电子科技大学

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

一、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；

四、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。

2. 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。

五、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。

六、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

七、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：

```
    # 数据预处理
    # 训练集
    file_train_path = 'data/iris_train.xlsx'
    file_train_xlsx = xd.open_workbook(file_train_path)
    file_train_sheet = file_train_xlsx.sheet_by_name('Sheet1')
    x_train = []
    y_train = []
    for row in range(file_train_sheet.nrows):
        x_data = []
        for col in range(file_train_sheet.ncols):
            if col < file_train_sheet.ncols - 1:
                x_data.append(file_train_sheet.cell_value(row, col))
            else:
                if file_train_sheet.cell_value(row, col) == 'Iris-setosa':
                    y_train.append(0)
                elif file_train_sheet.cell_value(row, col) == 'Iris-versicolor':
                    y_train.append(1)
                else:
                    y_train.append(2)
        x_train.append(list(x_data))
    # 测试集
    file_test_path = 'data/iris_test.xlsx'
    file_test_xlsx = xd.open_workbook(file_test_path)
    file_test_sheet = file_test_xlsx.sheet_by_name('Sheet1')
    x_test = []
    y_test = []
    for row in range(file_test_sheet.nrows):
        x_data = []
        for col in range(file_test_sheet.ncols):
            if col < file_test_sheet.ncols - 1:
                x_data.append(file_test_sheet.cell_value(row, col))
            else:
                if file_test_sheet.cell_value(row, col) == 'Iris-setosa':
                    y_test.append(0)
                elif file_test_sheet.cell_value(row, col) == 'Iris-versicolor':
                    y_test.append(1)
                else:
                    y_test.append(2)
```

```
        x_test.append(list(x_data))
    # print(x_train)
    # print(y_train)
    # print(x_test)
    # 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：

```
    # 将特征值的类型转换为 tensor 类型，避免后面的矩阵乘法报错
    x_train = tf.cast(x_train, tf.float32)
    x_test = tf.cast(x_test, tf.float32)
```

2. 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：

```
    # 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
    train_data = tf.data.Dataset.from_tensor_slices((x_train, y_train)).batch(4)
    test_data = tf.data.Dataset.from_tensor_slices((x_test, y_test)).batch(4)
    # print(test_data)

    # 初始化梯度和偏置
    # 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
    w = tf.Variable(tf.random.truncated_normal([4, 3], stddev=0.1))
    # 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
    b = tf.Variable(tf.random.truncated_normal([3], stddev=0.1))

    # 设置学习率
    learn_rate = 0.1
    # 梯度下降次数
    epoch = 500
    # 每轮分 4 个 step，loss_all 记录四个 step 生成的 4 个 loss 的和
    loss_all = 0
    # 绘图相关数据
    train_loss = []
    test_acc = []

    for epoch in range(epoch):
        for step, (x_train, y_train) in enumerate(train_data):
            with tf.GradientTape() as tape:
                y = tf.matmul(x_train, w) + b
                # 使输出符合概率分布
                y = tf.nn.softmax(y)
                # 将目标值转换为独热编码，方便计算 loss 和 acc
```

```
                y_true = tf.one_hot(y_train, depth=3)
                # 回归性能评估采用 MSE
                loss = tf.reduce_mean(tf.square(y_true - y))
                # print(loss)
                loss_all += loss.numpy()
            # 对每个梯度和偏置求偏导
            grads = tape.gradient(loss, [w, b])
            # 梯度自更新，这两行代码相当于:
            # w = w - lr * w_grads
            # b = b - lr * b_grads
            w.assign_sub(learn_rate * grads[0])
            b.assign_sub(learn_rate * grads[1])
        print(f"第{epoch}轮,损失是:{loss_all / 4}")
        train_loss.append(loss_all / 4)
        loss_all = 0  # loss_all 归零，为记录下一个 epoch 的 loss 做准备

        # 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
        total_correct, total_number = 0, 0
        for x_test, y_test in test_data:
            y = tf.matmul(x_test, w) + b
            y = tf.nn.softmax(y)
            # 返回最大值所在的索引，即预测的分类
            y_pred = tf.argmax(y, axis=1)
            # print(y_pred)
            y_pred = tf.cast(y_pred, dtype=y_test.dtype)
            # 预测正确为 1，错误为 0
            correct = tf.cast(tf.equal(y_pred, y_test), dtype=tf.int32)
            correct = tf.reduce_sum(correct)
            # 将所有 batch 中的 correct 数加起来
            total_correct += int(correct)
            # 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
            total_number += x_test.shape[0]
        accuracy_rate = total_correct / total_number
        test_acc.append(accuracy_rate)
        print("测试集的准确率为:", accuracy_rate)
        print("----------------------------------")
```

3. 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：

```
    # 绘制图像
    plt.figure(figsize=(6, 8))
```

```
    plt.xlabel('epoch')
    plt.ylabel('train_loss')
    plt.plot(train_loss, marker='.', color='r', linestyle='--', label="loss")
    plt.legend(loc="best")
    plt.show()
    plt.figure(figsize=(10, 8))
    plt.xlabel('epoch')
    plt.ylabel('test_acc')
    plt.plot(test_acc, marker='.', color='r', linestyle='--', label="accuracy")
    plt.legend(loc="best")
    plt.show()
```

4. 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。

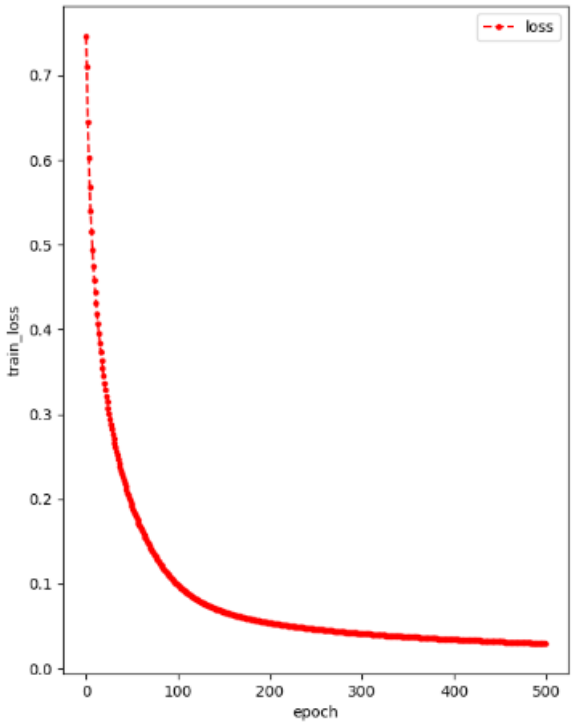

图 2 训练损失

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。

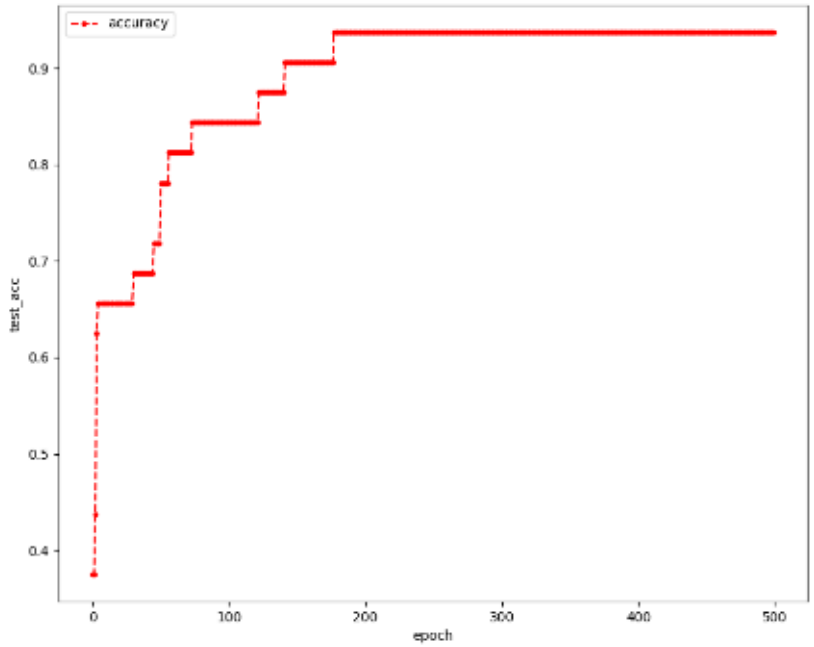

图 3 测试集准确率

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。

八、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。