

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**   **张驰**

**学 号 201530613603**

**邮 箱 916583810@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: 张驰

## 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。

## 数据集以及数据分析：

## （1）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。

## （2）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。

## 实验步骤:

## **线性回归和梯度下降**

## 图片1

## **线性分类和梯度下降**



## 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

def get\_data():

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

return data[0], data[1]

X, y = get\_data()

import numpy as np

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

X = X.dot(np.eye(X.shape[1]))

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

X = np.hstack((X,o)) y = y.reshape((y.shape[0],1))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)w = np.zeros((13,1))

w = np.zeros((13,1))

n = 100 /

Ltrain = np.zeros((n))

Ltest = np.zeros((n))

def Lfun(w,X,y):

m = y.shape[0]

return(((y-X.dot(w)).T.dot(y-X.dot(w)))\*0.5)/m

def DER(w,X,y):

m = y.shape[0]

return ((X.T).dot(X.dot(w)-y))/m

for i in range (n):

G = (DER(w,X\_train,y\_train))

w = w - 0.1\*G

Ltrain[i] = Lfun(w,X\_train,y\_train)

Ltest[i] = Lfun(w,X\_test,y\_test)

import matplotlib.pyplot as plt

x = np.arange(0,n,1)

plt.plot(x, Ltrain, 'r',label = 'training')

plt.plot(x, Ltest, 'b',label = 'testing')

plt.legend(loc='upper right')

plt.xlabel('Times of iteration')

plt.ylabel('Loss')

plt.show()

## **线性分类和梯度下降**

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

def get\_data():

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

return data[0], data[1]

X, y = get\_data()

import numpy as np

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

X = X.dot(np.eye(X.shape[1]))

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

X = np.hstack((X,o))

y = y.reshape((y.shape[0],1))

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

w = np.zeros((13,1))

n = 100

Ltrain = np.zeros((n))

Ltest = np.zeros((n))

def Lfun(w,X,y):

m = y.shape[0]

o = np.ones((m,1))

l = o-(X.dot(w))\*y

for i in range (m):

if l[i] < 0:

l[i] = 0

return l.sum()+0.5\*np.sum(w\*w)

def DER(w,X,y):

m = y.shape[0]

j = (X.dot(w))\*y

o = np.zeros((m,1))

for i in range (m):

if j[i] < 1:

o[i] = y[i]

return -((X.T).dot(o))

for i in range (n):

G = (DER(w,X\_train,y\_train))

w = w - 0.1\*G

Ltrain[i] = Lfun(w,X\_train,y\_train)

Ltest[i] = Lfun(w,X\_test,y\_test)

import matplotlib.pyplot as plt

x = np.arange(0,n,1)

plt.plot(x, Ltrain, 'r',label = 'training')

plt.plot(x, Ltest, 'b',label = 'testing')

plt.legend(loc='upper right')

plt.xlabel('Times of iteration')

plt.ylabel('Loss')

plt.show()

## 线性回归和梯度下降8-12：

## 8.选择的评估方法（留出法，交叉验证，k折交叉验证等）:

使用了交叉验证法，将数据集随机切分为训练集和验证集，不切分测试集。训练集和验证集大小约为2：1。

## 9.模型参数的初始化方法:

全零初始化。

## 选择的loss函数及其导数:

Loss函数：

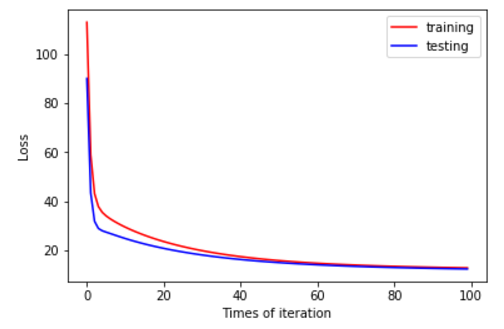
梯度： 的梯度

## 11.实验结果和曲线图:

## 超参数选择（η,epoch等）：η选定为0.1

迭代100次后Ltrain=12.78114594, Ltest=12.33486608

## loss曲线图：



## 12.实验结果分析:

在实验中，θ被初始化为零，在每一次迭代更新θ参数时，利用损失函数对θ的导数，乘以学习率η得到新的θ，重复此过程直至θ收敛。我们从图中可以看到在前几次更新迭代周期后Loss函数值就能快速下降，在大概100次迭代后Loss值已经下降到初始值的1/10左右，且随着迭代一直表现为极小幅的下降，可以认为稳定在一个较小的值上，说明模型训练完成，θ收敛到局部最优解，满足要求。可以认为其有继续下降的趋势。

## 线性分类和梯度下降8-12：

## 8.选择的评估方法（留出法，交叉验证，k折交叉验证等）:

使用了交叉验证法，将数据集随机切分为训练集和验证集，不切分测试集。训练集和验证集大小约为2：1。

## 9.模型参数的初始化方法:

全零初始化。

## 10.选择的loss函数及其导数:

Loss函数：

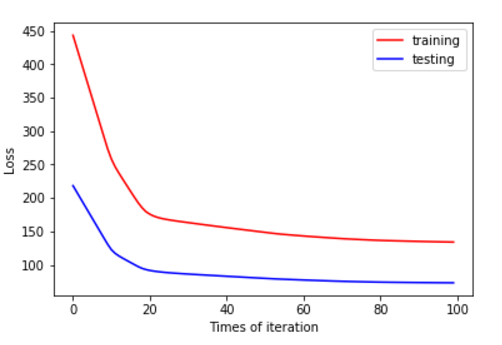
梯度：对wi的梯度

## 11.实验结果和曲线图:

## 超参数选择（η,epoch等）：η选定为0.0001

迭代100次后Ltrain=133.97088545, Ltest=73.05523108

## loss曲线图：



## 12.实验结果分析:

在实验中，θ被初始化为零，在每一次迭代更新θ参数时，利用损失函数对θ的导数，乘以学习率η得到新的θ，重复此过程直至θ收敛。我们从图中可以看到在前20次更新迭代周期后Loss函数值就能快速下降，20次以后降幅明显减小，在大概100次迭代后Loss值已经下降到初始值的1/3左右，并且往后的降幅也非常小，也有表现出极小的升幅，可以认为Loss值在一个确值附近上下波动，忽略误差则可以视为稳定，说明模型训练完成，θ收敛到局部最优解，满足要求。

## 对比线性回归和线性分类的异同点：

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)。

## 实验总结：

本次实验我学习到了很多在实践运用的经验，将课程中学习到的知识运用在实际问题上。并且让我更深地掌握了python,了解了通过数据样本训练特定机器学习模型的过程，此外还让我巩固了线性代数相关的知识。