文章编号:1001-9081(2001)08-0160-03

# 一种机器学习系统的设计与实现

何友鸣,方辉云
(中南财经政法大学 信息系,湖北 武汉 430064)

**摘　要**:在试图完成一种分类专家系统的实践中,采用系统与有专门知识用户之间的交互作用,完成了系统获取知识的设计,但还不能说很完善。

**关键词**:专家系统;机器学习;机器学习系统
**中图分类号**: TP391.6　　**文献标识码**:A

## 1　引言

我们知道,专家系统必需有学习功能,即所谓机器学习。这种功能要设计机器学习系统来实现。很明显,机器学习系统要具备两个功能—获取知识和更新知识。我们采用基于告知机器学习(learning by being told)的方法,即用反馈的手段使系统与具有专门知识的用户交互作用,达到为分类专家系统获取知识的目的。所获取的知识丰富系统的知识库,推理机再利用这些知识回答用户提出的、关于某个领域的分类问题。

目前,知识获取的主要方法有:通过告知学习;通过范例的归纳学习和通过观察和发现学习。我们选用告知学习方法来设计学习系统,是因为该方法比较适合于建立中小型专家系统,其实现也比较容易。

告知学习理论主要是三条准则:

如果告知的知识能从现有知识库推导出,则该知识不能加入知识库;

若告知的知识,与知识库的知识不一致,则拒绝接收;

如果告知的知识不能从现有知识中导出,又与知识库知识具有一致性,则以新知识接收。

我们所设计的学习系统,就是基于以上三准则,通过搜索知识库,反馈搜索信息,并与具有专门知识的用户会话,实现了知识的获取。

所谓"反馈"过程,是指由具有专门知识的用户(专家)输入某领域的事物或属性,学习系统以此搜索知识库,反馈搜索到的信息给用户,由用户根据经验判断搜索到的结果是否正确,学习系统再根据用户的回答,增加或减去用户输入的事物或属性。搜索知识库的过程,看作是系统猜测所学知识正确性的过程,反馈过程则看作是对所学知识的修正和调整过程。

具体实现中,系统猜测通过记分高低来选择,总是选择最高得分的事物来完成知识的增加即获取。

更新方面,先约定学习系统只是更新那些已经学到(系统猜测)的、但现在认为是错误(用户指出)的知识。

本文主要介绍知识获取的实现,以下是C++语言的描述。

## 2　C++语言描述

C++语言所使用的几个基本数据结构为:

1) 实现过程中运用了两个库:知识库 abase.dat 和共性库 ahare.dat。

2) 知识库的每一项都由一个结点和它的一系列属性所组成。构造包含结点的知识库组件的一个较容易的方法是,使用一个具有如下结构的链表:

```
struct NODE
{
  int score;
  char name[40]
  struct ATTRIBUTE * alist;
  struct NODE * NEXT;
}
att1;
```

结构中,项 int score 并不是所要学的内容,而是为了反馈学习所设置的一个变量。

3) 由于并不知属性的确切数目,因而属性表是一种具有如下结构的单关联表:

```
struct ATTRIBUTE
{
  char attrib [80]
  struct ATTRIBUTE * NEXT;
}
att1;
```

4) 共性库 ahare.dat 的每一项是由一个共有属性和具有该属性的一系列结点组成。其构造方法形同知识库,具体为:

· 属性结点:

```
struct attrib
{
  char attribute [80]
  struct node * alist;
  struct attrib * anext;
}
att2;
```

· 由结点所组成的单关联表:

```
struct note
{
  char notename[40]
  struct node * next;
}
nod2;
```

作者简介:何友鸣,男,副教授,主要研究方向:人工智能、信息管理;　方辉云,女,副教授,主要研究方向:高级语言、数据库.

## 2 学习系统的算法

1) 输入所要识别或学习的对象个数 n;

2) 输入这 n 个结点的名称(事物名称)。若知识库中有相同的结点，则将其连同属性取出来，建立由这 n 个结点组成的链表(简称结点链表);

3) 识别每条属性属于哪个对象:

a) 输入一组属性，并建立由这一组属性组成的链(简称属性链表);

b) 把属性链表同结点链表的的属性进行比较。若有 k 项属性相同，则该对象得分加 k，无相同属性得分也不变;

c) 凭得分对结点链表中各个结点降序排列;

d) 遍加结点链表。因为某结点得分越高，则该结点的可能性越大。

e) 假设现在访问结点 P:

机器: 是结点 P 吗?

人: 是。将该组属性加入 P 中; 不是。继续往下询问，直至遍历完所有结点;

4) 重复动作 3)，直至输入完所有属性。

## 3 程序组成与说明

### 3.1 结点的输入

首先，用户要告诉机器，你现在想学习什么，即通过人机对话,输入想学的结点名; 然后机器才能告诉你怎样来识别这些结点。输入结点后，将其并入结点链表。要注意，在知识库中有些结点本来就存在，若要学习它，可将该结点连同属性取出并入链表。这项工作用如下输入程序来实现。

```
enter()                                          / * 输入结点名 * /
{
  int i,j,rr;
  FILE * fp1;
  struct NODE * object, * ptr;
  i = 0;
  fp1 = fopen ( "abase.daf","rw");
  object = (struct NODE * )malloc (sizeof (nodl));
  printf ( "please enter the node name.");
  while (i < n)
  {
    printf (":")
    gets (string);
    j = 0;
    while (! fseek (fp1,j,SEEK - SET));
    {
      rr = fread (ptr,1000,1,fp1);
      if (rr < > 1) exit(1);
      if (strcmp (prt - > name,string) = = 0;
      {
        object = ptr;
        object - > score = 0;
        object - > NEXT = head;
        object = (struct NODE * ) malloc (sizeof (nodl));
        break;
      }
      j ++;
    }
    if (fp1 = = NULL);
    {
      object - > name = string;
      object - > = score = 0;
      object - > NEXT = head;
      head = object;
      object = (struct NODE * ) malloc (sizeof (nodl));
    }
    i ++;
  }
}
```

该程序中，head 是结点链表的头。

### 3.2 属性的输入

由于现在是机器学习，我们并不知所输入的属性属于哪个结点。所以，只是根据已知的特征对其进行猜测。变量 score 是为后面排序服务的。这里,每输入一条属性，先访问链表，若结点有此属性，则加 1 分，否则加 0; 其次，查看共性表中是否有此属性。若有，将其取出，因为共性表说明了哪些结点共有哪一项属性，这在机器学习中并不用到，仅是一个准备动作而已。实现过程如下:

```
enter2()                                         / * 输入属性 * /
{
  struct NODE * objp;
  struct ATTRIBUTE * pt1, * pt2;
  FILE * FP2;
  void ptr;
  struct attribb * pr1;
  int k,rr;
  printf ("please enter the attribute.");
  objp = head;
  pt1 = (struct ATTRIBUTE * ) malloc (sizeof (att1));
  while (1);
  {
    printf (":");
    gets (pt1 - > attrib);
    if (! lpt1 - > attrib [0]) break;
    pthead = pt1;
    pt1 = (struct ATTIBUTE * ) malloc (sizeof (att1));
    while (objp - > name [0];
    {
      pt2 = objp - > alist;
      while (pt2 - > attrib [0];
      {
        if (strcmp (pt2 - > attrib,pt1 - > attrjb) = = 0);
        {
          objp - > acore ++;
          treak;
        }
        pt2 = pt2 - > ANEXT;
      }
      objp = objp - > NEXT;
    }
    fp2 = fopen ("share.daf","fw");
    pr11 = (struct attribb * ) malloc (sizeof (att2));
    k = 0;
```

```
    while (! fseek (fp2,k,SEEK - SRT));
    {
      prr = fread (ptr,50,1,fp2);
      if (rr < > 1) exit (1);
      if (strcmp (ptr2 - > sttribute,pt1 - > attrib) = = 0)
      {
        pr1 = pt2;
        pr1 - > anext = prhead;
        prhead = pr1;
        pr1 = (struct attribb * ) malloc (sizeof (att2));
        break;
      }
      k ++;
    }
  }
}
```

### 3.3 排序

对链表排序比较麻烦。前面的 score 正是为排序所设置的。它的大小正是猜测结点的程度。

```
stor (struct NODE * P)                    / * 对结点进行排序 * /
{
  struct NODE * pp = p, * pointer1 = p, * pointer2, * maxp, *
  temp;
  maxp = (struct NODE * )malloc (aizeof (nod1));
  while (pointer1 - > name [0];
  {
    maxp - > score = pp - > score;
    maxp - > name = pp - > name;
    maxp - > alist = pp - > alist;
    while (pp - > mane [0];
    {
      maxp = NULL;
      maxp - > score = pp - > score;
      maxp - > name = pp - > name;
      maxp - > alist = pp - > alist;
      pointer2 = pp;
    }
  }
  if (maxp - > score > pointer1 - > score);
  {
    temp - > score = pointer1 - > score;
    temp - > mane = pointer - > name;
    temp - > alist = pointer - > alist;
    pointer1 - > score = maxp - > score;
    pointer1 - > name = maxp - > name;
    pointer1 - > alist = maxp - > alist;
    pointer2 - > score = maxp - > score;
    pointer2 - > name = maxp - > name;
    pointer2 - > alist = maxp - > alist; free (maxp);
    free (temp);
  }
}
```

sort()中，maxp 暂时存放最大结点，temp 只是用来交换。指针 pp 用来遍历链表，而指针 pointer1 和 pointer2 则分别指出排序已排到了哪个结点和目前的大结点信息。

系统中的主程序用来控制调用各子程序、控制反馈和与用户对话的过程，限于篇幅，这些内容从略。

要提及一下的是，这种学习系统只适合于中小型知识库的专家系统。因为大型专家系统的搜索时间长，效率也很低。若要对大型知识库进行操作，则需要对算法进行修改。

本文着重讲知识的增加，未涉及知识的更新，因为知识的更新对知识的专业性要求较高，其实现方法还有待探讨。

**参考文献**


[1] 徐新华. Database 和 Midas 编程技术[M]. 北京:人民邮电出版社, 1998.

[2] 何友鸣, 等. 机器人学与计算机经济信息管理[A]. 中国人工智能学会:中国 2000 年机器人学大会论文专辑[C]. 中南工业大学学报(自然科学版), 2000,31.

[3] 史美林, 等. WFWS:工作流程管理系统[J]. 计算机学报, 1999, 22(3).


(上接第 159 页)

2) 循环: $\dfrac{\vdash [\rho \wedge c]\ S\ [\rho]}{\vdash [\rho]\ \text{while c do S od}\ [\rho \wedge \neg c]}$ 下面用公理学方法来证明程序 p'的正确性。

```
{x2 ≥ 0}                                    - - - - 前置断言
x4 : = x2; x3 : = 0;
{x2 ≥ 0 ∧ x4 ≥ 0 ∧ x3 = 0}
{ρ: x3 + x1 * x4 = x1 * x2}                 - - - - 循环不变式
while x4 ≠ 0 do
begin {x4 ≠ 0 ∧ x3 + x1 * x4 = x1 * x2}
x3 : = x1 + x3; x4 : = x4 - 1;
{ρ: x3 + x1 * x4 = x1 * x2}                 - - - - 循环不变式
end;
{x4 = 0 ∧ x3 = x1 * x2}                     - - - - 结果断言
```

现对这两种证明方法加以比较:公理学方法的关键是找出循环不变式 ρ,前述方法的关键是构造归纳谓词 A,此例中两者具有相同的形式,且都有在循环语句执行前后保持不变的性质。因此找出 ρ 和 A 是证明循环程序的重要步骤,且在前述方法中,适当引入外加的谓词可以大大减少证明的工作量。

## 6 结束语

本文只是用一个简单的模型介绍了这种证明方法,可以为更复杂的程序语言定义出该方法的扩充,使之适用于过程和函数、递归过程以及复合结构等构造。

**参考文献**


[1] Manna Z. The Correctness of Programs[J]. ComPut. Syst. Sci., 1969.

[2] Hoare CAR. Proof of Correctness of Data Representation[M]. Acta Informatica, 1972.

[3] 胡潇, 王凤林, 等. 一种静态的协商算法[J]. 计算机学报, 1996, (6).

[4] 石纯一. 基于解释的机器学习方法[M]. 北京:清华大学出版社, 1997.

[5] 陈火旺, 罗朝晖, 马庆鸣. 程序设计方法学基础[M]. 长沙:湖南科学技术出版社, 1993.