

# 《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。<sup>[1]</sup>

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。<sup>[2]</sup>

#### · 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： $P(U = u_i) = p_i, \quad i = 1, 2, \dots, n$  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： $P(U = u_i, V = v_j) = p_{ij}, \quad i = 1, 2, \dots, n; \quad j = 1, 2, \dots, m$  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

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

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。

```
###
#从 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)
```

```

labelCounts = countFeatClass(dataSet,-1)      #即统计各类结果的个数
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):
    S = calcEnt(dataSet)                        #父节点的香农熵
    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
        p = float(labelCounts[key])/len(dataSet)        #当前分化所占权重比例
        sumOfKidsEnt += p*EntOfKid                      #累加计算子节点总 Ent
    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

```



#### 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			
Type	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-O2	{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			
Type	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]
Ba	[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，该树的分类准确率为 66.7% 。



