

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  廖瑞杰

**学 号 201530612170**

**邮 箱 287462612@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:廖瑞杰

## 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。

## 6. 实验步骤:

线性回归和梯度下降

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。

## 7. 代码内容:

（针对线性回归和线性分类分别填写8-12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("C:\\Users\\Administrator\\Desktop\\housing.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=0.1, 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):  
 hnum = len(x)  
 k1 = mat(ones((hnum, 1)))  
 xx = hstack((x, k1))  
 y\_t = mat(y).T  
 tt = y\_t - xx\*w1  
 return tt.T\*tt  
  
def gradient(x, y):  
 hnum = len(x)  
 k1 = mat(ones((hnum, 1)))  
 xx = hstack((x, k1))  
 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(2000):  
 w1 = w1 - rate \* gradient(X\_train, y\_train)  
 listx.append(i)  
 loss\_train=loss(X\_train, y\_train)  
  
 listy.append(loss\_train[0, 0])  
 loss\_test = loss(X\_test, y\_test)  
  
 listz.append(loss\_test[0, 0])  
  
  
plt.figure('model')  
plt.plot(listx, listy, label="$losstrain$", color="green", linewidth=3)  
plt.plot(listx, listz, label="$losstest$", color="red", linewidth=3)  
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("C:\\Users\\Administrator\\Desktop\\australian.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=0.1, random\_state=0)  
  
from numpy import \*  
w = mat(random.rand(14, 1))  
b = np.random.normal()  
  
c = 0.02  
def loss(x,y):  
 b1 = mat(b)  
 w1 = vstack((w, b1))  
 ww = w.T\*w/2  
 hnum = len(x)  
 k1 = mat(ones((hnum, 1)))  
 xx = hstack((x, k1))  
 t1 = xx\*w1  
 t2 = t1  
 for i in range(hnum):  
 t1[i, 0] = 1-y[i]\*t1[i, 0]  
 for i in range(hnum):  
 t2[i, 0] = max(0, t1[i, 0])  
 ww2 = 0  
 for i in range(hnum):  
 ww2 += t2[i, 0]  
 return ww+c\*ww2  
  
def gw(x,y):  
 b1 = mat(b)  
 w1 = vstack((w, b1))  
 hnum = len(x)  
 k1 = mat(ones((hnum, 1)))  
 xx = hstack((x, k1))  
 t1 = xx \* w1  
 x1 = x  
 x2 = x  
 for i in range(hnum):  
 t1[i, 0] = 1 - y[i] \* t1[i, 0]  
 if t1[i, 0] >= 0:  
 x1[i] = -y[i]\*x[i]  
 else:  
 x1[i] = 0  
 x2[0] = 0  
 for i in range(hnum):  
 x2[0] += x1[i]  
 return x2[0]  
  
def gb(x, y):  
 b1 = mat(b)  
 w1 = vstack((w, b1))  
 hnum = len(x)  
 k1 = mat(ones((hnum, 1)))  
 xx = hstack((x, k1))  
 t1 = xx \* w1  
 y1 = y  
 for i in range(hnum):  
 t1[i, 0] = 1 - y[i] \* t1[i, 0]  
 if t1[i, 0] >= 0:  
 y1[i] = -y[i]  
 else:  
 y1[i] = 0  
 y2 = 0  
 for i in range(hnum):  
 y2 += y1[i]  
 return y2  
  
  
def gradient\_w(x, y):  
  
 return w.T+c\*gw(x, y)  
  
  
def gradient\_b(x, y):  
  
 return c\*gb(x, y)  
  
  
rate = 0.0001  
listx = []  
listy = []  
listz = []  
  
for i in range(500):  
 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)  
 loss\_train = loss(X\_train, y\_train)  
  
 listy.append(loss\_train[0, 0])  
 loss\_test = loss(X\_test, y\_test)  
  
 listz.append(loss\_test[0, 0])  
  
plt.figure('model')  
plt.plot(listx, listy, label="$losstrain$", color="green", linewidth=3)  
plt.plot(listx, listz, label="$losstest$", color="red", linewidth=3)  
plt.title(["rate=", rate])  
plt.legend()  
plt.show()

## 8. 选择的评估方法（留出法，交叉验证，k折交叉验证等）:

线性回归：留出法

线性分类：交叉验证

## 9. 模型参数的初始化方法:

线性回归：参数全零初始化

线性分类：参数随机初始化

## 10.选择的lo

## ss函数及其导数:

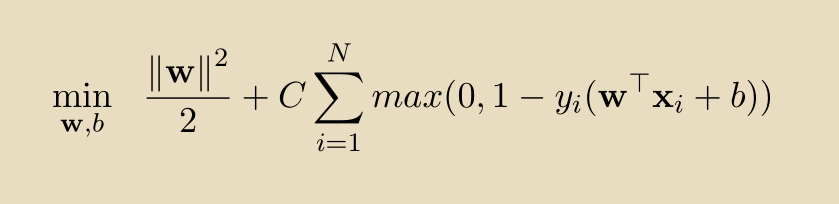
线性回归:

loss函数为L(X)=(y-X\*w).T \*(y-X\*w)

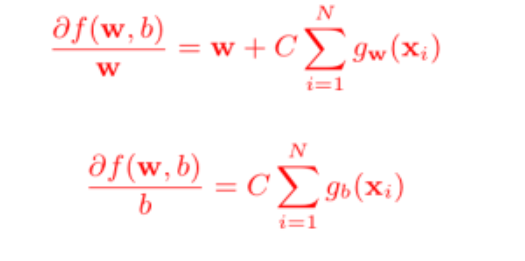
导数为g(X)=2\*X.T(X\*w-y)

线性分类:

loss函数为



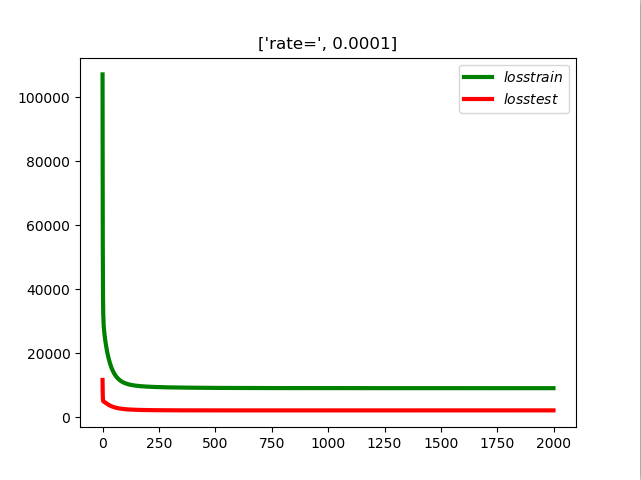
导数为



## 11.实验结果和曲线图:

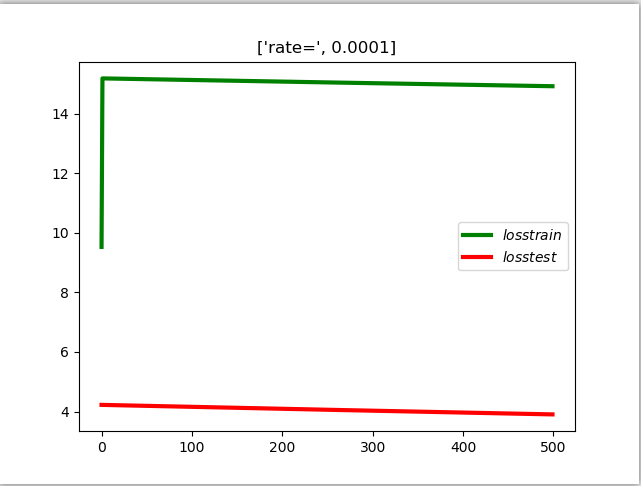
线性回归: η=0.0001

## loss曲线图：



线性分类: η=0.0001

loss曲线:



## 12.实验结果分析:

随着迭代次数增加，loss函数值降低

## 13.对比线性回归和线性分类的异同点：

同:随着迭代次数增加，loss函数值降低

异:loss函数曲线有较大差异。

## 14.实验总结：经过这次实验，我更好地掌握了线性回归和线性分类的知识，能够对给定数据集进行测试，计算损失值，并通过模型进行预测。