

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员 陈安妮**

**学 号 201530611142**

**邮 箱 1412209084@qq.com**

**指导教师 吴庆耀**

**提交日期 2017年12月06 日**

## 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02 日

## 3. 报告人:

陈安妮（2015级5班）

## 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，属性取值为[-1,1]。本次实验我使用sklearn库的load\_svmlight\_file函数读取数据，再采用train\_test\_split函数随机将数据集67%切分为训练集，33%切分为验证集。

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，属性取值为[-1,1]，样本的值分为正类和负类两类{-1，+1}。本次实验我使用sklearn库的load\_svmlight\_file函数读取数据，再采用train\_test\_split函数随机将数据集67%切分为训练集，33%切分为验证集。

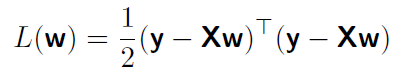
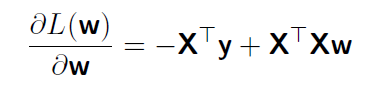
## 6. 实验步骤:

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

3.线性模型参数全1初始化。

4.选择Loss函数：并对其求导，得：。

5.求得所有样本对Loss函数的梯度G。取梯度G的负方向，记为D。

6.更新模型参数Wt = Wt-1 +ηD。η为学习率,是人为调整的超参数。

7.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。

8.重复步骤5-8若干次，画出和随迭代次数的变化图。

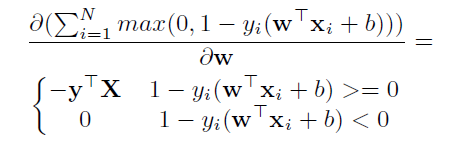
#### 6.2线性分类和梯度下降

1.读取实验数据，使用sklearn库的load\_svmlight\_file函数读取数据。

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

3.支持向量机模型参数全1初始化。

4.选择Loss函数：,对其求导，得：



5.求得所有样本对Loss函数的梯度G。

取梯度G的负方向，记为D。

6.更新模型参数Wt = Wt-1 +ηD。η为学习率,是人为调整的超参数。

7.选取阈值1，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。

8.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:

**RegressionExperiment.ipynb**

import numpy as np

import matplotlib.pyplot as plt

import random

from sklearn import datasets

from sklearn.datasets import load\_svmlight\_file

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

from scipy.sparse import csr\_matrix, hstack

#获取文件数据

def get\_data(file):

inputs,labels = datasets.load\_svmlight\_file(file) #读取文件

one = np.ones((inputs.shape[0],1))

inputs = hstack( [inputs, csr\_matrix(one)] ).toarray()

labels = np.array(labels).reshape(inputs.shape[0],1)

return inputs, labels

#训练

def train( X, Y, learning\_rate=0.001, epoch=1000):

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

list = [] #定义一个空列表

for i in range(epoch):

w -= learning\_rate \* np.dot(X.transpose(), X.dot(w) - Y )

Y\_prediction = X.dot(w)

loss=0.5/X.shape[0]\*np.dot((Y-Y\_prediction).transpose(), (Y-Y\_prediction) ).sum()

list.append(loss)

#print(list)

return w,list

#测试

def test( X\_test, Y\_test, w):

Y\_prediction = X\_test.dot(w)

loss = 0.5/X\_test.shape[0]\*np.dot((Y\_test-Y\_prediction).transpose(), (Y\_test - Y\_prediction) ).sum()

return loss

#迭代测试loss

def test\_loss\_list( X\_test, Y\_test, X\_train, Y\_train,learning\_rate, epoch):

loss\_list = []

for i in range(epoch):

w,train\_list = train(X\_train, Y\_train,learning\_rate, i)

loss = test(X\_test, Y\_test, w)

loss\_list.append(loss)

return loss\_list

#画图

def plot\_loss(train\_loss,test\_loss):

plt.xlabel('Iteration')

plt.ylabel('loss')

plt.plot(train\_loss, label="train\_loss",color="blue")

plt.plot(test\_loss, label="test\_loss",color="red")

plt.legend(loc='upper right')

plt.show()

if \_\_name\_\_ =="\_\_main\_\_":

learning\_rate = 0.0001

epoch= 1000

x,y = get\_data('housing\_scale')

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

w,train\_loss = train(x\_train, y\_train,learning\_rate, epoch)

test\_loss = test\_loss\_list( x\_test, y\_test,x\_train, y\_train, learning\_rate,epoch)

#print (test\_loss)

plot\_loss(train\_loss,test\_loss)

**ClassificationExperiment.ipynb**

import numpy as np

import matplotlib.pyplot as plt

import random

from sklearn import datasets

from sklearn.datasets import load\_svmlight\_file

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

from scipy.sparse import csr\_matrix, hstack

#获取文件数据

def get\_data(file):

inputs,labels = datasets.load\_svmlight\_file(file) #读取文件

one = np.ones((inputs.shape[0],1))

inputs = hstack( [inputs, csr\_matrix(one)] ).toarray()

labels = np.array(labels).reshape(inputs.shape[0],1)

return inputs, labels

#训练

def train( X, Y, learning\_rate=0.001, epoch=1000):

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

list = [] #定义一个空列表

for i in range(epoch):

llist = []

for j in range(X.shape[0]):

if (1 - np.dot(Y[j],X[j].dot(w).transpose()) >= 0):

gradients = - (Y[j]\*X[j].reshape(1,15).transpose())

else:

gradients = 0

w -= learning\_rate \* gradients

llist.append(max(1 - np.dot(Y[j],X[j].dot(w).transpose()),0))

loss = np.sum(llist)/ X.shape[0]

list.append(loss)

#print(list)

return w,list

#测试

def test( X\_test, Y\_test, w):

Y\_prediction =np.ones((Y\_test.shape[0],1))

for j in range( X\_test.shape[0]):

if ((X\_test[j].dot(w).transpose()) >= 0):

Y\_prediction [j] = 1

else :

Y\_prediction [j] = -1

loss = 1.0/ X\_test.shape[0]\* np.sum(abs(Y\_test - Y\_prediction))

return loss

#迭代测试loss

def test\_loss\_list( X\_test, Y\_test, X\_train, Y\_train,learning\_rate, epoch):

loss\_list = []

for i in range(epoch):

w,train\_list = train(X\_train, Y\_train,learning\_rate, i)

loss = test(X\_test, Y\_test, w)

loss\_list.append(loss)

return loss\_list

#画图

def plot\_loss(train\_loss,test\_loss):

plt.xlabel('Iteration')

plt.ylabel('loss')

plt.plot(train\_loss, label="train\_loss",color="blue")

plt.plot(test\_loss, label="test\_loss",color="red")

plt.legend(loc='upper right')

plt.show()

def plot\_test\_loss(test\_loss):

plt.xlabel('Iteration')

plt.ylabel('loss')

plt.plot(train\_loss, label="train\_loss",color="blue")

plt.legend(loc='upper right')

plt.show()

def plot\_train\_loss(train\_loss):

plt.xlabel('Iteration')

plt.ylabel('loss')

plt.plot(test\_loss, label="test\_loss",color="red")

plt.legend(loc='upper right')

plt.show()

if \_\_name\_\_ =="\_\_main\_\_":

learning\_rate = 0.001

epoch= 100

x,y = get\_data('australian\_scale')

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

w,train\_loss = train(x\_train, y\_train,learning\_rate, epoch)

test\_loss = test\_loss\_list( x\_test, y\_test,x\_train, y\_train, learning\_rate,epoch)

#print (test\_loss)

plot\_test\_loss(test\_loss)

plot\_train\_loss(train\_loss)

plot\_loss(train\_loss,test\_loss)

## 8. 选择的评估方法（留出法，交叉验证，k折交叉验证等）:

 线性回归和线性分类两个实验都是采用简单交叉验证（hold -out cross validation）的评估方法。

1.从全部数据集中随机选择67%的样例作为训练集，剩余的33%作为测试集。2.通过对测试集训练，分别得到线性回归模型和线性分类模型 。  
3.在测试集对每一个样本根据训练出来的模型进行预测，再与真实数据作比较，通过loss函数计算出loss值。

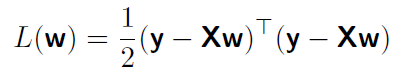
简单交叉验证评估方法由于其测试集和训练集是分开的，就避免了过拟合的现象。

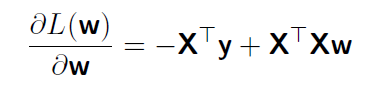
## 9. 模型参数的初始化方法:

 线性回归和线性分类两个实验都是采用全1初始化的方法，即在最初时，把未知参数w和b都全部声明为1，然后参与迭代运算。

## 10.选择的loss函数及其导数:

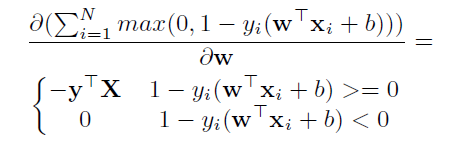
#### 10.1线性回归和梯度下降

选择Loss函数：

对其求导，得：

#### 10.2线性分类和梯度下降

选择Loss函数：

对其求导，得： 

## 11.实验结果和曲线图:

### 11.1线性回归和梯度下降

#### 11.1.1超参数选择（η,epoch等）：

η=0.0001

epoch=50

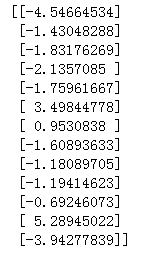
test\_size=0.33，train\_size=0.67

#### 11.1.2评估结果（根据选择的评估方法）：

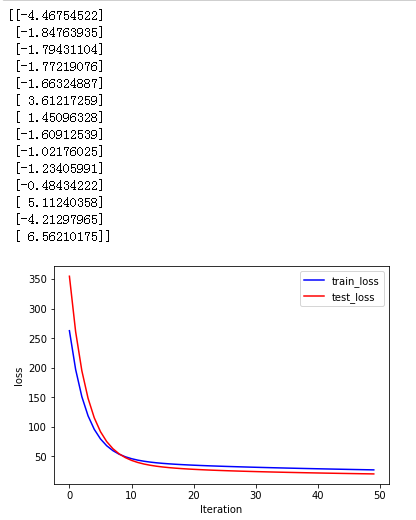
简单交叉验证，把数据集随机切分，选择67%的样例作为训练集，剩余的33%作为测试集。

#### 11.1.3预测结果（最佳结果）：

Y = WT+b,其中

W= b=6.68193109

#### 11.1.4loss曲线图：



η=0.0001，epoch=50，test\_size=0.33，train\_size=0.67，最佳。

### 11.2线性分类和梯度下降

#### 11.2.1超参数选择（η,epoch等）：

η=0.0001

epoch=300

test\_size=0.33，train\_size=0.67

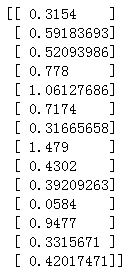
#### 11.2.2评估结果（根据选择的评估方法）：

简单交叉验证，把数据集随机切分，选择67%的样例作为训练集，剩余的33%作为测试集。

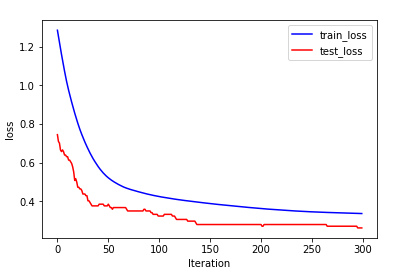
#### 11.2.3预测结果（最佳结果）：

Y=+1， WT+b≥1,

Y =-1， WT+b≤-1,其中

W =，b = 1.6772

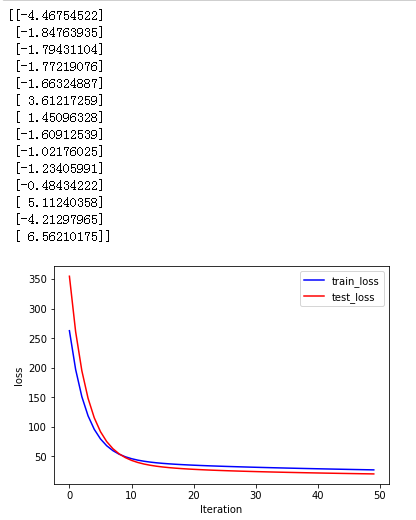
#### 11.2.4loss曲线图：



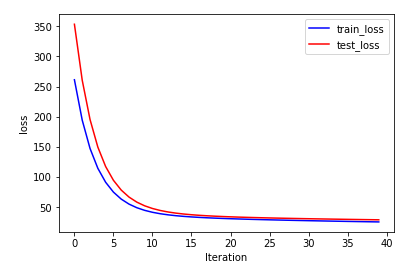
η=0.0001，epoch=300，test\_size=0.33，train\_size=0.67，最佳。

## 12.实验结果分析:

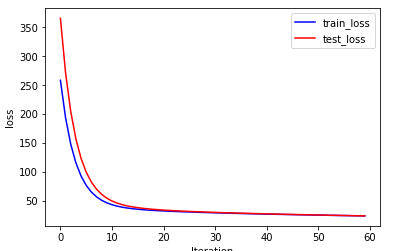
### 12.1线性回归和梯度下降



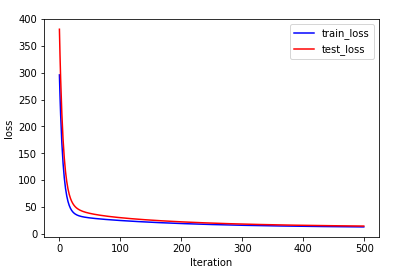
η=0.0001，epoch=50，test\_size=0.33，train\_size=0.67，最佳。



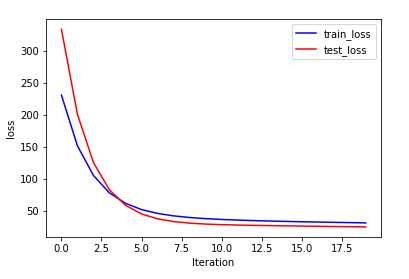
η=0.0001，epoch=40，test\_size=0.33，train\_size=0.67.



η=0.0001，epoch=60，test\_size=0.33，train\_size=0.67.



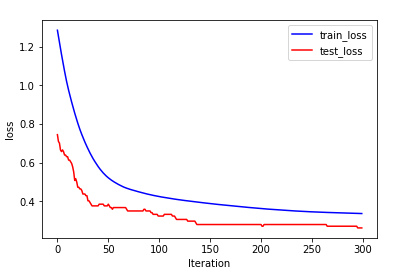
η=0.00005，epoch=500，test\_size=0.33，train\_size=0.67，loss与η=0.0001接近，但迭代次数多，运行较慢。



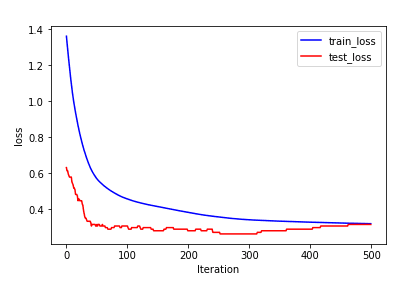
η=0.00015，epoch=20，test\_size=0.33，train\_size=0.67，迭代次数较少，运行速度快，但loss大于η=0.0001，拟合效果较差。

综合以上几种情况，最优解可能为η=0.0001，epoch=50，test\_size=0.33，train\_size=0.67。

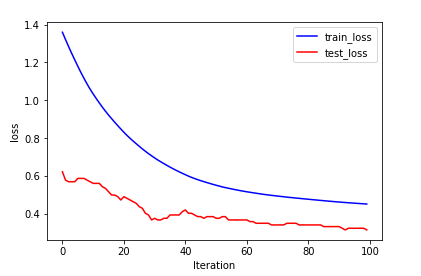
### 12.2线性分类和梯度下降



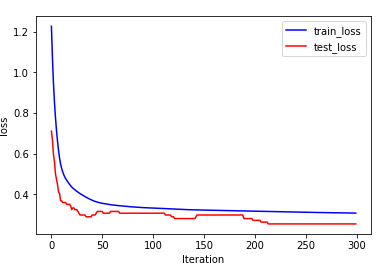
η=0.0001，epoch=300，test\_size=0.33，train\_size=0.67，最佳。



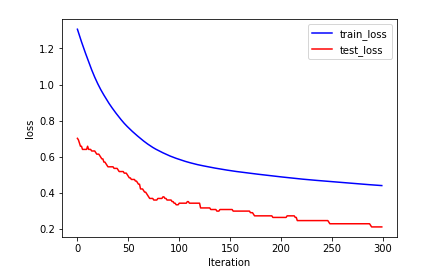
η=0.0001，epoch=500，test\_size=0.33，train\_size=0.67，显然train\_loss和test\_loss在epoch=300就已经达到最低了，当迭代次数超过300时，出现了过拟合现象，test\_loss增大。



η=0.0001，epoch=100，test\_size=0.33，train\_size=0.67，显然train\_loss和test\_loss还未收敛完全。



η=0.0005，epoch=300，test\_size=0.33，train\_size=0.67，由图可知迭代次数较少（大约250次就收敛了），运行速度快，但loss大于η=0.0001，且test\_loss的曲线波动比较大，拟合效果较差。



η=0.00005，epoch=300，test\_size=0.33，train\_size=0.67，loss与η= 0.0001接近（比η= 0.0001小），但是明显迭代次数多，由上图可知，要使其收敛epoch>300，运行速度较慢。

综合以上几种情况，最优解可能为η=0.0001，epoch=300，test\_size=0.33，train\_size=0.67。

## 13.对比线性回归和线性分类的异同点：

对于线性回归和线性分类这两类问题都有以下几个步骤,

1.选取一个合理的线性模型，例如一次函数等。

2.选择一个"美好"的loss函数 (可以评估拟合程度，而且还是convex函数，这样才能求导)

3.采取一切可能的技术(例如，梯度下降法，随机梯度下降法，解极值方程法等) 求出最好的模型参数。

线性回归和线性分类的区别：

总的来说两个问题本质上都是一致的，就是线性模型的拟合（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).

## 14.实验总结：

本次实验使我进一步理解线性回归和线性分类的原理，同时也更加熟练的掌握了梯度下降调参的原理和方法。通过在LIBSVM Data中的Housing数据集上应用线性回归的算法，和在LIBSVM Data中的australian数据集上应用线性分类算法，我体会到了优化和调参的过程。

但是，在这次实验中，我也遇到了不少难题。其中最为突出的就是优化和调参。优化模型一般是对模型的学习率，迭代次数epoch以及数据划分比例进行调参的，而这三个参数都是相互影响的，而且调整的步长也是不确定的。但是本次实验中，我采取的是固定学习率和数据划分比例，调整迭代次数，因为学习率影响的是收敛时间，所以我假设了一个较小的学习率，然后调整迭代次数，而数据划分比例的影响较小，我就只是调了几个数值进行比较。我的调参能力还需要继续加强！