

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  王煜

**学 号 201530612927**

**邮 箱 1628021431@qq.com**

**指导教师**  **吴庆耀**

**提交日期** **2017年 月 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:王煜

## 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。

## 6. 实验步骤:

一、线性回归

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)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**

二、线性分类

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)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**。

## 7. 代码内容:

（针对线性回归和线性分类分别填写8-12内容）

* 1. 线性回归

# -\*- coding: utf-8 -\*-

from numpy import \*

import numpy as np

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

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

import matplotlib.pyplot as plt

#import time

import warnings

warnings.filterwarnings("ignore")

mem = Memory("./mycache")

@mem.cache

def get\_data():

data = load\_svmlight\_file("C:/Users/wangyu/Desktop/housing\_scale.txt")

return data[0], data[1]

def model(theta,x,y): #模型

return ((x.dot(theta)-y)\*\*2).sum()/2

def grad(theta,x,y): #梯度计算

return x.T.dot(x.dot(theta)-y)

X, y = get\_data()

X=X.toarray()

X=np.c\_[np.ones(len(X)),X]#在训练集前添加一列1

#切割数据集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=42)

theta = np.zeros(X\_train.shape[1])

L\_train = []

L\_validaion = []

iters = 8#迭代次数

for i in range(iters):

G = grad(theta,X\_train,y\_train)

learning\_rate = 0.0004#学习率

theta = theta - learning\_rate\*G#更新theta

L\_train.append(model(theta,X\_train,y\_train))

L\_validaion.append(model(theta,X\_test,y\_test))

plt.plot(range(iters),L\_train, label='train loss')

plt.plot(range(iters),L\_validaion, label='validtion loss')

plt.xlabel('iters')#横坐标

plt.ylabel('loss')#纵坐标

plt.legend()

plt.show()

* 1. 线性分类

# -\*- coding: utf-8 -\*-

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

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

import numpy as np

from matplotlib import pyplot as plt

import warnings

warnings.filterwarnings("ignore")

mem = Memory("./mycache")

@mem.cache

def get\_data():

data = load\_svmlight\_file("C:/Users/wangyu/Desktop/australian\_scale.txt")

return data[0], data[1]

# loss function

def loss(X, y, theta, C):

l = 0

idex = 0

y = mat(y)

y = y.T

for line in y:

s = (y[idex,:] \* (np.dot(X[idex,:], theta))).sum()

l = (theta \*\* 2).sum() / 2 + C \* max(0,1 - s)

idex = idex+1

return l

def grad(X, y, w, C): #梯度#w是要更新的参数

temp = np.zeros(len(w))

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

if y[i] \* (X[i,:].dot(w)) < 1:

temp += -y[i] \* X[i,:]

return w + C \* temp

#读取实验数据

X, y = get\_data()

X = X.toarray()#将X转化为数组

#在训练集前添加一列1

X = np.c\_[np.ones(len(X)), X]

#数据集切分

X\_train,X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=42)

#参数初始化

w = np.zeros(X\_train.shape[1])

C = 5

L\_train = []

L\_validation = []

iters = 150#迭代次数

for i in range(iters):

#求梯度

G = grad(X\_train, y\_train, w, C)

learning\_rate = 0.00005

#更新参数

w = w - learning\_rate \* G

L\_train.append(loss(X\_train, y\_train, w, C))

L\_validation.append(loss(X\_test, y\_test, w, C))

#画图

plt.plot(range(iters),L\_train,label='train loss')

plt.plot(range(iters),L\_validation, label='validation loss')

plt.xlabel('iters')#x轴

plt.ylabel('loss')#y轴

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）:

此次的线性回归和线性分类均为留出法来进行评估。

## 9模型参数的初始化方法:

* 1. 线性回归

将训练集的X矩阵按列填充，每个位置都初始化为0.



* 1. 线性分类

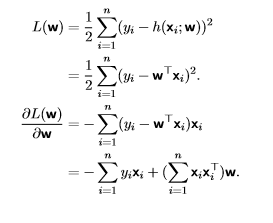
将训练集的w矩阵按列填充为全0

超参数C设置为5（经过多次调参之后为5）

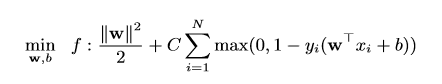


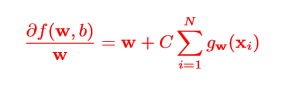
## 选择的loss函数及其导数:

* 1. 线性回归



* 1. 线性分类





## 11.实验结果和曲线图:

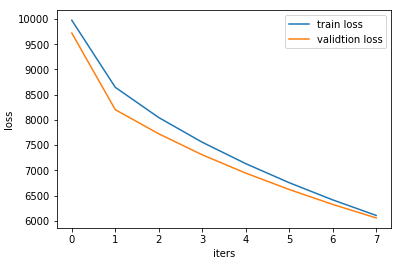
## 超参数选择（η,epoch等）：

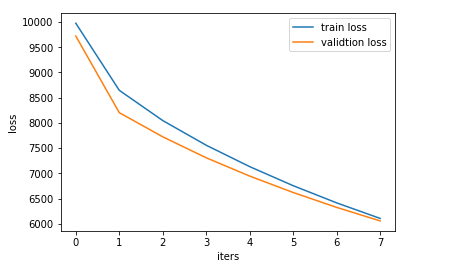
线性回归：学习率learning\_rate=0.001

线性分类：学习率learning\_rate=0.00005,C=5

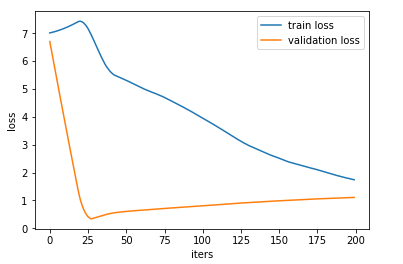
## 评估结果（根据选择的评估方法）

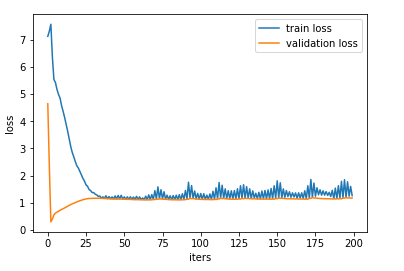
* 1. 线性回归





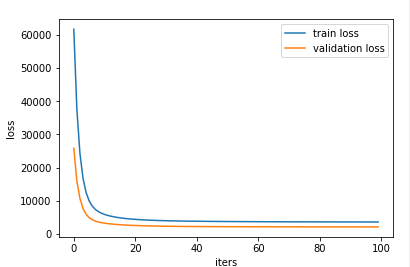
* 1. 线性分类



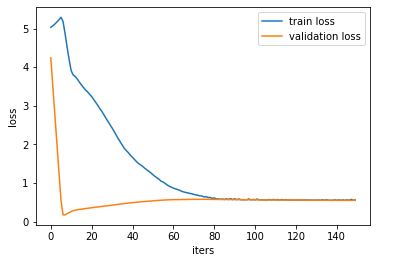


## 预测结果（最佳结果）

* 1. 线性回归

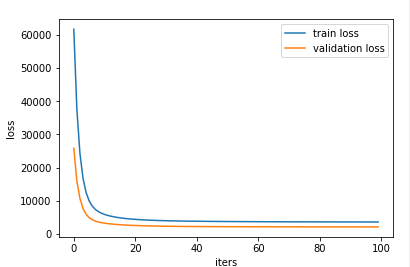


* 1. 线性分类

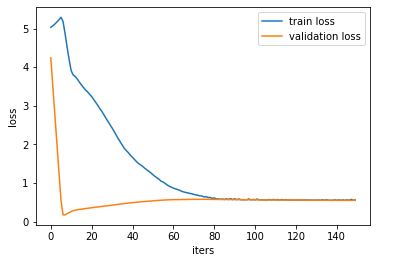


## loss曲线图：

一、线性回归



二、线性分类



## 实验结果分析:

* 1. 线性回归

经过对参数的调节和对模型的优化可以看到，loss函数不断降低，最终我们得到了趋于收敛并且loss降低到一定程度后几乎不再变化，此时我们可以说得到了较好的模型，并且可以用这个模型去预测这个数据集里的其他数据。

* 1. 线性分类

同样的，也是通过对于参数的调节和对loss函数的梯度下降方法来更新参数，可以看出loss在后续迭代中趋于收敛并且小于一。这是经过数次调参之后的最好的结果。

总之，最后得到的结果都是loss函数逐渐变小的模型。

## 对比线性回归和线性分类的异同点：

相同点：

本质上是相同的，都是进行模型的选取和拟合

不同点：

分类问题的y值（label）相对于回归问题更加离散化一些。也就是说，分类问题一半时一定区域的x对应着一个y，但是回归问题更倾向于一个x对应着一个y。

## 13.实验总结：

（1）对于线性回归和线性分类的问题要区别开，一个是连续值得预测，另一个是离散值的预测，对于不同的数据集要选择不同的方法。

（2）拟合这两种模型，都可以通过梯度下降的方法进行，通过调整学习率和其他一些参数，使最终的loss趋于收敛，即是我们所需的模型。

（3）在实验过程中，需要先对这两个的模型和求解方法进行深入理解才能写出代码。

（4）在写代码的过程中需要用到很多库的很多函数，这需要不断地去搜索，去学习，只有不停学习，才能做出来更好的结果。