|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 课程名称 | 机器学习与数据挖掘 | | | | |
| 实验名称 | 决策树学习分类 | | | | |
| 实验时间 | 2021/10/22 | | 实验地点 | 12J661-2 | |
| 学 院 | 信息科学与工程学院 | | 专 业 | 计算机科学与技术 | |
| 姓 名 | 吴春城 | 班 级 | 计科1903 | 学 号 | 201931222121 |
| 同组实验者  姓名 |  | | | | |
| 实验成绩 |  | | 指导教师  （签字） |  | |
| **实验报告内容基本要求参考格式**  一、实验目的  二、实验环境  三、实验步骤/过程  四、实验结果  五、实验分析及反馈 | | | | | |
| **实验目的**  1. 学习决策树分类算法原理 2. 熟练掌握信息熵以及信息增益的计算公式 3. 了解决策树的剪枝方法（预剪枝和后剪枝） 4. 熟悉连续属性离散化技术 5. **实验环境**   matlab R2018a   1. **实验步骤**   **3.1 训练集预处理**  **%从iris.txt文件中加载数据**   |  | | --- | | Sample = importdata('iris.txt');  n = length(Sample); | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **%处理数据，标记所属类别为Num**   |  | | --- | | sample = zeros(n,5);  for i=1:n  S = regexp(char(Sample(i)) ,',', 'split');  for j = 1:4  sample(i,j) = str2double(char(S(j)));  end  if strcmpi(S(5),"Iris-setosa")  sample(i,5) = 0;  elseif (strcmpi(S(5),"Iris-versicolor"))  sample(i,5) = 1;  else  sample(i,5) = 2;  end  end |   **%调用maketree函数创建决策树**   |  | | --- | | maketree(sample); |   **3.2如何生成决策树（maketree函数）**  **%设置树的高度为2**   |  | | --- | | [sample1,sample2] = childtree(sample,0);  [sample3,sample4] = childtree(sample1,1);  [sample5,sample6] = childtree(sample2,2); |   **%为每个节点创建结构体(自下而上)**   |  | | --- | | tree\_c3=struct(  %父节点  'parent',1,  %所属类别  'class',decideclass(sample3),  %节点值  'value',sample3,  %子节点1  'child1',0,  %子节点2  'child2',0);  tree\_c4...  tree\_c5...  tree\_c6...  tree\_c1...  tree\_c2...  tree\_root... |   **%分别测试叶节点tree\_c3\c4\c5\c6的正确率均值**   |  | | --- | | r = (testTree(sample3)+testTree(sample4)+testTree(sample5)+testTree(sample6))/4; |   **%输出分类结果**   |  | | --- | | fprintf('共分为四类\n样本数量分别为：  %i，%i，%i，%i\n',length(sample3(:,1)),length(sample4(:,1))-1,length(sample5(:,1)),length(sample6(:,1))-1);  fprintf('分类结果分别为：  %i，%i，%i，%i\n',tree\_c3.class,tree\_c4.class,tree\_c5.class,tree\_c6.class);  fprintf('该决策树的正确率为：%.2f\n',r); |   **%后剪枝（num代表剪枝次数）**   |  | | --- | | if testTree(tree\_root.value)>=(testTree(tree\_root.child1.value)+testTree(tree\_root.child2.value))/2  tree\_root.child1 = 0;  tree\_root.child2 = 0;  num=num+1;  end  %内部节点1  if testTree(tree\_c1.value)>=(testTree(tree\_c1.child1.value)+testTree(tree\_c1.child2.value))/2  tree\_c1.child1 = 0;  tree\_c1.child2 = 0;  num=num+1;  end  %内部节点2  if testTree(tree\_c2.value)>=(testTree(tree\_c2.child1.value)+testTree(tree\_c2.child2.value))/2  tree\_c2.child1 = 0;  tree\_c2.child2 = 0;  num=num+1; |   **%输出剪枝结果**   |  | | --- | | fprintf('完成剪枝%i次\n',num);  r = (testTree(sample1)+testTree(sample2))/2;  fprintf('正确率变为：%.2f\n',r); |   **3.3如何生成孩子节点（childtree函数）**  **%判断是否能生成孩子节点**   |  | | --- | | if (n ==1 || testTree(sample)==1)只有一个样本或者正确率为1  sample1 = sample;  sample2 = 0;  elseif n==0该节点没有样本  sample1 = 0;  sample2 = 0;  else  G = zeros(1,4);  T = zeros(1,4);  for i=1:4  [G(i),T(i)] = Gain(sample,i,n); 计算每一个属性的信息增益  end  [Gb,index] = max(G);选择最大的信息增益对应的属性作为最佳分类线  Tb = T(index); |   **%输出最大信息增益属性信息**   |  | | --- | | switch time  case 0  fprintf('对于根节点，采用第 %i个特征对样本进行分类，最优分类标准T为：%.2f，信息增益为%.2f\n',index, Tb,Gb);  case 1  fprintf('对于内部节点1，采用第 %i个特征对样本进行分类，最优分类标准T为：%.2f，信息增益为%.2f\n',index, Tb,Gb);  case 2  fprintf('对于内部节点2，采用第 %i个特征对样本进行分类，最优分类标准T为：%.2f，信息增益为%.2f\n',index, Tb,Gb);  end |   **%软边界分离节点**   |  | | --- | | [sample1,sample2] = dividetree\_soft(sample,index,Tb); |   **3.4 如何求信息增益（Gain函数-连续型）**  **%根据公式计算信息熵**   |  | | --- | | function E = entropy(sample,f)  [n,~] = size(sample);  x0 = 0;  x1 = 0;  x2 = 0;  for i = 1:n  switch sample(i,f)  case 0  x0 =x0+1;  case 1  x1 =x1+1;  case 2  x2 =x2+1;  end  end  p0 = x0/n;  p1 = x1/n;  p2 = x2/n;    if (p0 ==0)  s(1) = 0;  else  s(1)=(p0\*log2(p0));  end    if (p1 ==0)  s(2) = 0;  else  s(2)=(p1\*log2(p1));  end    if (p2 ==0)  s(3) = 0;  else  s(3)=(p2\*log2(p2));  end    E =-sum(s);  end |   **%连续信息离散化**   |  | | --- | | A = sort(unique(sample(:,f)));去掉属性中重复样本、升序排序  T = (A(1:end-1)+A(2:end))/2;取相邻两个样本中间点作为分隔点 |   **%计算T中每个分隔点的信息增益，选出最大的那个**   |  | | --- | | num = length(T);  G = zeros(1,num); %每一种可能的信息增益  for j = 1:num  [s1,s2] = dividetree\_soft(sample,f,T(j));  G(j) = entropy(sample,5)-(length(s1(:,1))/n\*entropy(s1,5)+length(s2(:,1))/n\*entropy(s2,5));  end  [Gbest,index] = max(G);  Tbest = T(index); |   **3.5 如何根据某个属性分离树（软边界和硬边界）**  **%软边界**   |  | | --- | | function [sample1,sample2] = dividetree\_soft(sample,f,T)  a=1;  b=1;  [n,~] = size(sample);  x1 = min(sample(:,f));  x2 = max(sample(:,f));  range = (x2-x1)/10;    for i = 1:n %实现分类  if (sample(i,f) < T-range)  sample1(a,:) = sample(i,:);  a = a + 1;  elseif(sample(i,f) > T+range)  sample2(b,:) = sample(i,:);  b = b + 1;  else  sample1(a,:) = sample(i,:);  sample2(b,:) = sample(i,:);  a = a + 1;  b = b + 1;  end  end    end |   **%硬边界**   |  | | --- | | function [sample1,sample2] = dividetree\_hard(sample,f,T)  a=1;  b=1;  [n,~] = size(sample);  for i = 1:n %实现分类  if (sample(i,f) < T)  sample1(a,:) = sample(i,:);  a = a+1;  else  sample2(b,:) = sample(i,:);  b=b + 1;  end  end  end |   **%离散属性边界**   |  | | --- | | function [sample1,sample2] = dividetree\_discrete(sample,f,T)  a=1;  b=1;  [n,~] = size(sample);  for i = 1:n %实现分类  if (sample(i,f) == T)  sample1(a,:) = sample(i,:);  a = a+1;  else  sample2(b,:) = sample(i,:);  b=b + 1;  end  end  end |  1. **实验结果**   **4.1 软边界分离子树**    **4.2 硬边界分离子树**     1. **实验分析与反馈** 2. 本实验重点在于如何对连续属性离散化求最佳信息增益，我采用二分法，去相邻两个实际值的中点作为阈值求信息增益，求出最大信息增益对应的阈值就是分离子树的指标。 3. 离散属性的信息增益算法，因为本实验数据集属性都是连续性，于是无法测试。 4. 对于连续取值属性，软边界和硬边界算法相比较而言，软边界分离子树更加精准，但在本次实验两种方法的正确率实验结果都是0.90，没有体现优劣之分。 |

|  |
| --- |
|  |

|  |
| --- |
|  |