第121天：机器学习之决策树

原创 某某白米饭 Python技术 2月6日

决策树（Decision Tree）

决策树是一种分类回归算法，决策树采用的是树形结构。每一个内部节点表示对于特征属性的判断，每一个分支就代表对这个特征属性判断的输出，每一个叶节点就是对决策结果的分类。

举个例子：在贷款买房、买车时，为了防止不良贷款，银行一般会看借款人的银行流水是否合格，如：月收入是否达标、时间是否合规、流水是否是造假。

决策树是一种常用的分类方法，是监督学习的一种，需要给出一些数据集样本，这些样本中包括了特征属性、决策分类的结果。通过这些数据集样本能够得到一个决策树，通过决策树得出新样本的分类结果。

决策树的优点是计算的复杂度不高、输出结果容易理解、对中间值的缺省不敏感。缺点是可能产生过度匹配的问题。

特征属性

在构造决策树时，需要解决的第一个问题就是，在数据集样本中，每个样本都有许多特征属性，每个特征属性对决策结果的影响都有大有小。为了找到那些决定性的特征，必须对每个特征进行评估、选择。对每个特征属性进行评估、选择就是对数据集的划分，一个数据集将被分为多个数据子集。

在选择特征属性时通常使用的方法为：信息增益。

信息增益

在选择特征属性之前和之后数据集发生的变化称之为信息增益。只要知道如何计算信息增益，就可以知道哪个特征属性就是最好的选择。

集合信息的度量方式被称为：香浓熵或者简称为熵。熵：信息的期望值，集合的无序程度，熵越大表示集合越无序，熵越小表示集合越有序。

如果待分类的数据集中可能会划分出多个分类，则符号 x~i~ 的信息定义为

$$l(x_i) = -log_2 p(x_i)$$

其中 p(x~i~) 是这个分类的概率。

通过下面的公式可以得到所有类别的信息期望值，n 是分类的数目:

ID3.5 算法

ID3.5 算法的核心思想是在决策树各个结点上选择最优的信息增益得出特征属性，递归地构建决策树，直到没有特征选择为止。最后得到一个决策树。

举个例子使用 ID3.5 算法计算熵与信息增益，下表是简单的银行流水是否达标

申请人序号	月收入是否达标	时间是否合规	流水是否造假	银行流水是否达标
1	是	是	否	是
2	是	是	否	是
3	是	否	否	否
4	否	是	否	否
5	否	否	是	否
6	是	否	是	否
7	是	是	是	否

使用 ID3.5 算法计算上表中熵的值，把【是】用 1 表示，【否】用 0 表示，最后是否达标用 y/n 表示

```
from math import  log


def createDataSet():
    '''
    创建数据集
    '''

    dataSet = [[1, 1, 0, 'y'],
               [1, 1, 0, 'y'],
               [1, 0, 0, 'n'],
               [0, 1, 0, 'n'],
               [0, 0, 1, 'n'],
               [1, 0, 1, 'n'],
               [1, 1, 1, 'n']]
    labels = ['Salary', 'Time', 'Bank flow']
    return dataSet,labels


def calcEntropy(dataSet):
    '''
    计算熵
```

```
    计算熵
    :param dataSet: 数据集
    :return: 熵值
    '''

    numEntries = len(dataSet)
    labelCounts = {}
    for line in dataSet:
        currentLabel = line[-1]
        if currentLabel not in labelCounts.keys():
            labelCounts[currentLabel] = 0
        labelCounts[currentLabel] += 1
    entropy = 0.0
    for key in labelCounts:
        prob = float(labelCounts[key]) / numEntries
        entropy -= prob * log(prob, 2)
    return entropy


mydata = createDataSet()

entropy = calcEntropy(mydata)

print('熵值为：', entropy)
```

示例结果

```
熵值为：0.863120568566631
```

熵值越大表示集合越无序，熵越小表示集合越有序。

下面使用 Python 代码计算出示例中的最优特征

```
def splitDataSet(dataSet,axis,value):
    '''
    划分数据集
    :param dataSet: 按照给定特征划分数据集
    :param axis: 划分数据集的特征
    :param value: 需要返回的特征的值
    :return: 经验熵
    '''
    retDataSet=[]
    for featVec in dataSet:
        if featVec[axis]==value:
            reducedFeatVec=featVec[:axis]
            reducedFeatVec.extend(featVec[axis+1:])
            retDataSet.append(reducedFeatVec)
```

```
    return retDataSet

def chooseBestFeatureToSplit(dataSet):
    '''
    计算数据集的熵
    :param dataSet: 数据集
    :return: 最优的特征值的索引
    '''

    # 特征个数
    numFeatures = len(dataSet[0]) - 1
    # 数据集的熵
    baseEntropy = calcEntropy(dataSet)
    # 最优信息增益
    bestInfoGain = 0.0
    # 最优特征的索引值
    bestFeature = -1

    for i in range(numFeatures):
        # 获取数据集的第 i 个所有特征
        featList = [example[i] for example in dataSet]
        #创建 set集合{}，元素不可重复
        uniqueVals = set(featList)
        # 经验条件熵
        newEntropy = 0.0
        #计算信息增益
        for value in uniqueVals:
            # 数据集划分后的子集
            subDataSet = splitDataSet(dataSet, i, value)
            # 计算子集的概率
            prob = len(subDataSet) / float(len(dataSet))
            # 根据公式计算经验条件熵
            newEntropy += prob * calcEntropy((subDataSet))
        # 信息增益
        infoGain = baseEntropy - newEntropy
        # 打印每个特征的信息增益
        print("第%d个特征属性的信息增益为%.3f" % (i, infoGain))

        if (infoGain > bestInfoGain):
            bestInfoGain = infoGain
            bestFeature = i
    return bestFeature

mydata = createDataSet()

print("最优的索引值为：", str(chooseBestFeatureToSplit(mydata)))
```

示例结果

```
第0个特征属性的信息增益为0.170
第1个特征属性的信息增益为0.292
第2个特征属性的信息增益为0.292
最优的索引值为：1
```

在计算出第二个最优特征属性后，可以继续使用递归方式计算第二个最优特征属性，直至得出所有可能的决策类别。

下面构建决策树

```
import operator

def majorityCnt(classList):
    '''
    类别数多的类别
    :param classList: 类别
    :return: 返回类别数多的类别
    '''
    classCount={}
    for vote in classList:
        if vote not in classCount.keys(): classCount[vote] = 0
        classCount[vote] += 1
    sortedClassCount = sorted(classCount.iteritems(), key=operator.itemgetter(1), reverse=True)
    return sortedClassCount[0][0]

def createTree(dataSet,labels):
    '''
    构建决策树
    :param dataSet: 数据集样本
    :param labels: 特征属性
    :return: 决策树
    '''

    # 决策类别
    classList = [example[-1] for example in dataSet]
    # 类别完全相同停止继续划分
    if classList.count(classList[0]) == len(classList):
        return classList[0]
    # 返回出现次数最多的类别
    if len(dataSet[0]) == 1:
        return majorityCnt(classList)
    # 返回最优的特征属性
    bestFeature = chooseBestFeatureToSplit(dataSet)
    bestFeatLabel = labels[bestFeature]
    myTree = {bestFeatLabel:{}}
    del(labels[bestFeature])
    # 最优特征值
    featureValues = [example[bestFeature] for example in dataSet]
    uniqueVals = set(featureValues)
```

```
    for value in uniqueVals:
        subLabels = labels[:]
        myTree[bestFeatLabel][value] = createTree(splitDataSet(dataSet, bestFeature, value), subLabe
    return myTree

```

示例结果

```
第0个特征属性的信息增益为0.170
第1个特征属性的信息增益为0.292
第2个特征属性的信息增益为0.292
第0个特征属性的信息增益为0.311
第1个特征属性的信息增益为0.311
第0个特征属性的信息增益为0.918
{'Time': {0: 'n', 1: {'Salary': {0: 'n', 1: {'Bank flow': {0: 'y', 1: 'n'}}}}}}
```

总结

简单的介绍了决策树和 ID3.5 算法，用了一个示例构造了一个简单的决策树，希望对大家有所帮助。

> 示例代码：https://github.com/JustDoPython/python-100-day/tree/master/day-121

PS： 公号内回复：Python，即可进入Python 新手学习交流群，一起**100天计划！**

-END-

Python 技术

关于 Python 都在这里