

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  潘楷乐

**学 号 201530612583**

**邮 箱 826114548@qq.com**

**指导教师**  **吴庆耀**

**提交日期** **2017年 12 月 7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: 潘楷乐

## 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/" \t "_blank)中的Housing数据，包含506个样本，每个样本有13个属性。

线性分类使用的是[LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/" \t "_blank)中的australian数据，包含690个样本，每个样本有14 个属性。

## 6. 实验步骤:

本次实验代码及画图均在jupyter上完成。

*线性回归和梯度下降*

1. 读取实验数据，使用sklearn库的load\_svmlight\_file函数读取数据。
2. 将数据集切分为训练集和验证集，本次实验不切分测试集。使用train\_test\_split函数切分数据集。
3. 线性模型参数初始化，可以考虑全零初始化，随机初始化或者正态分布初始化。
4. 选择Loss函数及对其求导，过程详见课件ppt。
5. 求得**所有样本**对Loss函数的梯度G。
6. 取梯度G的负方向，记为D。
7. 更新模型参数，Wt=Wt-1+nD。n为学习率，是人为调整的超参数。
8. 在训练集上测试并得到Loss函数值Ltrain，在验证集上测试并得到Loss函数值Lvalidation。
9. 重复步骤5-8若干次，**画出**Ltrain**和随迭代次数的变化图**。

*线性分类和梯度下降*

1. 读取实验数据，使用sklearn库的load\_svmlight\_file函数读取数据。
2. 将数据集切分为训练集和验证集，本次实验不切分测试集。使用train\_test\_split函数切分数据集。

3. 线性模型参数初始化，可以考虑全零初始化，随机初始化或者正态分布初始化。

4. 选择Loss函数及对其求导，过程详见课件ppt。

5. 求得**所有样本**对Loss函数的梯度G。

6. 取梯度G的负方向，记为D。

7.更新模型参数，Wt=Wt-1+nD。n为学习率，是人为调整的超参数。

8. 选择合适的阈值，将计算结果**大于阈值的标记为正类，反之为负类**。

在训练集上测试并得到Loss函数值Ltrain，在验证集上测试并得到Loss函数值Lvalidation。

9.重复步骤5-8若干次，**画出**Ltrain**和随迭代次数的变化图**整理实验结果并完成实验报告

## 7. 代码内容:

*线性回归和梯度下降*

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

import numpy as np

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

import matplotlib.pyplot as plt

%matplotlib inline

#读取数据

def get\_data():

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

return data[0], data[1]

#切分数据

x, y = get\_data()

row=x.shape[0]

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

#初始化为0，设置学习率和梯度下降次数

w =[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]

w0=0

learning\_rate=0.01

num=8

loss\_train\_group=[]

loss\_test\_group=[]

#计算测试集上的方差

def test\_loss():

test\_fx=[0 for i in range(167)]

for i in range(0,167):

for j in range(0,13):

test\_fx[i]+=w[j]\*x\_test[i,j]

test\_fx[i]+=w0;

sum=0;

for i in range(167):

sum+=(test\_fx[i]-y\_test[i])\*(test\_fx[i]-y\_test[i])

return sum;

#计算训练集上的方差

def train\_loss():

train\_fx=[0 for i in range(339)]

for i in range(0,339):

for j in range(0,13):

train\_fx[i]+=w[j]\*x\_train[i,j]

train\_fx[i]+=w0;

sum=0;

for i in range(339):

sum+=(train\_fx[i]-y\_train[i])\*(train\_fx[i]-y\_train[i])

return sum;

#梯度下降算法

def gradient\_descent():

mylist =[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]

for i in range(339):

temp\_1=0

temp\_2=0

for j in range(13):

temp\_1+=w[j]\*x\_train[i,j]

temp\_2=y\_train[i]-temp\_1

for k in range(13):

mylist[k]+=x\_train[i,k]\*temp\_2

for m in range(13):

mylist[m]/=row

w[m]=w[m]-(-learning\_rate\*mylist[m])

#执行梯度下降

def optimizer():

for i in range(num):

gradient\_descent()

loss\_train\_group.append(train\_loss())

loss\_test\_group.append(test\_loss())

for i in range(num):

print("训练集第",i+1,"次误差：",loss\_train\_group[i])

for i in range(num):

print("测试集第",i+1,"次误差：",loss\_test\_group[i])

print()

#主函数

def main():

optimizer()

#画出图表

n = np.arange(num)

print("w=",w,"after 8 times")

plt.plot(n,loss\_train\_group,label='loss\_train')

plt.plot(n,loss\_test\_group,label='loss\_test')

plt.legend(loc='upper right')

plt.xlabel('times')

plt.ylabel('loss')

main()

*线性分类和梯度下降*

from sklearn.datasets import load\_svmlight\_file

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

import numpy as np

import numpy.linalg as nl

import matplotlib.pyplot as plt

%matplotlib inline

X,Y = load\_svmlight\_file("Desktop//australian\_scale.txt")

row=X.shape[0]

B = np.mat(np.ones((X.shape[0],1)))

X=np.mat(np.c\_[X.A,B])

Y=Y.reshape(len(Y),-1)

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

#初始化全零

W=np.mat(np.zeros((15,1)))

learning\_rate=0.04

C=2

#迭代次数

num=8

#阀值

threshold=0.0

loss\_train=[]

loss\_test=[]

#目标函数

def score(w,x):

return x\*w

#loss函数

def loss(y,w,C,x):

temp=0

for i in range(len(y)):

temp+=max(1-y[i,0]\*score(w,x[i]),0)

return 0.5\*w.T.dot(w)+C\*temp

#求梯度

def gradient(x,y,w):

temp=w.T

for i in range(len(x)):

if (1-y[i,0]\*score(w,x\_train[i])>=0):

temp=temp-(y[i]\*x[i])\*C

return temp.T/x\_train.shape[0]

#梯度下降

def gradient\_descent(w):

temp\_w=w-learning\_rate\*gradient(x\_train,y\_train,w)

return temp\_w

#优化

def optimizer(w):

temp\_w=w

for i in range(num):

temp\_w=gradient\_descent(temp\_w)

loss\_train.append(loss(y\_train,temp\_w,C,x\_train)[0,0])

loss\_test.append(loss(y\_test,temp\_w,C,x\_test)[0,0])

return temp\_w

def main():

w=optimizer(W)

print("8次递归下降后w的取值为：")

print(w)

print()

for i in range(num):

print("训练集第",i+1,"次误差：",loss\_train[i])

print()

for i in range(num):

print("测试集第",i+1,"次误差：",loss\_test[i])

print()

#训练集命中率

train\_hit=0

for i in range(len(x\_train)):

if (score(w,x\_train[i]) >= threshold and y\_train[i] == 1) or (score(w,x\_train[i]) < threshold and y\_train[i] == -1):

train\_hit += 1

#验证集命中率

test\_hit=0

for i in range(len(x\_test)):

if (score(w,x\_test[i]) >= threshold and y\_test[i] == 1) or (score(w,x\_test[i]) < threshold and y\_test[i] == -1):

test\_hit += 1

print("训练集命中率：",train\_hit/len(x\_train))

print("测试集命中率：",test\_hit/len(x\_test))

n = np.arange(num)

plt.plot(n,loss\_train,label='loss\_train')

plt.plot(n,loss\_test,label='loss\_test')

plt.legend(loc='upper right')

plt.xlabel('times')

plt.ylabel('loss')

main()

## 8. 选择的评估方法（留出法，交叉验证，k折交叉验证等）:

线性回归：留出法

线性分类：留出法

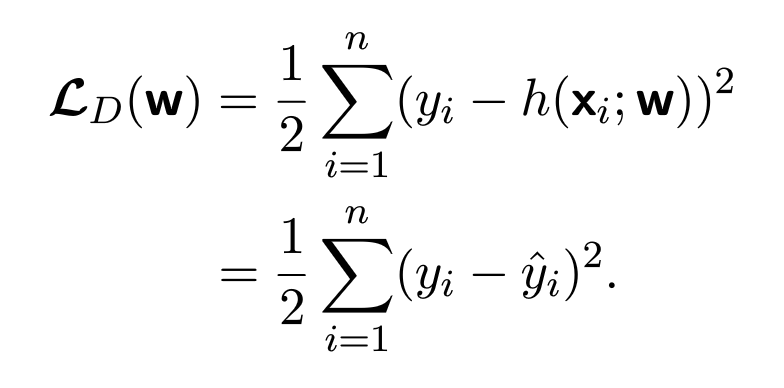
## 9. 模型参数的初始化方法:

线性回归：全零初始化

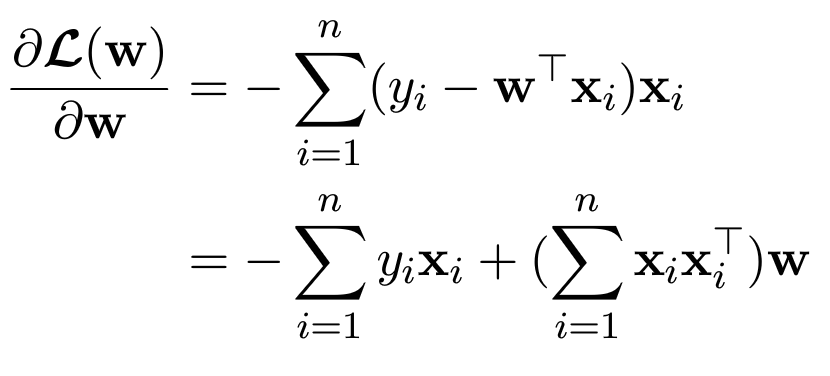
线性分类：全零初始化

## 10.选择的loss函数及其导数:

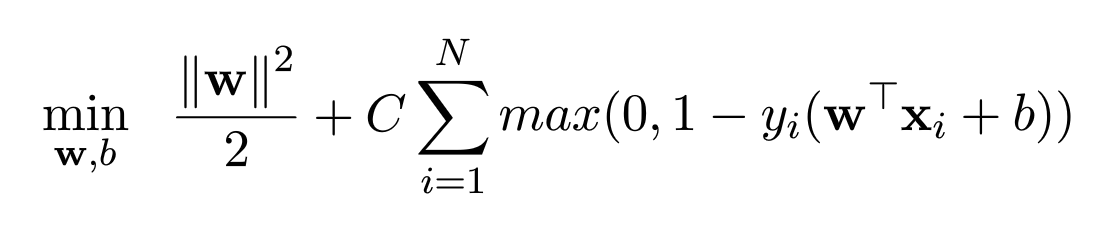
线性回归： loss函数：



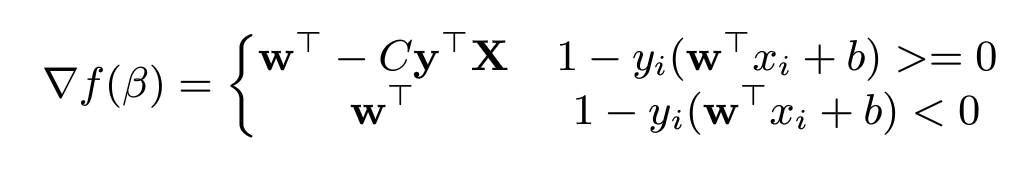
导数：



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learning\_rate=0.01

线性分类：学习率earning\_rate=0.04

惩罚系数C=2

阈值threshold=0.0

## 评估结果（根据选择的评估方法）：

线性回归：

训练集第 1 次误差： 199654.355991

训练集第 2 次误差： 191011.193694

训练集第 3 次误差： 182804.002799

训练集第 4 次误差： 175010.630142

训练集第 5 次误差： 167610.04931

训练集第 6 次误差： 160582.303313

训练集第 7 次误差： 153908.450179

训练集第 8 次误差： 147570.511316

测试集第 1 次误差： 86592.4259148

测试集第 2 次误差： 82538.4888148

测试集第 3 次误差： 78696.1737579

测试集第 4 次误差： 75054.5469127

测试集第 5 次误差： 71603.2353534

测试集第 6 次误差： 68332.3983953

测试集第 7 次误差： 65232.7003923

测试集第 8 次误差： 62295.2849215

线性分类：

训练集第 1 次误差： 858.638201616

训练集第 2 次误差： 793.287720802

训练集第 3 次误差： 727.948555599

训练集第 4 次误差： 662.620704047

训练集第 5 次误差： 597.304164187

训练集第 6 次误差： 533.674908576

训练集第 7 次误差： 488.933069511

训练集第 8 次误差： 457.341901672

测试集第 1 次误差： 422.268962671

测试集第 2 次误差： 388.546504319

测试集第 3 次误差： 354.83262322

测试集第 4 次误差： 321.127317653

测试集第 5 次误差： 287.430585895

测试集第 6 次误差： 254.326402646

测试集第 7 次误差： 231.763503337

测试集第 8 次误差： 219.909188864

训练集命中率： 0.8528138528138528

测试集命中率： 0.8552631578947368

## 预测结果（最佳结果）：

线性回归：w= [-1.0746391274625267, -0.78747387856637863, -0.34502018509874066, -0.88884963212441503, -0.42274620795562706, 0.16227601354617827, 0.30660644747550775, -0.54330357603097423, -0.42563050862502932, -0.30248810904994206, 0.11647441260319669, 0.97712541747820592, -0.60236011264650846, 0.0]

线性分类：w= [[-0.01575014]

[ 0.03398042]

[ 0.06377558]

[ 0.0559112 ]

[ 0.13364944]

[ 0.07486666]

[ 0.06124735]

[ 0.44089995]

[ 0.28302572]

[ 0.05677651]

[ 0.02163603]

[ 0.01852186]

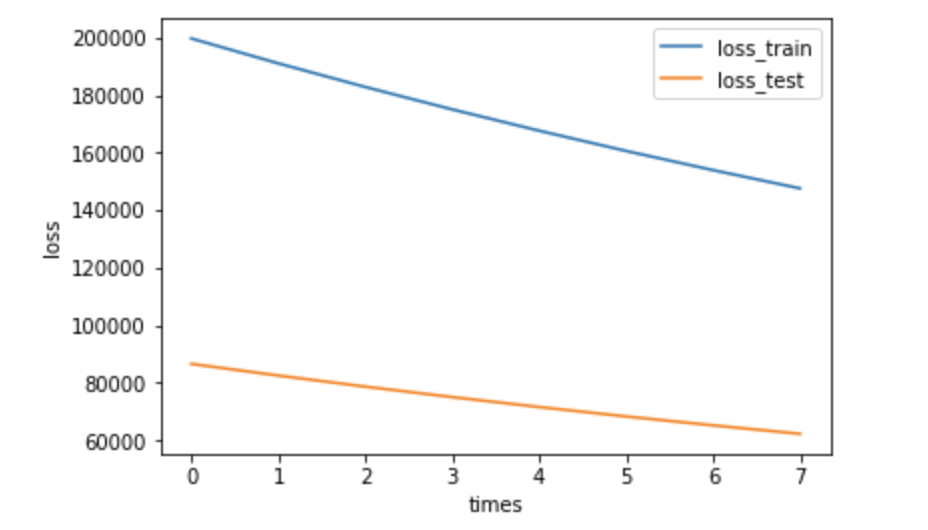
[ 0.01084468]

[ 0.03272613]

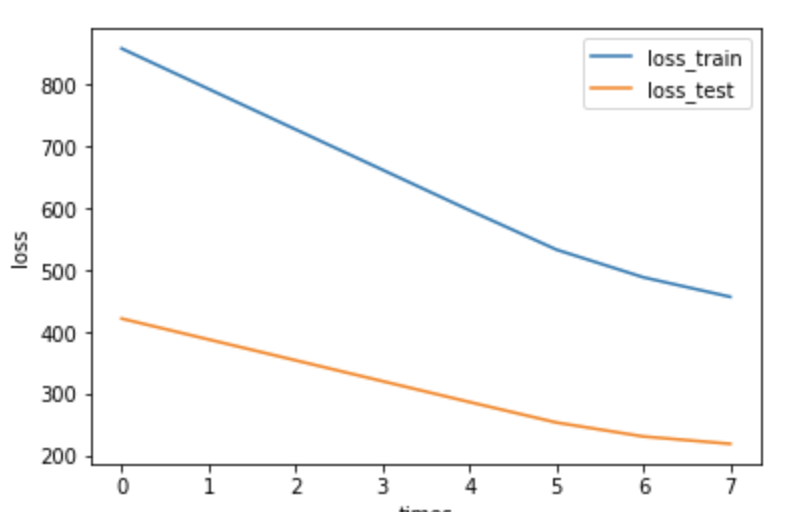
[-0.02267172]]

## loss曲线图：

线性回归：



线性分类：



## 12.实验结果分析:

线性回归：随着递归次数增加，训练集和测试集的loss都明显下降

线性分类：随着递归次数增加，训练集和测试集的loss都明显下降

## 13.对比线性回归和线性分类的异同点：

相同点：线性回归和线性分类最终都是最优化问题，都可以用梯度下降的方法解决。两者都是用线性的目标函数去拟合。这两类问题都有以下几个步骤,

选取一个 合理的模型(线性的，or 非线性的(e.g. 阶跃函数， 高斯函数)).

制造一个合理的误差函数 (可以评估拟合程度，而且还是convex函数)

采取一切可能的技术(e.g. 导数下降法，解极值方程法) 求出最好的模型参数

不同点：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).

## 14.实验总结：

通过本次实验，我对于线性分类和线性回归有了更深入的了解，也掌握了梯度下降方法去求最优化问题。但是我的代码实现能力较差，代码写的不够规范，花的时间也较多，效率不够高，希望以后我能有更多这样的实践机会，提高自己的编码能力和解决问题的能力。