

机器学习A课程

综合报告

专业班级：191191

学生姓名：叶宇涛

学生学号：20191000595

指导老师：刘超

**中国地质大学计算机学院**

**2022年 5月**

目 录

[一、基础题 5](#_Toc2762)

[（1） 浅谈对线性模型原理、优缺点的认识。 5](#_Toc7823)

[（2） 浅谈对决策树模型原理、优缺点的认识。 5](#_Toc16578)

[（3） 浅谈对神经网络模型原理、优缺点的认识。 5](#_Toc788)

[（4） 浅谈对支持向量机模型原理、优缺点的认识。 5](#_Toc15628)

[（5） 浅谈对贝叶斯分类器原理、优缺点的认识。 6](#_Toc22856)

[（6） 浅谈对集成学习原理、优缺点的认识。 6](#_Toc32599)

[（7） 浅谈对机器学习课程的主要收获和课程建议。 6](#_Toc20819)

[参考文献 7](#_Toc12694)

[二、课内实验报告包括验收情况 7](#_Toc26140)

[第三章编程题完成清单 7](#_Toc202)

[第1题 7](#_Toc8049)

[(1) 编程题目理解 7](#_Toc28950)

[(2) 对率回归算法原理阐述 8](#_Toc17998)

[(3) 对率回归算法设计思路 8](#_Toc12126)

[(4) 实验数据、实验流程、测试结果分析以及检查情况 8](#_Toc9333)

[(5) 代码结构注释、核心源代码简要分析 10](#_Toc13577)

[(6) 本次实验解决的主要问题、在理论学习与动手编程上的主要收获 11](#_Toc29112)

[(7) 编码及内容撰写中的参考来源。 11](#_Toc30359)

[第2题 12](#_Toc25764)

[(1) 编程题目理解 12](#_Toc23598)

[(2) 十折交叉验证与留一法原理阐述 12](#_Toc18797)

[(3) 十折交叉验证、留一法算法设计思路 12](#_Toc31564)

[(4) 实验数据、实验流程、测试结果分析以及检查情况 12](#_Toc22857)

[(5) 代码结构，核心代码简要分析 14](#_Toc826)

[(6) 本次实验解决的主要问题、在理论学习与动手编程上的主要收获 14](#_Toc7582)

[(7) 编码及内容撰写中的参考来源 14](#_Toc6652)

[第3题 15](#_Toc17842)

[(1) 编程题目理解 15](#_Toc5581)

[(2) 线性判别分析算法原理阐述 15](#_Toc19130)

[(3) 线性判别分析算法设计思路 15](#_Toc8038)

[(4) 实验数据、实验流程、测试结果分析以及检查情况 16](#_Toc23931)

[(5) 代码结构，核心代码简要分析 17](#_Toc27557)

[(6) 本次实验解决的主要问题、在理论学习与动手编程上的主要收获 17](#_Toc12761)

[(7) 编码及内容撰写中的参考来源 18](#_Toc28499)

[第四章编程题完成清单 18](#_Toc8207)

[第1题 18](#_Toc17938)

[(1) 编程题目理解 18](#_Toc4187)

[(2) 决策树算法原理阐述 18](#_Toc3511)

[(3) 决策树算法设计思路 20](#_Toc3435)

[(4) 实验数据、实验流程、测试结果分析以及检查情况 21](#_Toc16809)

[(5) 代码结构，核心代码简要分析 32](#_Toc31436)

[(6) 本次实验解决的主要问题、在理论学习与动手编程上的主要收获 33](#_Toc15864)

[(7) 编码及内容撰写中的参考来源 33](#_Toc6181)

[第2题 34](#_Toc4157)

[(1) 编程题目理解 34](#_Toc32523)

[(2) 决策树深度遍历算法原理阐述 34](#_Toc11406)

[(3) 决策树深度遍历算法设计思路 34](#_Toc30893)

[(4) 实验数据、实验流程、测试结果分析以及检查情况 35](#_Toc3851)

[(5) 代码结构，核心代码简要分析 36](#_Toc12716)

[(6) 本次实验解决的主要问题、在理论学习与动手编程上的主要收获 37](#_Toc22220)

[(7) 编码及内容撰写中的参考来源 38](#_Toc17876)

[第3题 38](#_Toc3381)

[(1) 编程题目理解 38](#_Toc10906)

[(2) 非递归队列广度优先算法原理阐述 38](#_Toc9469)

[(3) 非递归队列广度优先算法设计思路 39](#_Toc30346)

[(4) 实验数据、实验流程、测试结果分析以及检查情况 39](#_Toc22130)

[(5) 代码结构，核心代码简要分析 40](#_Toc13569)

[(6) 本次实验解决的主要问题、在理论学习与动手编程上的主要收获 41](#_Toc27315)

[(7) 编码及内容撰写中的参考来源 41](#_Toc12524)

[第五章编程题完成清单 41](#_Toc22019)

[第1题 41](#_Toc13931)

[(1) 编程题目理解 41](#_Toc16732)

[(2) 标准BP算法原理阐述 41](#_Toc2686)

[(3) 标准BP算法设计思路 42](#_Toc26089)

[(4) 实验数据、实验流程、测试结果分析以及检查情况 43](#_Toc4393)

[(5) 代码结构，核心代码简要分析 44](#_Toc7604)

[(6) 本次实验解决的主要问题、在理论学习与动手编程上的主要收获 45](#_Toc7765)

[(7) 编码及内容撰写中的参考来源 45](#_Toc1524)

[第2题 46](#_Toc23493)

[(1) 编程题目理解 46](#_Toc6328)

[(2) 累积BP算法原理阐述 46](#_Toc14688)

[(3) 累积BP算法设计思路 47](#_Toc22740)

[(4) 实验数据、实验流程、测试结果分析以及检查情况 47](#_Toc7778)

[(5) 代码结构，核心代码简要分析 49](#_Toc10599)

[(6) 本次实验解决的主要问题、在理论学习与动手编程上的主要收获 49](#_Toc24316)

[(7) 编码及内容撰写中的参考来源 50](#_Toc15681)

[第六章编程题完成清单 50](#_Toc23702)

[第1题 50](#_Toc27292)

[(1) 编程题目理解 50](#_Toc27946)

[(2) 支持向量机算法原理阐述 50](#_Toc19848)

[(3) 支持向量机算法设计思路 53](#_Toc27271)

[(4) 实验数据、实验流程、测试结果分析以及检查情况 54](#_Toc14002)

[(5) 代码结构，核心代码简要分析 56](#_Toc4206)

[(6) 本次实验解决的主要问题、在理论学习与动手编程上的主要收获 57](#_Toc4465)

[(7) 编码及内容撰写中的参考来源 58](#_Toc18674)

# 一、基础题

1. **浅谈对线性模型原理、优缺点的认识。**

线性模型是最简单的一种机器学习模型，是一类统计模型的总称，从原理来看，给出的形式如下：Y=wx+b，用线性的函数来模拟样例的分布，并且，在模拟的过程中，要求残差MSE最小，用来做分类或者回归任务都有不错的表现。在优化损失函数的过程中，可以用牛顿迭代或者梯度下降方法优化。线性模型优点是容易实现，原理清晰易懂，实现起来的效果也不是很差，但是，缺点也很明显，当输入变量不是相互独立，而是存在一定的关联性的时候，预测会比较困难[1]，这是因为线性模型并不可以判别出关联变量的内在关联。因此，线性模型更适合简单的预测。

1. **浅谈对决策树模型原理、优缺点的认识。**

决策树顾名思义，是一种树模型，通过训练数据形成if-then的判断结构。从树的根节点到叶节点的每一条路径构成一个判断规则。我们需要选择合适的特征作为判断节点，可以快速地分类，减少决策树的深度。由于是树模型，建立的方式和常规的树模型一样，通过递归的方式或者队列、栈的方式建立。在划分的时候，根据不同的数据特征函数，可以得出ID3、CART、C4.5等经典决策树模型。决策树的优点在于其计算量简单，实现简单，解释性强[2]，并且，易于处理有缺失值的样本，能够处理不相关的特征，可以同时处理离散或者连续性的数据。缺点是决策树模型容易过拟合，在大数据集的情况下，决策树规模会变得十分庞大。因此，需要进行预剪枝或者后剪枝来减少树的深度。并且，类别过多的时候，决策树的效果就不会很理想。

1. **浅谈对神经网络模型原理、优缺点的认识。**

神经网络模型是一种模拟大脑神经网络的人工模型。是由大量的神经元相互构建、连接，形成一个复杂的神经元网络。根据神经元的不同连接方式，可以形成不同类型的神经网络。其中，最经典的是前馈神经网络，分为输入层、隐藏层、输出层，且为全连接方式，经过反向传播算法，迭代优化神经网络中的总体残差，更新连接权、阈值，最后得出一个训练好的神经网络模型。神经网络模型可被证明是一种表达力非常强的模型，对于许多问题都有比较好的效果，具有比较好的鲁棒性、自适应性，可以充分逼近复杂的非线性关系[3].缺点在于其收敛的速度比较慢，在大量数据集的情况下，通常需要训练比较长的时间。其次，由于其强大的拟合能力，经常会出现过拟合的情况，需要进行人工修正。

1. **浅谈对支持向量机模型原理、优缺点的认识。**

支持向量机是一种二分类的模型，在所有数据点中给出一个超平面，使得不同分类的数据点到这个超平面的距离之和最大，通俗来说，就是找出一个平面，可以最大划分两种不同分类的数据点。对于线性问题可以分为平面，对于非线性问题就需要映射到高维空间中进行求解，这时候就需要核函数进行高维平面的求解。支持向量机的优点十分明显，在最终的模型中，起决定性作用的是支持向量，而不是所有的数据点，因此，支持向量机可以帮助我们抓住关键的特征[4]，剔除大量的冗余样本，并且，具有比较好的鲁棒性，在删除掉一些数据之后仍然不会影响整个模型的效果。缺点在于其对于大规模的数据集，训练速度十分缓慢，针对于这个问题，有SMO算法进行优化。其次，支持向量机对于多分类问题的解决方法不是很完善，需要组合多个支持向量机进行一对多问题的求解。

1. **浅谈对贝叶斯分类器原理、优缺点的认识。**

贝叶斯分类器是一种基于贝叶斯公式的概率统计分类模型，给出某个对象的先验概率，利用贝叶斯公式给出其后验概率，统计出后验概率最大的一个类别，将其作为该对象的类别。根据采取的条件不同，可以分为朴素贝叶斯、半朴素贝叶斯分类器。但是，在数据集中常常会有一些类别不存在数据的情况，因此，需要进行Laplace修正。贝叶斯分类器优点很明显，由于是有贝叶斯公式作为理论基础和数学基础，因此，在进行大规模数据的训练的时候，训练速度十分显著，并且，对于缺失值不敏感，在有数据集的增加的时候，可以比较轻易地修改模型概率。缺点在于其假设的条件比较苛刻，一般是假设其特征符合独立分布，但是现实中很多数据集特征存在一定的相关性，因此，贝叶斯分类器适用于垃圾信息分类、情感判别的环境中效果比较好。

1. **浅谈对集成学习原理、优缺点的认识。**

集成学习是一种应用多种机器学习框架组成的多个机器学习学习器，目的在于整合多个机器学习框架的优点、减少单个学习器带来的负面影响。可以通过组合多种学习器，最终形成一个强学习器。集成学习的过程中，首先需要构建基学习器，再组合基学习器，形成最终的模型。在构建的时候，根据构建的顺序，可分为平行构建、顺序构建两种方法。根据个体学习器的不同，集成学习方法大致可分为两大类：即个体学习器间存在强依赖关系，必须串行生成的序列化方法以及个体学习器间不存在强依赖关系、可同时生成的并行化方法[5]。根据组合的方式不同，有投票法、加权计算法、stacking。集成学习的优点在于其组合能力，利用多个学习器，可以达到较高的准确率，并且参数简单，容易实现。但是对于大规模数据，通常需要训练比较多的模型，导致训练速度的减缓。因此，集成学习通常采用并行化训练的方法执行。

1. **浅谈对机器学习课程的主要收获和课程建议。**

机器学习课程是一个充满挑战和趣味的课程。在学习的过程中，明白了机器学习的特点和研究对象，以及具体的应用。了解了基本的机器学习框架，包括线性模型、决策树、SVM等模型。主要收获在于可以自己编写其中的代码，在了解原理之后，可以根据原理以及流程图来手写代码，实现一个基本的机器学习框架。这对我来说既充满挑战，又对我后面写代码有帮助。了解了许多数学公式的推导以及应用，充分了解了高等数学和线性代数的重要性，并且，在学习的过程中，也巩固了我数学的基础，可以自己推导出相应的数学公式。最大的收获在于对于各类模型的原理的认识，以及相应实现代码的编写。

在学习的过程中，决策树第一道题的内容过多，需要写四个数据集的三种剪枝方式的三种不同的划分函数决策树，也就是36种情况。尤其是还要自己手写的情况下，只能完成大部分情况，对于对率回归的划分方法没有完成。因此，我建议削减这道题的内容，只需要1种数据集，3种剪枝方式和3种不同划分函数即可，完全不需要4个数据集来对比。在总体学习中，主要学习了分类的机器学习方式，对于回归并没有涉猎。然而回归任务在现实中也有广泛的应用。因此，可以适当加些回归的内容作为额外的讲解，可以帮助同学们更好理解其中模型的含义。

## 参考文献

1. <https://baijiahao.baidu.com/s?id=1570695125279051&wfr=spider&for=pc>
2. <https://wenku.baidu.com/view/6b5f1a020422192e453610661ed9ad51f01d543c.html>
3. <https://baike.baidu.com/item/%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C%E6%A8%A1%E5%9E%8B/10658952?fr=aladdin>
4. <https://www.csdn.net/tags/MtTakg3sMTU3MjktYmxvZwO0O0OO0O0O.html>
5. <https://baike.baidu.com/item/%E5%88%86%E7%B1%BB%E5%99%A8%E9%9B%86%E6%88%90/21512231?fromtitle=%E9%9B%86%E6%88%90%E5%AD%A6%E4%B9%A0&fromid=3440721&fr=aladdin>

# 二、课内实验报告包括验收情况

## 第三章编程题完成清单

## 第1题

### 编程题目理解

对率回归，是一种广义的线性模型，只是将线性回归方程中的换成了，其中，为。虽然是回归的方式，但是是用来做分类任务的，所以需要找一个单调可微函数，将分类任务的真实标记和线性回归模型的预测值联系起来。

和回归方法的方式一样，都是通过训练损失函数，得到最好的参数，带入到机器学习模型中。因此，通过划分西瓜数据集中的数据为训练集、测试集，计算查全率、F1-score，来对比出模型效果的好坏。

### 对率回归算法原理阐述

对率回归算法实际上是分类算法，利用线性模型函数，再代入函数中计算出的值从而判断分类的类别*。*因此，对数几率回归其实是广义线性模型的特例，原始线性模型公式为：

通过函数将值单位化，即转化为(0,1)区间上面的值。其中，函数如下：

将线性回归方程带入激活函数中。其中，通过极大似然法估计线性模型的。得到的最后结果如下：

其中，求解该最大化函数可用梯度下降算法或者牛顿法求解非线性函数的解。

### 对率回归算法设计思路

以这道题为例子，首先需要读取西瓜数据集，利用pandas库读取十分方便。将好瓜、坏瓜单独设置符号，画出散点图。

因此，首先需要设计激活函数，根据书上面的公式对结果进行梯度下降，得到和。根据参数，画出根据以密度为轴，含糖率为轴的直线，对比效果。对进行预测，得到的进行激活函数的转化，如果结果大于0.5，判定为好瓜，如果小于0.5，判定为坏瓜，根据原始值，计算精确度。

### 实验数据、实验流程、测试结果分析以及检查情况

给出数据集如下：

表格

描述已自动生成

图 1 西瓜数据集

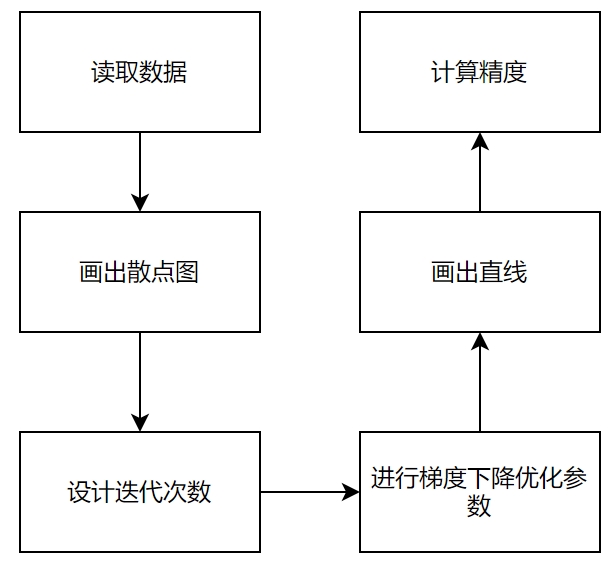


图 2 对率回归流程

计算得到：



图 3 准确率

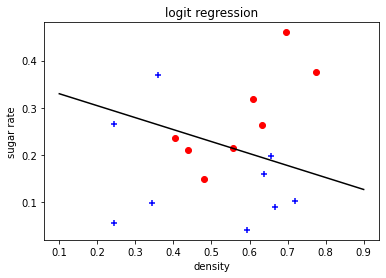


图 4 含糖率、密度图

其中，红点表示好瓜，蓝点表示坏瓜，直线表示对率回归拟合直线。在直线上面的点判定为好瓜，在直线下方的点判定为坏瓜。

观察可以发现，最后的对率回归直线将大部分好瓜、坏瓜划分出来，但是，仍然存在部分好瓜坏瓜划分错误，这是因为对率回归是线性回归，无法完全将所有品类划分出来，如果需要精确划分，需要非线性映射。

### 代码结构注释、核心源代码简要分析

定义激活函数，返回的值用于二分类。



定义梯度下降法，将参数值初始化为，计算线性方程的值，进行梯度下降优化参数，返回得到的参数值。



最后，根据得到的参数值计算y，对y进行激活函数映射，结果大于0.5判定为好瓜，结果小于0.5判定为坏瓜。对比原来的y值，相同说明正确。最后计算得到的准确率。

### 本次实验解决的主要问题、在理论学习与动手编程上的主要收获

本次实验让我了解到了线性回归用于分类问题要如何进行。对于该问题，首先需要定义激活函数，用于分类，再对最大化函数进行梯度下降，得到参数值。最后计算准确率。

遇到的问题主要在于不熟悉numpy和pandas的操作，需要对数据集的形态进行反复的确认，不然会出现矩阵大小不匹配的情况。并且，对于公式的编码也需要反复debug测验数据。

收获在于了解了对率回归算法的实现，可以自己手动实现算法，了解了激活函数在分类问题中的作用。并且，对公式的推导更加熟悉于心。对于matplotlib库的应用更加熟悉，可以画出符合要求的图像。

### 编码及内容撰写中的参考来源。

[1]https://zhuanlan.zhihu.com/p/27585172，机器学习：对数几率回归

[2]https://zhuanlan.zhihu.com/p/259848053, 对数几率回归

[3]https://blog.csdn.net/llwleon/article/details/79204790, 《机器学习》周志华 课后习题3.3：编程实现对率回归，并给出西瓜数据集上的结果.

## 第2题

### 编程题目理解

需要对于iris数据集，比较十折交叉验证和留一法的错误率。对于模型，可以选用对率回归或者其他方法进行验证。查看iris数据集，发现要预测的species存在三种数据，而对率回归只能够完成二分类任务。如果需要完成多分类任务，需要指定分类方式：ovr即one-vs-rest(OvR)，multinomial是many-vs-many(MvM)。由于对于速度没有要求，选择multinomial进行分类。题目中的错误率是一种指标，公式如下：

### 十折交叉验证与留一法原理阐述

十折交叉验证是一种用来测试算法准确性常用的算法，将数据集分成十份，轮流将其中9份作为训练数据，1份作为测试数据，进行试验。每次实验会得出相应的正确率，将十次的正确率的平均值作为对算法精度的估计。本题目中的错误率实际上就是1-正确率。

留一法也是同样的评估方法，实际上是交叉验证法的特例。每次都只留下一个样本作为测试集，剩下的样本作为训练集。最后对该样本进行预估。比如对于iris数据集，将149个样本作为训练集，将1个样本作为测试集，对模型训练完之后，对该样本进行预测测试。重复进行150次。这种方法一般比较准确，但是训练的时间比较长，时间开销大对于数据量大的数据集是无法忍受的。

### 十折交叉验证、留一法算法设计思路

将iris数据集导入，画出每个特征之间的关系，查看每个特征对于分类的影响。导入sklearn中的对率回归模型、十折交叉验证以及留一法。对于十折交叉验证，采用函数对模型进行训练，计算出准确率。对于留一法，分150次划分训练集，测试集，通过拟合模型，对剩下的一个样本进行测试，如果分类错误，错误数量+1.最后，计算出错误率。

### 实验数据、实验流程、测试结果分析以及检查情况

实验数据采用UCI中的iris数据集。

图示

描述已自动生成

图 5 3.4实验流程

图表

描述已自动生成

图 6 iris各数据集之间关系

图形用户界面, 文本, 应用程序, Word, 电子邮件

描述已自动生成

图 7 十折交叉验证的错误率

图形用户界面, 文本

描述已自动生成

图 8 留一法错误率

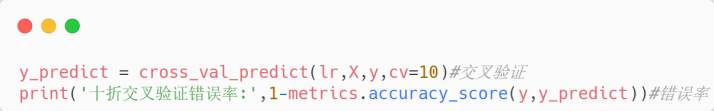
对比可以看出，留一法的错误率比较高，而留一法一般来说，准确率会高一点，可以相信该指标对于该模型的评估作用。但是留一法的性能消耗太大，一般还是采用十折交叉验证进行评估。

### 代码结构，核心代码简要分析

导入数据集，画出特征之间的关系：



进行十折交叉验证，计算错误率：



进行留一法验证，计算错误率：



### 本次实验解决的主要问题、在理论学习与动手编程上的主要收获

本次实验主要解决的问题是逻辑回归对于多分类问题应该如何处理。并且，对于十折交叉验证和留一法相关的数据输入需要特别注意，不然会导致报错。主要收获是了解了十折交叉验证和留一法的基本原理，了解了错误率的计算公式，可以利用sklearn中的相关函数对iris数据集进行测试，对比错误率。

### 编码及内容撰写中的参考来源

[1]https://blog.csdn.net/weixin\_44350982/article/details/102667884, 周志华《机器学习》课后习题3.410折交叉验证法和留一法UCI实例比较

[2]https://baike.baidu.com/item/%E7%95%99%E4%B8%80%E6%B3%95/22499706，留一法简介

[3]https://baike.baidu.com/item/%E5%8D%81%E6%8A%98%E4%BA%A4%E5%8F%89%E9%AA%8C%E8%AF%81，十折交叉验证简介

## 第3题

### 编程题目理解

线性判别分析是一种经典的线性学习方法，本题目要求调用该算法对西瓜数据集3.0α中的数据进行分析，得出结果。因此，需要进行该方法的实现，并且根据该方法对西瓜数据集进行判别，画出相应图像。因此，需要导入西瓜数据集，并且根据书上面的公式拟合直线。

### 线性判别分析算法原理阐述

线性判别分析要求将样本投影到一条直线上，并且，要求同类样本的距离尽可能近，不同样本尽可能远。对于新的样本，进行投影之后，根据投影点的位置确定新样本的类别。其中，直线的方程为：

为了使同类样本距离尽可能近，不同样本距离尽可能远，给出了最大化目标：

其中，利用拉格朗日乘子法有：

最后得到的公式如下，其中为平均值。

我们的目标就是求出.有了公式，写代码就会变得比较简单。

### 线性判别分析算法设计思路

首先输入数据集，定义函数，划分为坏瓜，好瓜，根据坏瓜，好瓜样本的平均值，协方差，求出，根据以及求出最后的直线参数，我们要求的直线斜率可以根据的值进行求解。

然后，根据原来的值，根据映射公式，映射到直线上，画出新映射点。

最后，画出原来的值，直线.

### 实验数据、实验流程、测试结果分析以及检查情况

实验数据采用西瓜数据集3.0.

图示

描述已自动生成

图 9 线性判别分析流程图

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图 10 计算成果

图表, 散点图

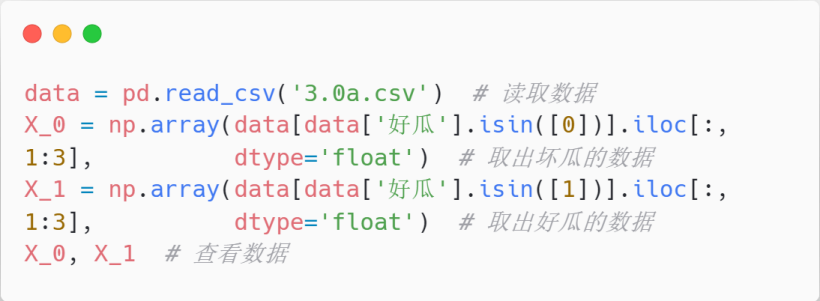
描述已自动生成

图 11 好瓜、坏瓜、映射点、投影直线

可以看出，效果不是很好，好瓜、坏瓜都粘合在了一起，好瓜与坏瓜的投影点不够远离，坏瓜与坏瓜之间的投影点不够聚集。原因可能是数据集的数量太少，不能很好的拟合直线，导致出现这种情况。后续需要改进的话，可以增加数据集的个数，来更好的拟合直线。

### 代码结构，核心代码简要分析

首先，需要读取数据：



核心代码，用来计算直线的参数，需要输入两个矩阵，求出矩阵以及协方差，返回.



定义一个映射函数，通过LDA直线的，映射原来的x值到LDA直线上，画出相应图像。



### 本次实验解决的主要问题、在理论学习与动手编程上的主要收获

这次实验主要收获是了解了线性判别分析的算法原理，可以自己推导、计算得到最后的公式。并且，更加熟悉了numpy中的matrix数据结构的用法，对matplotlib的理解更加深刻。了解了numpy中常见的函数用法。了解了线性模型具有一定的局限性，对于圆圈数据不能很好的划分。

主要的问题是对于scatter的函数用法不太熟悉，需要反复确认验证数据，查阅资料，才可以画出图像。

### 编码及内容撰写中的参考来源

[1]https://blog.csdn.net/A993852/article/details/80099258，周志华《机器学习》3.5答案-编程实现线性判别分析，并给出西瓜数据集上的结果

[2]周志华 机器学习[M] 清华大学出版社 2016：P60-62

[3]周志华 机器学习[M] 清华大学出版社 2016：P29-30

## 第四章编程题完成清单

## 第1题

### 编程题目理解

题目要求选择4个UCI数据集，分别进行基于信息增益、基尼指数、对率回归的决策树算法的实现，并且还要实现未剪枝、预剪枝、后剪枝三种不同的算法。对于不同的决策树，改变的是划分的依据，也就是信息增益、基尼指数这一类指标的不同。在这里选择了以下数据集：Iris、wine、breast\_cancer、diabetes。都是UCI中的分类数据集，并且没有少数据和异常数据的情况。其中，主要利用iris数据集进行测试。

### 决策树算法原理阐述

决策树是一种分类模型。其输入是带有标签的数据，输出是一颗决策树。其非叶节点代表的是逻辑判断；叶节点代表的是分类的子集。决策树算法原理是通过训练数据形成if-then的判断结构。从树的根节点到叶节点的每一条路径构成一个判断规则。我们需要选择合适的特征作为判断节点，可以快速地分类，减少决策树的深度。最理想的情况是，通过特征的选择把不同类别的数据集贴上对应类标签，树的叶子节点代表一个集合，集合中数据类别差异越小，其数据纯度越高。

决策树划分伪代码如下：



图 12 决策树算法伪代码

其中，根据数据特征函数的不同，可以划分不同的算法。比较经典的算法有ID3、C4.5、Cart算法。

**ID3：**

ID3使用的数据特征函数是信息增益，给出公式如下：

其中，

计算所有类别的信息增益，选择信息增益最大的作为分类节点，这就是ID3算法的划分依据。

**CART：**

CART算法用基尼指数作为划分指标：

这代表每个类别中的两个样本不同的概率，也可以依据这个判断样本的纯度。

**基于对率回归：**

基于对率回归算法的指标就是利用对数几率回归的算法划分样本，对于一个树节点中的样本，根据全部特征可以进行对率转换为如下形式：

然后通过梯度下降算法求得对率回归的一组参数，通过这组参数，可以将该节点样本值分为两类。实现的决策树应如下图所示：



图 13 基于对率回归算法示意图

**预剪枝：**

预剪枝需要预先设置一个验证集，在每次划分节点的时候，计算不划分时候的精确度，以及假如划分之后的精确度。如果不划分时候的精确度比划分的高，那么划分，否则，将当前节点设置为叶子节点，类别设置为当前数据集中样本数量最多的类别。

**后剪枝：**

对于后剪枝算法，同样需要设置验证集。在已经建好树的情况下，递归判断当前树节点的验证集精确度和划分之后的验证集精确度，来确定是否需要划分。如果不需要划分，将当前节点设置为叶子节点，类别设置为数据集中样本数量最多的类别。

### 决策树算法设计思路

首先需要构建一棵决策树，因此，将按照书上面的流程进行建树。设置一个节点类，存放该节点的子树节点、特征类别等变量。再设置一个决策树类，初始化的时候初始化根节点、以及采用基尼指数或者信息增益的方法。并判断：

1. 如果输入的数据集中样本全部属于一类，判定为叶子节点，返回；

2. 如果输入的数据集中特征用完，判定为叶子节点返回；

3. 否则调用分割函数，找到最佳的特征名称和分割点；

4. 如果是离散值，根据最佳的特征名称进行划分，并且删除这个最佳的特征，对该特征下的特征值循环生成子树。

5. 如果是连续值，根据最佳的分割点划分D+和D-，对于D+部分递归生成子树，对于D-部分递归生成子树。

6. 打印该树。

**对于如何选择最佳信息增益，分别根据离散值以及连续值进行判断：**

1. 如果这个特征是离散值，那么，该特征求出的信息增益值，返回信息增益值。

2. 如果这个特征是连续值，那么，对该特征下的所有值求中点，求取完中点后，遍历每一个中点，划分为D+和D-，求取信息增益值。最后，选择最大的信息增益值，返回信息增益值和划分点。

3. 根据最大的信息增益值选择特征，如果是离散特征，需要删除这个特征，如果是连续特征，则不需要删除。

**预剪枝算法设计：**

1. 判断当前节点是否是叶子节点或者数据集为空，是则返回；
2. 计算当前节点中样本最多的类别，计算未划分时候的精确度；
3. 计算最佳的划分点，进行假设划分，计算划分后的验证集精确度，如果划分效果好于不划分，那么划分。
4. 设置子树节点，递归调用计算。

**后剪枝算法设计：**

1. 判断当前节点是否是叶子节点或者数据集为空，是则返回；
2. 计算当前节点中样本最多的类别，计算未划分时候的精确度；
3. 递归调用建树。
4. 如果递归到左右子树都为叶子节点，判断是否划分时候验证集的精确度好于不划分时候的精确度，是则设置当前节点为叶子节点，重新设置高度、类别设置为数据集中样本数量最多的类别，进行剪枝。
5. 返回。

**最后，利用matplotlib对该树进行打印。打印步骤如下：**

1. 根据叶子节点的个数，选择合适的x偏移量和y偏移量进行打印。
2. 输入节点，判定这个节点是不是叶子节点，如果是叶子节点，直接打印。
3. 如果不是叶子节点，递归调用该函数，增加x偏移量和y偏移量。并且打印线的名称。

### 实验数据、实验流程、测试结果分析以及检查情况

实验数据为UCI数据集中的Iris、wine、breast\_cancer、diabetes数据集。

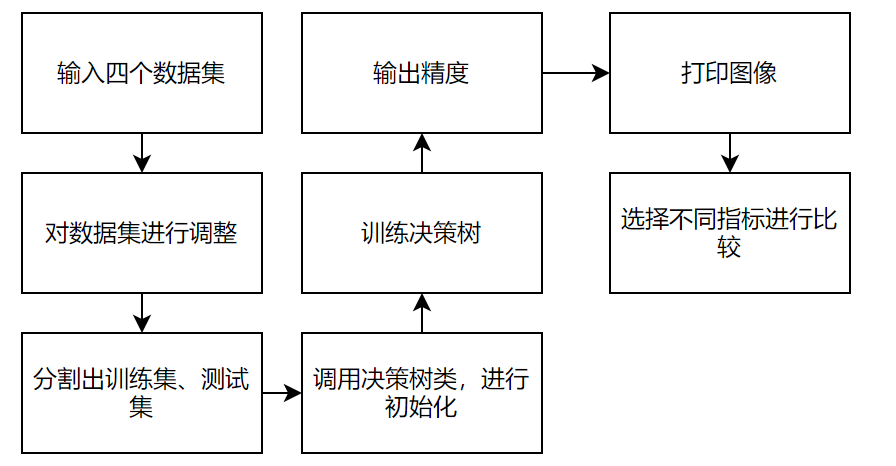


图 14 决策树算法流程示意图

对于信息增益：

**未剪枝：**

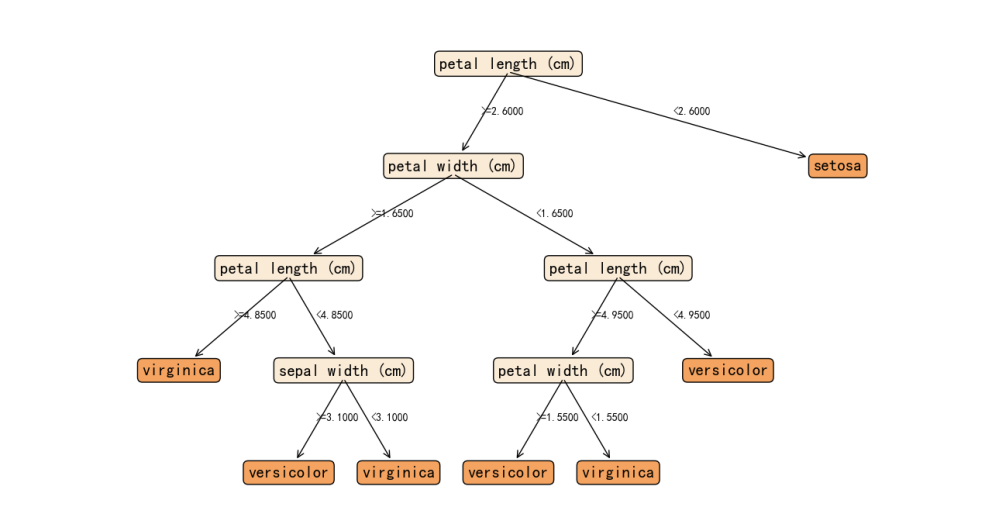




图 15 信息增益IRIS数据集

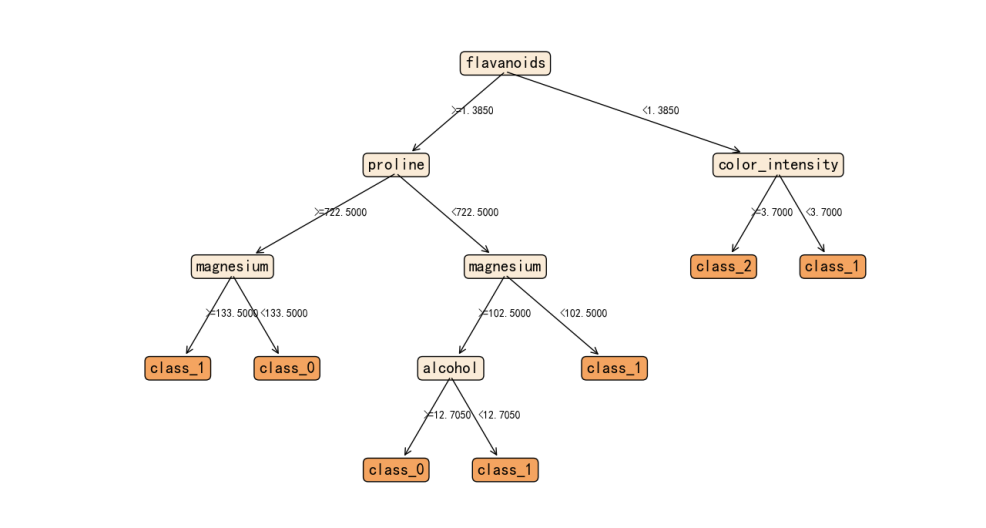




图 16 wine数据集

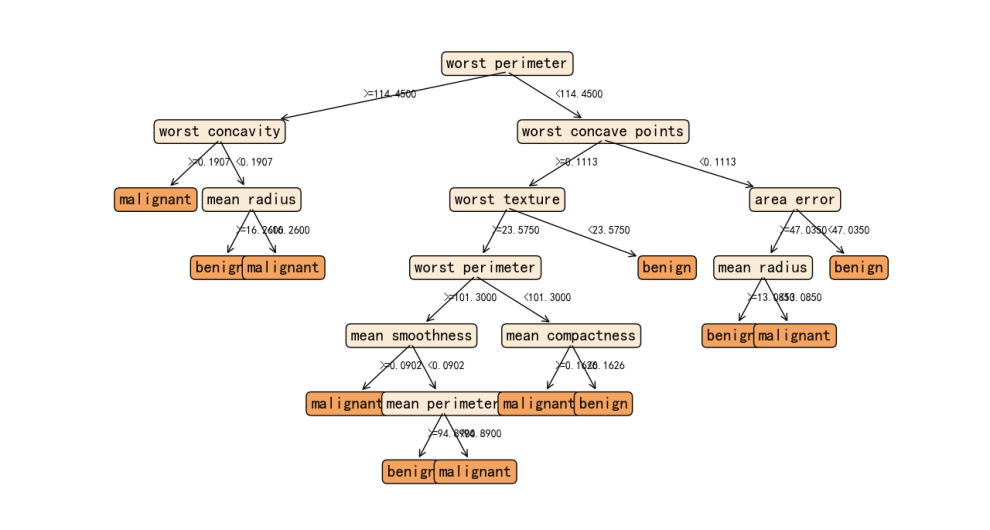


图 17 breast\_cancer数据集

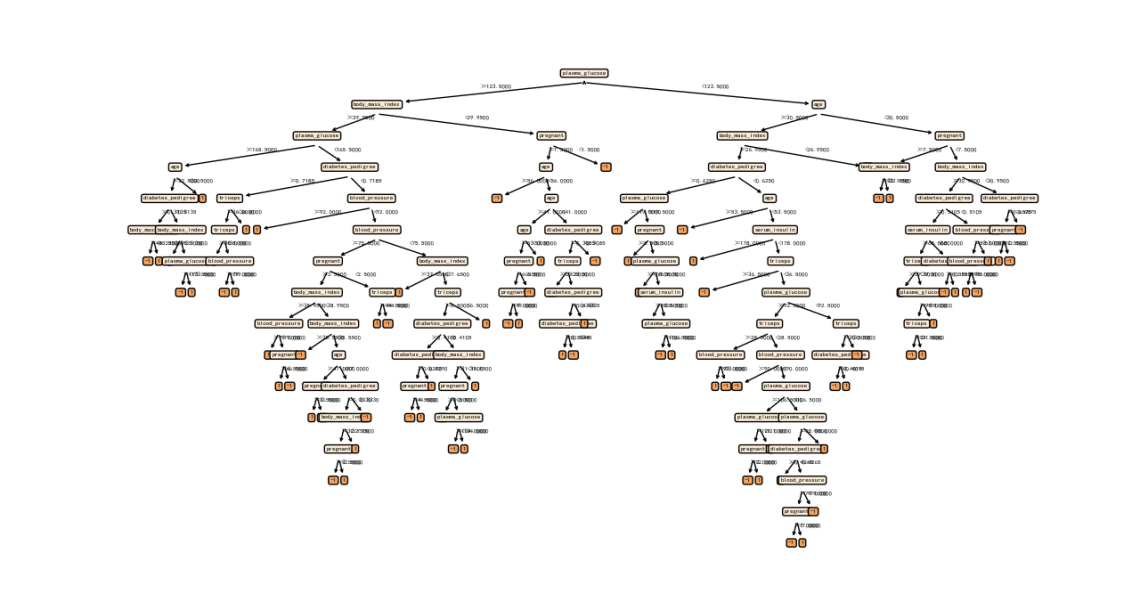




图 18diabetes数据集

预剪枝：

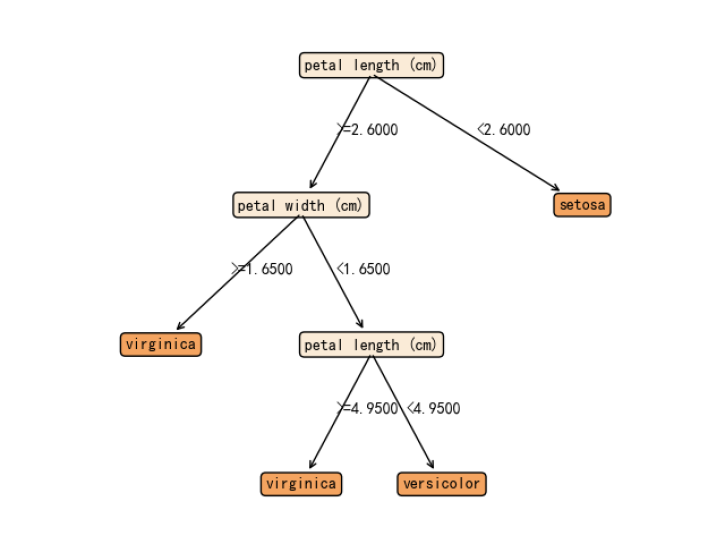




图 19 预剪枝的iris数据集基于信息增益决策树

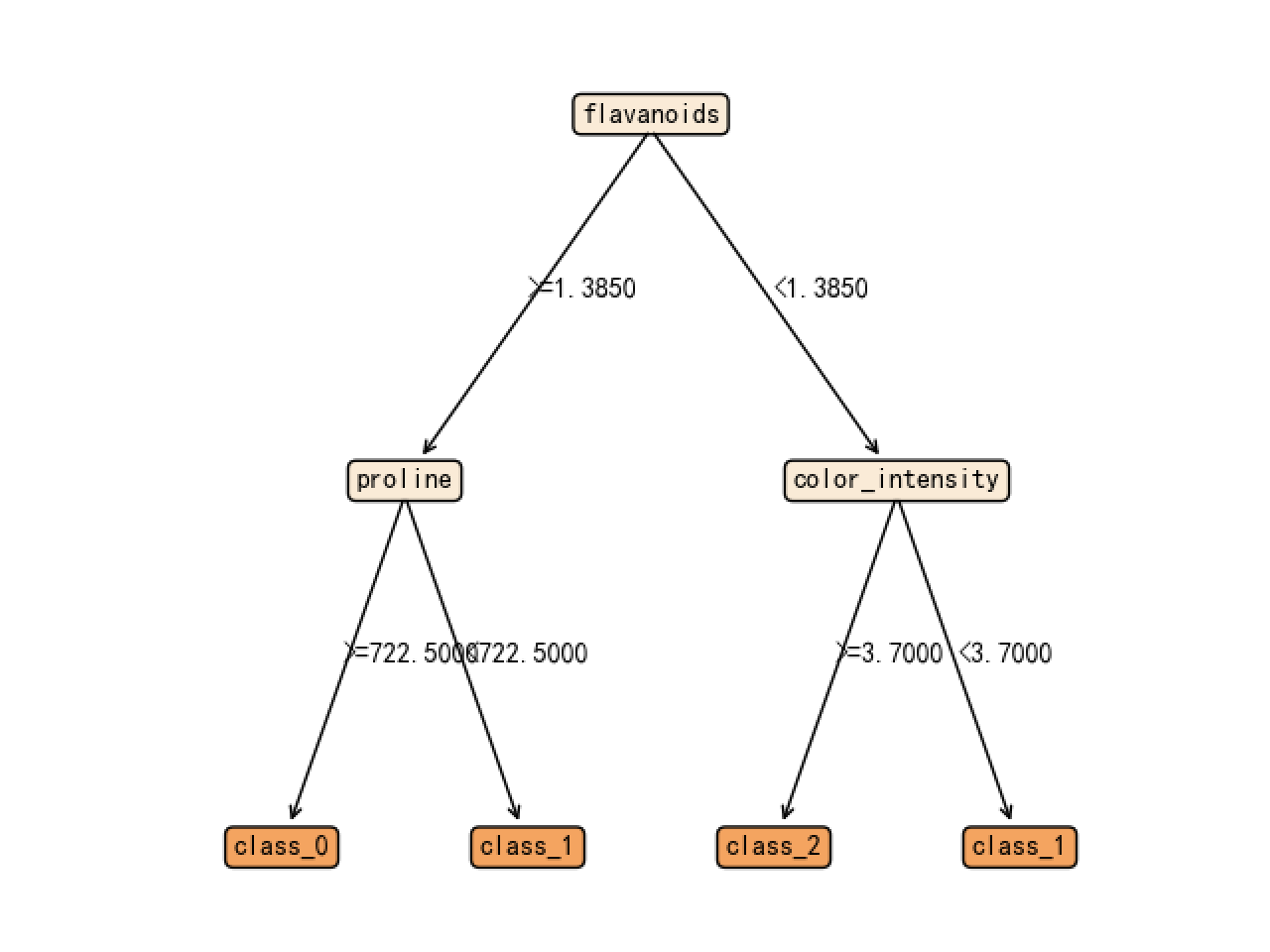




图 20 预剪枝的wine数据集基于信息增益决策树

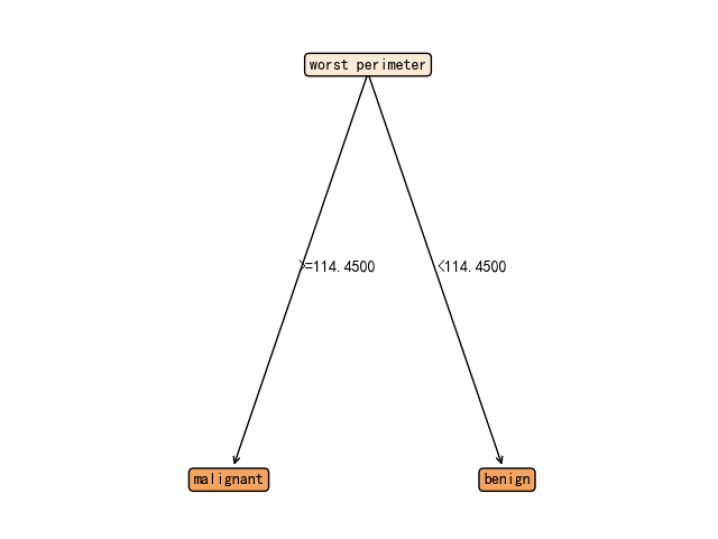




图 21 预剪枝的breast\_cancer数据集基于信息增益的决策树

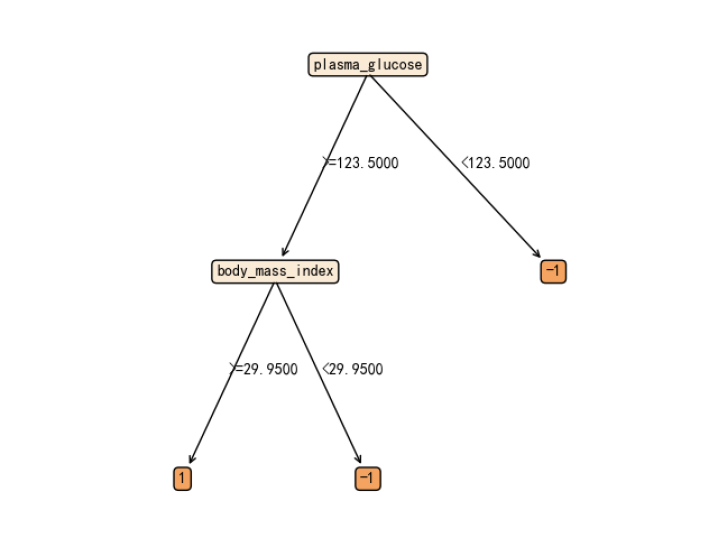




图 22 预剪枝的diabetes数据集的基于信息增益的决策树

**后剪枝：**

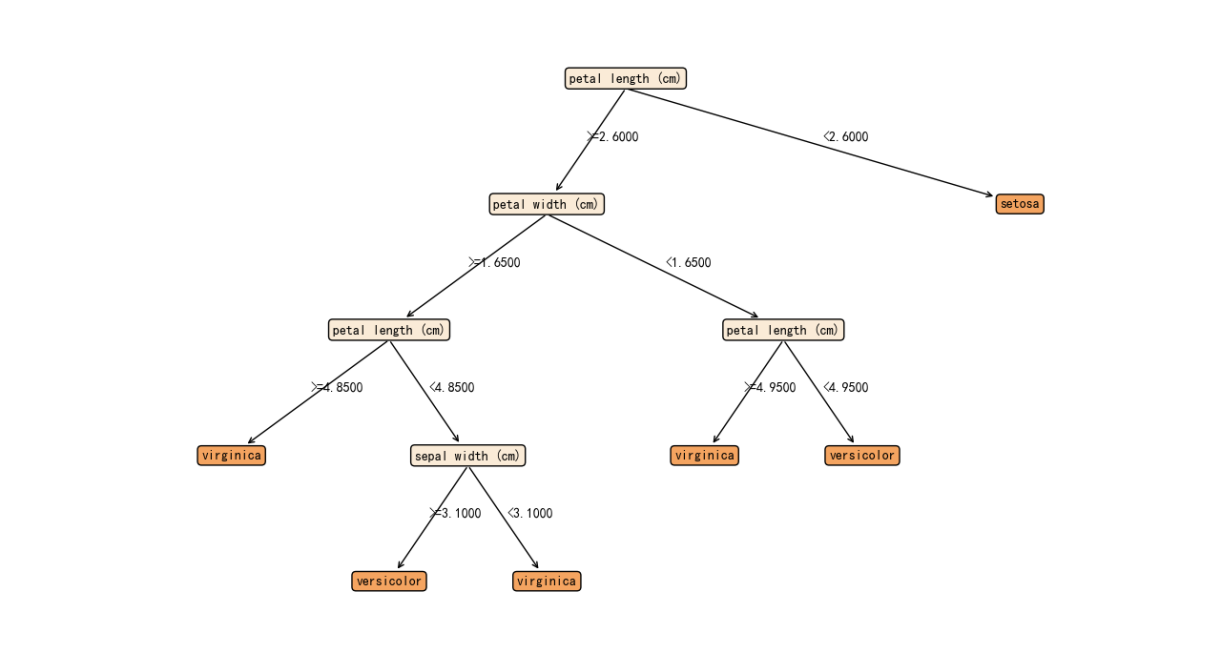




图 23 后剪枝的iris数据集基于信息增益的决策树

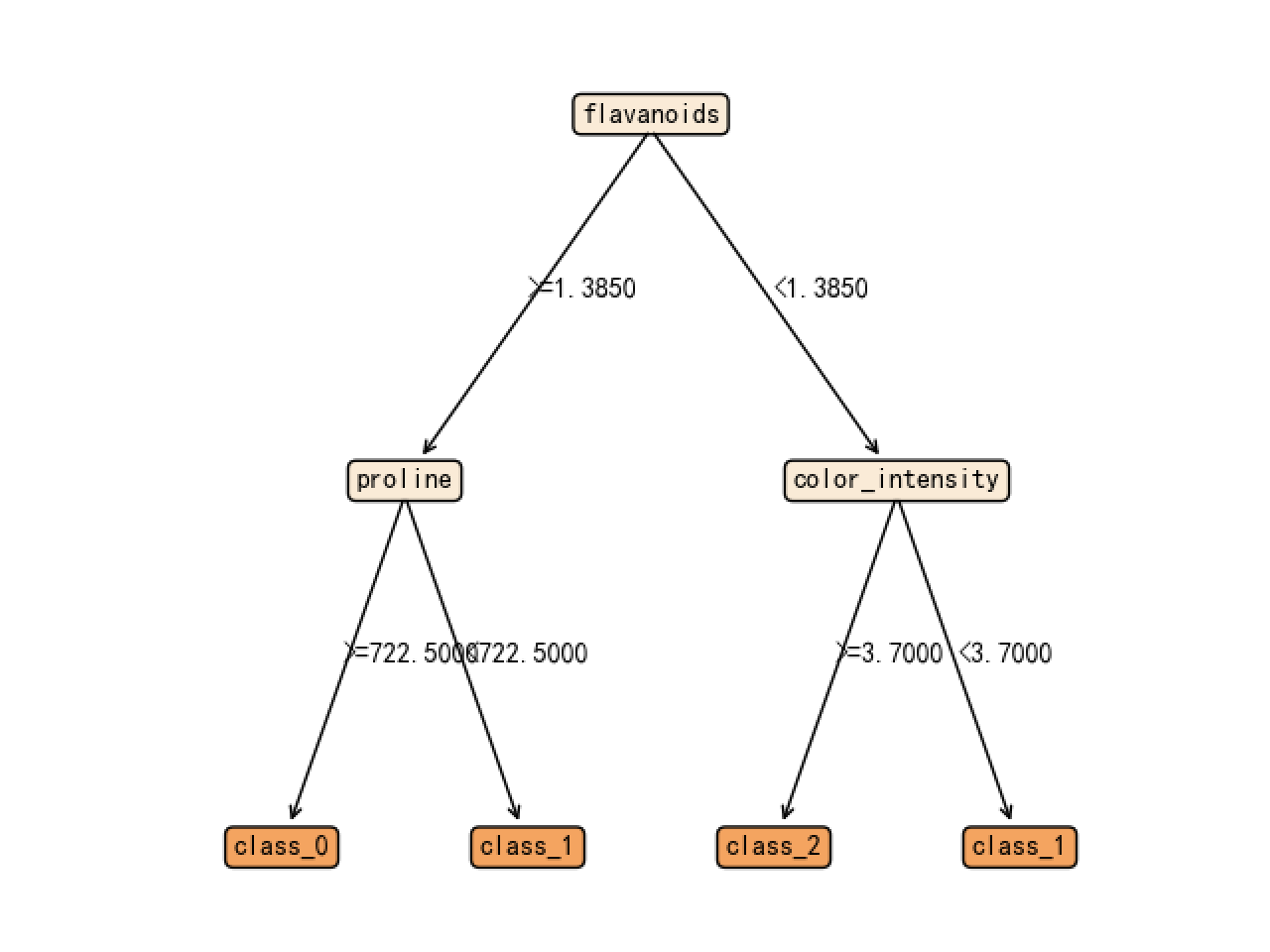




图 24 后剪枝的wine数据集基于信息增益的决策树

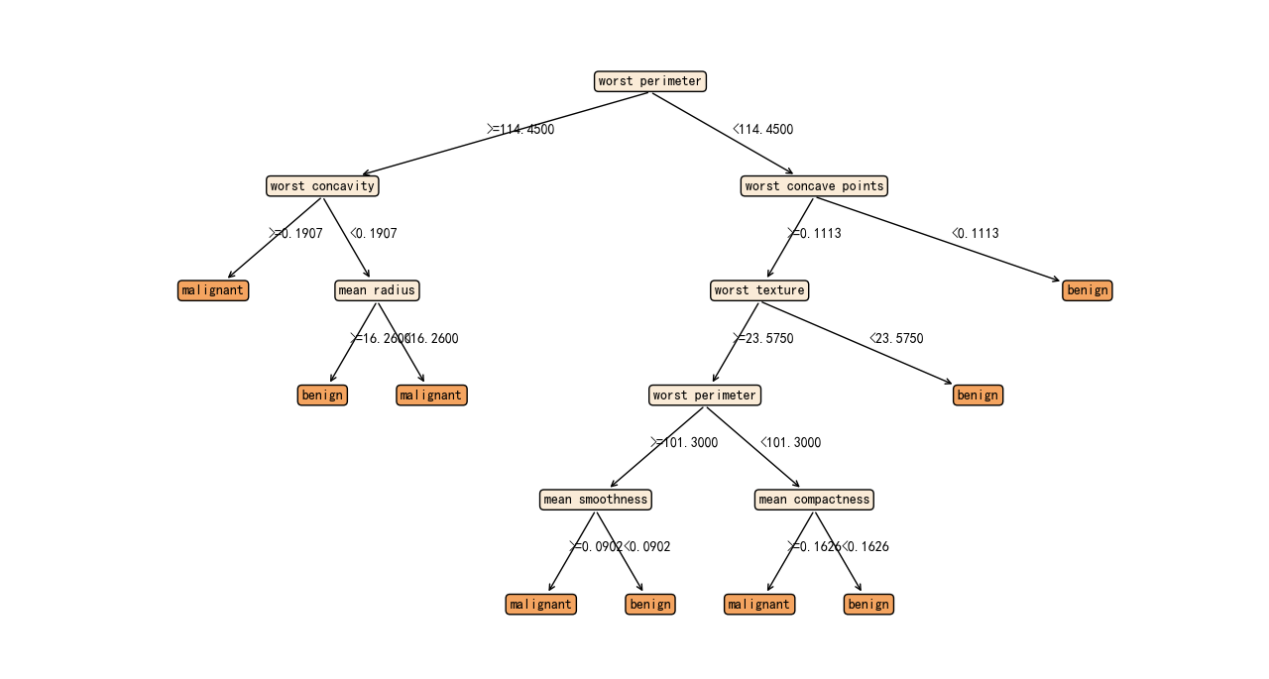




图 25 后剪枝的breast\_cancer数据集基于信息增益的决策树

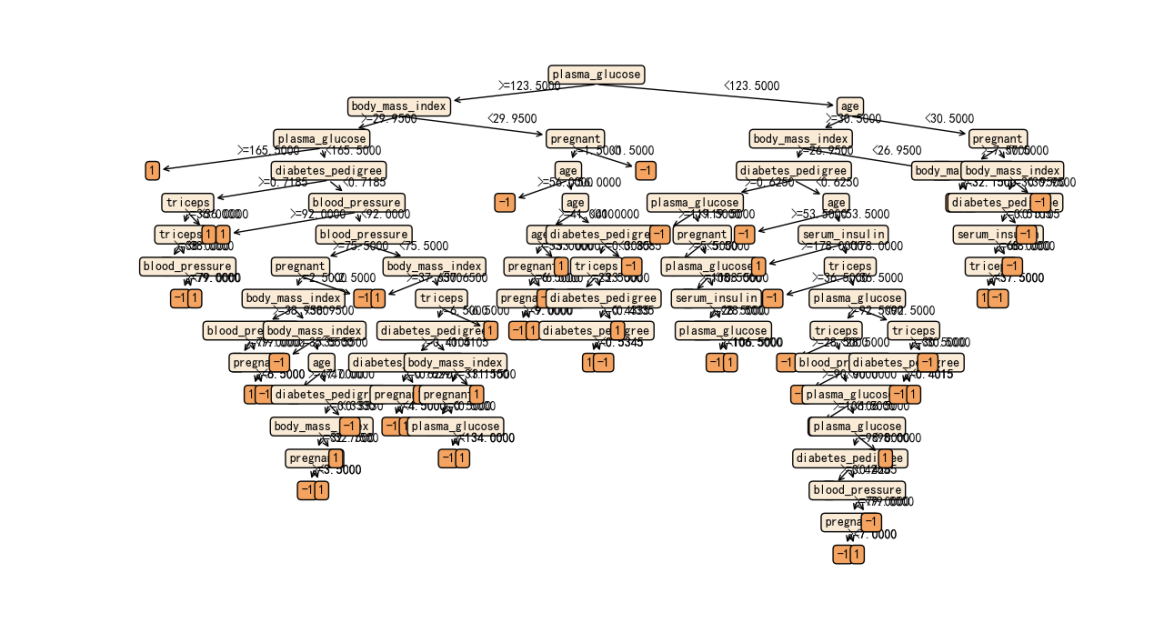




图 26 后剪枝的diabetes数据集基于信息增益的决策树

对于基尼指数：

未剪枝：

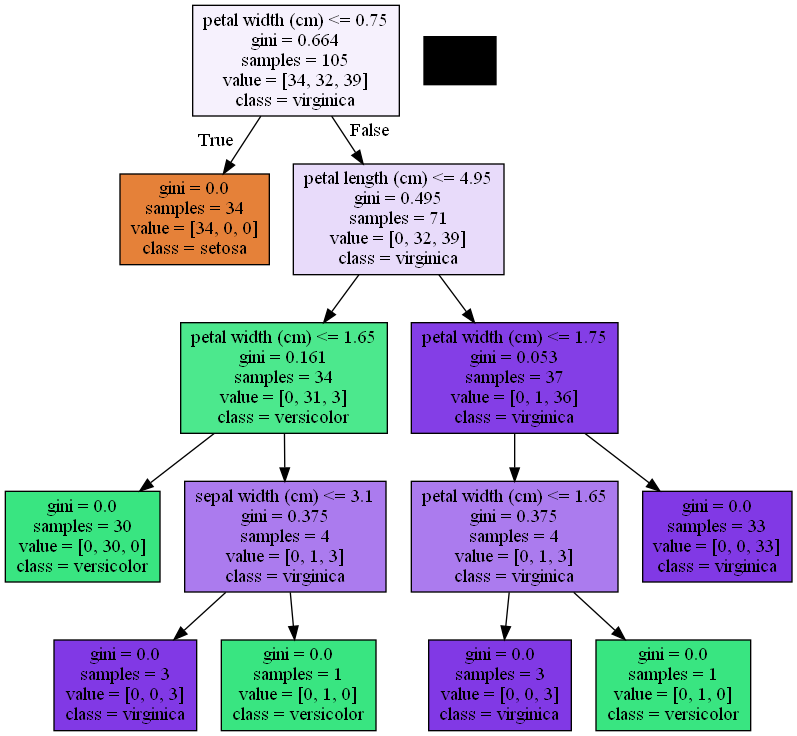




图 27 iris数据集基于基尼指数的决策树

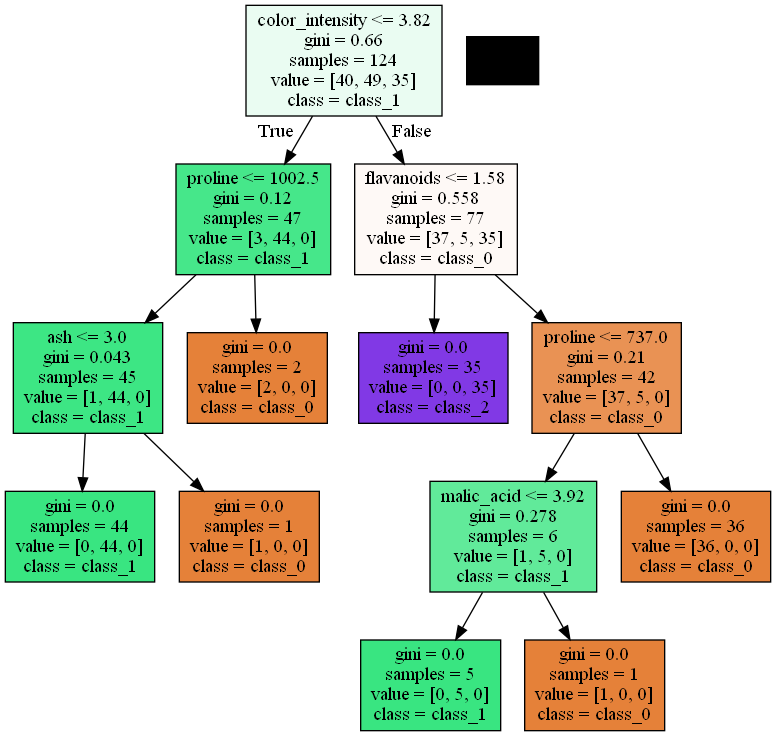




图 28 wine数据集基于基尼指数的决策树

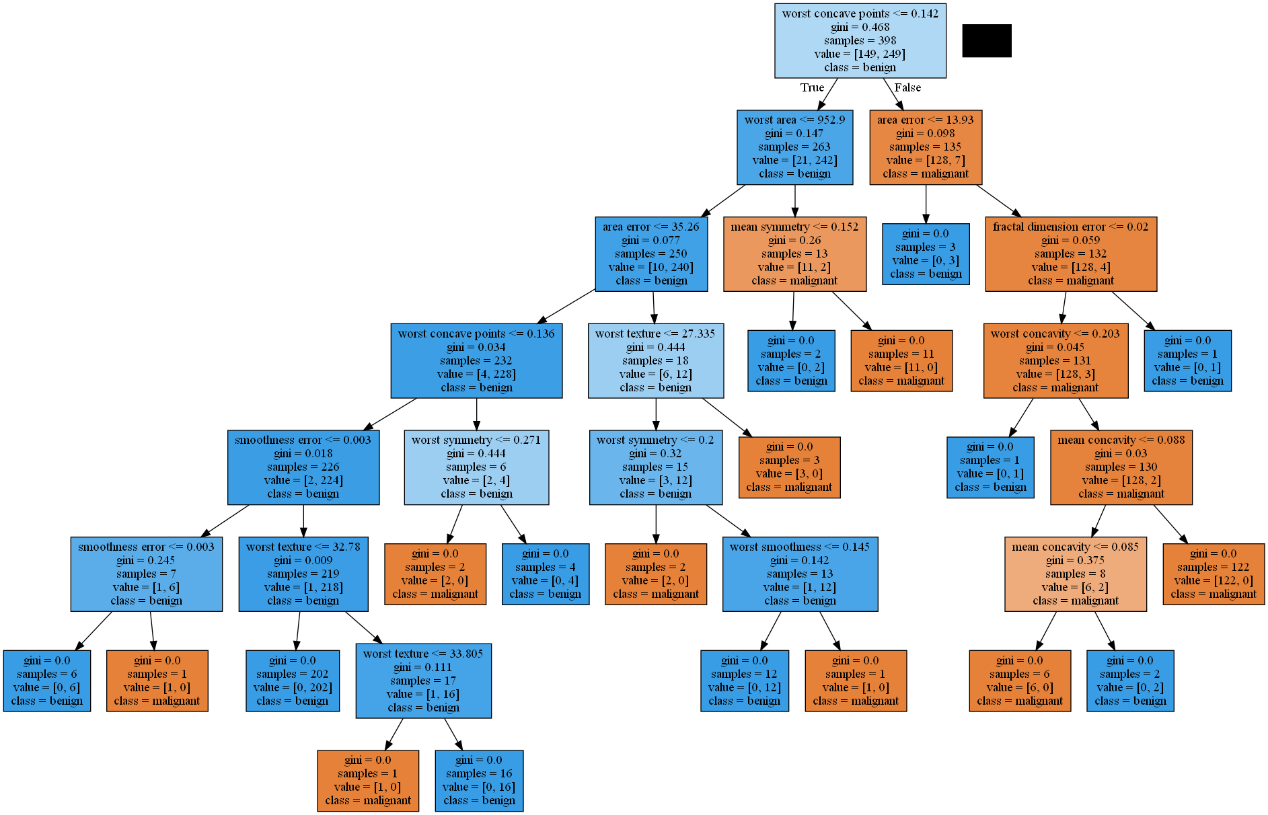




图 29 breast\_cancer数据集基于基尼指数的决策树

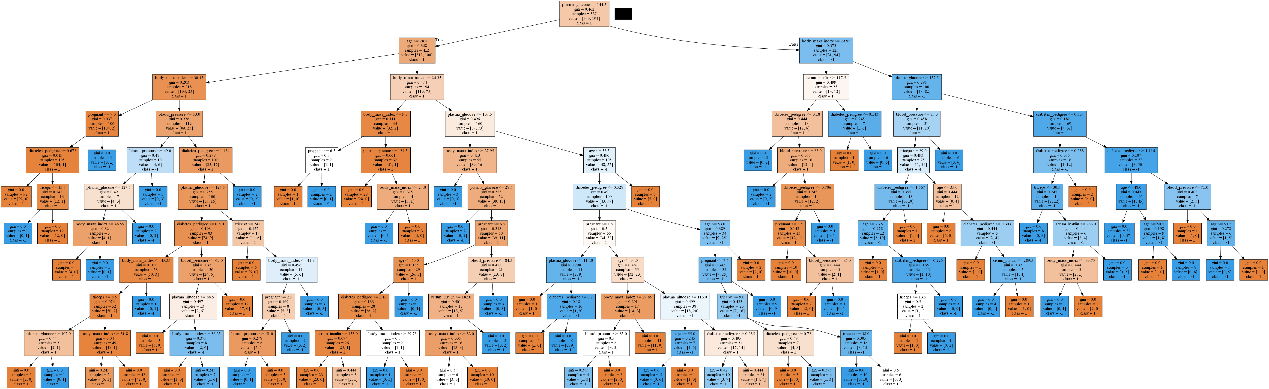




图 30 breast\_cancer数据集基于基尼指数的决策树

预剪枝：

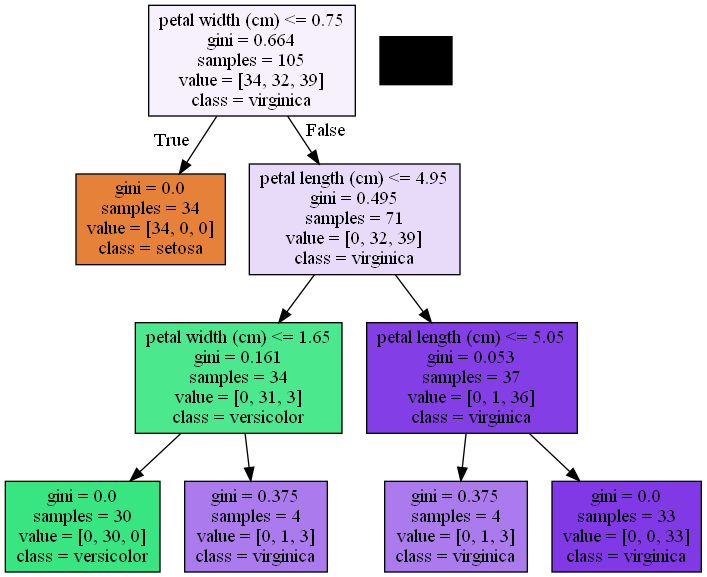




图 31 预剪枝后的iris数据集基于基尼指数的决策树

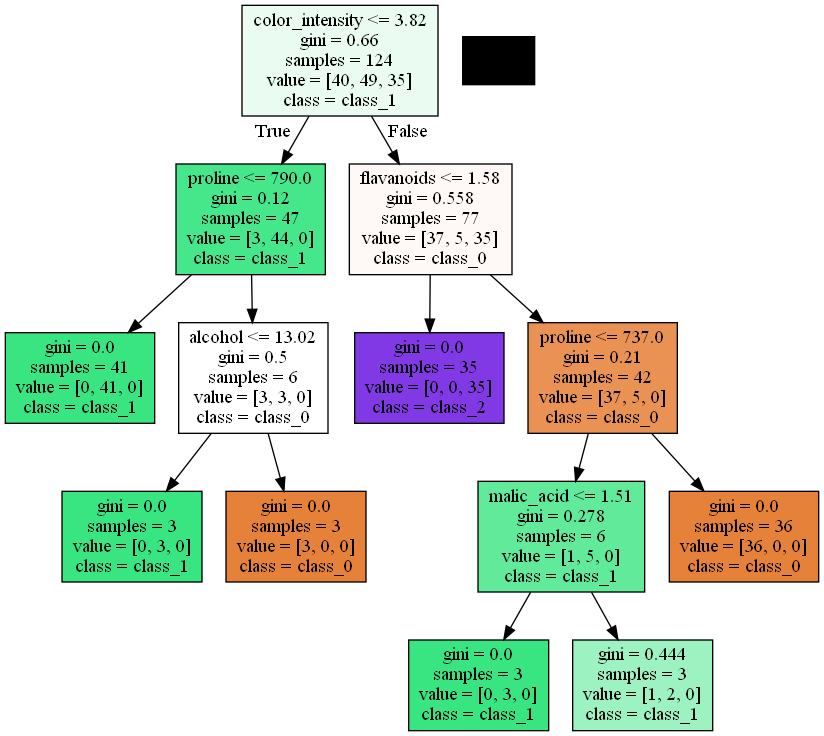




图 32 预剪枝后的wine数据集基于基尼指数的决策树

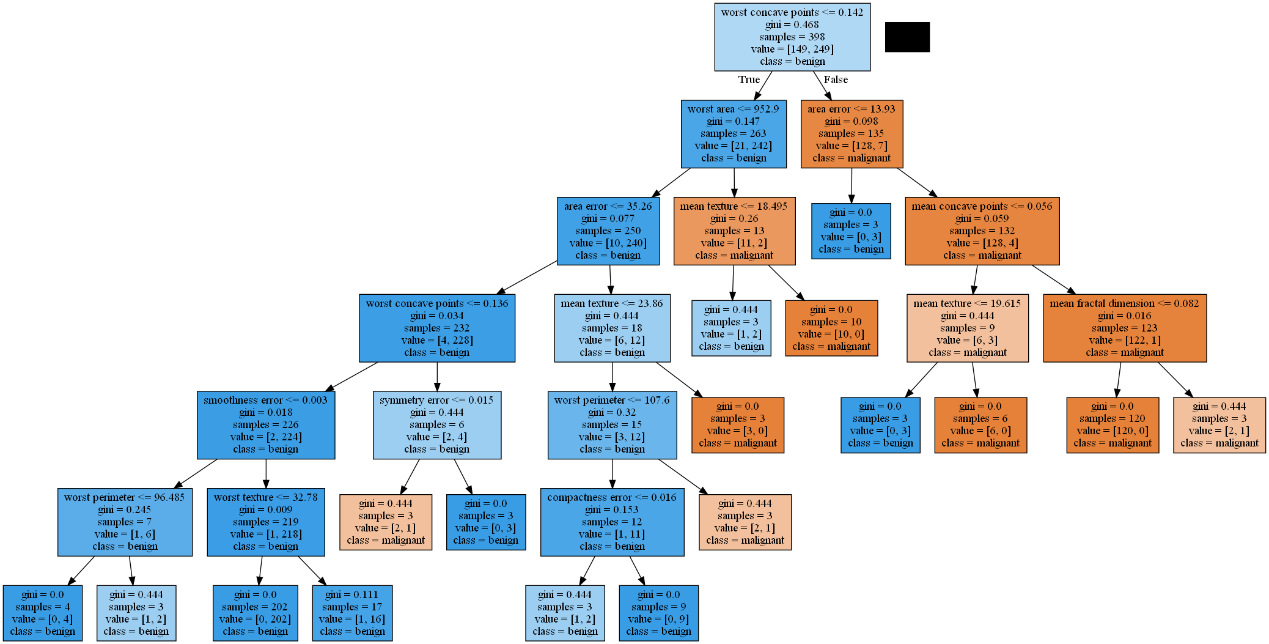




图 33 预剪枝后的bresat\_cancer数据集基于基尼指数的决策树

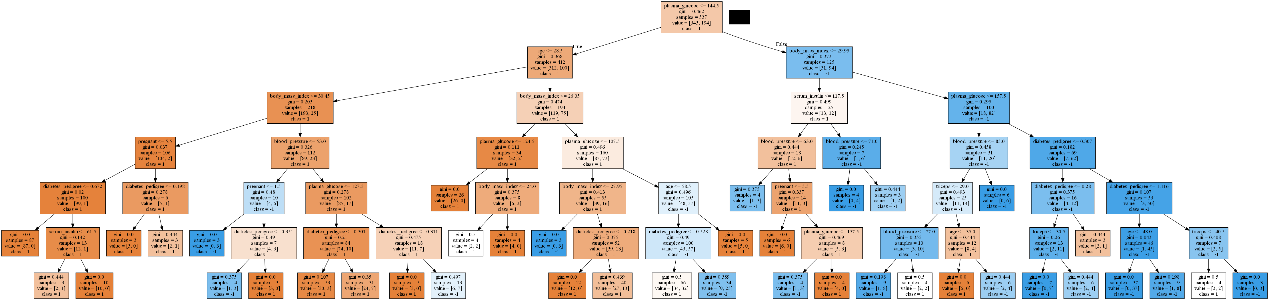




图 34 预剪枝后的diabetes数据集基于基尼指数的决策树

可以画出如下表格：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **正确率** | **信息增益** | **信息增益(预剪枝)** | **信息增益(后剪枝)** | **基尼指数(预剪枝)** | **基尼指数(未剪枝)** |
| **Iris** | 96.6% | 100% | 100% | 97.9% | 97.8% |
| **wine** | 97.2% | 94.4% | 94.4% | 96.2% | 94.4% |
| **breast\_cancer** | 91.2% | 86.8% | 91.2% | 94.2% | 91.2% |
| **diabetes** | 72.7% | 72.7% | 72.7% | 74.9% | 74.0% |

表格 1 决策树比较结果

对比可以发现，iris数据集因为数据量比较小的原因，拟合的效果比较好，最后的正确率也比较高。而breast\_cancer和diabetes数据集涉及的数据量和特征比较多，因此，训练要消耗比较长的时间，并且，数据量的增大也导致决策树过拟合，导致树的规模过大，测试的效果也就不理想。比如diabetes数据集，在未剪枝的情况下树的规模达到11层，并且很多节点包含的样本太少，出现明显的过拟合情况。如果限制节点的大小和树的深度，就可以提高准确率，训练速度也会高很多。

预剪枝对于决策树通常会导致欠拟合，从diabetes数据集可以看出，未剪枝和后剪枝时候的决策树是十分庞大的，而预剪枝只有两个节点，说明欠拟合程度比较严重，泛化能力可能不是很好。而后剪枝效果比较好，适合大多数情况下使用。

**检查情况：**

1. *from sklearn.utils.multiclass import type\_of\_target* 这行代码的作用。

这行代码的目的是判定一个数据集中的某个特征是离散值还是连续值，由于在我选择的数据集中可能存在连续特征也可能存在离散特征，并且，决策树的执行需要这行代码来判断某一个特征的离散情况来决定分裂点的分裂方法。因此，需要引入这个函数来判定。

1. 简要概括决策树画图函数的执行流程。

画图函数主要是利用已经建立好的决策树，在建立好的节点中保存了该节点的特征名称、深度、叶子个数、左右子树。当画图函数运行的过程中，递归判断当前节点是否是叶子节点，如果不是叶子节点，将进行深度遍历，进入左子树。如果是叶子节点，将根据该节点的深度来设置相对于根节点的偏移量。比方说如果相对于根节点的深度为4，将相对于根节点的y偏移量设置为0.2\*4，如果根节点的叶子个数为10，将其x偏移量设置为0.1\*10，画出该节点。直到将全部节点遍历完。

### (5)代码结构，核心代码简要分析

本次实验重要代码在于创建决策树的流程，将根据决策树的流程图来进行代码的编写：其中，重要的点在于选择最佳特征函数，要输入当前节点的训练集、测试集。并且，判断连续值的时候，需要用type\_of\_target函数来判断该特征是否是连续值。并且，需要重置该节点的叶子节点个数、深度、是否是叶子节点。

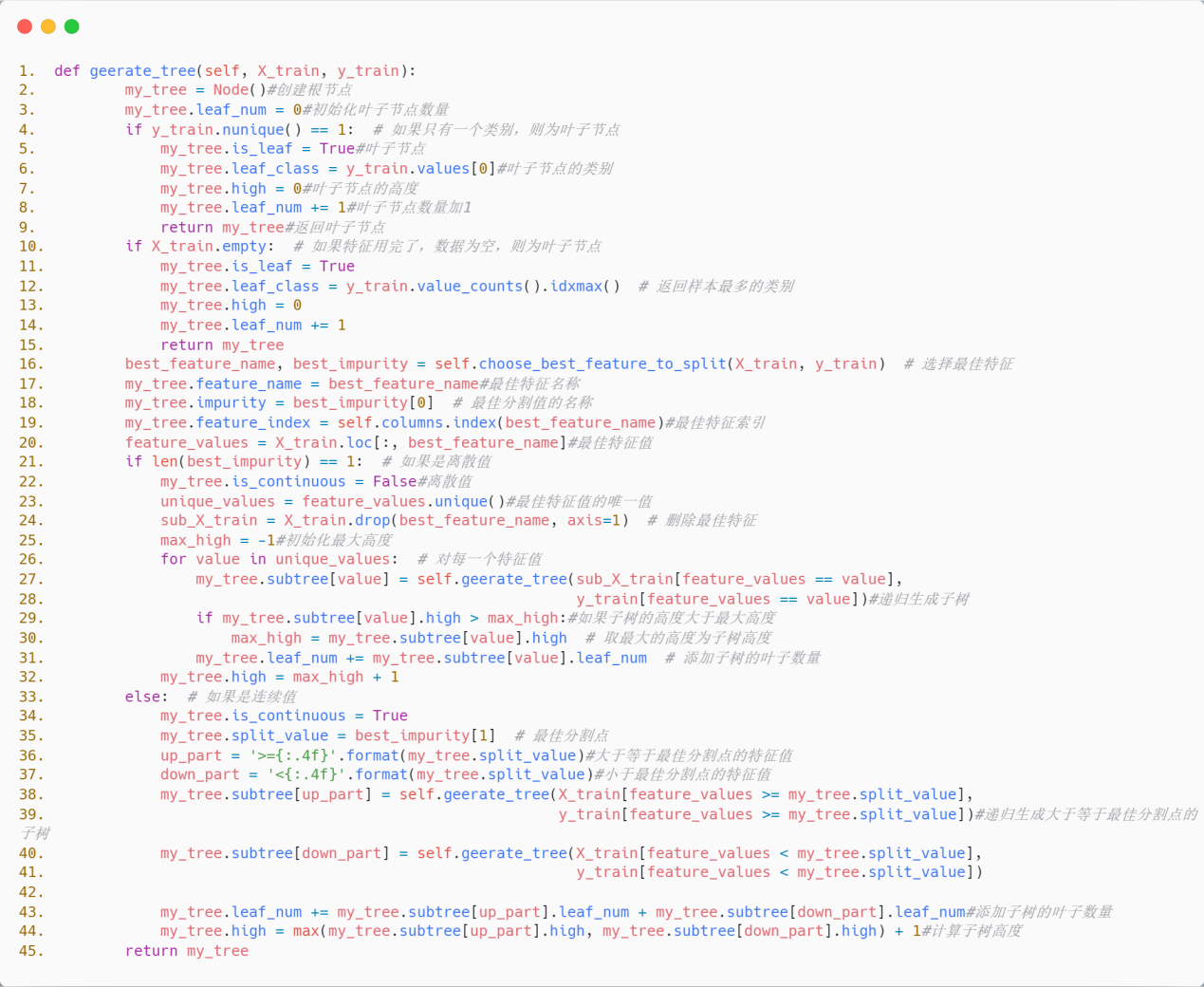


图 35 决策树主要流程代码

如何确定最佳增益点也是代码的关键步骤，对连续值，需要进行分割取中值，对每个中值计算信息增益。取出其中最大的一个信息增益点，就是数据集的最佳划分点，记录下这个划分点的特征名称，返回特征名称与最佳划分点。



图 36 决策树判断连续值的最佳划分点代码

### (6)本次实验解决的主要问题、在理论学习与动手编程上的主要收获

1.每次实验主要依靠西瓜书中的公式，实现了一个决策树，并对4个UCI数据集进行了测试。从底层类开始一步步实现了决策树类，节点类，特征的选择以及递归调用建立子树。了解了决策树这个经典机器学习算法的原理，提高了自身的编程能力。

2.遇到的主要问题集中在对于连续特征的处理，如何选取最佳划分点成为了关键。经过资料的查阅，明白了划分点依赖于该连续特征下的所有特征值的中点，遍历求出信息增益值最大的点，根据这点来划分连续特征。

3.了解了过拟合问题对于算法的影响。在diabetes数据集中，过拟合的状况导致决策树规模过于庞大，因此，需要进行预剪枝或者后剪枝。

4.由于时间的关系，没有自己实现对率回归的代码。但是对率回归本质上和信息增益是一样的，只不过少了求最大值的过程，而是直接通过对率回归求出最佳划分点，根据最佳划分点直接划分节点。

### (7)编码及内容撰写中的参考来源

[1] 机器学习 周志华P73-P92

[2] https://blog.csdn.net/quinn1994/article/details/80083933西瓜书学习（一）—决策树（上）

[3] https://blog.csdn.net/w417950004/article/details/77600913【西瓜书笔记二】决策树

## 第2题

### 编程题目理解

题目要求利用队列的数据结构，将原本的递归建树改造为非递归的深度搜索。由于是深度搜索，所以应该采用的是栈这一先进后出的数据结构。并且，利用书中的西瓜数据集对树进行测试。由于是采用信息增益的方式，可以对4.1的代码稍加修改，将递归改造成栈建树，就可以完成。

### 决策树深度遍历算法原理阐述

由于是要求栈结构建树，因此，需要一个list作为栈数据结构，而栈是一种先进后出的结构。因此，可以用来模拟树的深度遍历。用栈建树，每次取栈中最后一个元素作为当前节点，判断当前节点的状态，如果当前节点的高度已经超过最大高度，则跳过该节点。否则取该节点的最佳划分特征进行划分。划分后，再根据最佳划分特征，修改节点的数据集和标签集，遍历当前节点的最佳特征的所有特征值，将这些特征值作为节点判断。

因此，和递归算法类似，都是修改当前节点的训练集、标签集，找出最佳的划分特征进行划分出来的子节点加入栈，每次弹出最后一个元素作为当前节点继续划分，直到满足终止条件。

### 决策树深度遍历算法设计思路

非递归栈算法的设计思路如下：

1. 设计一个栈结构node\_queue;
2. 如果栈结构非空，进行如下循环；
3. 取出最后一个元素作为curnode，如果curnode的高度大于限定的最大高度，则continue；
4. 否则，取出curnode的最佳特征的唯一值，对这些值进行如下循环；
5. 创建子节点nextnode，该节点的数据集为curnode的训练集对应的最佳特征的特征值的子集。标签集为curnode的标签集对应的最佳特征的特征值的子集。
6. 如果nextnode的数据集、标签集、属性集为空，则将该节点标记为叶子节点，设置nextnode的标签为nextnode数据集中最多的样本的标签。
7. 如果不为空，根据nextnode的数据集选出最佳特征，赋值对应的属性，将curnode的subtree添加nextnode，node\_queue添加nextnode，返回4.
8. 在步骤4循环结束后，退出循环。
9. 计算出当前决策树的各个节点包含的叶子数量，高度，方便接下来绘图。
10. 画出决策树，计算验证集正确率。

### 实验数据、实验流程、测试结果分析以及检查情况

实验数据采用表4.2的西瓜数据集。

为了显示出深度遍历，将在每次入栈、出栈的时候输出对应的节点标签名，进行检查。并且，最后的验证集将会输入到决策树中，跟原本的验证集的标签集进行对比，计算精度。具体步骤和递归建立树的步骤一样，需要将递归方法改为遍历方法。

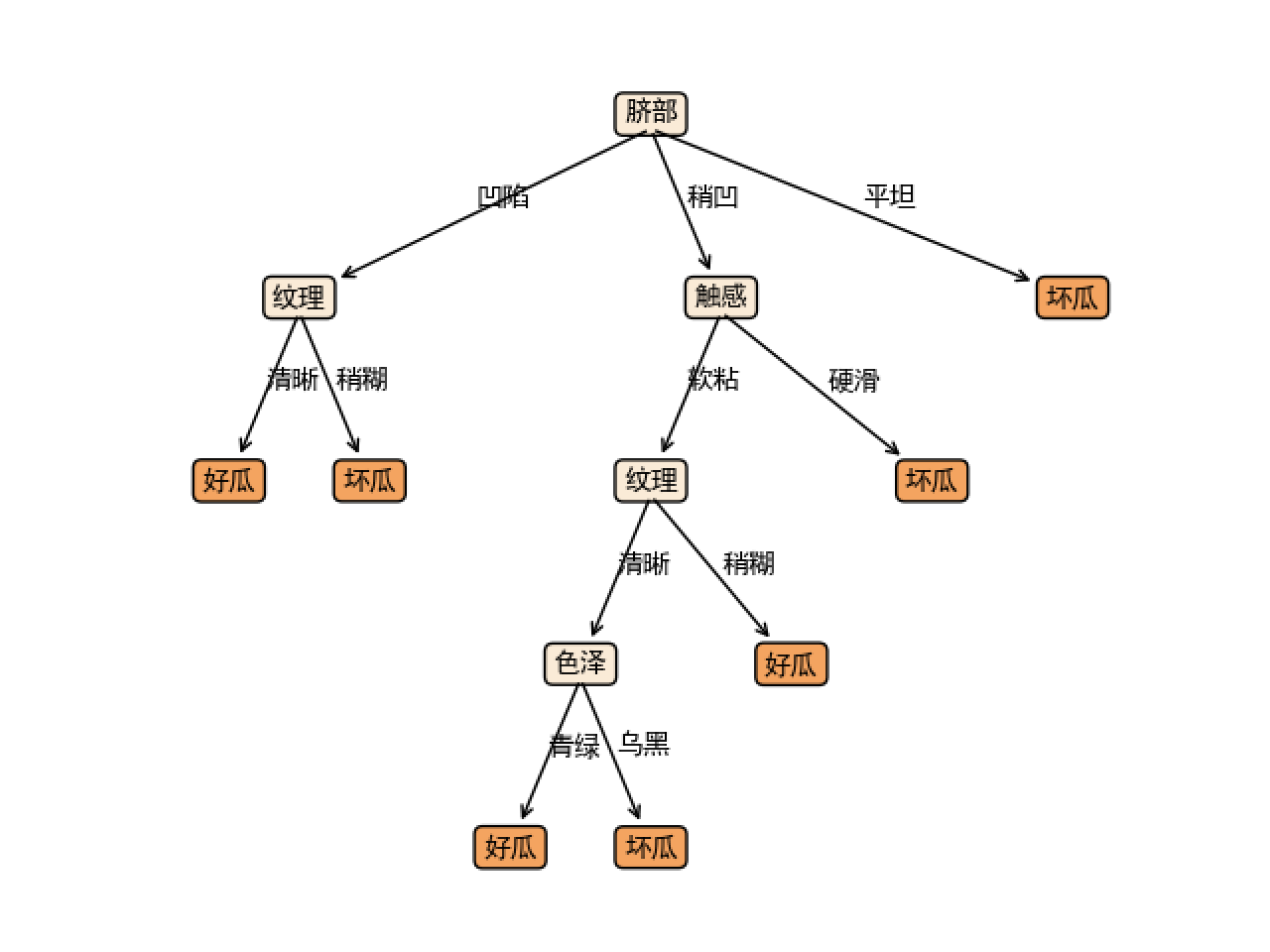


图 37 利用递归生成的基于信息增益的未剪枝决策树

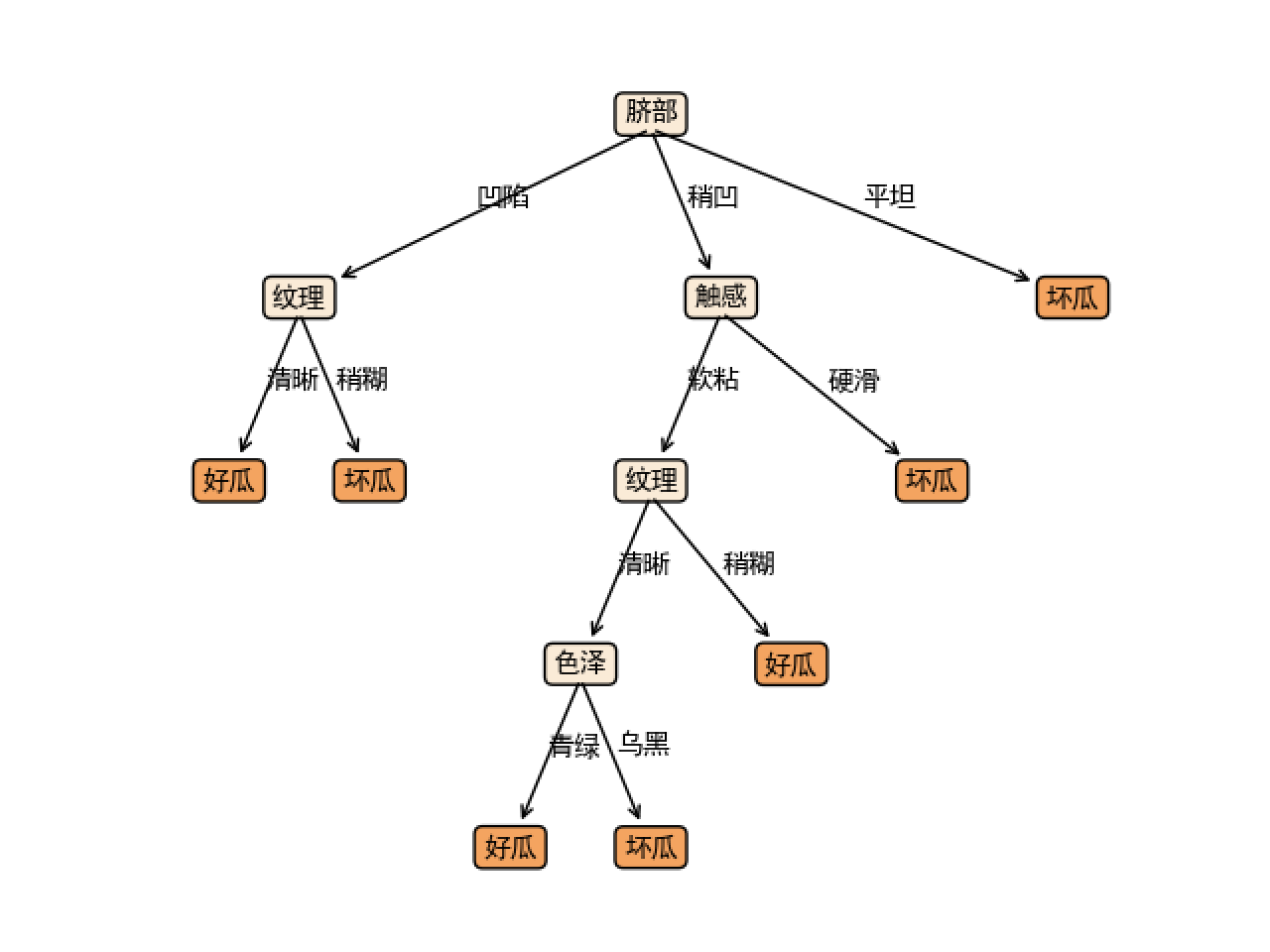


图 38 未限制高度的栈结构决策树

可以看出，未限制高度时当前决策树和遍历生成的决策树结构一致。

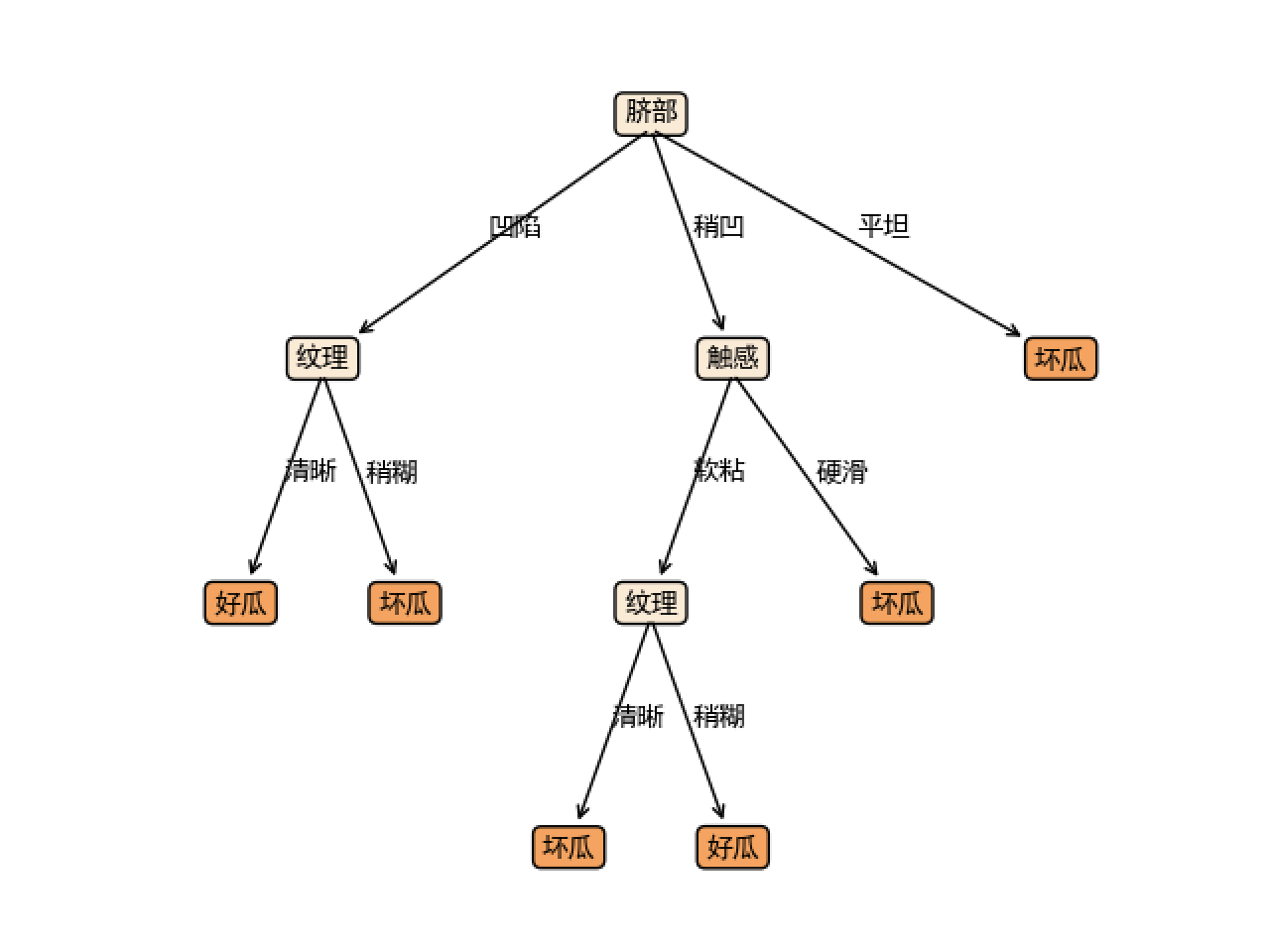


图 39 限制最大高度为2的决策树

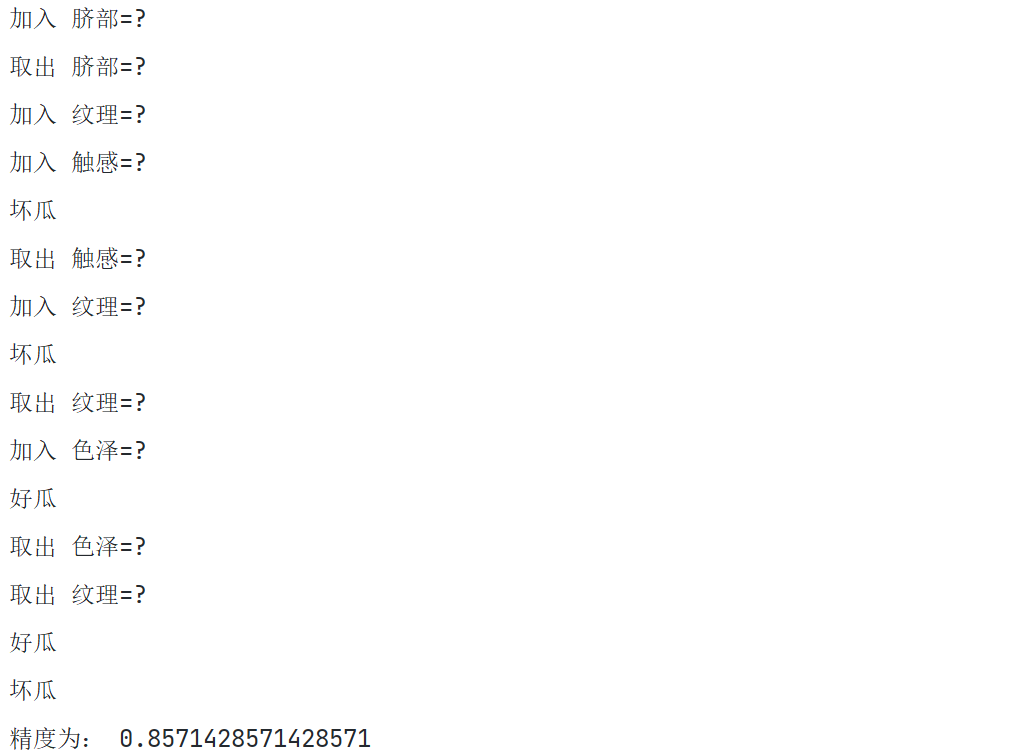


图 40 遍历节点顺序以及精度

从图 40可以看出，利用栈结构生成的深度遍历决策树确实是深度遍历，且在限制最大高度的情况下，精度为85.7%。

### 代码结构，核心代码简要分析

核心代码如下，为深度遍历循环，该代码首先定义一个栈，接着将树节点压入栈。接着，取当前元素为栈顶元素，对栈顶元素的每一个特征进行判定，选择最佳特征作为子节点，并压入栈。循环遍历栈顶元素，直到栈为空为止。



图 41 决策树深度遍历循环代码

### 本次实验解决的主要问题、在理论学习与动手编程上的主要收获

这次实验主要解决了基于栈结构的非递归决策树的建立问题。遇到的问题主要在于节点的数据集赋值需要特别注意为原来数据集的子集，并且需要特别注意节点的高度和叶子节点的高度问题，不然会报错空数值。主要收获在于更加了解了决策树算法的原理，可以自己实现底层的决策树代码，对信息增益公式了解更加深刻。对于栈结构的深度搜索算法的编写更加熟练，可以轻松完成编程任务。

发现的主要问题在于第一个节点的最佳特征的选择上面，存在两个特征的信息增益值是相同的，如果选了第一个特征，那么画出来的图像和书上面会很不同，并且精度会降低很多。如果选择第二个特征，那么精度会提升至八十多。主要原因在于数据集太小，西瓜数据集只包含十七个数据，不足以支撑整个决策树，欠拟合。

### 编码及内容撰写中的参考来源

[1]https://blog.csdn.net/qq\_37691909/article/details/85235472西瓜书课后题——第四章（决策树）

[2] 机器学习 周志华P73-P92

## 第3题

### 编程题目理解

题目要求利用队列结构实现非递归的广度优先搜索的未剪枝决策树算法。和4.2题目相似，只需将选择节点的顺序从先进后出改为先进先出，即可完成对应结构的实现。

### 非递归队列广度优先算法原理阐述

由于是要求队列结构建树，因此，需要一个list作为队列数据结构，而栈是一种先进先出的结构。因此，可以用来模拟树的广度遍历。用队列建树，每次取队列中最后一个元素作为当前节点，判断当前节点的状态，如果当前节点的高度已经超过最大高度，则跳过该节点。否则取该节点的最佳划分特征进行划分。划分后，再根据最佳划分特征，修改节点的数据集和标签集，遍历当前节点的最佳特征的所有特征值，将这些特征值作为节点判断。

因此，和递归算法类似，都是修改当前节点的训练集、标签集，找出最佳的划分特征进行划分出来的子节点加入队列，每次弹出第一个元素作为当前节点继续划分，直到满足终止条件。

### 非递归队列广度优先算法设计思路

非递归队列广度优先算法的设计思路如下：

1. 设计一个队列结构node\_queue;
2. 如果栈结构非空，进行如下循环；
3. 取出最后一个元素作为curnode，如果curnode的高度大于限定的最大高度，则continue；
4. 否则，取出curnode的最佳特征的唯一值，对这些值进行如下循环；
5. 创建子节点nextnode，该节点的数据集为curnode的训练集对应的最佳特征的特征值的子集。标签集为curnode的标签集对应的最佳特征的特征值的子集。
6. 如果nextnode的数据集、标签集、属性集为空，则将该节点标记为叶子节点，设置nextnode的标签为nextnode数据集中最多的样本的标签。
7. 如果不为空，根据nextnode的数据集选出最佳特征，赋值对应的属性，将curnode的subtree添加nextnode，node\_queue添加nextnode，返回4.
8. 4循环结束后，退出循环。
9. 计算出当前决策树的各个节点包含的叶子数量，高度，方便接下来绘图。
10. 画出决策树，计算验证集正确率。

### 实验数据、实验流程、测试结果分析以及检查情况

实验数据采用表4.2的西瓜数据集。为了显示出广度遍历，将在每次入队，出队的时候输出对应的节点标签名，进行检查。并且，最后的验证集将会输入到决策树中，跟原本的验证集的标签集进行对比，计算精度。

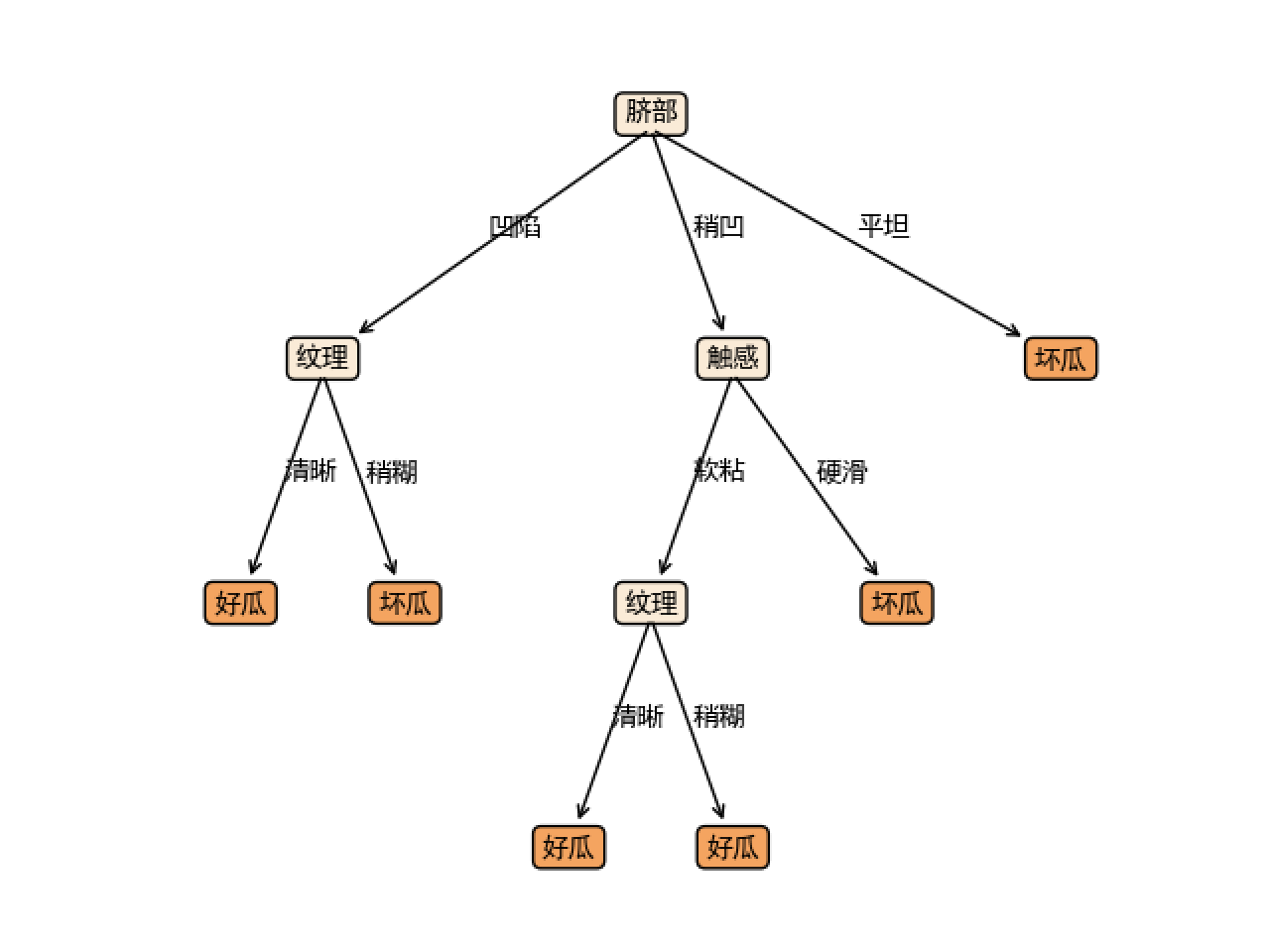


图 42 递归生成的决策树

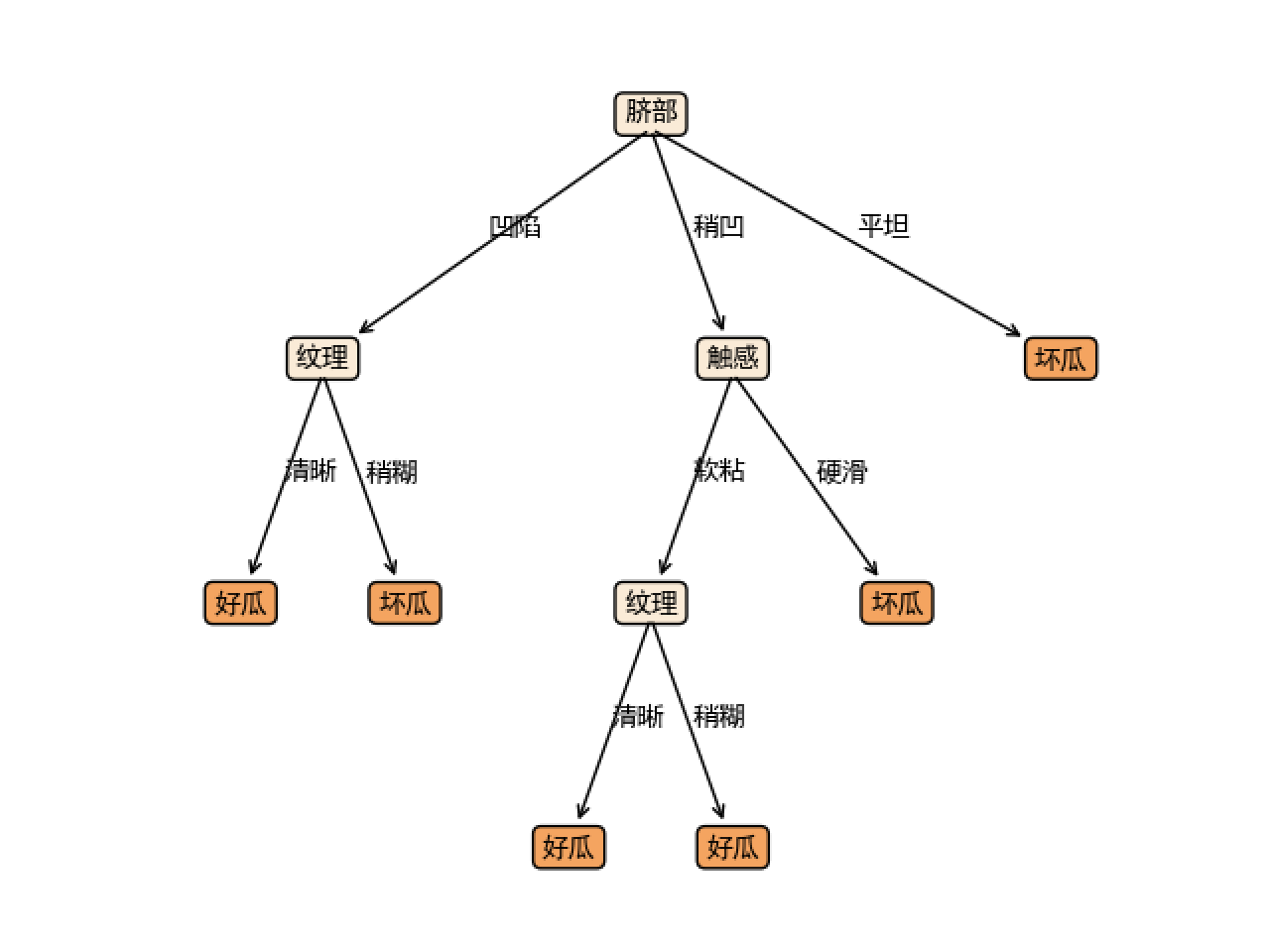


图 43 队列生成的决策树

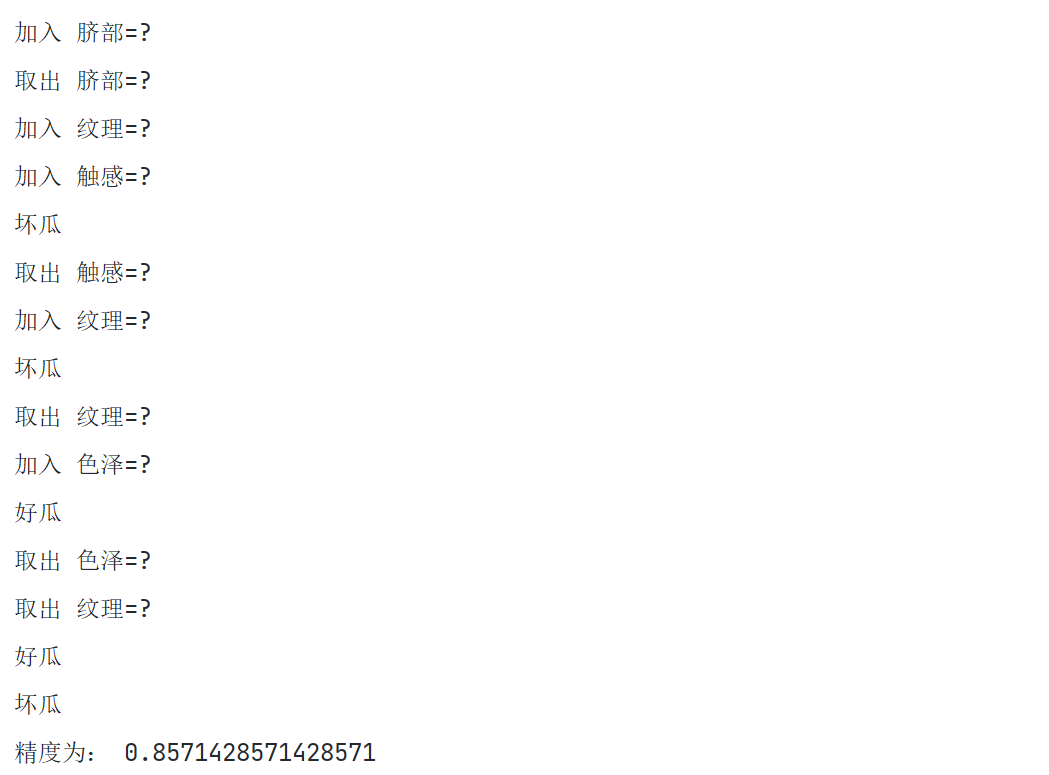


图 44 深度遍历对应的输出值以及精度

经过输出值的检查，可以发现确实是广度优先遍历，并且建树的结构和递归建树相同。精度和深度遍历一样，都为85.7%。

### 代码结构，核心代码简要分析

核心代码和深度遍历相同，唯一的不同在于修改了如下代码：

curnode = node\_queue.pop(0) # 出队，取当前节点为队顶元素

修改取元素的顺序，改为取队首元素。这样做的目的是将深度遍历转化为广度遍历，也就是用队列的方式实现取元素。

### 本次实验解决的主要问题、在理论学习与动手编程上的主要收获

主要收获与问题已经在4.2中解决，唯一遇到的问题就是python List数据结构如何取队首元素，经过查阅已经解决该问题，list.pop(index)函数提供了index，可以弹出指定的索引元素。

### 编码及内容撰写中的参考来源

[1]https://blog.csdn.net/qq\_37691909/article/details/85235472西瓜书课后题——第四章（决策树）

[2] 机器学习 周志华P73-P92

## 第五章编程题完成清单

## 第1题

### 编程题目理解

题目中要求我们利用书上面的伪代码以及西瓜数据集3.0，实现一个标准BP算法，其中，只包含一个隐层结构，并且为全连接前馈网络。根据题目要求，需要设计1个输入层，1个隐层，1个输出层，其中，节点的个数任选，再利用书上面的步骤，对连接权和阈值进行更新。最后，利用测试集测试精度。

### 标准BP算法原理阐述

标准BP算法是一种经典的算法，把学习过程分为信号的正向传播和反向传播。该模型的训练经常采用误差反向传播算法，来更新神经元的连接权和阈值。

正向传播的时候，输入样本从输入层进入网络，经过隐层到输出层输出。而反向传播的过程，就是计算出网络的均方误差后，求均方误差的累计平均值最小值的过程。其中，均方误差公式为

我们的目标，就是将优化上面的，直到收敛到一定精度。其中，标准BP算法在一次迭代的时候，分别带入不同的样本进行训练，因此，需要训练的连接权和阈值次数会非常多，其中，可能会有不同的样本相互抵消，导致训练权和阈值没有变化。这就演化出了累积BP算法。

具体的迭代算法，可以根据书上面公式推导得出。其中，主要的公式如下：

上面的公式中，最主要的是、分别代表隐层的输入，输出层的输入，代表输出层的最后输出，给出函数为sigmoid函数。输入层到隐层连接权、隐层到输出层连接权分别为、，阈值分别为、。这四个参数是需要主要更新的参数，而是用来简化计算的参数，也就是梯度下降中的中间量。根据这些公式，可以写出算法的主要设计思路。

### 标准BP算法设计思路

根据上面的原理，可以设计出算法的思路如下：

1. 导入西瓜数据集，处理数据，将类别转化为数字编码、将特征转化为数字编码。
2. 设计类Data，用来初始化参数：迭代次数、训练集、测试集的划分、连接权和阈值。
3. 进入标准BP算法的训练步骤。
4. 随机初始化两个连接权矩阵和阈值矩阵，学习率设定为0.1.
5. 当迭代次数小于指定的迭代次数，进行如下循环：
6. 对训练集的每一个样本，进行如下循环：
7. 计算，计算输出层的预估输出，计算出此时的均方误差，添加到中，根据正向传播的参数，更新反向传播参数：连接权、阈值、梯度下降用到的两个中间量。
8. 如果样本没有用完，返回第六步。
9. 如果迭代次数小于指定的迭代次数，返回第五步。
10. 将训练完成的神经网络参数记录，将迭代次数作为横坐标，作为纵坐标，画出收敛图。
11. 将测试集输入至神经网络中，如果最后的预测值大于0.5，表示好瓜，小于0.5表示坏瓜。将其与真正的测试结果进行对比，计算精度。

### 实验数据、实验流程、测试结果分析以及检查情况

实验流程如上面算法设计思路一般进行。其中，各类超参数设置不同会导致不同的结果，一般来说，隐层节点设置为7个为最佳。在这里，输入层节点数设置为8，也就是西瓜数据集3.0的特征数量，而输出层指定为1，隐层指定为9，也就是特征数量+1。最后的测试结果如下：

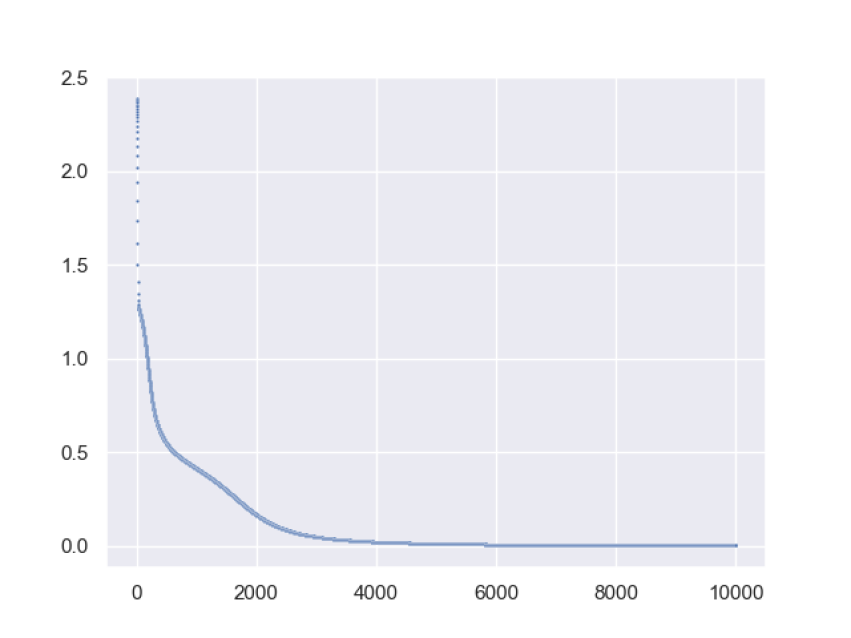


图 45 标准BP算法迭代10000次累计误差变化



图 46 标准BP算法精确率

**数据分析及结果分析：**

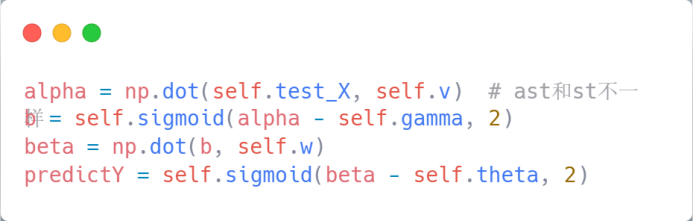
由于西瓜数据集的样本数量太少，对其训练出的曲线看不出来是过拟合还是欠拟合。但是标准BP算法一般会导致过拟合，所以会产生早停和正则两种修正方法。准确率较低的原因是对于训练集的拟合效果太好，也就是过拟合，并且样本数量太少，无法真正体现出神经网络的强大表达能力。并且，测试集的数据也不是特别多，随机性比较大，该结果是正常表现。

**检查情况：**

**1.精准率较低的原因？**

由于西瓜数据集的样本只有17个，训练集可以分到11个样本，这就导致训练集的数据过少，极其容易过拟合，并且，参数容易陷入局部最优。因此，导致测试集的测试效果不佳，如果需要改进，可以尝试添加数据的个数，或者添加正则来避免过拟合。

**2.解释下列代码中各个变量的意思。**



这个代码来自predict函数，也就是根据训练集训练的模型来对测试集进行预测。其中，alpha表示输入的测试集样本乘以输入层到隐藏层的连接边权值，也就是输入层的输出值。b表示隐藏层经过激活函数后的输出值，也就是.而beta指隐藏层输出到输出层的输出值，也就是隐藏层的b乘以隐藏层到输出层的连接权权值。predictY指的是最后的输出层的输出值，用表示，也就是beta减去输出层的阈值，再经过sigmoid函数激活，就是最后的结果。

### 代码结构，核心代码简要分析

**代码结构：**

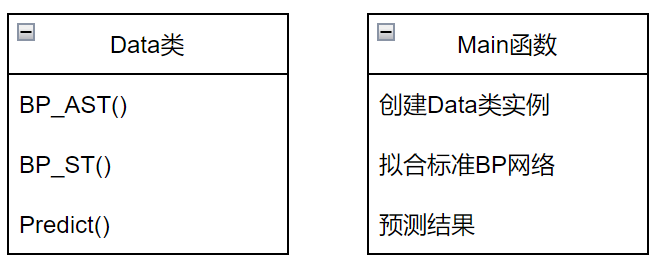


图 47 标准BP算法代码结构



图 48 标准BP算法核心代码及注释

在迭代的时候，利用书中的公式计算出每次的需要的参数，也就是以及梯度下降算法需要的方便计算。最后，计算出，并存在列表里面，方便最后绘制图表使用。

### 本次实验解决的主要问题、在理论学习与动手编程上的主要收获

本次实验主要遇到的问题是对于numpy库的使用不够熟练，矩阵的乘除需要查阅相关资料才能实现。并且，对于矩阵的维度不是很熟悉，需要反复调试。对于公式的实现能力还有待加强。

本次实验还让我深入了解了标准BP算法的实现，可以从零开始实现一个标准BP算法，了解其中的原理，清楚了梯度下降算法的具体推导步骤和实现，清楚了阈值和对应激活函数的关系，以及神经网络强大的表达能力，过拟合该如何修正等等。了解了BP算法的局限性，例如梯度消失等等问题。

### 编码及内容撰写中的参考来源

[1]https://blog.csdn.net/weixin\_42398658/article/details/83929474，深度学习 --- BP算法详解（流程图、BP主要功能、BP算法的局限性）

[2]机器学习 周志华 P101-P105

## 第2题

### 编程题目理解

累积BP算法和标准BP算法差不多的算法，题目要求我们利用累积BP算法对西瓜数据集3.0进行神经网络的训练，其中，节点的个数任选，再利用书上面的步骤，对连接权和阈值进行更新。最后，利用测试集测试精度。

### 累积BP算法原理阐述

和标准BP算法原理相似，都是对神经网络的反向传播过程中的参数进行训练，将其中的均方误差最小化。但是和标准BP算法不同的是，累积BP算法输入的不再是一个个样本，而是对训练集扫描一次后，将训练集整个输入进去。因此，标准BP算法的参数更新效率会十分频繁，时间复杂度达到了，而累积BP算法的参数更新频率少，时间复杂度为。累积BP算法直接对累积误差最小化，标准BP算法可能会因为不同样例的更新，导致参数相互“抵消”的情况。

但是在很多情况下，累积BP算法的累计误差下降到一定程度后，进一步的下降会非常慢，这时候标准BP算法会更快获得更好的解，在训练集非常大的时候尤为明显。

原理和标准BP算法相同：

累积误差公式为：

我们的目标，就是将优化上面的，直到收敛到一定精度。累积BP算法的更新就更为简单，具体的迭代算法，可以根据书上面公式推导得出。其中，主要的公式如下：

上面的公式中，最主要的是、分别代表隐层的输入，输出层的输入，代表输出层的最后输出，给出函数为sigmoid函数。输入层到隐层连接权、隐层到输出层连接权分别为、，阈值分别为、。这四个参数是需要主要更新的参数，而是用来简化计算的参数，也就是梯度下降中的中间量。根据这些公式，可以写出算法的主要设计思路。

### 累积BP算法设计思路

和标准BP算法差不多的设计思路，累积BP算法差别在于设计每一代迭代的时候，不是对参数进行单个样本的更新，而是让整个训练集参加参数的更新。因此，可以给出设计思路如下：

1. 导入西瓜数据集，处理数据，将类别转化为数字编码、将特征转化为数字编码。
2. 设计类Data，用来初始化参数：迭代次数、训练集、测试集的划分、连接权和阈值。
3. 进入累积BP算法的训练步骤。
4. 随机初始化两个连接权矩阵和阈值矩阵，学习率设定为0.1.
5. 当迭代次数小于指定的迭代次数，进行如下循环：
6. 计算α、β，计算输出层的预估输出，计算出此时的均方误差E，根据正向传播的参数，更新反向传播参数：连接权、阈值、梯度下降用到的两个中间量g 、e。
7. 如果迭代次数小于指定的迭代次数，返回第五步。
8. 将训练好的θ、γ进行降维，从(m、n)维降至(m、1)维，其中，具体取每一列的平均值作为结果。这么做的原因是与标准BP算法同一维度，避免后续预测算法出错。
9. 将训练完成的神经网络参数记录，将迭代次数作为横坐标，E作为纵坐标，画出收敛图。
10. 将测试集输入至神经网络中，如果最后的预测值¯y大于0.5，表示好瓜，小于0.5表示坏瓜。将其与真正的测试结果进行对比，计算精度。

可以看出，具体的设计思路和标准BP算法没什么太大的差别，主要在于X、Y矩阵的维度，以及各类参数的维度从(m,1)维上升到(m,n)维。

### 实验数据、实验流程、测试结果分析以及检查情况

实验流程如上面算法设计思路一般进行。其中，各类超参数设置不同会导致不同的结果，一般来说，隐层节点设置为7个为最佳。在这里，输入层节点数设置为8，也就是西瓜数据集3.0的特征数量，而输出层指定为1，隐层指定为9，也就是特征数量+1。最后的测试结果如下：

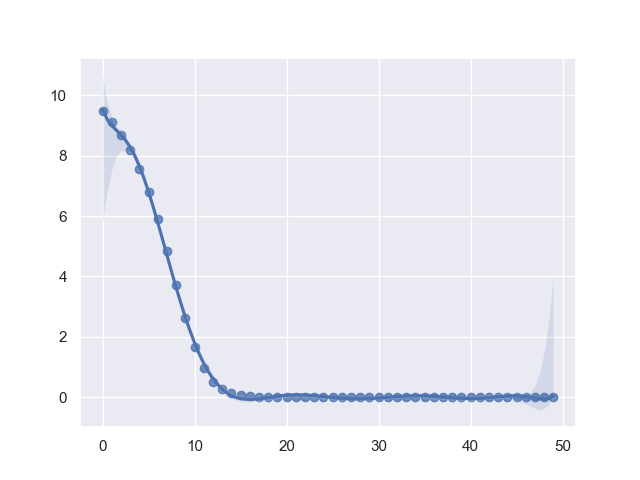


图 49 累积BP算法收敛速度图



图 50 累积BP算法精确率

**数据分析和结果评价：**

可以看出，累积BP算法的收敛速度十分快，在20代的时候基本上已经收敛，再接着提升的难度就会很高。在准确率方面，累积BP算法的准确率虽然看起来比较高，但其中的随机性非常大，经计算，波动区间在40%~90%，这是因为西瓜数据集样本太少，导致随机数的产生对测试集影响因子加大。

**和标准BP算法的对比：**

和标准BP算法的对比，可以由如下图表给出：

表格 2 累计误差随收敛次数变化表

不难看出，累积BP算法的收敛速度非常快，相比标准BP算法，提升了至少十倍效率。并且，能避免个别异常样本带来的过拟合的问题。但是需要的样本量来说，累积BP算法需要比较多的样本才能更好地发挥其速率优势，并且，忽略了个体更注重于整体，对于模型的泛化能力来讲，累积BP算法对于边界的把握能力不一定比标准BP算法强。

就准确率而言，二者的准确率波动过大，不适合进行比较，除非有更大数据量的样本。

### 代码结构，核心代码简要分析

和标准BP算法的代码类似，区别在于修改了每次迭代的时候输入得值，从单一的样本改为整个数据集的输入。并且，总体误差改为总体的预测值减去实际值的平方除以2的加和。最后，把误差记录到列表里面，方便后面绘制图表。



图 51 累积BP算法关键代码及注释

### 本次实验解决的主要问题、在理论学习与动手编程上的主要收获

本次实验还让我深入了解了累积BP算法的实现，可以从零开始实现一个累积BP算法，了解其中的原理，清楚了梯度下降算法的具体推导步骤和实现，清楚了阈值和对应激活函数的关系，以及神经网络强大的表达能力，过拟合该如何修正等等。了解了BP算法的局限性，例如梯度消失等等问题。

了解了标准BP算法和累积BP算法的不同，对比了不同迭代次数对于累积误差的影响，发现累积BP算法的收敛速度较标准BP算法要快十倍不止。

最简单的BP算法的实现，对于日后学习更深一步的神经网络算法或者深度学习算法有铺垫作用。相信在自己实现了BP算法之后，对于神经网络算法的理解会更为深刻。

### 编码及内容撰写中的参考来源

[1]https://blog.csdn.net/weixin\_42398658/article/details/83929474，深度学习 --- BP算法详解（流程图、BP主要功能、BP算法的局限性）

[2]机器学习 周志华 P101-P105

## 第六章编程题完成清单

## 第1题

### 编程题目理解

题目要求我们利用西瓜数据集3.0α，分别利用不同的核函数：线性核以及高斯核，训练一个支持向量机。

而对于支持向量机SVM，基本的想法就是求解能够正确划分训练数据集，并且使几何间隔最大的分离超平面，是一种二分类模型。本题需要通过两个核函数，将原本线性不可分的点投影至高维空间，转化为线性可分，来区分密度、含糖率不同的好瓜以及坏瓜。

### 支持向量机算法原理阐述

**硬间隔问题：**

SVM学习的基本原理是求解能够正确划分训练数据集并且几何间隔最大的分离超平面。如下图所示，即为分离超平面，对于线性可分的数据集来说，这样的超平面有无穷多个（即感知机），但是几何间隔最大的分离超平面却是唯一的。

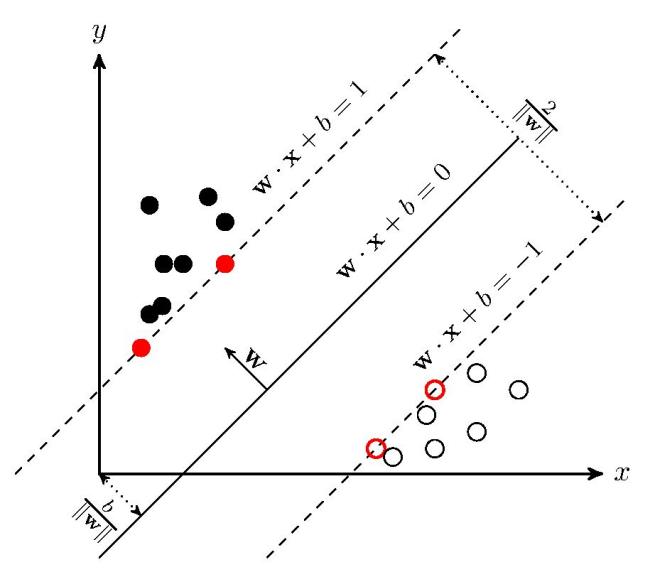


图 52 SVM示例图

因此，求解最大超平面分割问题就可以转化为约束优化问题：

此时，用有约束的拉格朗日乘子法求解上式：

其中，为拉格朗日乘子，并且。因此，需要求出带入中就可以求出最大值。

而求最大值，通常采用求偏导的方式。即对求偏导为0，可分别得出如下式：

带入到中，消去有：

该式即为对偶问题，我们需要的就是优化的值，在满足条件之后，用SMO算法求解凸二次规划问题，可以求出。

最终，可以求出模型:

对于SMO算法，这里太难了，个人没有看懂，但是大致明白这是求解上面函数的一个优化方法，因此，可以不必采用SMO算法对其进行实现。对于算法的实现，采取调库的方式进行。

**软间隔问题：**

由于训练集中常常出现错误的数据，因此，难免会出现分类错误的情况，SVM中给出软间隔的定义，目的是允许分类中出现一些错误，保证SVM的鲁棒性。

SVM把把分类错误的个数作为损失函数，有：

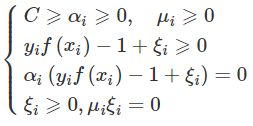
令作为分类正确，则计数为1，损失函数显见为 0-1损失，这是一个非凸的函数，在求解的过程中，存在很多的不足，通常在实际的使用中将0-1损失函数作为一个标准.

但是0-1损失函数不连续非凸，性质不好，西瓜书中，给了hinge损失函数作为替代。

与硬间隔一样，用拉格朗日乘子法转换，求偏导，得出对偶问题，经对比发现，与硬间隔的区别是对偶变量的约束不同：x前者，后者是。

根据之前推导的公式：

以及KKT条件：



**可以得出下面分析：**

1. 当时，可等于零，也可以大于0。说明：边界外的样本权重必为零，即对模型没有影响。
2. 当时，必为零，说明边界上的权重不为零，即模型建模主要取决于支持向量。
3. 当时，则必为零(根据公式：)，则取值任意。
4. ：错误分类。
5. ：样本落在最大间隔的内部。
6. 当时，必大于零(根据公式：)，进一步必有。即样本就在最大分类边界上。

**简单来说，软间隔就是给SVM一些缓冲空间，允许一些错误的分类。**其中，支持向量中错误分类为0，并且，很明显看出，C越大可以取得的值越大，即建模的时候越看重支持向量，越不能分错，因此，C的取值很重要。

**核函数：**

**核函数是为了解决线性不可分时候计算困难的情况。**在低维空间中，分类平面可能无法正确区别一些线性不可能情况，此时，利用非线性函数将低维样本映射至高维空间中，可转换为线性可分的情况。但是，高维空间的数据计算很困难，于是，出现了核函数来简化高维运算量**。**

将相近的点映射到一个空间，将远的点分到另一个空间，核函数的诀窍在于解决了映射后高维空间中样本距离的计算，但又不显式地展示出映射函数。

**线性核：**

让转换函数为，得到线性核函数，则两个向量的点积为：

**当不需要在特征空间进行运算时，可以用线性核函数。**如原始数据已经是高维的、可比较的，并且在输入空间线性可分。其实就是指样本点本来就是线性可分，不做任何映射。但是在西瓜数据集中，并不是线性可分的，这点可以从西瓜数据集的分布图可以看出，无法用线性函数区分好瓜坏瓜。于是，需要映射到高维空间。

**高斯核：**

此时，就出现了高斯核。高斯核是机器学习中比较普遍的核函数，将低维映射至无限维空间中，公式如下：

### 支持向量机算法设计思路

由于该算法中SMO算法没有看懂，个人实现比较困难，所以算法设计中采取调库的方式进行。Sklearn库中自带有SVM模型，可以帮助我们训练。因此，算法设计思路如下：

1. 数据统一格式化，转化为narray数据结构，将转换为，方便分类。
2. 初始化各类参数，设置迭代次数为1000次。
3. 调用sklearn中的SVM库，分别用线性核以及高斯核进行分类。
4. 打印出预测值、真实值、支持向量，进行对比。
5. 画图，分别画出正例、负例、支持向量、超分类平面，用不同样例表示。

### 实验数据、实验流程、测试结果分析以及检查情况

实验数据采用西瓜数据集3.0:

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **密度** | **含糖率** | **好瓜** |
| **1** | 0.697 | 0.46 | 1 |
| **2** | 0.774 | 0.376 | 1 |
| **3** | 0.634 | 0.264 | 1 |
| **4** | 0.608 | 0.318 | 1 |
| **5** | 0.556 | 0.215 | 1 |
| **6** | 0.403 | 0.237 | 1 |
| **7** | 0.481 | 0.149 | 1 |
| **8** | 0.437 | 0.211 | 1 |
| **9** | 0.666 | 0.091 | 0 |
| **10** | 0.243 | 0.267 | 0 |
| **11** | 0.245 | 0.057 | 0 |
| **12** | 0.343 | 0.099 | 0 |
| **13** | 0.639 | 0.161 | 0 |
| **14** | 0.657 | 0.198 | 0 |
| **15** | 0.36 | 0.37 | 0 |
| **16** | 0.593 | 0.042 | 0 |
| **17** | 0.719 | 0.103 | 0 |

表格 3 西瓜数据集3.0

测试结果：

**线性核：**

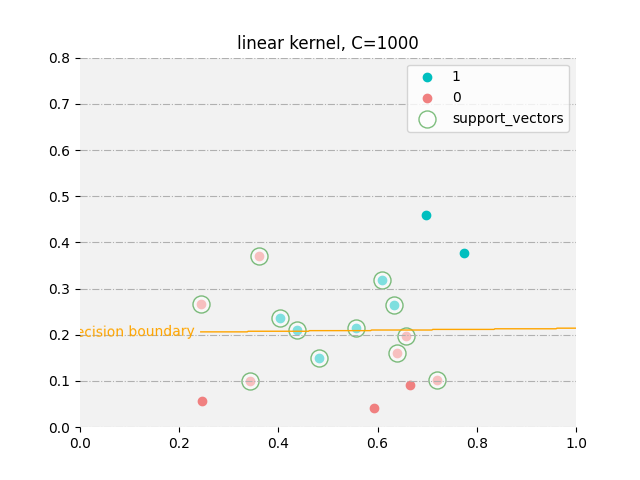


图 53 线性核拟合的SVM

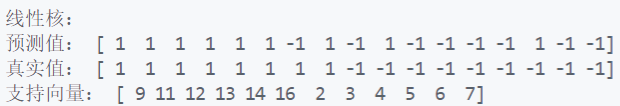


图 54 线性核结果

**高斯核：**

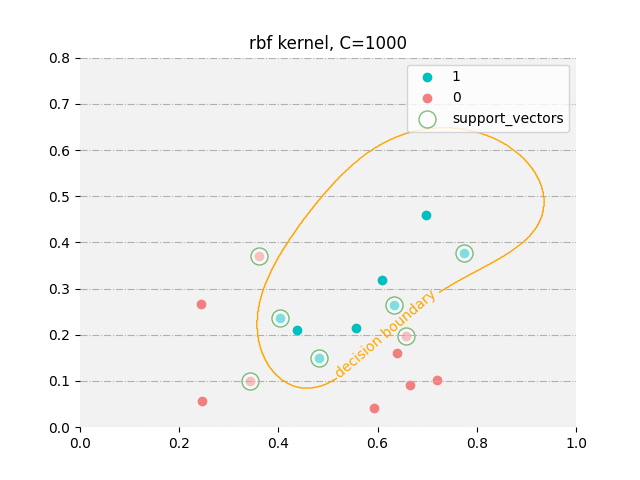


图 55 高斯核拟合的SVM

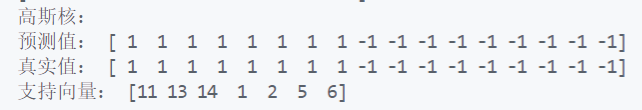


图 56 高斯核结果

**结果分析：**

由于西瓜数据集二维上不是线性可分，因此，用线性核无法完美区分所有点数据点，总会有一两个样本点分类错误。这也就是为什么线性核用的很少的原因。而高斯核划定的超平面可以恰好将样本点分隔开，实现了比较好的效果。

根据图片也可以看出，线性核划分的超平面有三四个点分类错误，高斯核就可以刚好区分开所有的样本点。当然，也可能是样本点数据量不够大的原因。

### 代码结构，核心代码简要分析

训练代码以及注释，其中，调用了sklearn中的svm函数进行训练。svm函数默认训练的模型是高斯核，因此，加入linear可以转变为线性核进行训练。画图函数首先画出了网格，再画出得出的模型中的正例以及负例。对其中的支持向量，额外画出其中的边框。对于范围，采用了等高线进行绘制。

****

图 57 训练代码以及注释

**画图代码以及注释：**

****

图 58 画图代码以及注释

### 本次实验解决的主要问题、在理论学习与动手编程上的主要收获

这次的模型来说，比前面三个实验要困难许多。通俗的原理并不是很难，困难的地方在于数学公式的推导，以及各类公式的使用，都对我的数学功底是一大考验。尤其在拉格朗日乘子法的推导，以及相应对偶问题的推导，都需要我查阅大量资料，才勉强看懂。遇到了很多数学公式推导上面的问题，对SMO算法没有完全掌握，对其的了解仅限于优化SVM的目标函数。并且，太过于困难，时间比较仓促，导致最后没有手写实现SVM，是比较遗憾的事情。

主要收获在于对于SVM的原理的理解，以及对转换问题思路有了更加清晰的认知，遇到一些比较困难的优化问题的时候，常常可以转化其为对偶问题。并且，对于核函数有了全新的认知，刚开始认为核函数只是将低维样本映射到高维空间的函数。查阅相关资料发现，这种认知是错误的，映射函数是单独的一类函数，核函数存在的意义在于简化高维空间的运算，而不是映射函数。

对于SVM的推导过程有了比较清楚的认识，相信在日后的学习过程中，还会接触到SVM相关的内容。

### 编码及内容撰写中的参考来源

[1]https://blog.csdn.net/mengjizhiyou/article/details/103437423，核函数(Kernel function)(举例说明，通俗易懂)

[2]https://blog.csdn.net/c406495762/article/details/78072313#2smo%E7%AE%97%E6%B3%95, Python3《机器学习实战》学习笔记（八）：支持向量机原理篇之手撕线性SVM