

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  潘文杰

**学 号 201530612590**

**邮 箱 3213359017@qq.com**

**指导教师**  吴庆耀

**提交日期** **2017年 12 月 2 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:潘文杰

## 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。

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。

6)取梯度的负方向，记为。

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，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。

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，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Ltrain和Lvalidation随迭代次数的变化图。

## 7. 代码内容:

#### 线性回归和梯度下降

# 实验：线性回归和梯度下降

## 1. 读取实验数据，划分训练集合验证集

import sklearn

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

import jupyter

from scipy.sparse import csr\_matrix

mem = Memory("./mycache")

@mem.cache

def get\_data():

data = load\_svmlight\_file("./data/housing\_scale")

return data[0], data[1]

X, y = get\_data()

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

## 2.初始化模型参数

m, n = np.shape(X\_train)

# 迭代次数

iterations = 1000

# 学习率

alpha = 0.001

# 参数W

W = np.zeros((n, 1))

## 3.选择Loss函数

# 计算损失

def getLoss(X, y, W):

m = y.shape[0]

Loss = (np.sum((X.dot(W) - y) \*\* 2)) / (2 \* m)

return Loss

## 4.训练

# 存储历史Loss

loss\_trains = np.zeros((iterations, 1))

loss\_validations = np.zeros((iterations, 1))

# 转成numpy矩阵

# X\_train = X\_train.toarray()

# X\_valid = X\_valid.toarray()

y\_train = y\_train.reshape(-1,1)

y\_valid = y\_valid.reshape(-1,1)

for iter in range(iterations):

G = (X\_train.T.dot(X\_train.dot(W) - y\_train))/m

W = W - alpha \* G

loss\_trains[iter] = getLoss(X\_train, y\_train, W)

loss\_validations[iter] = getLoss(X\_valid, y\_valid, W)

for j in loss\_trains:

print("Train loss:%d" % j[0])

for j in loss\_validations:

print("Validation loss:%d" % j[0])

print(W)

## 5.绘制Loss train和Loss validation随迭代次数的变化图

import matplotlib.pyplot as plt

table\_x = range(1,iterations+1)

table\_y1 = loss\_trains

table\_y2 = loss\_validations

plt.title('Loss(train) vs Loss(iterations)')

plt.xlabel('iterations')

plt.ylabel('Loss')

plt.plot(table\_x, table\_y1,'r', label='train')

plt.plot(table\_x, table\_y2,'b',label='validation')

plt.legend(bbox\_to\_anchor=[1, 1])

plt.grid()

plt.show()

#### 线性分类和梯度下降

# 实验：线性分类和梯度下降

## 1. 读取实验数据，划分训练集合验证集

import sklearn

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

import jupyter

import matplotlib

from scipy.sparse import csr\_matrix

mem = Memory("./mycache")

@mem.cache

def get\_data():

data = load\_svmlight\_file("./data/australian\_scale")

return data[0], data[1]

X, y = get\_data()

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

## 2.初始化模型参数

m, n = np.shape(X\_train)

# 迭代次数

iterations = 1000

# 学习率

alpha = 0.001

# 参数W

W = np.zeros((n, 1))

C = 0.01

## 3.选择Loss函数

# SVM

def getLoss(X, y, W):

return 1/2 \* np.sum(np.square(W[:-1])) + C \* np.sum(np.maximum(0, 1 - np.multiply(y, X.dot(W))))

## 4.训练

# 存储历史Loss

loss\_trains = np.zeros((iterations, 1))

loss\_validations = np.zeros((iterations, 1))

# X\_train = X\_train.toarray()

# X\_valid = X\_valid.toarray()

y\_train = y\_train.reshape(-1,1)

y\_valid = y\_valid.reshape(-1,1)

# for iter in range(iterations):

# G = (X\_train.T.dot(X\_train.dot(W) - y\_train))/m

# W = W - alpha \* G

# loss\_trains[iter] = getLoss(X\_train, y\_train, W)

# loss\_validations[iter] = getLoss(X\_valid, y\_valid, W)

for iter in range(iterations):

margin = 1 - np.multiply(y\_train, X\_train.dot(W))

minus\_y = -y\_train

minus\_y[margin<0] = 0

result = W + C \* X\_train.T.dot(minus\_y)

result[-1] -= W[-1]

G = result

W = W - alpha \* G

loss\_trains[iter] = getLoss(X\_train, y\_train, W)

loss\_validations[iter] = getLoss(X\_valid, y\_valid, W)

for j in loss\_trains:

print("Train loss:%f" % j[0])

for j in loss\_validations:

print("Validation loss:%f" % j[0])

print(W)

## 5.绘制Loss train和Loss validation随迭代次数的变化图

import matplotlib.pyplot as plt

table\_x = range(1,iterations+1)

plt.title('Loss(train) vs Loss(iterations)')

plt.xlabel('iterations')

plt.ylabel('Loss')

plt.plot(table\_x, loss\_trains,'r', label='train')

plt.plot(table\_x, loss\_validations,'b',label='validation')

plt.legend(bbox\_to\_anchor=[1, 1])

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）:

#### 线性回归和梯度下降

留出法:2/3训练集，1/3做验证集。

#### 线性分类和梯度下降

留出法:2/3训练集，1/3做验证集。

## 9. 模型参数的初始化方法:

#### 线性回归和梯度下降

m, n = np.shape(X\_train)

# 迭代次数

iterations = 1000

# 学习率

alpha = 0.001

# 参数W

W = np.zeros((n, 1))

#### 线性分类和梯度下降

m, n = np.shape(X\_train)

# 迭代次数

iterations = 1000

# 学习率

alpha = 0.001

# 参数W

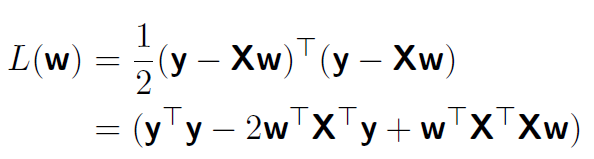
W = np.zeros((n, 1))

C = 0.01

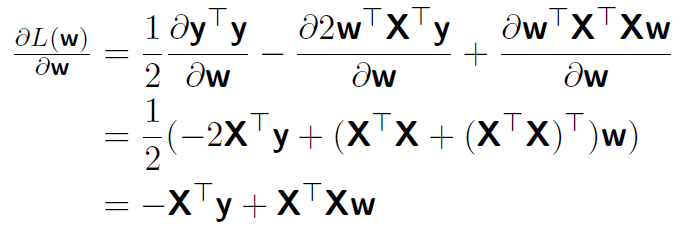
## 10.选择的loss函数及其导数:

#### 线性回归和梯度下降

Loss函数：平方损失函数

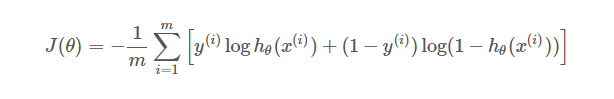


导数：

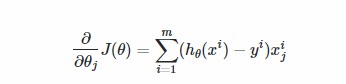


#### 线性分类和梯度下降

Loss函数：Hinge损失函数（SVM）



导数：



## 11.实验结果和曲线图:

#### 线性回归和梯度下降

1. 超参数选择（η,epoch等）：

# 迭代次数

iterations = 1000

# 学习率

alpha = 0.001

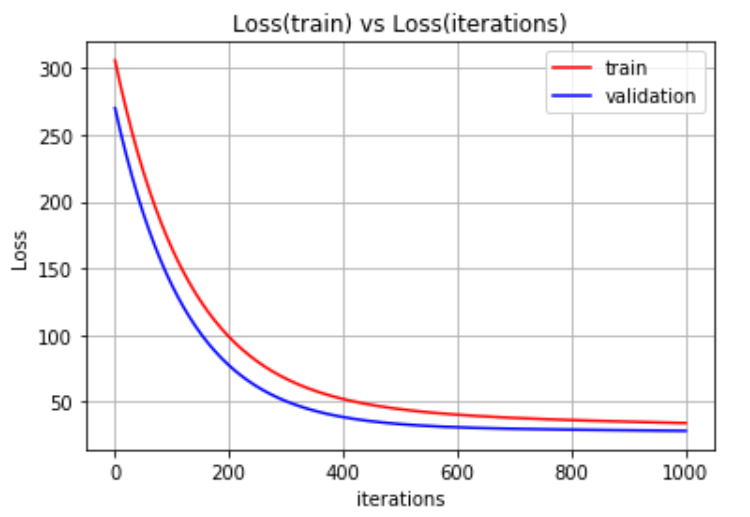
2) 评估结果（根据选择的评估方法）：

Min Loss= 28

3) 预测结果（最佳结果）：

Min Loss= 28

4) loss曲线图：



#### 线性分类和梯度下降

1) 超参数选择（η,epoch等）：

# 迭代次数

iterations = 1000

# 学习率

alpha = 0.001

C = 0.01

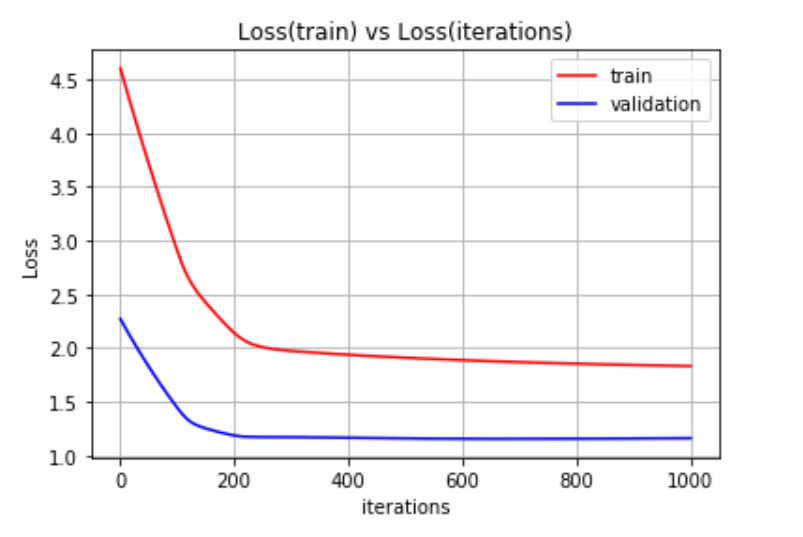
2) 评估结果（根据选择的评估方法）：

Min Loss= 1.16138075731

3) 预测结果（最佳结果）：

Min Loss= 1.16138075731

4) loss曲线图：



## 12.实验结果分析:

#### 线性回归和梯度下降

Loss曲线平滑下降，这意味着该方法具有良好拟合效果。

#### 线性分类和梯度下降

Loss曲线下滑较为平滑，具有较为良好的拟合效果，但是训练集的loss曲线和验证集的loss曲线仍存在不小的差异，应该还有不小的提升空间。

## 13.对比线性回归和线性分类的异同点：

相同：

线性回归和线性分类都可以总结成为一种流程

即

1.选择一个合适的模型

2.选合适的loss函数

3.通过各种方法（这里是梯度下降的方法）求出最好的模型参数

差异：

线性回归致力于解决连续性问题，而线性分类致力于解决离散性问题。他们的输出具有较大的差异。

## 14.实验总结：

在本次实验中，我使用python实现线性回归、线性分类和梯度下降算法，这加深了我对于线性回归和线性分类问题的理解。此外，由于两个实验放在一起进行，通过对比和分析，我对于线性回归和线性分类问题的异同点有了新的看法。