

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**   **赵云潇**

**学 号 201530613757**

**邮 箱 zhao.yunxiao@outlook.com**

**指导教师**  **吴庆耀**

**提交日期** **2017年 12 月 7 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:赵云潇

## 4. 实验目的:

4.1. 进一步理解线性回归和梯度下降的原理。

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。

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。

## 6. 实验步骤:

6.1. 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。

6.2. 线性分类和梯度下降

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。

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。

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。

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。

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:

线性回归：

from sklearn.datasets import load\_svmlight\_file

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

import matplotlib.pyplot as plt

import numpy.random

def get\_data():

data = load\_svmlight\_file("housing\_scale")

return data[0], data[1]

def loss(W, X, y, b):

\_loss = 0

for i in range(len(y)):

\_fx = 0

for j in range(len(W)):

\_fx += W[j]\*X[i,j] + b

ll = (y[i] - \_fx)

ll \*= ll

\_loss += ll

\_loss \*= 0.5

return \_loss

def gradient(W, X, y):

gri = []

for i in range(len(W)):

mem = 0

square = 0

for j in range(len(y)):

mem+=y[j]\*X[j,i]

for k in range(len(W)):

square+=X[j,k]\*X[j,k]

gri.append(square\*W[i]-mem)

return gri

def decent(W, G, l):

for i in range(len(W)):

W[i] -=l\*G[i]

def optimize(W, X\_train, y\_train, X\_validation, y\_validation, times, b):

step = [0.01, 0.0001, 0.00001, 0.00001]

loss\_train = []

loss\_validation = []

for i in range(times):

dW = gradient(W, X\_train, y\_train)

\_loss = loss(W, X\_train, y\_train, b)

if (\_loss > 200000):

decent(W, dW, step[0])

elif (\_loss > 30000):

decent(W, dW, step[1])

elif (\_loss > 10000):

decent(W, dW, step[2])

else:

decent(W, dW, step[3])

loss\_train.append(\_loss)

loss\_validation.append(loss(W, X\_validation, y\_validation, b))

return loss\_train, loss\_validation

def main():

X, y = get\_data()

X\_train, X\_validation, y\_train, y\_validation = train\_test\_split(

X, y, test\_size=0.3)

W = [numpy.random.normal() for i in range(13)]

b = numpy.random.normal()

print(loss(W, X\_validation, y\_validation, b))

loss\_train, loss\_validation = optimize(W, X\_train, y\_train, X\_validation, y\_validation, 8, b)

print(loss(W, X\_validation, y\_validation, b))

x1=range(1,9)

x2=range(1,9)

plt.figure(figsize=(10, 6))

plt.plot(x1,loss\_train,'',label="train set")

plt.plot(x2,loss\_validation,'',label="validation set")

plt.legend(loc='upper right')

plt.xlabel('times')

plt.ylabel('loss')

plt.show()

main()

线性分类：

from sklearn.datasets import load\_svmlight\_file

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

import matplotlib.pyplot as plt

import numpy.random

def get\_data():

data = load\_svmlight\_file("australian\_scale")

return data[0], data[1]

def loss(W, X, y, b, C):

\_loss = 0

for i in range(len(W)):

\_loss+=W[i]\*W[i]/2

for i in range(len(y)):

minus = 0

for j in range(len(W)):

minus+=y[i]\*W[j]\*X[i,j]

minus = 1 - minus - y[i]\*b

\_loss+=max(minus, 0)\*C

return \_loss

def gradient(W, X, y, b, C):

gri = [W[i] for i in range(14)]

for i in range(len(y)):

fx = 0

for j in range(len(W)):

fx+=y[i]\*W[j]\*X[i,j]

fx+=y[i]\*b

if (fx <= 1):

for j in range(len(W)):

gri[j]-=C\*y[i]\*X[i,j]

return gri

def decent(W, G, l):

for i in range(len(W)):

W[i]-=l\*G[i]

def optimize(W, X\_train, y\_train, X\_validation, y\_validation, times, b, C):

loss\_train = []

loss\_validation = []

for i in range(times):

dW = gradient(W, X\_train, y\_train, b, C)

decent(W, dW, 0.0001)

loss\_train.append(loss(W, X\_train, y\_train, b, C))

loss\_validation.append(loss(W, X\_validation, y\_validation, b, C))

return loss\_train, loss\_validation

def main():

X, y = get\_data()

X = X.todense()

X\_train, X\_validation, y\_train, y\_validation = train\_test\_split(

X, y, test\_size=0.3)

W = [numpy.random.normal() for i in range(14)]

b = numpy.random.normal()

C = 100000

print(loss(W, X\_validation, y\_validation, b,C))

loss\_train, loss\_validation = optimize(W, X\_train, y\_train, X\_validation, y\_validation, 8, b, 100)

print(loss(W, X\_validation, y\_validation, b, C))

x1=range(1,9)

x2=range(1,9)

plt.figure(figsize=(10, 6))

plt.plot(x1,loss\_train,'',label="train set")

plt.plot(x2,loss\_validation,'',label="validation set")

plt.legend(loc='upper right')

plt.xlabel('times')

plt.ylabel('loss')

plt.show()

main()

## 8. 选择的评估方法:

下降次数太小不需评估

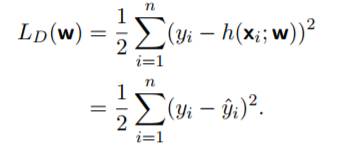
## 9. 模型参数的初始化方法:

随机数初始化

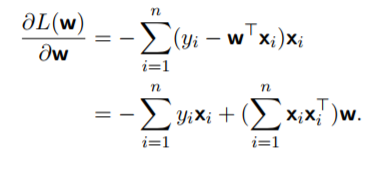
## 10.选择的loss函数及其导数:

10.1. 线性回归：

Loss函数：

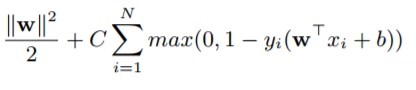


导数：

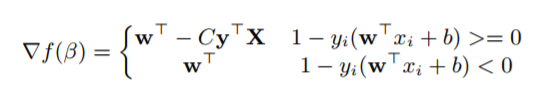


10.2. 线性分类：

Loss函数：



导数：



## 11.实验结果和曲线图:

## 超参数选择：

线性回归：

b为随机初始化

线性分类：

b为随机初始化

C设为100000

## 评估结果：

线性回归

47304.3263929

15621.2604438

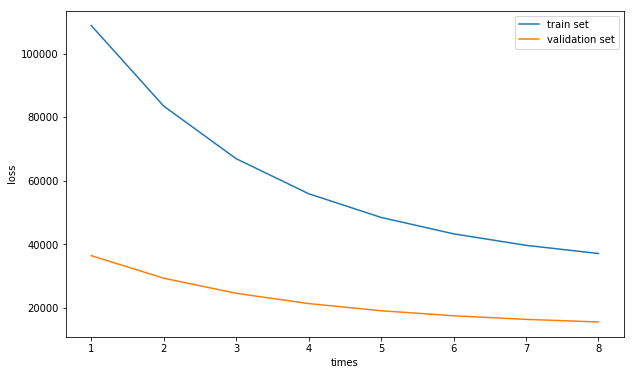
线性分类

43772510.0275

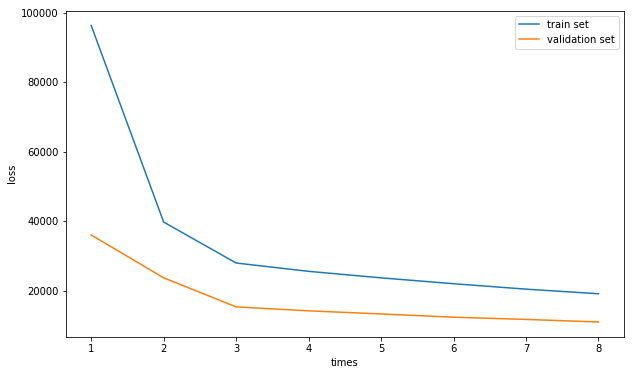
13237099.441

## loss曲线图：

线性回归：



线性分类：



## 12.实验结果分析:

确实能看出来损失有下降，但是最终误差还是很大

## 13.对比线性回归和线性分类的异同点：

基于梯度下降的步骤大同小异，都是先求出W的梯度，然后以一定的步长向梯度的负方向调整W。

## 14.实验总结：

本次实验没有根据数据优化超参数，而是直接选定初始值不变；也没有科学的步长调整方法（步长的数组和选择部分完全本人多次运行目测而得出，而且花了很多时间）；且由于下降次数太少，也没有判定时候停止梯度下降的方法；梯度和损失函数的计算都是根据老师提供的PDF而写成；python语法不熟悉，花了很多时间；还有通过load\_svmlight\_file函数读取的sparse CSR matrix没明白怎么转化为稠密矩阵。总之是一次十分简陋的实验，大概相当于学C++写的hello world。