软件学院 机器学习 课程实验报告

实验题目：决策树		学号：201700301166
日期：2019/10/21	班级： 网安 17	姓名： 尹成林
Email：sdyinruichao@163.com		

实验目的：

实现 ID3 决策树，并在给定的数据集上进行 5 折交叉验证。并观测训所得到的决策树在训练集和测试集的准确率，从而判断该决策树是否存在过拟合。在此基础上实现预剪枝和后剪枝，并比较预剪枝树与后剪枝树在训练集和测试集上的准确率。

实验软件和硬件环境：

Win10

Pycharm

使用语言：python

使用的库：numpy、math、csv、random

实验原理和方法：

熵值计算公式

信息增益计算公式

连续值二分法

剪枝留出法

实验步骤：（不要求罗列完整源代码）

题目分析：

1.建立树

2.找到信息增益最大的特征

3.如何使用递归的方法去构建一颗决策树

4.如何通过决策树做出决策

先来写一个数据结构，作为树，存储我们的决策树，这里我选择了用一个 list 存储一个树节点来完成构建一棵树。

```
class treenode:
    def __init__(self, FenLei, FenLeiZhi, YeziPanDuan, node_id):
        self.FenLei = FenLei
        self.FenLeiZhi = FenLeiZhi
        self.YeziPanDuan = YeziPanDuan
        self.fenleiindex = [0, 1, 2]
        self.node_id = node_id
        if int(YeziPanDuan) == -1:
            print("产生中间节点{}，为第{}特征，分类值为{}".format(node_id, FenLei, FenLeiZhi))
        else:
            print("产生叶子节点{}，判断结果为{}".format(node_id, dict_siri[YeziPanDuan]))
    def popfenlei(self, fenlei):
        ii = 0
        for i in self.fenleiindex:
            if int(i) == int(fenlei):
                self.fenleiindex.pop(ii)
                return
            ii = ii+1
# 按照树节点的编号找到节点在数组中的位置
def find_(node_id):
    for node in range(len(nodes)):
        if nodes[node].node_id == node_id:
            return node+1
    return -1
```

随后完成连续值二分法的代码实现：

```
# 把第num个数据进行排列,并取出可能的分割点
def get_mid(dataset,num):
    l1 = []
    l2 = []
    for i in range(len(dataset)):
        l1.append(dataset[i][num])
    l1 = list(set(l1))
    l1.sort()
    for i in range(len(l1)-1):
        l2.append(((float)(l1[i])+(float)(l1[i+1]))/2)
    return l2


# 分割函数
def randDivision(dataset , trainSize):
    copy = list(dataset)
    train = []
    while len(train)<trainSize:
        index = random.randrange(len(copy))
        train.append(copy.pop(index))
    return [train, copy]
```

然后进行熵值的计算（代码较长放在后面）

计算完成之后我们要写一个 max 函数来选出信息增益最大的特征和分类点

上面这些函数都会作为 buildtree 函数的子函数，帮助建造树

写一个五折验证函数，分数据集和递归的构造一颗决策树。

```
经五折交叉验证后得知，正确率为99.33333333333334%
```

然后继续看题目要求，完成预剪枝和后剪枝，因为上课这部分讲的比较少，于是又自己看书和上网搜集了一下相关的知识，采用了课本上提出的留出法，先留一部分作为剪枝集，通过看这一部分对树的影响，看一下会不会有效果，预剪枝是在构建过程中进行提前的结束，而后剪枝是构建完成后对树进行剪枝。这里代码也在后面会贴。

```
经预剪枝五折交叉验证后得知，正确率为96.66666666666666%
```

```
经后剪枝五折交叉验证后得知，正确率为99.33333333333334%
```

结果发现正确率有时会比不剪还要低。这里是我疑惑的一点。后来发现是训练集的精度降低，但是验证集的精度会有所上升，那么就提高了泛化能力。

结论分析与问题：

问题 1：结果发现正确率有时会比不剪还要低。这里是我疑惑的一点。后来发现是训练集的精度降低，但是验证集的精度会有所上升，那么就提高了泛化能力。

问题 2：实验数据集只有 150 条，本身就带有极大的偶然性，无法正确判断是不是坏的扰乱视听的数据，所以在判断泛化能力上，我觉得并不具有说服力。

实验代码：

```
# 201700301166 尹成林
import csv
import random
from math import log
import numpy as np

dict_iris = {'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2}
dict_siri = {0: 'Iris-setosa', 1: 'Iris-versicolor', 2: 'Iris-virginica'}



class treenode:
    def __init__(self, FenLei, FenLeiZhi, YeziPanDuan, node_id):
        self.FenLei = FenLei
        self.FenLeiZhi = FenLeiZhi
        self.YeziPanDuan = YeziPanDuan
        self.fenleiindex = [0, 1, 2]
        self.node_id = node_id
        if int(YeziPanDuan) == -1:
            print("产生中间节点{}，为第{}特征，分类值为{}".format(node_id, FenLei, FenLeiZhi))
        else:
            print("产生叶子节点{}，判断结果为{}".format(node_id, dict_siri[YeziPanDuan]))
    def popfenlei(self, fenlei):
        ii = 0
        for i in self.fenleiindex:
            if int(i) == int(fenlei):
                self.fenleiindex.pop(ii)
                return
```

```
            ii = ii+1

# 建立用来储存我们的树节点的
nodes = []


# 构建一个树
def buildTree(dataset, node_id):
    if get_ent(dataset) != 0:
        fenlei, fenleizhi = get_fenleidian(dataset)
        trnode = treenode(fenlei, fenleizhi, -1, node_id)
        trnode.popfenlei(fenlei)
        nodes.append(trnode)
        datasmall, databig = divide(dataset, fenlei, fenleizhi)
        #print(get_ent(datasmall))
        #print(get_ent(databig))
        buildTree(datasmall, 2 * node_id)
        #print("hhhhhhhhhhh{}".format(dataset[0][4]))
        buildTree(databig, 2 * node_id + 1)
    else:
        #print("hhhhhhhhhhh{}".format(dataset[0][4]))
        if len(dataset) == 0:
            #print("ffff")
            return
        panduan = dataset[0][4]
        trnode = treenode(0, 0, panduan, node_id)
        nodes.append(trnode)


# 加载函数
def loadcsv(name):
    f = csv.reader(open(name, 'r'))
    dataset = list(f)
    return dataset


# 把数据数字化
def digital(dataset):
    for i in range(len(dataset)):
        dataset[i][4] = dict_iris[dataset[i][4]]

# 把第 num 个数据进行排列,并取出可能的分割点
def get_mid(dataset,num):
    l1 = []
```

```
    l2 = []
    for i in range(len(dataset)):
        l1.append(dataset[i][num])
    l1 = list(set(l1))
    l1.sort()
    for i in range(len(l1)-1):
        l2.append(((float)(l1[i])+(float)(l1[i+1]))/2)
    return l2


# 分割函数
def randDivision(dataset , trainSize):
    copy = list(dataset)
    train = []
    while len(train)<trainSize:
        index = random.randrange(len(copy))
        train.append(copy.pop(index))
    return [train, copy]


'''
计算信息熵和信息增益
'''


# 划分数据集方便计算条件信息熵
def divide(dataset, index, dnum):
    dataset2 = list(dataset)
    dataset1 = []
    ii  = 0
    for i in dataset:
        if (float)(i[index])<(float)(dnum):
            dataset1.append(i)
            dataset2.pop(ii)
            ii = ii-1
        ii = ii+1
    return [dataset1, dataset2]


# 计算信息熵
def getP(dataset,num):
    p = 0.0
    a = len(dataset)
    if a == 0:
```

```
        return 0
    b = 0
    for i in dataset:
        if i[4] == num:
            b = b+1
    p = b/a
    #print(p)
    return p


def get_ent(dataset):
    p = []
    ent = 0.0
    for i in range(3):
        pp = getP(dataset, i)
        p.append(pp)
    for i in p:
        if i != 0:
            ent = ent - i*log(i, 2)
    return ent


# 计算信息增益
def get_gain(dataset, index, dnum):
    entD = get_ent(dataset)
    data1, data2 = divide(dataset, index, dnum)
    a1 = 0.0
    a2 = 0.0
    gain = 0.0
    '''print("data 的熵{}".format(entD))
    print("data1 的熵{}".format(get_ent(data1)))
    print("data2 的熵{}".format(get_ent(data2)))'''
    a2 = len(data2)/len(dataset)
    a1 = len(data1)/len(dataset)
    gain = entD - a1*get_ent(data1) - a2*get_ent(data2)
    return gain


# 得到单个特征最高的信息增益所在的分类点
def get_maxgain(dataset, index):
    max = -10000
    iii = 0
    dnums = get_mid(dataset, index)
    for i in dnums:
```

```
        gg = get_gain(dataset, index, i)
        if gg>=max:
            max = gg
            iii = i
    return iii


# 多个特征中选择一个分类点
'''
输入：数据集，各个特征的最佳分类点
输出：排序 list
'''
def get_maxtezheng(dataset,node):
    max = 0
    max_i = 0
    max_list = []
    for i in range(len(node)):
        a = get_gain(dataset, i, node[i])
        max_list.append(a)
        if max<a:
            max_i = i
            max = a
    return [max_list, max_i]


def get_fenleidian(dataset):
    divide_node = []
    for tezheng in range(4):
        divide_node.append(get_maxgain(dataset, tezheng))
        #print("**********")
        #print("所以应该第{}个特征选择{}作为分割点".format(tezheng, get_maxgain(dataset,
tezheng)))
        #print("**********")
    max_list, max_i = get_maxtezheng(trainSet, divide_node)
    #print(max_i)
    #print(max_list)
    return [max_i, divide_node[max_i]]


def find_(node_id):
    for node in range(len(nodes)):
        if nodes[node].node_id == node_id:
            return node+1
    return -1
```

```
# 验证函数
def classify(dataset):
    corr = 0    # 记录正确的数目

    allof = len(dataset)    # 总数
    for item in dataset:
        aa = 1
        #print(aa)
        nnn = nodes[aa-1]
        while int(nnn.YeziPanDuan) == int(-1):  # 在主节点之间遨游
            if float(item[int(nnn.FenLei)]) <= float(nnn.FenLeiZhi):
                aa = 2*aa
                #print("左")
                nnn = nodes[find_(aa)-1]   # 左子树
            else:
                #print("
                aa = 2*aa+1
                nnn = nodes[find_(aa)-1]     # 右子树
        #print("树结果为{}， 实际为{}".format(nnn.YeziPanDuan, item[4]))
        if nnn.YeziPanDuan == item[4]:  # 判断是否正确
            #print("树结果为{}， 实际为{}".format(nnn.YeziPanDuan, item[4]))
            corr = corr+1
        aa = 1
    bibi = 0.0
    bibi = corr/allof
    return bibi


# 五折交叉验证划分训练集
def getTrainTest(kkk, datas):
    kkkk = 0
    traindata = []
    for i in datas:
        #print("12112121")
        if kkkk == kkk:
            #print("kkk:{}, kkkk:{}".format(kkk, kkkk))
            kkkk = kkkk + 1
            continue
        kkkk = kkkk+1
        traindata = traindata + i
        #print(len(traindata))
    return traindata
```

```
# 五折交叉验证
def FiveFordCV(dataset, size, nums):
    s = int(size/nums)
    print(s)
    dataForTfcv = []
    kkk = 0
    Rate = []
    testSet = dataset
    for i in range(nums-1):
        trainSet, testSet = randDivision(testSet, s)
        dataForTfcv.append(trainSet)
    dataForTfcv.append(testSet)
    # print(len(dataForTfcv))
    for i in dataForTfcv:
        # print(kkk)
        testSetForTen = getTrainTest(kkk, dataForTfcv)
        kkk = kkk+1
        buildTree(testSetForTen, 1)
        # print(len(testSetForTen))
        aaa = classify(i)
        Rate.append(aaa)
        #print("第{}次正确率为{}".format(kkk,rate[1] / (rate[0] + rate[1])))
    return np.mean(Rate)


# main function
if __name__=="__main__":
    name = "..\data\iris.csv"
    dataset = loadcsv(name)
    digital(dataset)
    trainSet, testSet = randDivision(dataset, 100)
    #print(get_fenleidian(trainSet))
    #set1, set2 = divide(trainSet, 3, 0.8)
    #tr = treenode(0,0,0,0)
    #buildTree(trainSet, 1)
    #a = classify(testSet)
    #print("验证得知，正确率为{}%".format(a*100))
    aaaaa = FiveFordCV(dataset, 150, 5)
    print("经五折交叉验证后得知，正确率为{}%".format(aaaaa*100))
```