**面向对象程序设计C++**

**大作业报告**

**题 目： 误差反向传播算法（BP）实现Iris数据分类**

**专 业： 网络空间安全，计算机科学与技术**

**组 别： 第五组**

**组 长： 吴凯文 21620202202459**

**组 员： 刘恩众 21620202204304**

**组 员： 周诚宇 25120202201960**

**组 员： 张嘉昌 35820202202385**

**组 员： 王欣雨 33120202203906**

**日 期：** **2022年6月9日**

**目录**

[一． 算法分析 3](#_Toc16226)

[二． 程序设计 3](#_Toc28495)

[三． 实验结果 9](#_Toc8293)

[四． 讨论分析 10](#_Toc5632)

[五． 小组分工 12](#_Toc30794)

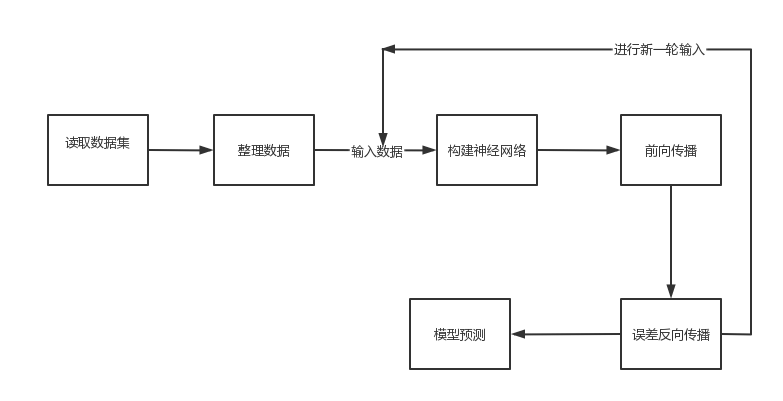
1. **算法分析**
2. 打开并读取数据集文件，将总数据集分化成训练集结构体数组train和测试集结构体数组test，并把文件内容进行整理，将样本四个属性存入结构体数组中的数组in，将样本标签化为{1，0，0}以及类似形式存入sample中的数组out，同时读取数据时遍历获得每组属性的最大值max以及最小值min，之后对每一项属性进行归一化即

x=（x-min）/（max-min）

完成数据的初始化；

1. 构建神经网络，初始化输入层，隐藏层与输出层，同时再对网络的权值W和偏置b随机初始化；
2. 神经网络读取训练集和测试集，用训练集进行训练：每一个单样本的四个属性为输入值，根据神经网络的前向传播求出输出的预测值；
3. 通过损失函数进行误差计算，根据梯度下降公式更新W和b，进行误差反向传播；
4. 所有样本遍历之后重新做新一轮的遍历和参数更新，直至迭代次数达到最大值或者误差收敛；
5. 用测试集进行前向传播得到预测值，三输出中的最大输出为样本预测标签，通过比较预测标签和正确结果标签是否相等判断是否预测正确，根据正确分类的样本以及测试集中样本数量的比值来判断模型的准确率。
6. **程序设计**

概要：对各部分代码内容模块化,函数化，简要分为数据读取，数据处理，神经网络构建，前向传播，误差反向传播，模型预测等模块，由小组成员分别完成各模块。



1. 读取数据。

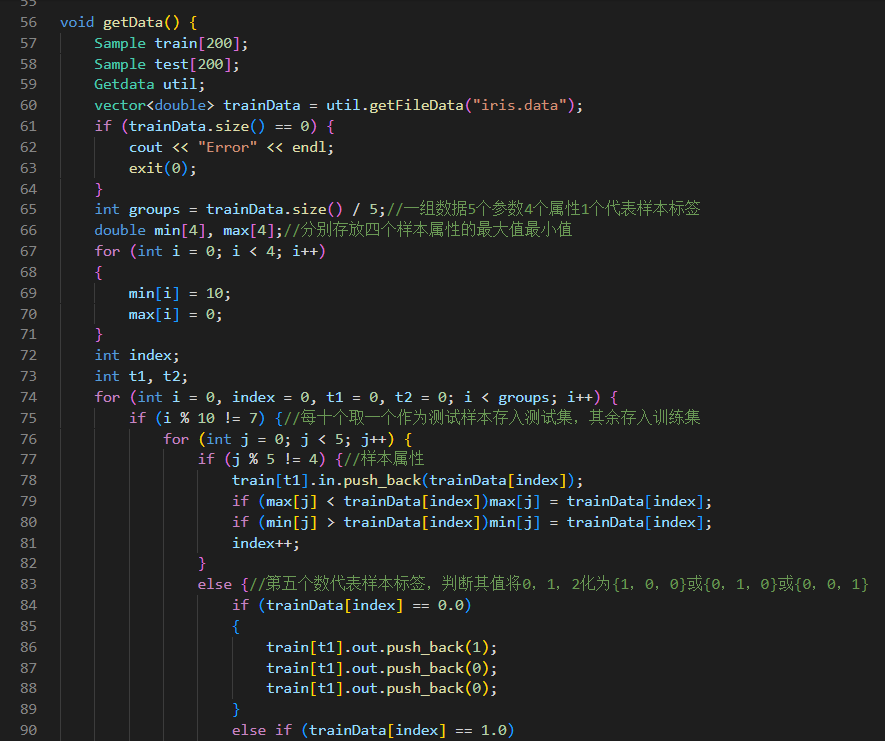
打开与读取文件数据集，简单将数据化为一维数组，

具体类别化为数字“0，1，2”。

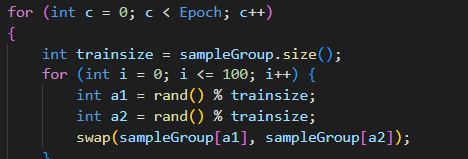


1. 数据整理。

将一维数组trandata数据进行每5个元素划分输入结构体数组train的数组in与数组out中，同时按9：1划分训练集与测试集。

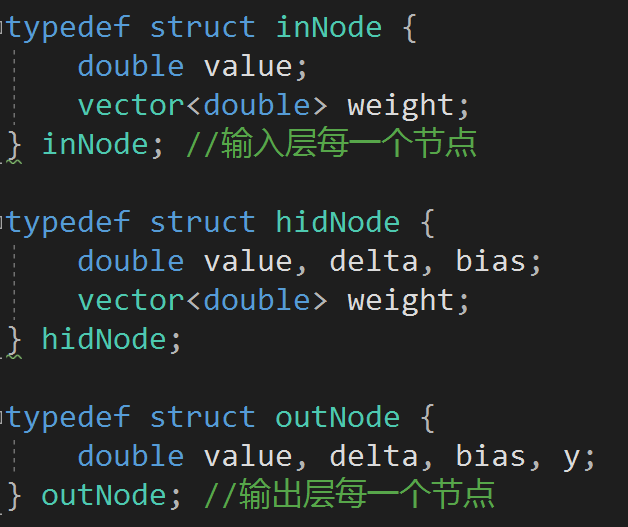


随机数选取两个样本交换进行训练集乱序。

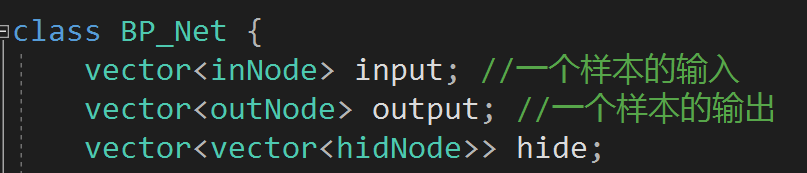


1. 初始化神经网络

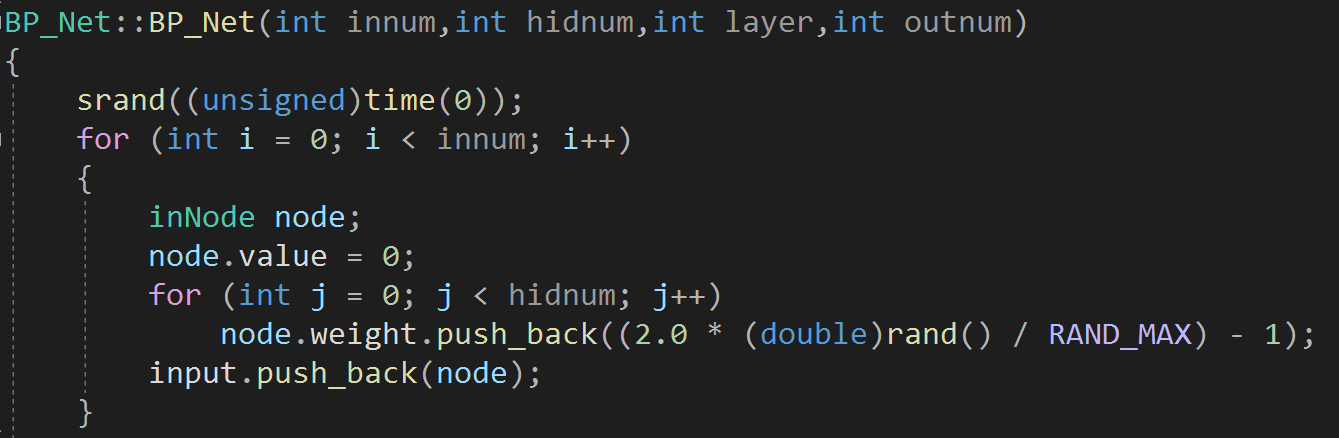
定义结构体表示各层节点，输入层节点inNode包含输出值value和对于下一层各节点的权值数组weight；隐藏层hidNode包含输出值value，残差delta，偏置bias，和对于下一层各节点的权值数组weight；输出层outNode包含输出值value，残差delta，偏置bias和该样本真实输出y。



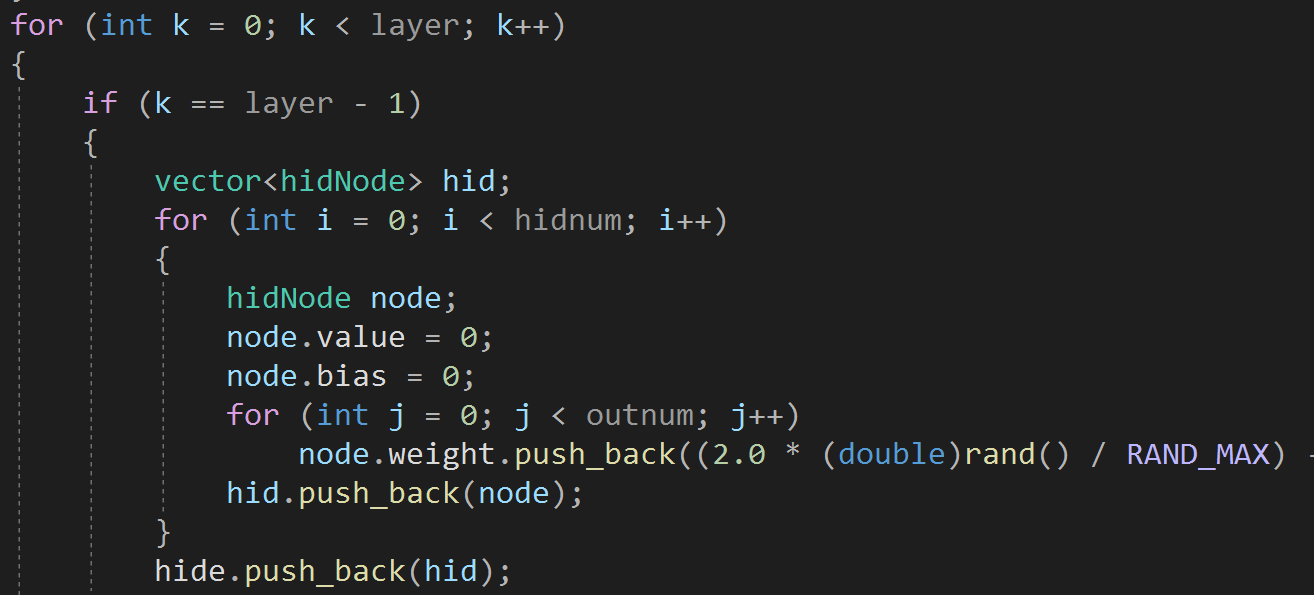
input、output、hide数组分别存放输入、输出、隐藏层各节点。



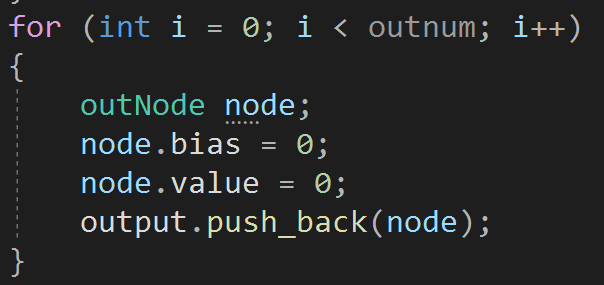
对输入层innum个节点权值数组weight随机赋值。



对layer层，每层hidnum个节点的权值数组weight随机赋值，初始化各变量。

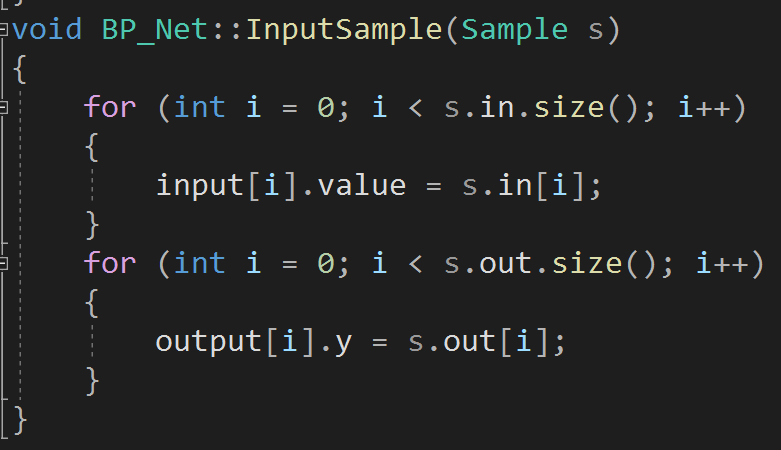


初始化输出层outnum个节点。

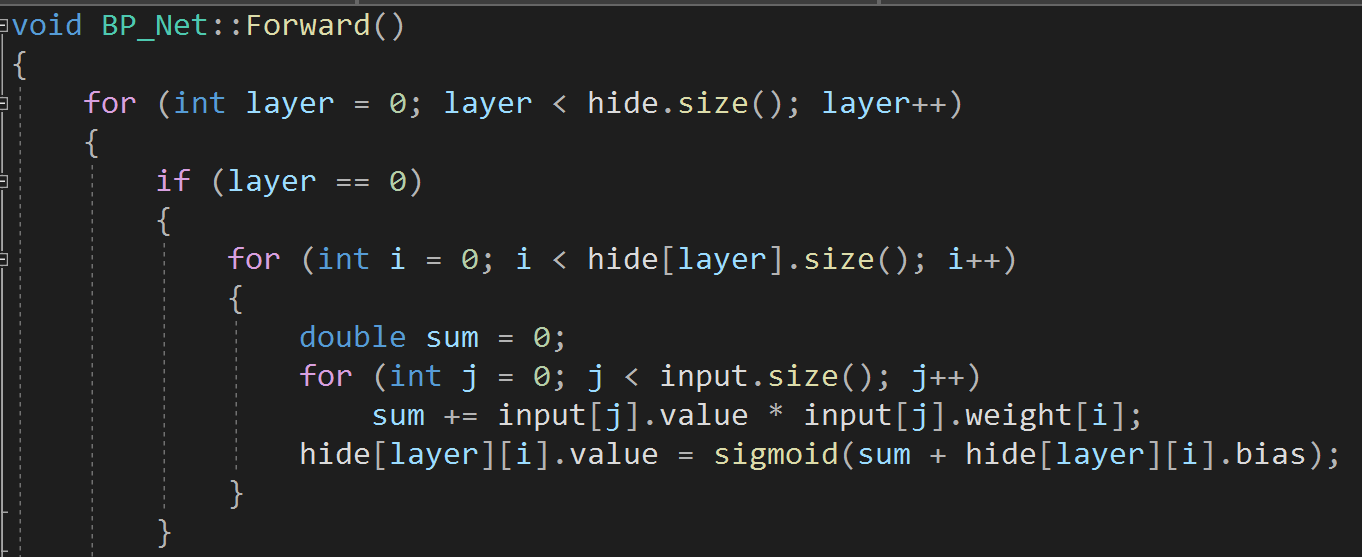


1. 数据输入，进行前向传播

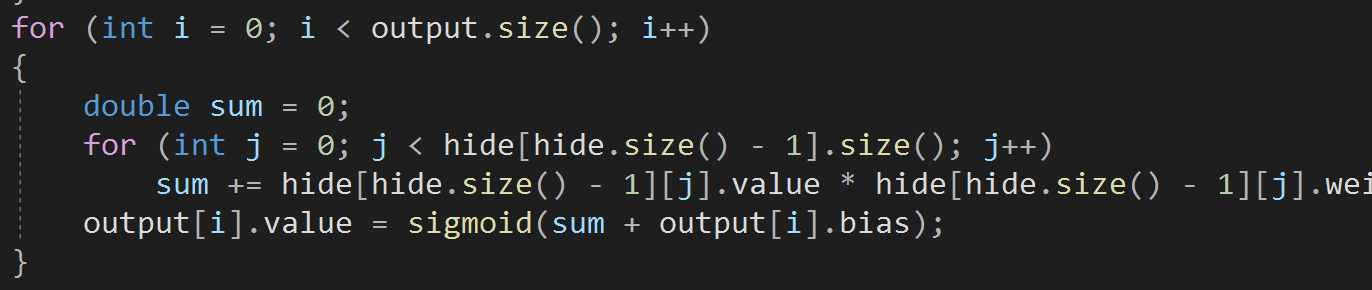
将样本赋值到输入、输出层各节点。



通过公式计算隐藏层各节点输出值value.

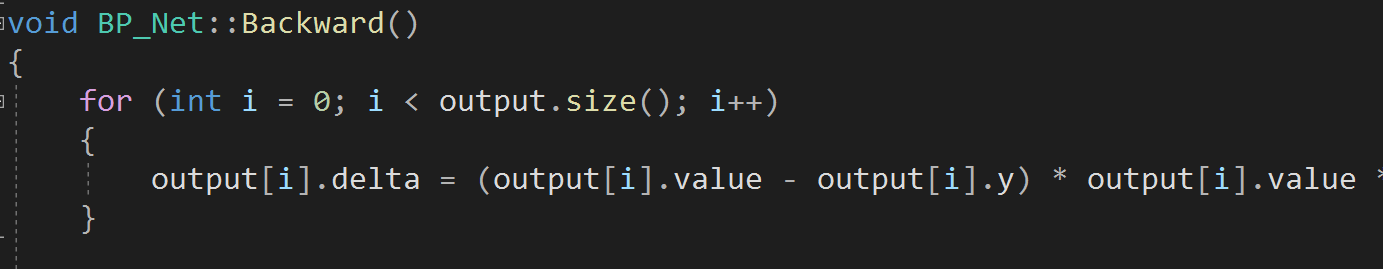


计算得到输出层各节点输出值value。

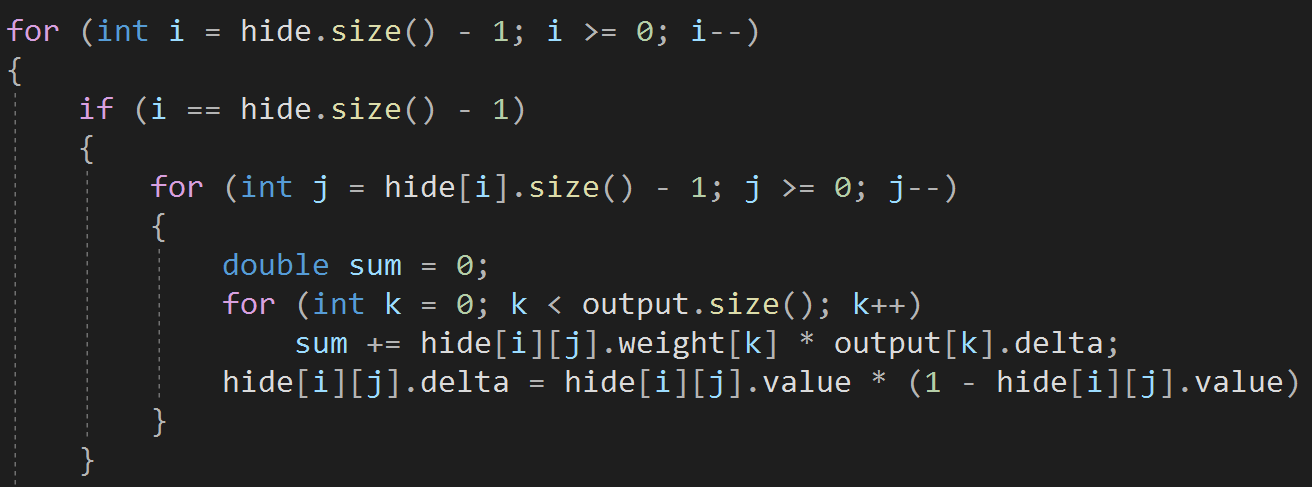


1. 进行误差反向传播，更新参数

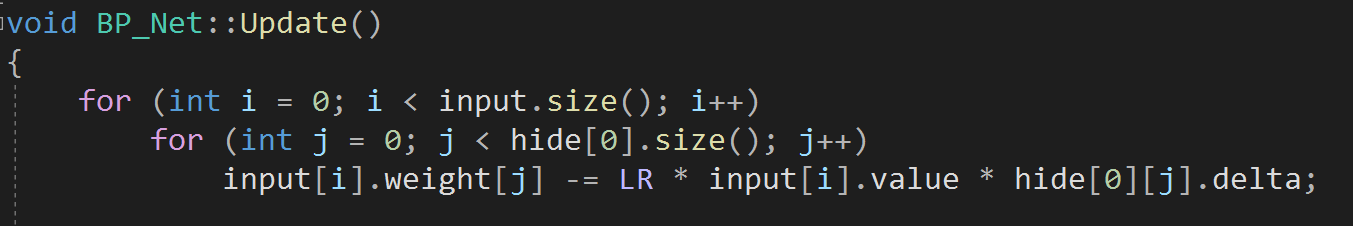
通过公式计算输出层各节点残差delta。



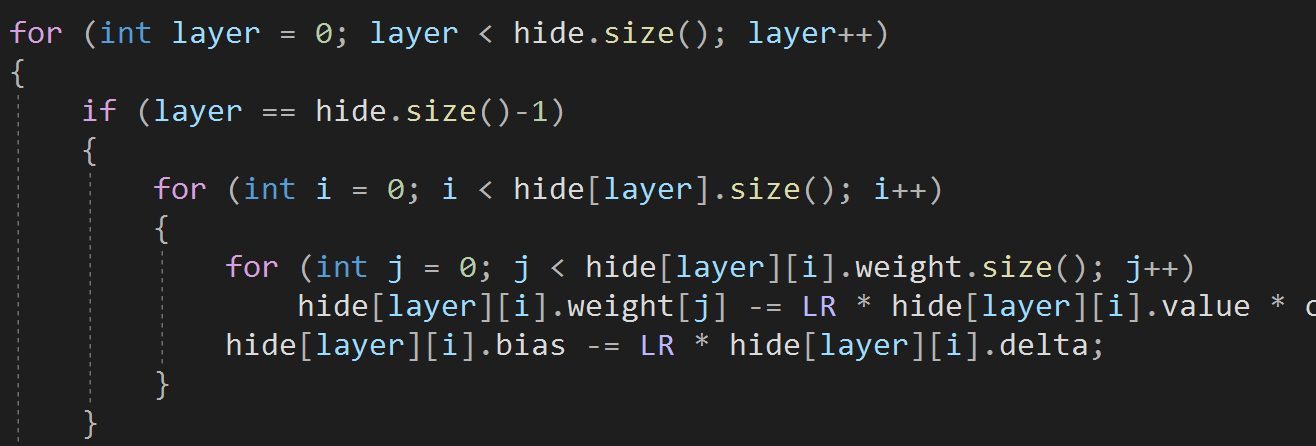
计算隐藏层各节点残差delta。



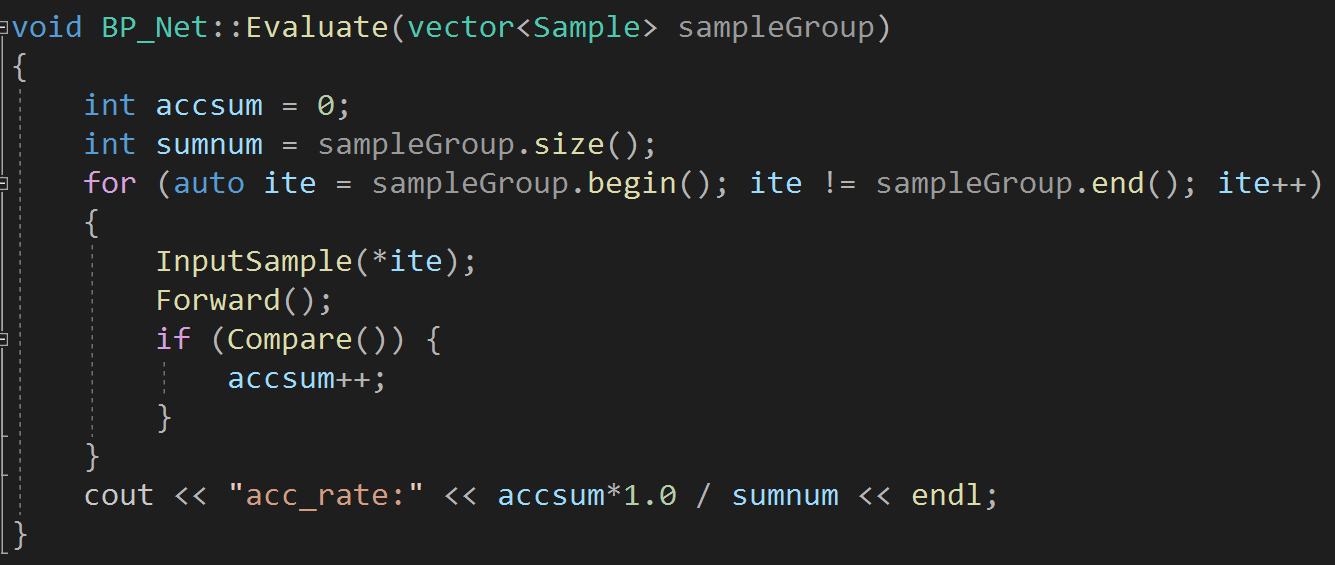
通过公式更新输出层各节点权值数组weight。



更新隐藏层各节点权值数组weight和偏置bias。

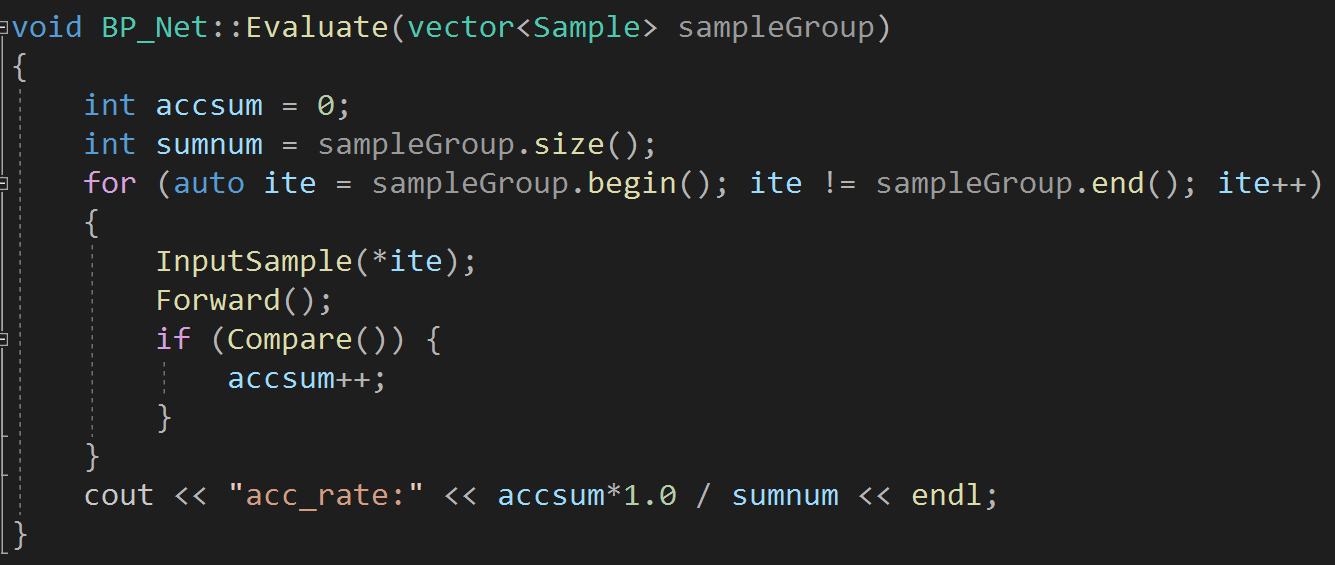


1. 模型预测



1. **实验结果**

测试代码：输出对测试集测试的准确率。

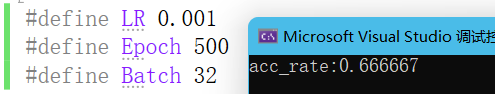
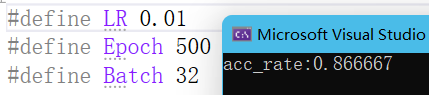
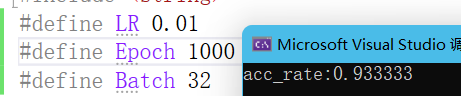
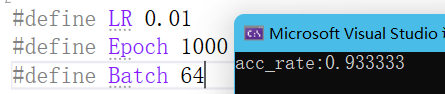
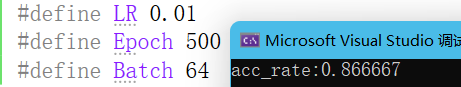


通过调试参数来改变训练的准确度。

LR：学习速率，也叫学习步长。

Epoch： 训练的轮数，迭代次数。

Batch：每次训练的批次大小。

1. 
2. 
3. 
4. 
5. 

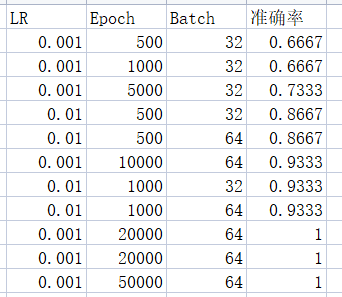
最高准确率为0.9333

1. **讨论分析**

程序实现过程中主要的困难在于设计程序框架，即选择合适的数据结构存放参数，以及各函数功能模块分配等，具体实现主要根据BP神经网络计算公式，困难较少，最终实现的神经网络在较高学习率下的准确率较高。

1.结果讨论：

本次实验结果我们对参数进行了10次修改，每次修改完后对结果进行准确的预测，其中五次的准确率整理如下图：

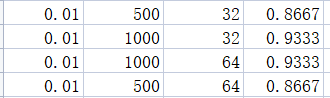


（按准确率排序）

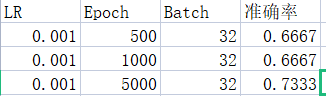
发现学习速率（LR）在一定范围约大（步长越大），准确率越高。训练的迭代次数（Epoch）约高，准确率约高。在本次实验中，每批次训练的大小（Batch）对结果的准确率没有影响。我们认为是样本数据较少，训练的数据较少导致每批次训练大小对结果的影响很小，而在最后结果中呈现不出来。

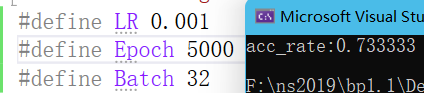
学习速率（学习步长）决定了参数每次更新的幅度，如果幅度过大，那么可能导致参数在极优值的两侧来回移动，而且可能造成每次迭代不会减小代价函数的结果，甚至会超过局部最小值导致无法收敛。相反，当学习率过小时，虽然能保证收敛性，但是这会大大降低优化速度。

本次实验中通过测试发现学习速率在0.01左右为最优，再保证训练速率较快的情况下准确率也较高。

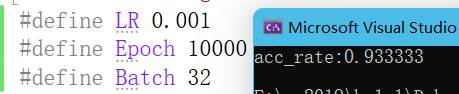


在本次实验中还测试了学习速率为0.001的情况，这时学习步长较小，需要非常大的训练轮次来支持它的准确率。而且在我们修改的几组参数中，准确率普遍较低。

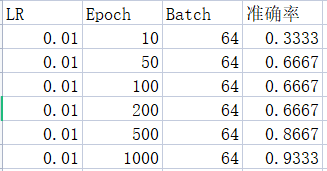




如果要使准确率和学习速率为0.01的准确率差不多，要将训练轮数达到10000，这会是学习速率为0.01的准确率的10倍。



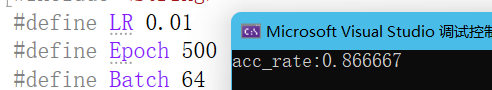
至于迭代轮次大小对本次结果的影响，我们通过调参发现：

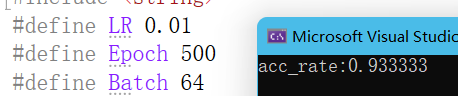


因为样本量较小，训练的样本量也较小，训练的样本量很小，所以迭代轮数对准确率的影响整体呈正向增加的关系，在一定区间内趋于稳定。

2.遇到的问题：

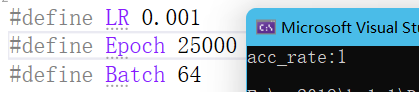
（1）在结果的预测中，我们发现当参数不变的时候，每次预测出来的准确率也不一样，这就是机器学习系统产生的误差。

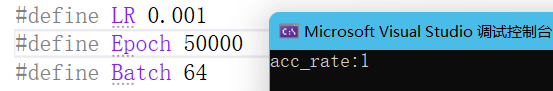


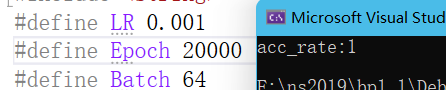


1. 本次实验的结果最高的测试结果为0.9333，低于我们的最初设想。我们想了以下办法可能会对结果的准确率进行提升：
2. 对BP算法的学习率LR进行进行动态调整，使指既能加速收敛，又能根据具体情况达到全局最优。
3. 对LR进行手动调整，我们了解到，当LR变小时可以使收敛度增加，以达到较高的准确率。所以将LR略微调小，增大训练轮数，提高准确率。
4. 加入抖度因子（参考网络）。

下面是我们用B方法调整后的结果准确率：







准确率有了明显的提高，但同时因为训练轮数的增加速率也明显下降。

1. **小组分工**

组长 吴凯文：程序的设计，误差的正向以及反向传播代码设计，数据训练，ppt制作等。

组员 刘恩众：数据乱序，准确率预测代码设计等。

组员 周诚宇：归一化代码设计，数据合并，报告的制作等。

组员 张嘉昌：文件导入代码设计，数据整理，报告制作等。

组员 王欣雨：配合工作。