软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(3):604–619 [doi: 10.13328/j.cnki.jos.005691]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

劣质数据上代价敏感决策树的建立*

齐志鑫, 王宏志, 周 雄, 李建中, 高 宏

(哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

通讯作者: 王宏志, E-mail: wangzh@hit.edu.cn

摘 要: 代价敏感决策树是以最小化误分类代价和测试代价为目标的一种决策树.目前,随着数据量急剧增长,劣质数据的出现也愈发频繁.在建立代价敏感决策树时,训练数据集中的劣质数据会对分裂属性的选择和决策树结点的划分造成一定的影响.因此在进行分类任务前,需要提前对数据进行劣质数据清洗.然而在实际应用中,由于数据清洗工作所需要的时间和金钱代价往往很高,许多用户给出了自己可接受的数据清洗代价最大值,并要求将数据清洗的代价控制在这一阈值内.因此除了误分类代价和测试代价以外,劣质数据的清洗代价也是代价敏感决策树建立过程中的一个重要因素.然而,现有代价敏感决策树建立的相关研究没有考虑数据质量问题.为了弥补这一空缺,着眼于研究劣质数据上代价敏感决策树的建立问题.针对该问题,提出了 3 种融合数据清洗算法的代价敏感决策树建立方法,并通过实验证明了所提出方法的有效性.

关键词: 代价敏感决策树;劣质数据;数据清洗;误分类代价;测试代价

中图法分类号: TP311

中文引用格式: 齐志鑫,王宏志,周雄,李建中,高宏.劣质数据上代价敏感决策树的建立.软件学报,2019,30(3):604–619. http://www.jos.org.cn/1000-9825/5691.htm

英文引用格式: Qi ZX, Wang HZ, Zhou X, Li JZ, Gao H. Cost-sensitive decision tree induction on dirty data. Ruan Jian Xue Bao/Journal of Software, 2019,30(3):604–619 (in Chinese). http://www.jos.org.cn/1000-9825/5691.htm

Cost-sensitive Decision Tree Induction on Dirty Data

QI Zhi-Xin, WANG Hong-Zhi, ZHOU Xiong, LI Jian-Zhong, GAO Hong

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

Abstract: Cost-sensitive decision tree is a kind of decision tree which maximizes the sum of misclassification costs and test costs. Recently, with the explosive growth of data size, dirty data appears more frequently. In the process of cost-sensitive decision tree induction, dirty data in training datasets have negative impacts on selection of splitting attributes and division of decision tree nodes. Therefore, dirty data cleaning is necessary before classification tasks. Nevertheless, in practice, many users provide an acceptable threshold of data cleaning costs since time costs and expenses of data cleaning are expensive. Therefore, in addition to misclassification cost and test cost, data-cleaning cost is also an essential factor in cost-sensitive decision tree induction. However, existing researches have not considered data quality in the problem. To fill this gap, this study aims to focus on cost-sensitive decision tree induction on dirty data. Three decision tree induction methods integrated with data cleaning algorithms are presented. Experimental results demonstrate the effective of the proposed approaches.

Key words: cost-sensitive decision tree; dirty data; data cleaning; misclassification cost; test cost

分类是数据挖掘和机器学习领域中常见的一类任务,该任务从训练数据中提取模型,再利用该模型预测未

* 基金项目: 国家自然科学基金(U1509216, 61472099); 国家科技支撑计划(2015BAH10F01)

Foundation item: National Natural Science Foundation of China (U1509216, 61472099); National Sci-Tech Support Plan (2015BAH10F01)

本文由智能数据管理与分析技术专刊特约编辑樊文飞教授、王国仁教授、王朝坤副教授推荐.

收稿时间: 2018-07-19; 修改时间: 2018-09-20; 采用时间: 2018-11-01

知的数据类别[1].目前,有很多用于完成分类任务的方法被提出,如决策树[2]、贝叶斯网[3]、神经网络[4]、基于实例的推理[5]等.在这些方法中,决策树凭借易解释、计算效率高、能够生成易理解的分类规则等优势得到了众多关注,并被广泛应用于分类任务中[6].

起初,大量决策树研究着眼于最大化分类准确率或最小化分类误差[7−9].然而,由于一个分类任务可能产生多种类型的代价,越来越多的研究工作关注于代价敏感决策树的建立[10−12].在目前研究的代价类型中,两种最常见的代价是误分类代价和测试代价[13].误分类代价是指一个属于类 j 的实例被误分类为类 i 的代价,而误分类所产生的代价常常是不均衡的.例如,将一个病人诊断为健康的代价往往比将一个健康的人诊断为病人的代价大很多.除了误分类代价,每一次测试也关联着相应的代价.例如在医疗诊断中,每一次血常规检测都存在一个相关联的代价[1].

现如今,随着数据量急剧增长,劣质数据的出现也愈发频繁[14].在建立代价敏感决策树时,训练数据集中的劣质数据会对分裂属性的选择和决策树结点的划分造成一定的影响.因此在进行分类任务前,需要提前对数据进行劣质数据清洗.然而在实际应用中,由于数据清洗工作所需要的时间和金钱代价往往很高,许多用户给出了自己可接受的数据清洗代价最大值,并要求将数据清洗的代价控制在这一阈值内[15].因此,除了误分类代价和测试代价以外,劣质数据的清洗代价也是代价敏感决策树建立过程中的一个重要因素.然而,现有代价敏感决策树建立的相关研究没有考虑数据质量问题.为了弥补这一空缺,本文将劣质数据的清洗代价纳入代价敏感决策树建立问题的考虑因素中,该问题带来了以下挑战.

(1) 由于清洗后的数据将用于建立代价敏感决策树,第 1 个挑战是如何将代价敏感决策树的建立目标融合到数据清洗方法中;

(2) 由于不同用户设定的数据清洗代价阈值不同,不同数据集中劣质数据所占比率不同,第 2 个挑战是针对不同的情况,如何选择最优的代价敏感决策树建立方法.

鉴于这两点挑战,本文从问题定义入手,将用户设定的数据清洗代价阈值纳入到代价敏感决策树建立的问题中.然后,针对这一问题给出融合数据清洗算法的代价敏感决策树建立方法.最后,测试改变用户设定的数据清洗代价阈值或改变数据集中劣质数据比率时不同方法的总代价、分类准确率和效率,从而得出不同情况下最优的代价敏感决策树建立方法.本文的主要贡献有以下几点.

(1) 研究了劣质数据上代价敏感决策树的建立问题,是目前首个考虑到代价敏感决策树建立过程中的数据质量问题的工作;

(2) 针对劣质数据上的代价敏感决策树建立问题,本文提出了 3 种融合数据清洗算法的代价敏感决策树建立方法,即融合基于分裂属性收益的分步清洗算法的代价敏感决策树建立方法、融合基于分裂属性收益和清洗代价的一次性清洗算法的代价敏感决策树建立方法、融合基于分裂属性收益和清洗代价的分步清洗算法的代价敏感决策树建立方法;

(3) 本文通过大量实验验证了所提出的 3 种代价敏感决策树建立方法的总代价、分类准确率和分类效率,并给出了不同情况下最优的代价敏感决策树建立方法.

本文第 1 节对代价敏感决策树建立问题相关的研究工作进行总结.第 2 节对本文问题相关的定义进行介绍.第 3 节对本文提出的 3 种融合数据清洗算法的代价敏感决策树建立方法进行详细地阐述.第 4 节对本文的实验评估过程和实验结果进行讨论和分析,并给出实验结论.第 5 节总结全文,并对未来值得关注的研究方向进行初步探讨.

1 相关工作

目前,代价敏感决策树建立问题的研究目标主要包括 3 种:第 1 种是最小化决策树的误分类代价;第 2 种是最小化决策树的测试代价;第 3 种是最小化决策树的误分类代价和测试代价总和[13].本文选择的研究目标是第 3 种.

针对不同的问题目标,许多代价敏感决策树建立方法被提出.这些方法可以分为两大类.

- 第 1 类是采用贪心的方法来建立一棵单独的决策树,例如:CS-ID3 算法[16]采用基于熵的选择方法在决策树建立过程中最小化代价,AUCSplit 算法[17]在决策树建立之后最小化代价;
- 第 2 类是非贪心的方法,该方法生成多个决策树,例如遗传算法 ICET[18]、将现有基于准确率的方法包装在一起的 MetaCost 算法[19].

按照时间顺序来看,Hunt 等人首先发现了误分类和测试对人们的决策存在一定的影响,并提出了概念学习系统框架[20].随后,ID3 算法[21]采用了概念学习系统框架的部分思想,并使用了信息论评估参数来选择特征.在信息论方法被提出后,许多在决策树建立过程中最小化代价的方法被陆续提出,如 CS-ID3 算法[16]、EG2 算法[22].然后,在决策树建立后最小化代价的方法被提出,如 AUCSplit 算法[17].之后,遗传方法如 ICET 算法[18]、提升法如 UBoost 算法[23]、装袋法如 MetaCost 算法[19]等被提出.随后,多重结构的方法如 LazyTree 算法[24]、随机方法如 ACT 算法[25]、TATA 算法[26]被陆续提出.

在最小化代价敏感决策树的误分类代价和测试代价的基础上,有研究着眼于包含其他限制因素的代价敏感决策树建立问题.例如:由于有些分类任务需要在规定的时间内完成,在时间限制下的代价敏感决策树建立方法被提出[1].有时用户会对分类任务的准确率提出预期的要求,因此面向用户需求的代价敏感决策树建立方法被提出[27].然而,目前并未有研究关注劣质数据上的代价敏感决策树建立.因此,本文将弥补这一空缺.

2 问题定义

本节将对劣质数据上代价敏感决策树建立问题的相关定义进行介绍.

2.1 决策树

决策树 T 具有树结构,是有向无环图的一个特例.决策树中包含根结点、内部结点和叶子结点的有限集合、连接两个结点的边的集合.图 1 是决策树的实例.

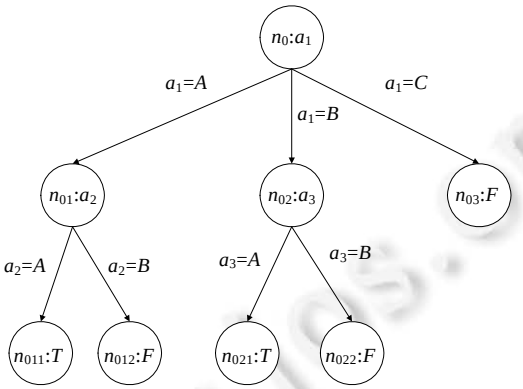


Fig.1 An instance of decision tree

图 1 决策树实例

如图 1 所示,n_i 表示第 i 个结点.如果 n_i 是内部结点,那么该结点会关联着属性 $a(n_i)$,$a(n_i)$被称为在结点 n_i 中被用到的测试属性.如果 n_i 是叶子结点,那么该结点关联着一个类别标签值.本文用 $inter(T)$表示 T 中所有内部结点的集合,用 $leaf(T)$表示 T 中所有叶子结点的集合,T_i 表示 T 中根为 n_i 的子树.

本文用有序对(n_i,n_j)表示决策树中一条关联着结点 n_i 和 n_j 的边,其中,n_i 是 n_j 的父结点.当 n_i 是关联测试属性 $a(n_i)$的内部结点时,边(n_i,n_j)关联一个单独的值或一个值集合,这些值都是 $a(n_i)$可能的属性值.决策树中的一条路径是边的一个序列,本文用 $path(n_i,n_j)$表示从 n_i 到 n_j 的路径.假设 n_i 到 n_j 之间的序列是 $n_1,n_2,\ldots,n_m$,那么 $path(n_i,n_j)$=$\{(n_i,n_1),(n_1,n_2),\ldots,(n_m,n_j)\}$,本文用 $n(n_i,n_j)$=$\{n_1,n_2,n_3,\ldots,n_m\}$表示从 n_i 到 n_j 的路径上所有中间结点的集合.

决策树分类器通过在训练数据集 D 上学习构建而成,d_k 表示 D 中的第 k 条记录.每条记录由属性值集合和类别标签构成.本文用 a 表示属性值集合,a_x 表示 a 中的第 x 个属性.表 1 是训练数据集的一个实例,每一条记录

由 ID、5 个属性和 1 个类别标签组成.

Table 1 An instance of training dataset

表 1 训练数据集实例

ID	a_1	a_2	a_3	a_4	a_5	类别
1	*A*	*A*	*B*	*C*	*C*	*T*
2	*A*	*B*	*A*	*A*	*C*	*F*
3	*A*	*B*	*C*	*B*	*A*	*F*
4	*B*	*A*	*A*	*C*	*A*	*T*
5	*B*	*B*	*A*	*B*	*C*	*T*
6	*B*	*A*	*B*	*B*	*C*	*F*
7	*C*	*A*	*C*	*A*	*A*	*F*
8	*C*	*B*	*B*	*A*	*C*	*F*

本文用 $D(n_i)$表示 D 中属性值遵循 $path(root,n_i)$中边上属性值的记录集合,这说明 $D(n_i)$是满足 $path(root,n_i)$上所有测试的记录集合.$D(root)$表示整个训练数据集 D,$|D(n_i)|$表示 $D(n_i)$中的记录个数.根据表 1 中的训练数据集构建的决策树如图 2 所示.

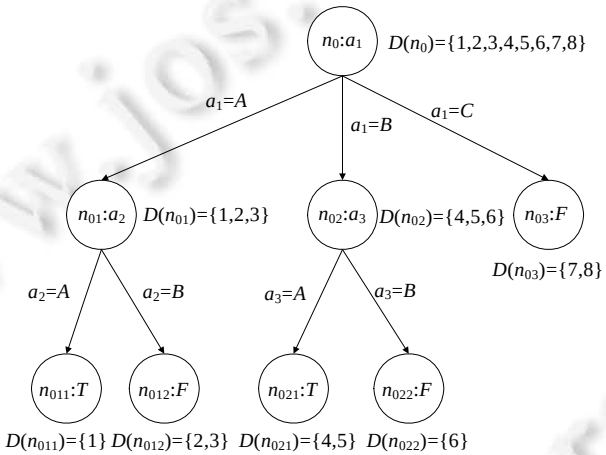


Fig.2 Decision tree trained by Table 1

图 2 根据表 1 训练成的决策树

2.2 误分类代价和测试代价

误分类代价是指一个属于类 j 的实例被误分类为类 i 的代价.假设训练数据集中有 m 个类别,那么误分类代价矩阵可以用 $m\times m$ 的矩阵表示,见表 2.本文用 $ClassCost(i,j)$表示当一个实例属于类 j 时,将其误分类为类 i 的代价.从表 2 可知,$ClassCost(T,T)=0$,$ClassCost(T,F)=10$,$ClassCost(F,T)=20$,$ClassCost(F,F)=0$.

Table 2 A matrix of misclassification costs

表 2 误分类代价矩阵

类别	*T*	*F*
T	0	10
F	20	0

当为叶子结点分配一个标签时,可能会导致该叶子结点关联的记录产生一定的误分类代价.因此,本文用 $ClassCost(n_i,l_j)$表示给结点 n_i 分配标签为 l_j 的叶子结点时产生的误分类代价,即:

$$ClassCost(n_i,l_j)=\sum_{\forall d_k\in D(n_i)}ClassCost(l_j,l(d_k)).$$

例如,对于表 1 中的训练数据,$ClassCost(root,T)=0+10+10+0+0+10+10+10=50$,$ClassCost(root,F)=20+0+0+20+20+0+0+0=60$.

本文用 $TestCost(a_x)$表示对属性 a_x进行测试所需要的代价,用 $ArrCost(n_i,T)$表示一条记录从 T的根结点到达 n_i 所需的总测试代价,即 $ArrCost(N_I,T)=\sum_{n_j\in N(root,n_i)\cup\{root\}}TestCost(a(n_j))$.表 1 中属性的测试代价见表 3.从表 3 可知,$ArrCost(n_{01},T)$=4,$ArrCost(n_{011},T)$=4+6=10.

Table 3 TestCost of attributes

表 3 属性的测试代价

属性	测试代价
a_1	4
a_2	6
a_3	3
a_4	2
a_5	8

每一条记录都会经过从决策树根结点到叶子结点的测试,因此,结点 n_i 中的记录到达 n_i 所花费的总测试代价为 $ArrCost(n_i,T)\times|D(n_i)|$.此外,$n_i$ 获得一个类别标签时会产生一定的误分类代价.本文用 $NodeCost(n_i)$表示误分类代价和测试代价的总代价.如果将结点 n_i 作为一个叶子结点,则

$$NodeCost(n_i)=\min_{\forall l_j}(ClassCost(n_i,l_j))+ArrCost(n_i,T)\times|D(n_i)|.$$

例如,对于图 2 中的决策树,$NodeCost(root)$=50+0×8=50,$NodeCost(n_{01})$=20+4×3=32.

图 2 中其余结点的 *NodeCost* 值见表 4.

Table 4 NodeCost of nodes

表 4 结点的 NodeCost 值

结点 n_i	n_0	n_{01}	n_{02}	n_{03}	n_{011}	n_{012}	n_{021}	n_{022}
$NodeCost(n_i)$	50	32	22	8	10	20	14	7

本文用 $TreeCost(T)$表示决策树 T的总代价,即 $TreeCost(T)=\sum_{n_i\in leaf(T)}NodeCost(n_i)$.例如,图 2 中决策树的总代价为 $TreeCost(T)=NodeCost(n_{011}+n_{012}+n_{021}+n_{022}+n_{03})$=10+20+14+7+8=59.

2.3 检测代价和修复代价

劣质数据清洗工作往往由错误检测和修复两部分组成[28].在对训练数据集进行清洗的过程中,首先需要对数据集中的劣质数据进行检测,然后针对检测到的劣质数据进行相应的修复.数据集中的每一个属性都对应着检测该属性值中可能存在的错误所需要花费的代价和修复该属性错误值所需要花费的代价.

本文用 $DetCost(a_x)$表示对属性 a_x中的每个值进行错误检测所需要花费的代价.因此,对属性 a_x中全部属性值进行劣质数据检测所需要的代价是 $DetCost(a_x)\times|a_x|$.如果检测到属性 a_x 中的劣质数据个数为$|Er(a_x)|$,那么对属性 a_x中的全部劣质数据值进行修复所需要的代价是 $Rep(a_x)\times|Er(a_x)|$,其中,$Rep(a_x)$是修复属性 a_x中的每个劣质数据值所需要的代价.在本文中,x_d表示属性 a_x中完成了劣质数据检测的属性值个数,x_r表示属性 a_x中完成了劣质数据修复的属性值个数.然而在实际应用中,由于数据清洗工作所需要的时间和金钱代价往往很高,许多用户给出了自己可接受的数据清洗代价最大值.因此,本文用 Max*Cost* 表示用户可接受的数据清洗代价阈值,并将数据清洗的代价控制在这一阈值内.

2.4 劣质数据上代价敏感决策树建立问题

给定一个训练数据集、用户设定的数据清洗代价最大值、误分类代价矩阵、每个属性对应的测试代价、检测代价和修复代价,劣质数据上代价敏感决策树建立问题定义如下:基于训练数据集,建立一个代价敏感决策树,使得该决策树的误分类代价和测试代价总和最小,且清洗该数据集的检测代价和修复代价总和不超过用户设定的数据清洗阈值.

该问题的形式化描述是已知训练数据集 D,数据清洗代价最大值 Max*Cost*,误分类代价 $ClassCost(n_i,l_j)$,测试代价 $TestCost(a_x)$,检测代价 $DetCost(a_x)$和修复代价,建立 T,使得:

$$\min TreeCost(T) \quad \text{s.t.} \quad Det(a_x)\times x_d + Rep(a_x)\times x_r \leqslant \text{Max}Cost.$$

其中,

$$TreeCost(T)=\sum_{n_i\in leaf(T)} NodeCost(n_i);$$

$$NodeCost(n_i)=\min_{\forall l_j}(ClassCost(n_i,l_j))+ArrCost(n_i,T)\times|D(n_i)|;$$

$$ArrCost(N_I,T)=\sum_{n_j\in N(root,n_i)\cup\{root\}} TestCost(a(n_j)).$$

3 融合数据清洗算法的代价敏感决策树建立方法

为了解决劣质数据上代价敏感决策树的建立问题,本文提出了 3 种融合数据清洗算法的代价敏感决策树建立方法,即融合基于分裂属性收益的分步清洗算法的代价敏感决策树建立方法、融合基于分裂属性收益和清洗代价的一次性清洗算法的代价敏感决策树建立方法和融合基于分裂属性收益和清洗代价的分步清洗算法的代价敏感决策树建立方法.本节将分别对这 3 种代价敏感决策树建立方法进行阐述,并对各个方法的适用情况进行讨论.

3.1 融合基于分裂属性收益的分步清洗算法的代价敏感决策树建立方法

由于代价敏感决策树的建立目标是最小化误分类代价和测试代价的总和,因此可以将误分类代价和测试代价总和的减少程度作为决策树分裂属性选取的标准.本文定义了分裂属性收益的概念来表示误分类代价和测试代价总和的减少程度.

本文用 $Benefit(a_x,n_i)$表示用属性 a_x 分裂结点 n_i 的收益,结点 n_i 会被分裂为若干个孩子结点.分裂属性收益 $Benefit(a_x,n_i)$定义为

$$Benefit(a_x,n_i)=NodeCost(n_i)-\sum NodeCost(child(n_i)).$$

例如在图 2 中,$NodeCost(n_{01})$=32,$NodeCost(n_{011})$=10,$NodeCost(n_{012})$=20.因此,$Benefit(a_2,n_{01})$=32−(10+20)=2.

融合基于分裂属性收益的分步清洗算法的代价敏感决策树建立方法的过程如下:在决策树建立的每一步中,选择收益最大的分裂属性;先对该分裂属性中的值进行劣质数据检测和修复,再将该属性用于决策树的分裂过程中;当所花费的清洗代价达到用户设定的最大值后,不再对后续的分裂属性进行清洗,直接用于决策树的分裂过程中;当结点中所有记录所对应的类别标签相同,或没有分裂属性的收益为正值时不再对该结点进行分裂,将其标记为叶子结点,其类别标签为使得误分类代价最小的类别.

上述过程中,基于分裂属性收益的分步清洗算法见算法 1.算法的输入是待清洗的训练数据集、用户给定的清洗代价阈值、数据集中每个属性对应的劣质数据检测代价和修复代价.首先,对根结点的分裂属性收益进行计算,选取收益最大的属性作为分裂属性(第 1 行~第 5 行).对该属性中的属性值进行劣质数据检测和修复,并计算剩余的清洗代价是否足够对训练数据集中该属性的全部属性值进行检测:如果足够,那么对该属性的所有属性值进行劣质数据检测(第 6 行~第 9 行);如果不足够,那么对该属性部分属性值进行检测(第 10 行~第 14 行).在检测出分裂属性中包含的劣质数据后,对劣质数据进行修复,并计算剩余的清洗代价是否足够对全部劣质数据进行修复:如果足够,那么对该属性中包含的所有劣质数据进行修复(第 15 行~第 17 行);如果不足够,那么对部分劣质数据进行修复(第 18 行~第 21 行).然后,继续为下一个结点寻找分裂属性并清洗(第 22 行).当清洗代价达到用户给定的阈值或决策树中全部结点都被清洗完毕后,返回清洗后的训练数据集(第 23 行).

算法 1. 基于分裂属性收益的分步清洗算法.

输入:训练数据集 $D_{m\times n}$,清洗代价阈值 Max*Cost*,每个属性 $x\in\{1,2,\ldots,n\}$的劣质数据检测代价 $Det(a_x)$和修复代价 $Rep(a_x)$;

输出:清洗后的训练数据集 $D_{m\times n}$.

1: $remain\leftarrow$Max*Cost*

2: $n_i\leftarrow n_0$

3: **While** ((*remain*>0) && (n_i!=*null*))

4: $Benefit(a_x,n_i)\leftarrow NodeCost(n_i)-\sum NodeCost(child(n_i))$

5: $a'_x\leftarrow \arg\max_{a_x} Benefit(a_x,n_i)$

6: **If** $\left\lceil \frac{remain}{Det(a'_x)} \right\rceil \geqslant m$

7: $Detect(\sum_{u=1}^{m}[a'_x][t_u])$

8: $e\leftarrow |Er(a'_x)|$

9: $remain\leftarrow remain-Det(a'_x)\times m$

10: **Else If** $\left\lceil \frac{remain}{Det(a'_x)} \right\rceil \geqslant 1$

11: $q\leftarrow \left\lceil \frac{remain}{Det(a'_x)} \right\rceil$

12: $Detect(\sum_{u=1}^{q}[a'_x][t_u])$

13: $e\leftarrow |Er(a'_x)|$

14: $remain\leftarrow remain-Det(a'_x)\times q$

15: **If** $\left((e>1)\ \&\&\ \left(\left\lceil \frac{remain}{Det(a'_x)} \right\rceil \geqslant e\right)\right)$

16: $Repair(\sum_{u=1}^{e}[a'_x][t_u])$

17: $remain\leftarrow remain-Rep(a'_x)\times e$

18: **Else If** $\left((e>1)\ \&\&\ \left(\left\lceil \frac{remain}{Rep(a'_x)} \right\rceil \geqslant 1\right)\right)$

19: $r\leftarrow \left\lceil \frac{remain}{Rep(a'_x)} \right\rceil$

20: $Repair(\sum_{u=1}^{r}[a'_x][t_u])$

21: $remain\leftarrow remain-Rep(a'_x)\times r$

22: $n_i\leftarrow n_{i+1}$

23: **Return** $D_{m\times n}$

算法 1 的时间复杂度取决于决策树中的结点个数 N、训练数据集中的属性个数 n 和记录个数 m.假设劣质数据检测函数 $Detect(x)$的时间复杂度为 $O(f(x))$,劣质数据修复函数 $Repair(x)$的时间复杂度为 $O(g(x))$,当 $\max(f(x),g(x))\geqslant n\log n$ 时,算法 1 的时间复杂度是 $O(N\max(f(x),g(x)))$;当 $\max(f(x),g(x))<n\log n$ 时,算法 1 的时间复杂度是 $O(Nn\log n)$.

融合基于分裂属性收益的分步清洗算法的代价敏感决策树建立方法尽可能地将代价敏感决策树中前几层的分裂属性进行清洗后再用于分裂过程.由于决策树中层次低的分裂结点重要性高于层数高的分裂结点,所以该方法保证了决策树结点分裂的有效性,有效降低了决策树的误分类代价和测试代价总和,使得决策树在后续的分类任务中受训练数据集中的劣质数据影响较小.然而,当用户设定的清洗代价阈值较小时,该方法可能会对清洗代价高的属性优先进行清洗,使得能够被清洗到的属性个数较少,从而导致在这种情况下,该方法对劣质数据的清洗效果不大.

3.2 融合基于分裂属性收益和清洗代价的一次性清洗算法的代价敏感决策树建立方法

为了解决融合基于分裂属性收益的分步清洗算法的代价敏感决策树建立方法所存在的问题,本文将数据清洗代价考虑到数据清洗算法中,希望选择收益高且清洗代价低的属性优先进行清洗.本文定义了收益与清洗代价比来平衡分裂属性收益和清洗代价这两个冲突的因素,即 $Benefit(a_x,n_0)/(Det(a_x)+Rep(a_x))$,其中,$Benefit(a_x,$

n_0)是属性 a_x 在决策树根结点 n_0 时的分裂属性收益,$Det(a_x)$和 $Rep(a_x)$分别是对属性 a_x 的每个属性值进行劣质数据检测和修复的代价.

融合基于分裂属性收益和清洗代价的一次性清洗算法的代价敏感决策树建立方法的过程如下:将训练数据集中的各个属性按照收益与清洗代价比进行排序,按照顺序进行一次性数据清洗.在每一次选择分裂属性时,对每个属性重新计算分裂属性收益 $Benefit(a_x,n_0)$,选择收益最大的属性作为分裂属性.当结点中所有记录所对应的类别标签相同,或没有分裂属性的收益为正值时停止对该结点的分裂,将其标记为叶子结点,其类别标签为使得误分类代价最小的类别.

上述过程中基于分裂属性收益和清洗代价的一次性清洗算法见算法 2.算法的输入是待清洗的训练数据集、用户给定的清洗代价阈值、数据集中每个属性对应的劣质数据检测代价和修复代价.首先,对每个属性在根结点的收益与清洗代价比从大到小排序(第 1 行~第 4 行).按照顺序选取属性进行劣质数据检测和修复(第 5 行、第 6 行).计算剩余的清洗代价是否足够对训练数据集中该属性的全部属性值进行检测:如果足够,那么对该属性的所有属性值进行劣质数据检测(第 7 行~第 10 行);如果不足够,那么对该属性部分属性值进行检测(第 11 行~第 15 行).在检测出分裂属性中包含的劣质数据后,对劣质数据进行修复,并计算剩余的清洗代价是否足够对全部劣质数据进行修复:如果足够,那么对该属性中包含的所有劣质数据进行修复(第 16 行~第 18 行);如果不足够,那么对部分劣质数据进行修复(第 19 行~第 22 行).然后,按照顺序对下一个属性进行清洗(第 23 行).当清洗代价达到用户给定的阈值或全部属性都被清洗完毕后,返回清洗后的训练数据集(第 24 行).

算法 2. 基于分裂属性收益和清洗代价的一次性清洗算法.

输入:训练数据集 $D_{m\times n}$,清洗代价阈值 $MaxCost$,每个属性 $x\in\{1,2,\ldots,n\}$的劣质数据检测代价 $Det(a_x)$和修复代价 $Rep(a_x)$;

输出:清洗后的训练数据集 $D_{m\times n}$.

1: $remain\leftarrow \mathrm{Max}Cost$

2: $i\leftarrow 0$

3: $Benefit(a_x,n_0)\leftarrow NodeCost(n_0)-\sum NodeCost(child(n_0))$

4: $array\leftarrow sort\left(a_x,\dfrac{Benefit(a_x,n_0)}{Det(a_x)+Rep(a_x)}\right)$

5: **While** (($remain>0$) && ($i\leqslant array.length-1$)

6: $a'_x\leftarrow array[i]$

7: **If** $\left\lceil\dfrac{remain}{Det(a'_x)}\right\rceil\geqslant m$

8: $Detect(\sum_{u=1}^{m}[a'_x][t_u])$

9: $e\leftarrow|Er(a'_x)|$

10: $remain\leftarrow remain-Det(a'_x)\times m$

11: **Else If** $\left\lceil\dfrac{remain}{Det(a'_x)}\right\rceil\geqslant 1$

12: $q\leftarrow\left\lceil\dfrac{remain}{Det(a'_x)}\right\rceil$

13: $Detect(\sum_{u=1}^{q}[a'_x][t_u])$

14: $e\leftarrow|Er(a'_x)|$

15: $remain\leftarrow remain-Det(a'_x)\times q$

16: **If** $\left((e>1)\ \&\&\ \left(\left\lceil\dfrac{remain}{Rep(a'_x)}\right\rceil\geqslant e\right)\right)$

17: $Repair(\sum_{u=1}^{e}[a'_x][t_u])$

18: $remain \leftarrow remain - Rep(a'_x)\times e$

19: **Else If** $\left((e>1)\ \&\&\ \left(\left[\frac{remain}{Rep(a'_x)}\right]\geqslant 1\right)\right)$

20: $r \leftarrow \left[\frac{remain}{Rep(a'_x)}\right]$

21: $Repair(\sum_{u=1}^{r}[a'_x][t_u])$

22: $remain \leftarrow remain - Rep(a'_x)\times r$

23: $i \leftarrow i+1$

24: **Return** $D_{m\times n}$

算法 2 的时间复杂度取决于训练数据集中的属性个数 n 和记录个数 m.假设劣质数据检测函数 $Detect(x)$ 的时间复杂度为 $O(f(x))$,劣质数据修复函数 $Repair(x)$ 的时间复杂度为 $O(g(x))$,算法 2 的时间复杂度是 $O(n\log n\times\max(f(x),g(x)))$.

融合基于分裂属性收益和清洗代价的一次性清洗算法的代价敏感决策树建立方法尽可能地优先清洗收益大且清洗代价小的属性.该方法提前将数据集中的劣质数据清洗完毕,使得在决策树建立过程中能够直接利用清洗后的属性收益选取分裂属性,很大程度上降低了劣质数据对决策树的影响.此外,该方法采用一次性清洗策略,与分步清洗策略相比节约了时间.然而,该方法一次性清洗到的属性不一定都会在后续的决策树建立过程中被选作分裂属性.因此,在劣质数据比例较大的情况下,该方法可能没有清洗到决策树建立过程中的某些分裂属性,导致数据清洗过程对分类任务的作用不大.

3.3 融合基于分裂属性收益和清洗代价的分步清洗算法的代价敏感决策树建立方法

为了解决融合基于分裂属性收益和清洗代价的一次性清洗算法的代价敏感决策树建立方法所存在的问题,本文提出了融合基于分裂属性收益和清洗代价的分步清洗算法的代价敏感决策树建立方法.该方法的过程如下:在决策树建立的每一步中,选择收益与清洗比最大的分裂属性;对该属性中的值进行劣质数据检测和修复,再将该属性用于决策树的分裂过程中;所花费的清洗代价达到了用户设定的代价阈值时,不再对后续的分裂属性进行清洗;结点中所有记录所对应的类别标签相同,或没有分裂属性的收益为正值时,停止对该结点的分裂,将其标记为叶子结点,其类别标签为使得误分类代价最小的类别.

上述过程中,基于分裂属性收益和清洗代价的分步清洗算法见算法 3.算法的输入是待清洗的训练数据集、用户给定的清洗代价阈值、数据集中每个属性对应的劣质数据检测代价和修复代价.

首先,对根结点的收益与清洗代价比进行计算,选取比值最大的属性作为分裂属性(第 1 行~第 5 行).对该属性中的属性值进行劣质数据检测和修复,计算剩余的清洗代价是否足够对训练数据集中该属性的全部属性值进行检测:如果足够,那么对该属性的所有属性值进行劣质数据检测(第 6 行~第 9 行);如果不足够,那么对该属性部分属性值进行检测(第 10 行~第 14 行).在检测出分裂属性中包含的劣质数据后,对劣质数据进行修复,并计算剩余的清洗代价是否足够对全部劣质数据进行修复:如果足够,那么对该属性中包含的所有劣质数据进行修复(第 15 行~第 17 行);如果不足够,那么对部分劣质数据进行修复(第 18 行~第 21 行).然后,继续为下一个结点寻找分裂属性并清洗(第 22 行).当清洗代价达到用户给定的阈值或决策树中全部结点都被清洗完毕后,返回清洗后的训练数据集(第 23 行).

算法 3. 基于分裂属性收益和清洗代价的分步清洗算法.

输入:训练数据集 $D_{m\times n}$,清洗代价阈值 $MaxCost$,每个属性 $x\in\{1,2,\ldots,n\}$ 的劣质数据检测代价 $Det(a_x)$ 和修复代价 $Rep(a_x)$;

输出:清洗后的训练数据集 $D_{m\times n}$.

1: $remain \leftarrow \mathrm{Max}Cost$

2: $n_i \leftarrow n_0$

3: **While** ((*remain*>0) && (n_i!=*null*))

4: $Benefit(a_x, n_0) \leftarrow NodeCost(n_i) - \sum NodeCost(child(n_i))$

5: $a'_x \leftarrow \arg\max_{a_x} \dfrac{Benefit(a_x, n_i)}{Det(a_x) + Rep(a_x)}$

6: **If** $\left\lfloor \dfrac{remain}{Det(a_x)} \right\rfloor \geqslant m$

7: $Detect(\sum_{u=1}^{m}[a_x][t_u])$

8: $e \leftarrow |Er(a_x)|$

9: $remain \leftarrow remain - Det(a_x) \times m$

10: **Else If** $\left\lfloor \dfrac{remain}{Det(a_x)} \right\rfloor \geqslant 1$

11: $q \leftarrow \left\lfloor \dfrac{remain}{Det(a_x)} \right\rfloor$

12: $Detect(\sum_{u=1}^{q}[a_x][t_u])$

13: $e \leftarrow |Er(a'_x)|$

14: $remain \leftarrow remain - Det(a_x) \times q$

15: **If** $\left((e>1)\ \&\&\ \left(\left\lfloor \dfrac{remain}{Rep(a_x)} \right\rfloor \geqslant e\right)\right)$

16: $Repair(\sum_{u=1}^{e}[a_x][t_u])$

17: $remain \leftarrow remain - Rep(a_x) \times e$

18: **Else If** $\left((e>1)\ \&\&\ \left(\left\lfloor \dfrac{remain}{Rep(a_x)} \right\rfloor \geqslant 1\right)\right)$

19: $r \leftarrow \left\lfloor \dfrac{remain}{Rep(a_x)} \right\rfloor$

20: $Repair(\sum_{u=1}^{r}[a_x][t_u])$

21: $remain \leftarrow remain - Rep(a_x) \times r$

22: $n_i \leftarrow n_{i+1}$

23: **Return** $D_{m\times n}$

算法 3 的时间复杂度取决于决策树中的结点个数 *N*、训练数据集中的属性个数 *n* 和记录个数 *m*.假设劣质数据检测函数 *Detect*(*x*)的时间复杂度为 $O(f(x))$,劣质数据修复函数 *Repair*(*x*)的时间复杂度为 $O(g(x))$,当 $\max(f(x),g(x)) \geqslant n\log n$ 时,算法 3 的时间复杂度是 $O(N\max(f(x),g(x)))$;当 $\max(f(x),g(x)) < n\log n$ 时,算法 3 的时间复杂度是 $O(Nn\log n)$.

融合基于分裂属性收益和清洗代价的分步清洗算法的代价敏感决策树建立方法将收益大且清洗代价小的属性选为分裂属性并优先进行清洗.在决策树建立的每一步中,分裂属性被清洗完毕后再用于分裂.由于决策树中层数低的分裂结点重要性高于层数高的分裂结点,该方法保证了决策树结点分裂的有效性,有效降低了决策树的误分类代价和测试代价总和,使得决策树在后续的分类任务中受训练数据集中的劣质数据影响较小.

3.4 适用情况讨论和时间复杂度比较

当用户设定的数据清洗代价阈值较小时,融合基于分裂属性收益的分步清洗算法的代价敏感决策树建立方法可能会对清洗代价高的属性优先进行清洗,使得能够被清洗到的属性个数较少,对劣质数据的清洗效果不

大.因此,该方法不适用于这种情况.而融合基于分裂属性收益和清洗代价的分步清洗算法的代价敏感决策树建立方法优先对收益大且清洗代价小的属性进行清洗,避免了优先清洗代价高的属性,保证了数据清洗的效果和决策树结点分裂的有效性.因此,该方法适用于这种情况.

当劣质数据比例较大时,融合基于分裂属性收益和清洗代价的一次性清洗算法的代价敏感决策树建立方法一次性清洗到的属性不一定都会在后续的决策树建立过程中被选作分裂属性.该方法可能没有清洗到决策树建立过程中的某些分裂属性,导致数据清洗过程对分类任务的作用不大.因此,该方法不适用于这种情况.而融合基于分裂属性收益和清洗代价的分步清洗算法的代价敏感决策树建立方法在决策树建立的每一步中选择分裂属性,将其清洗完毕后再用于分裂.该方法保证了数据清洗的效果和决策树结点分裂的有效性,有效降低了决策树的误分类代价和测试代价总和,使得决策树在后续的分类任务中受训练数据集中的劣质数据影响较小.因此,该方法适用于这种情况.

此外,为了方便读者选择最适合的数据清洗算法完成劣质数据上的代价敏感决策树建立,表 5 从时间复杂度方面对本文所提出的 3 种清洗算法进行了比较,时间复杂度分析的详细过程参见本文第 3.1 节~第 3.3 节.

Table 5 Comparison of time complexity of three algorithms

表 5 3 种算法的时间复杂度比较

算法名称	时间复杂度
基于分裂属性收益的分步清洗算法	$\max(f(x),g(x))\geqslant n\log n$ 时,$O(N\max(f(x),g(x)))$
	$\max(f(x),g(x))<n\log n$ 时,$O(Nn\log n)$
基于分裂属性收益和清洗代价的一次性清洗算法	$O(n\log n\times\max(f(x),g(x)))$
基于分裂属性收益和清洗代价的分步清洗算法	$\max(f(x),g(x))\geqslant n\log n$ 时,$O(N\max(f(x),g(x)))$
	$\max(f(x),g(x))<n\log n$ 时,$O(Nn\log n)$

4 实验评估

为了验证所提出方法的性能,本文从 3 个方面对所提出的 3 种融合数据清洗算法的代价敏感决策树建立方法进行了测试:(1) 利用本文所提出的方法建立的代价敏感决策树用于分类时产生的误分类代价和测试代价总和;(2) 利用本文所提出的方法建立的代价敏感决策树的分类准确率;(3) 利用本文所提出的方法建立的代价敏感决策树的分类效率.

本实验使用的数据集是 UCI 公测集上的 6 个经典数据集,其基本信息见表 6.在实验过程中,本文选用了劣质数据中的不一致数据进行实验,根据在不同数据集上定义的函数依赖,向训练数据集中人为注入错误.本实验采用 10 折交叉验证方法,随机生成训练集和测试集,并将在测试集上进行分类任务的结果取平均值作为最终的实验结果.

Table 6 Datasets information

表 6 数据集信息

数据集名称	记录个数	属性个数
Hepatitis Prognosis	129	19
Breast Cancer	286	9
BUPA Liver Disorders	345	5
Car	1 728	6
HTRU	17 898	9
Adult	48 842	14

所有实验均在因特尔 i7CPU@2.4GHz,8GB 内存,1TB 硬盘的电脑上完成,所有算法均用 Python 语言编写.本实验分别对以下 4 种代价敏感决策树进行了评估和比较.

(1) T1:随机对训练数据集中的劣质数据进行清洗,再根据分裂属性收益建立的代价敏感决策树;

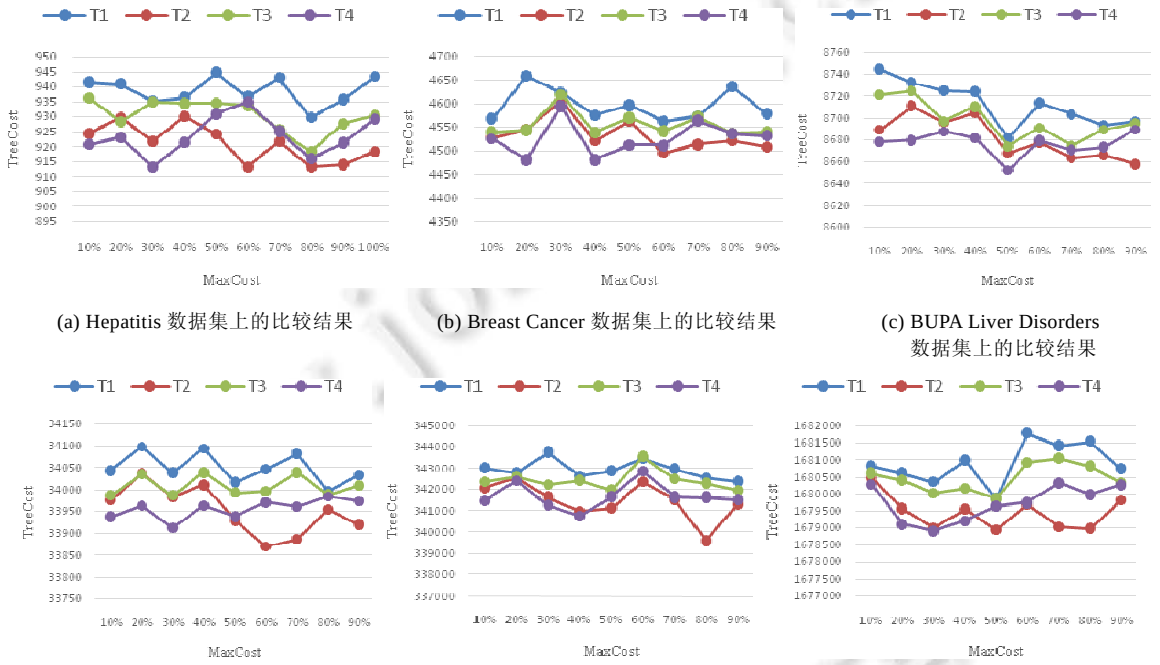
(2) T2:采用融合基于分裂属性收益的分步清洗算法的代价敏感决策树建立方法建立的代价敏感决策树;

(3) T3:采用融合基于分裂属性收益和清洗代价的一次性清洗算法的代价敏感决策树建立方法建立的代价敏感决策树;

(4) T4:采用融合基于分裂属性收益和清洗代价的分步清洗算法的代价敏感决策树建立方法建立的代价敏感决策树.

4.1 分类任务产生的总代价

为了验证所提出的代价敏感决策树建立方法的有效性,本文对上述 4 种代价敏感决策树用于分类任务时产生的误分类代价和测试代价总和(即 *TreeCost*)进行了比较.首先,通过改变用户给定的清洗代价最大值,测试分类任务产生的总代价所发生的变化.实验结果如图 3 所示.



(a) Hepatitis 数据集上的比较结果 (b) Breast Cancer 数据集上的比较结果 (c) BUPA Liver Disorders 数据集上的比较结果

(d) Car 数据集上的比较结果 (e) HTRU 数据集上的比较结果 (f) Adult 数据集上的比较结果

Fig.3 Comparison experiments varying Max*Cost*

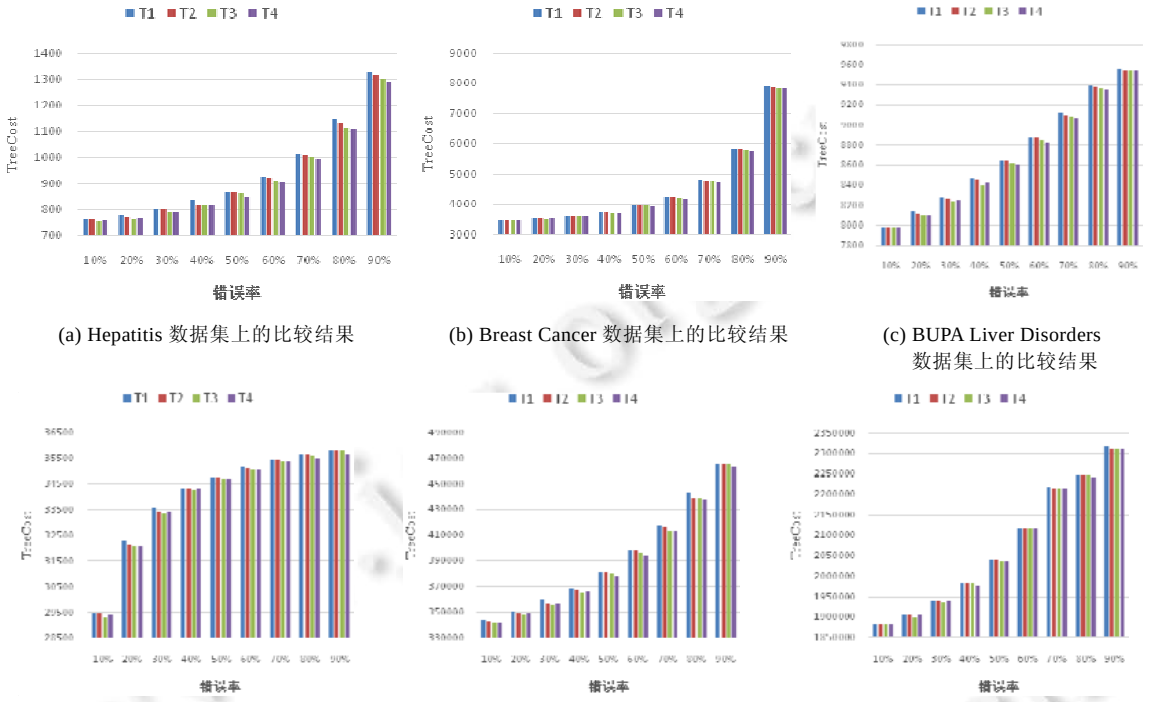
图 3 改变 Max*Cost* 值时的对比实验

从图 3 中能够观察到,决策树 T2 和 T4 的总代价始终小于 T1 和 T3,而 T3 的总代价往往小于 T1.此外,从图 3(a)、图 3(d)~图 3(f)中可以看出:当用户设定的清洗代价最大值 Max*Cost*≤40%时,决策树 T4 的总代价小于 T2;当 Max*Cost*>40%时,T2 的总代价小于 T4.在图 3(b)和图 3(c)中,当 Max*Cost*≤50%时, T4 的总代价小于 T2;当 Max*Cost*>50%时,T2 的总代价小于 T4.这是由于当用户给定的清洗代价较高时,只考虑分裂属性收益而不考虑清洗代价,也能够清洗到较多的分裂属性;而当用户给定的清洗代价较低时,如果不考虑属性对应的清洗代价,可能会优先清洗收益高但清洗代价也高的分裂属性,导致总清洗代价很快被用完,而能够被清洗的分裂属性却很有限,从而影响了代价敏感决策树的总代价.

因此可以得出结论:当用户设定的清洗代价最大值较小时,代价敏感决策树 T4 的总代价最小,即采用融合基于分裂属性收益和清洗代价的分步清洗算法的代价敏感决策树建立方法建立的代价敏感决策树最优;当用户设定的清洗代价最大值较大时,代价敏感决策树 T2 的总代价最小,即采用融合基于分裂属性收益的分步清洗算法的代价敏感决策树建立方法建立的代价敏感决策树最优.

此外,没有进行数据清洗而直接构建的代价敏感决策树的总代价明显高于融合数据清洗方法构建的代价敏感决策树.由此可知,采用融合数据清洗的代价敏感决策树建立方法大幅度节省了误分类代价和测试代价.因此,数据清洗过程对于劣质数据上代价敏感决策树的建立是十分必要的.

然后,本文通过改变训练数据集中的错误率来评估分类任务产生的总代价所发生的变化.实验结果如图 4 所示.



(a) Hepatitis 数据集上的比较结果 (b) Breast Cancer 数据集上的比较结果 (c) BUPA Liver Disorders 数据集上的比较结果

(d) Car 数据集上的比较结果 (e) HTRU 数据集上的比较结果 (f) Adult 数据集上的比较结果

Fig.4 Comparison experiments varying error rate

图 4 改变错误率时的对比实验

从图 4 中能够观察到:决策树 T3 和 T4 的总代价始终小于 T1 和 T2,而 T2 的总代价往往小于 T1.此外,从图 4(a)、图 4(c)~图 4(e)中可以看出:当训练数据集中的错误率小于等于 40%时,决策树 T3 的总代价小于 T4;当错误率大于 40%时,T4 的总代价小于 T3.在图 4(b)和图 4(f)中,当错误率小于等于 30%时,T3 的总代价小于 T4;当错误率值大于 30%时,T4 的总代价小于 T3.这是由于决策树 T3 用到的一次性清洗算法并不能保证清洗到的属性在建立决策树的过程中被选做分裂属性.在训练数据集错误率较高的时候,可能导致许多存在劣质数据的分裂属性并没有被清洗到,从而影响了代价敏感决策树分类时的总代价.

此外,通过图 4 可以看出:当错误率上升时,总代价也不断增大.这是由于训练数据集中的劣质数据增多,使得代价敏感决策树的结构更加复杂,结点数目增加,相应地,测试代价增大,总代价增大.

因此可以得出结论:当错误率较小时,代价敏感决策树 T3 的总代价最小,即采用融合基于分裂属性收益和清洗代价的一次性清洗算法的代价敏感决策树建立方法建立的代价敏感决策树最优;当错误率较大时,代价敏感决策树 T4 的总代价最小,即采用融合基于分裂属性收益和清洗代价的分步清洗算法的代价敏感决策树建立方法建立的代价敏感决策树最优.

综上所述,当错误率较大,而用户设定的清洗代价最大值较小时,代价敏感决策树 T4 的总代价最小,即采用融合基于分裂属性收益和清洗代价的分步清洗算法的代价敏感决策树建立方法建立的代价敏感决策树最优.

4.2 分类任务的准确率

为验证所提出的代价敏感决策树建立方法的有效性,本文对 4 种代价敏感决策树的分类准确率进行了比较.实验结果见表 7,其中,Max*Cost* 为 0%的情况表示没有对劣质数据进行清洗,直接构建代价敏感决策树.

Table 7 Experimental results of classification accuracy

表 7 分类准确率的实验结果

数据集	Max*Cost*	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Hepatitis	T1	58.75%	65.83%	64.17%	66.11%	65.35%	65.07%	65.38%	65.63%	63.68%	64.38%	65.76%
	T2	58.78%	65.21%	65.07%	65.00%	64.79%	64.96%	65.31%	65.76%	67.99%	65.69%	64.78%
	T3	58.38%	65.97%	63.06%	66.67%	64.10%	66.04%	67.36%	65.00%	67.22%	63.82%	65.21%
	T4	58.25%	65.97%	65.07%	65.63%	64.17%	64.31%	66.67%	65.07%	65.28%	64.24%	66.88%
Breast	T1	75.26%	85.79%	86.12%	86.14%	85.91%	85.81%	85.13%	86.04%	85.75%	85.93%	86.57%
	T2	75.95%	86.37%	86.39%	85.09%	86.73%	85.98%	86.49%	86.94%	85.93%	87.27%	84.56%
	T3	75.33%	87.52%	85.93%	85.83%	87.39%	86.94%	85.98%	85.69%	86.41%	86.20%	86.51%
	T4	75.33%	86.00%	85.42%	86.69%	86.63%	86.32%	85.91%	86.96%	87.04%	86.16%	87.12%
BUPA	T1	46.57%	51.46%	51.11%	52.98%	50.22%	50.86%	50.79%	51.75%	52.98%	51.49%	51.05%
	T2	45.43%	50.44%	52.35%	51.43%	52.03%	50.06%	51.65%	50.22%	52.67%	52.44%	50.86%
	T3	46.14%	52.44%	51.59%	51.49%	51.75%	51.75%	51.68%	50.44%	50.86%	52.83%	50.76%
	T4	45.71%	49.71%	49.62%	51.43%	53.40%	50.22%	51.90%	50.22%	50.06%	52.22%	52.25%
Car	T1	61.53%	71.66%	71.27%	71.30%	71.96%	70.69%	71.55%	72.20%	71.48%	71.89%	71.93%
	T2	61.50%	71.70%	71.55%	71.74%	71.91%	71.18%	71.70%	72.05%	71.36%	72.58%	71.29%
	T3	61.14%	71.75%	71.77%	71.69%	71.59%	71.75%	72.03%	71.89%	71.10%	71.66%	71.71%
	T4	61.38%	71.45%	71.32%	71.84%	71.65%	72.11%	71.05%	71.64%	71.76%	72.18%	71.18%
HTRU	T1	93.12%	96.10%	97.18%	97.27%	97.20%	97.55%	97.92%	97.95%	97.95%	97.88%	97.97%
	T2	92.97%	97.77%	98.28%	98.69%	98.55%	98.73%	98.83%	98.96%	99.11%	99.15%	99.13%
	T3	92.84%	96.93%	98.05%	98.15%	98.91%	98.97%	99.11%	98.94%	98.98%	99.01%	99.19%
	T4	92.20%	97.94%	98.42%	98.78%	98.71%	98.90%	99.13%	98.97%	98.97%	99.09%	99.06%
Adult	T1	74.81%	80.07%	80.75%	80.96%	81.12%	81.86%	81.66%	81.23%	81.79%	81.86%	81.94%
	T2	74.73%	79.79%	81.71%	81.94%	81.97%	82.15%	81.89%	82.05%	82.92%	82.73%	82.97%
	T3	74.59%	80.09%	80.93%	80.99%	81.74%	82.58%	82.04%	82.21%	82.88%	82.97%	82.87%
	T4	74.47%	80.68%	81.76%	81.88%	81.89%	82.62%	82.35%	81.99%	82.96%	82.89%	82.88%
	平均值	68.13%	75.36%	75.37%	75.82%	75.82%	75.73%	75.98%	75.83%	76.31%	76.11%	76.02%

从表 7 可以看出,代价敏感决策树 T1~ T4 在分类准确率上相差极小.因此,在选择代价敏感决策树建立方法时,可以不必将分类准确率作为考虑因素.此外,没有进行数据清洗而直接构建的代价敏感决策树(即 Max*Cost* 为 0%)的分类准确率比融合数据清洗方法构建的代价敏感决策树至少低 7.23%(根据表 7 中的平均值计算).由此可知,采用融合数据清洗的代价敏感决策树建立方法能够有效提高决策树的分类质量.因此,数据清洗过程对于劣质数据上代价敏感决策树的建立是十分必要的.

4.3 分类任务的效率

为了验证所提出的代价敏感决策树建立方法的有效性,本文对 4 种代价敏感决策树的分类效率进行了比较.实验结果见表 8.从表 8 可以看出,代价敏感决策树 T1~T4 在分类效率上相差极小.因此,在选择代价敏感决策树建立方法时,可以不必将分类效率作为考虑因素.

Table 8 Experimental results of classification efficiency (ms)

表 8 分类效率的实验结果 (毫秒)

数据集	Max*Cost*	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Hepatitis	T1	0.513 1	0.479 1	0.639 8	0.446 1	0.496 5	0.557 0	0.561 7	0.563 7	0.495 5	0.562 5
	T2	0.484 2	0.461 8	0.545 8	0.519 2	0.518 6	0.533 6	0.640 5	0.496 7	0.523 4	0.658 1
	T3	0.538 9	0.606 0	0.529 7	0.513 0	0.579 1	0.607 5	0.529 2	0.574 9	0.546 1	0.547 5
	T4	0.580 2	0.561 1	0.546 6	0.451 0	0.558 9	0.506 7	0.500 6	0.478 3	0.523 6	0.469 8
Breast	T1	1.819 1	1.787 2	1.765 7	1.878 0	1.724 1	1.777 1	1.719 5	1.800 6	1.790 4	1.774 5
	T2	1.784 1	1.849 3	1.805 1	1.816 4	1.771 9	1.823 8	1.801 7	1.850 7	1.734 1	1.751 6
	T3	1.866 8	1.799 0	1.719 9	1.770 1	1.782 3	1.768 4	1.815 1	1.833 5	1.839 5	1.924 3
	T4	1.794 9	1.745 4	1.799 3	1.782 9	1.804 4	1.773 3	1.861 9	1.756 1	1.738 3	1.732 4
BUPA	T1	1.080 0	1.042 6	1.051 9	1.021 1	1.041 7	1.020 6	1.053 9	1.015 7	1.119 6	1.041 5
	T2	1.042 4	1.086 0	1.098 7	1.054 2	1.109 3	1.099 8	1.001 8	1.118 7	1.043 8	1.101 7
	T3	1.037 2	0.999 2	0.969 6	1.040 6	1.097 6	1.059 0	1.047 4	1.018 6	1.024 7	1.059 4
	T4	1.068 6	1.047 8	1.127 9	1.097 3	1.069 1	1.063 9	1.057 9	1.057 5	1.084 0	1.058 9

Table 8 Experimental results of classification efficiency (Continued) (ms)

表 8 分类效率的实验结果(续) (毫秒)

数据集	Max*Cost*	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Car	T1	4.556 8	4.618 6	4.750 8	4.973 7	4.448 4	4.660 2	4.677 5	4.634 9	4.845 4	4.716 3
	T2	4.474 3	4.444 9	4.998 1	4.845 7	4.472 9	4.610 5	4.576 1	4.507 1	4.749 2	4.345 4
	T3	4.786 0	4.757 6	4.304 2	4.465 1	4.479 9	4.880 2	4.948 4	4.722 8	4.740 7	4.438 9
	T4	4.538 4	4.763 5	4.468 4	4.669 4	4.610 5	4.295 1	4.479 4	4.865 8	4.376 9	4.257 3
HTRU	T1	140.7 5	126.38	129.03	129.77	134.31	127.83	129.68	129.16	128.01	128.66
	T2	140.87	130.17	132.00	130.91	144.08	130.45	130.39	127.87	127.40	127.43
	T3	131.98	132.39	132.39	128.13	131.07	127.98	130.86	133.45	128.02	128.56
	T4	129.6	133.63	133.63	135.37	126.67	129.10	129.72	129.84	131.24	129.61
Adult	T1	120.66	120.29	121.91	123.45	120.18	121.61	121.58	123.12	127.49	121.88
	T2	123.12	128.33	129.65	123.34	124.40	125.11	120.26	125.96	127.09	123.17
	T3	126.24	122.90	119.45	127.35	122,70	121.23	121.13	122.73	123.31	122.79
	T4	124.29	125.53	124.83	121.87	123.66	123.10	124.13	123.11	124.91	125.17

综合上述实验,本文对所提出的 3 种代价敏感决策树建立方法进行了充分地比较.根据实验结果,3 种方法的具体适用情况见表 9.

Table 9 Applicable conditions of three methods

表 9 3 种方法的适用情况

方法名称	适用情况
融合基于分裂属性收益的分步清洗算法的代价敏感决策树建立方法	用户设定的清洗代价最大值较大
融合基于分裂属性收益和清洗代价的一次性清洗算法的代价敏感决策树建立方法	训练数据集的错误率较小
融合基于分裂属性收益和清洗代价的分步清洗算法的代价敏感决策树建立方法	用户设定的清洗代价最大值较小或训练数据集的错误率较大

5 结 论

本文研究了劣质数据上代价敏感决策树的建立问题.首先对决策树、误分类代价、测试代价、检测代价和修复代价等相关概念进行了介绍,并对本文的研究问题进行了定义;然后,本文提出了 3 种融合数据清洗算法的代价敏感决策树建立方法;最后,本文通过大量实验验证了 3 种方法的总代价、分类准确率和分类效率,并给出了不同情况下最优的代价敏感决策树建立方法.

References:

[1] Chen YL, Wu CC, Tang K. Time-constrained cost-sensitive decision tree induction. Information Sciences, 2016,354:140−152.

[2] Safavian SR, Landgrebe D. A survey of decision tree classifier methodology. IEEE Trans. on Systems, Man, and Cybernetics, 1991, 21(3):660−674.

[3] Friedman N, Geiger D, Goldszmidt M. Bayesian network classifiers. Machine Learning, 1997,29(2-3):131−163.

[4] Wan EA. Neural network classification: A Bayesian interpretation. IEEE Trans. on Neural Networks, 1990,1(4):303−305.

[5] Aamodt A, Plaza E. Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications, 1994,7(1):39−59.

[6] Murthy SK. Automatic construction of decision trees from data: A multi-disciplinary survey. Data Mining and Knowledge Discovery, 1998,2(4):345−389.

[7] Quinlan JR. Simplifying decision trees. Int'l Journal of Man-Machine Studies, 1987,27(3):221−234.

[8] Loh WY, Shih YS. Split selection methods for classification trees. Statistica Sinica, 1997,(7):815−840.

[9] Chandra B, Varghese PP. Moving towards efficient decision tree construction. Information Sciences, 2009,179(8):1059−1069.

[10] Ting KM. An instance-weighting method to induce cost-sensitive trees. IEEE Trans. on Knowledge and Data Engineering, 2002, 14(3):659−665.

[11] Sun Y, Kamel MS, Wong AK, *et al*. Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition, 2007, 40(12):3358−3378.

[12] Sahin Y, Bulkan S, Duman E. A cost-sensitive decision tree approach for fraud detection. Expert Systems with Applications, 2013, 40(15):5916−5923.

[13] Lomax S, Vadera S. A survey of cost-sensitive decision tree induction algorithms. ACM Computing Surveys, 2013,45(2):1−35.
[14] Kim W, Choi BJ, Hong EK, *et al*. A taxonomy of dirty data. Data Mining and Knowledge Discovery, 2003,7(1):81−99.
[15] Rahm E, Do HH. Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin, 2000,23(4):3−13.
[16] Tan M. Cost-sensitive learning of classification knowledge and its applications in robotics. Machine Learning, 1993,13(1):7−33.
[17] Ferri C, Flach P, Hernández-Orallo J. Learning decision trees using the area under the ROC curve. In: Proc. of the 19th Int'l Conf. on Machine Learning., Vol.2. ACM Press, 2002. 139−146.
[18] Turney PD. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. Journal of Artificial Intelligence Research, 1994,2:369−409.
[19] Domingos P. Metacost: A general method for making classifiers cost-sensitive. In: Proc. of the 5th Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 1999. 155−164.
[20] Mingers J. An empirical comparison of selection measures for decision-tree induction. Machine Learning, 1989,3(4):319−342.
[21] Quinlan JR. Induction of decision trees. Machine Learning, 1986,1(1):81−106.
[22] Núñez M. The use of background knowledge in decision tree induction. Machine Learning, 1991,6(3):231−250.
[23] Ting KM, Zheng Z. Boosting cost-sensitive trees. In: Proc. of the 1st Int'l Conf. on Discovery Science. Springer-Verlag, 1998. 244−255.
[24] Ling CX, Sheng VS, Yang Q. Test strategies for cost-sensitive decision trees. IEEE Trans. on Knowledge and Data Engineering, 2006,18(8):1055−1067.
[25] Esmeir S, Markovitch S. Anytime induction of low-cost, low-error classifiers: A sampling-based approach. Journal of Artificial Intelligence Research, 2008,33:1−31.
[26] Esmeir S, Markovitch S. Anytime learning of anycost classifiers. Machine Learning, 2011,82(3):445−473.
[27] Qi ZX. Research on key technologies of on-demand data cleaning for dirty data [MS. Thesis]. Harbin: Harbin Institute of Technology, 2018. (in Chinese with English abstract)
[28] Chu X, Ilyas IF, Papotti P. Holistic data cleaning: Putting violations into context. In: Proc. of the 29th Int'l Conf. on Data Engineering. IEEE, 2013. 458−469.

附中文参考文献:

[27] 齐志鑫.劣质数据按需清洗的关键技术研究[硕士学位论文].哈尔滨:哈尔滨工业大学,2018.

齐志鑫(1994－),女,黑龙江哈尔滨人,硕士,主要研究领域为数据质量,劣质数据分析,知识图谱,轨迹数据计算.

王宏志(1978－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据质量管理,海量数据管理,知识图谱,XML 数据管理,工业大数据.

周雄(1996－),男,硕士生,主要研究领域为机器学习,图像处理.

李建中(1950－),男,博士,教授,博士生导师,CCF 会士,主要研究领域为数据库系统实现技术,数据仓库,半结构化数据,传感器网络,压缩数据库技术,Web 数据集成,数据挖掘,计算生物学.

高宏(1966－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为复杂结构数据管理,无线传感器网络.