

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  **张梓佳**

**学 号 201530613726**

**邮 箱 969035716@qq.com**

**指导教师**  **吴庆耀**

**提交日期** **2017年 12月 8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 3日

## 3. 报告人:张梓佳

## 4. 实验目的: 在本次实验中，实验在于进一步理解线性回归和线性分类以及梯度下降，并在小规模的数据集上进行实践并体会优化和调参的过程

## 5. 数据集以及数据分析：在线性回归中，使用的是LIBSVM Data 中的Housing\_scale版本数据，实验中将506个样本（每个样本包含13个属性）并以0.33的比例切分为训练集和验证集。而在线性分类中使用的是LIBSVM Data中的astranlian\_scale版本数据，实验中将690个样本（每个样本包含14个属性）并同样以0.33的比例切割数据集为训练数据集和测试数据集

## 6. 实验步骤:

## （1） 线性回归

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

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。

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。

选择Loss函数及对其求导，过程详见课件ppt。

求得所有样本对Loss函数的梯度。

取梯度的负方向，记为。

更新模型参数，。为学习率，是人为调整的超参数。

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。

重复步骤5-8若干次，画出和随迭代次数的变化图。

**（2） 线性分类**

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

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。

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。

选择Loss函数及对其求导，过程详见课件ppt。

求得所有样本对Loss函数的梯度。

取梯度的负方向，记为。

更新模型参数，。为学习率，是人为调整的超参数。

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。

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:

**线性回归关键代码：**

**第一段：**

gradient=-np.dot(X\_train.T,Y\_train.T)+np.dot(np.dot(X\_train.T, X\_train),init\_w)

**第二段：**

for loss in range(iter\_number):

w=w-gradient\*learning\_rate

gradient=-np.dot(X\_train.T,Y\_train.T)+np.dot(np.dot(X\_train.T, X\_train),w)

#求损失函数

loss\_train=np.dot((np.dot(X\_train,w)-Y\_train.T).T,(np.dot(X\_train,w)-Y\_train.T))

loss\_test=np.dot((np.dot(X\_test,w)-Y\_test.T).T,(np.dot(X\_test,w)-Y\_test.T))

l1=loss\_train.tolist()

l2=loss\_test.tolist()

train\_list.append(l1[0][0])

test\_list.append(l2[0][0])

线性分类关键代码：

第一段：

lj1=0

#梯度求解

for i in range(0,462):

lj1=lj1+gw(X\_train[i], y\_train[i], init\_w)

gradients = init\_C \* lj1

gradient=init\_w+gradients

**第二段：**

for loss in range(iter\_number):

w=w-gradient\*learning\_rate

m1 = 0

m2 = 0

lj1 = 0

for i in range(0,462):

lj1 = lj1 + gw(X\_train[i], y\_train[i], w)

try1=gg(X\_train[i],y\_train[i],w)

max1=max(0,try1)

m1=m1+max1

gradient = w + init\_C \* lj1

for i in range(0,228):

try2=gg(X\_train[i],y\_train[i],w)

max2=max(0,try2)

m2=m2+max2

loss\_train=np.dot(w.T,w)/2+init\_C\*m1

loss\_test=np.dot(w.T,w)/2+init\_C\*m2

l1 = loss\_train.tolist()

l2 = loss\_test.tolist()

train\_list.append(l1[0][0])

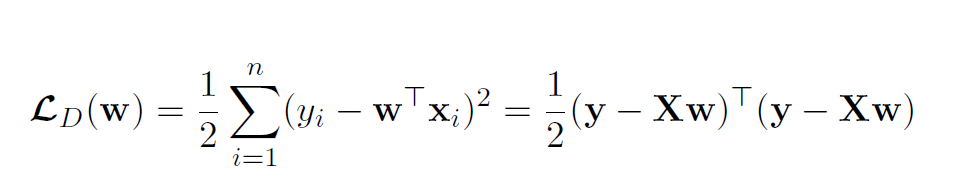
test\_list.append(l2[0][0])

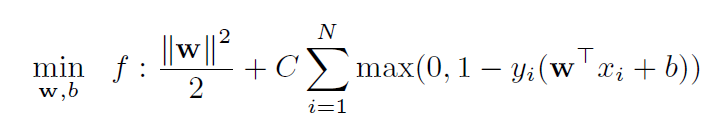
（针对线性回归和线性分类分别填写8-12内容）

## 8. 选择的评估方法：均为交叉验证法

## 9. 模型参数的初始化方法:

## 10.选择的loss函数及其导数:

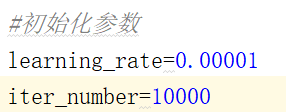
线性回归：

线性分类：

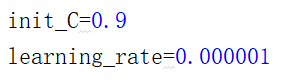
## 11.实验结果和曲线图:

## 超参数选择（η,epoch等）：

**线性回归：**



**线性分类：**

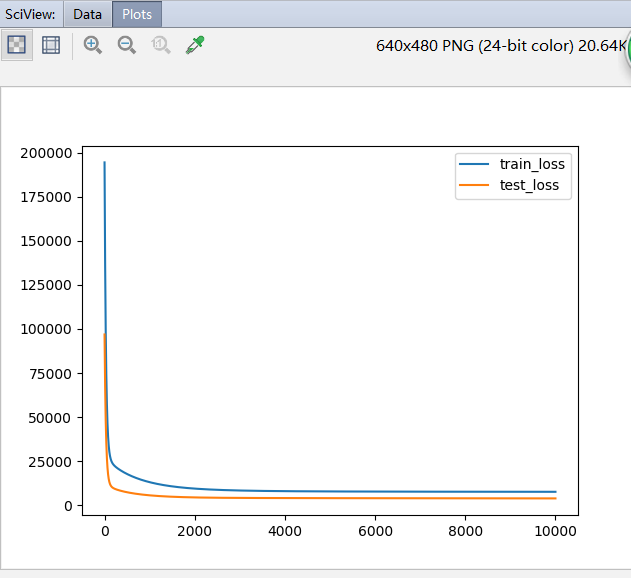


## 评估结果（根据选择的评估方法）：

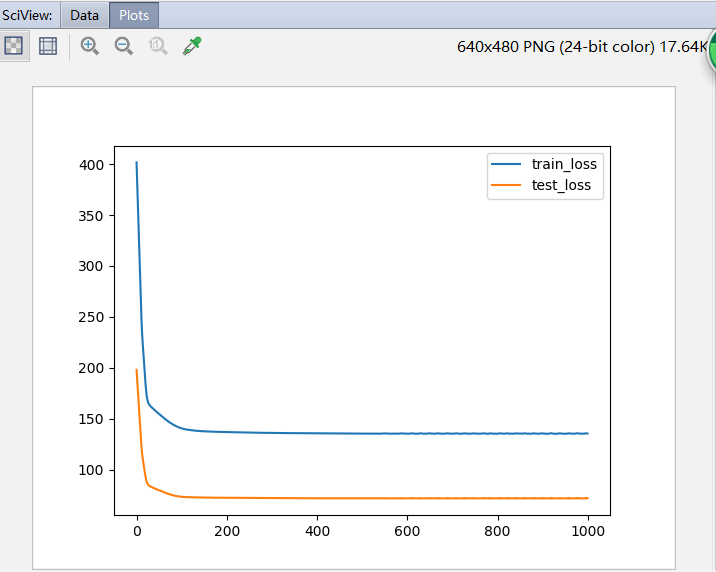
## 预测结果（最佳结果）：

## loss曲线图：

**线性回归：**



**线性分类：**



## 12.实验结果分析: 在线性回归中，其loss函数的结果在迭代初期出现急剧下降的趋势，随着迭代次数的增多，loss函数开始进入十分缓慢的下降阶段（区域最优？） 在线性分类中，同样在初始对w和b值优化是的效果最明显，而随着迭代次数的增多，w和b值逐渐趋于平稳。

## 13.对比线性回归和线性分类的异同点：

相同之处：在线性回归和线性分类中均先全零初始化参数w和b的值（具体实验中将w 和b合并为一起计算）之后通过梯度下降的方法实现w和b值的优化。并将xi矩阵加一列（以对应合并的w和b矩阵）

差异之处：计算线性回归的梯度时我们只需要将整个训练集矩阵带入计算，而计算线性分类的梯度时我们需要对每一个训练姐矩阵的每一行逐一计算再相加，相比之下线性分类的计算更加复杂。

## 14.实验总结：

通过本次实验，较好地掌握了通过梯度下降方法对线性分类和线性回归进行优化的步骤。在今后的实验中将逐步尝试其他更多的方法进行线性分类和线性回归的实现