|  |
| --- |
|  |
| **实验报告** |
| **课程名称： 机器学习**  **学　　院： 电子科技大学（深圳）高等研究院**  **专　　业：电子信息**  **指导教师：张栗粽**  **学生姓名： 刘文晨**  **学　　号：202222280328** |
|  |
|  |

**电 子 科 技 大 学**

**实 验 报 告**

# 实验二

## 实验项目名称

利用逻辑回归进行鸢尾花的分类

## 二、实验学时：4学时

## 三、实验目的

1. 掌握逻辑回归的基本原理；
2. 了解TensorFlow框架。

## 实验原理

1. 逻辑回归

逻辑回归，又称对数几率回归，是一种用于解决二分类问题的机器学习方法，属于机器学习中的监督学习，用于估计某种事物的可能性。其推导过程与计算方式类似于回归的过程，但实际上主要是用来解决二分类问题（也可以解决多分类问题）。通过给定的数据（训练集）来训练模型，并在训练结束后对给定的一组或多组数据（测试集）进行分类。其中每一组数据都是由若干个指标构成。

逻辑回归与线性回归都是一种广义线性模型。逻辑回归假设因变量服从伯努利分布，而线性回归假设因变量服从高斯分布。 因此与线性回归有很多相同之处，去除Sigmoid映射函数的话，逻辑回归算法就是一个线性回归。可以说，逻辑回归是以线性回归为理论支持的，但是逻辑回归通过Sigmoid函数引入了非线性因素，因此可以轻松处理0/1分类问题。

1. TensorFlow框架

TensorFlow是由Google团队开发的深度学习框架之一，它是一个完全基于Python语言设计的开源的软件。TensorFlow的初衷是以最简单的方式实现机器学习和深度学习的概念，它结合了计算代数的优化技术，使它便计算许多数学表达式。

TensorFlow可以训练和运行深度神经网络，它能应用在许多场景下，比如，图像识别、手写数字分类、递归神经网络、单词嵌入、自然语言处理、视频检测等等。TensorFlow可以运行在多个CPU或GPU上，同时它也可以运行在移动端操作系统上（如安卓、IOS 等），它的架构灵活，具有良好的可扩展性，能够支持各种网络模型（如OSI七层和TCP/IP四层）。

TensorFlow这个词由Tensor和Flow两个词组成，这两者是TensorFlow最基础的要素。Tensor代表张量（也就是数据），它的表现形式是一个多维数组；而Flow意味着流动，代表着计算与映射，它用于定义操作中的数据流。

## 实验内容与要求

1. 数据预处理；
2. 建立逻辑回归模型；
3. 通过极大似然方法求解；
4. 采用负对数似然函数，构建损失函数；
5. 训练模型，使用梯度下降法最小化损失；
6. 迭代计算损失，测试集的准确率；
7. 绘制图像。

## 实验器材（设备、元器件）

处理器：Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz

Python 3.9.0

matplotlib 3.4.0

tensorflow 2.11.0rc1

xlrd 1.2.0

## 实验步骤

1. 数据预处理

给定的训练集和测试集是data文件，先把它们改为xlsx文件，再调用Python第三方库——xlrd，将当中的数据转换为列表x\_train、y\_train、x\_test、y\_test。代码如下：

1. # 数据预处理
2. # 训练集
3. file\_train\_path = 'data/iris\_train.xlsx'
4. file\_train\_xlsx = xd.open\_workbook(file\_train\_path)
5. file\_train\_sheet = file\_train\_xlsx.sheet\_by\_name('Sheet1')
6. x\_train = []
7. y\_train = []
8. **for** row **in** range(file\_train\_sheet.nrows):
9. x\_data = []
10. **for** col **in** range(file\_train\_sheet.ncols):
11. **if** col < file\_train\_sheet.ncols - 1:
12. x\_data.append(file\_train\_sheet.cell\_value(row, col))
13. **else**:
14. **if** file\_train\_sheet.cell\_value(row, col) == 'Iris-setosa':
15. y\_train.append(0)
16. **elif** file\_train\_sheet.cell\_value(row, col) == 'Iris-versicolor':
17. y\_train.append(1)
18. **else**:
19. y\_train.append(2)
20. x\_train.append(list(x\_data))
21. # 测试集
22. file\_test\_path = 'data/iris\_test.xlsx'
23. file\_test\_xlsx = xd.open\_workbook(file\_test\_path)
24. file\_test\_sheet = file\_test\_xlsx.sheet\_by\_name('Sheet1')
25. x\_test = []
26. y\_test = []
27. **for** row **in** range(file\_test\_sheet.nrows):
28. x\_data = []
29. **for** col **in** range(file\_test\_sheet.ncols):
30. **if** col < file\_test\_sheet.ncols - 1:
31. x\_data.append(file\_test\_sheet.cell\_value(row, col))
32. **else**:
33. **if** file\_test\_sheet.cell\_value(row, col) == 'Iris-setosa':
34. y\_test.append(0)
35. **elif** file\_test\_sheet.cell\_value(row, col) == 'Iris-versicolor':
36. y\_test.append(1)
37. **else**:
38. y\_test.append(2)
39. x\_test.append(list(x\_data))
40. # print(x\_train)
41. # print(y\_train)
42. # print(x\_test)
43. # print(y\_test)

将特征值的类型转换为tensor类型，避免后面的矩阵乘法报错，代码如下：

1. # 将特征值的类型转换为tensor类型，避免后面的矩阵乘法报错
2. x\_train = tf.cast(x\_train, tf.float32)
3. x\_test = tf.cast(x\_test, tf.float32)
4. 训练模型

设学习率η=0. 1，将特征值和目标值一一配对，并且每4组数据为一个batch，喂入神经网络的数据以batch为单位，初始化梯度和偏置，设梯度下降次数为500，每轮分4个step，loss\_all记录四个step生成的4个loss的和，在每次迭代过程中，对每个梯度和偏置求偏导，计算损失和测试集的准确率，并打印。代码如下：

1. # 将特征值和目标值一一配对 并且每4组数据为一个batch，喂入神经网络的数据以batch为单位
2. train\_data = tf.data.Dataset.from\_tensor\_slices((x\_train, y\_train)).batch(4)
3. test\_data = tf.data.Dataset.from\_tensor\_slices((x\_test, y\_test)).batch(4)
4. # print(test\_data)
6. # 初始化梯度和偏置
7. # 由于输入的特征为4个，目标值是三分类，所以我们将 梯度 随机初始化为 四行三列的tensor
8. w = tf.Variable(tf.random.truncated\_normal([4, 3], stddev=0.1))
9. # 同理，我们的目标值是一维数据，所以将 偏置 初始化为随机的1维tensor
10. b = tf.Variable(tf.random.truncated\_normal([3], stddev=0.1))
12. # 设置学习率
13. learn\_rate = 0.1
14. # 梯度下降次数
15. epoch = 500
16. # 每轮分4个step，loss\_all记录四个step生成的4个loss的和
17. loss\_all = 0
18. # 绘图相关数据
19. train\_loss = []
20. test\_acc = []
22. **for** epoch **in** range(epoch):
23. **for** step, (x\_train, y\_train) **in** enumerate(train\_data):
24. with tf.GradientTape() as tape:
25. y = tf.matmul(x\_train, w) + b
26. # 使输出符合概率分布
27. y = tf.nn.softmax(y)
28. # 将目标值转换为独热编码，方便计算loss和acc
29. y\_true = tf.one\_hot(y\_train, depth=3)
30. # 回归性能评估采用MSE
31. loss = tf.reduce\_mean(tf.square(y\_true - y))
32. # print(loss)
33. loss\_all += loss.numpy()
34. # 对每个梯度和偏置求偏导
35. grads = tape.gradient(loss, [w, b])
36. # 梯度自更新，这两行代码相当于:
37. # w = w - lr \* w\_grads
38. # b = b - lr \* b\_grads
39. w.assign\_sub(learn\_rate \* grads[0])
40. b.assign\_sub(learn\_rate \* grads[1])
41. **print**(f"第{epoch}轮,损失是:{loss\_all / 4}")
42. train\_loss.append(loss\_all / 4)
43. loss\_all = 0  # loss\_all归零，为记录下一个epoch的loss做准备
45. # total\_correct为预测对的样本个数, total\_number为测试的总样本数，将这两个变量都初始化为0
46. total\_correct, total\_number = 0, 0
47. **for** x\_test, y\_test **in** test\_data:
48. y = tf.matmul(x\_test, w) + b
49. y = tf.nn.softmax(y)
50. # 返回最大值所在的索引，即预测的分类
51. y\_pred = tf.argmax(y, axis=1)
52. # print(y\_pred)
53. y\_pred = tf.cast(y\_pred, dtype=y\_test.dtype)
54. # 预测正确为1，错误为0
55. correct = tf.cast(tf.equal(y\_pred, y\_test), dtype=tf.int32)
56. correct = tf.reduce\_sum(correct)
57. # 将所有batch中的correct数加起来
58. total\_correct += int(correct)
59. # total\_number为测试的总样本数，也就是x\_test的行数，shape[0]返回变量的行数
60. total\_number += x\_test.shape[0]
61. accuracy\_rate = total\_correct / total\_number
62. test\_acc.append(accuracy\_rate)
63. **print**("测试集的准确率为:", accuracy\_rate)
64. **print**("---------------------------------")
65. 绘图

使用Python第三方库——matplotlib中的plot()进行绘图。第一个图绘制训练过程中损失随梯度下降次数增加的曲线，第二个图绘制测试过程中测试集的准确率随梯度下降次数增加的折线。代码如下：

1. # 绘制图像
2. plt.figure(figsize=(6, 8))
3. plt.xlabel('epoch')
4. plt.ylabel('train\_loss')
5. plt.plot(train\_loss, marker='.', color='r', linestyle='--', label="loss")
6. plt.legend(loc="best")
7. plt.show()
8. plt.figure(figsize=(10, 8))
9. plt.xlabel('epoch')
10. plt.ylabel('test\_acc')
11. plt.plot(test\_acc, marker='.', color='r', linestyle='--', label="accuracy")
12. plt.legend(loc="best")
13. plt.show()
14. 实验结果

编写好代码后，运行run.py。运行的部分结果如图1所示。



图1 程序运行部分结果

训练过程中损失随梯度下降次数增加的曲线如图2所示。其中，每一个点的横坐标是训练趟数，纵坐标是该趟时的训练损失。

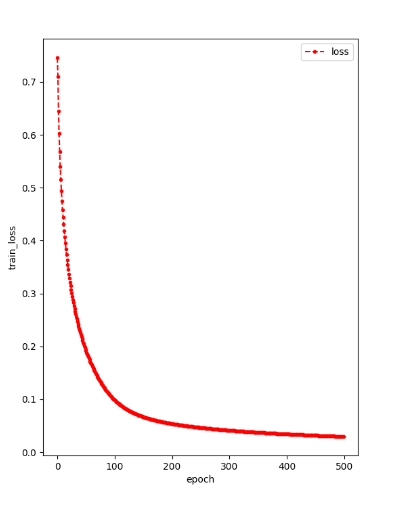


图2 训练损失

测试过程中测试集的准确率随梯度下降次数增加的折线如图3所示。其中，每一个点的横坐标是训练趟数，纵坐标是该趟时的测试集的准确率。

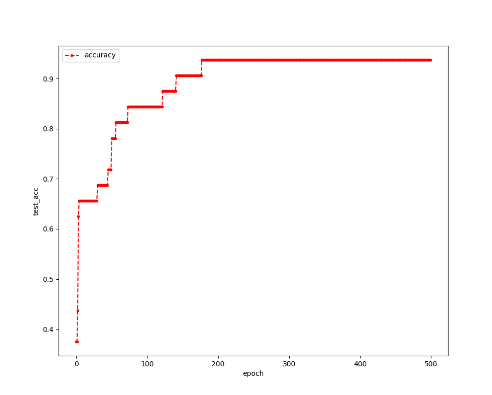


图3 测试集准确率

在本次实验中，预设的学习率为0.1。从图2中我们可以看出，训练损失随迭代趟数的增加而降低；而从图2中我们可以看出，测试集的准确率随迭代趟数的增加而增加。从图1中我们得知，在第500次迭代后，损失为0.029640378761541797，测试集的准确率为0.9375，超过预期的70%，实验成功。

## 心得体会

本实验实现了利用逻辑回归进行鸢尾花的分类，使用了TensorFlow框架，测试集的准确率为0.9375，达到预期目标。通过此次实验，很好地掌握了逻辑回归的原理，熟悉了Python第三方库——tensorflow、matplotlib和xlrd的使用。在使用xlrd进行数据预处理时遇到了问题，通过查阅资料得以解决。解决过程见于学生博客：[关于xlrd.biffh.XLRDError: Excel xlsx file； not supported的解决方法](https://blog.csdn.net/ProgramNovice/article/details/127578222)。