

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  **梁子豪**

**学 号 201530612163**

**邮 箱 1020224260@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](https://www.zybuluo.com/chenyaofo/note/949882" \t "https://www.zybuluo.com/chenyaofo/note/_blank)**

## 2. 实验时间：**2017年12月2日**

## 3. 报告人：梁子豪

## 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。

## 数据集以及数据分析：

数据集分别是：

线性回归使用的是[LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/" \t "https://www.zybuluo.com/chenyaofo/note/_blank)中的[Housing](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html" \l "housing" \t "https://www.zybuluo.com/chenyaofo/note/_blank)数据，包含506个样本，每个样本有13个属性。请自行下载scaled版本，并将其切分为训练集，验证集。   
线性分类使用的是[LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/" \t "https://www.zybuluo.com/chenyaofo/note/_blank)中的[australian](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html" \l "australian" \t "https://www.zybuluo.com/chenyaofo/note/_blank)数据，包含690个样本，每个样本有14 个属性。请自行下载scaled版本，并将其切分为训练集，验证集。

## 6. 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](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html" \t "https://www.zybuluo.com/chenyaofo/note/_blank)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](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html" \t "https://www.zybuluo.com/chenyaofo/note/_blank)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。

6取梯度的负方向，记为。

7更新模型参数，。为学习率，是人为调整的超参数。

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，画出和随迭代次数的变化图。

线性分类和梯度下降

1读取实验数据，使用sklearn库的[load\_svmlight\_file](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html" \t "https://www.zybuluo.com/chenyaofo/note/_blank)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](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html" \t "https://www.zybuluo.com/chenyaofo/note/_blank)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。

6取梯度的负方向，记为。

7更新模型参数，。为学习率，是人为调整的超参数。

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，画出和随迭代次数的变化图。

## 代码内容:

线性回归：

**from** sklearn.externals.joblib **import** Memory  
**from** sklearn.datasets **import** load\_svmlight\_file  
  
mem = Memory(**"./mycache"**)  
@mem.cache  
**def** get\_data():  
 data = load\_svmlight\_file(**"housing.txt"**)  
 **return** data[0], data[1]  
  
X, y = get\_data()  
train\_X = X[0:506]  
train\_y = y[0:506]  
  
**from** numpy **import** \*  
**import** numpy **as** np  
**from** sklearn.model\_selection **import** train\_test\_split  
  
X\_train, X\_validation, y\_train, y\_validation = train\_test\_split(X, y, test\_size=0.33, random\_state=42)  
Xt=X\_train.toarray()  
Xt=mat(Xt)  
Xv=X\_validation.toarray()  
Xv=mat(Xv)  
Yt=mat(y\_train).T  
Yv=mat(y\_validation).T  
  
*#print(Xt.shape)  
#print(y\_train.shape)  
#print(y\_train)*w=[np.float(10)**for** i **in** range(14)]  
w=mat(w).T  
  
one=[np.float64(1) **for** x **in** range(339)]  
one2=[np.float64(1) **for** x **in** range(167)]  
  
  
nita=0.0001  
Xtt = np.column\_stack((Xt,one))  
Xvv = np.column\_stack((Xv,one2))  
  
  
print(Yt.shape)  
time=100  
loss=[np.float(10)**for** i **in** range(0,time)]  
lossV=[np.float(10)**for** i **in** range(0,time)]  
**for** i **in** range(0,time):  
 w1=2\*(Xtt.T)\*(Xtt\*w-Yt)  
 w=w-nita\*w1  
 L=np.float(((Xtt.dot(w)-Yt).T)\*(Xtt.dot(w)-Yt))  
 LV=np.float(((Xvv.dot(w)-Yv).T)\*(Xvv.dot(w)-Yv))  
 loss[i]=L  
 lossV[i]=LV  
 *#lossV[i]=LV*print(loss[time-1])  
x\_data=np.arange(0,time,1)  
y\_data=loss  
  
  
y\_data2=lossV  
**import** matplotlib.pyplot **as** plt  
plt.plot(x\_data,y\_data)  
plt.plot(x\_data,y\_data2)  
plt.title(**u"Loss"**)  
plt.legend()  
plt.xlabel(**u"time"**)  
plt.ylabel(**u"loss"**)  
plt.show()

线性分类：

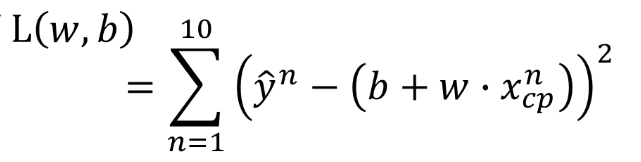
**from** sklearn.externals.joblib **import** Memory  
**from** sklearn.datasets **import** load\_svmlight\_file  
  
  
**def** get\_data():  
 data = load\_svmlight\_file(**"australian.txt"**)  
 **return** data[0], data[1]  
  
X, y = get\_data()  
*#print(X.shape)*row=690  
col=14  
train\_X = X[0:690]  
train\_y = y[0:690]  
  
**from** numpy **import** \*  
**import** numpy **as** np  
**from** sklearn.model\_selection **import** train\_test\_split  
  
X\_train, X\_validation, y\_train, y\_validation = train\_test\_split(X, y, test\_size=0.33, random\_state=42)  
Xt=X\_train.toarray()  
Xv=X\_validation.toarray()  
Yt=mat(y\_train)  
Yt=np.array(Yt)  
Yv=mat(y\_validation)  
Yv=np.array(Yv)  
row,col=Xt.shape  
rowV,colV=Xv.shape  
  
print(**"Xt.shape"**,Xt.shape)  
print(**"Yt.shape"**,Yt.shape)  
  
  
  
w=[np.float(1)**for** i **in** range(0,col)]  
w=np.array(w)  
print(**"w"**,w)  
  
  
Wgrad\_average=([1.00 **for** i **in** range(0,col)])  
Wgrad\_average=np.array(Wgrad\_average)  
print(**"Wgrad\_average"**,Wgrad\_average)  
  
**def** lossGrad(X,y,w):  
 row,col=X.shape  
 err=np.array([np.float(0) **for** i **in** range(0,row)]).reshape(row,1)  
 x=mat(X)  
 y=np.array(y).reshape(row,1)  
 **for** i **in** range(0,row):  
 Xi=x[i,:].reshape(1,col)  
 Yi=np.float(y[i,:])  
 err[i]=Yi \* Xi.dot(w)  
 **return** err  
time=100  
loss=[np.float(0)**for** i **in** range(0,time)]  
errorRate =[np.float(0)**for** i **in** range(0,time)]  
errorRateV=[np.float(0)**for** i **in** range(0,time)]  
print(**"loss"**,loss)  
nita=0.01  
**for** j **in** range(0,time):  
 k=0  
 kV=0  
 err=lossGrad(Xt,Yt,w)  
 errV = lossGrad(Xv, Yv, w)  
 C=0.9  
 Wgrad = np.zeros((row, col)).reshape(row,col)  
 **for** i **in** range(0, row):  
 k+=1  
 Wgrad[i,:]=w  
 **for** i **in** range(0,row):  
 erri=np.float(err[i])  
 **if** 1-erri>=0:  
 k-=1  
 Wgrad[i,:]=Wgrad[i,:]-C\*Yt.dot(Xt)  
 **for** i **in** range(0, rowV):  
 kV += 1  
 **for** i **in** range(0,rowV):  
 erriV = np.float(errV[i])  
 **if** 1 - erriV >= 0:  
 kV-=1  
 errorRate[j] = 1.0 - np.float(k / row)  
 errorRateV[j] = 1.0 - np.float(kV / rowV)  
 Wgrad\_average=np.mean(Wgrad,axis=0)  
 w=w-nita\*Wgrad\_average  
  
  
print(**"w"**,w)  
err=lossGrad(Xt,Yt,Wgrad\_average)  
print(**"err"**,err.shape)  
  
  
x\_data=np.arange(0,time,1)  
y\_data=errorRate  
y\_data2=errorRateV  
**import** matplotlib.pyplot **as** plt  
plt.plot(x\_data,y\_data)  
plt.plot(x\_data,y\_data2)  
plt.title(**u"Error rate "**)  
plt.legend()  
plt.xlabel(**u"time"**)  
plt.ylabel(**u"Error rate "**)  
plt.show()

## 8. 选择的评估方法（留出法，交叉验证，k折交叉验证等）:留出法

## 9. 模型参数的初始化方法:全常量初始化

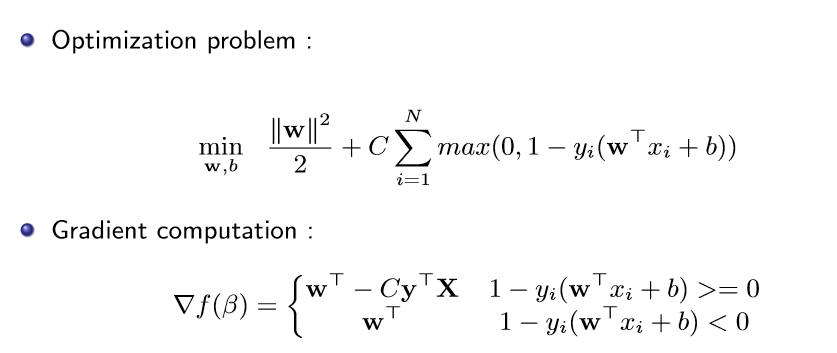
## 10.选择的loss函数及其导数:

## 线性回归：



导数=2\*(X.T)\*(X\*w-Yt)

线性分类：



## 11.实验结果和曲线图:

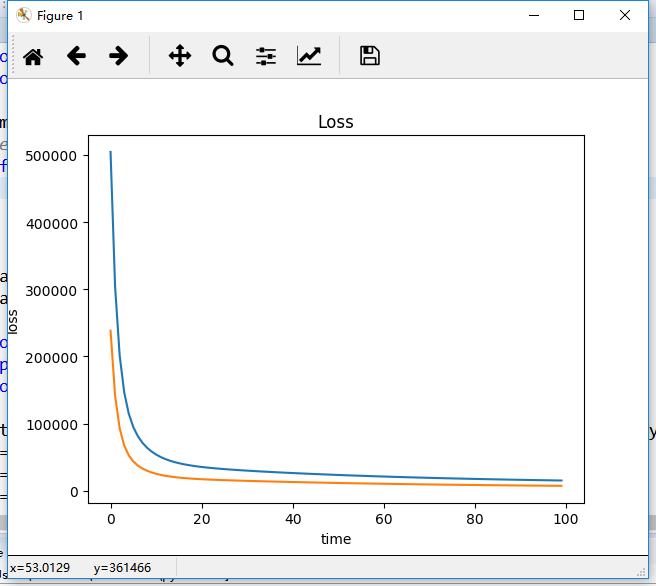
## **超参数选择（η,epoch等）：**η=0.0001

## **评估结果（根据选择的评估方法）：**

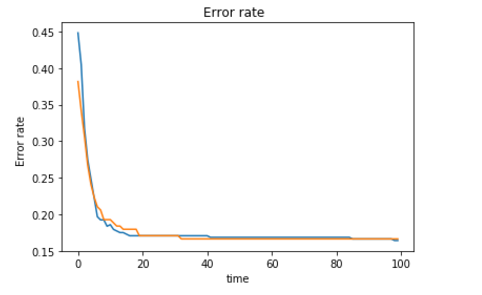
## **预测结果（最佳结果）：**

## **loss曲线图：**

①线性回归：



线性分类：



## 实验结果分析:

通过梯度下降算法，参数wT不断发生调整，使得loss下降，函数对数据的拟合程度提升。

## 13.对比线性回归和线性分类的异同点：

同：都是有w矩阵对属性加权运算模型。

两个问题本质上都是一致的，就是模型的拟合

异：求minLoss的方式不一样，线性分类要比线性回归难实现orz。

分类问题的y值, 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。

## 14.实验总结：

经过这个实验，我发现更多机器学习的魅力

对选择这个方向感到欣慰。