答案1

clear;clc;

% OutlookType=struct('Sunny',1,'Rainy',2,'Overcast',3);

% TemperatureType=struct('hot',1,'warm',2,'cool',3);

% HumidityType=struct('high',1,'norm',2);

% WindyType={'True',1,'False',0};

% PlayGolf={'Yes',1,'No',0};

% data=struct('Outlook',[],'Temperature',[],'Humidity',[],'Windy',[],'PlayGolf',[]);

Outlook=[1,1,3,2,2,2,3,1,1,2,1,3,3,2]';

Temperature=[1,1,1,2,3,3,3,2,3,3,2,2,1,2]';

Humidity=[1,1,1,1,2,2,2,1,2,2,2,1,2,1]';

Windy=[0,1,0,0,0,1,1,0,0,0,1,1,0,1]';

data=[Outlook Temperature Humidity Windy];

PlayGolf=[0,0,1,1,1,0,1,0,1,1,1,1,1,0]';

propertyName={'Outlook','Temperature','Humidity','Windy'};

delta=0.1;

decisionTreeModel=decisionTree(data,PlayGolf,propertyName,delta);

label=decisionTreeTest(decisionTreeModel,data,propertyName);

1.

function decisionTreeModel=decisionTree(data,label,propertyName,delta)

global Node;

Node=struct('fatherNodeName',[],'EdgeProperty',[],'NodeName',[]);

BuildTree('root','Stem',data,label,propertyName,delta);

Node(1)=[];

model.Node=Node;

decisionTreeModel=model;

2.

function BuildTree(fatherNodeName,edge,data,label,propertyName,delta)

%UNTITLED9 Summary of this function goes here

% Detailed explanation goes here

global Node;

sonNode=struct('fatherNodeName',[],'EdgeProperty',[],'NodeName',[]);

sonNode.fatherNodeName=fatherNodeName;

sonNode.EdgeProperty=edge;

if length(unique(label))==1

sonNode.NodeName=label(1);

Node=[Node sonNode];

return;

end

if length(propertyName)<1

labelSet=unique(label);

labelNum=length(labelSet);

for i=1:labelNum

labelNum=length(find(label==labelSet(i)));

end

[~,labelIndex]=max(labelNum);

sonNode.NodeName=labelSet(labelIndex);

Node=[Node sonNode];

return;

end

[sonIndex,BuildNode]=CalcuteNode(data,label,delta);

if BuildNode

dataRowIndex=setdiff(1:length(propertyName),sonIndex);

sonNode.NodeName=propertyName(sonIndex);

Node=[Node sonNode];

propertyName(sonIndex)=[];

sonData=data(:,sonIndex);

sonEdge=unique(sonData);

for i=1:length(sonEdge)

edgeDataIndex=find(sonData==sonEdge(i));

BuildTree(sonNode.NodeName,sonEdge(i),data(edgeDataIndex,dataRowIndex),label(edgeDataIndex,:),propertyName,delta);

end

else

labelSet=unique(label);

labelNum=length(labelSet);

for i=1:labelNum

labelNum=length(find(label==labelSet(i)));

end

[~,labelIndex]=max(labelNum);

sonNode.NodeName=labelSet(labelIndex);

Node=[Node sonNode];

return;

end

3

function [NodeIndex,BuildNode]=CalcuteNode(data,label,delta)

LargeEntropy=CEntropy(label);

[m,n]=size(data);

EntropyGain=LargeEntropy\*ones(1,n);

BuildNode=true;

for i=1:n

pData=data(:,i);

itemList=unique(pData);

for j=1:length(itemList)

itemIndex=find(pData==itemList(j));

EntropyGain(i)=EntropyGain(i)-length(itemIndex)/m\*CEntropy(label(itemIndex));

end

% 此处运行则为增益率，注释掉则为增益

% EntropyGain(i)=EntropyGain(i)/CEntropy(pData);

end

[maxGainEntropy,NodeIndex]=max(EntropyGain);

if maxGainEntropy<delta

BuildNode=false;

end

4.

function result=CEntropy(propertyList)

result=0;

totalLength=length(propertyList);

itemList=unique(propertyList);

pNum=length(itemList);

for i=1:pNum

itemLength=length(find(propertyList==itemList(i)));

pItem=itemLength/totalLength;

result=result-pItem\*log2(pItem);

end

答案2

1. function [Tree RulesMatrix]=DecisionTree(DataSet,AttributName)

%ÊäÈëÎªÑµÁ·¼¯£¬ÎªÀëÉ¢ºóµÄÊý×Ö£¬Èç¼ÇÂ¼1£º1 1 3 2 1£»

%Ç°ÃæÎªÊôÐÔÁÐ£¬×îºóÒ»ÁÐÎªÀà±ê

if nargin<1

error('ÇëÊäÈëÊý¾Ý¼¯');

else

if isstr(DataSet)

[DataSet AttributValue]=readdata2(DataSet);

else

AttributValue=[];

end

end

if nargin<2

AttributName=[];

end

Attributs=[1:size(DataSet,2)-1];

Tree=CreatTree(DataSet,Attributs);

disp([char(13) 'The Decision Tree:']);

showTree(Tree,0,0,1,AttributValue,AttributName);

Rules=getRule(Tree);

RulesMatrix=zeros(size(Rules,1),size(DataSet,2));

for i=1:size(Rules,1)

rule=cell2struct(Rules(i,1),{'str'});

rule=str2num([rule.str([1:(find(rule.str=='C')-1)]) rule.str((find(rule.str=='C')+1):length(rule.str))]);

for j=1:(length(rule)-1)/2

RulesMatrix(i,rule((j-1)\*2+1))=rule(j\*2);

end

RulesMatrix(i,size(DataSet,2))=rule(length(rule));

end

end

function Tree=CreatTree(DataSet,Attributs) %¾ö²ßÊ÷³ÌÐò ÊäÈëÎª£ºÊý¾Ý¼¯£¬ÊôÐÔÃûÁÐ±í

%disp(Attributs);

[S ValRecords]=ComputEntropy(DataSet,0);

if(S==0) %µ±ÑùÀýÈ«ÎªÒ»ÀàÊ±ÍË³ö£¬·µ»ØÒ¶×Ó½ÚµãÀà±ê

for i=1:length(ValRecords)

if(length(ValRecords(i).matrix)==size(DataSet,1))

break;

end

end

Tree.Attribut=i;

Tree.Child=[];

return;

end

if(length(Attributs)==0) %µ±Ìõ¼þÊôÐÔ¸öÊýÎª0Ê±·µ»ØÕ¼¶àÊýµÄÀà±ê

mostlabelnum=0;

mostlabel=0;

for i=1:length(ValRecords)

if(length(ValRecords(i).matrix)>mostlabelnum)

mostlabelnum=length(ValRecords(i).matrix);

mostlabel=i;

end

end

Tree.Attribut=mostlabel;

Tree.Child=[];

return;

end

for i=1:length(Attributs)

[Sa(i) ValRecord]=ComputEntropy(DataSet,i);

Gains(i)=S-Sa(i);

AtrributMatric(i).val=ValRecord;

end

[maxval maxindex]=max(Gains);

Tree.Attribut=Attributs(maxindex);

Attributs2=[Attributs(1:maxindex-1) Attributs(maxindex+1:length(Attributs))];

for j=1:length(AtrributMatric(maxindex).val)

DataSet2=[DataSet(AtrributMatric(maxindex).val(j).matrix',1:maxindex-1) DataSet(AtrributMatric(maxindex).val(j).matrix',maxindex+1:size(DataSet,2))];

if(size(DataSet2,1)==0)

mostlabelnum=0;

mostlabel=0;

for i=1:length(ValRecords)

if(length(ValRecords(i).matrix)>mostlabelnum)

mostlabelnum=length(ValRecords(i).matrix);

mostlabel=i;

end

end

Tree.Child(j).root.Attribut=mostlabel;

Tree.Child(j).root.Child=[];

else

Tree.Child(j).root=CreatTree(DataSet2,Attributs2);

end

end

end

function [Entropy RecordVal]=ComputEntropy(DataSet,attribut) %¼ÆËãÐÅÏ¢ìØ

if(attribut==0)

clnum=0;

for i=1:size(DataSet,1)

if(DataSet(i,size(DataSet,2))>clnum) %·ÀÖ¹ÏÂ±êÔ½½ç

classnum(DataSet(i,size(DataSet,2)))=0;

clnum=DataSet(i,size(DataSet,2));

RecordVal(DataSet(i,size(DataSet,2))).matrix=[];

end

classnum(DataSet(i,size(DataSet,2)))=classnum(DataSet(i,size(DataSet,2)))+1;

RecordVal(DataSet(i,size(DataSet,2))).matrix=[RecordVal(DataSet(i,size(DataSet,2))).matrix i];

end

Entropy=0;

for j=1:length(classnum)

P=classnum(j)/size(DataSet,1);

if(P~=0)

Entropy=Entropy+(-P)\*log2(P);

end

end

else

valnum=0;

for i=1:size(DataSet,1)

if(DataSet(i,attribut)>valnum) %·ÀÖ¹²ÎÊýÏÂ±êÔ½½ç

clnum(DataSet(i,attribut))=0;

valnum=DataSet(i,attribut);

Valueexamnum(DataSet(i,attribut))=0;

RecordVal(DataSet(i,attribut)).matrix=[]; %½«±àºÅ±£ÁôÏÂÀ´£¬ÒÔ·½±ãºóÃæ°´Öµ·Ö¸îÊý¾Ý¼¯

end

if(DataSet(i,size(DataSet,2))>clnum(DataSet(i,attribut))) %·ÀÖ¹ÏÂ±êÔ½½ç

Value(DataSet(i,attribut)).classnum(DataSet(i,size(DataSet,2)))=0;

clnum(DataSet(i,attribut))=DataSet(i,size(DataSet,2));

end

Value(DataSet(i,attribut)).classnum(DataSet(i,size(DataSet,2)))= Value(DataSet(i,attribut)).classnum(DataSet(i,size(DataSet,2)))+1;

Valueexamnum(DataSet(i,attribut))= Valueexamnum(DataSet(i,attribut))+1;

RecordVal(DataSet(i,attribut)).matrix=[RecordVal(DataSet(i,attribut)).matrix i];

end

Entropy=0;

for j=1:valnum

Entropys=0;

for k=1:length(Value(j).classnum)

P=Value(j).classnum(k)/Valueexamnum(j);

if(P~=0)

Entropys=Entropys+(-P)\*log2(P);

end

end

Entropy=Entropy+(Valueexamnum(j)/size(DataSet,1))\*Entropys;

end

end

end

function showTree(Tree,level,value,branch,AttributValue,AttributName)

blank=[];

for i=1:level-1

if(branch(i)==1)

blank=[blank ' |'];

else

blank=[blank ' '];

end

end

blank=[blank ' '];

if(level==0)

blank=[' (The Root):'];

else

if isempty(AttributValue)

blank=[blank '|\_\_\_\_\_' int2str(value) '\_\_\_\_\_\_'];

else

blank=[blank '|\_\_\_\_\_' value '\_\_\_\_\_\_'];

end

end

if(length(Tree.Child)~=0) %·ÇÒ¶×Ó½Úµã

if isempty(AttributName)

disp([blank 'Attribut ' int2str(Tree.Attribut)]);

else

disp([blank 'Attribut ' AttributName{Tree.Attribut}]);

end

if isempty(AttributValue)

for j=1:length(Tree.Child)-1

showTree(Tree.Child(j).root,level+1,j,[branch 1],AttributValue,AttributName);

end

showTree(Tree.Child(length(Tree.Child)).root,level+1,length(Tree.Child),[branch(1:length(branch)-1) 0 1],AttributValue,AttributName);

else

for j=1:length(Tree.Child)-1

showTree(Tree.Child(j).root,level+1,AttributValue{Tree.Attribut}{j},[branch 1],AttributValue,AttributName);

end

showTree(Tree.Child(length(Tree.Child)).root,level+1,AttributValue{Tree.Attribut}{length(Tree.Child)},[branch(1:length(branch)-1) 0 1],AttributValue,AttributName);

end

else

if isempty(AttributValue)

disp([blank 'leaf ' int2str(Tree.Attribut)]);

else

disp([blank 'leaf ' AttributValue{length(AttributValue)}{Tree.Attribut}]);

end

end

end

function Rules=getRule(Tree)

if(length(Tree.Child)~=0)

Rules={};

for i=1:length(Tree.Child)

content=getRule(Tree.Child(i).root);

%disp(content);

%disp([num2str(Tree.Attribut) ',' num2str(i) ',']);

for j=1:size(content,1)

rule=cell2struct(content(j,1),{'str'});

content(j,1)={[num2str(Tree.Attribut) ',' num2str(i) ',' rule.str]};

end

Rules=[Rules;content];

end

else

Rules={['C' num2str(Tree.Attribut)]};

end

end

2. function [ matrix,attributes,activeAttributes ] = id3\_preprocess( )

%% ID3Ëã·¨Êý¾ÝÔ¤´¦Àí£¬°Ñ×Ö·û´®×ª»»Îª0,1±àÂë

% Êä³ö²ÎÊý£º

% matrix£º ×ª»»ºóµÄ0,1¾ØÕó£»

% attributes: ÊôÐÔºÍLabel£»

% activeAttributes : ÊôÐÔÏòÁ¿£¬È«1£»

%% ¶ÁÈ¡Êý¾Ý

txt = { 'ÐòºÅ' 'Outlook' 'Temperature' 'Humidity' 'Wind' 'PlayTennis'

'' 'Sunny' 'Hot' 'High' 'Weak' 'No'

'' 'Sunny' 'Hot' 'High' 'Strong' 'No'

'' 'Overcast' 'Hot' 'High' 'Weak' 'Yes'

'' 'Rain' 'Mild' 'High' 'Weak' 'Yes'

'' 'Rain' 'Cool' 'Normal' 'Weak' 'Yes'

'' 'Rain' 'Cool' 'Normal' 'Strong' 'No'

'' 'Overcast' 'Cool' 'Normal' 'Strong' 'Yes'

'' 'Sunny' 'Mild' 'High' 'Weak' 'No'

'' 'Sunny' 'Cool' 'Normal' 'Weak' 'Yes'

'' 'Rain' 'Mild' 'Normal' 'Weak' 'Yes'

'' 'Sunny' 'Mild' 'Normal' 'Strong' 'Yes'

'' 'Overcast' 'Mild' 'High' 'Strong' 'Yes'

'' 'Overcast' 'Hot' 'Normal' 'Weak' 'Yes'

'' 'Rain' 'Mild' 'High' 'Strong' 'No' },

attributes=txt(1,2:end);

activeAttributes = ones(1,length(attributes)-1);

data = txt(2:end,2:end);

%% Õë¶ÔÃ¿ÁÐÊý¾Ý½øÐÐ×ª»»

[rows,cols] = size(data);

matrix = zeros(rows,cols);

for j=1:cols

matrix(:,j) = cellfun(@trans2onezero,data(:,j));

end

end

function flag = trans2onezero(data)

if strcmp(data,'Sunny') ||strcmp(data,'Hot')||strcmp(data,'High')||strcmp(data,'Weak')...

||strcmp(data,'No')

flag =1;

return ;

elseif strcmp(data,'Overcast') ||strcmp(data,'Mild')||strcmp(data,'Normal')||strcmp(data,'Strong')...

||strcmp(data,'Yes')

flag =2;

return ;

end

flag =3;

end

3. function ID3333

[matrix,attributes\_label,attributes] = id3\_preprocess();

decisionTree(matrix,attributes\_label);

end