** **

**决策树分类算法**

院系名称 自动化科学与电气工程学院

专业名称 模式识别与智能系统

学生学号 15031184

学生姓名 李思奇

2018年5月25日

# 1. 简介

决策树（Decision Tree，DT）是最简单也是最成功的学习算法之一，它由内部和外部结点组成，结点之间的互连称为树的分支。内部结点是一个决策单元，根据相关变量的不同可能值，决定下一步访问哪个子结点。相反，外部结点也称为叶结点，是分支的终止结点。它没有子结点，但是与描述给定数据的类别相关联。决策树是一组相关的规则所形成的树状结构，每个分支都是与其子结点相关的决策规则。

# 2. 基本原理

决策树对样本进行分类，将它们从根结点分类到叶结点。这种树形结构的分类器将数据集的输入空间递归地划分为互斥的空间，在经过分类后，每个训练数据被标识为属于某个子空间，该子空间会被定义一个标签或一个值来描述。决策树具有很好的透明性，我们可以很容易地根据决策树的结构来解释决策是如何做出的。

随机变量的熵是通过观察其值产生的平均信息量。考虑概率为0.9的掷硬币随机实验，P(Head) = 0.9, P(Tail) = 0.1。这比P(Head) = 0.5和P(Tail) = 0.5的情况提供了更多的信息。熵是用来评价物理中的随机性的，熵值越大说明这个过程越随机。决策树根据每个属性的信息熵进行启发式的引导。信息熵用于评价每个属性的表征;作为一种分类方法。假设我们有m个类，对于一个特定的属性，我们用pi表示它属于类Ci的比例，i = 1,2,…,m。

则这个属性的信息熵是：

 (1-1)

我们也可以说熵是一种对于训练样本集中杂质的度量：熵越大，数据越不纯。基于熵，信息增益(IG)式用来衡量一个属性的有效性，作为区分类的手段。

 (1-2)

所有的例子都根据a的值分成几个组（例如*Sv* for *v* ∈ *Values(A)*），这仅仅是根据这个属性对例子进行划分所导致的熵的预期减少。

# 3. 基础实验

使用UCI Machine Learning Repository提供的Iris Dataset来构造一棵决策树。该样本集样本个数N=150，类别分为Iris-setosa, Iris-versicolor, Iris-virginia。属性分为Sepal length (SL)、Sepal width (SW)、Petal length (PL)、Petal width(PW)。

## 3.1 连续值处理

对于本数据集，其四种属性的取值认为是连续值。由于连续属性的可取值不再是有限数目。因此，可以采用连续属性的离散化技术。在本文中，采用最简单的二分法对连续属性进行处理。具体的处理方法如下。

给定样本集D和连续属性a，假定a在D上出现了n个不同的取值，将这些值从小到大进行排序，记为。基于划分点t可以将D分为子集和，其中包含那些在属性a上取值不大于t的样本，而包含那些在属性a上取值大于t的样本。对于连续属性a，我们可以考察n-1个元素的候选划分点集合：

 (3-1)

即，把区间的中位点作为候选划分点。然后就可以像离散属性值一样来考察这些划分点。对式(1-2)进行改变，可得：

 (3-3)

## 3.2 根据信息增益(IG)构造决策树

### 3.2.1 计算根结点

根据式(3-3)，计算出样本集各个属性的信息增益，选择信息增益最大的作为根结点的划分属性。

经计算，根结点以PL为划分属性，信息增益为0.9183，划分值为2.45。其中，PL>2.45的部分信息熵为0.6667，PL<2.45的部分信息熵为0。这意味着，PL<2.45的部分只有一类，即PL<2.45的部分被判定为Iris-setosa类。此时的决策树如图3.2.1所示。

PL<2.45?

待计算

是

否

**图3.2.1 生成的根结点**

### 3.2.2 计算内部结点

对于PL>2.45的数据集，计算出样本集各个属性的信息增益，选择信息增益最大的作为叶结点的划分属性。

经计算，叶结点以PW为划分属性，信息增益为0.6902，划分值为1.75。其中，PW>1.75的部分信息熵为0.0695，PW<1.75的部分信息熵为0.2403。

对于PW>1.75的部分计算其各个属性的信息增益。选择PL为划分属性，信息增益为0.0912，划分值为4.85。其中PL>4.85的部分信息熵为0，PL<4.85的部分信息熵为0.0599。这意味着在此结点下PL>4.85的部分种类单一，即PL>4.85的部分被判定为Iris-virginia类。

再对PL<4.85的部分进行计算，选择SW为划分属性，SW>3.1的部分被判定为Iris-versicolor类，SW<3.1的部分被判定为Iris-virginia类。

此时的决策树如图3.2.2所示。

PL<2.45?

PW>1.75?

Y

N

PL>4.85?

待计算

SW>3.1?

Y

N

Y

N

N

Y

**图3.2.2 生成部分叶结点**

对于PW<1.75的部分计算其各个属性的信息增益。选择PL为划分属性，信息增益为0.2132，划分值为4.95。其中PL>4.95的部分信息熵为0.102，PL<4.95的部分信息熵为0.1299。

对于PL>4.95的部分计算。选择PW为划分属性，信息增益为0.4591，划分值为1.55。其中PW>1.55的部分信息熵为0.4591，PW<1.55的部分信息熵为0，被划分为Iris-virginia类。对于PW>1.55的部分进行划分。选择PL进行划分，PL>5.45被划分为Iris-virginia类，PL<5.45被划分为Iris-versicolor类。

对于PL<4.95的部分计算。选择PW为划分属性，信息增益为0.1461，划分值为1.65。其中PW>1.65的部分被分为Iris-virginia类，PW<1.65的部分被分为Iris-versicolor类。

做出最终的决策树如图3.2.3所示。

PL<2.45?

PW>1.75?

Y

N

PL>4.85?

PL>4.95?

SW>3.1?

Y

N

Y

N

N

Y

PW>1.55??

PW>1.65?

PL>5.45?

Y

Y

Y

Y

N

N

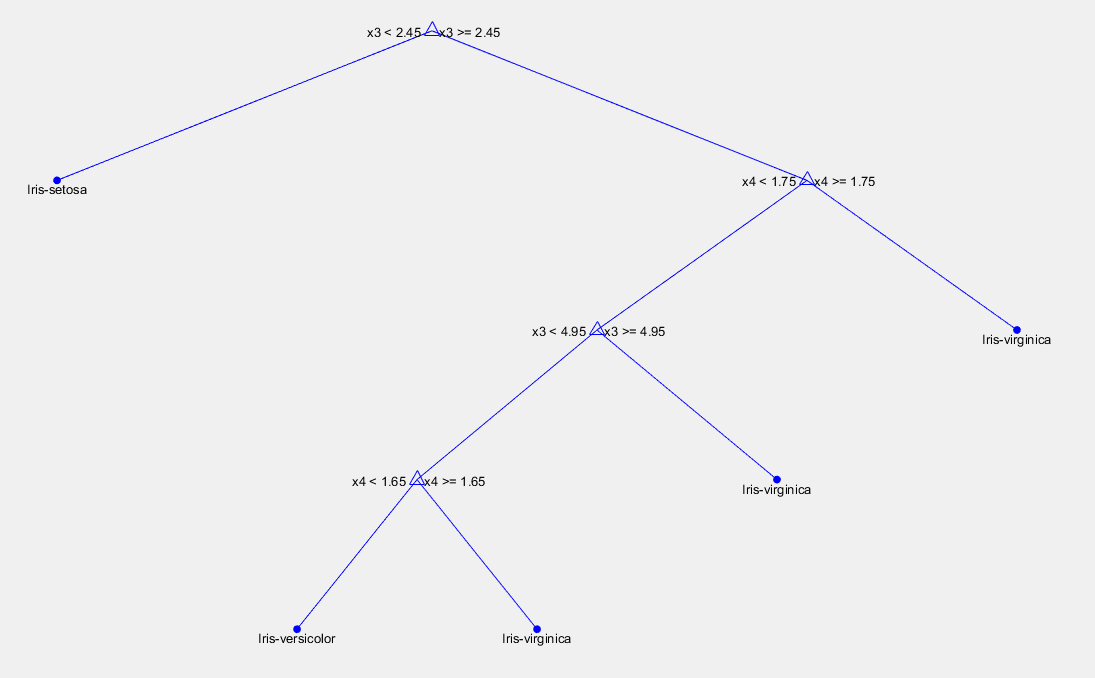
N

N

**图3.2.2 最终生成的决策树**

## 3.3 利用Matlab中的决策树工具箱生成决策树

直接调用Matlab中的决策树工具箱，生成决策树，如图3.3所示。其中x3为PL属性，x4为PW属性。



**图3.3 利用决策树工具箱生成决策树**

## 3.4 对比分析

通过对比自己计算的决策树与调用Matlab工具箱得到的决策树，可以看出自己计算的决策树的结构相当的复杂。

通过分析发现，我自己计算得到的决策树结构中，有两个分支中（标红的两支）都只有一个数据，属于过拟合的现象。若将这两支剪去，对于准确性影响较小。若进行剪枝处理，处理后的决策树与利用工具箱生成的决策树基本一致。

# 4. 问题讨论

## 4.1连续值离散化处理

对于连续属性的离散化处理是在实际应用中常见的问题。一般处理的方法如前文3.1节中所述，即C4.5决策树算法。

## 4.2 结构复杂度与精确度

对于一棵决策树，树的大小和模型的准确性之间存在着一种需要权衡的关系。在决策树的学习中，为了尽可能正确的分类训练集中的样本，结点划分的过程将要不断重复。一般情况下，这样可以使得模型准确性得到提升，但是却是以增大树的规模为代价。但是有些时候，由于决策树的分支太多，会导致把训练集的一些特质当作所有数据都有的一般性质而导致过度拟合。在这种情况下，尽管训练集的精度可能非常高，测试集的性能可能不理想。

剪枝是在紧凑和性能之间达到平衡的一种典型方法。决策树剪枝的基本策略有“预剪枝(prepruning)”和“后剪枝(post-pruning)”。

预剪枝是指在决策树生成过程中，对每个结点在划分之前先进行估计，若当前结点的划分不能带来决策树泛化性能的提升，则停止划分并将当前结点标记为叶节点。

后剪枝是先从训练集生成一棵完整的决策树，然后自底向上地对于非叶结点进行考察，若将该结点对应的子树替换为叶结点可以带来决策树泛化性能提升，则将该子树替换为叶结点。

4.3

在经典决策树中，每个样本完全属于一个类，但是软方法使样本可以被看作是几个部分。边界区域外的样本被视为正常，而其他样本则采用软分类处理。在边界层中，样本的概率密度成为预测变量，已知的类比例成为目标变量。通过这种方法，我们可以使用数据集构造树，结果就是我们想要的结果。

在典型的硬决策树中，每个决策节点都选择一个子节点。因此，从根到其中一个叶子的一条路径是遍历的。

然而，在软决策树中，所有的孩子都是被选中的，但都有一定的概率。也就是说，我们沿着所有的路径走到所有的叶子上所有的叶子都有助于最终的决定但是有不同的可能性。

软决策树有几个优点。首先，它们提供了软响应，而硬树在叶边界处有不连续的响应。这使得软决策树能够更平滑地适应，从而降低了分割边界附近的偏差。其次，一个线性的门控功能使得软树可以与硬树的轴-正交劈裂形成对比。

## 4.4 与朴素贝叶斯分类器相比的优缺点

决策树算法与朴素贝叶斯分类器相比的优点有：

(1) 决策树非常的灵活、可视性强、更直观、易于解释、调试方便，可以较容易的找出错误；

(2) 需要准备的数据量不大，而其他的技术往往需要很大的数据集，需要创建虚拟变量，去除不完整的数据，但是该算法对于丢失的数据不能进行准确的预测；

(3) 在相对短的时间内能够对大型数据源做出可行且效果良好的结果；

(4) 能够处理数字和数据的类别（需要做相应的转变），而其他算法分析的数据集往往是只有一种类型的变量；

(5) 可以很方便的和其他分类器相结合。

决策树分类算法的缺点有：

(1) 分类结果容易受到外界干扰的影响，容易过度拟合；

(2) 如果许多值不确定或者许多结果相互关联，则计算可能会变得非常复杂；

(3) 当有些属性相关联时无法处理；

(4) 对连续性的属性比较难预测。

# 附录：MATLAB程序代码

主函数：

clear;clc;

[SL,SW,PL,PW,T] = textread('data.txt','%f%f%f%f%s',150,'delimiter',',');

data=[SL,SW,PL,PW];

n = length(T);

for i=1:n/3

Type(i,1) = 1; Type(i+n/3,1) = 2; Type(i+2\*n/3,1) = 3;

end

[NodeNum,edge,E1,E2,g] = CNode(data,Type); t1=find(data(:,NodeNum)<edge);t2=find(data(:,NodeNum)>edge);

data11 = data(t1,:);data12 = data(t2,:);

Type11 = Type(t1,:);Type12 = Type(t2,:);

[NodeNum\_1,edge\_1,E1\_1,E1\_2,g1] = CNode(data12,Type12);

t1=find(data12(:,NodeNum\_1)<edge\_1);t2=find(data12(:,NodeNum\_1)>edge\_1);

data21 = data12(t1,:);data22 = data12(t2,:);

Type21 = Type12(t1,:);Type22 = Type12(t2,:);

[NodeNum\_21,edge\_21,E1\_21,E1\_22,g21] = CNode(data22,Type22);

t1=find(data22(:,NodeNum\_21)<edge\_21);t2=find(data22(:,NodeNum\_21)>edge\_21);

data211 = data22(t1,:);data212 = data22(t2,:);

Type211 = Type22(t1,:);Type212 = Type22(t2,:);

[NodeNum\_2,edge\_2,E2\_1,E2\_2,g2] = CNode(data21,Type21);

t1=find(data21(:,NodeNum\_2)<edge\_2);t2=find(data21(:,NodeNum\_2)>edge\_2);

data31 = data21(t1,:);data32 = data21(t2,:);

Type31 = Type21(t1,:);Type32 = Type21(t2,:);

[NodeNum\_3,edge\_3,E3\_1,E3\_2,g3] = CNode(data32,Type32);

t1=find(data32(:,NodeNum\_3)<edge\_3);t2=find(data32(:,NodeNum\_3)>edge\_3);

data41 = data32(t1,:);data42 = data32(t2,:);

Type41 = Type32(t1,:);Type42 = Type32(t2,:);

t1=find(data32(:,NodeNum\_3)<edge\_3);t2=find(data32(:,NodeNum\_3)>edge\_3);

data41 = data32(t1,:);data42 = data32(t2,:);

Type41 = Type32(t1,:);Type42 = Type32(t2,:);

求信息熵：

function EntD = CEntD(x)

EntD = 0;

TotalNum = length(x);

A = unique(x);

PropertyNum = length(A);

for i=1:PropertyNum

n1=length(find(x==A(i)));

p1=n1/TotalNum;

EntD=EntD-p1\*log2(p1);

end

end

求最大信息增益：

function [NodeNum,edge,E1,E2,g]=CNode(data,Type)

EntD=CEntD(Type);

[m,n]=size(data);

Gain=EntD\*ones(1,n);

for i=1:n

data1=data(:,i);

A1=unique(data1);

n1=length(A1);

for j=1:n1-1

A(j)=(A1(j)+A1(j+1))/2;

end

A=A(1:n1-1);

for p=1:length(A)

t1=find(data1>A(p));t2=find(data1<A(p));

type1 = Type(t1);type2 = Type(t2);

Ent1(p) = CEntD(type1)\*length(type1)/m;

Ent2(p) = CEntD(type2)\*length(type2)/m;

G(p) = EntD - Ent1(p) - Ent2(p);

end

G = G(1:n1-1);

[Y,I] = max(G);

Gain(i) = Y;

edge(i) = A(I);

E1(i) = Ent1(I);E2(i) = Ent2(I);

end

[Y,I] = max(Gain);

g = Y;

NodeNum = I;

edge = edge(I);

E1= E1(I);E2 = E2(I);

end