

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**   **吕睿**

**学 号 201530612477**

**邮 箱 729239617@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: 吕睿

## 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**

## （1）读取实验数据，使用sklearn库的[load\_svmlight\_file](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html" \t "https://www.zybuluo.com/chenyaofo/note/_blank)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。（注：由于试验要求文档的改动问题，本报告中的验证集会有test和validation两个代称）使用[train\_test\_split](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html" \t "https://www.zybuluo.com/chenyaofo/note/_blank)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。

## （6）取梯度的负方向，记为。

## （7）更新模型参数，。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，在验证集上测试并得到Loss函数值。

## (9）重复步骤5-8若干次，画出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](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html" \t "https://www.zybuluo.com/chenyaofo/note/_blank)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。（注：由于试验要求文档的改动问题，本报告中的验证集会有test和validation两个代称）使用[train\_test\_split](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html" \t "https://www.zybuluo.com/chenyaofo/note/_blank)函数切分数据集。

## (3)支持向量机模型参数初始化，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。

## (6)取梯度的负方向，记为。

## (7)更新模型参数，。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，在验证集上测试并得到Loss函数值。

## (9)重复步骤5-8若干次，画出和随迭代次数的变化图。

## 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()#使用sklearn库的load\_svmlight\_file函数读取数据

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))#把所有x的属性加上一个1，用于和w14（b）相乘

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) #使用train\_test\_split函数将数据集切分为训练集和验证集

w = np.zeros((13,1))#将线性模型参数w初始化为全零

n = 100 #设置迭代次数

Ltrain = np.zeros((n))#初始化用于保存Ltrain的值随迭代次数变化的数组

Ltest = np.zeros((n))#初始化用于保存Ltest的值随迭代次数变化的数组

def Lfun(w,X,y):#定义Loss函数

m = y.shape[0]

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

def DER(w,X,y):#定义计算梯度的函数

m = y.shape[0]

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

for i in range (n):#迭代若干次，更新模型参数w

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)

print(Ltrain)

print(Ltest)

print(w)

import matplotlib.pyplot as plt#绘制Ltrain和Ltest随迭代次数的变化图

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() #使用sklearn库的load\_svmlight\_file函数读取数据

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)) #把所有x的属性加上一个1，用于和w14（b）相乘

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) #使用train\_test\_split函数将数据集切分为训练集和验证集

w = np.zeros((13,1)) #将线性模型参数w初始化为全零

n = 100 #设置迭代次数

Ltrain = np.zeros((n)) #初始化用于保存Ltrain的值随迭代次数变化的数组

Ltest = np.zeros((n)) #初始化用于保存Ltest的值随迭代次数变化的数组

def Lfun(w,X,y): #定义Loss函数

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): #定义计算梯度的函数

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): #迭代若干次，更新模型参数w

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 #绘制Ltrain和Ltest随迭代次数的变化图

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：

多属性时等同于

（其中m为样本数,X为在原X右侧加了一列1的矩阵）

梯度： 的梯度

（对于w14(b)而言x=1）

则整个向量的梯度向量

## 11.实验结果和曲线图:

## 超参数选择（η,epoch等）：η选定为0.1

## 预测结果（最佳结果）：

(1)迭代100次后Ltrain=12.78114594, Ltest=12.33486608

W=[[-6.77395751]

[-0.65595281]

[-1.97776718]

[ 1.38495937]

[-2.68764612]

[ 6.91622697]

[ 0.07764992]

[-4.81254178]

[ 1.46780528]

[ 0.06077186]

[-4.26670389]

[ 3.52318838]

[-9.61890099]

[ 7.64213617]]

(2)迭代1000次后Ltrain= 11.4997037, Ltest=10.47714677

W=[[ -6.45099044]

[ 1.79664471]

[ 0.63151552]

[ 1.58878016]

[ -3.9350815 ]

[ 10.05697753]

[ -0.56498928]

[ -7.82749318]

[ 2.65439269]

[ -1.90187406]

[ -4.385118 ]

[ 2.33897269]

[-10.01785618]

[ 9.58123915]]

(3)迭代10000次后Ltrain=11.49250804, Ltest=10.36205064

W=[[ -5.7279196 ]

[ 1.89117106]

[ 0.7939804 ]

[ 1.61930757]

[ -3.92933801]

[ 10.18237179]

[ -0.62389036]

[ -7.82013895]

[ 2.7008918 ]

[ -2.15187957]

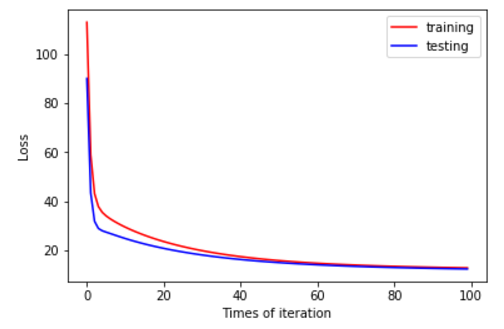
[ -4.36500898]

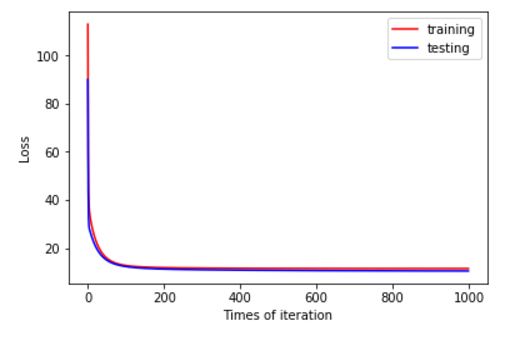
[ 2.33379957]

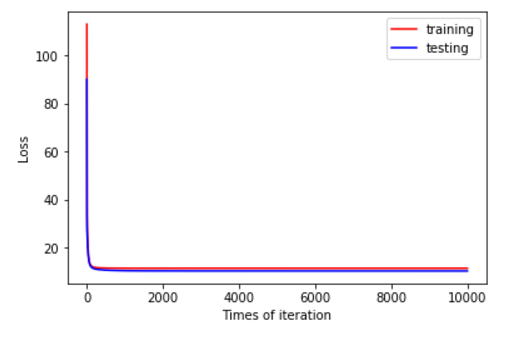
[ -9.92189906]

[ 10.4220418 ]]

## loss曲线图：







## 12.实验结果分析:

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，而在1000次迭代后Loss值降幅已经非常小了，但随着迭代一直表现为极小幅的下降，可以认为其有继续下降的趋势。

Ltrain和Lvalidation一直保持较小的差值，随着迭代的进行（特别是在50次迭代后）逐渐稳定在一个极小的范围内。

## 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

## 预测结果（最佳结果）：

(1)迭代100次后Ltrain=133.97088545, Ltest=73.05523108

W=[[-0.0119 ]

[ 0.01403777]

[ 0.08262953]

[ 0.0905 ]

[ 0.1862231 ]

[ 0.09315 ]

[ 0.04803665]

[ 1.0207 ]

[ 0.0863 ]

[ 0.03101641]

[-0.0145 ]

[ 0.0105 ]

[-0.0741274 ]

[-0.00325718]

[ 0.0345 ]]

(2)迭代1000次后Ltrain=130.84088513, Ltest=70.87565347

W=[[ 3.00000000e-04]

[ -8.98140099e-03]

[ 1.70236081e-03]

[ 1.89000000e-02]

[ 2.47919449e-02]

[ -2.42500000e-03]

[ 3.33375751e-03]

[ 1.00850000e+00]

[ 3.90000000e-03]

[ 6.49359695e-02]

[ 7.00000000e-04]

[ 7.60000000e-03]

[ -4.67837000e-02]

[ 4.90812320e-02]

[ 5.69000000e-02]]

(3)迭代10000次后Ltrain= 131.08894521 Ltest=70.77695065

W=[[ 1.20000000e-03]

[ -1.24921190e-02]

[ -7.20795873e-04]

[ 2.14000000e-02]

[ 2.53349705e-02]

[ -3.22500000e-03]

[ 2.59558715e-03]

[ 1.01080000e+00]

[ 4.40000000e-03]

[ 8.38702987e-02]

[ -2.00000000e-04]

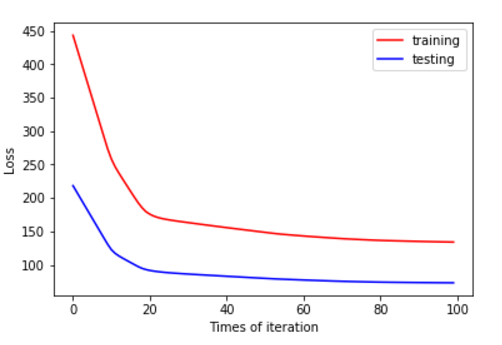
[ 6.00000000e-03]

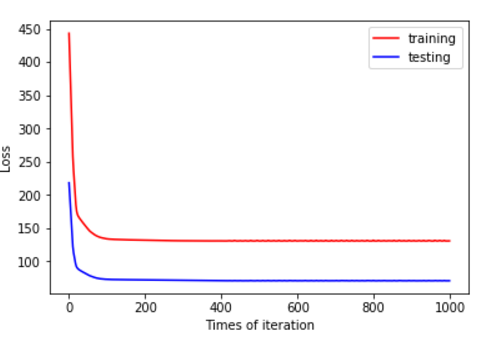
[ -4.45489000e-02]

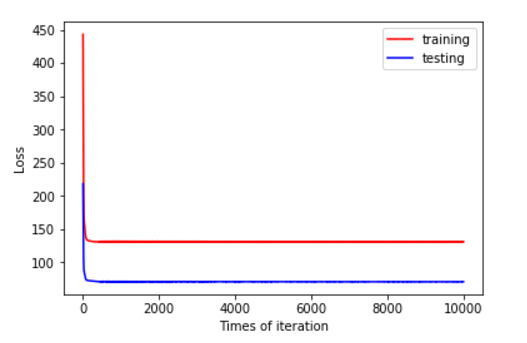
[ 1.67868254e-01]

[ 2.12400000e-01]]

## loss曲线图：







## 12.实验结果分析:

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。

由于训练集和验证集大小约为2：1，所以Ltrain和Lvalidation在整个迭代中基本保持相同的比例。

## 对比线性回归和线性分类的异同点：

线性回归和线性分类最基本的共同点就是通过

的线性公式构造模型。

线性分类的标签y取值是有限的几个（本实验为2个）离散值，但同一个值可以同时对应超空间中一定区域的x的取值，而线性回归中，只有很小的区域，甚至特定的x取值才对应一个y值。

从实验结果来看，线性分类的Loss值大小和数据集样本数始终成正比，对于不同大小的数据集，具有相同“训练程度”的模型产生不匹配的概率是一定的，所以数据集越大，Loss值越大。

## 实验总结：

本次实验让我更深地了解了通过数据样本训练特定机器学习模型的过程，此外还让我更好地掌握了python的语法和相关的包，并且让我巩固了线性代数相关的知识。

除了课堂已有的知识，实验还让我了解了模型变化对实验结果的巨大影响，包括模型结构、参数如学习率和迭代次数等。