

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  周东东

**学 号 201530613849**

**邮 箱 2210663134@qq.com**

**指导教师** **吴庆耀**

**提交日期** **2017年 12 月2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:周东东

## 4. 实验目的:

4.1进一步理解线性回归、分类和梯度下降的原理。

4.2在小规模数据集上实践。

4.3体会优化和调参的过程。

## 5. 数据集以及数据分析：

**线性回归和梯度下降**：

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。使用的是scaled版本，并将其切分为训练集，验证集。

**线性分类和梯度下降**：

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。使用的是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t。

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，。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。

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，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。

## 7. 代码内容:

**7.1线性回归和梯度下降：**

**from sklearn.datasets import load\_svmlight\_file**

**import numpy as np**

**from sklearn.model\_selection import train\_test\_split**

**from scipy.sparse import csr\_matrix, hstack**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**def data\_loader(train\_file):**

**X,Y = load\_svmlight\_file(train\_file)**

**#housing\_scale中有506个数据**

**one = np.ones(506).reshape(506,1)#one是506行1列的矩阵**

**X = hstack( [X, csr\_matrix(one)] ).todense()**

**Y = np.array(Y).reshape(506,1)**

**return X, Y**

**def train(X, Y):**

**Y\_prediction = X.dot(theta)**

**loss=1/2/(506\*0.7) \* (np.dot( (Y - Y\_prediction).transpose(), Y - Y\_prediction )).sum()**

**return loss**

**def test(X\_test,Y\_test):**

**Y\_prediction = X\_test.dot(theta)**

**loss=1/2/(506\*0.3) \* (np.dot( (Y\_test - Y\_prediction).transpose(), Y\_test - Y\_prediction )).sum()**

**return loss**

**if \_\_name\_\_ =="\_\_main\_\_":**

**X,Y = data\_loader("housing\_scale")**

**X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,test\_size=0.3,random\_state=0)#划分数据集**

**theta = np.zeros((X\_train.shape[1], 1))**

**L\_train = np.zeros((150))**

**L\_validation = np.zeros((150))**

**learning\_rate = 0.000350**

**for t in range(150):**

**Y\_prediction = X\_train.dot(theta)**

**G = -np.dot(X\_train.transpose(), Y\_train ) + np.dot(X\_train.transpose(),Y\_prediction)**

**theta -= learning\_rate \* G**

**L\_train [t]=train(X\_train, Y\_train)**

**L\_validation[t]=test(X\_test,Y\_test)**

**print('L\_train:\n',L\_train)**

**print('L\_validation:\n',L\_validation)**

**plt.plot(loss1,label='train')**

**plt.plot(loss2,label='test')**

**plt.ylabel('loss value')**

**plt.xlabel('literation number')**

**plt.legend()**

**7.2线性分类和梯度下降：**

**from sklearn.datasets import load\_svmlight\_file**

**import numpy as np**

**from sklearn.model\_selection import train\_test\_split**

**from scipy.sparse import csr\_matrix, hstack**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**def data\_loader(train\_file):**

**X,Y = load\_svmlight\_file(train\_file)**

**#housing\_scale中有690个数据**

**#one = np.ones(690).reshape(690,1)#one是690行1列的矩阵**

**#X = hstack( [X, csr\_matrix(one)] ).todense()**

**#Y = np.array(Y).reshape(690,1)**

**X = X.toarray()**

**X = np.c\_[np.ones((X.shape[0], 1)),X]**

**return X, Y**

**def train\_test(X, Y, theta):**

**#Y\_prediction = X.dot(theta)**

**a = np.sum(theta \* X, axis = 1)**

**b = Y \* a**

**c = 0**

**for i in range(X.shape[0]):**

**c = c + max(0, 1 - b[i])**

**loss = np.linalg.norm(theta)\*\*2 / 2 + c**

**return loss**

**if \_\_name\_\_ =="\_\_main\_\_":**

**X,Y = data\_loader("australian\_scale")**

**X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,test\_size=0.3,random\_state=0)#划分数据集**

**theta = np.zeros((1,X\_train.shape[1]))**

**L\_train = np.zeros((150))**

**L\_validation = np.zeros((150))**

**prediction\_rate = np.zeros((150))**

**learning\_rate = 0.000150**

**threshold = -1**

**for t in range(150):**

**#Y\_prediction = X.dot(theta)**

**G = theta**

**e = np.sum(theta \* X\_train, axis = 1)**

**f = 1 - Y\_train \* e**

**for j in range(0, 482):**

**if f[j] >= 0:**

**G = G - X\_train[j] \* Y\_train[j]**

**else:**

**G = G**

**theta -= learning\_rate \* G**

**pre = np.sum(theta \* X, axis = 1)**

**count = 0**

**for j in range(690):**

**if pre[j] > threshold:**

**pre[j] = 1**

**elif pre[j] < threshold:**

**pre[j] = -1**

**if pre[j] == Y[j]:**

**count += 1**

**prediction\_rate[t] = count / 690**

**L\_train[t]=train\_test(X\_train, Y\_train, theta)/(690\*0.7)**

**L\_validation[t]=train\_test(X\_train, Y\_train, theta)/(690\*0.3)**

**print('L\_train:\n',L\_train)**

**print('L\_validation:\n',L\_validation)**

**plt.plot(prediction\_rate,label='rate')**

**plt.legend()**

**plt.show()**

**plt.plot(L\_train,label='train')**

**plt.plot(L\_validation,label='test')**

**plt.ylabel('loss value')**

**plt.xlabel('literation number')**

**plt.legend()**

## 8. 选择的评估方法:留出法

## 9. 模型参数的初始化方法:

9.1线性回归和梯度下降

theta = np.zeros((X\_train.shape[1], 1))

learning\_rate = 0.000350

9.2 线性分类和梯度下降

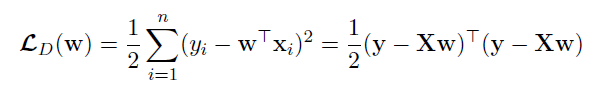
theta = np.zeros((1,X\_train.shape[1]))

learning\_rate = 0.000150

threshold = -1

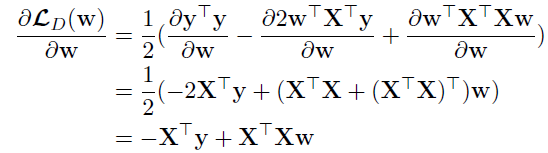
## 10.选择的loss函数及其导数:

**10.1线性回归和梯度下降**



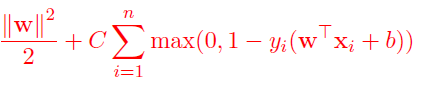
loss=1/2/(506\*0.7) \* (np.dot( (Y - Y\_prediction).transpose(), Y - Y\_prediction )).sum()

loss=1/2/(506\*0.3) \* (np.dot( (Y\_test - Y\_prediction).transpose(), Y\_test - Y\_prediction )).sum()



G=-np.dot(X\_train.transpose(),Y\_train)+np.dot(X\_train.transpose(),Y\_predictio)

**10.2 线性分类和梯度下降**

L(w)=

def train\_test(X, Y, theta):

#Y\_prediction = X.dot(theta)

## a = np.sum(theta \* X, axis = 1)

## b = Y \* a

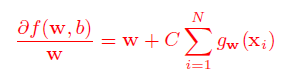
## c = 0

## for i in range(X.shape[0]):

## c = c + max(0, 1 - b[i])

## loss = np.linalg.norm(theta)\*\*2 / 2 + c

## return loss



本次实验中我没有用参数b

G = theta

e = np.sum(theta \* X\_train, axis = 1)

f = 1 - Y\_train \* e

for j in range(0, 482):

if f[j] >= 0:

G = G - X\_train[j] \* Y\_train[j]

else:

G = G

theta -= learning\_rate \* G

## 11.实验结果和曲线图:

**11.1 线性回归与梯度下降**

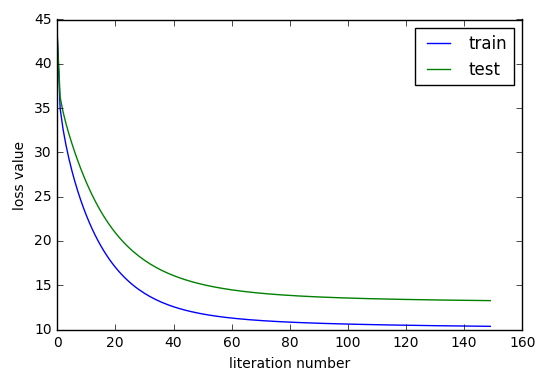
## 超参数选择（η,epoch等）：learning\_rate = 0.000350

## 评估结果（根据选择的评估方法）：loss值减小最终趋于稳

定14.22233566

## 预测结果（最佳结果）：loss=14.28223566

## loss曲线图：



11.2 线性分类与梯度下降

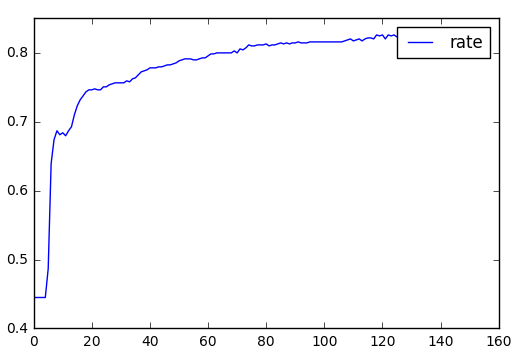
## 超参数选择（η,epoch等）：learning\_rate = 0.000150

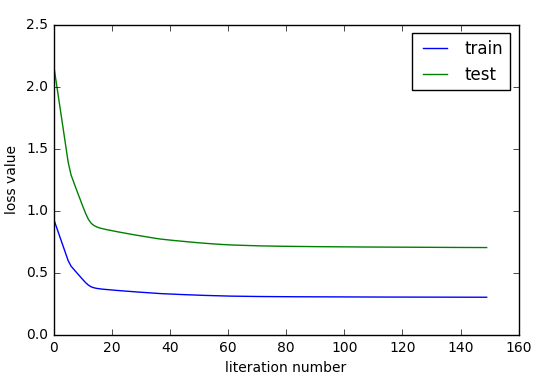
## threshold = -1

## 评估结果（根据选择的评估方法）：rate=0.836086956522

## 预测结果（最佳结果）：loss=0.70117138

## loss曲线图：





## 12.实验结果分析:

回归与分类都完整的实现了，由上面的loss曲线图可以看出loss函数如

果取了平均的话，train集表现的比test集要好，这跟上课讲的一样。

分类实验，最终的准确率有0.83多，这个可能是test数据集比较小的缘

故。

## 13.对比线性回归和线性分类的异同点：

相同：总的来说两个问题本质上都是一致的，就是模型的拟合。

对于这两类问题都有以下几个步骤：

1.如何选取一个合理的模型(线性的，or 非线性的).

2.制造一个"美好"的误差函数 (可以评估拟合程度)

3.采取一切可能的技术(e.g. 导数下降法，解极值方法)

求出最好的模型参数

区别：回归问题预测的结果是连续的值，而分类问题的预测结

果是离散的。

## 14.实验总结：

本次实验的三个目的基本完成。理解了线性分类、回归和梯度下降的完

整过程，并实现了算法，调整了参数，基本找到了最优解。

这是第一次完整地写机器学习的代码，以前上课的时候都是学习算法，

感觉听懂了，就完事了，也拿过别人写的代码跑过，觉得自己也能写。

但是真正开始写的时候就知道自己算法还是没理解透彻。主要是那几个

导数真的把我搞糊涂了。以前上课的时候还能糊弄过去，现在不行了，

要真的把它的原理搞清楚。代码其它地方还是比较简单的。另外，我觉

的有必要重新学习数学了，要把数学学好一点，这样才能在程序员的路

上走的更远。