

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  王佳晨

**学 号 201530612859**

**邮 箱 1379666872@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 2日

## 3. 报告人:王佳晨

## 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](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/" \t "_blank)中的[Housing](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html" \l "housing" \t "_blank)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](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/" \t "_blank)中的[australian](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html" \l "australian" \t "_blank)数据，包含690个样本，每

个样本有14个属性。label的值为+1或-1

数据集使用sklearn库的load\_svmlight\_file函数读取，返回的数据是稀疏的

矩阵，所以需要转化为稠密矩阵。

## 6. 实验步骤:

一、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， 采用全零初始化。

4、选择Loss函数为 ,对其求导得到

5、求得所有样本对Loss函数的梯度

6、取梯度G的负方向，记为D

7、更新模型参数，。为学习率。

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。

二、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、线性模型参数初始化， 采用全零初始化。

4、选择Loss函数为,对其求导得到

5、求得所有样本对Loss函数的梯度

6、取梯度G的负方向，记为D

7、更新模型参数，。为学习率。

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。

## 7. 代码内容:

一、 线性回归

1、读取数据

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

x=data[0].todense()

y=data[1]

2、分割数据集

X\_train,X\_validation,y\_train,y\_validation=train\_test\_split(x,y,test\_size=0.2,

random\_state=100)

3、数据的预处理

X\_train=X\_train.T#x为列向量

x\_0=np.ones(X\_train.shape[1])#x0=1

X\_train=np.row\_stack((X\_train,x\_0))

y\_train=np.mat(y\_train)#y为行向量

X\_validation=X\_validation.T#x为列向量

x\_0=np.ones(X\_validation.shape[1])#x0=1

X\_validation=np.row\_stack((X\_validation,x\_0))

y\_validation=np.mat(y\_validation)#y为行向量

4、定义Loss函数

def compute\_loss(w,x,y):

totalError = 0

y\_pre=w.T\*x

for i in range(y.shape[1]):

totalError+=((y[0,i]-y\_pre[0,i])\*\*2)/2.0

return (totalError/float(x.shape[1]))

5、定义计算梯度并更新参数的函数

def compute\_gradient(w\_current,x,y,learning\_rate):

w\_gradient = np.mat(np.zeros(x.shape[0])).T

N = float(x.shape[1])

X=x.T#每行对应一个样本

w\_gradient=2\*(X.T\*(X\*w\_current-y.T))

new\_w = w\_current-(learning\_rate \* w\_gradient)

return new\_w

6、训练模型并计算出训练集与验证集的LOSS

learning\_rate = 0.00001

initial\_w = np.mat(np.zeros(X\_train.shape[0])).T

num\_iter = 5000

loss\_train=[]

loss\_validation=[]

loss\_train.append(compute\_loss(initial\_w,X\_train,y\_train))

loss\_validation.append(compute\_loss(initial\_w,X\_validation,y\_validation))

#train model

w=initial\_w

for i in range(num\_iter):

w=compute\_gradient(w,X\_train,y\_train,learning\_rate)

loss\_train.append(compute\_loss(w,X\_train,y\_train))

loss\_validation.append(compute\_loss(w,X\_validation,y\_validation))

7、画出训练集与验证集LOSS随迭代次数变化的曲线

plt.figure(figsize=(20,8))

plt.plot(loss\_train,color='red',label='Loss-train')

plt.plot(loss\_validation,color='blue',label='Loss-validation')

plt.xlabel('iteration number')

plt.ylabel('loss')

plt.legend(fontsize=20)

plt.show()

二、线性分类

1、读取数据

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

X=data[0].todense()

y=data[1]

2、分割数据集

X\_train,X\_validation,y\_train,y\_validation=train\_test\_split(X,y,test\_size=0.2,

random\_state=100)

3、数据预处理

#x转为列向量

X\_train=X\_train.T

X\_validation=X\_validation.T

#x\_0 = 1

x\_0=np.ones(X\_train.shape[1])

X\_train=np.row\_stack((X\_train,x\_0))

x\_0=np.ones(X\_validation.shape[1])

X\_validation=np.row\_stack((X\_validation,x\_0))

y\_train=np.mat(y\_train)

y\_validation=np.mat(y\_validation)

4、定义LOSS函数

def Loss(x,y,w):

loss=0

for i in range(x.shape[1]):

if (1-y[0,i]\*(w.T\*x[:,i]))>0:

loss+=1-y[0,i]\*(w.T\*x[:,i])

loss+=(w.T\*w)/2

return loss[0,0]

5、定义计算accuracy的函数

def accuracy(x,y,w):

correct=0

for i in range(x.shape[1]):

if (y[0,i]\*(w.T\*x[:,i]))>0:

correct+=1

return correct/float(x.shape[1])

6、定义计算梯度并更新参数的函数

def compute\_gradient(w\_current,x,y,learning\_rate):

w\_gradient=np.zeros(x.shape[0])

w\_gradient=np.mat(w\_gradient).T

for i in range(x.shape[1]):

if (1-y[0,i]\*(w\_current.T\*x[:,i]))>0:

w\_gradient+=-y[0,i]\*x[:,i]

w\_gradient+=w\_current

new\_w=w\_current-learning\_rate\*w\_gradient

return new\_w

7、训练模型并计算LOSS与accuracy

#初始化参数

W=np.ones(X\_train.shape[0])

W=np.mat(W).T

learning\_rate=0.0001

num\_iter=2000

Loss\_train=[]

Loss\_validation=[]

accuracy\_train=[]

accuracy\_validation=[]

Loss\_train.append(Loss(X\_train,y\_train,W))

Loss\_validation.append(Loss(X\_validation,y\_validation,W))

accuracy\_train.append(accuracy(X\_train,y\_train,W))

accuracy\_validation.append(accuracy(X\_validation,y\_validation,W))

for i in range(num\_iter):

W=compute\_gradient(W,X\_train,y\_train,learning\_rate)

Loss\_train.append(Loss(X\_train,y\_train,W))

Loss\_validation.append(Loss(X\_validation,y\_validation,W))

accuracy\_train.append(accuracy(X\_train,y\_train,W))

accuracy\_validation.append(accuracy(X\_validation,y\_validation,W))

8、画出分别LOSS和accuracy随迭代次数变化的曲线

plt.figure(figsize=(18,6))

plt.plot(Loss\_train,color='red',label='Loss-train')

plt.plot(Loss\_validation,color='blue',label='Loss-validation')

plt.xlabel('iteration number')

plt.ylabel('loss')

plt.legend(fontsize=20)

plt.show()

plt.figure(figsize=(18,6))

plt.plot(accuracy\_train,color='red',label='accuracy-train')

plt.plot(accuracy\_validation,color='blue',label='accuracy-validation')

plt.xlabel('iteration number')

plt.ylabel('loss')

plt.legend(fontsize=20)

plt.show()

## 8. 选择的评估方法（留出法，交叉验证，k折交叉验证等）:

这次实验采用留出法作为评估方法，随机将数据集划分为两个互斥的集合，其中一个作为训练集，另一个用来模型评估。

## 9. 模型参数的初始化方法:

在线性回归中采用全为零的初始化方法，

在线性分类中采用全为一的初始化方法。

## 10.选择的loss函数及其导数:

线性回归的loss函数为 其导数为

线性分类的loss函数为 其导数为

## 11.实验结果和曲线图:

## 超参数选择（η,epoch等）：

线性回归参数：学习率= 0.00001，迭代次数 = 5000

线性分类参数：学习率= 0.0001，迭代次数=2000，正则化常数C=1，L2正

则系数=1

## 评估结果（根据选择的评估方法）：

线性回归留出法：验证集loss与训练集loss基本相等，没有出现验证集loss与训练集loss相差较大的情况，模型泛华能力较好。

线性分类留出法：验证集准确率与训练集准确率在迭代次数较小时有较大的差别，迭代次数大时，相差不大，但在准确率很小的范围内波动。

## 预测结果（最佳结果）：

线性回归：学习率= 0.00001，迭代次数 = 5000 时，训练集 mean square

error = 10.9888102744，验证集 mean square error= 11.6955694087

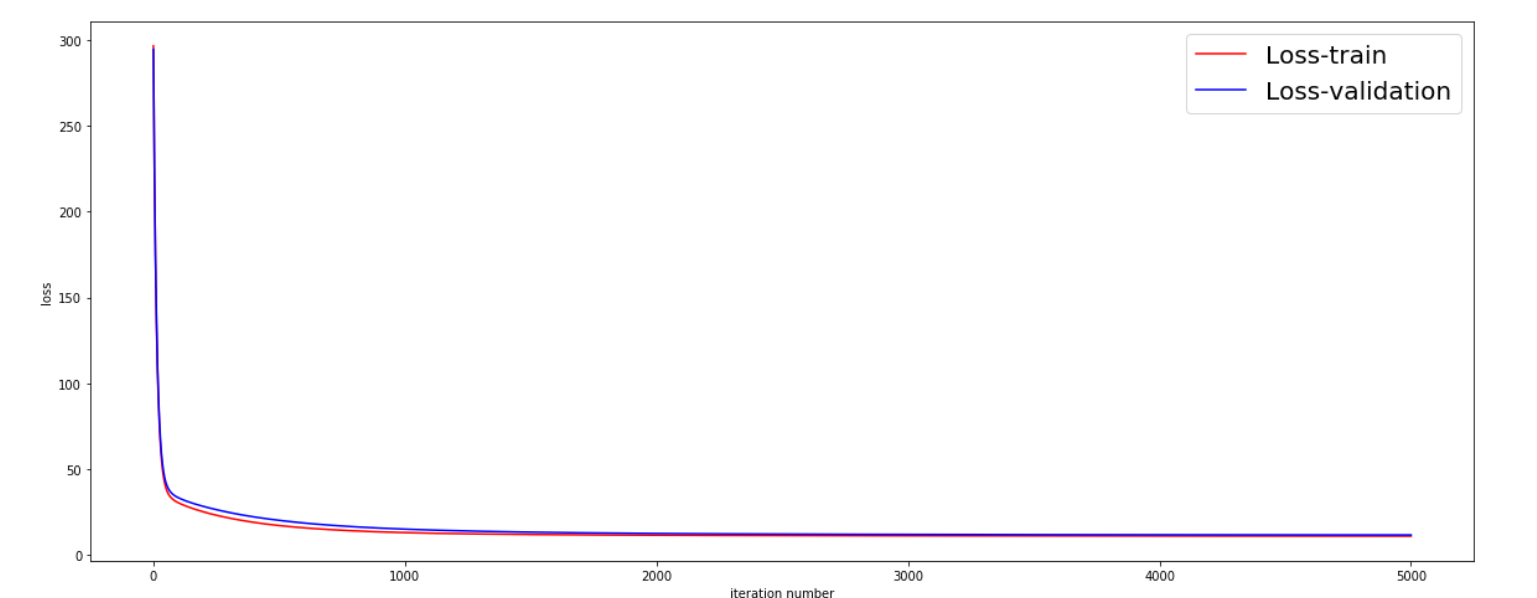
线性分类：学习率= 0.0001，迭代次数=2000，正则化常数C=1，L2正

则系数=1时，训练集准确率= 0.855072463768116，验证集准确率=

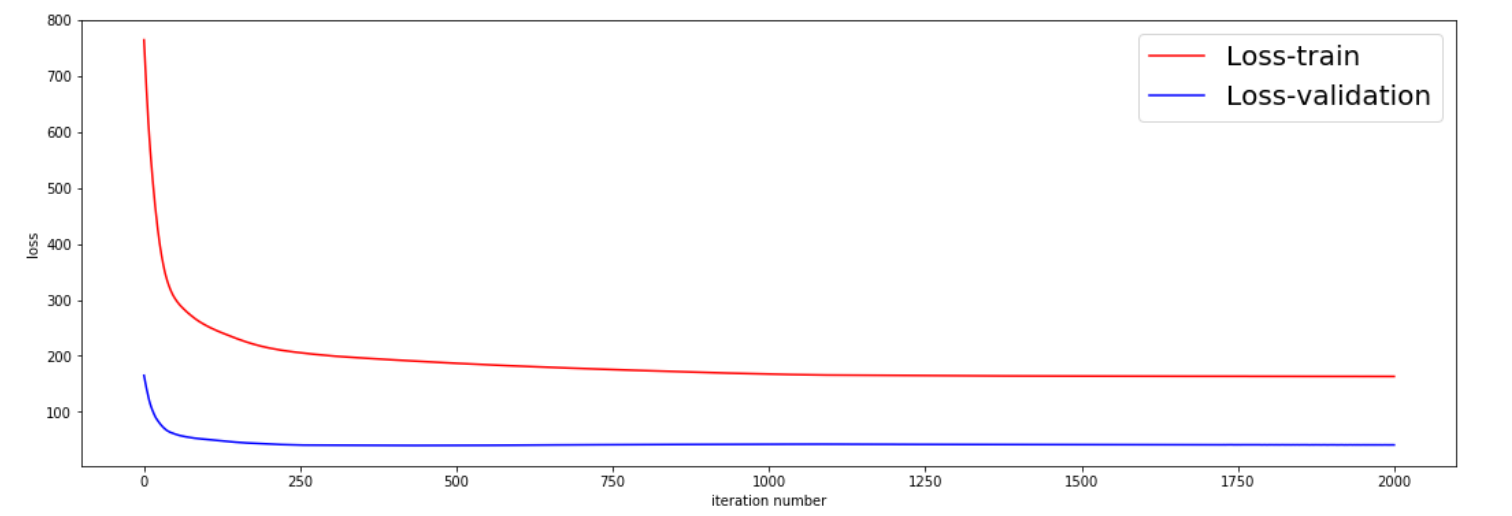
0.8623188405797102

## loss曲线图：

线性回归：



线性分类：



## 12.实验结果分析:

实验结果基本达到了预期的要求，随着迭代次数的增加，模型的学习能力更

强，验证集结果基本与训练集结果相差不大，但是随着迭代次数的进一步增

大并没有出现过拟合的情况，不管是线性分类还是线性回归验证集的运行结

果都是收敛，可能是由于数据集太小的缘故。也有可能是迭代次数太小的原

因，因为自己的电脑运行更多的次数花费时间太长。

## 13.对比线性回归和线性分类的异同点：

问题的本质都是模型的训练，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：

这次实验我实现了利用梯度下降的方法去实现线性回归与线性分类，加强了我对于梯度下降的掌握，同时也对线性回归、线性分类的公式推导过程有了更深的理解，也有自己推导一边书本上的公式。同时，本次实验中我还有很多不足的地方，例如对于python numpy库的矩阵操作有很多不熟练的地方，写代码中犯了许多错误、还有对loss函数的理解不够深刻，导致写出错误的loss函数，甚至写不出loss函数。总之通过这次实验，我受益良多。