

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**   **胡菁菁**

**学 号 201530611609**

**邮 箱 767816313@qq.com**

**指导教师**  **谭明奎**

**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：胡菁菁

## 4. 实验目的:

## 进一步理解线性回归和梯度下降的原理。

## 在小规模数据集上实践。

## 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，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，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

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，*Wt = Wt-1 +* 。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*Ltrain*，在验证集上测试并得到Loss函数值*Lvalidation*。
9. 重复步骤5-8若干次，画出*Ltrain*和*Lvalidation*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，*Wt = Wt-1 +* 。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*Ltrain*，在验证集上测试并得到Loss函数值*Lvalidation*。
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:

线性回归：

%matplotlib inline

from sklearn import datasets as ds

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

import numpy as np

import matplotlib as mpl

import matplotlib.pyplot as plt

#读取libsvm格式数据

train\_data,train\_target=ds.load\_svmlight\_file("/Users/humeng/Desktop/housing\_scale.txt")

#调整数据格式

train\_target = train\_target.reshape(-1, 1)

#避免过拟合，采用交叉验证，验证集占训练集30%，

x\_train, x\_validation, y\_train, y\_validation = train\_test\_split(train\_data,train\_target, test\_size=0.3, random\_state=0)

#print (x\_train.shape,x\_validation.shape)

#初始化参数b

b1 = np.ones((x\_train.shape[0],1))

x\_train = np.hstack((x\_train.toarray(),b1))

b2 = np.ones((x\_validation.shape[0],1))

x\_validation = np.hstack((x\_validation.toarray(),b2))

#参数初始化

l\_train = []

l\_validation = []

learningrate = 0.001

lamda = 1

loop = 100

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

#对loss函数求导

def gradient(lamda,w,x,y):

return (lamda \* w - np.dot(x.T,(y - np.dot(x,w))))/x.shape[1]

#求loss

def loss(w,x,y):

l = float((0.5 \* lamda \* np.dot(w.T,w) + 0.5 \* np.dot((y - np.dot(x,w)).T,(y - np.dot(x,w))))/x.shape[0])

return l

for k in range(loop):

#求得所有样本对Loss函数的梯度

grad = gradient(lamda,w,x\_train,y\_train)

w = w - learningrate \* grad

l\_train.append(loss(w,x\_train,y\_train))

l\_validation.append(loss(w,x\_validation,y\_validation))

#设置图像

plt.rcParams['figure.figsize'] = (10.0, 8.0)

plt.plot(np.arange(loop),l\_train,label='Train loss')

plt.plot(np.arange(loop),l\_validation,label='Validation loss')

plt.xlabel('Loop')

plt.ylabel('Loss')

plt.legend(loc='best')

plt.grid()

plt.show()

线性分类：

%matplotlib inline

from sklearn import datasets as ds

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

import numpy as np

import matplotlib as mpl

import matplotlib.pyplot as plt

#读取libsvm格式数据

train\_data,train\_target=ds.load\_svmlight\_file("/Users/humeng/Desktop/australian\_scale.txt")

#调整数据格式

train\_target = train\_target.reshape(-1, 1)

#避免过拟合，采用交叉验证，验证集占训练集30%，

x\_train, x\_validation, y\_train, y\_validation = train\_test\_split(train\_data,train\_target,

test\_size=0.3, random\_state=0)

#初始化参数b

b1 = np.ones((x\_train.shape[0],1))

x\_train = np.hstack((x\_train.toarray(),b1))

b2 = np.ones((x\_validation.shape[0],1))

x\_validation = np.hstack((x\_validation.toarray(),b2))

#print (x\_train.shape,x\_validation.shape)

#参数初始化

C = 10

learningrate = 0.008

loop = 1000

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

l\_train = []

l\_validation = []

best = 0.0

#对loss函数求导

def gradient(w,C,x,y):

#设立阀值

condition = 1 - np.multiply(y,np.dot(x,w))

y[condition < 0] = 0

#求梯度

grad = w - C \* np.dot(x.T, y)

#消除参数b的影响

grad[-1] -= w[-1]

return grad

#梯度下降

def loss(w,C,x,y):

hinge = np.maximum(0, 1 - np.multiply(y,np.dot(x,w)))

l = float(0.5 \* np.dot(w.T,w) + C \* np.sum(hinge) / x.shape[0])

return l

#求准确率

def acc(w, x, threshold=0.0):

score = x.dot(w)

y\_pre = np.ones((x.shape[0], 1))

y\_pre[score < threshold] = -1

return y\_pre

for k in range(loop):

#求得所有样本对Loss函数的梯度

grad = gradient(w,C,x\_train,y\_train)

w = w - learningrate \* grad

l\_train.append(loss(w,C,x\_train,y\_train))

l\_validation.append(loss(w,C,x\_validation,y\_validation))

#计算准确率

y\_pre = acc(w, x\_train, threshold=0.0)

a = np.mean(y\_pre == y\_train)

if a > best:

best = a

print(l\_train[-1],l\_validation[-1])

print('Best accuracy is %f' % (best))

#设置图像

plt.rcParams['figure.figsize'] = (10.0, 8.0)

plt.plot(np.arange(loop),l\_train,label='Train loss')

plt.plot(np.arange(loop),l\_validation,label='Validation loss')

plt.xlabel('Loop')

plt.ylabel('Loss')

plt.legend(loc='best')

plt.grid()

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：留出法

## 9. 模型参数的初始化方法:

线性回归：b初始化为1，w全零初始化，学习率、和迭代次数随机初始化

线性分类：b初始化为1，w全零初始化，学习率、*C*和迭代次数随机初始化

## 10.选择的loss函数及其导数:

线性回归

loss :

gradient :

线性分类

loss :

gradient :

## 11.实验结果和曲线图:

## 超参数选择（η,epoch等）：

线性回归：η= 0.001，epoch = 1000

线性分类：C = 10，η= 0.008，epoch = 1000

## 评估结果（根据选择的评估方法）：

线性回归：

1. learningrate = 0.0001，lamda = 1，loop = 100

loss未完全收敛，train\_loss = 57.78，validation\_loss = 57.18

1. learningrate = 0.001，lamda = 1，loop = 100

loss收敛，train\_loss = 20.57，validation\_loss=25.41

1. learningrate = 0.001，lamda = 1，loop = 1000

loss收敛，train\_loss = 10.86，validation\_loss=15.33

1. learningrate = 0.001，lamda = 1，loop = 10000

loss收敛，train\_loss = 10.62，validation\_loss=15.11

线性分类：

1. C = 1，learningrate = 0.0001，loop = 100

Loss未收敛，train\_loss 趋于无穷，validation\_loss趋于无穷

1. C = 1，learningrate = 0.008，loop = 100

Loss未收敛，train\_loss = 1.41，validation\_loss = 2.04

1. C = 10，learningrate = 0.008，loop = 100

Loss收敛，train\_loss = 24.99，validation\_loss = 41.94

1. C = 10，learningrate = 0.008，loop = 1000

Loss收敛，train\_loss = 10.38，validation\_loss = 9.16

## 预测结果（最佳结果）：

线性回归：

learningrate = 0.001，lamda = 1，loop = 10000

loss收敛，train\_loss = 10.62，validation\_loss=15.11

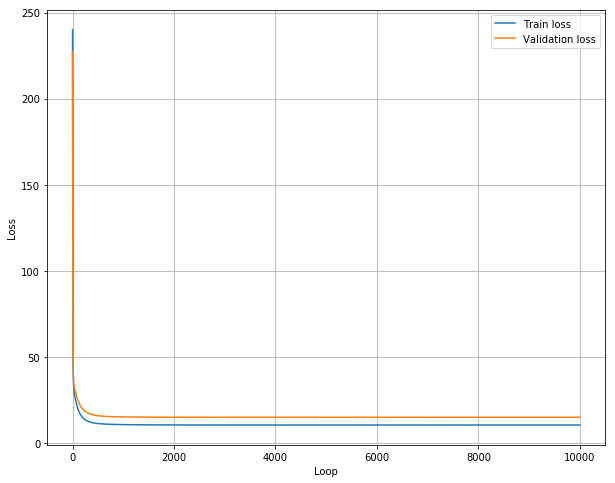
线性分类：

C = 10，learningrate = 0.008，loop = 1000

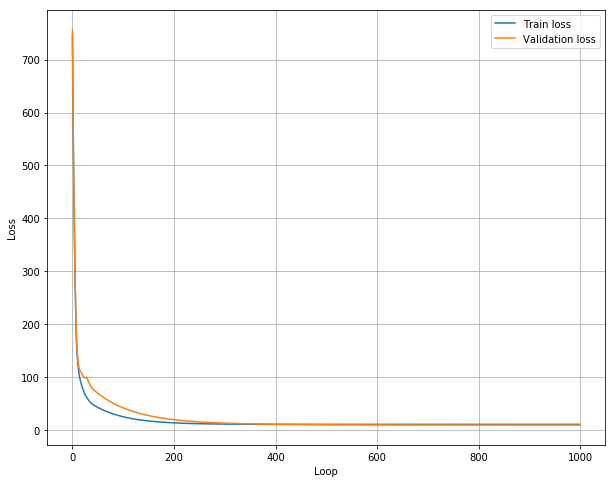
Loss收敛，train\_loss = 10.38，validation\_loss = 9.16

## loss曲线图：

线性回归



线性分类



## 12.实验结果分析:

线性回归：loss在1000次迭代后趋于稳定

线性分类：loss在300次迭代后趋于稳定，正确率达到0.838509

## 13.对比线性回归和线性分类的异同点：

* 线性回归是连续变量预测，线性分类是离散变量预测
* 线性分类需要设定阀值，线性回归不需要阀值
* 线性回归的目的是找到最优拟合，线性分类的目的是寻找决策边界
* 两者模型都是线性的，均可以用梯度下降来解决问题

## 14.实验总结：

线性回归和线性分类模型不算复杂，均可用梯度下降方式得到最优模型，通过代码实现，比较直观地了解了梯度下降算法的原理和步骤。得到结果后，调参需要耐心和技巧。