

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  **苏婉婷**

**学 号 201530612767**

**邮 箱 352191482@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: 苏婉婷

## 4. 实验目的:

## 4.1.进一步理解线性回归和梯度下降的原理。

## 4.2在小规模数据集上实践。

## 4.3体会优化和调参的过程。

## 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/" \t "https://www.zybuluo.com/chenyaofo/note/_blank)中的[Housing](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html" \l "housing" \t "https://www.zybuluo.com/chenyaofo/note/_blank)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](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/" \t "https://www.zybuluo.com/chenyaofo/note/_blank)中的[australian](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html" \l "australian" \t "https://www.zybuluo.com/chenyaofo/note/_blank)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。

## 实验步骤:

## 6.1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 "https://www.zybuluo.com/chenyaofo/note/_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" \t "https://www.zybuluo.com/chenyaofo/note/_blank)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。

## 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。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，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。

6.2**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 "https://www.zybuluo.com/chenyaofo/note/_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" \t "https://www.zybuluo.com/chenyaofo/note/_blank)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。

5.求得所有样本对Loss函数的梯度G。

6.取梯度的负方向G，记为D。

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:

7.1 线性回归源代码

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

"""

Created on Wed Dec 6 15:56:41 2017

@author: sususu

"""

from sklearn import datasets

from sklearn import model\_selection

import numpy as np

import matplotlib.pyplot as plt

def get\_data():

X,y = datasets.load\_svmlight\_file("C:/Users/siye/Desktop/housing\_scale.txt",n\_features=13)

X = np.matrix(X.toarray())

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

X = np.concatenate((ones,X),axis=1)

y = np.matrix(y).T

train\_x,test\_x,train\_y,test\_y = model\_selection.train\_test\_split(X,y,test\_size=0.2)

return train\_x,test\_x,train\_y,test\_y

def loss(X,y,w):

m = y.shape[0] #读取矩阵的长度

return 0.5\*(((y-X\*w).T\*(y-X\*w))).sum()/m

def gradient(X,y,w):

return X.T\*(X\*w-y)

def gradientDecent(X,y,w,alpha,time\_iteration,valid\_x,valid\_y):

m = y.shape[0]

train\_loss = []

valid\_loss = []

print("Original train loss:%f"%loss(X,y,w))

train\_loss.append(loss(X,y,w))

print("Original validation loss:%f"%loss(valid\_x,valid\_y,w))

valid\_loss.append(loss(valid\_x,valid\_y,w))

print("")

for i in range(time\_iteration):

w=w-gradient(X,y,w)\*alpha/m

#train\_loss=loss(X,y,w)

#valid\_loss=loss(valid\_x,valid\_y,w) #是valid\_x,valid\_y

train\_loss.append(loss(X,y,w))

valid\_loss.append(loss(valid\_x,valid\_y,w))

return w,train\_loss,valid\_loss

def train(X,y,valid\_x,valid\_y):

m=X.shape[1]

init\_w=np.matrix(np.zeros(m)).T

print("Begin to train")

#学习率

alpha=0.13

#迭代次数

time\_iteration=50

print("Learning rate alpha: %f"%alpha)

print("Iteration times:%d"%time\_iteration)

print("")

w,train\_loss,valid\_loss=gradientDecent(X,y,init\_w,alpha,time\_iteration,valid\_x,valid\_y)

#绘图

plt.plot(np.arange(time\_iteration+1),train\_loss,label='train loss')

plt.plot(np.arange(time\_iteration+1),valid\_loss,label='validation loss')

plt.legend(loc=1)

plt.xlabel('times\_of\_iteration')

plt.ylabel('loss')

return w,train\_loss,valid\_loss

train\_x,test\_x,train\_y,test\_y = get\_data()

w,train\_loss,val\_loss = train(train\_x,train\_y,test\_x,test\_y)

print("final train loss:%f"%train\_loss.pop())

print("final validation loss:%f"%val\_loss.pop())

7.2 线性分类源代码

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

"""

Created on Thu Dec 7 15:30:07 2017

@author: sususu

"""

from sklearn import datasets

from sklearn import model\_selection

import numpy as np

import matplotlib.pyplot as plt

def get\_data():

#读取实验数据，使用sklearn库的load\_svmlight\_file函数读取数据

X,y =datasets.load\_svmlight\_file('C:/Users/siye/Desktop/australian\_scale.txt',n\_features=14)

X=np.matrix(X.toarray())

y=np.matrix(y).T

#将数据集切分为训练集和验证集

train\_x,test\_x,train\_y,test\_y=model\_selection.train\_test\_split(X,y,test\_size=0.2)

return train\_x,test\_x,train\_y,test\_y

#选择loss 函数

def loss(X,y,w,b,C):

m = y.shape[0]

hinge = sum(list(map(lambda x:max(0,x[0]),(1-y.A\*(X\*w+b).A))))

w\_2 = sum(w.A\*\*2)[0]

return (0.5\*w\_2+C\*hinge)/m

def gradient(X,y,w,C,b):

m = y.shape[0]

dw = np.zeros((X.shape[1],1))

db = 0

db = 0

indicator = 1-y.A\*((X\*w+b).A)

for i in range(m):

if indicator[i]>=0:

dw += w - C\*(y[i]\*X[i]).T

db += -C\*y[i]

else:

dw += w

return [dw,db]

def checkGradient(X,y,w,C,b):

delta = 1e-6

dw = (loss(X,y,w+delta,b,C)-loss(X,y,w-delta,b,C))/(np.ones(w.shape)\*delta\*2)

db = (loss(X,y,w,b+delta,C)-loss(X,y,w,b-delta,C))/delta\*2

return dw,db

def gradientDecent(X,y,w,C,b,alpha,num\_rounds,val\_x,val\_y):

m = y.shape[0]

train\_loss = []

valid\_loss = []

print("origin train loss:%f"%loss(X,y,w,b,C))

train\_loss.append(loss(X,y,w,b,C))

print("origin validation loss:%f"%loss(val\_x,val\_y,w,b,C))

valid\_loss.append(loss(val\_x,val\_y,w,b,C))

print("")

for i in range(num\_rounds):

new\_w = w - gradient(X,y,w,C,b)[0]\*alpha/m

new\_b = b - gradient(X,y,w,C,b)[1]\*alpha/m

w = new\_w

b = new\_b

train\_loss.append(loss(X,y,w,b,C))

valid\_loss.append(loss(val\_x,val\_y,w,b,C))

return w,b,train\_loss,valid\_loss

def predict(X,y,w,b):

pred = X\*w+b

pred\_y = list(map(lambda x:1 if x[0]>0 else -1,pred.A))

acc = (y.A1==pred\_y).sum()/len(y.A)

print("acc:%f"%acc)

def train(X,y,val\_x,val\_y):

m = X.shape[1]

init\_w = np.matrix(np.zeros(m)).T

print("begin to train")

print("")

C=1

b = 1

#学习率

alpha = 0.1

#迭代次数

time\_iteration=100

w,b,train\_loss\_history,val\_loss\_history=gradientDecent(X,y,init\_w,C,b,alpha,time\_iteration,val\_x,val\_y)

plt.plot(np.arange(time\_iteration+1),train\_loss\_history,label='train loss')

plt.plot(np.arange(time\_iteration+1),val\_loss\_history,label='validation loss')

plt.legend(loc=1)

plt.xlabel('number\_of\_rounds')

plt.ylabel('loss')

return w,b,train\_loss\_history,val\_loss\_history

train\_x,test\_x,train\_y,test\_y = get\_data()

w,b,train\_loss,val\_loss = train(train\_x,train\_y,test\_x,test\_y)

print("final train loss:%f"%train\_loss.pop())

print("final validation loss:%f"%val\_loss.pop())

print("train:")

predict(train\_x,train\_y,w,b)

print("test:")

predict(test\_x,test\_y,w,b)

**8.选择的评估方法（留出法，交叉验证，k折交叉验证等）:**

## 8.1 线性回归：留出法

## 8.2 线性分类：留出法

## 9. 模型参数的初始化方法:

## 9.1 线性回归：线性模型参数初始化，全零初始化

## 9.2 线性分类：支持向量机模型参数初始化，全零初始化

## 10.选择的loss函数及其导数:

## 10.1 线性回归：

## 10.2 线性分类：

## 实验结果和曲线图:

**11.1 线性回归**

## 1）超参数选择（η,epoch等 ）：

学习率η在本实验中表示为alpha，且alpha的取值为 0.13

## （2）评估结果（根据选择的评估方法）：

Original validation loss:265.449167

Final validation loss:15.899950

## （3）预测结果（最佳结果）：

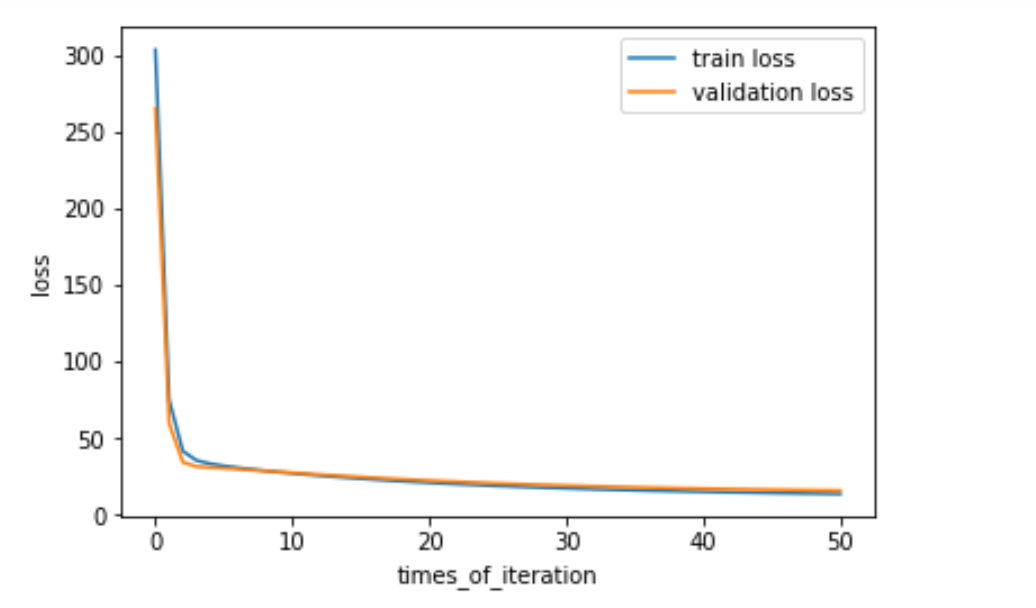
最初的 train loss:303.805334

最初的 validation loss:265.449167

最终的 train loss:13.706270

最终的 validation loss:15.899950

## （4）loss曲线图：



**11.2 线性分类**

## （1）超参数选择（η,epoch等 ）：

学习率η在本实验中表示为alpha，且alpha的取值为0.1

## （2）评估结果（根据选择的评估方法）：

origin validation loss:1.072464

final validation loss:0.447242

## （3）预测结果（最佳结果）：

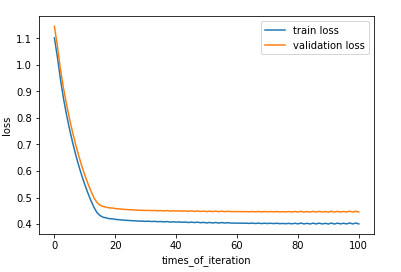
origin train loss:1.119565

origin validation loss:1.072464

final train loss:0.400619

final validation loss:0.447242

## （4）loss曲线图：



## 实验结果分析:

在线性回归和线性分类中，通过不断调试学习率，最终结果都得到了有效的收敛。

## 对比线性回归和线性分类的异同点：

线性回归和线性分类本质上都是一致的，就是模型的拟合（匹配）。

分类问题的y值更离散化， 同事对于同一个y值可能对应一大批x,  x是具有一定范围的。

分类问题更多的是一定区域的一些x对应着一个y，而回归问题的模型更倾向于很小区域内的x，或一般是一个x对应一个y。

## 14.实验总结：

机器学习中最基本的两类问题，线性回归和分类。 对于这两类问题都有以下几个步骤：如何选取一个 合理的模型(线性的，or 非线性的(e.g. 阶跃函数， 高斯函数)).，制造一个"美好"的 误差函数 (可以评估拟合程度，而且还是convex函数)，采取可能的技术(e.g. 导数下降法，解极值方程法) 求出最好的模型参数。

通过本次的实验，实现了Linear Model的算法，进一步加深了对线性回归和线性分类模型的理解。