决策树

**概念原理**

决策树是一种非参数的监督学习方法，它主要用于分类和回归。决策树的目的是构造一种模型，使之能够从样本数据的特征属性中，通过学习简单的决策规则——IF THEN规则，从而预测目标变量的值。

决策树是一种用于对实例进行分类的树形结构。决策树由节点（node）和有向边（directed edge）组成。节点的类型有两种：内部节点和叶子节点。其中，内部节点表示一个特征或属性的测试条件（用于分开具有不同特性的记录），叶子节点表示一个分类。

**决策树学习步骤**：1 特征选择 2 决策树的生成 3 决策树的修剪

**那么如何进行特征选择**：

由于特征选择的方法不同，衍生出了三种决策树算法：ID3、C4.5、CART

例子代码：

#include <iostream>

#include <fstream>

#include <math.h>

#include <string>

using namespace std;

#define ROW 14

#define COL 5

#define log2 0.69314718055

typedef struct TNode

{

char data[15];

char weight[15];

TNode \* firstchild,\*nextsibling;

}\*tree;

typedef struct LNode

{

char OutLook[15];

char Temperature[15];

char Humidity[15];

char Wind[15];

char PlayTennis[5];

LNode \*next;

}\*link;

typedef struct AttrNode

{

char attributes[15];//属性

int attr\_Num;//属性的个数

AttrNode \*next;

}\*Attributes;

char \* Examples[ROW][COL] = {//"OverCast","Cool","High","Strong","No",

// "Rain","Hot","Normal","Strong","Yes",

"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","Normal","Strong","Yes",

"OverCast","Hot","Normal","Weak","Yes",

"Rain","Mild","High","Strong","No"

};

char \* Attributes\_kind[4] = {"OutLook","Temperature","Humidity","Wind"};

int Attr\_kind[4] = {3,3,2,2};

char \* OutLook\_kind[3] = {"Sunny","OverCast","Rain"};

char \* Temperature\_kind[3] = {"Hot","Mild","Cool"};

char \* Humidity\_kind[2] = {"High","Normal"};

char \* Wind\_kind[2] = {"Weak","Strong"};

/\*int i\_Exampple[14][5] = {0,0,0,0,1,

0,0,0,1,1,

1,0,0,1,0,

2,1,0,0,0,

2,2,1,0,0,

2,2,1,1,1,

1,2,1,1,0,

0,1,0,0,1,

0,2,1,0,0,

2,1,1,0,0,

0,1,1,1,0,

1,1,1,1,0,

1,1,1,0,0,

2,1,0,0,1

};\*/

void treelists(tree T);

void InitAttr(Attributes &attr\_link,char \* Attributes\_kind[],int Attr\_kind[]);

void InitLink(link &L,char \* Examples[][COL]);

void ID3(tree &T,link L,link Target\_Attr,Attributes attr);

void PN\_Num(link L,int &positve,int &negative);

double Gain(int positive,int negative,char \* atrribute,link L,Attributes attr\_L);

void main()

{

link LL,p;

Attributes attr\_L,q;

tree T;

T = new TNode;

T->firstchild = T->nextsibling = NULL;

strcpy(T->weight,"");

strcpy(T->data,"");

attr\_L = new AttrNode;

attr\_L->next = NULL;

LL = new LNode;

LL->next = NULL;

//成功建立两个链表

InitLink(LL,Examples);

InitAttr(attr\_L,Attributes\_kind,Attr\_kind);

ID3(T,LL,NULL,attr\_L);

cout<<"决策树以广义表形式输出如下："<<endl;

treelists(T);//以广义表的形式输出树

// cout<<Gain(9,5,"OutLook",LL,attr\_L)<<endl;

cout<<endl;

}

//以广义表的形式输出树

void treelists(tree T)

{

tree p;

if(!T)

return;

cout<<"{"<<T->weight<<"}";

cout<<T->data;

p = T->firstchild;

if (p)

{

cout<<"(";

while (p)

{

treelists(p);

p = p->nextsibling;

if (p)cout<<',';

}

cout<<")";

}

}

void InitAttr(Attributes &attr\_link,char \* Attributes\_kind[],int Attr\_kind[])

{

Attributes p;

for (int i =0;i < 4;i++)

{

p = new AttrNode;

p->next = NULL;

strcpy(p->attributes,Attributes\_kind[i]);

p->attr\_Num = Attr\_kind[i];

p->next = attr\_link->next;

attr\_link->next = p;

}

}

void InitLink(link &LL,char \* Examples[][COL])

{

link p;

for (int i = 0;i < ROW;i++)

{

p = new LNode;

p->next = NULL;

strcpy(p->OutLook,Examples[i][0]);

strcpy(p->Temperature,Examples[i][1]);

strcpy(p->Humidity,Examples[i][2]);

strcpy(p->Wind,Examples[i][3]);

strcpy(p->PlayTennis,Examples[i][4]);

p->next = LL->next;

LL->next = p;

}

}

void PN\_Num(link L,int &positve,int &negative)

{

positve = 0;

negative = 0;

link p;

p = L->next;

while (p)

{

if (strcmp(p->PlayTennis,"No") == 0)

negative++;

else if(strcmp(p->PlayTennis,"Yes") == 0)

positve++;

p = p->next;

}

}

//计算信息增益

//link L: 样本集合S

//attr\_L：属性集合

double Gain(int positive,int negative,char \* atrribute,link L,Attributes attr\_L)

{

int atrr\_kinds;//每个属性中的值的个数

Attributes p = attr\_L->next;

link q = L->next;

int attr\_th = 0;//第几个属性

while (p)

{

if (strcmp(p->attributes,atrribute) == 0)

{

atrr\_kinds = p->attr\_Num;

break;

}

p = p->next;

attr\_th++;

}

double entropy,gain=0;

double p1 = 1.0\*positive/(positive + negative);

double p2 = 1.0\*negative/(positive + negative);

entropy = -p1\*log(p1)/log2 - p2\*log(p2)/log2;//集合熵

gain = entropy;

//获取每个属性值在训练样本中出现的个数

//获取每个属性值所对应的正例和反例的个数

//声明一个3\*atrr\_kinds的数组

int \*\* kinds= new int \* [3];

for (int j =0;j < 3;j++)

{

kinds[j] = new int[atrr\_kinds];//保存每个属性值在训练样本中出现的个数

}

//初始化

for (j = 0;j< 3;j++)

{

for (int i =0;i < atrr\_kinds;i++)

{

kinds[j][i] = 0;

}

}

while (q)

{

if (strcmp("OutLook",atrribute) == 0)

{

for (int i = 0;i < atrr\_kinds;i++)

{

if(strcmp(q->OutLook,OutLook\_kind[i]) == 0)

{

kinds[0][i]++;

if(strcmp(q->PlayTennis,"Yes") == 0)

kinds[1][i]++;

else

kinds[2][i]++;

}

}

}

else if (strcmp("Temperature",atrribute) == 0)

{

for (int i = 0;i < atrr\_kinds;i++)

{

if(strcmp(q->Temperature,Temperature\_kind[i]) == 0)

{

kinds[0][i]++;

if(strcmp(q->PlayTennis,"Yes") == 0)

kinds[1][i]++;

else

kinds[2][i]++;

}

}

}

else if (strcmp("Humidity",atrribute) == 0)

{

for (int i = 0;i < atrr\_kinds;i++)

{

if(strcmp(q->Humidity,Humidity\_kind[i]) == 0)

{

kinds[0][i]++;

if(strcmp(q->PlayTennis,"Yes") == 0)

kinds[1][i]++;//

else

kinds[2][i]++;

}

}

}

else if (strcmp("Wind",atrribute) == 0)

{

for (int i = 0;i < atrr\_kinds;i++)

{

if(strcmp(q->Wind,Wind\_kind[i]) == 0)

{

kinds[0][i]++;

if(strcmp(q->PlayTennis,"Yes") == 0)

kinds[1][i]++;

else

kinds[2][i]++;

}

}

}

q = q->next;

}

//计算信息增益

double \* gain\_kind = new double[atrr\_kinds];

int positive\_kind = 0,negative\_kind = 0;

for (j = 0;j < atrr\_kinds;j++)

{

if (kinds[0][j] != 0 && kinds[1][j] != 0 && kinds[2][j] != 0)

{

p1 = 1.0\*kinds[1][j]/kinds[0][j];

p2 = 1.0\*kinds[2][j]/kinds[0][j];

gain\_kind[j] = -p1\*log(p1)/log2-p2\*log(p2)/log2;

gain = gain - (1.0\*kinds[0][j]/(positive + negative))\*gain\_kind[j];

}

else

gain\_kind[j] = 0;

}

return gain;

}

//在ID3算法中的训练样本子集合与属性子集合的链表需要进行清空

void FreeLink(link &Link)

{

link p,q;

p = Link->next;

Link->next = NULL;

while (p)

{

q = p;

p = p->next;

free(q);

}

}

void ID3(tree &T,link L,link Target\_Attr,Attributes attr)

{

Attributes p,max,attr\_child,p1;

link q,link\_child,q1;

tree r,tree\_p;

int positive =0,negative =0;

PN\_Num(L,positive,negative);

//初始化两个子集合

attr\_child = new AttrNode;

attr\_child->next = NULL;

link\_child = new LNode;

link\_child->next = NULL;

if (positive == 0)//全是反例

{

strcpy(T->data,"No");

return;

}

else if( negative == 0)//全是正例

{

strcpy(T->data,"Yes");

return;

}

p = attr->next; //属性链表

double gain,g = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 建立属性子集合与训练样本子集合有两个方案：

一：在原来链表的基础上进行删除；

二：另外申请空间进行存储子集合；

采用第二种方法虽然浪费了空间，但也省了很多事情，避免了变量之间的应用混乱

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(p)

{

while (p)

{

gain = Gain(positive,negative,p->attributes,L,attr);

cout<<p->attributes<<" "<<gain<<endl;

if(gain > g)

{

g = gain;

max = p;//寻找信息增益最大的属性

}

p = p->next;

}

strcpy(T->data,max->attributes);//增加决策树的节点

cout<<"信息增益最大的属性：max->attributes = "<<max->attributes<<endl<<endl;

//下面开始建立决策树

//创建属性子集合

p = attr->next;

while (p)

{

if (strcmp(p->attributes,max->attributes) != 0)

{

p1 = new AttrNode;

strcpy(p1->attributes,p->attributes);

p1->attr\_Num = p->attr\_Num;

p1->next = NULL;

p1->next = attr\_child->next;

attr\_child->next = p1;

}

p = p->next;

}

//需要区分出是哪一种属性

//建立每一层的第一个节点

if (strcmp("OutLook",max->attributes) == 0)

{

r = new TNode;

r->firstchild = r->nextsibling = NULL;

strcpy(r->weight,OutLook\_kind[0]);

T->firstchild = r;

//获取与属性值相关的训练样例Example(vi),建立一个新的训练样本链表link\_child

q = L->next;

while (q)

{

if (strcmp(q->OutLook,OutLook\_kind[0]) == 0)

{

q1 = new LNode;

strcpy(q1->OutLook,q->OutLook);

strcpy(q1->Humidity,q->Humidity);

strcpy(q1->Temperature,q->Temperature);

strcpy(q1->Wind,q->Wind);

strcpy(q1->PlayTennis,q->PlayTennis);

q1->next = NULL;

q1->next = link\_child->next;

link\_child->next = q1;

}

q = q->next;

}

}

else if (strcmp("Temperature",max->attributes) == 0)

{

r = new TNode;

r->firstchild = r->nextsibling = NULL;

strcpy(r->weight,Temperature\_kind[0]);

T->firstchild = r;

//获取与属性值相关的训练样例Example(vi),建立一个新的训练样本链表link\_child

q = L->next;

while (q)

{

if (strcmp(q->Temperature,Temperature\_kind[0]) == 0)

{

q1 = new LNode;

strcpy(q1->OutLook,q->OutLook);

strcpy(q1->Humidity,q->Humidity);

strcpy(q1->Temperature,q->Temperature);

strcpy(q1->Wind,q->Wind);

strcpy(q1->PlayTennis,q->PlayTennis);

q1->next = NULL;

q1->next = link\_child->next;

link\_child->next = q1;

}

q = q->next;

}

}

else if (strcmp("Humidity",max->attributes) == 0)

{

r = new TNode;

r->firstchild = r->nextsibling = NULL;

strcpy(r->weight,Humidity\_kind[0]);

T->firstchild = r;

//获取与属性值相关的训练样例Example(vi),建立一个新的训练样本链表link\_child

q = L->next;

while (q)

{

if (strcmp(q->Humidity,Humidity\_kind[0]) == 0)

{

q1 = new LNode;

strcpy(q1->OutLook,q->OutLook);

strcpy(q1->Humidity,q->Humidity);

strcpy(q1->Temperature,q->Temperature);

strcpy(q1->Wind,q->Wind);

strcpy(q1->PlayTennis,q->PlayTennis);

q1->next = NULL;

q1->next = link\_child->next;

link\_child->next = q1;

}

q = q->next;

}

}

else if (strcmp("Wind",max->attributes) == 0)

{

r = new TNode;

r->firstchild = r->nextsibling = NULL;

strcpy(r->weight,Wind\_kind[0]);

T->firstchild = r;

//获取与属性值相关的训练样例Example(vi),建立一个新的训练样本链表link\_child

q = L->next;

while (q)

{

if (strcmp(q->Wind,Wind\_kind[0]) == 0)

{

q1 = new LNode;

strcpy(q1->OutLook,q->OutLook);

strcpy(q1->Humidity,q->Humidity);

strcpy(q1->Temperature,q->Temperature);

strcpy(q1->Wind,q->Wind);

strcpy(q1->PlayTennis,q->PlayTennis);

q1->next = NULL;

q1->next = link\_child->next;

link\_child->next = q1;

}

q = q->next;

}

}

int p = 0,n = 0;

PN\_Num(link\_child,p,n);

if (p != 0 && n != 0)

{

ID3(T->firstchild,link\_child,Target\_Attr,attr\_child);

FreeLink(link\_child);

}

else if(p == 0)

{

strcpy(T->firstchild->data,"No");

FreeLink(link\_child);

// strcpy(T->firstchild->data,q1->PlayTennis);//----此处应该需要修改----:)

}

else if(n == 0)

{

strcpy(T->firstchild->data,"Yes");

FreeLink(link\_child);

}

//建立每一层上的其他节点

tree\_p = T->firstchild;

for (int i = 1;i < max->attr\_Num;i++)

{

//需要区分出是哪一种属性

if (strcmp("OutLook",max->attributes) == 0)

{

r = new TNode;

r->firstchild = r->nextsibling = NULL;

strcpy(r->weight,OutLook\_kind[i]);

tree\_p->nextsibling = r;

//获取与属性值相关的训练样例Example(vi),建立一个新的训练样本链表link\_child

q = L->next;

while (q)

{

if (strcmp(q->OutLook,OutLook\_kind[i]) == 0)

{

q1 = new LNode;

strcpy(q1->OutLook,q->OutLook);

strcpy(q1->Humidity,q->Humidity);

strcpy(q1->Temperature,q->Temperature);

strcpy(q1->Wind,q->Wind);

strcpy(q1->PlayTennis,q->PlayTennis);

q1->next = NULL;

q1->next = link\_child->next;

link\_child->next = q1;

}

q = q->next;

}

}

else if (strcmp("Temperature",max->attributes) == 0)

{

r = new TNode;

r->firstchild = r->nextsibling = NULL;

strcpy(r->weight,Temperature\_kind[i]);

tree\_p->nextsibling = r;

//获取与属性值相关的训练样例Example(vi),建立一个新的训练样本链表link\_child

q = L->next;

while (q)

{

if (strcmp(q->Temperature,Temperature\_kind[i]) == 0)

{

q1 = new LNode;

strcpy(q1->OutLook,q->OutLook);

strcpy(q1->Humidity,q->Humidity);

strcpy(q1->Temperature,q->Temperature);

strcpy(q1->Wind,q->Wind);

strcpy(q1->PlayTennis,q->PlayTennis);

q1->next = NULL;

q1->next = link\_child->next;

link\_child->next = q1;

}

q = q->next;

}

}

else if (strcmp("Humidity",max->attributes) == 0)

{

r = new TNode;

r->firstchild = r->nextsibling = NULL;

strcpy(r->weight,Humidity\_kind[i]);

tree\_p->nextsibling = r;

//获取与属性值相关的训练样例Example(vi),建立一个新的训练样本链表link\_child

q = L->next;

while (q)

{

if (strcmp(q->Humidity,Humidity\_kind[i]) == 0)

{

q1 = new LNode;

strcpy(q1->OutLook,q->OutLook);

strcpy(q1->Humidity,q->Humidity);

strcpy(q1->Temperature,q->Temperature);

strcpy(q1->Wind,q->Wind);

strcpy(q1->PlayTennis,q->PlayTennis);

q1->next = NULL;

q1->next = link\_child->next;

link\_child->next = q1;

}

q = q->next;

}

}

else if (strcmp("Wind",max->attributes) == 0)

{

r = new TNode;

r->firstchild = r->nextsibling = NULL;

strcpy(r->weight,Wind\_kind[i]);

tree\_p->nextsibling = r;

//获取与属性值相关的训练样例Example(vi),建立一个新的训练样本链表link\_child

q = L->next;

while (q)

{

if (strcmp(q->Wind,Wind\_kind[i]) == 0)

{

q1 = new LNode;

strcpy(q1->OutLook,q->OutLook);

strcpy(q1->Humidity,q->Humidity);

strcpy(q1->Temperature,q->Temperature);

strcpy(q1->Wind,q->Wind);

strcpy(q1->PlayTennis,q->PlayTennis);

q1->next = NULL;

q1->next = link\_child->next;

link\_child->next = q1;

}

q = q->next;

}

}

int p = 0,n = 0;

PN\_Num(link\_child,p,n);

if (p != 0 && n != 0)

{

ID3(tree\_p->nextsibling,link\_child,Target\_Attr,attr\_child);

FreeLink(link\_child);

}

else if(p == 0)

{

strcpy(tree\_p->nextsibling->data,"No");

FreeLink(link\_child);

}

else if(n == 0)

{

strcpy(tree\_p->nextsibling->data,"Yes");

FreeLink(link\_child);

}

tree\_p = tree\_p->nextsibling;//建立所有的孩子结点

}//建立决策树结束

}

else

{

q = L->next;

strcpy(T->data,q->PlayTennis);

return;//这个地方要赋以训练样本Example中最普遍的Target\_attributes的值

}

}

