

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**   **温乔渊**

**学 号 201530612989**

**邮 箱 627947078@qq.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: 温乔渊

## 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G。

6.取梯度G的负方向，记为D。

7.更新模型参数，W=W+aD。a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，过程详见课件ppt。

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，W=W+aD。a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9.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。

## 代码内容:

线性回归和梯度下降:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import time

from sklearn.datasets import load\_svmlight\_file

from sklearn.cross\_validation import train\_test\_split

# 读数据

def get\_data():

data = load\_svmlight\_file("housing.txt")

return data

# 梯度

def g(w, X\_train , y\_train):

g = np.dot(X\_train.transpose(), np.dot(X\_train, w)) - np.dot( X\_train.transpose(),y\_train)

return g

# 画图

def paint(number, loss\_train, loss\_test):

plt.plot(loss\_train,label='Train ')

plt.plot(loss\_test, label='Test')

plt.xlabel('number')

plt.ylabel('loss')

plt.legend()

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

X = get\_data()[0]

y = get\_data()[1]

X = X.toarray()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1)

y\_train = y\_train.reshape(X\_train.shape[0], 1)

y\_test = y\_test.reshape(X\_test.shape[0], 1)

w = np.random.random(size = (X\_train.shape[1], 1))#初始化

a = 0.001#学习率

number = 100 # 循环次数

number\_now = 0 # 当前循环次数

Train = []#cost\_train

Test = []#cost\_test

while number\_now < number:

number\_now += 1

w = w - a \* (g(w,X\_train,y\_train))#更新w

loss\_train = 0.5 \* (y\_train - X\_train.dot(w)).transpose().dot((y\_train - X\_train.dot(w)))

loss\_test = 0.5 \* (y\_test-X\_test.dot(w)).transpose().dot((y\_test - X\_test.dot(w)))

Train.append(loss\_train[0] / X\_train.shape[0])

Test.append(loss\_test[0] / X\_test.shape[0])

print('number', number\_now, ':', ' loss\_train: ',loss\_train, ' loss\_test: ',loss\_test)

paint(number,Train,Test)

线性分类和梯度下降:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import time

from sklearn.datasets import load\_svmlight\_file

from sklearn.cross\_validation import train\_test\_split

def get\_data():

data = load\_svmlight\_file("australian.txt")

return data

def g(X, Y, w,b):

grad = 0.01\*w#1/2c

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

if Y[i] \* (w.transpose().dot(X[i]) + b) < 1:

grad -= (X[i] \* Y[i]).reshape(X.shape[1], 1)

return grad

def Loss(X, Y, w, b):

loss = 0.5 \*0.01\* w.transpose().dot(w)#1/2c

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

if Y[i] \* (w.transpose().dot(X[i]) + b) < 1:

loss = loss +(1 - Y[i] \* (w.transpose().dot(X[i]) + b))

return loss

def paint(loops, train\_loss, test\_loss, train\_rate, test\_rate):

#tloss

plt.plot(train\_loss, label='Train')

plt.plot(test\_loss, label='Test')

plt.xlabel('loop')

plt.ylabel('loss')

plt.title('loss')

plt.legend()

plt.show()

#CorrectRate

plt.plot(train\_rate,label='Train')

plt.plot(test\_rate, label='Test')

plt.xlabel('loops')

plt.ylabel('CorrectRate')

plt.title('CorrectRate')

plt.legend()

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

X = get\_data()[0]

y = get\_data()[1]

X = X.toarray()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

y\_train = y\_train.reshape(X\_train.shape[0], 1)

y\_test = y\_test.reshape(X\_test.shape[0], 1)

w = np.random.random(size = (X\_train.shape[1], 1))

b = 2#初始化

a = 0.0001#学习率

number = 1000

number\_now = 0

losstrain = []

losstest = []

ratetrain = []

ratetest = []

while number\_now < number:

number\_now += 1

w -= a \* g(X\_train, y\_train,w, b)

loss\_train = Loss(X\_train, y\_train,w, b)

loss\_test = Loss(X\_test, y\_test,w, b)

losstrain.append(loss\_train[0] / X\_train.shape[0])

losstest.append(loss\_test[0] / X\_test.shape[0])

#correct rate

number\_train = 0

h1 = y\_train \* (X\_train.dot(w) + b)

for i in h1:

if i > 0:

number\_train += 1

rate\_train = number\_train / h1.shape[0]

ratetrain.append(rate\_train)

#correct rate

number\_test = 0

h2 = y\_test \* (X\_test.dot(w) + b)

for i in h2:

if i > 0:

number\_test += 1

rate\_test = number\_test / h2.shape[0]

ratetest.append(rate\_test)

print('correctrate: train: {}, test: {}'.format(ratetrain[number\_now-1], ratetest[number\_now-1]))

paint(number, losstrain, losstest, ratetrain, ratetest )

（针对线性回归和线性分类分别填写8-12内容）

## 选择的评估方法（留出法，交叉验证，k折交叉验证等）:

均采用留出法

## 9. 模型参数的初始化方法:

w = np.random.random(size = (X\_train.shape[1], 1))

随机初始化

## 选择的loss函数及其导数:

线性回归：

loss = 0.5 \* (y - x.dot(w)).transpose().dot((y - x.dot(w)))

线性分类：

loss = 0.5\*w.transpose().dot(w)+(1 - Y \* (w.transpose().dot(x) + b))

## 实验结果和曲线图:

线性回归：

## 超参数选择（η,epoch等）：

## 学习率0.001

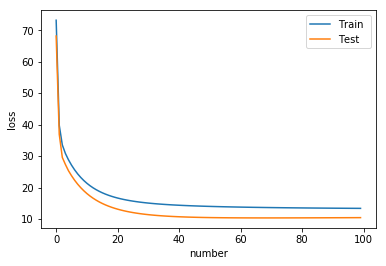
## 评估结果（根据选择的评估方法）：

## cost\_test: [ 10.42457866]

## 预测结果（最佳结果）：

## cost\_train: [ 13.37289136]

## loss曲线图：



线性分类：

## 超参数选择（η,epoch等）：

## 学习率0.0001

## 评估结果（根据选择的评估方法）：

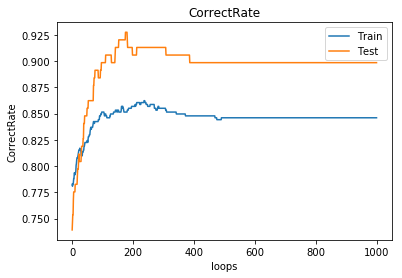
train的分类正确率: 0.8460144927536232,

## 预测结果（最佳结果）：

test的分类正确率: 0.8985507246376812

## loss曲线图：

## 2.1



## 12.实验结果分析:

线性回归：

训练集loss与测试集loss一起减小并差距减小，说明模型经过训练后逐渐变得更加合适

线性分类：

训练集loss与测试集loss一起减小说明模型在继续优化中，正确率的提高也说明了分类模型更加合适

## 13.对比线性回归和线性分类的异同点：

模型参数相似

均采用梯度下降的方法确定了合适的模型

不同之处在于，线性回归的y是一个具体的数值，而线性分类的y则是只有1与-1表示分类

## 14.实验总结：

通过这次实验，更加地了解到了线性回归，线性分类以及梯度下降的原理，作为机器学习中最为基础的方法，这对之后的学习有这很大的帮助。在调整参数过程中也遇到了学习率过大而导致最终结果不断波动的情况，这让我更进一步理解到了调参对实验结果的巨大影响以及其重要性。在实验中将原理转化为代码时遇到了不少问题，经过网上搜索以及求问他人后终于完成了实验，及时的继续学习python语言对以后的学习十分重要。