

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员 李鸿境**

**学 号 201530081266**

**邮 箱 [420962182@qq.com](mailto:420962182@qq.com)**

**指导教师** **吴庆耀**

**提交日期** **2017年 12 月 6 日**

## 实验题目:

线性回归、线性分类与梯度下降

## 实验时间：2017年 12月 2日

## 3. 报告人:

## 李鸿境

## 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。

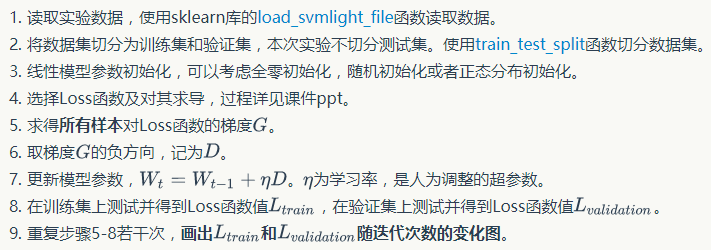
## 数据集以及数据分析：

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。( housing\_scale.txt )

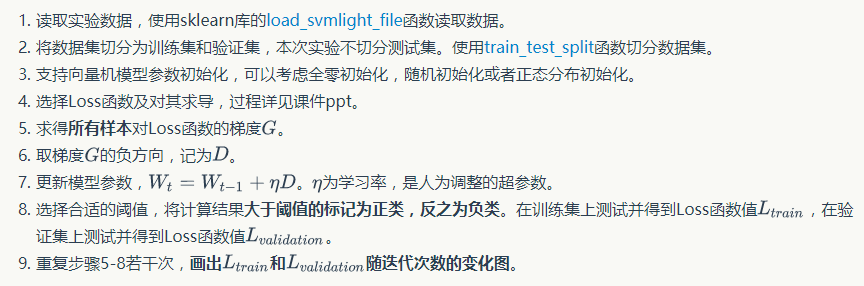
线性分类使用的是LIBSVM Data中的australian数据，包含690个样本，每个样本有14 个属性。( australian\_scale.txt )

## 实验步骤:

1. 线性回归和梯度下降：



（2）线性分类和梯度下降：



## 7. 代码内容:

（针对线性回归和线性分类分别填写8-12内容）

1. ：线性回归和梯度下降：

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cross\_validation import train\_test\_split

from sklearn.datasets import load\_svmlight\_file

learningRate=0.38

G=np.zeros((14,1))

D=np.zeros((14,1))

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

X, y = data[0], data[1]

X=X.dot(np.eye(13))

b=np.ones((506,1))

X=np.column\_stack((X,b))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1)

w=np.zeros((14,1))

n\_train=X\_train.shape[0]

n\_test=X\_test.shape[0]

y=y.reshape(506,1)

y\_train=y\_train.reshape(354,1)

y\_test=y\_test.reshape(152,1)

L\_train=np.zeros(200)

L\_test=np.zeros(200)

for i in range(200):

for j in range(14):

G[j]=(((X\_train.dot(w)-y\_train).T).dot(X\_train[:,j]))/n\_train

D=-G

w=w+learningRate\*D

L\_train[i]=((X\_train.dot(w)-y\_train).T).dot(X\_train.dot(w)-y\_train)/2/n\_train

L\_test[i]=((X\_test.dot(w)-y\_test).T).dot(X\_test.dot(w)-y\_test)/2/n\_test

#print("第",i,"次：\n","L\_train:",L\_train[i]," L\_test:",L\_test[i])

print(L\_train[199],"\n",L\_test[199])

x=np.arange(0,200,1)

%matplotlib inline

plt.plot(x,L\_train,'r',label='L\_train')

plt.plot(x,L\_test,'g',label='L\_test')

plt.legend(loc='upper right')

plt.xlabel('iterator times (learningRate=0.38)')

plt.ylabel('Loss')

（2）线性分类和梯度下降：

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cross\_validation import train\_test\_split

from sklearn.datasets import load\_svmlight\_file

G=np.zeros((15,1))

D=np.zeros((15,1))

L\_train=np.zeros((800,1))

L\_test=np.zeros((800,1))

C = 2

learningRate=0.015

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

X, y = data[0], data[1]

X=X.dot(np.eye(14))

b=np.ones((690,1))

X=np.column\_stack((X,b))

#多次试验后，random\_state为10的时候准确率true最高

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=10)

w=np.zeros((15,1))

#690

m=X.shape[0]

#483

n\_train=X\_train.shape[0]

#207

n\_test=X\_test.shape[0]

y = y.reshape(690,1)

y\_train = y\_train.reshape(483,1)

y\_test = y\_test.reshape(207,1)

for i in range(800):

M\_train = 1-y\_train\*(X\_train.dot(w))

M\_train[M\_train<0] = 0

M\_test = 1-y\_test\*(X\_test.dot(w))

M\_test[M\_test<0] = 0

for j in range(n\_train):

if M\_train[j]>0:

G = G+(y\_train[j]\*X\_train[j,:]).reshape(15,1)

G=w-C\*G/n\_train

D=-G

w=w+learningRate\*D

#hinge损失

L\_train[i] = sum(w\*w)/2 + C\*sum(M\_train)

L\_test[i] = sum(w\*w)/2 + C\*sum(M\_test)

f = X\_test.dot(w)

for k in range(n\_test):

if f[k]>=0:

f[k]=1

else:

f[k]=-1

true=0

for j in range(n\_test):

if f[j]==y\_test[j]:

true = true + 1;

T = true/n\_test

x = np.arange(0,800,1)

%matplotlib inline

plt.plot(x,L\_train,'r',label='L\_train')

plt.plot(x,L\_test,'g',label='L\_test')

plt.legend(loc='upper right')

plt.xlabel('iterator times')

plt.ylabel('Loss')

print(T)

***对于线性回归：***

## 选择的评估方法（留出法，交叉验证，k折交叉验证等）:

交叉验证（70%作为训练集，30%作为测试集）

## 模型参数的初始化方法:

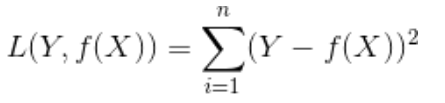
全零初始化

## 选择的loss函数及其导数:

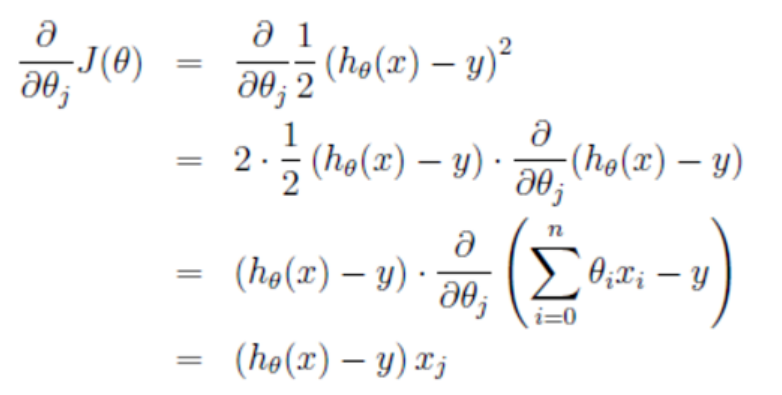
Loss函数：

平方损失函数（最小二乘法, Ordinary Least Squares ）

公式：



导数（梯度函数）：



11.实验结果和曲线图:

## 超参数选择（η,epoch等）：

## η = 0.1，0.2， 0.3， 0.4, 0.35

## 评估结果（根据选择的评估方法）：

(Loss指的是当迭代200次后的loss结果）

## η=0.1时，训练集Loss为12.298424 ，测试集为10.262706

η=0.2 时，训练集Loss为11.933770 ，测试集为9.931992

η=0.3 时，训练集Loss为11.829032 ，测试集为9.876990

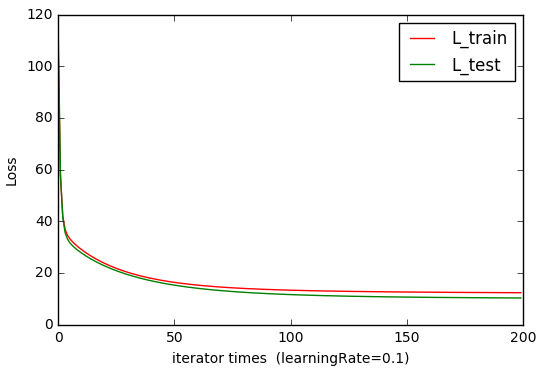
η=0.4 时，训练集Loss为11.790224 ，测试集为9.871606

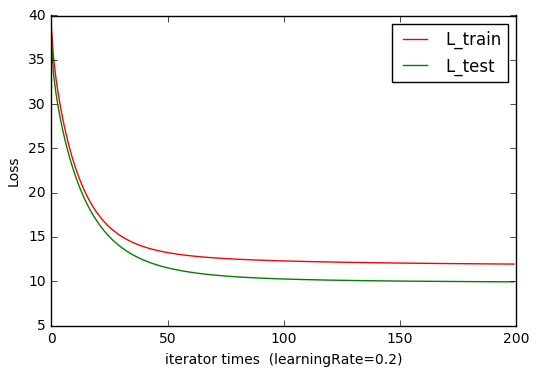
η=0.35 时，训练集Loss为11.805216 ，测试集为9.871377

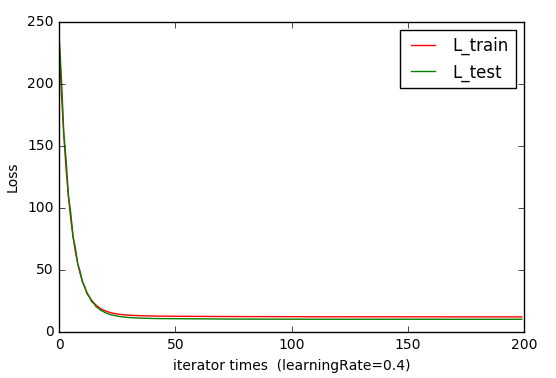
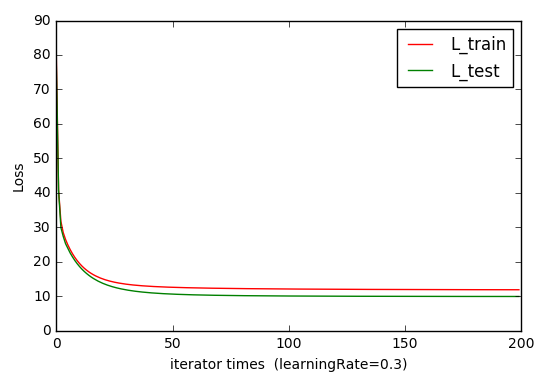
## 预测结果（最佳结果）：

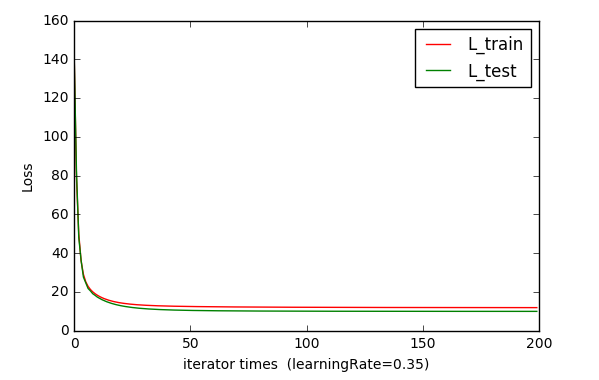
当η 在0.35-0.4之间时，loss达到最小

## loss曲线图：









***对于线性分类：***

## 8.选择的评估方法（留出法，交叉验证，k折交叉验证等）:

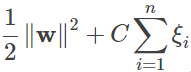
交叉验证（70%作为训练集，30%作为测试集）

## 9.模型参数的初始化方法:

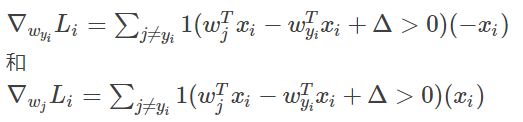
全零初始化

## 10.选择的loss函数及其导数:

Hinge损失函数：



导数（梯度函数）：



11.实验结果和曲线图:

## 超参数选择（η,epoch等）：

C=2

## η = 0.02， 0.1， 0.05，

## 评估结果（根据选择的评估方法）：

(Loss指的是当迭代200次后的loss结果，T指对于测试集分类准确率）

## η=0.02时，训练集Loss为366.067915 ，测试集为124.291302

T = 0.8840579710144928

η=0.1 时，训练集Loss为3780929442 ，测试集为131.011128

T = 0.8840579710144928

η=0.05 时，训练集Loss为368.292914 ，测试集为125.900621

T = 0.8840579710144928

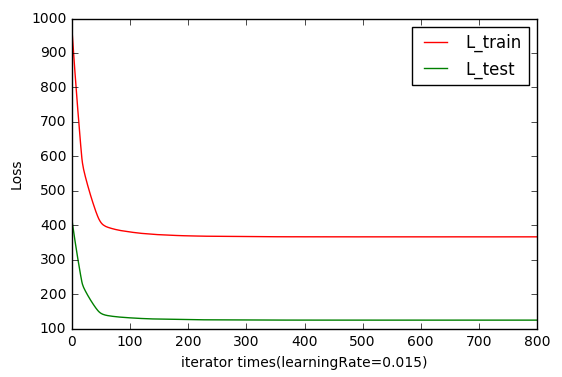
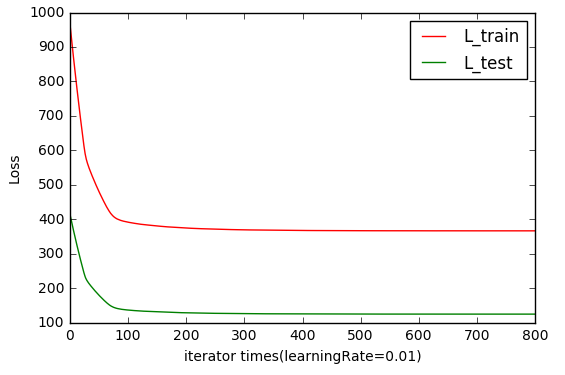
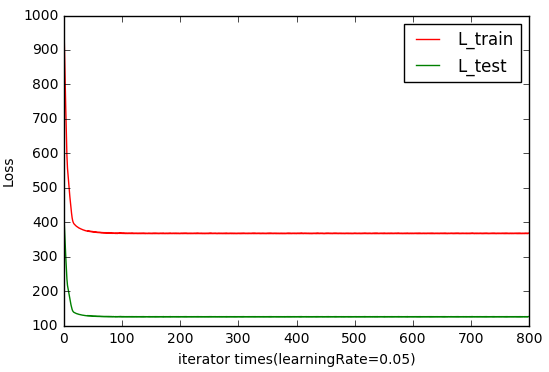
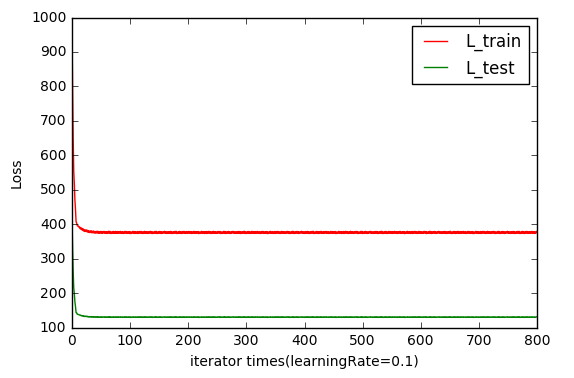
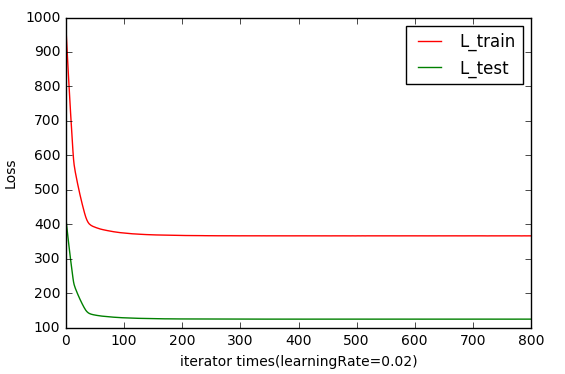
η=0.01 时，训练集Loss为366.120561 ，测试集为124.320666

η=0.015 时，训练集Loss为366.107298 ，测试集为124.313064

## 预测结果（最佳结果）：

当η 在0.01-0.03之间时，loss达到最小

## loss曲线图：



## 实验结果分析:

实验结果表面，学习率的改变会影响到loss函数，但在线性分类中对于分类准确率基本不影响。经过多次迭代后，loss趋于稳定。

## 13.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).

## 14.实验总结：

经过这次实验，对于机器学习的应用有了初步的理解，也算是处于入门阶段。但目前所掌握的知识不多，在接下来的阶段中会不断学习，不断进步，熟悉这个领域的知识。