

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  姚永铖

**学 号 201530613450**

**邮 箱 337144043@qq.com**

**指导教师** 吴庆耀

**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:姚永铖

## 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：

1. 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。
2. 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:

*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+ηD。η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+ηD。η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Ltrain和Lvalidation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）

线性回归

from sklearn.datasets import load\_svmlight\_file

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

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

X,Y = load\_svmlight\_file("housing\_scale.txt")

# x add column

row=X.shape[0]

B = np.mat(np.ones((X.shape[0],1)))

X=np.mat(np.c\_[X.A,B])

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X,Y, test\_size=0.33, random\_state=42)

#参数初始化全零

w=np.mat(np.zeros((14,1)))

learning\_rate=0.03

#迭代次数

num=100

loss\_train=[]

loss\_validation=[]

#模型

def score(w,x):

return x\*w

#loss函数

def loss(x,y,w):

y=y.reshape(len(y),-1)

return 0.5\*(y-x\*w).T\*(y-x\*w)/x.shape[0]

#求导

def gradient(x,y,w):

y=y.reshape(len(y),-1)

return (-x.T\*y+x.T\*x\*w)/x.shape[0]

#梯度下降

def gradient\_descent(w):

temp\_w=w-learning\_rate\*gradient(x\_train,y\_train,w)

return temp\_w

#优化

def optimizer(w):

temp\_w=w

for i in range(num):

temp\_w=gradient\_descent(temp\_w)

loss\_train.append(loss(x\_train,y\_train,temp\_w)[0,0])

loss\_validation.append(loss(x\_test,y\_test,temp\_w)[0,0])

return temp\_w

w=optimizer(w)

n = np.arange(num)

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

plt.plot(n,loss\_validation,label='loss\_validation')

plt.legend(loc='upper right')

plt.xlabel('time')

plt.ylabel('loss')

plt.title('Linear Regression')

线性分类

from sklearn.datasets import load\_svmlight\_file

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

import numpy as np

import numpy.linalg as nl

import matplotlib.pyplot as plt

%matplotlib inline

X,Y = load\_svmlight\_file("australian\_scale.txt")

row=X.shape[0]

B = np.mat(np.ones((X.shape[0],1)))

X=np.mat(np.c\_[X.A,B])

Y=Y.reshape(len(Y),-1)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X,Y, test\_size=0.33, random\_state=42)

#初始化全零

W=np.mat(np.zeros((15,1)))

learning\_rate=0.03

C=1

#迭代次数

num=100

#阀值

threshold=0

loss\_train=[]

loss\_validation=[]

#目标函数

def score(w,x):

return x\*w

#loss函数

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

temp=0

for i in range(len(y)):

temp+=max(1-y[i,0]\*score(w,x[i]),0)

return 0.5\*w.T.dot(w)+C\*temp

#求梯度

def gradient(x,y,w):

temp=0

for i in range(len(x)):

temp+=w.T

if (1-y[i,0]\*score(w,x\_train[i]))>=0:

temp=temp-(y[i]\*x[i])\*C

return temp.T/x\_train.shape[0]

#梯度下降

def gradient\_descent(w):

temp\_w=w-learning\_rate\*gradient(x\_train,y\_train,w)

return temp\_w

#优化

def optimizer(w):

temp\_w=w

for i in range(num):

temp\_w=gradient\_descent(temp\_w)

loss\_train.append(loss(x\_train,y\_train,temp\_w,C)[0,0])

loss\_validation.append(loss(x\_test,y\_test,temp\_w,C)[0,0])

return temp\_w

W=optimizer(W)

# print(W)

#训练集命中率

train\_hit=0

for i in range(len(x\_train)):

if (score(W,x\_train[i]) >= threshold and y\_train[i] == 1) or (score(W,x\_train[i]) < threshold and y\_train[i] == -1):

train\_hit += 1

#验证集命中率

test\_hit=0

for i in range(len(x\_test)):

if (score(W,x\_test[i]) >= threshold and y\_test[i] == 1) or (score(W,x\_test[i]) < threshold and y\_test[i] == -1):

test\_hit += 1

print(train\_hit/len(x\_train))

print(test\_hit/len(x\_test))

n = np.arange(num)

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

plt.plot(n,loss\_validation,label='loss\_validation')

plt.legend(loc='upper right')

plt.xlabel('time')

plt.ylabel('loss')

plt.title('Linear Classification')

## 8. 选择的评估方法（留出法，交叉验证，k折交叉验证等）:

## 交叉验证

## 9. 模型参数的初始化方法:

## 线性回归：W全零初始化、learning\_rate=0.02、迭代100次

## 线性分类：W全零初始化、learning\_rate=0.02、迭代100次、C=1、 阈值=0

## 10.选择的loss函数及其导数:

## 线性回归

## loss：最小二乘法 1/2\*(y − Xw) T (y − Xw)

## 导数：∂L(w)/∂w=−XTy + XTXw

## 线性分类

## loss：

## 导数：

## 11.实验结果和曲线图:

## 超参数选择（η,epoch等）： η=0.02 η=0.03

## 评估结果（根据选择的评估方法）：

## 线性回归： η=0.02

## 

## 线性分类：η=0.02

## 

## 预测结果（最佳结果）：

## 线性回归： η=0.03

## 线性分类： η=0.03 C=1 threshold=0

## loss曲线图：

## 

## 

## 12.实验结果分析:

## 随着迭代次数，loss明显降低，说明了模型越拟合数据

## 13.对比线性回归和线性分类的异同点：

## 两者都是模型对数据的拟合，线性回归是由输入，预测输出，大多情况是一对一，而线性分类多个输入可能对应一个输出，将输入进行分类，正类或负类。

## 14.实验总结：

## 实验中一步步选取模型、选取loss函数、求梯度、以及进行梯度下降，让我对线性回归、线性分类有了更深的了解。