《数据挖掘技术概论》实验报告

实验一 决策树分类实验

学号 2017141463226

姓名 刘雨欣

一、实验目的

● 目的:

通过实现决策树算法,理解决策树的基本原理和基本的程序实现方法。

● 要求:

- (1) 实现基本的 ID3 算法(信息增益、离散型属性) 包括训练和测试功能,并自己选择数据集进行验证;
- (2) 实现对连续值属性的处理,并自己选择数据集进行实验验证;
- (3)尝试其他的属性筛选方法,如 Gini Index,选择数据集进行验证。
- 程序设计语言: C、C++、Java 或 Python

二、 算法描述和分析

决策树(decision tree)是一种树形结构,其中每个内部节点表示一个属性上的测试,每个分支代表一个测试输出,每个叶节点代表一种类别。决策树可以用来对未知样本分类,分类过程如下:从决策树的根节点开始,从上往下沿某个分支往下搜索,直到叶节点,以叶节点的类标号值作为该未知样本所属的类标号。[1]

决策树的主要优点是模型具有可读性,分类速度快。学习时,利用训练数据,根据损失函数最小化的原则建立决策树模型;预测时,对新的数据,利用决策树模型进行分类。决策树学习通常包括三个步骤:特征选择、决策树的生成和决策树的剪枝。[2]

• 特征选择:

特征选择是决定用哪个特征来划分特征空间,即使用哪个特征作为分支依据生成子树。通常来说,使用信息增益(information gain)作为度量特征分类能力好坏的准则。

首先引入熵的概念,在信息论与概率统计中,熵(entropy)是表示随机变量不确定性的度量。设U是一个取有限个数值的离散随机变量,其概率分布为: $P(U=u_i)=p_i$, $i=1,2,\cdots,n$ 则随机变量U的熵定义为

$$H(U) = E[-\log p_i] = -\sum_{i=1}^{n} p_i \log p_i$$

式中对数一般取 2 为底。又引入条件熵的概念,设有随机变量 (U,V),其联合概率分布为: $P(U=u_i,V=v_i)=p_{ij}$, $i=1,2,\cdots,n$; $j=1,2,\cdots,m$ 则随机变量 U 给定的条件下随机变量 V 的条件熵 H(V|U)定义为

$$H(V|U) = \Sigma p_i H(V|U = u_i)$$

最后信息增益 g(D,A)定义为集合 D 的信息熵与特征 A 给定条件下 D 的条件熵 H(D|A)之差,即

$$g(D,A) = H(D) - H(D|A)$$

根据信息增益准则的特征选择方法是:对训练数据集(或子集)D,计算 其每个特征的信息增益,比较它们的大小,选择信息增益最大的特征。

· 决策树的生成:

决策树的生成主要有三种算法,主要是 ID3 算法、C4.5 算法和 CART 算法。本次实验中,选择了 ID3 算法进行对离散值决策树的构建,选择信息增益值最大的属性产生决策树结点,由该属性的不同取值建立分支,再对各分支的子集递归调用该方法建立决策树结点的分支,直到所有节点仅包含同一个类别的数据为止。对于连续值决策树,我们在进行信息增益判断时,再加入一次对于最优连续值划分点的选择判断:对于每一个可能的连续值划分点,求得划分后的信息增益,选择信息增益最大的阈值进行划分。

三、 程序实现技术技巧的介绍和分析

本次实验将在 Jupyter 环境下使用 python 3 编码实现决策树的构建,并在所选的数据集上进行训练和验证。程序实现将分为四部分,分别是数据处理、香农熵计算、构建决策树和决策树可视化。

数据处理部分即为从 txt 文件按行读取数据,按分词符划分属性存入数组,生成列表,其中标签类别存为每个向量的最后一个值。

```
###
#M TXT 中读取数据创建数据集
###

def createDataSet(filename):
    f = open(filename)
    lines = f.readlines()
    f.close()
    dataSet = []
    for line in lines:
        data = []
        line = line.strip('\n')
        data = line.split(', ')
        dataSet.append(data)
    return dataSet
```

算法计算部分分为香农熵计算和信息增益计算,实现思路主要为,将分化后的数据子集传入 calcEnt()函数,根据公式计算出当前子集的信息熵;信息增益函数 getGain()则对于传入的分化属性,都提取出子集并使用 calcEnt()函数计算熵,并返回信息增益。主要的代码实现如下:

```
#计算香农熵
def calcEnt(dataSet):
    numData = len(dataSet)
```

```
#即统计各类结果的个数
    labelCounts = countFeatClass(dataSet,-1)
    Ent = 0.0
    for key in labelCounts:
       prob = float(labelCounts[key]) / numData
                                             #每类所占的比例
       Ent -= prob*log(prob,2)
                                            #根据公式计算熵
    return Ent
#获取在当前结点分化第i 特征后的信息增益
def getGain(dataSet,i):
                                              #父节点的香农熵
    S = calcEnt(dataSet)
    labelCounts = countFeatClass(dataSet,i)
    sumOfKidsEnt = 0.0
    for key in labelCounts:
       EntOfKid = calcEnt(splitDataSet(dataSet,i,key)) #子节点一个分支的Ent
                                              #当前分化所占权重比例
       p = float(labelCounts[key])/len(dataSet)
                                               #累加计算子节点总 Ent
       sumOfKidsEnt += p*EntOfKid
    return S - sumOfKidsEnt
                                               #返回增益
```

对于连续值的决策树,我们在每次计算信息增益时,对于每一个可能的划分点都进行一次计算。首先对于每条数据,按照该所选特征的值大小排序,选择每相邻值的中点作为可能的划分点,依次假设划分后,计算信息增益,返回信息增益最大的最优划分值和信息增益值。主要代码如下:

```
#计算两类分化的香农熵
def calcEntOf2Classes(Set1,Set2):
    num = len(Set1) + len(Set2)
    Ent = len(Set1)/num*calcEnt(Set1)+len(Set2)/num*calcEnt(Set2)
    return Ent
#计算第i 个特征最好的连续值划分点及其信息增益
def calcSeriesGain(dataSet,i):
    S = calcEnt(dataSet)
    maxGain = 0.0
    splitValue = -1
    valueList = [float(data[i]) for data in dataSet]#
    classList = [data[-1] for data in dataSet]
    dictList = dict(zip(valueList,classList))
    sortedValueList = sorted(dictList.items())
    midValueList = []
    for j in range(len(sortedValueList)-1):
         midValueList.append(round((sortedValueList[j][0] +
                                        sortedValueList[j+1][0]/2.0, 3)
     #计算相邻值的中值
    for mid in midValueList:
         smallerSet, biggerSet = splitSeriesDataSet(dataSet, i, mid)
         EntOfmid = calcEntOf2Classes(smallerSet,biggerSet)
         gain = S-EntOfmid
```

```
if gain > maxGain:

splitValue = mid

maxGain = gain

return maxGain,splitValue
```

在建树板块,我们从整个数据集开始,遍历每个特征值计算划分后的信息增益,选择当前最大的特征划分出子集,并递归调用 buildTree()函数,传入数据集子集,自上而下依次生成子树,直到最大信息增益为 0,即该子集下所有数据属于同一类别的情况,停止生成树。主要实现代码如下:

```
def buildTree(dataSet,kidTree):
    i = 0
    rtn = kidTree
                                             #暂时存储各种分化情况的 Gain
    tempGain = []
    while i<8:
        tempGain.append(getGain(dataSet,i))
        i += 1
    attr = tempGain.index(max(tempGain))
    labelCounts = countFeatClass(dataSet,attr)
                                    #Gain 为0 建树完成,返回分类作为叶子
    if(tempGain[attr]) == 0:
            rtn = dataSet[0][-1]
            return rtn
    rtn[label] = \{\}
    for key in labelCounts:
                                                #分化下一子树的数据集
        kidDataSet = splitDataSet(dataSet,attr,key)
        rtn[label][key] = buildTree(kidDataSet, {})
                                                   #递归建子树
                                                   #返回本结点的子树
    return rtn
```

最后,调用 matplotlib 库进行绘图,实现决策树的可视化,主要实现思路为计算树的宽度(叶子节点数)和深度,在预设的图像大小下计算每一条树的分支的长度,并把划分的特征值标记在分支上。

在数据测试阶段,对于每一条测试集数据,我们从树的根节点开始,从 上到下按照属性值选择分支,最终到达叶节点完成分类,比较是否与预期 分类相同。主要实现代码如下:

```
def classify(inputTree,featList,testVec):
    firstFeat = list(inputTree.keys())[0]
    currentTree = inputTree[firstFeat]
    value = testVec[featList.index(firstFeat)]
    deeperTree = currentTree[value]
    if isinstance(deeperTree,dict): #非叶子结点则继续递归深入
        rtn = classify(deeperTree,featList,testVec)
    else:
        rtn = deeperTree #叶子节点则返回叶子
```

return rtn

四、实验数据和实验方法

对于本实验,我们选用两个数据集来分别实现离散值决策树和连续值决策树。两个数据集都选自 KEEL-dataset。首先使用训练集构建离散值决策树和连续值决策树,再利用树对测试集中每一条数据进行分类,比较预测分类与预期分类是否相同,最后计算每个决策树的分类准确度。

离散值决策树数据集 Post-Operative

- 数据集任务描述:
- 8个特征值描述患者术后的恢复情况,判断下一步送往哪个恢复区。
- 数据集基本概况:

Post-Operative data set				
Туре	Classification	Origin	Real world	
Features	8	(Real / Integer / Nominal)	(0 / 0 / 8)	
Classes	3	Missing values?	Yes	
Total instances	90	Instances without missing values	87	

Attribute	Domain	备注	
L-CORE	{mid, high, low}	病人内部体温	
L-SURF	{low, high, mid}	病人外部体温	
L-02	{excellent, good}	病人氧饱和度	
L-BP	{mid, high, low}	病人最后血压	
SURF-STBL	{stable, unstable}	体表温度稳定性	
CORE-STBL	{stable, unstable, mod-stable}	体内温度稳定性	
BP-STBL	{stable, mod-stable, unstable}	血压稳定性	
COMFORT	{15, 10, 05, 07}	病人舒适程度	
Decision	{A, S, I}	-I 重症监护室 -S 回家 -A 一般楼层病院	

数据集中每个属性的值都为离散值,共有8个特征,3种类别,共90条数据,其中有三条数据有缺失值。对于有缺失值的数据,在实验中我们直接选择舍去不使用,整个数据集按照约9:1的比例来划分训练集和测试集,即训练集78条,测试集9条。

连续值决策树数据集 Glass Identification

- · 数据集任务描述:
- 9个连续特征值表示不同化学元素含量,对玻璃种类进行分类。

• 数据集基本概况

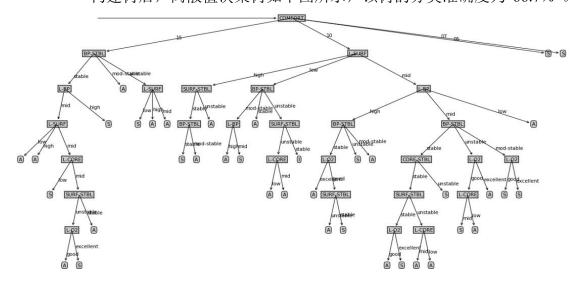
Glass Identification data set				
Туре	Classification	Origin	Real world	
Features	9	(Real / Integer / Nominal)	(9 / 0 / 0)	
Instances	214	Classes	7	
Missing values?			No	

Attribute	Domain
RI	[1.51115, 1.53393]
Na	[10.73, 17.38]
Mg	[0.0, 4.49]
Al	[0.29, 3.5]
Si	[69.81, 75.41]
K	[0.0, 6.21]
Ca	[5.43, 16.19]
Ва	[0.0, 3.15]
Fe	[0.0, 0.51]
TypeGlass	{1, 2, 3, 4, 5, 6, 7}

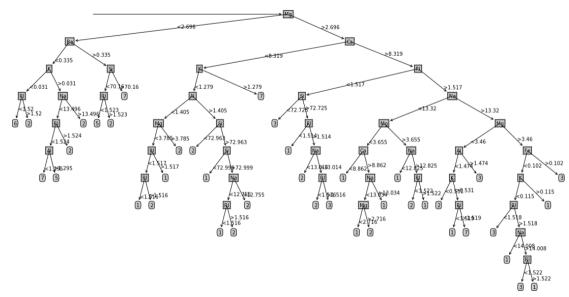
数据集中每个属性的值都为连续值, 共有 9 个特征, 7 种类别, 共 214 条数据, 所有数据都完整无缺失值。整个数据集按照约 9:1 的比例来随机划分训练集和测试集, 即训练集 191 条, 测试集 23 条。

五、 实验结果、结果分析和实验结论

构建树后,离散值决策树如下图所示,该树的分类准确度为66.7%。



构建树后,连续值决策树如下图所示,该树的分类准确度为 65.2%。



可以看出,决策树作为一个较为基本的分类器,在本次实验中准确度并不是太高。但在实验过程中,我们并未考虑到过拟合问题以及决策树的剪枝操作,对于实验结果有较大影响,因此整个实验还有很大的优化空间,对于决策树的知识,我们可以进行更深一步的学习和完善。

六、 小结

在进行了决策树分类实验后,我对信息熵、信息增益以及决策树构建的相关知识,还有 ID3、C4.5、CART 算法等都有了初步了解,而通过实际上手编写代码实现决策树分类器,我理解了决策树的基本原理和基本的程序实现方法。对于机器学习和数据挖掘知识,我有了更浓厚的兴趣,并希望在今后可以进行更多实践探索。

参考文献

- [1] 蒋盛益, 李霞, 郑琪.数据挖掘原理与实践[M].北京: 电子工业出版社, 2013
- [2] 李航.统计学习方法[M].北京:清华大学出版社,2012:55-58.