编程实验作业一 决策树

|  |
| --- |
| 姓名：任晓路 |
| 学号：16130120190 |
| 时间：2019年4月2日 |

1. 实验简介：

本次实验通过学习决策树的三大经典算法ID3、C4.5和CART，对决策树算法实现分类任务的原理和代码进行详细分析和学习，并通过前两个算法的思路改写实现CART算法。

1. 实验原理：

https://pic4.zhimg.com/80/v2-d673f426d5b4a93f1b3ef3e6ae4902ab_hd.jpg

https://pic1.zhimg.com/80/v2-b15214f5fae511335fd12ecdfed82ff8_hd.jpg

https://pic4.zhimg.com/80/v2-476ea9f62014c042af2a19af865490cf_hd.jpg

最后delta值越小的属性划分方法效果越好。

1. 实验代码与注释：

# -\*- coding: utf-8 -\*-

from math import log

import operator

import treePlotter

def calcGini(dataSet):

numEntries = len(dataSet) #统计数据实例个数

labelCounts = {} # 字典,一个键值对

for featVec in dataSet: #按照行遍历数据集，本质是记录键出现的次数

currentLabel = featVec[-1] #又定义了一个变量，取得每行最后一个元素

#如果最后一个元素不是已有键值

if currentLabel not in labelCounts.keys():

#键值对的值为0，即这是第一次出现的键，需要记录在一个列表中

labelCounts[currentLabel] = 0

labelCounts[currentLabel] += 1 #这是个计数函数

Gini=1.0 #基尼指数是 1 - pk^2，故最大默认值应该是1

for key in labelCounts:

prob = float(labelCounts[key]) / numEntries

Gini -= prob \* prob

return Gini

def splitDataSet(dataSet, axis, value):

"""

输入：数据集，选择维度，选择值

输出：划分数据集

描述：按照给定特征划分数据集；去除选择维度中等于选择值的项 #如果等于选择值就是刚好处于分界点的值

"""

retDataSet = [] # 存储分类完毕的集合

for featVec in dataSet:

# 若第 axis号元素等于初始给定的value值 则记录下来，相等意味着不要

if featVec[axis] == value:

# python里第二个参数的实际位置为n - 1

reduceFeatVec = featVec[:axis]

# 这两条语句是把featVec数组中除了特征值的元素给保留下来

# 就是会让所有元素少上一列

reduceFeatVec.extend(featVec[axis+1:])

# 把剩下的项目放回去，因为树的子树不需要这个特征值了

retDataSet.append(reduceFeatVec)

return retDataSet

def chooseBestFeatureToSplit(dataSet):

"""

输入：数据集

输出：最好的划分维度

描述：选择最好的数据集划分维度

"""

numFeatures = len(dataSet[0]) - 1 #特征值的数量，减去yes或no(就是列数)

bestGiniIndex = 100 #初始化Gini的index，要大于列表大小

bestFeature = -1 # 初始默认最好的特征为-1

for i in range(numFeatures):

#每次循环取第i列所有特征值，计算基尼指数

featList = [example[i] for example in dataSet]

#set集合中存放的是所有不同的特征值，本题最多只有[0,1,2]

uniqueVals = set(featList)

newGiniIndex = 0.0 #初始化一个暂存基尼指数的值

for value in uniqueVals: # 进行遍历 调用splitDataSet()进行划分

subDataSet = splitDataSet(dataSet, i, value)

#划分后在原集合中占的比例

prob = len(subDataSet)/float(len(dataSet))

#比例乘以划分后集合的基尼值 循环算出总基尼指数值

newGiniIndex += prob \* calcGini(subDataSet)

GiniIndex=newGiniIndex

#与目前最好基尼指数比，若小于则表示这种划分方式是更好的

if (GiniIndex < bestGiniIndex):

bestGiniIndex = GiniIndex

bestFeature = i #这两行记录最优划分的值和其对应位置

return bestFeature

def majorityCnt(classList):

"""

输入：分类类别列表

输出：子节点的分类

描述：数据集已经处理了所有属性，但是类标签依然不是唯一的，

采用多数判决的方法决定该子节点的分类

"""

classCount = {}

for vote in classList:

if vote not in classCount.keys():

classCount[vote] = 0 #iteritems是返回当前字典操作后的迭代

classCount[vote] += 1

# reverse默认False是升序 True表示降序

sortedClassCount=sorted(classCount.iteritems(),key=operator.itemgetter(1), reversed=True)

return sortedClassCount[0][0] #返回的是值最大的那个

def createTree(dataSet, labels): #最核心的函数

"""

输入：数据集，特征标签

输出：决策树

描述：递归构建决策树，利用上述的函数

"""

#取出双重列表里的最后一列元素，也就是一整列标记的值（Y或者N）

classList = [example[-1] for example in dataSet]

#如果classlist中的第一个值在classlist中的总数等于长度

#也就是classlist中所有值都一样，则停止划分

if classList.count(classList[0]) == len(classList):

return classList[0]

#说明属性栏中目前只有一项属性，又由于上一个IF没有执行故属性还不相同

if len(dataSet[0]) == 1:

# 遍历完所有特征时返回出现次数最多的

return majorityCnt(classList)

#最优的特征 先划分出判断节点

bestFeat = chooseBestFeatureToSplit(dataSet)

bestFeatLabel = labels[bestFeat] #最优特征对应的标签

#把标签存在树中 这时候树是用循环字典来保存的

myTree = {bestFeatLabel:{}}

del(labels[bestFeat]) #在标签中删除已经选择过的标签

# 得到列表包括节点所有的属性值

featValues = [example[bestFeat] for example in dataSet]

uniqueVals = set(featValues)

for value in uniqueVals:

subLabels = labels[:] #去掉已经删除过的标签后的标签集合

#循环字典的值，这里是根据最优特征划分出来的集合，然后把该集合放入树中

myTree[bestFeatLabel][value]=createTree(splitDataSet(dataSet,bestFeat, value), subLabels)

return myTree

def classify(inputTree, featLabels, testVec):

"""

输入：决策树，分类标签，测试数据

输出：决策结果

描述：跑决策树

"""

firstStr = list(inputTree.keys())[0] #得到决策树第一个分类标签

secondDict = inputTree[firstStr] #得到第一个标签对应第二个目录

#得到在分类标签中决策树第一个标签的位置

featIndex = featLabels.index(firstStr)

for key in secondDict.keys():

if testVec[featIndex] == key:

if type(secondDict[key]).\_\_name\_\_ == 'dict': #要么继续区分

#这里是一个递归

classLabel = classify(secondDict[key], featLabels, testVec)

else:

#要么不是一个目录，说明到叶子节点了，得到分类标签

classLabel = secondDict[key]

return classLabel

def classifyAll(inputTree, featLabels, testDataSet):

"""

输入：决策树，分类标签，测试数据集

输出：决策结果

描述：跑决策树

"""

classLabelAll = []

for testVec in testDataSet: #循环遍历测试数据集

#循环测试完每个数据集

classLabelAll.append(classify(inputTree, featLabels, testVec))

return classLabelAll #返回测试结果

def createDataSet():

"""

outlook-> 0: sunny | 1: overcast | 2: rain

temperature-> 0: hot | 1: mild | 2: cool

humidity-> 0: high | 1: normal

windy-> 0: false | 1: true

"""

dataSet = [[0, 0, 0, 0, 'N'],

[0, 0, 0, 1, 'N'],

[1, 0, 0, 0, 'Y'],

[2, 1, 0, 0, 'Y'],

[2, 2, 1, 0, 'Y'],

[2, 2, 1, 1, 'N'],

[1, 2, 1, 1, 'Y']]

labels = ['outlook', 'temperature', 'humidity', 'windy']

return dataSet, labels

def createTestSet():

"""

outlook-> 0: sunny | 1: overcast | 2: rain

temperature-> 0: hot | 1: mild | 2: cool

humidity-> 0: high | 1: normal

windy-> 0: false | 1: true

"""

testSet = [[0, 1, 0, 0],

[0, 2, 1, 0],

[2, 1, 1, 0],

[0, 1, 1, 1],

[1, 1, 0, 1],

[1, 0, 1, 0],

[2, 1, 0, 1]]

return testSet

def main():

dataSet, labels = createDataSet() #函数返回了两个参数

labels\_tmp = labels[:] # 拷贝，createTree会改变labels

desicionTree = createTree(dataSet, labels\_tmp)

print('desicionTree:\n', desicionTree)

treePlotter.createPlot(desicionTree)

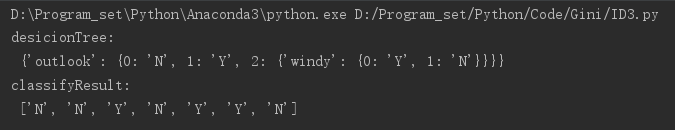
testSet = createTestSet()

print('classifyResult:\n', classifyAll(desicionTree, labels, testSet))

if \_\_name\_\_ == '\_\_main\_\_': #如果是通过命令行启动，则直接启动

main()

1. 实验结果：



N N Y N Y Y N

