模型-机器学习-分类-决策树【hxy】

1. 模型名称

决策树（Decision Tree，DT）

2. 模型评价

2.1 优点

- 计算复杂度不高
- 输出结果易于理解
- 对中间值的缺失不敏感
- 可以处理不相关特征数据

2.2 缺点

- 可能会产生过度匹配问题

3. 基本算法

```
%% 创建分支createBranch()的伪代码
检测数据集中的每个子项是否属于同一分类:
    If so: return 类标签;
    Else:
        寻找划分数据集的最好特征
        划分数据集
        创建分支节点
            for 每个划分的子集
                调用函数createBranch并增加返回结果到分支节点中
        return 分支节点
```

- 在**寻找划分数据集最好特征**时
    1. 计算每种划分方式的**信息熵**

符号x_i的信息定义：$l(x_i)=-log_2p(x_i)$，其中$p(x_i)$是选择该分类的概率

熵$(entropy)$定义为信息的期望值：$E=-\sum_{i=1}^{n}p(x_i)log_2p(x_i)$，其中$n$是分类的个数

2. 计算最好的**信息增益**

   样本集D中所有在属性a上取值为ax的样本记为Dx，则属性a对样本集D进行划分所获得的信息增益：

$$Gain(D,a)=Ent(D)-\sum_{x=1}^{X}\frac{|Dx|}{D}Ent(Dx)$$

3. 最好的**信息增益**对应的就是最好的**划分数据集的特征**

4. 实例

4.1 数据介绍

隐形眼镜数据集包含很多患者眼部状况的观察条件以及医生推荐的隐形眼镜类型，隐形眼镜类型包括硬材质、软材质以及不适合佩戴隐形眼镜，数据来源于UCI数据库，分类标准为年龄、规定、是否散光、泪率

```
young myope no  reduced no lenses
young myope no  normal  soft
young myope yes reduced no lenses
young myope yes normal  hard
young hyper no  reduced no lenses
young hyper no  normal  soft
young hyper yes reduced no lenses
young hyper yes normal  hard
pre myope no  reduced no lenses
pre myope no  normal  soft
pre myope yes reduced no lenses
pre myope yes normal  hard
pre hyper no  reduced no lenses
pre hyper no  normal  soft
pre hyper yes reduced no lenses
pre hyper yes normal  no lenses
presbyopic  myope no  reduced no lenses
presbyopic  myope no  normal  no lenses
presbyopic  myope yes reduced no lenses
presbyopic  myope yes normal  hard
presbyopic  hyper no  reduced no lenses
presbyopic  hyper no  normal  soft
presbyopic  hyper yes reduced no lenses
presbyopic  hyper yes normal  no lenses
```

4.2 实验目的

得到不同患者需要佩戴的隐形眼镜类型

4.3 代码实现

DT.py

代码：

```
from math import log
import operator

# 计算香农熵(Shannon Entropy)
def calcShannonEnt(dataSet):
    # 计算样本数量
    numEntries=len(dataSet)
    # 字典记录每一个属性(key)和其对应的数量(value)
    labelCounts={}
    for featVec in dataSet:
        # 每行数据的最后一个是类别
        currentLabel=featVec[-1]
        # 统计有多少个类以及每个类的数量
        if currentLabel not in labelCounts.keys():
            labelCounts[currentLabel]=0
        labelCounts[currentLabel]+=1
    # 计算香农熵(Shannon Entropy)
    shannonEnt=0
    for key in labelCounts:
        prob=float(labelCounts[key])/numEntries # 计算单个类的熵值
        shannonEnt-=prob*log(prob,2) # 累加每个类的熵值
    return shannonEnt

# 按照给定属性的给定取值提取数据集
def splitDataSet(dataSet,axis,value):
    # 用来存储提取后的数据集
    retDataSet=[]
    # 如果样本的给定属性的取值恰好等于给定取值
    for featVec in dataSet:
        if featVec[axis]==value:
            # 保留此属性前的属性的取值
            reducedFeatVec =featVec[:axis]
            # 加上保留此属性后的属性的取值
            reducedFeatVec.extend(featVec[axis+1:])
            # 将保留前后属性取值的样本加到总的列表中
            retDataSet.append(reducedFeatVec)
    return retDataSet

# 选择最好的数据集的划分方式
def chooseBestFeatureToSplit(dataSet):
```

```
    # 计算属性数量
    numFeatures = len(dataSet[0])-1
    # 计算数据集整体熵
    baseEntropy = calcShannonEnt(dataSet)
    # 用来存储最好的信息增益(bestInfoGain)和最好划分特征(bestFeature)
    bestInfoGain = 0
    bestFeature = -1
    for i in range(numFeatures):
        # 建立该属性所有样本的取值的列表
        featList = [example[i] for example in dataSet]
        # 建立该属性所有可能的取值(即分类标签)的列表
        uniqueVals = set(featList)
        # 用来存储该属性的熵
        newEntropy = 0
        # 对于该属性的每一个分类标签
        for value in uniqueVals:
            # 得到该属性该分类标签的样本数据列表
            subDataSet = splitDataSet(dataSet,i,value)
            prob =len(subDataSet)/float(len(dataSet))
            # 按特征分类后的熵
            newEntropy +=prob*calcShannonEnt(subDataSet)
        # 计算信息增益
        infoGain = baseEntropy - newEntropy
        # 若按某特征划分后，熵值减少的最大，则次特征为最优分类特征
        if (infoGain>bestInfoGain):
            bestInfoGain=infoGain
            bestFeature = i
    return bestFeature

#按分类后类别数量排序，消除噪音，比如：最后分类为2R1A，则判定为R；
def majorityCnt(classList):
    classCount={}
    for vote in classList:
        if vote not in classCount.keys():
            classCount[vote]=0
        classCount[vote]+=1
    sortedClassCount =
sorted(classCount.items(),key=operator.itemgetter(1),reverse=True)
    return sortedClassCount[0][0]

# 创建树
def createTree(dataSet,labels):
    # 建立所有可能结果的列表
    classList=[example[-1] for example in dataSet]
    # 如果所有的结果都一致，则停止继续划分
    if classList.count(classList[0])==len(classList):
        return classList[0]
    # 如果遍历完所有属性仍有未分类的，则将其划分到次数最多的结果上
    if len(dataSet[0])==1:
```

```
        return majorityCnt(classList)
    # 计算最好的数据集划分方式
    bestFeat=chooseBestFeatureToSplit(dataSet)
    # 得到最好的数据集划分方式的属性名称
    bestFeatLabel=labels[bestFeat]
    # 建立以该属性为根节点的树
    myTree={bestFeatLabel:{}}
    # 将这个属性从属性列表中删除（不再作为后续选最好划分方式的属性）
    del(labels[bestFeat])
    # 得到该属性所有样本的取值的列表
    featValues=[example[bestFeat] for example in dataSet]
    # 得到该属性所有可能的取值的列表
    uniqueVals=set(featValues)
    # 对于每一个可能的取值
    for value in uniqueVals:
        subLabels=labels[:]
        myTree[bestFeatLabel][value]=createTree(splitDataSet\
                        (dataSet,bestFeat,value),subLabels)
    return myTree


if __name__=='__main__':
    fr = open('/Users/xinyuanhe/Desktop/working/2021美赛/模型/【正式】模型-机器学习-分类-决策树【hxy】/lenses.txt')
    lenses = [inst.strip().split('\t') for inst in fr.readlines()]
    lensesLabels = ['age', 'precript', 'astigmatic', 'tearRate']
    print(createTree(lenses, lensesLabels))
```

结果：

```
>>>
= RESTART: /Users/xinyuanhe/Desktop/working/2021美赛/模型/【正式】模型-机器学习-分类-决策树【hxy】/DT.py
{'tearRate': {'normal': {'astigmatic': {'yes': {'precript': {'hyper': {'age': {'young': 'hard', 'presbyopic': 'no lenses', 'pre': 'no lenses'}}, 'myope': 'hard'}}, 'no': {'age': {'young': 'soft', 'presbyopic': {'precript': {'hyper': 'soft', 'myope': 'no lenses'}}, 'pre': 'soft'}}}}, 'reduced': 'no lenses'}}
```

5. 参考资料

1. 决策树+代码实现（来源于教材）
2. 《机器学习实战》P18-P28 机器学习实战.pdf
3. 数模官网-决策树