验结果分析:迭代过后逐渐稳定

## 13.对比线性回归和线性分类的异同点：

同：都是属于线性模型

不同：loss函数不同

## 14.实验总结：

对于这次实验自己感觉自己还是有很多不足的地方，还要多多实验才可以提高自己的水平