Computer Science and Application 计算机科学与应用, 2016, 6(10), 617-628
Published Online October 2016 in Hans. http://www.hanspub.org/journal/csa
http://dx.doi.org/10.12677/csa.2016.610076

Hans汉斯

Minimal Decision Tree Generation for Multi-Label Decision Tables

Ying Qiao, Meiling Xu, Farong Zhong, Jing Zeng, Yuchang Mo

Zhejiang Normal University, Jinhua Zhejiang
Email: 1823652081@qq.com

Received: Oct. 5th, 2016; accepted: Oct. 23rd, 2016; published: Oct. 28th, 2016

Abstract

Decision tree is a widely used classification in data mining. It can discover the essential knowledge from the common decision tables (each row has a decision). However, it is difficult to do data mining from the multi-label decision tables (each row has a set of decisions). In a multi-label decision tables, each row contains several decisions, and several decision attributes are represented using a set. By testing the existing heuristic algorithms, such as greedy algorithms, their performance is not stable, *i.e.*, the size of the decision tree might become very large. In this paper, we propose a dynamic programming algorithm to minimize the size of the decision trees for a multi-label decision table. In our algorithm, the multi-label decision table is divided into several subtables, and the decision tree is constructed by using all subtables of the multi-label decision table, then useful information can be discovered from the multi-label decision tables.

Keywords

Multi-Label Decision Tables, Decision Trees, Dynamic Programming Algorithm

多值决策表的最小决策树生成

乔　莹，许美玲，钟发荣，曾　静，莫毓昌

浙江师范大学，浙江　金华
Email: 1823652081@qq.com

收稿日期：2016年10月5日；录用日期：2016年10月23日；发布日期：2016年10月28日

文章引用: 乔莹, 许美玲, 钟发荣, 曾静, 莫毓昌. 多值决策表的最小决策树生成[J]. 计算机科学与应用, 2016, 6(10): 617-628. http://dx.doi.org/10.12677/csa.2016.610076

摘　要

决策树技术在数据挖掘的分类领域应用极其广泛，可以从普通决策表(每行记录包含一个决策值)中挖掘有价值的信息，但是要从多值决策表(每行记录包含多个决策值)中挖掘潜在的信息则比较困难。多值决策表中每行记录包含多个决策值，多个决策属性用一个集合表示。针对已有的启发式算法，如贪心算法，由于性能不稳定的特点，该算法获得的决策树规模变化较大，本文基于动态规划的思想，提出了使决策树规模最小化的算法。该算法将多值决策表分解为多个子表，通过多值决策表的子表进行构造最小决策树，进而对多值决策表进行数据挖掘。

关键词

多值决策表，决策树，动态规划算法

1. 引言

多值决策表每行记录的多个决策被标记为一个决策集，在现实生活中这样的表很常见，因为没有足够多的属性值去标记单独的行，因此就有条件属性值相同而决策值不同的实体。目前多值决策表已经得到了人们的关注，例如图像的语义标注问题[1]，音乐情感分类[2]，基因组的功能分类[3]和文本分类[4]等。此外，这类数据集在优化问题中是很常见的，如在旅行商中找出 Hamiltonian 回路的最小长度问题，在邮局中找出最近的邮局问题。这种情况下，我们通常在输出的多个解中选择最优解[5]。

在已有研究中，多值数据的决策树和其他分类通常认为是预测多值分类问题[6] [7] [8] [9]。文献[10]研究了使用基于边界子表的多值决策表构造决策树的贪心算法。除此之外，文献[11]研究了在最常用决策情况下的贪心算法和广义的决策方法，但所研究的多值决策表局限于单值决策表，即决策集中只有一个元素。与多值数据有关的问题常常被认为是分类学习问题：多标签学习[12]、多实例学习[9]等，也有一些部分被标记的半监督学习[13]。此外，如在局部学习[14]、模糊学习[15]、多标签学习[16]中认为只有一个决策值是正确的。这些文献只是关注了分类的结果，而不是数据模型的优化。但我们需要解决的是多值决策问题。

本文研究用决策树对多值决策表进行信息挖掘，考虑信息表达和模型优化问题，目标是利用动态规划算法使得决策树的规模达到最小。包含五部分内容：第二部分给出了相关的概念，第三部分中提出了构造决策树最小化算法，即动态规划算法，第四部分给出了实例分析并与已有的贪心算法进行比较，第五部分进行全文总结。

2. 概念

多值决策表 T 是由非负整数填充的矩形表表示。这个表的列记为条件属性 f_1，…，f_n，且每个条件属性对应的属性值用非负整数表示。如果属性值为字符串，那么必须将字符串编译为非负整数值。在表中没有重复的行，且每一行的多个决策用一个非空有限自然数集(决策集)表示。我们用 $N(T)$表示表 T 的行数，r_i 表示第 i 行，其中 $i = 1$，…，$N(T)$。如 r_1 表示第一行，r_2 表示第二行等(见表 1)。

如果一个决策属于表 T 每一行记录的决策集，那么我们称它为表 T 的常用决策。如果表 T 没有记录或有一个常用决策，那么称表 T 为退化表。如表 2 中的 T′是一个退化表，常用决策为 1。

从表 T 中删除一些行形成的表称为表 T 的子表。表 T 的子表是由行和列交叉组成的，列代表条件属性，用 f_{i1}，…，f_{im} 表示，对应的条件属性值用 a_1，…，a_m 表示，因此表 T 的子表可用 $T(f_{i1}, a_1)$，…，$(f_{im},$

Table 1. A multi-label decision table T
表 1. 多值决策表 T

T	f_1	f_2	f_3	d
r_1	0	1	0	{1}
r_2	2	1	0	{1,2}
r_3	1	0	2	{1,3}
r_4	0	0	1	{2}

Table 2. A degenerate table T' of the multi-label decision table T
表 2. 多值决策表 T 的退化表 T'

T'	f_1	f_2	f_3	d
r_1	0	1	0	{1}
r_2	2	1	0	{1,2}
r_3	1	0	2	{1,3}

a_m)表示。这样的非空子表(包括表 T)称为 T 的可分离子表。如表 1 的多值决策表 T 的子表 $T(f_1, 0)$由第 1 行和第 4 行组成(见表 3)；类似地，子表 $T(f_1, 0)(f_2, 0)$由第 4 行组成(见表 4)。

用 $E(T)$表示表 T 的每个条件属性值不同的属性集。如表 T 中，$E(T) = \{f_1, f_2, f_3\}$。同理，对于子表 $T(f_1, 0)$有 $E(T(f_1, 0)) = \{f_2, f_3\}$，因为在子表 $T(f_1, 0)$中，条件属性 f_1 的值是恒为 0 的常量。对于 $f_i \in E(T)$，我们用 $E(T, f_i)$表示条件属性 f_i(f_i 列)的一组值。如表 T 和条件属性 f_1，$E(T, f_1) = \{0, 1\}$。

在决策集中属性值出现次数最多且数值最小的决策，称为 T 的最常用决策。如表 T^0 的最常用决策是 1。即使 1 和 2 在决策集中都出现 3 次，但是 1 是最小决策，因此我们选择 1 作为最常用决策。$H(T)$表示表 T 的决策集中包含最常用决策的行数。对于表 T，$H(T) = 3$。

3. 决策树最小化算法

3.1. 决策树

根据表 T 构造决策树，每个叶子节点代表一个决策用一个自然数表示，每个非叶子节点代表属性集合$\{f_1, \ldots, f_n\}$中的一个属性。从每个非叶子节点出发的输出边用不同的非负整数表示，如二值属性的两条边就用 0 和 1 表示。

令 Γ 为根据表 T 构造的决策树，v 为 Γ 的节点。节点 v 和 T 的子表是一一映射，即对于每个节点 v，都有唯一的 T 的子表与之对应。我们定义表 T 的子表 $T(v)$对应于节点 v。如果 v 是 Γ 的根节点，那么 $T(v) = T$，即子表 $T(v)$与 T 是一样的。否则 $T(v)$是表 T 的子表 $T(f_{i1}, \delta_1) \ldots (f_{im}, \delta_m)$，属性 $f_{i1}, \ldots, f_{im}$ 和属性值 $\delta_1, \ldots, \delta_m$ 分别是从根节点到节点 v 整条路径上的节点和边。如果对于 Γ 的任何节点 v 满足以下条件，我们称 Γ 是 T 的决策树：

1) 如果 $T(v)$是退化的，那么 v 被标记为 $T(v)$的常用决策。

2) 如果 $T(v)$是非退化的，那么 v 用 $f_i \in E(T(v))$表示，假设 $E(T(v), f_i) = \{a_1, \ldots, a_k\}$，则来自节点 v 的 k 条输出边为 $a_1, \ldots, a_k$。

假设图 1 给出的是多值决策表的决策树例子，如果节点 v 用属性 f_3 表示，那么对应于节点 v 的子表 $T(v)$记为 $T(f_1, 0)$。类似地，对应于节点 2 的子表为 $T(f_1, 0)(f_3, 1)$。

$N(\Gamma)$表示决策树 Γ 的节点数，$N^t(\Gamma)$和 $N^n(\Gamma)$分别表示决策树 Γ 的叶子节点数和非叶子节点数。

Table 3. A subtable $T(f_1, 0)$ of the multi-label decision table T
表 3. 多值决策表 T 的子表 $T(f_1, 0)$

$T(f_1, 0)$	f_1	f_2	f_3	d
r_1	0	1	0	{1}
r_4	0	0	1	{2}

Table 4. A subtable $T(f_1, 0)(f_2, 0)$ of the multi-label decision table T
表 4. 多值决策表 T 的子表 $T(f_1, 0)(f_2, 0)$

$T(f_1, 0)(f_2, 0)$	f_1	f_2	f_3	d
r_4	0	0	1	{2}

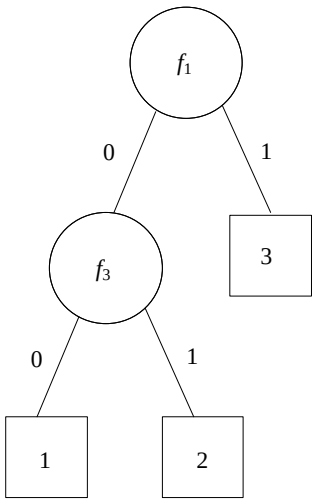

Figure 1. A decision tree for the multi-label decision table
图 1. 多值决策表的决策树

3.2. 动态规划算法

在这一节中，我们给出动态规划算法 A_d 用以构造最小决策树。A_d 算法是针对一个给定的多值决策表构建最小决策树的算法(节点数、非叶子节点数或叶子节点数最少)。这个算法是以动态规划方法为基础[17]，在最坏的情况下该算法的复杂度表现为指数阶。该算法将多值决策表分解成若干个互相关联的子表，将子表在各个阶段按照一定的次序排列，对某个给定的阶段状态先求解子表的问题，然后从子表的解中得到多值决策表的最优解。它的动态性体现在对于重复出现的子问题，第一次遇到对它进行求解，并将解保存起来，以备后续再次使用。

令多值决策表 T，其中用 f_1，…，f_n 表示 n 个条件属性，包含表 T 所有可分离子表的集合记为 $S(T)$。算法 A_d 的第一部分是构造集合 $S(T)$。在构造集合 $S(T)$时，首先将给定的 T 表指定为 $S(T) = \{T\}$，即 $S(T)$ 集中只有一个 T 表，显然 T 表是未处理过的并且 $E(T)$非空，然后以子表 $T(f_i, \delta)$的形式添加到 $S(T)$集合中，并判断集合中的子表是否被处理过，属性值不同的属性集是否为空集。如果是未处理过的，并且属性集非空，那么就将该表的子表添加到 $S(T)$集合中。以此类推，直到 $S(T)$集合中的所有表都被处理过，最后返回 $S(T)$ (见算法 1)。

算法 1 可分离子表 $S(T)$集合的构造

输入：一个多值决策表 T，条件属性为 $f_1, \ldots, f_n$

输出：$S(T)$集

指定 $S(T) = \{T\}$且对未处理过的 T 进行标记；

While(true) **do**

if $S(T)$中没有未处理过的表 **then**

返回 $S(T)$；

else

选择一个 $S(T)$中未处理过的表 T_s

if $E(T_s) = \varnothing$ **then**

标记表 T_s 已被处理

else

以 $T_s(f_i, \delta)$的形式添加到 $S(T)$集合中，$f_i \in E(T_s)$，$\delta \in E(T_s, f_i)$，添加后的 $S(T)$集合中不会有重复的表，标记 T_s 表已被处理，且新的子表 $T_s(f_i, \delta)$未被处理

end if

end if

end while

算法 A_d 的第二部分是构造最小决策树，决策树大小考虑的是树的节点数或者是非叶子节点、叶子节点数。从算法 1 得到的 $S(T)$集合中选择一个未指定为决策树的表，选择的标准是该表要么是退化表要么是它所有的子表是已被指定为决策树的可分离子表。如果是退化表，那么将退化表的常用决策作为叶子节点；否则用决策树表示子表，其中决策树的节点由属性表示，决策树的边由属性值表示。最后在所得到决策树中选出规模最小的(见算法 2)。A_d 算法返回对应于表 T 的最小化决策树。

算法 2 对 $S(T)$的每个表构造最小化决策树

输入：一个多值决策表 T，条件属性 $f_1, \ldots, f_n$ 和集合 $S(T)$

输出：T 的决策树 $A_d(T)$

While(true) **do**

if T 已被指定为完整的决策树 **then**

返回决策树 $A_d(T)$；

else

选择一个 $S(T)$集中仍然未指定为决策树的 T_s 表，这个表要么是退化表要么是 T_s 表的所有已被指定为决策树的可分离子表

if T_s 是退化表 **then**

指定 T_s 表是由一个节点组成的决策树，并且用 T_s 表的共同决策作为叶子节点

else

用决策树 $\Gamma(f_i, \delta)$将子表 $T_s(f_i, \delta)$指定为树的一部分，其中每个 $f_i \in E(T_s)$，$\delta \in E(T_s, f_i)$，决策树 Γ_{fi} 的根节点用 f_i 表示，这里 $f_i \in E(T_s)$，$E(T_s, f_i) = \{\delta_1, \ldots, \delta_r\}$,从这个根节点出发恰好有 r 条边 $d_1, \ldots, d_r$，每条边分别由数字 $\delta_1, \ldots, \delta_r$ 表示。在被指定的 T_s 表中选择出所有决策树规模最小的决策树 Γ_{fi}。

end if

end if

end while

4. 实例分析

4.1. 实例介绍

动态规划算法在实际生活中应用非常广泛，如在学校的教务系统中根据学生的考试成绩，就可以预测到学生的文理科情况。表 5 是某中学教务系统中学生的语文、数学、英语考试成绩，其中 r_i 表示学生。

用 0 和 1 分别表示条件属性值中的“一般”和“好”；用 1、2 和 3 分别表示决策集中的“艺术生”、“文科生”、“理科生”；用 f_i 表示“语文”、“数学”、“英语”科目。这样将表 5 转化成抽象的二值属性的多值决策表(见表 6)。

4.2. 最小决策树构造

本节以多值决策表 T^0 为例(见表 6)，利用第 3 节中提到的动态规划算法构造多值决策表的最小决策树。具体步骤如下：

Step1：假定 $S(T) = \{T^0\}$，且对 T^0 未做过任何的处理，令 $T_s = T^0$ 可写出表 T_s 的子表 $T_s(f_1, 0)$，$T_s(f_1, 1)$，$T_s(f_2, 0)$，$T_s(f_2, 1)$，$T_s(f_3, 0)$，$T_s(f_3, 1)$。见表 7~12。

表 7~12 在 $S(T)$集合中未出现，所以将这 6 个子表添加到 $S(T)$集合中，得到新的 $S(T) = \{T^0, T_s(f_1, 0), T_s(f_1, 1), T_s(f_2, 0), T_s(f_2, 1), T_s(f_3, 0), T_s(f_3, 1)\}$，并对表 T^0 作标记表示已被处理。

Step2：在 $S(T)$集合中选择一个未处理过的子表，令 $T_s = T_s(f_1, 0)$，这时 T_s 的子表为 $T_s(f_1, 0)(f_2, 0)$，$T_s(f_1, 0)(f_2, 1)$，$T_s(f_1, 0)(f_3, 0)$，$T_s(f_1, 0)(f_3, 1)$。列举如下(见表 13~16)。

将这 4 个子表添加到 $S(T)$集合中，得到新的 $S(T) = \{T^0, T_s(f_1, 0), T_s(f_1, 1), T_s(f_2, 0), T_s(f_2, 1), T_s(f_3, 0), T_s(f_3, 1), T_s(f_1, 0)(f_2, 0), T_s(f_1, 0)(f_2, 1), T_s(f_1, 0)(f_3, 0), T_s(f_1, 0)(f_3, 1)\}$，并对表 T^0，$T_s(f_1, 0)$，$T_s(f_1, 0)(f_2, 1)$，$T_s(f_1, 0)(f_3, 0)$作标记表示已被处理。

按照同样的方法，最后得出 $S(T)=\{T^0, T_s(f_1, 0), T_s(f_1, 1), T_s(f_2, 0), T_s(f_2, 1), T_s(f_3, 0), T_s(f_3, 1), T_s(f_1, 0)(f_2, 0), T_s(f_1, 0)(f_2, 1), T_s(f_1, 0)(f_3, 0), T_s(f_1, 0)(f_3, 1), T_s(f_1, 1)(f_2, 0), T_s(f_1, 1)(f_2, 1), T_s(f_2, 0)(f_3, 1), T_s(f_1, 0)(f_2, 0)(f_3, 0), T_s(f_1, 0)(f_2, 0)(f_3, 1)\}$。其中在 $S(T)$集合中的结果是互异的，但是同一个结果可能是多个不同表的子表，如 $T_s(f_1, 1)(f_2, 0)$和 $T_s(f_1, 1)(f_3, 1)$。

Step3：选择 $S(T)$集合中的任意一个表，当选择 T 表时，由于 T 表为非退化表，所以根据算法 2 得出以下结果(图 2)，其中 Γ_{f1}、Γ_{f2}、Γ_{f3} 分别表示 $T_s(f_1, 0)$和 $T_s(f_1, 1)$，$T_s(f_2, 0)$和 $T_s(f_2, 1)$，$T_s(f_3, 0)$和 $T_s(f_3, 1)$所对应的决策树子树。

在以上的决策树中选择规模最小的决策树。由于它们的规模是一样的，所以要对每一个决策树进行拓展。下面以选择表 $T_s(f_1, 0)$为例。

Step4：当选择 T 的子表 $T_s(f_1, 0)$时，得到的结果是 Γ_{f2} 和 Γ_{f3}，但 Γ_{f2} 和 Γ_{f3} 分别表示 $T_s(f_1, 0)(f_2, 0)$和 $T_s(f_1, 0)(f_2, 1)$，$T_s(f_1, 0)(f_3, 0)$和 $T_s(f_1, 0)(f_3, 1)$所对应的决策树子树。

当选择 T 的子表 $T_s(f_1, 1)$时，由于该表为退化表，所以将共同决策 3 作为叶子节点，故结合以上三个表的选取，得到的决策树如图 3 所示。

Step5：当选择 $T_s(f_1, 0)$的子表 $T_s(f_1, 0)(f_2, 0)$时，得到的决策树为 $\Gamma_{f1,f2,f3}$；当选择 $T_s(f_1, 0)$的子表 $T_s(f_1, 0)(f_3, 0)$和 $T_s(f_1, 0)(f_3, 1)$，得到决策树 $\Gamma_{f1,f3}$。由于 $\Gamma_{f1,f3}$ 的规模比 $\Gamma_{f1,f2,f3}$ 小，因此选择决策树 $\Gamma_{f1,f3}$ (图 4)。

利用同样的方法可对决策树 Γ_{f2} 和 Γ_{f3} 进行构造，构造后的结果如图 5 所示。

由于 $\Gamma_{f3,f2,f1}$ 会使决策树的规模变得更大，所以在拓展过程中不考虑它的拓展。由以上的决策树可以看出，$\Gamma_{f2,f1,f3}$，$\Gamma_{f2,f3,f1}$，$\Gamma_{f3,f2,f1}$ 比 $\Gamma_{f1,f3}$ 的规模大，且 $\Gamma_{f1,f3}$ 已经是完整的决策树，所以 $\Gamma_{f1,f3}$ 是多值决策表 T^0 规模最小的决策树。因此，多值决策表(表 5)根据动态规划算法构造出最小决策树(见图 6)。

Table 5. A student's rank table
表 5. 学生成绩等级表

学生	数学	英语	语文	类别
r_1	一般	一般	一般	{艺术生}
r_2	一般	好	好	{艺术生，文科生}
r_3	好	一般	好	{艺术生，理科生}
r_4	好	好	一般	{文科生，理科生}
r_5	一般	一般	好	{文科生}

Table 6. A multi-label decision table T^0
表 6. 多值决策表 T^0

T^0	f_1	f_2	f_3	d
r_1	0	0	0	{1}
r_2	0	1	1	{1,2}
r_3	1	0	1	{1,3}
r_4	1	1	0	{2,3}
r_5	0	0	1	{2}

Table 7. A subtable $T_s(f_1, 0)$ of the multi-label decision table T^0
表 7. 多值决策表 T^0 的子表 $T_s(f_1, 0)$

$T_s(f_1, 0)$	f_1	f_2	f_3	d
r_1	0	0	0	{1}
r_2	0	1	1	{1,2}
r_5	0	0	1	{2}

Table 8. A subtable $T_s(f_1, 1)$ of the multi-label decision table T^0
表 8. 多值决策表 T^0 的子表 $T_s(f_1, 1)$

$T_s(f_1, 1)$	f_1	f_2	f_3	d
r_3	1	0	1	{1,3}
r_4	1	1	0	{2,3}

Table 9. A subtable $T_s(f_2, 0)$ of the multi-label decision table T^0
表 9. 多值决策表 T^0 的子表 $T_s(f_2, 0)$

$T_s(f_2, 0)$	f_1	f_2	f_3	d
r_1	0	0	0	{1}
r_3	1	0	1	{1,3}
r_5	0	0	1	{2}

Table 10. A subtable $T_s(f_2, 1)$ of the multi-label decision table T^0
表 10. 多值决策表 T^0 的子表 $T_s(f_2, 1)$

$T_s(f_2, 1)$	f_1	f_2	f_3	d
r_2	0	1	1	{1,2}
r_4	1	1	0	{2,3}

Table 11. A subtable $T_s(f_3, 0)$ of the multi-label decision table T^0
表 11. 多值决策表 T^0 的子表 $T_s(f_3, 0)$

$T_s(f_3, 0)$	f_1	f_2	f_3	d
r_1	0	0	0	{1}
r_4	1	1	0	{2,3}

Table 12. A subtable $T_s(f_3, 1)$ of the multi-label decision table T^0
表 12. 多值决策表 T^0 的子表 $T_s(f_3, 1)$

$T_s(f_3, 1)$	f_1	f_2	f_3	d
r_2	0	1	1	{1,2}
r_3	1	0	1	{1,3}
r_5	0	0	1	{2}

Table 13. A subtable $T_s(f_1, 0)(f_2, 0)$ of the multi-label decision table T^0
表 13. 多值决策表 T^0 的子表 $T_s(f_1, 0)(f_2, 0)$

$T_s(f_1, 0)(f_2, 0)$	f_1	f_2	f_3	d
r_1	0	0	0	{1}
r_5	0	0	1	{2}

Table 14. A subtable $T_s(f_1, 0)(f_2, 1)$ of the multi-label decision table T^0
表 14. 多值决策表 T^0 的子表 $T_s(f_1, 0)(f_2, 1)$

$T_s(f_1, 0)(f_2, 1)$	f_1	f_2	f_3	d
r_2	0	1	1	{1,2}

Table 15. A subtable $T_s(f_1, 0)(f_3, 0)$ of the multi-label decision table T^0
表 15. 多值决策表 T^0 的子表 $T_s(f_1, 0)(f_3, 0)$

$T_s(f_1, 0)(f_3, 0)$	f_1	f_2	f_3	d
r_1	0	0	0	{1}

Table 16. A subtable $T_s(f_1, 0)(f_3, 1)$ of the multi-label decision table T^0
表 16. 多值决策表 T^0 的子表 $T_s(f_1, 0)(f_3, 1)$

$T_s(f_1, 0)(f_3, 1)$	f_1	f_2	f_3	d
r_2	0	1	1	{1,2}
r_5	0	0	1	{2}

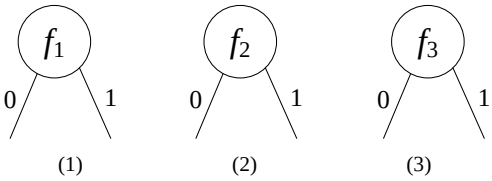

Figure 2. Some subtrees of the decision tree, (1) Γ_{f1}, (2) Γ_{f2}, (3) Γ_{f3}
图 2. 决策树子树。(1) Γ_{f1}；(2) Γ_{f2}；(3) Γ_{f3}

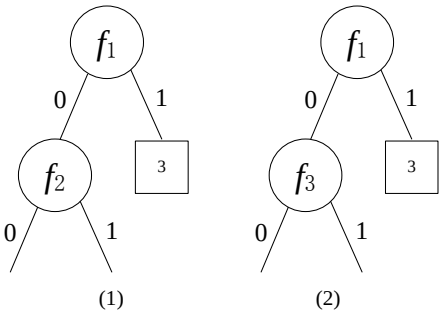

Figure 3. Some subtrees of the decision tree, (1) $\Gamma_{f1,f2}$, (2) $\Gamma_{f1,f3}$
图 3. 决策树子树。(1) $\Gamma_{f1,f2}$；(2) $\Gamma_{f1,f3}$

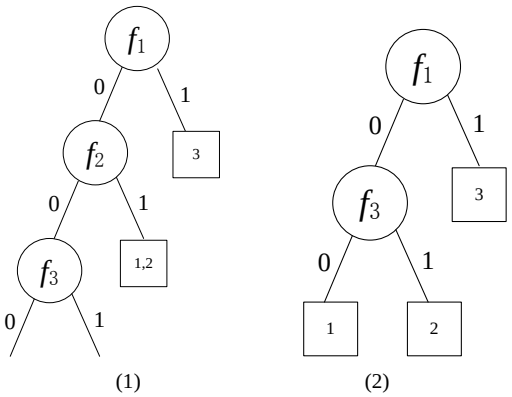

Figure 4. Some subtrees of the decision tree, (1) $\Gamma_{f1,f2,f3}$, (2) $\Gamma_{f1,f3}$
图 4. 决策树子树。(1) $\Gamma_{f1,f2,f3}$；(2) $\Gamma_{f1,f3}$

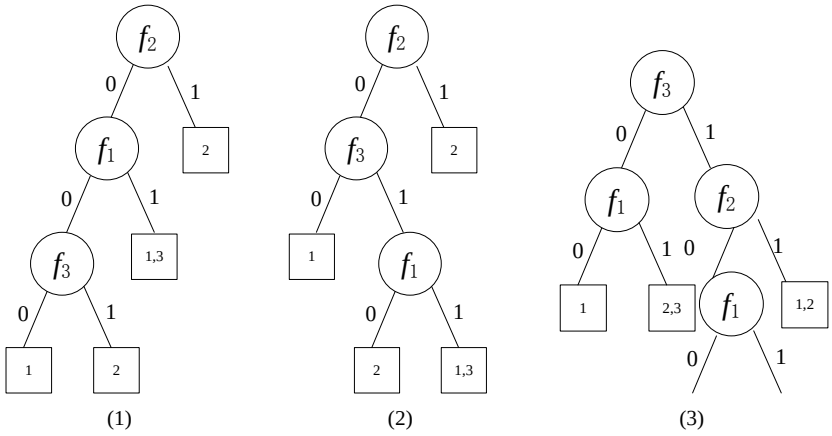

Figure 5. Some subtrees of the decision tree, (1) $\Gamma_{f2,f1,f3}$, (2) $\Gamma_{f2,f3,f1}$, (3) $\Gamma_{f3,f2,f1}$
图 5. 决策树子树。(1) $\Gamma_{f2,f1,f3}$，(2) $\Gamma_{f2,f3,f1}$，(3) $\Gamma_{f3,f2,f1}$

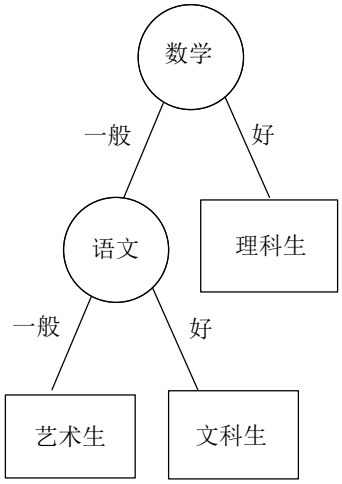

Figure 6. The result of classification by dynamic programming algorithm
图 6. 动态规划算法的分类结果

4.3. 性能比较

本节利用已有的贪心算法构造多值决策表的决策树，通过比较进一步说明动态规划算法的构造性能。在贪心算法中，我们需要选择更小的子表，直到我们得到可以用于标记叶子节点的退化表，使用自顶向下的顺序构造决策树。在每一步中进行的贪心选择属性是根据不确定性测度和不纯度函数的类型。具体构造过程如下：

Step1：选择的子表 T^0 为表 T，即 $T^0 = T$，树 G 中只有一个根节点 v，所以用 T^0 标记节点 v。通过使用表 6 中的数据计算不纯度函数 $I(T^0, f_1)$、$I(T^0, f_2)$、$I(T^0, f_3)$的值分别为 3，3，7，由于 $I(T^0, f_1) = I(T^0, f_2)$相等，所以取 f_i 中 i 值小的 $I(T^0, f_1)$，用 f_1 代替 T^0 标记节点 v，对于每个 δ，$\delta \in E(T', f_2) = \{0, 1\}$，在树 G 中加入节点 v_0，v_1，分别与连接节点 v 相连并将边标记为 0，1，分别用子表 $T(f_1, 0)$，$T(f_1, 1)$标记节点 v_0，v_1，结果如图 7 所示。

Step2：在子表 $T(f_1, 0)$,$T(f_1, 1)$中任选选择一个如 $T(f_1, 0)$即 $T^0 = T(f_1, 0)$。使用表 7~11 中的数据计算不纯度函数 $I(T^0, f_2)$、$I(T^0, f_3)$的值分别为 5 和 0，所以选择 f_3，用 f_3 代替 T^0 标记节点 v_0，分别用子表 $T(f_1, 0)$ $(f_3, 0)$、$T(f_1, 1)$ $(f_3, 1)$标记节点 v_0，v_1，结果如图 8 所示。

重复上述过程得到最终的决策树(图 9)。因此，多值决策表(表 5)根据贪心算法构造出的决策树(见图 10)。

由动态规划算法构造多值决策表的决策树(图 6)与贪心算法构造多值决策表的决策树(图 10)进行对比，可得出贪心算法构造出的决策树不是最小的。

5. 总结

决策树技术在数据挖掘的分类领域应用极其广泛，从多值决策表中挖掘潜在的信息是当前研究热点。多值决策表中每行记录包含多个决策值，多个决策属性用一个集合表示。针对已有的启发式算法，如贪心算法，由于性能不稳定的特点，该算法获得的决策树规模变化较大。

本文主要研究了多值决策表的决策树最优解问题，通过对动态规划算法和贪心算法进行比较，提出了利用动态规划算法使多值决策表的决策树规模最小。通过实例的演示，说明了决策树的最小化有利于人们对多值决策表进行决策，提高决策预测的准确率和降低决策树的复杂度。

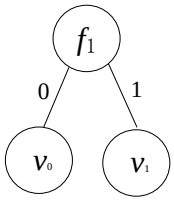

Figure 7. Decision tree for the first step
图 7. 第一步贪心选择构造的决策树

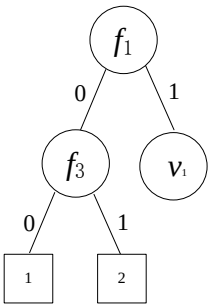

Figure 8. Decision tree for the second step
图 8. 第二步贪心选择构造的决策树

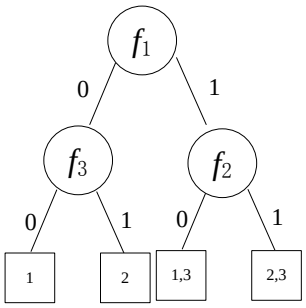

Figure 9. Decision tree by greedy algorithm
图 9. 贪心选择构造的决策树

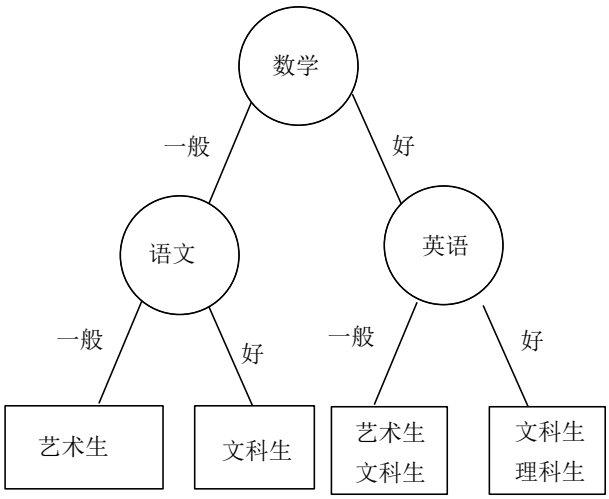

Figure 10. The result of classification by greedy algorithm
图 10. 贪心算法的分类结果

基金项目

国家自然科学基金面上项目(61572442, 61272130)浙江省科技厅公益性技术应用研究项目(2015C33085)浙江省教育厅项目(Y201226127)。

参考文献 (References)

[1] Boutell, M.R., Luo, J., Shen, X. and Brown, C.M. (2004) Learning Multi-Label Scene Classification. *Pattern Recognition*, **37**, 1757-1771. http://dx.doi.org/10.1016/j.patcog.2004.03.009

[2] Wieczorkowska, A., Synak, P., Lewis, R.A. and Ras, Z.W. (2005) Extracting Emotions from Music Data. *ISMIS*, Volume 3488 of the series Lecture Notes in Computer Science, 456-465.

[3] Blockeel, H., Schietgat, L., Struyf, J., Dzeroski, S. and Clare, A. (2006) Decision Trees for Hierarchical Multilabel Classification: A Case Study in Functional Genomics. In: Fürnkranz, J., Scheffer, T. and Spiliopoulou, M., Eds., *Proceedings PKDD*, ser. LNCS, Springer, Berlin, Vol. 4213, 18-29.

[4] Zhou, Z.-H., Jiang, K. and Li, M. (2005) Multi-Instance Learning Basedweb Mining. *Applied Intelligence*, **22**, 135-147. http://dx.doi.org/10.1007/s10489-005-5602-z

[5] Moshkov, M. and Zielosko, B. (2011) Combinatorial Machine Learning -A Rough Set Approach. ser. Studies in Computational Intelligence, Springer, Vol. 360. http://dx.doi.org/10.1007/978-3-642-20995-6

[6] Comité, F.D., Gilleron, R. and Tommasi, M. (2003) Learning Multi-Label Alternating Decision Trees from Texts and Data. *Proceedings of* 3*rd International Conference*, *MLDM* 2003, Leipzig, 5-7 July 2003, 35-49. http://dx.doi.org/10.1007/3-540-45065-3

[7] Loza Mencía, E. and Fürnkranz, J. (2008) Pairwise Learning of Multilabel Classifications with Perceptrons. *IEEE International Joint Conference on Neural Networks*, 1-8 June 2008, 2899-2906. http://dx.doi.org/10.1109/IJCNN.2008.4634206

[8] Tsoumakas, G., Katakis, I. and Vlahavas, I.P. (2010) Mining Multi-Label Data. In: Maimon, O. and Rokach, L., Eds., *Data Mining and KnowledgeDiscovery Handbook*, Tel Aviv University, 667-685.

[9] Zhou, Z.-H., Zhang, M.-L., Huang, S.-J. and Li, Y.-F. (2012) Multi-Instance Multi-Label Learning. *Artificial Intelligence*, **176**, 2291-2320. http://dx.doi.org/10.1016/j.artint.2011.10.002

[10] Azad, M., Chikalov, I., Moshkov, M. and Zielosko, B. (2012) Greedy Algorithm for Construction of Decision Trees for Tables with Many-Valued Decisions. *Proceedings of the* 21*st International Workshop on Concurrency, Specification and Programming*, Berlin, 26-28 September 2012, ser. CEUR Workshop Proceedings, L. Popova-Zeugmann, Ed.CEUR-WS.org, 2012, Vol. 928.

[11] Azad, M., Chikalov, I. and Moshkov, M. (2013) Three Approaches to Deal within Consistent Decision Tables—Comparison of Decision Tree Complexity. *RSFDGrC*, Halifax, 11-14 October 2013, 46-54. http://dx.doi.org/10.1007/978-3-642-41218-9

[12] Tsoumakas, G. and Katakis, I. (2007) Multi-Label Classification: An Overview. *IJDWM*, **3**, 1-13. http://dx.doi.org/10.4018/jdwm.2007070101

[13] Zhu, X. and Goldberg, A.B. (2009) Introduction to Semi-Supervised Learning, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, San Rafael, California.

[14] Cour, T., Sapp, B., Jordan, C. and Taskar, B. (2009) Learning from Ambiguously Labeled Images. *CVPR*, Miami, Florida, 20-25 July 2009, 919-926.

[15] Hüllermeier, E. and Beringer, J. (2006) Learning from Ambiguously Labeled Examples. *Intelligent Data Analysis*, **10**, 419-439.

[16] Jin, R. and Ghahramani, Z. (2002) Learning with Multiple Labels. *NIPS*, Vancouver, British Columbia, 9-14 December 2002, 897-904.

[17] Moshkov, M. and Chikalov, I. (2000) On Algorithm for Constructing of Decision Trees with Minimal Depth. *Fundamenta Informaticae*, **41**, 295-299.