

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  **舒梓杰**

**学 号 201530612712**

**邮 箱** [**510788094@qq.com**](mailto:510788094@qq.com)

**指导教师**  **吴庆耀**

**提交日期** **2017年12月8日**

## 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：舒梓杰

## 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中的Housing数据，包含506个样本，每个样本有13个属性，我们将其切分为训练集，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，我们将其切分为训练集，验证集。

## 6. 实验步骤：

**（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 "_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*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。

**（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 "_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*G*。

⑥取梯度的负方向*G*，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：

**（1）线性回归和梯度下降：**

import numpy as np

from matplotlib import pyplot as plt

from sklearn.datasets import load\_svmlight\_file

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

def loadDataSet():

# 读取数据

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

#将稀疏矩阵转化为完整特征矩阵

X.todense()

# 将数据集切分为训练集和验证集

X\_train, X\_validation, y\_train, y\_validation = train\_test\_split(X, y, random\_state=0)

print(X\_train.shape,y\_train.shape)

return X\_train, X\_validation, y\_train, y\_validation

def gradDescent(alpha,maxCycles,X\_data,y\_data):

num = y\_data.shape[0] #样本数量

# 线性模型参数正态分布初始化

w = np.random.normal(size=(X\_data.shape[1]))

b = np.random.normal(size=1)

losss = []

#迭代次maxCycles次

for n in range(maxCycles):

grad\_w = np.zeros(X\_data.shape[1])

grad\_b = np.zeros(1)

loss = 0

for i in range(num):

y = np.dot( X\_data[i].data, w ) + b

loss += np.power((y - y\_data[i]),2) / ( 2 \* num)

grad\_w += ( y - y\_data[i] ) \* X\_data[i].data / num

grad\_b += ( y - y\_data[i] ) / num

#更新模型参数

w -= alpha \* grad\_w

b -= alpha \* grad\_b

losss.append(loss)

print("loss = %f" % loss)

return losss

def plotLossPerTime(n,losss\_train,losss\_validation):

plt.xlabel('iteration times')

plt.ylabel('loss of train or validation')

plt.title('linear regression & gradient decrease')

n\_cycles = range(1,n+1)

plt.plot(n\_cycles, losss\_train, label = "Loss of Train", color='blue', linewidth=3)

plt.plot(n\_cycles, losss\_validation, label = "Loss of Validation", color='red', linewidth=3)

plt.legend(loc=0)

plt.grid()

plt.show()

# main

X\_train, X\_validation, y\_train, y\_validation = loadDataSet()

alpha = 0.1

maxCycles = 500

losss\_train = gradDescent(alpha,maxCycles,X\_train,y\_train)

losss\_validation = gradDescent(alpha,maxCycles,X\_validation,y\_validation)

plotLossPerTime(maxCycles,losss\_train,losss\_validation)

**（2）线性分类和梯度下降：**

import numpy as np

from matplotlib import pyplot as plt

from sklearn.datasets import load\_svmlight\_file

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

def loadDataSet():

# 读取数据

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

#将稀疏矩阵转化为完整特征矩阵

X = X.todense()

# 将数据集切分为训练集和验证集

X\_train, X\_validation, y\_train, y\_validation = train\_test\_split(X, y, random\_state=0)

print(X\_train.shape,y\_train.shape)

return X\_train, X\_validation, y\_train, y\_validation

def gradDescent(alpha,maxCycles,X\_data,y\_data):

num = y\_data.shape[0] #样本数量

# 线性模型参数正态分布初始化

w = np.random.normal(size=(X\_data.shape[1]))

b = np.random.normal(size=1)

losss = []

tv = 0

#迭代次maxCycles次

for n in range(maxCycles):

grad\_w = np.ones(X\_data.shape[1])\*(np.linalg.norm(w,ord=2))

grad\_b = np.zeros(1)

loss = 0

error = 0

C = 1/np.power(2,2)

for i in range(num):

y = np.dot( X\_data[i][0].getA()[0], w ) + b

if y\_data[i] \* y < 1:

loss += C \* max(0,1 - y\_data[i] \* y)

grad\_w += - C \* y\_data[i] \* X\_data[i][0].getA()[0]

grad\_b += - C \* y\_data[i]

if y > tv :

y = 1

else: y = -1

if not y == y\_data[i]:

error += 1

#更新模型参数

w -= alpha \* grad\_w

b -= alpha \* grad\_b

losss.append(loss)

print("loss = %f" % loss)

print("accuracy = %f" % (1-error/num))

return losss

def plotLossPerTime(n,losss\_train,losss\_validation):

plt.xlabel('iteration times')

plt.ylabel('loss of train or validation')

plt.title('linear classification & gradient decrease')

n\_cycles = range(1,n+1)

plt.plot(n\_cycles, losss\_train, label = "Loss of Train", color='blue', linewidth=3)

plt.plot(n\_cycles, losss\_validation, label = "Loss of Validation", color='red', linewidth=3)

plt.legend(loc=0)

plt.grid()

plt.show()

# main

X\_train, X\_validation, y\_train, y\_validation = loadDataSet()

alpha = 0.001

maxCycles = 500

losss\_train = gradDescent(alpha,maxCycles,X\_train,y\_train)

losss\_validation = gradDescent(alpha,maxCycles,X\_validation,y\_validation)

plotLossPerTime(maxCycles,losss\_train,losss\_validation)

## 8. 选择的评估方法（留出法，交叉验证，k折交叉验证等）：

略

## 9. 模型参数的初始化方法：

**（1）线性回归和梯度下降：**采用正态分布初始化的方法；

**（2）线性分类和梯度下降：**采用正态分布初始化的方法。

## 10.选择的loss函数及其导数：

**（1）线性回归和梯度下降：**

**（2）线性分类和梯度下降：**

## 11.实验结果和曲线图:

**（1）线性回归和梯度下降：**

①超参数选择（*η*,epoch等）：*η*= 10-4~10-1，epoch = 500

②评估结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *η* | 0.0001 | 0.001 | 0.01 | 0.1 |
| 迭代500次后训练集的Loss | 210.62313686 | 35.43473495 | 19.75599154 | 9.89725578 |
| 迭代500次后验证集的Loss | 247.83691398 | 39.84704739 | 14.95859773 | 12.80347517 |

③预测结果（最佳结果）：*η*= 0.1，epoch = 500

训练集：Ltrain = 9.89725578

w = [-9.38501146 1.73035793 -4.65678217 -1.86657919 2.32528401 -7.92909165 -0.19495226 9.31935626 -3.59115311 1.11421123 -0.94681163 1.88395808 -6.83012265]

b = [ 9.16193818]

验证集：Lvalidation = 12.80347517

w = [-11.2162461 2.92936964 -4.10327696 -1.44416132 4.55864349

-6.94469404 0.9743756 10.81325306 -4.34290038 1.84981603

0.10596646 1.21519481 -5.39103842]

b = [ 9.86065481]

④loss曲线图：



**（2）线性分类和梯度下降：**

①超参数选择（*η*,epoch等）：*η*= 10-5~10-2，epoch = 500, C = 2-2

②评估结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *η* | 0.00001 | 0.0001 | 0.001 | 0.01 |
| 迭代500次后训练集的Loss | 140.60690054 | 46.26093851 | 40.2406033 | 80.44288405 |
| 迭代500次后验证集的Loss | 113.56439935 | 32.12461163 | 24.30009639 | 1.72168708e+11 |

③预测结果（最佳结果）：*η*= 0.001，epoch = 500, C = 2-2

训练集：Ltrain = 40.2406033

w = [ 0.00267332 -0.03194497 -0.09535773 0.01926535 0.14501468 -0.00407475 -0.02458462 1.06804941 0.06131693 0.06545022 0.00538985 0.05737022 -0.70777263 -0.27955857]

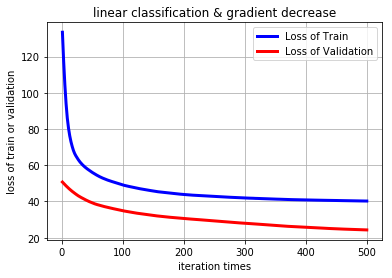
b = [ -0.94068717]

验证集：Lvalidation = 24.30009639

w = [ -2.79659157e-02 -1.75349270e-01 3.12654128e-01 -8.87355969e-02 1.64767433e-02 -1.78665928e-04 -2.20175259e-01 1.16716317e+00 6.97914092e-02 -6.96222560e-01 -7.92853565e-02 -2.67960928e-01 4.43287190e-01 8.57450085e-02]

b = [ -0.29833218]

④loss曲线图：



## 12.实验结果分析：

我在评估结果的时候进行了调超参数，通过改变学习率、惩罚参数C以及不同的迭代次数分别观察训练集和验证集Loss值的变化。

调整学习率是梯度下降算法的关键。学习率其实就是走的步长，太大的学习率可能会导致一步跨的太大，直接跨过了我们想要的最小均值；太小的学习率又会造成跨的步子太小，可能你走了好多步，其实离我们要目标点还有很大的距离。

在训练过程中，学习率的选择是不断变化的，在刚开始的时候，参数是相对随机的，所以我们应该选择一个相对较大的学习速率，这样损失就会降低得更快。当训练一段时间时，参数更新的振幅较小，所以学习率一般会衰减，衰减也很多种，例如一定数量的步骤乘以0.1的学习速率或是指数衰减。

在SVM模型的惩罚参数C的调整中，我采用了在2-5~25的范围内取值并且分别测试，最后的到在C = 2-2时收敛效果最好。这里的惩罚参数C表示在线性不可分的情况下，对分类错误的惩罚程度。C值越大，分类器就越不愿意允许分类错误（“离群点”）。如果C值太大，分类器就会竭尽全力地在训练数据上少犯错误，而实际上这是不可能/没有意义的，于是就造成过拟合。如果C值过小时，分类器就会过于“不在乎”分类错误，于是分类性能就会较差。

## 13.对比线性回归和线性分类的异同点：

**相同点：**线性回归和线性分类作为机器学习中最基本的两类问题，解决这两类问题都有以下三个基本步骤：

（1）选取一个合理的模型（线性的或非线性的），定义目标函数

（2）选择一个合适的损失函数，评估损失

（3）采用梯度下降法求出最好的模型参数，取梯度G的负方向更新模型参数。

**区别：**

对于回归问题：

用周志华的《机器学习》一书里面的例子来讲，如果想要预测的是连续值，例如西瓜的成熟度0.95 ,0.37,此类学习任务称为"回归"。其输出空间B是一个度量空间，即所谓“定量”。也就是说，回归问题的输出空间定义了一个度量去衡量输出值与真实值之间的“误差大小”。如果我们预测一个西瓜的成熟度为0.95时误差为1，预测其为0.37时误差为2。那么这两个预测结果是不一样的，是有度量定义来衡量这种“不一样”的。

对于分类问题：

如果我们想要预测的是离散值，例如"好瓜""坏瓜"，此类学习任务称为 "分类"。其输出空间B不是度量空间，即所谓“定性”。也就是说，在分类问题中，只有分类“正确”与“错误”之分，至于错误时是将Class 5分到Class 6,还是Class 7，并没有区别，都是在error的计数上加1。

## 14.实验总结：

通过这次实验，我深入理解了线性回归和梯度下降的原理，并且在小规模数据集上进行实践，同时也体会优化和调参的过程，深入认识到线性回归问题和线性分类问题的异同点，通过实践得到的收获会比在课本里理论知识的学习得到的收获更多。