

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  黄亦昕

**学 号 201530611784**

**邮 箱 562145211@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 2 日

## 3. 报告人:黄亦昕

## 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。采用了特征放缩后的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。采用了特征放缩后的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**

# write your code here

from sklearn.datasets import load\_svmlight\_file

from numpy import \*

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

t\_X,t\_y=load\_svmlight\_file("/Users/hyx/Desktop/machinelearning/project1/Housing.txt")

t\_X=t\_X.todense() #转化为普通矩阵

t\_X\_column\_num=t\_X[0].size

t\_X\_row\_num=int(t\_X.size/t\_X\_column\_num)

ones\_column = ones((t\_X\_row\_num,1))

t\_X=hstack((ones\_column,t\_X))

t\_X\_column\_num +=1 #加一列1

#将数据集切分为训练集和验证集

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split( t\_X, t\_y, test\_size=0.33, random\_state=53)

w = zeros((1,t\_X\_column\_num))

w\_temple=zeros((1,t\_X\_column\_num))

alpha = 0.00116

train\_size = y\_train.size

test\_size = y\_test.size

p\_x = []

p\_train\_loss = []

p\_validation\_loss = []

for j in range(100):

for i in range(train\_size):

diff=y\_train[i]-float(w\*X\_train[i].T)

w\_temple=w\_temple+(alpha\*diff\*X\_train[i])

w=w\_temple

#计算train loss

train\_loss=0

for lp in range(train\_size):

train\_loss += (y\_train[lp]-float(w\*X\_train[lp].T))\*\*2/2/train\_size

p\_x.append(j+1)

p\_train\_loss.append(train\_loss)

#计算validation loss

validation\_loss=0

for lp in range(test\_size):

validation\_loss += (y\_test[lp]-float(w\*X\_test[lp].T))\*\*2/2/test\_size

p\_validation\_loss.append(validation\_loss)

plt.plot(p\_x, p\_train\_loss,label="train loss")

plt.plot(p\_x, p\_validation\_loss,label="validation loss")

plt.title('graph of Ltrain ')

plt.ylabel('loss')

plt.xlabel('the number of iterations')

print(' blue for train loss\n orange for validation loss')

# print('train loss:',p\_train\_loss)

# print('validation loss:',p\_validation\_loss)

**线性分类**

from sklearn.datasets import load\_svmlight\_file

from numpy import \*

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

t\_X,t\_y=load\_svmlight\_file("/Users/hyx/Desktop/machinelearning/project1/australian.txt")

t\_X=t\_X.todense() #转化为普通矩阵

t\_X\_column\_num=t\_X[0].size

t\_X\_row\_num=int(t\_X.size/t\_X\_column\_num)

ones\_column = ones((t\_X\_row\_num,1))

t\_X=hstack((ones\_column,t\_X))

t\_X\_column\_num +=1 #矩阵加一列1

#将数据集切分为训练集和验证集

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split( t\_X, t\_y, test\_size=0.33, random\_state=3)

#w全零初始化

w = zeros((1,t\_X\_column\_num))

w\_temple=zeros((1,t\_X\_column\_num))

eta = 0.0008125

train\_size = y\_train.size

test\_size=y\_test.size

#定义函数c()

def c(x,y,w):

if(y\*(w\*x.T)<1):

return -y

return 0

#用于图显示

p\_x = []

p\_train\_loss = []

p\_validation\_loss = []

for j in range(80):

for i in range(train\_size):

w\_temple=w\_temple-(eta\*(c(X\_train[i],y\_train[i],w\_temple))\*X\_train[i])

w=w\_temple

#计算train loss

train\_loss=0

for lp in range(train\_size):

train\_loss += float(max(0,1-y\_train[lp]\*(w\*X\_train[lp].T)))/train\_size

p\_x.append(j+1)

p\_train\_loss.append(train\_loss)

#计算validation loss

validation\_loss=0

for lp2 in range(test\_size):

validation\_loss += float(max(0,1-y\_test[lp2]\*(w\*X\_test[lp2].T)))/test\_size

p\_validation\_loss.append(validation\_loss)

plt.plot(p\_x, p\_train\_loss,label="train loss")

plt.plot(p\_x, p\_validation\_loss,label="validation loss")

plt.title('graph of Ltrain as well as Lvalidation with the number of iterations')

plt.ylabel('loss')

plt.xlabel('the number of iterations')

print(' blue for train loss\n orange for validation loss')

# print('train loss:',p\_train\_loss)

# print('validation loss:',p\_validation\_loss)

## 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:

线性回归模型：

../../屏幕快照%202017-12-07%20下午11.39.20.png

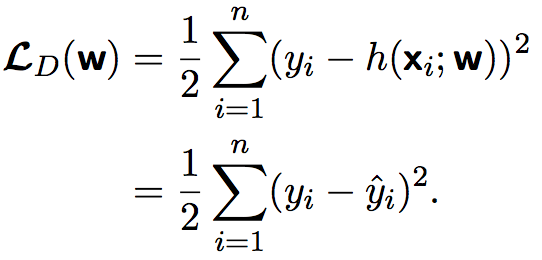
参数初始化方法：全零初始化

线性分类模型：

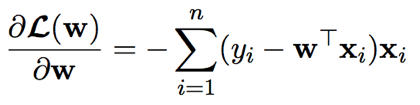
参数初始化方法：全零初始化

## 10.选择的loss函数及其导数:

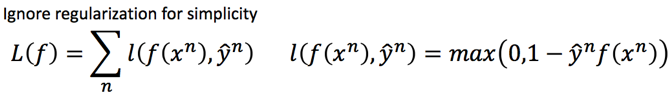
线性回归loss函数



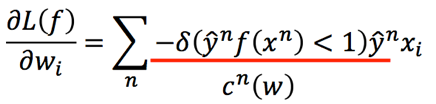
导数



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）：

线性回归更新w函数：

w=w+(alpha\*diff\*X\_train[i])/train\_size

**超参数alpha：**0.00116

线性分类更新w函数：

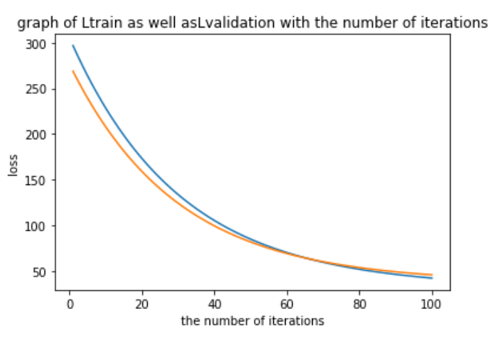
w=w-(eta\*(c(X\_train[i],y\_train[i],w\_temple))\*X\_train[i])/train\_size

**超参数η（eta）=**  0.0008125

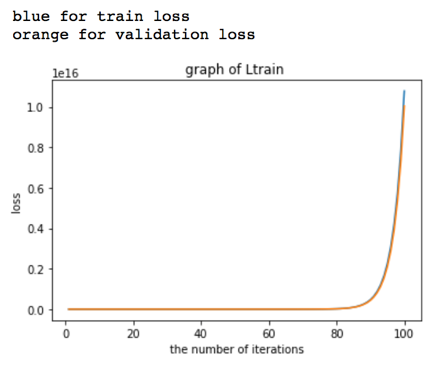
## 评估结果（根据选择的评估方法）：

线性回归调参中的结果

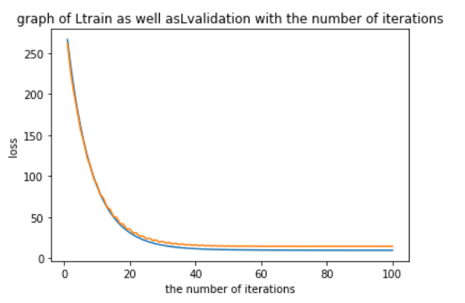
1. alpha过小（alpha = 0.00001）



（2）alpha过大（alpha = 0.002）



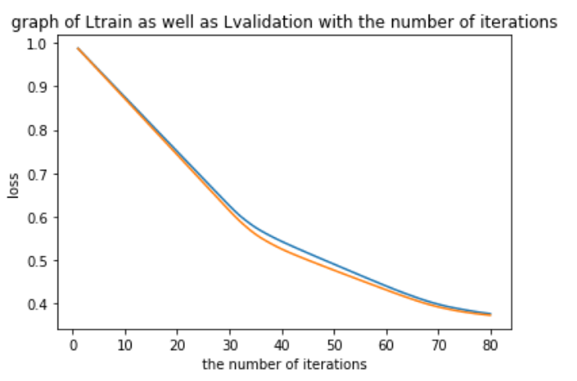
（3）alpha适合（alpha = 0.00116）



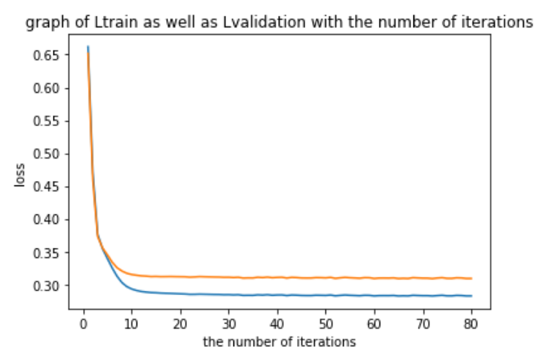
再经过更细致的调参后，选择（alpha = 0.00116）

线性分类调参中的结果

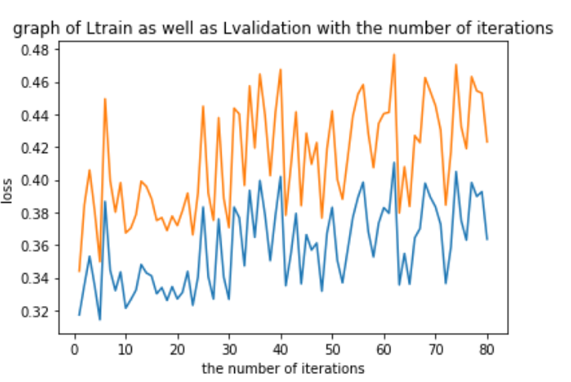
**（1）eta**过小（**eta** = 0.00003    ）



1. **eta**合适（**eta** = 0.0008125）



1. **eta**过大（0.05）



再经过更细致的调参后，选择（**eta** = 0.0008125）

## 预测结果（最佳结果）：

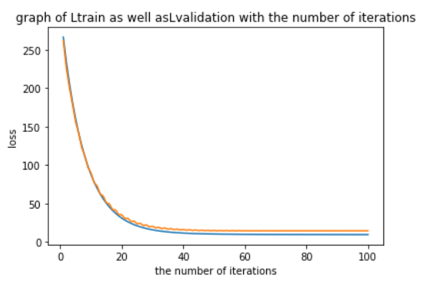
线性回归validation loss 11.623

线性分类validation loss 0.302 （分类正确率69.8%）

## loss曲线图：

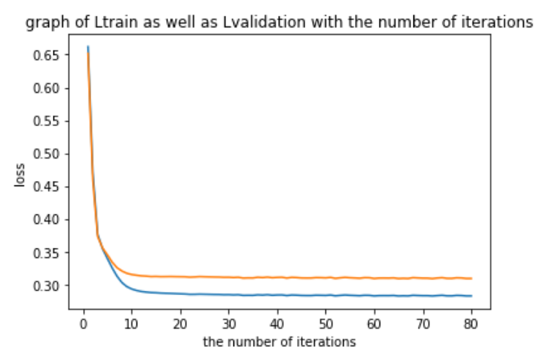
线性回归

线性回归loss 11.875



线性分类

线性分类loss 0.3089



## 12.实验结果分析:

选择合适的超参数，使用梯度下降，在一定范围内随着迭代次数的增加，线性回归和线性分类的loss值均有下降，能得到较优的模型。

## 13.对比线性回归和线性分类的异同点：

异

（1） 输出变量的类型不同。线性回归是定量输出，线性分类是定性输出。

（2） 线性分类是将实力书记划分到合适的分类中。线性回归通过给定数据点拟合线性函数

同

（1） 同属于监督学习，在本次试验中，既要给定影响因子，也要给出最终的数值或分类，用于更新改进测试模型。

（2） 都可以通过梯度下降的方法得到较优的模型

## 14.实验总结：

这次实验，在小规模数据集上实践线性回归，线性分类和梯度下降。使我进一步理解线性回归，线性回归和梯度下降的原理，加深了线性回，归线性回归和梯度下降的印象。体会优化和调参的过程。能够使用python库来实现数学运算和图形显示。