

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  **蔡兴阳**

**学 号 201530611074**

**邮 箱 779762069@qq.com**

**指导教师**  **吴庆耀**

**提交日期** **2017年 12 月 8 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

## 报告人:

## 蔡兴阳

## 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。

线性分类使用的是LIBSVM Data中的australian数据，包好690个样本，每个样本有14个属性。并将其切分为训练集，验证集

## 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。使用train\_test\_split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G。

6. 取梯度G的负方向，记为D。

7. 更新模型参数。

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，画出两个Loss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。使用train\_test\_split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G。

6. 取梯度的负方向G，记为D。

7. 更新模型参数。

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，画出两个Loss随迭代次数的变化图。

## 代码内容:

**线性回归：**

def cal\_loss(a,b,theta):

return (1/2 \*(np.linalg.norm(a\*theta-b))\*\*2)/ (a.shape[0])

train\_loss = []

test\_loss = []

for i in range(0, maxIteration):

gradient =2\* X\_train.T \* (X\_train \* theta - y\_train)

theta = theta - alpha \* gradient

train\_loss.append(cal\_loss(X\_train,y\_train,theta))

test\_loss.append(cal\_loss(X\_test,y\_test,theta))

**线性分类：**

def gradientDescent(w):

for i in range(maxIteration):

current\_train\_loss,gradient=cal\_hinge\_loss\_and\_grad(w,X\_train,y\_train)

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

test\_loss.append( cal\_hinge\_loss(w,X\_test,y\_test))

w = w-learning\_rate\*gradient.T

train\_accr.append(cal\_accur(X\_train,y\_train,w))

test\_accr.append(cal\_accur(X\_test,y\_test,w))

gradientDescent(theta)

## 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:**

留出法：划分77%的样本为训练集和33%的样本为测试集



**线性分类:**

留出法：划分的60%样本为训练集和40%的样本为测试集



## 模型参数的初始化方法:

线性回归：

全1初始化：theta = np.ones((n, 1))

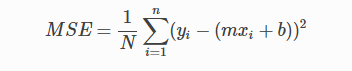
线性分类：

全1初始化：theta = np.ones((X.shape[1], 1))

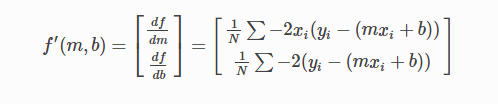
## 选择的loss函数及其导数:

**线性回归：**

Loss函数：

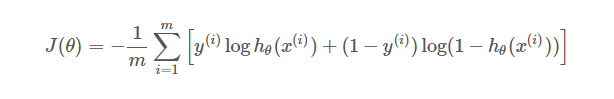


Loss函数导数：

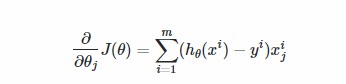


**线性分类：**

Loss函数：



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：

η = 0.00001

epoch = 1000

线性回归：

η= 0.0005

epoch = 100

C = 0.5

## 评估结果（根据选择的评估方法）：

线性回归：

Lossmin = 12.9073

线性分类：

Lossmin = 0.31072

## 预测结果（最佳结果）：

线性回归：

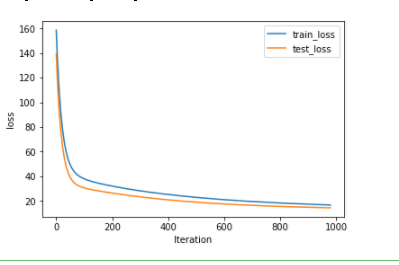
Lossmin = 11.0944

线性分类：

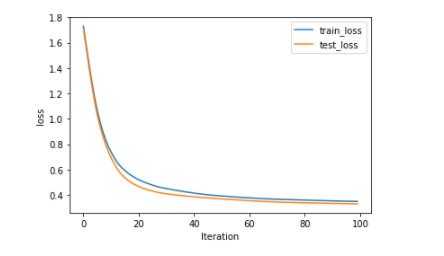
Lossmin= 0.3418

## loss曲线图：

线性回归：



**线性分类：**



## 实验结果分析:

梯度下降非常平稳，这意味着下降过程满足本实验的要求，而且损失的下降过程和评估损失相似，这意味着选择的超参数是合适的。

## 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.

## 实验总结：

通过本次实验，对python的基本语法有了更深入的了解，踩了许多类型转换的坑。对梯度下降、线性回归、svm、线性分类有了更深入的了解。