

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  张雨虹

**学 号 21530613665**

**邮 箱**

**指导教师**

**提交日期** **2017年 月 日**

## 1. 实验题目: 线性回归于线性分类

## 2. 实验时间：2017年 12月 日

## 3. 报告人:张雨虹

## 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](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/)中的[Housing](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html#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](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/)中的[australian](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#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。

为libsvm格式

## 6. 实验步骤:

本次实验代码及画图均在jupyter上完成。

*线性回归和梯度下降*

1. 读取实验数据，使用sklearn库的[load\_svmlight\_file](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html" \t "_blank)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](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)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，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](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html" \t "_blank)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](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)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。
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. 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**。

整理实验结果并完成实验报告（实验报告模板将包含在[示例仓库](https://github.com/chenyaofo/ML2017-lab-01" \t "_blank)中）

## 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

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

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

import sklearn

import numpy as np

import matplotlib.pyplot as plt

import time

import jupyter

%matplotlib inline

mem = Memory("./mycache")

@mem.cache

def get\_data():

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

return data[0], data[1]

# 计算损失函数

def cost\_fn(x, y, heta):

return 0.5 \* (y - x.dot(heta)).transpose().dot((y - x.dot(heta)))

#eturn np.sum((x.dot(heta) - y) \*\* 2) / (2 \* m)

# 计算梯度

def g(x, y, heta):

return np.dot(X\_train.transpose(), np.dot(X\_train, heta)) - np.dot(X\_train.transpose(), y\_train)

#return np.array((y\_pred(X\_train,theta)- y).dot(x)).reshape(theta) / m

X, y = get\_data()

X = X.toarray()

m = y.shape[0]

y=y.reshape(m,1)

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

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

alpha = 0.0011#学习率

# 迭代的次数

iteration = 200

Loss\_train = []

Loss\_val = []

# 梯度下降法更新参数

for i in range(iteration):

#更新参数

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

loss\_train = (y\_train - X\_train.dot(heta)).transpose().dot((y\_train - X\_train.dot(heta)))/2

loss\_val = (y\_val-X\_val.dot(heta)).transpose().dot((y\_val - X\_val.dot(heta)))/2

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

Loss\_val.append(loss\_val[0] / X\_val.shape[0])

plt.plot(Loss\_train,label='Loss\_train ')

plt.plot(Loss\_val, label='Loss\_val')

plt.xlabel('iteration')

plt.ylabel('loss')

plt.legend()

plt.show()

线性分类

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

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

import sklearn

import numpy as np

import matplotlib.pyplot as plt

import time

import jupyter

%matplotlib inline

mem = Memory("./mycache")

@mem.cache

def get\_data():

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

return data[0], data[1]

#计算梯度

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

#计算损失函数

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:

return loss +(1 - y[i] \* (w.transpose().dot(x[i]) + b))

X, y = get\_data()

X = X.toarray()

m = y.shape[0]

y=y.reshape(m,1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=10)

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

b = 2

a = 0.0001

iteration = 100

# 梯度下降法更新参数

Loss\_train = []

Loss\_val = []

Rate\_train = []

Rate\_val = []

for i in range(iteration):

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)

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

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

#correct rate

iteration\_train = 0

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

for i in h1:

if i > 0:

iteration\_train += 1

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

Rate\_train.append(rate\_train)

#correct rate

iteration\_test = 0

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

for i in h2:

if i > 0:

iteration\_test += 1

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

Rate\_val.append(rate\_test)

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

#tloss

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

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

plt.xlabel('iteration')

plt.ylabel('loss')

plt.title('loss')

plt.legend()

plt.show()

paint(iteration, Loss\_train,Loss\_val, Rate\_train, Rate\_val )

## 8. 选择的评估方法（留出法，交叉验证，k折交叉验证等）:

留出法

## 9. 模型参数的初始化方法:

随机取

## 10.选择的loss函数及其导数:

线性回归

def cost\_fn(x, y, heta):

return 0.5 \* (y - x.dot(heta)).transpose().dot((y - x.dot(heta)))

#eturn np.sum((x.dot(heta) - y) \*\* 2) / (2 \* m)

线性分类

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:

return loss +(1 - y[i] \* (w.transpose().dot(x[i]) + b))

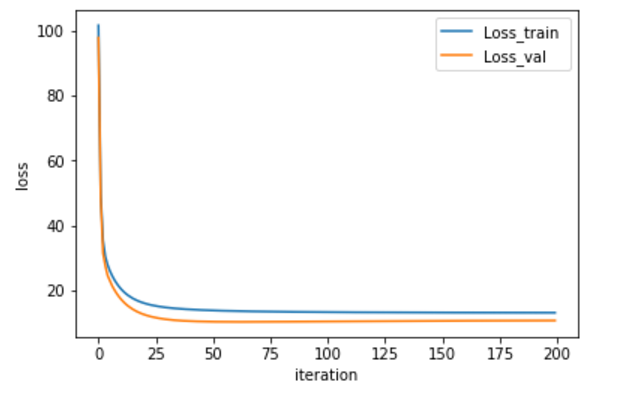
## 11.实验结果和曲线图:

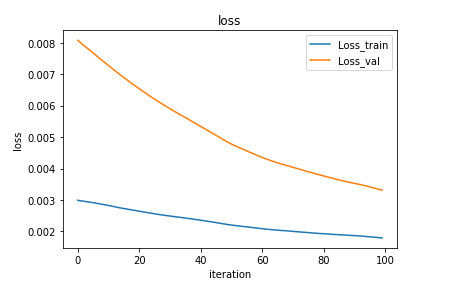
## 超参数选择（η,epoch等）：0.0011，0.0001

## 评估结果（根据选择的评估方法）：

## 预测结果（最佳结果）：

## loss曲线图：





## 12.实验结果分析:

## 13.对比线性回归和线性分类的异同点：

两个问题本质上都是一致的 但是分类问题(超平面分割)的更多的是一定区域的一些下，对应着一个y。回归问题（用来分析两个变量 x y）更倾向于很小区域内的x或者一般是一个x对应着一个y

## 14.实验总结：

代码实现上还是很困难，要继续加油