

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**   **黄班班**

**学 号 201530611647**

**邮 箱 863237407@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 8 日

## 3. 报告人: 黄班班

## 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。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中的australian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。

## 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。

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ltrain，在验证集上测试并得到Loss函 数值lvalidation。

9.重复步骤5-8若干次，画出ltrain和lvalidation随迭代次数的变化图。

## 7. 代码内容:

线性回归：

import sklearn

import numpy as np

import matplotlib.pyplot as plt

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("F:\\machinelearning\\11.txt")

return data[0], data[1]

X, y = get\_data()

X=X.todense()

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

from numpy import \*

w=mat(zeros((13,1)))

b=np.random.normal()

b1=mat(b)

w1=vstack((w,b1))

def loss(x,y):

a=len(x)

kk=mat(ones((a,1)))

xx=hstack((x, kk))

y\_t=mat(y).T

tt=y\_t -xx\*w1

return (tt.T)\*tt

def gradient(x,y):

a = len(x)

kk = mat(ones((a, 1)))

xx = hstack((x, kk))

xx\_t=xx.T

y\_t = mat(y).T

t1= xx\*w1

return 2\*xx\_t\*(( t1- y\_t ))

rate=0.0001

listx=[]

listy=[]

listz=[]

for i in range(1000):

w1=w1- rate \* gradient(X\_train,y\_train)

listx.append(i)

aaa=loss(X\_train,y\_train)

aaa2=array(aaa)

listy.append(aaa2[0][0])

bbb = loss(X\_test, y\_test)

bbb2 = array(bbb)

listz.append(bbb2[0][0])

plt.figure('model')

plt.plot(listx ,listy, label="$losstrain$",color="red", linewidth=2)

#plt.figure('losstest')

plt.plot(listx ,listz, label="$losstest$",color="blue", linewidth=2)

plt.title(["rate=",rate])

plt.legend()

plt.show()

线性分类：

import sklearn

import numpy as np

import matplotlib.pyplot as plt

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("F:\\machinelearning\\22.txt")

return data[0], data[1]

X, y = get\_data()

X=X.todense()

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

from numpy import \*

#w=mat(zeros((14,1)))

w=mat(random.rand(14,1))

b=np.random.normal()

c=0.1

def loss(x,y):

b1 = mat(b)

w1 = vstack((w, b1))

ww=(w.T)\*w/2

a = len(x)

kk = mat(ones((a, 1)))

xx = hstack((x, kk))

t1=xx\*w1

t2=t1

for i in range(a):

t1[i,0]=1-y[i]\*t1[i,0]

for i in range(a):

t2[i,0]=max(0,t1[i,0])

ww2=0

for i in range(a):

ww2+=t2[i,0]

return ww+c\*ww2

def gw(x,y):

b1 = mat(b)

w1 = vstack((w, b1))

a = len(x)

kk = mat(ones((a, 1)))

xx = hstack((x, kk))

t1 = xx \* w1

x1=x

x2=x

for i in range(a):

t1[i, 0] = 1 - y[i] \* t1[i, 0]

if(t1[i,0]>=0):

x1[i]=-y[i]\*x[i]

else:

x1[i]=0\*x[i]

x2[0]=0\*x2[0]

for i in range(a):

x2[0]+=x1[i]

return x2[0]

def gb(x,y):

b1 = mat(b)

w1 = vstack((w, b1))

a = len(x)

kk = mat(ones((a, 1)))

xx = hstack((x, kk))

t1 = xx \* w1

y1=y

for i in range(a):

t1[i, 0] = 1 - y[i] \* t1[i, 0]

if (t1[i, 0] >= 0):

y1[i]=-y[i]

else:

y1[i]=0

yy=0

for i in range(a):

yy+=y1[i]

return yy

def gradient\_w(x,y):

return w.T+c\*gw(x,y)

def gradient\_b(x,y):

return c\*gb(x,y)

rate=0.001

listx=[]

listy=[]

listz=[]

for i in range(300):

w\_t=w.T

w\_t=w\_t-rate\*gradient\_w(X\_train,y\_train)

b=b-rate\*gradient\_b(X\_train,y\_train)

w=w\_t.T

listx.append(i)

aaa = loss(X\_train, y\_train)

aaa2 = array(aaa)

listy.append(aaa2[0][0])

bbb = loss(X\_test, y\_test)

bbb2 = array(bbb)

listz.append(bbb2[0][0])

plt.figure('model')

plt.plot(listx, listy, label="$losstrain$", color="red", linewidth=2)

# plt.figure('losstest')

plt.plot(listx, listz, label="$losstest$", color="blue", linewidth=2)

plt.title(["rate=", rate])

plt.legend()

plt.show()

## 选择的评估方法（留出法，交叉验证，k折交叉验证等）:

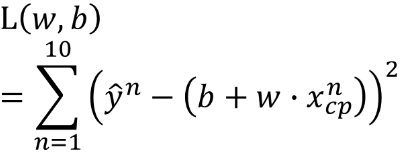
## 留出法

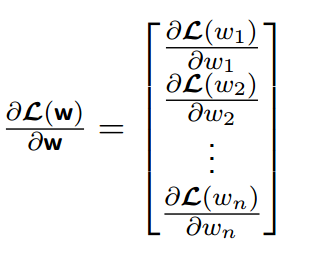
## 模型参数的初始化方法:

线性回归为线性模型全零初始化

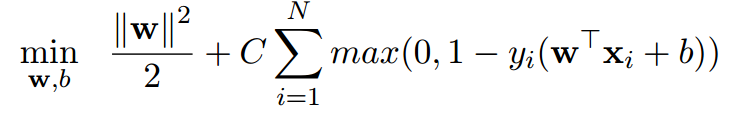
线性分类为支持向量机模型全零初始化

## 选择的loss函数及其导数:

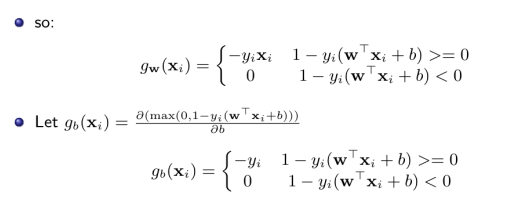
线性回归loss函数：

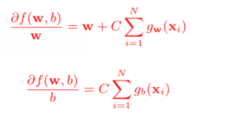
导数：

线性分类loss函数：

loss=

线性分类导数：

令

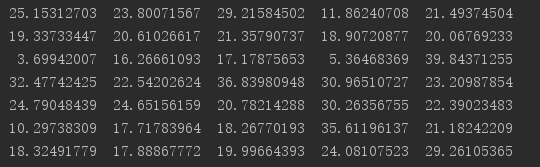
导数为：

## 11.实验结果和曲线图:

## 超参数选择（η,epoch等）：c=0.1

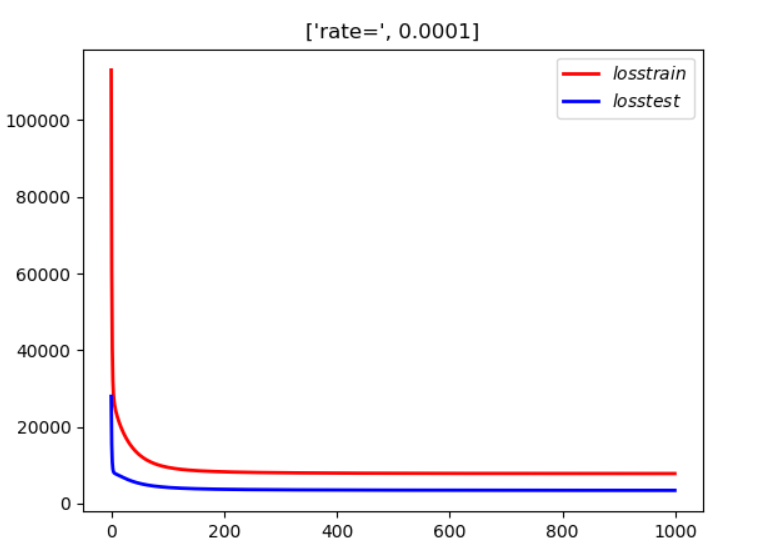
## 评估结果（根据选择的评估方法）：留出法

## 预测结果（最佳结果）：测试集预测：

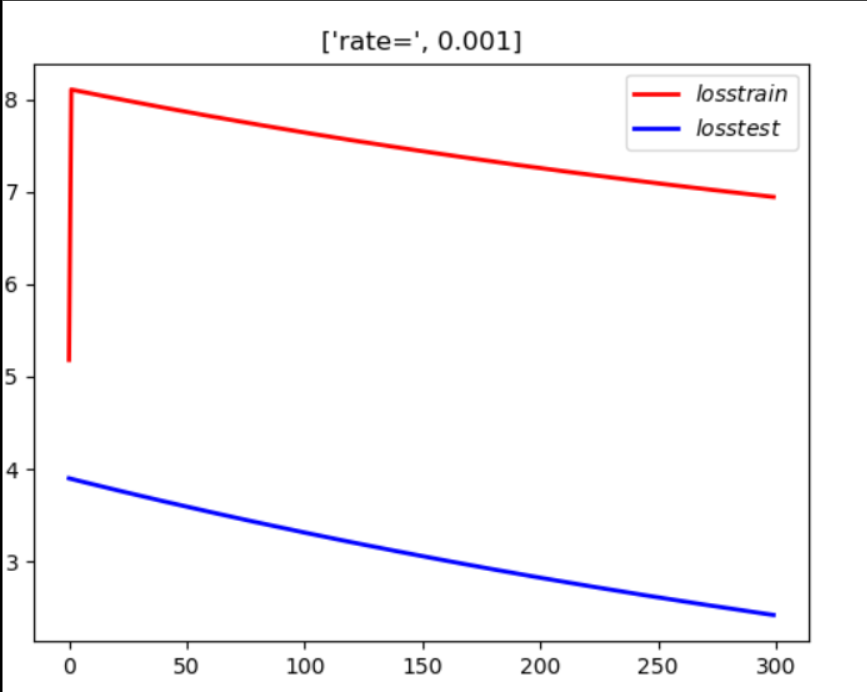


## loss曲线图：

线性回归：



线性分类：



## 实验结果分析:

对于线性回归，多次迭代后loss函数趋于稳定

对于线性分类，测试集的loss值一开始有反常，但两者多次迭代后loss函数趋于稳定

## 对比线性回归和线性分类的异同点：

同：都是线性模型

异：初始化方法不同，loss函数不同

## 实验总结：

这次试验让我学到了很多，对于线性模型的预测有了更好的了解，在如何调参和梯度下降方面收获颇大，也使我温故了一遍所学知识。