

人工智能导论课程实验报告

|  |  |
| --- | --- |
| 实验名称： | 实验六  BP神经网络算法 |
| 实验日期： | 2024-5-18 |
| 实验地点： | 西部片区4号楼301 |
|  | |
| 学号： | 33920212204567 |
| 姓名： | 任宇 |
| 专业年级： | 软工2021级 |
| 学年学期： | 2023-2024学年第二学期 |

1. **实验目的**

编程实现BP神经网络算法并理解算法原理。

1. **实验内容**

将Iris（鸢尾花）数据集分为训练集（Iris-train.txt）和测试集（Iris-test.txt），分别含75个样本，每个集合中每种花各有25个样本。为了方便训练，将3类花分别编号为1，2，3 。使用这些数据训练一个4输入（分别对应4个特征）、隐含层（10个神经元）、3输出（分别对应该样本属于某一品种的可能性大小）的神经网络（4\*10\*3）。

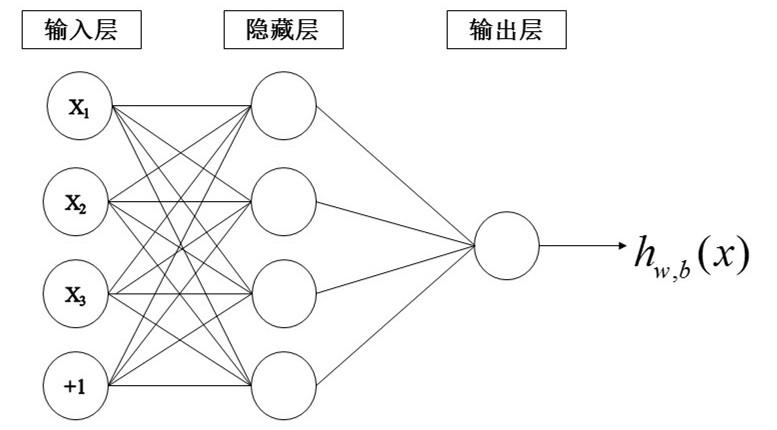
使用训练集对网络进行训练，再预测测试集中每个样本的标签，并输出预测准确率（独立运行10次，列出10次的准确率，并输出平均准确率和标准差）。

1. **实验过程**
2. BP神经网络算法的基本原理：

BP神经网络（Backpropagation Neural Network）是一种多层前馈神经网络，使用反向传播算法进行训练。这种算法在1980年代初被提出，并迅速成为训练多层神经网络的标准方法。BP神经网络广泛应用于多种机器学习任务，如分类、回归等。

BP神经网络通常包括输入层、隐藏层（一个或多个）和输出层。每一层包含多个神经元，神经元之间通过加权连接传递信息。输入层接收外部数据，输出层产生网络的预测结果。反向传播是BP神经网络核心，它的目的是通过网络的误差反向传递，有效地更新网络中的权重和偏置。从输出层开始，计算损失函数相对于每个权重的梯度，这些梯度指示了损失函数相对于每个权重的增长方向。然后利用这些梯度，通过优化算法（如梯度下降）调整权重，以最小化损失函数。权重的更新依赖于学习率，它决定了在每次迭代中更新的步长大小。

整个过程是迭代进行的，每次迭代包括一次前向传播和一次反向传播。通过多次迭代，网络权重逐渐调整，损失函数的值逐渐减小，模型的预测性能逐步提高。迭代继续进行直到达到一定的迭代次数，或者模型的性能不再显著提升。

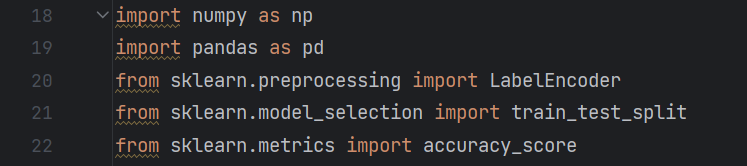


1. 使用BP神经网络算法编程解决Iris数据集分类问题：

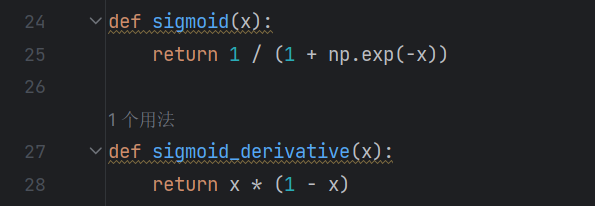
* 首先代码通过指定行号范围将Iris数据集分割成训练集和测试集，并保存到不同的文件中。



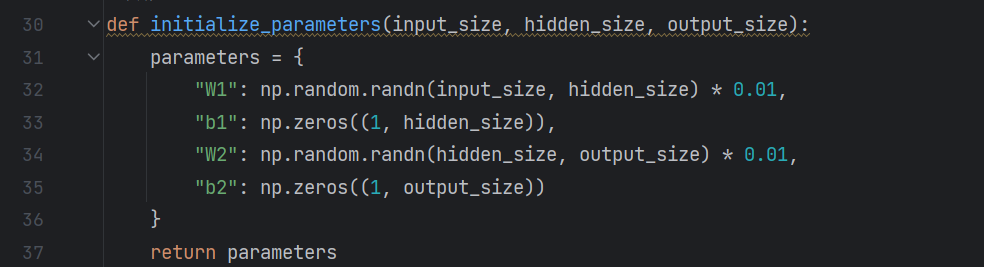
* 导入所需库，代码使用了numpy进行数学计算，pandas进行数据操作，sklearn的LabelEncoder和train\_test\_split进行数据预处理和分割，以及accuracy\_score来计算准确率。



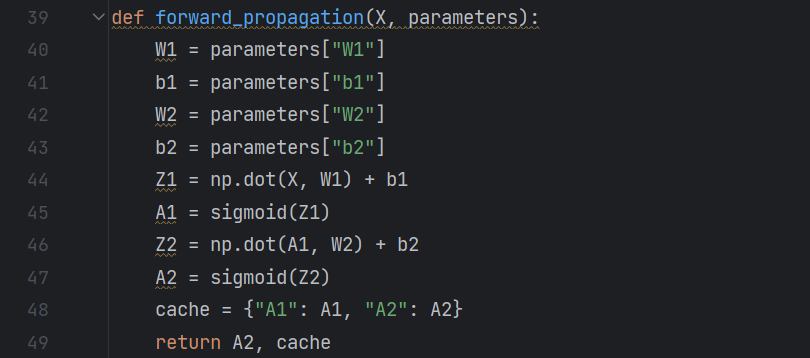
* 定义了Sigmoid激活函数及其导数。



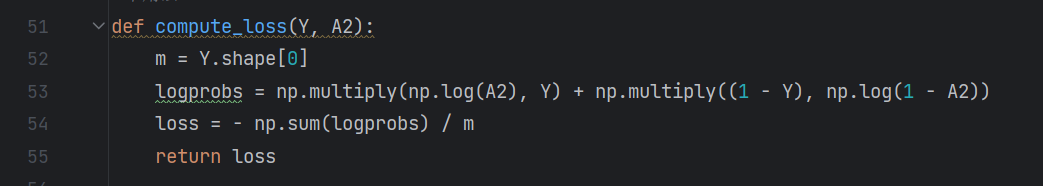
* initialize\_parameters函数用于初始化网络的权重和偏置，这些参数是后续训练过程中需要优化的。



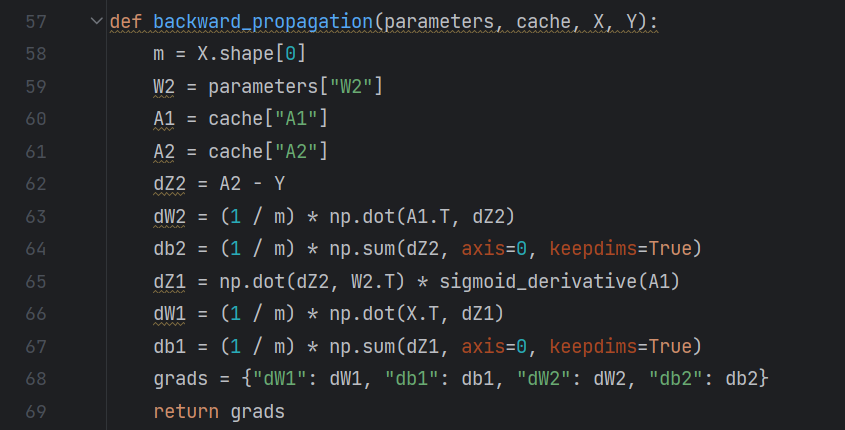
* forward\_propagation函数用于实现网络的前向传播过程，计算从输入层到输出层的信号传递，并返回最后的输出以及中间计算结果。



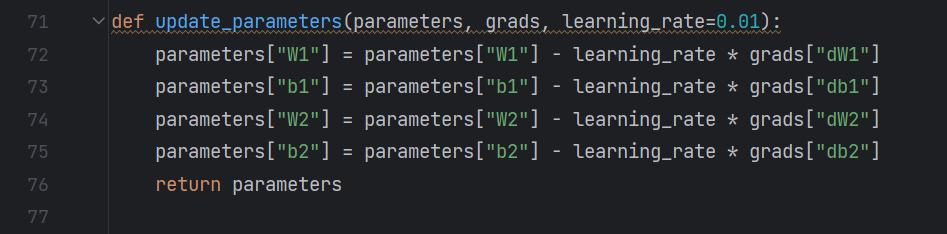
* compute\_loss函数用于计算损失，这里使用的是交叉熵损失函数，适用于分类问题。



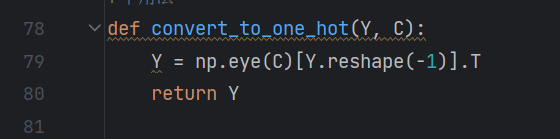
* backward\_propagation函数用于根据损失函数计算的结果，通过链式法则反向传播误差，并计算每个权重和偏置的梯度。



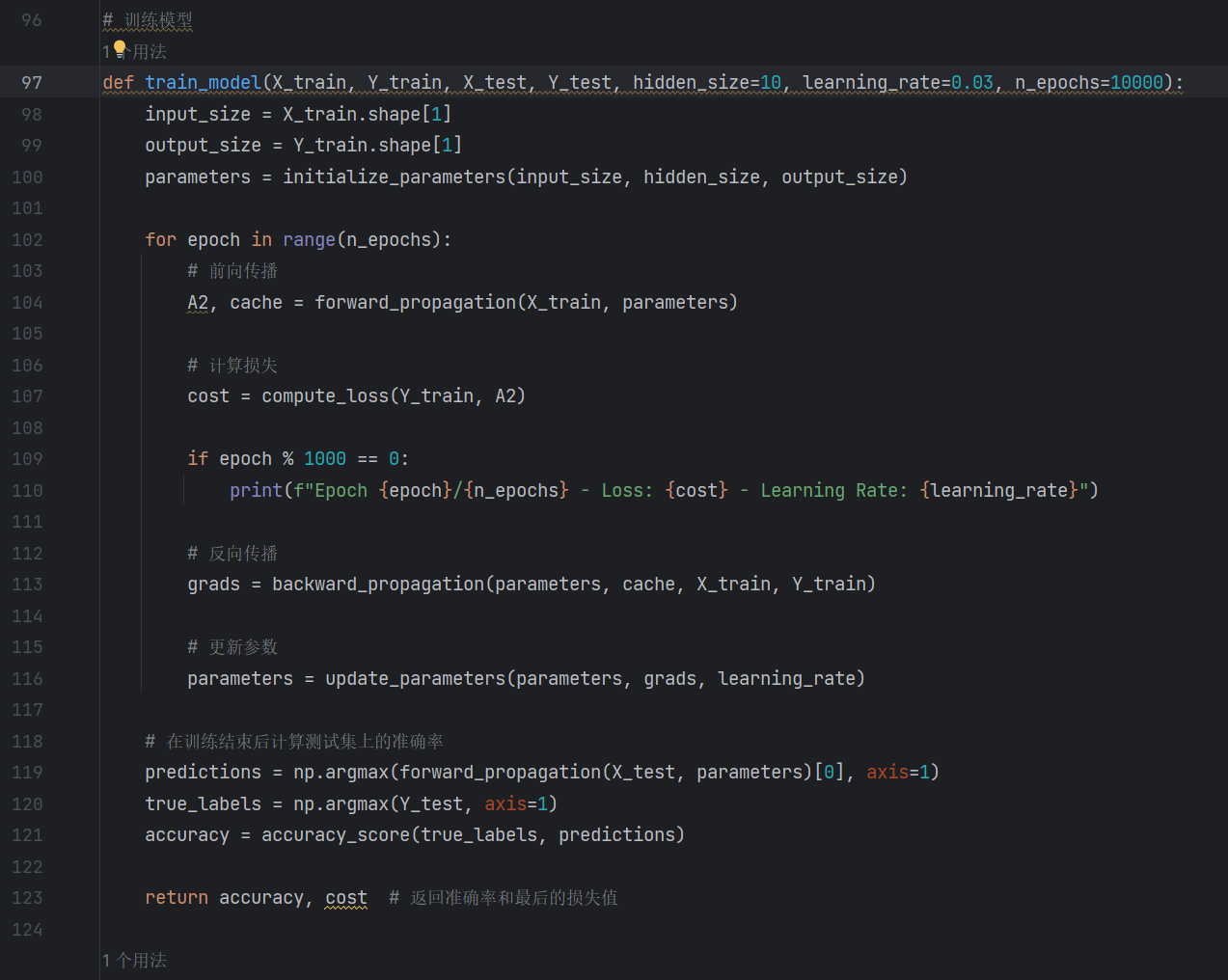
* update\_parameters函数用于根据梯度和学习率更新网络的权重和偏置。



* convert\_to\_one\_hot函数将标签转换为one-hot编码。



* train\_model函数负责模型的训练过程，包括多次迭代进行前向传播、损失计算、反向传播和参数更新，并在训练结束后在测试集上评估模型的性能。



* main函数设置实验，独立运行十次模型训练并记录准确率和损失值，最后输出平均准确率、准确率的标准差、平均损失和损失的标准差。

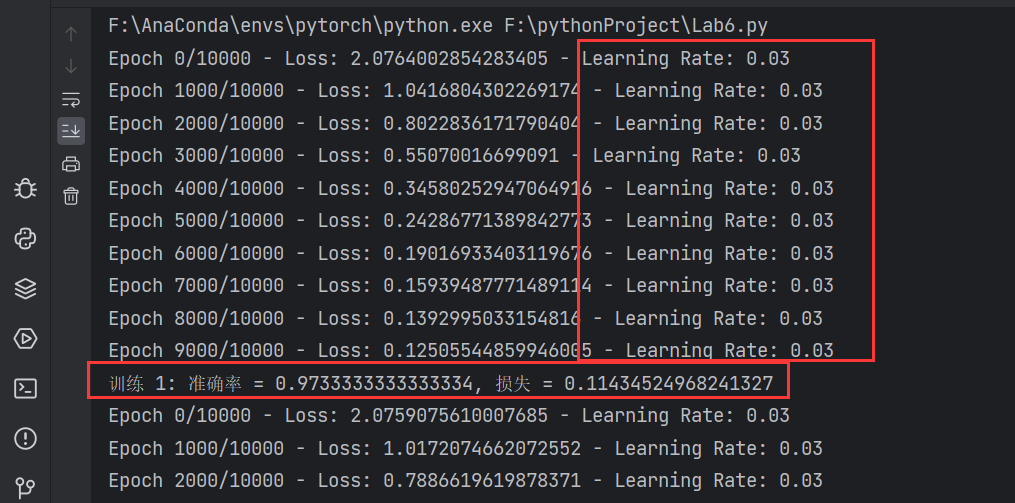
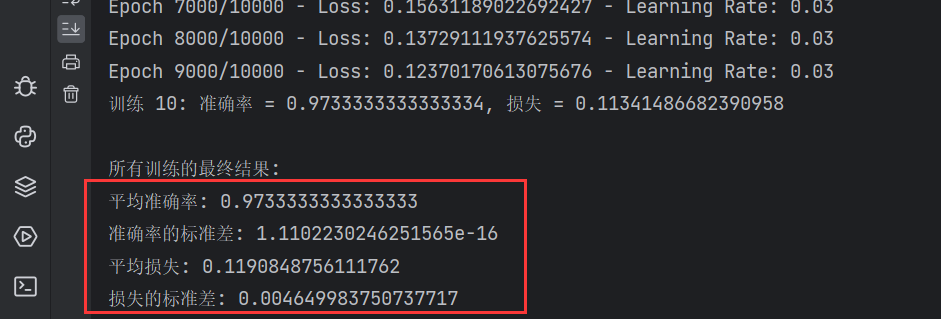


1. 运行结果：

运行的神经网络结构为4\*10\*3，学习率为0.03，10000次迭代：



运行结果如下：

1. **实验思考及心得**

完成这次实验让我更深入地理解了BP神经网络算法的原理和实现过程。通过对Iris数据集的处理和神经网络的训练，我学会了如何使用Python和相关库来构建和训练神经网络模型。在实验过程中，我注意到了调整学习率对模型训练的影响，以及如何通过多次运行来评估模型的稳定性。这次实验加深了我对神经网络的理解，也提升了我的编程能力。