一种频繁模式决策树处理可变数据流

韩 萌[1),2)] 王志海[2)] 丁 剑[1)]

[1)](北方民族大学计算机科学与工程学院 银川 750021)

[2)](北京交通大学计算机与信息工程学院 北京 100044)

摘 要 数据流中可能包含大量的无用信息或者噪声,频繁模式挖掘可以去除这些无用信息,且频繁模式比单个属性包含了更多的信息.因此,挖掘频繁的、有区分力的模式,可以用于有效的分类.该文提出一个两步骤算法 PatHT (Pattern-based Hoeffding Tree)生成决策树用于可变数据流分类.第一步,设计增量更新算法 CCFPM(Constraints-based and Closed Frequent Pattern Mining),用于生成闭合约束频繁模式集合 CFPSet(Closed Frequent Pattern Set).CCFPM 中采用滑动窗口模型和时间衰减模型处理实例,设计一种均值衰减因子设置方法得到高完整性和准确性的模式集合.第二步,增量更新方法 HTreeGrow(Hoeffding Tree Growing)生成基于 CFPSet 的概念漂移决策树.该方法使用概念漂移检测器监督概念改变,自动调整分类模型.针对高密度和低密度的数据流,设计了不同使用模式集合的方法.在真实和模拟数据流上的实验分析表明,与其他同类算法相比,提出的方法对稳态数据流处理时可以明显提高正确率或可以明显降低训练时间,在处理不同概念漂移特性的可变数据流时也具有很好的分类效果.

关键词 分类;可变数据流;决策树;频繁模式挖掘;Hoeffding 树;数据挖掘

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2016.01541

Efficient Decision Tree for Evolving Data Streams Based on Frequent Patterns

HAN Meng[1),2)] WANG Zhi-Hai[2)] DING Jian[1)]

[1)](*School of Computer Science and Engineering, Beifang University of Nationalities, Yinchuan* 750021)

[2)](*School of Computer and Information Technology, Beijing Jiaotong University, Beijing* 100044)

Abstract Data stream may contain a large number of useless information or noises. Frequent pattern mining can drop such useless information and discover patterns. Frequent patterns may contain more information than single attribute. Therefore, frequent and discriminative pattern can be used to train classification model effectively. In this paper, we propose a two-steps method PatHT (Pattern-based Hoeffding Tree) to generate decision tree for evolving data stream classification. First step, an incremental algorithm CCFPM (Constraints-based and Closed Frequent Pattern Mining) is proposed to discover frequent pattern set CFPSet (Closed Frequent Pattern Set). These patterns are closed, that is, they have total information of complete patterns and less numbers than them. These patterns must contain class attribute for classification in next step. The sliding window model and time decay model is used in CCFPM to deal with concept drift problem. And a novel average decay factor is designed to get pattern result set with high recall and high precision. Second step, an incremental algorithm HTreeGrow (Hoeffding Tree Growing) is proposed to train concept drift decision tree based on CFPSet. Concept drift detector is used to discover concept change; therefore classification model is adjusted automatically. For high-density and low-density data streams, we design different ways to use pattern sets. The performance of

收稿日期:2015-04-27;在线出版日期:2015-11-06.本课题得到国家自然科学基金(61563001)和国家民委科研基金(14BFZ008)资助.
韩 萌,女,1982 年生,博士研究生,副教授,主要研究方向为数据挖掘与机器学习.E-mail: compute2006_2@126.com. **王志海**,男,1963 年生,博士,教授,博士生导师,主要研究领域为数据挖掘与机器学习. **丁 剑**,男,1977 年生,硕士,副教授,主要研究方向为计算机应用与数据挖掘.

proposed method is evaluated via experiments. Using real life data streams shows that the proposed method can reduce the training time or improve the classification accuracy. Processing synthetic data streams also shows that the proposed method is superior to other analogous algorithms.

Keywords classification; evolving data stream; decision tree; frequent pattern mining; Hoeffding tree; data mining

1 引 言

随着数据流挖掘应用日趋广泛,数据流分类问题已成为一项重要且充满挑战的工作.数据流与传统的静态数据或数据库相比具有非常不同的特性,如动态、无限、高维、有序、非重复性、高速和随时间变化[1].在真实的数据流环境中,大部分数据流是可变的,即具有概念漂移[2]特征,称为可变数据流或概念漂移数据流.根据可变数据流的特点,一个有效的分类器必须能跟踪并快速适应其概念的变化.已有的处理可变数据流的分类模型包括决策树、神经网络、规则学习等.

尽管存在多种方法,但决策树模型是在线数据流分类的最先进方法.原因很大一部分来自于它们有能力快速地处理大量的数据,这超出任何其他流或批处理学习算法[3].最有影响力的算法之一是快速决策树 VFDT(Very Fast Decision Tree)[4].它是一种基于 Hoeffding 不等式针对数据流挖掘环境建立分类决策树的方法.它通过不断地将叶节点替换为分支节点而生成,即在每个决策节点保留一个重要的统计量,当该节点的统计量达到一定阈值,则进行分裂测试.其最主要的创新是利用 Hoeffding 不等式确定叶节点变为分支节点所需要的样本数目.该算法仅需扫描一次数据,具有较高的时空效率,且分类器性能近似于传统算法生成的分类器.VFDT 的不足在于不能很好的处理概念漂移问题.算法 CVFDT(Concept-adapting Very Fast Decision Tree)[5]对 VFDT 进行了扩展以快速解决概念漂移数据流的分类.其核心思想是当新的子树分类更准确时,用新的子树替换历史子树.它维持一个滑动训练窗口,并通过在样本流入和流出窗口时更新已生成的决策树使其与训练窗口内样本保持一致.VFDTc[6]算法扩展了增量树学习方法,从两个方面进行提高.一是设计二元搜索树用于处理数值属性;二是改进了使用,在树的叶子节点上使用朴素贝叶斯来训练实例.这种方式可以明显提高树的预测正确率.不足之处是在二元树上的统计可能相对比较大,尤其是当实例的数值属性具有许多独特值时.HOT(Hoeffding Option Trees)[7]在常规 Hoeffding 树的基础上增加了附加可选节点,允许进行多个测试,得到多个 Hoeffding 子树作为独立路径.它们由独立结构组成,可以有效的表示多棵树.一些特殊的实例可以沿树的多个路径向下,有利于以不同的方式进行不同的选择.HAT(Hoeffding Adaptive Tree)[8]也采用了 Hoeffding 树,它主要的优点在于不需要考虑数据流变化的速度和频度.它使用概念漂移检测器 ADWIN[9]来监控树分支的性能.如果新的树枝可以得到更高的正确率,则使用新树枝代替导致正确率降低的树枝.AdoHOT(Adaptive Hoeffding Option Tree)[10]是在 HOT 的基础上做了改进:每个叶子存储当前误差的估计值.在投票过程中的每个节点的权重正比于误差的倒数的平方.算法 ASHT(Adaptive-Size Hoeffding Tree)[10]是 Hoeffding 树的衍生,包括两处不同:(1)它设定了分裂节点的最大数目;(2)当一个节点分裂后,如果 ASHT 的节点数目高于最大限定值,则删除一些节点来降低树的大小.

决策树模型可以得到好的分类正确率,且可以用于实现简单和有效的集成分类模型.如 EM[1]是处理可变数据流的自适应集成分类方法,它可以用于检测新类.它采用传统的集成分类方式处理数据流,并且不断地自动更新以适应概念变化.但是对于新类,使用聚类方式检测.OBag 和 ORF[3]是基于决策树的在线数据流回归的集成方法.OBag 是一种在线基于 Hoeffding 模型树的打包集成方法,ORF 是一种随机森林方法,采用随机模型树学习方法作为基本的构建模块.

已有的决策树分类方法处理的数据流中包含无限数据,这些数据可能包含大量的无用信息甚至是噪声,而模式发现可以去除数据中的无用信息且不受噪声的影响.因此,挖掘有趣的、频繁的和有区分

力的模式，可以用于有效的分类. 基于模式的分类具有更高的准确性，并且可以很好地解决缺失值的问题. 为此，本文研究一种新的基于频繁模式的决策树训练方法，用于处理可变数据流. 主要的工作包括：(1) 已有的数据流分类流程包括 3 步骤：输入-训练-模型[10]，为了提高模型的创建效率和分类正确率，提出了 4 步骤处理流程：输入-模式-训练-模型；(2) 设计一种均值衰减因子，用于时间衰减模型(Time Decay Model，TDM)，目的是发现更加完整和准确的频繁模式集合；(3) 在 TDM 基础上，设计一种增量更新数据流闭合频繁模式挖掘方法，约定约束使得发现的模式都具有类属性；(4) 设计一种基于模式的自适应调整概念漂移决策树，使用概念漂移检测器检测概念变化从而自动调整分类模型.

2　预备知识

本节介绍数据流中频繁模式挖掘和时间衰减模型相关概念，决策树分类方法和概念漂移处理方法等相关知识.

2.1　频繁模式挖掘

数据流 $DS=\{T_1,T_2,\cdots,T_m,\cdots\}$ 是一个有时间顺序的、连续的、无限的事务(Transaction)/实例(Example、Instance)序列. 其中 $T_m(m=1,2,\cdots)$ 是第 m 个产生的实例，它是一个元组，可以表示为形式 $\langle X,C_{id}\rangle$. 其中 X 是一些条件属性值的集合，C_{id} 是该实例所属类标号.

由于数据流是无限的和不断流动的，项集 P 的支持数定义为已出现的 M 个实例中包含项集 P 的实例数据个数，记为 $freq(P,M)$[11](简记为 $freq(P)$). 若项集 P 为频繁模式则应满足定义 1.

定义 1(频繁模式).　令 M 为数据流中已有实例的个数，$\theta(\theta\in(0,1])$ 为最小支持度阈值. 如果项集 P 满足 $freq(P)\geq\theta\times M$，则 P 为频繁模式.

频繁模式挖掘存在的最大问题是生成结果集中包含数量巨大的模式. 为了减少模式的数量，通常挖掘压缩模式. 闭合模式是一种常用的无损压缩模式，它包含完整结果集的所有信息且数量减少了很多. 闭合模式的概念为定义 2 所示.

定义 2(闭合频繁模式).　对于频繁项集 P，若不存在频繁项集 Q 满足下列条件：(1) $P\subset Q$；(2) $freq(P)=freq(Q)$，则称 P 为闭合频繁模式.

示例 1.　包含 8 个实例的数据集如表 1 所示. 每个实例长度为 5，分别包含 4 个条件属性和 1 个分类属性. 条件属性 A_1,A_2,A_3,A_4 的取值个数分别为 3、3、2、2. 类属性为 Class，包括两个取值{yes，no}. 设定变量 A_i 表示条件属性，i 表示第 i 个属性. A_{ij} 为属性 A_i 的第 j 个取值. C_k 为类的第 k 个取值. V_{ijk} 表示在 C_k 条件下，A_{ij} 的个数.

表 1　数据流

实例	A_1	A_2	A_3	A_4	C
T_1	a1	b1	c1	d1	yes
T_2	a1	b1	c1	d2	yes
T_3	a1	b2	c1	d1	yes
T_4	a1	b2	c2	d2	no
T_5	a1	b3	c1	d1	yes
T_6	a1	b3	c1	d2	no
T_7	a2	b2	c2	d2	no
T_8	a3	b2	c2	d2	no

若设置最小支持数阈值 θ 为 0.3，则项集 $P_1=\langle$a1，c1，yes$\rangle$ 为闭合频繁模式. 这是由于它出现在实例 T_1,T_2,T_3,T_5 中，可以得出其支持数为 4，满足 $freq(P_1)>0.3\times 8=2.4$，且不存在与 P_1 支持数相同的父频繁项集. 而项集 $P_2=\langle$a1，yes$\rangle$ 不是闭合频繁模式，这是由于存在父项集 P_1，且满足条件 $freq(P_1)=freq(P_2)$.

由于数据流的连续无限性，其中包含的知识会随着时间推移而发生改变. 通常情况下，最近产生事务的价值比历史事务要高的多. 因此，需要增加最近事务的权重. 时间衰减模型是一种随着时间的推移而逐步衰减历史模式支持数权重的方法[11-12]. 设模式支持数在单位时间内的衰减比例为衰减因子 $f(f\in(0,1))$，记事务 T_n 到达时模式 P 的衰减支持数为 $freq_d(P,T_n)$. 则第 m 个事务 T_m 到达时，模式 P 的衰减支持数满足式(1)和(2). 即随着新事务的到达，每次模式的支持数都进行衰减. 其中如果新事务 T_m 中包含模式 P，则 r 的值为 1；否则为 0.

$$freq_d(P,T_m)=\begin{cases}r, & m=1\\ freq_d(P,T_{m-1})\times f+r, & m\geq 2\end{cases}\tag{1}$$

$$r=\begin{cases}1, & P\subseteq T_m\\ 0, & \text{其他}\end{cases}\tag{2}$$

2.2　决策树分类方法

Bifet 等人[10] 提出数据流分类循环过程包括 3 个步骤，如图 1 所示. 循环过程包括：(1) 传递数据流中下一个可用的数据至算法中，这一步需要实时处理且每个数据被处理一次；(2) 算法尽可能快的处理数据，更新数据结构；(3) 算法准备好接受下一

个数据,且随时可以对未知数据的类值进行预测.如此不断的循环反复对数据流进行处理.

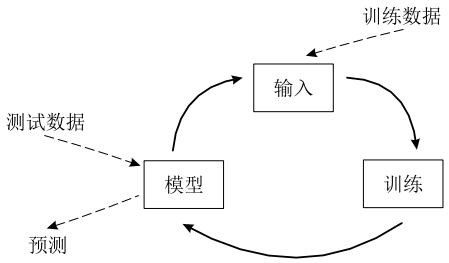


图 1 数据流分类循环示意[7]

决策树是数据流分类中常用的模型之一.常用的属性分裂准则包括信息增益(Information Gain)、增益率(Gain Rate)、基尼指数(Gini Index).本文中将重点考虑信息增益,其计算方法如式(3)所示.其中 $Gain()$ 表示信息增益值,$H()$ 表示熵,$P()$ 为概率值.

$$Gain(A_i)=H(C)-H(C|A_i),$$
$$H(C)=-\sum_k P(C_k)\log(P(C_k)),$$
$$H(C|A_i)=-\sum_j P(A_{ij})\sum_k P(C_k|A_{ij})\log(P(C_k|A_{ij})) \tag{3}$$

Domingos 和 Hulten[4] 使用了 Hoeffding 树(Hoeffding Trees)用于数据流分类,这是一种基于 Hoeffding 不等式建立分类决策树的方法.Hoffding 不等式的定义如下:假定一个实数型的随机变量 r 的取值范围是 R(例如针对信息增益范围是 $\log c$,c 为类别个数).假定 r 的 n 个独立样本点,计算它的样本均值为 $\bar{r}$.则 r 的真实平均值至少是 $\bar{r}-\varepsilon$ 的概率为 $1-\delta$,其中:

$$\varepsilon=\sqrt{\frac{R^2\ln(1/\delta)}{2n}} \tag{4}$$

公式中 n 的最小取值就是当前节点进行分裂时所需的最小实例数.VFDT 的算法描述如算法 1 所示,其中 $G()$ 为信息增益.

算法 1. VFDT.

输入:S data stream

δ desired probability of choosing the correct attribute at any given node

输出:HT decision tree

1. Let HT be a tree with a single leaf (root)
2. Compute counts V_{ijk} at root
3. For each example T_{new} in S do
4. Sort T_{new} to leaf l of HT
5. Updata V_{ijk} at leaf l
6. Compute information gain G for each attribute from counts V_{ijk}
7. If G(Best Attr. BA_1)$-G$(2nd best Attr. BA_2)$>\varepsilon$ Then

Spit leaf l on best attribute

For each branch do initialize new leaf counts at l.

2.3 概念漂移处理方法

可变数据流的特点是观测到的数据潜在分布会随着时间改变.处理此类数据流,分类器应能发现概念改变的信息,并快速调整分类模型以适应概念变化[13].研究中出现了很多方式处理数据流分类中的概念漂移问题,包括使用滑动窗口和实例权重[14]、检测概念改变点[15]、监控两个不同时间窗口内分布[16]等.如 Gama 等人[17]提出基于错误率的概念检测分类方法,Baena 等人[18]提出基于分类错误距离的概念检测方法,Gama 等人[13]提出一种两层学习系统来解决周期性概念问题等等.

最近常用的监控分布方式是 ADWIN 方法.它使用 Hoeffding 边界来保证窗口的最大宽度,且在窗口内没有概念改变.ADWIN 是一种概念漂移检测器和评估器,它是一种捕获流平均数的很好方法.它保留一个可变长度窗口大小的最新实例,窗口足够大使得在这个窗口内不存在概念漂移.ADWIN 主要工作思路是,当最新窗口 W 中两个足够大的子窗口 W_1 和 W_2 可以展示足够明显的平均数,并且可以推断出相应的预测值是不同的,则窗口中较旧的部分可以删除.其中足够大和足够明显可以用 Hoeffding 边界定义,即两个子窗口的平均大于变量 ε_{cut},如式(5)所示.其中 $|W|$ 为最新窗口 W 的大小,$|W_1|$ 和 $|W_2|$ 是两个子窗口 W_1 和 W_2 的大小,且满足 $|W|=|W_1|+|W_2|$.

$$m=\frac{2}{1/|W_0|+1/|W_1|},$$
$$\varepsilon_{cut}=\sqrt{\frac{1}{2m}\times\ln\frac{4|W|}{\delta}} \tag{5}$$

3 基于模式的分类决策树方法

本节介绍基于模式的分类决策树方法建立过程,该过程分为两步.首先挖掘基于约束的闭合频繁模式,然后基于模式建立分类决策树.

首先,本文提出一种基于模式的数据流分类循环过程,包括 4 个步骤如图 2 所示.该过程和图 1 中常规循环过程相比,在进行分类器训练之前,增加了一步生成模式.具体而言循环过程包括:

(1)传递数据流中下一个可用的数据 T_{new} 至算法中；

(2)对 T_{new} 进行频繁模式挖掘，更新模式相关数据结构；

(3)算法尽可能快的处理模式集合中的每个数据，更新数据结构；

(4)算法准备好接受下一个数据，且随时可以对未知数据的类值进行预测.

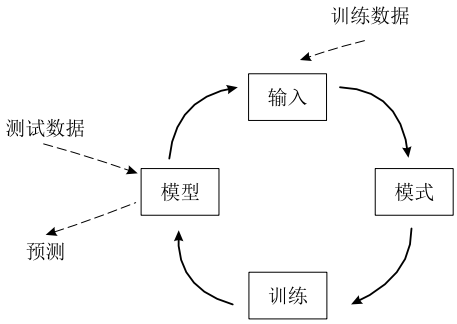


图 2　基于模式的数据流分类循环示意

如此不断的循环反复对数据流进行处理，这样的目的有两个：一是去除数据中的无用信息或者噪声；二是得到的频繁模式具有的信息多于单个的属性，有利于提高决策树创建的效率.

本文设计的算法采用滑动窗口和时间衰减模型对数据流进行增量更新的模式挖掘. 时间衰减模型在模式发现过程中，可以提高新事物的权重、降低历史事物权重，目的是与滑动窗口配合解决模式挖掘过程中的概念漂移问题，且分类模型训练过程中使用频度较高的 top-k 模式，相比原始数据而言包含了更多的信息，会生成更加合理的决策树. 因此，采用图 2 中的数据流分类循环是合理的.

3.1　基于约束的闭合频繁模式挖掘

时间衰减模型是数据流频繁模式挖掘中处理概念漂移问题的有效方法，其关键技术是设置衰减因子的方式. 已有的方式通常采用假定 100% 的查全率(Recall)和 100% 的查准率(Precision)来估计[11-12]，即设定 $Recall$ 为 100%时，f 应满足式(6)，称为下界值，其中 θ 为给定最小支持度阈值，ξ 为最大允许误差阈值. 设定 $Precision=100\%$ 时，f 满足式(7)，称为上界值. 现有的 f 估计方式是采用满足式(6)和(7)的上下边界值之一. 这种方式的最大不足是仅考虑了查全率或查准率，而忽略了对应的查准率或查全率. 由于 $Recall$ 和 $Precision$ 不可能同时为 100%，所以选择 f 是应考虑对二者的平衡. 为此本文提出了均值衰减因子设置方式，即设置 f 为上下边界的平均值，标记为 $f_{average}$.

$$f\geq \sqrt[(2N-\theta N-1)]{[(\theta-\xi)/\theta]^2},\ Recall=100\% \tag{6}$$

$$f<\frac{(\theta-\xi)N-1}{(\theta-\xi)N},\ Precision=100\% \tag{7}$$

例如，假设 $N=10$ K，设定 θ，ε 如表 2 所示. 其中 f_{recall} 为假定 $Recall=100\%$ 时得到的下界值. $f_{precision}$ 为假定 $Precision=100\%$ 时得到的上界值. 在得到 f_{recall} 和 $f_{precision}$ 后，如何选择 f 的值？可以有 3 个策略，如式(8)和(9)所示. 以 $\theta=0.025$，$\varepsilon=0.05\times\theta$ 为例，可以选定

$$f=f_1=f_{recall}=0.999995,$$
$$f=f_2=f_{precision}=0.995789 \text{ 或}$$
$$f=f_3=f_{average}=0.997892.$$

表 2　设置最佳的衰减因子

θ	ε	f_{recall}	$f_{precision}$	$f_{average}$
0.05	$0.05\times\theta$	0.999995	0.997895	0.998945
0.05	$0.1\times\theta$	0.999989	0.997778	0.998884
0.05	$0.5\times\theta$	0.999929	0.996	0.997965
0.025	$0.05\times\theta$	0.999995	0.995789	0.997892
0.025	$0.1\times\theta$	0.999989	0.995556	0.997773
0.025	$0.5\times\theta$	0.99993	0.992	0.995965

$$f_1=f_{recall},\quad f_2=f_{precision} \tag{8}$$

$$\begin{aligned} f_3=f\_average&=(f\_recall+f\_precision)/2\\ &=\frac{\sqrt[(2N-\theta N-1)]{[(\theta-\xi)/\theta]^2}+\frac{(\theta-\xi)N-1}{(\theta-\xi)N}}{2}\\ &=\frac{(\theta-\xi)N(\sqrt[(2N-\theta N-1)]{[(\theta-\xi)/\theta]^2}+1)-1}{2N(\theta-\xi)}\\ \Rightarrow f_3&=\frac{\sqrt[(2N-\theta N-1)]{[(\theta-\xi)/\theta]^2}+1}{2} \end{aligned} \tag{9}$$

通过韩萌等人[19]实验验证，设置均值衰减因子与同类频繁模式挖掘方式 MSW[11]、SWP[12] 和 CloStream*[20] 相比，可以得到具有高完整性和准确性的频繁模式结果集合.

为了下一步的分类使用，文中发现的频繁模式是满足类约束的，如约束 1 所示.

约束 1(类约束).　发现的模式满足：(1)形式 $\langle X,C\rangle$，必须至少包含一个属性值和一个类值；(2)是闭合的.

本文设计算法 CCFPM(Constraints-based and Closed Frequent Pattern Mining over data stream)增量更新的发现满足类约束的频繁模式，使用滑动窗口模型(Sliding Window Model，SWM)和 TDM 处理概念漂移问题，具体如算法 4. CCFPM 算法主要包括 2 个方法：一是处理新实例 T_{new} 的方法

CCFPMADD,它用于发现新实例带来的模式集合的变化;二是处理历史实例 T_{old} 的方法 CCFPMREMOVE,它用于处理移出窗口的历史实例信息.由于 CCFPMREMOVE 与 CCFPMADD 结构相似,过程相逆,因此本文重点介绍 CCFPMADD 的实现过程.

CCFPM 算法使用了 3 个数据结构,包括 *ClosedTable*[20]、*CidList*[20]和 *NewTransactionTable*.其中 *ClosedTable* 用于存储闭合模式相关的信息,包括 3 个字段:*Pid*,*CP* 和 *SCP*. *Pid* 用于唯一的标识每一个闭合项集 *CP*,*SCP* 是闭合项集 *CP* 对应的支持数. *PidList* 用于维护数据流中出现的每个项 *item* 和其对应的 *Pid* 集合. *NewTransactionTable* 包含与新实例 T_{new} 相关的信息,包括 2 个字段:*TempItem* 和 *Pid*.其中 *TempItem* 存储满足条件{$T_{new} \cap CP$, $CP \in ClosdeTable$}的项集信息. CCFPMADD 算法处理新实例的工作过程主要包括 3 步:

(1)参照 *ClosedTable* 查找与 T_{new} 相关的频繁项集 *interS*;

(2)若 $itemset \in interS$ 且为新的频繁项集,则加入 *ClosedTable*,同步更新 *PidList*;

(3)若 $itemset \in ClosedTable$,则更新已有模式;

① 如果依然是闭合模式,则更新其支持数;

② 新数据的到来使之成为非闭合模式,则删除,同步更新 *PidList*.

具体的过程是:实例 T_{new} 到达时将 T_{new} 存入 *NewTransactionTable*,接着比较 *PidList* 和 T_{new} 中的每个项 *item*,更新 *NewTransactionTable*.然后参照 *NewTransactionTable*,在 *ClosedTable* 中添加新的模式或者更新已有的模式.同时更新 *PidList*.如此反复,随着数据的到达不断的更新.图 3 介绍了频繁模式发现的增量更新过程,输入的是数据流 *DS*,输出的是模式流 *PS*.

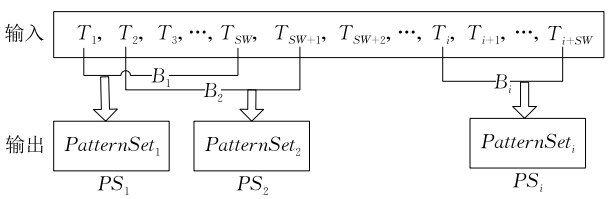


图 3 模式发现示意

示例 2. 以表 1 中的数据为例,设置模式满足的最小支持数为 2,得到满足类约束的频繁模式集合如表 3 所示,共得到 5 条模式.这些模式可用作训练实例.如 P_2〈a1,c1,d1,yes〉增加缺损值,补充为〈a1,?,c1,d1,yes〉即可.

表 3 最小支持数为 2 时得到的闭合模式集合

模式编号	模式	频度
P_1	〈a1,c1,yes〉	3
P_2	〈a1,c1,d1,yes〉	2
P_3	〈a1,d2,no〉	2
P_4	〈b2,c2,d2,no〉	3
P_5	〈d2,no〉	4

3.2 基于模式的决策树方法研究

由于数据流中可能存在着大量无用信息或噪声,对其进行模式挖掘得到的模式结果集合的优势在于:(1)可以去除其中的噪声;(2)得到信息量更大的模式数据.采用这些模式数据进行决策树训练,理论上可以提高创建分类器的效率以及分类的正确率.

示例 3. 以信息增益为例研究基于模式的决策树建立过程.使用模式建立决策树需要考虑频度/权重的使用.为此,将表 1 和表 3 中的数据进行格式修改,得到表 4 和表 5.处理过程是:(1)为原始数据中的每条实例加上权重 1,数据集表示为 *DS*;(2)为模式补充缺损值增加至原始实例的长度,且采用频度值表示权重,数据集表示为 *PS*.

表 4 带有权重的数据集

实例	A_1	A_2	A_3	A_4	C	权重
T_1	a1	b1	c1	d1	yes	1
T_2	a1	b1	c1	d2	yes	1
T_3	a1	b2	c1	d1	yes	1
T_4	a1	b2	c2	d2	no	1
T_5	a1	b3	c1	d1	yes	1
T_6	a1	b3	c1	d2	no	1
T_7	a2	b2	c2	d2	no	1
T_8	a3	b2	c2	d2	no	1

表 5 带有权重的模式集

实例	A_1	A_2	A_3	A_4	C	权重
T_1	a1	?	c1	?	yes	3
T_2	a1	?	c1	d1	yes	2
T_3	a1	?	?	d2	no	2
T_4	?	b2	c2	d2	no	3
T_5	?	?	?	d2	no	4

为了更好的比较在 *DS* 和 *PS* 上分裂属性的选择方法,引入以下变量:

WV_k:表示类属性取值为 C_k 的权重之和;

WV_{ij}:表示属性 A_i 为第 j 个值的权重之和;

WV_{ijk}:表示类属性取值为 C_k 的条件下,属性 A_i 为第 j 个值的权重之和;

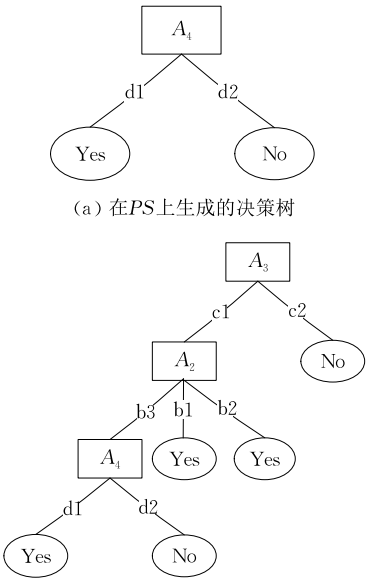
SumWV:表示数据集合中实例的权重之和.设

置的目的是由于存在缺损值.

在 DS 和 PS 上得到的 $Gain(A_1)$ 的计算过程如表 6 所示. 从表 6 中可以看出在 PS 上的计算量少于在 DS 上的计算量. 分别采用信息增益度量准则在 DS 和 PS 上生成决策树模型，得到的树结构如图 4 所示. 从得到的树结构可以看出，在不进行树结构限制的条件下，在 DS 上得到的树结构大于在 PS 上得到的结构. 相比较而言，DS 包含 8 条实例，需要进行 20 个概率值 WV_{ijk} 统计，得到的树包含 8 个节点，其中 5 个叶子节点；PS 包含 5 条实例，需要进行 12 个 WV_{ijk} 统计，得到的树包含 3 个节点，其中 2 个叶子节点. 整体而言，在 PS 进行分类模型训练可以有效减少训练时间，生成的树结构也会更加紧凑.

表 6　在 *DS* 和 *PS* 上得到 *Gain*(A_1)的过程

	DS	*PS*
$SumWV$	8	14
WV_k	$WV_1=4, WV_2=4$ $WV_{11}=6, WV_{12}=1, WV_{13}=1$ $WV_{111}=4, WV_{112}=2$ $WV_{122}=1, WV_{132}=1$	$WV_1=3+2=5, WV_2=2+3+4=9$ $WV_{11}=3+2+2=7$ $WV_{111}=3+2=5,\ WV_{112}=2$
$P(C_k)=\frac{WV_k}{SumWV}$	$P(\text{yes})=WV_1/SumWV=4/8=0.5$ $P(\text{no})=WV_2/SumWV=4/8=0.5$	$P(\text{yes})=WV_1/SumWV=5/14=0.36$ $P(\text{no})=WV_2/SumWV=9/14=0.64$
$P(A_{ij})=\frac{WV_{ij}}{SumWV}$	$P(\text{a1})=6/8=0.75$ $P(\text{a2})=1/8=0.125$ $P(\text{a3})=1/8=0.125$	$P(\text{a1})=7/14=0.5$
$P(C_k\mid A_{ij})=\frac{WV_{ijk}}{WV_{ij}}$	$P(\text{yes}\mid\text{a1})=4/6=0.67$ $P(\text{no}\mid\text{a1})=2/6=0.33$ $P(\text{no}\mid\text{a2})=1$ $P(\text{no}\mid\text{a2})=1$	$P(\text{yes}\mid\text{a1})=5/7=0.71$ $P(\text{no}\mid\text{a1})=2/7=0.29$
$H(C)=-\sum_k P(C_k)\log(P(C_k))$	$H(C)=-0.5\times\log0.5-0.5\times\log0.5=1$	$H(C)=-0.36\times\log0.36-0.64\times\log0.64=0.94$
$H(C\mid A_i)=-\sum_j P(A_{ij})\sum_k P(C_k\mid A_{ij})\log(P(C_k\mid A_{ij}))$	$H(C\mid A_1)=-0.75\times(0.67\times\log0.67+0.33\times\log0.33)-0.125\times(1\times\log1)-0.125\times(1\times\log1)=0.69$	$H(C\mid A_1)=-0.5\times(0.71\times\log0.71+0.29\times\log0.29)=0.43$
$Gain(A_i)=H(C)-H(C\mid A_i)$	$G(A_1)=1-0.69=0.31$	$G(A_1)=0.94-0.43=0.51$



(a) 在*PS*上生成的决策树

(b) 在*DS*上生成的决策树

图 4　在 DS 或 PS 上生成的决策树

采用 PS 生成决策树可能存在的问题在于：(1) 当生成的模式频度低，模式长度短时，生成的树结构可能会过于庞大，且分类正确率不一定提高；(2) 当设定的 k 值过低时，取 top-k 后生成的树结构包含的信息或许会不足，会导致一定的正确率出现降低. 因此，对密度高的数据流进行分类时，由于需要得到的模式频度高、长度合适，可以使用 PS 做训练数据；而对密度低的数据流进行分类时，由于需要得到的模式频度稍低或模式较短，可以考虑取频度最高的 k 个频繁模式和原始数据一起进行训练生成模型. 这么做的优势在于可以提高分裂属性的选择效率，且提高正确率.

3.3　算法设计

本文研究基于频繁模式的分类决策树算法. 由于需要模式参与训练但模式数据中存在缺损值，因此在使用数据时会考虑权重的统计策略. 为此，算法中设计使用 5 条规则：

规则 1. 模式挖掘时仅考虑具有类属性的闭合频繁模式.

规则 2. Top-k 个模式选取时，k 的取值尽可能使 PS 中包含类属性的全部或绝大部分取值.

规则 3. 缺损值参与 $SumWV$ 的统计.

规则 4. 当得到的最好的两个属性分裂准则值差异很小时($<\varepsilon$),则选择权重之和大的属性(缺损值不参与统计)作为分裂节点.

规则 5. 若使用权重比较依然无法区别最优和次优的分裂属性,则任意选择其中一个作为分裂节点.

针对不同特征的数据流,本文设计两种决策树方法.针对高密度数据流,设计基于模式的分类决策树算法 PatHT1(Pattern-based Hoeffding Tree)使用 PS 作为训练实例,如算法 2 所示;针对低密度的数据流,算法 PatHT2 采用少量 top-k PS 与 DS 的组合集合作为训练实例,如算法 3 所示.算法 PatHT1 和 PatHT2 的输入均包括数据流 S,滑动窗口大小 SW,最小支持数阈值 θ,选择正确分裂节点所需的概率 δ;输出为分类决策树模型 HT.

算法 CCFPM 用于增量更新的生成模式,其中的函数 $support()$ 表示支持数.生成的模式是具有类约束的,即必须包含条件属性与类属性,模式长度大于等于 2.使用的 3 个数据结构 $ClosedTable$,$CidList$ 和 $NewTransactionTable$ 在 3.1 节中已经介绍.

算法 HTreeGrow(Hoeffding Tree Growing)使用训练数据增量更新的生成决策树,它也是一种基于 Hoeffding 边界的衍生算法.不同之处在于增加了实例的权重,为此(1)统计信息时需要考虑权重值;(2)选择最佳分裂节点时,会考虑频度相关的统计信息,即当相同分裂准则值时,采用权重之和作为二次选择的标准.算法中使用 ADWIN 作为概念漂移估计器,函数 $G()$ 表示使用分裂准则得到的值;函数 $MaxWeightAttr()$ 用于找到权重值高的属性.

算法 2. PatHT1.

//基于模式的 Hoeffding 树 1

输入:S:数据流

SW:滑动窗口大小

δ:选择正确分裂节点所需的概率

f:衰减因子

ξ:最大允许误差率

输出:HT 决策树

1. For each transaction T_{new} in S Do
   Get novel set of patterns PS_{new}=CCFPM(T_{new},f,ξ,SW);
   Goto step 2 to 4
2. Let HT be a tree with a single leaf (root)
3. Initial counts WV_{ijk} at root
4. For each example (x,y,$weight$) in PS_{new} Do
   HTreeGrow((x,y,$weight$),HT,δ)

算法 3. PatHT2.

//基于模式的 Hoeffding 树 2

输入:S:数据流

SW:滑动窗口大小

δ:选择正确分裂节点所需的概率

f:衰减因子

ξ:最大允许误差率

输出:HT 决策树

1. For each transaction T_{new} in S Do
   Get novel set of patterns PS_{new}=CCFPM(T_{new},f,ξ,SW);
   Goto step 2 to 4
2. Let HT be a tree with a single leaf (root)
3. Initial counts WV_{ijk} at root
4. For each example (x,y,$weight$) in PS_{new} and T_{new} Do
   HTreeGrow((x,y,$weight$),HT,δ)

算法 4. CCFPM.

//发现可变数据流中的约束闭合频繁模式

输入:T_{new}:数据流中的最新实例

SW:滑动窗口大小

f:衰减因子

ξ:最大允许误差率

输出:PS 频繁模式集合

1. Add T_{new} to $NewTransactionTable$
2. Let $inters = T_{new} \cap ClosedTable()$ according to $PidList$
3. Add $inters$ to $NewTransactionTable$
4. For each $TempItem$ in $NewTransactionTable$ Do
   If $interS \in ClosedTable$
   Then update $support(interS)$ $support(interS)\times f+r$
   Else if $support(interS)\geq\xi$
   Then add $\langle interS, support(interS)\rangle$ To $ClosedTable$
5. If $item \in T_{new}$ And $item$ is not in $PidList$
   Then add $item$ To $PidList$

算法 5. HTreeGrow.

//创建 Hoeffding 树

输入:(x,y,$weight$):实例

HT:决策树

δ:选择正确分裂节点所需的概率

输出:HT 决策树

1. Sort (x,y,$weight$) to leaf l using HT
2. Update counts WV_{ijk} with $weight$ at leaf l
3. Compute information gain G for each attribute from counts WV_{ijk}
4. Split leaf
   4.1 If G(Best Attr. BA_1)$-G$(2nd best Attr. BA_2)$>\varepsilon$
   Then let BA_1 to be best attribute BA
   4.2 Else let best attribute

BA=*MaxWeightAttr*(Best Attr., 2nd best Attr.)

4.3 Split leaf l on *BA*

5. For each branch Do

5.1 Start new leaf l and initialize estimators ADWIN

5.2 If ADWIN has detected change

Then create a subtree *st*

5.3 If no sunbtree

Then *st* as a new subtree

Else if *st* is more accurate

Then replace current node with *st*

4 实验分析

4.1 评估方法

实验中将比较 PatHT 算法与贝叶斯分类算法 NaiveBayes(NB)，规则分类算法 DTNB[21]，Rule-Classifer(RC)[22]，决策树分类算法VFDT[4]，HAT[8]，HOT[7]，AdoHOT[10]，ASHT[10]. 其中 DTNB 采用 Leave-one-out 方式评估. 其余分类方式采用 MOA 中的 EvaluatePrequential[23-24] 评估方式来测试. 其中每个实例先作为测试数据而后作为训练数据. 这样得到的正确率是增量更新的，且不需要专门留出测试数据，就保证最大化利用每个数据的信息.

4.2 实验数据

实验中采用了 3 个模拟数据流 SEA、RBF、LED 和 1 个真实数据流 Poker-hand，具体信息如表 7 所示.

表 7　数据流

	#C	#A	#I	概念漂移	离散化
Pocker-hand	10	10	1×10^6	N	N
SEA	2	3	5×10^4	Y	Y
RBF	2	10	1×10^6	no drift drift 0.001	Y
LED	10	24	1×10^6	no drift W=500 W=2000	N

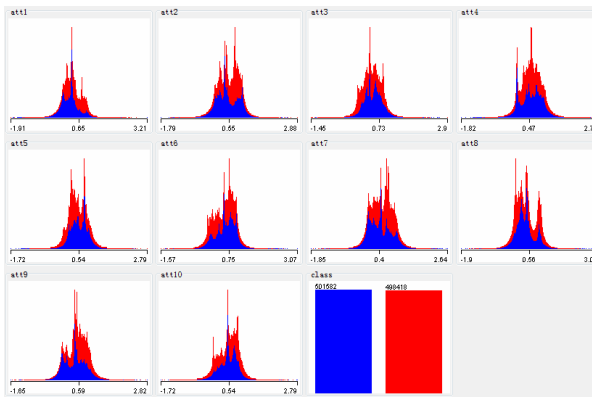
4.2.1 SEA

SEA 是数据流挖掘常用的模拟数据，它来源于 MOA[23]，包含了 3 个条件属性和 1 个类属性. 3 个属性中只有前两个是有关联的，且 3 个属性的值都在 0 与 10 之间. 这个数据分为了 4 块，每块有不同的概念. 分类是通过 $f_1+f_2\leq\eta$ 完成的，其中 f_1 和 f_2 表示前两个属性，η 是阈值. 数据块中最多的值是 9、8、7 和 9.5. SEA 是具有概念漂移特性的数据流. 为了发现频繁模式，使用 WEKA[25] 中的 Discretize 方法对其中的条件属性进行离散化.

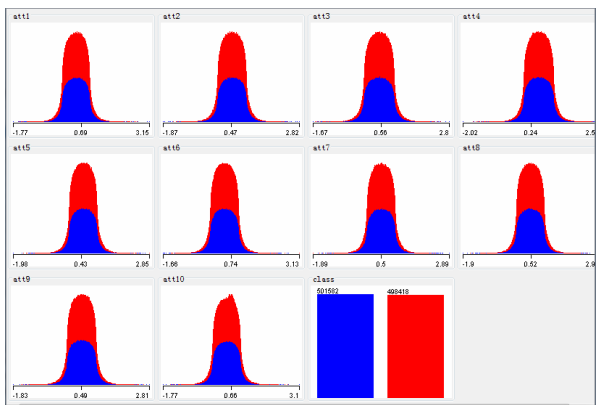
4.2.2 RBF

具有不同概念漂移特征的 RBF 数据由数据流生成器 generators.RandomRBFGeneratorDrift[23-24] 生成，分别生成包含 1 000 000 实例的两种特征数据，即无概念漂移的稳态数据流(no drift)和漂移度为 0.001 的可变数据流(drift 0.001).

RBF 的两个类值分布几乎一样，很平均. 但是属性值的分布满足高斯函数，它集中分布在中间值附近，如图 5 所示. 为了发现频繁模式，使用 Discretize 方法对其中的条件属性进行离散化.



(a) no drift



(b) drift 0.001

图 5　RBF 属性与类值分布

4.2.3 LED

LED 由 MOA 中数据流生成器 generators.LEDGenerator[23-24] 产生. LED 数据用于在一个 7 段 LED 显示器上预测显示的数字. 其包含 24 个二进制条件属性，每个属性有 10% 的可能性会反转；包含 1 个类属性，有 10 个不同值. 实验生成模拟数据流 LED 包含 1 000 000 条实例. 实验生成的数据包含 24 个二进制属性，其中 17 个是无关的.

为了分析 PatHT 算法对概念漂移数据流的处理能力，生成的 LED 具有两种特征：无概念漂移和有概念漂移. 设置概念漂移宽度 W(width of concept drift

change)使用的是 ConceptDriftStream 方法[23-24]. 概念改变的宽度设置为小于和大于滑动窗口宽度.

4.2.4 Poker-hand

真实数据 Poker-hand 来自 UCI①,包含 1000000 条实例,11 个条件属性和 1 个类属性. Poker-hand 数据中的每个实例是一手牌,分类属性用于描述"Poker-hand",包含 10 个取值,如表 7 中所示. 真实数据流的概念漂移特性不易发现.

在 Poker-hand 的 10 个类取值中,第 1 个类值出现在约 50%的实例中,第 2 个类值出现在约 40%的实例中,其余的 8 个类值出现在不足 10%的实例中. 因此,它是一种不平衡数据.

4.3 实验表现

实验运行环境的 CPU 为 2.1GHz,内存为 2GB,操作系统是 Win7,所有的实验采用 Java 实现. 文中将对 10 种算法训练模型的时间、内存消耗以及分类正确率进行比较.

4.3.1 频繁模式挖掘分析

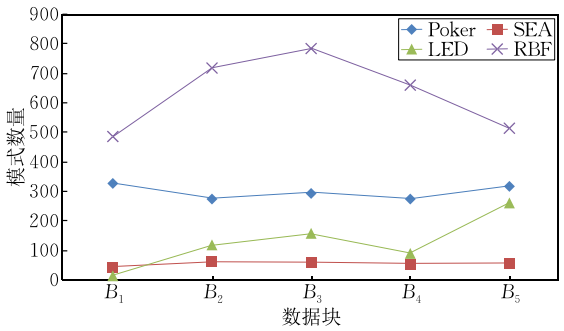
首先分析数据流中发现模式的相关信息. 文献[19]中实验验证了用数据流频繁模式挖掘 CCFPM 中当设定滑动窗口大小 $SW=1000$ 时,设定剪枝步长为 1000 是比较合理的. 这些参数的设置可以使算法 CCFPM 得到性能表现优于同类数据流频繁模式的挖掘方法 MSW[11]、SWP[12]和 CloStream*[20].

为了说明模式变化的趋势,分别对 5 个窗口大小内的实例进行处理,先把它们标记为 5 个数据块:$\{B_1,B_2,B_3,B_4,B_5\}$. 然后对每个数据块进行挖掘生成约束闭合频繁模式集合. 为了分析窗口大小对得到模式的影响,需要对多个数据流进行模式挖掘,结果如图 6 所示. 在设置相同的支持度的条件下,可以得出结论:

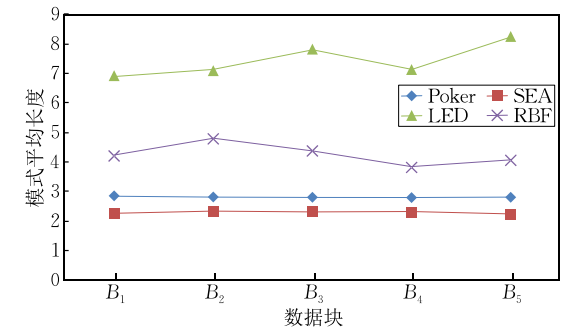
(1)数据流 Poker-hand 得到的模式数量比较多,在 5 个数据块上得到的模式平均长度和最大频度相似,如图 6 所示. 得到模式长度为 2～4,平均长度为 2.83. 由于数据集合中实例长度为 11,因此,模式-实例长度比约为 1∶3.9(2.83∶11). 每个数据块中发现的模式个数平均约为 300,模式-实例个数比约为 1∶3.3(300∶1000).

(2)可变数据流 SEA 得到的模式数量相比是最少的,如图 6 所示. 数据块中发现的模式长度为 2～3,平均长度为 2.3. 模式个数平均约为 57,模式-实例个数比约为 1∶17.5. 即每个数据块中生成的模式数量与数据块中实例数量相比是很少的.

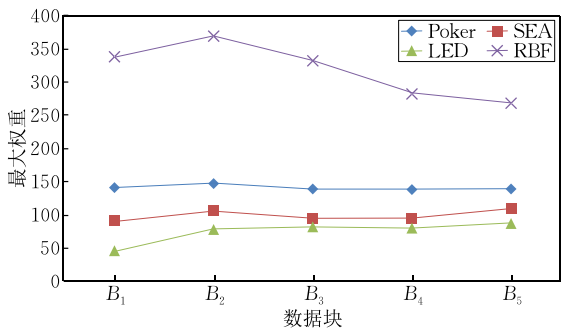
(3)对 RBF 进行模式挖掘得到的模式数量、最大权重在不同数据块中变化是最明显的. 这表明了数据的动态特性. 同时,这种方法得到的模式数量是最多的、权重是最大的,明显多于其他几个数据流,如图 6 所示. RBF 数据块中得到的模式平均长度为 4.27,因此模式-实例长度比约为 1∶2.34. 模式的平均个数为 635.4,模式-实例个数比约为 1∶1.57. 图 7(a)显示了不同数据块中模式的分布,可以看出得到的模式最多的是 3、4 和 5 这 3 个长度.



(a) 模式数量



(b) 模式平均长度

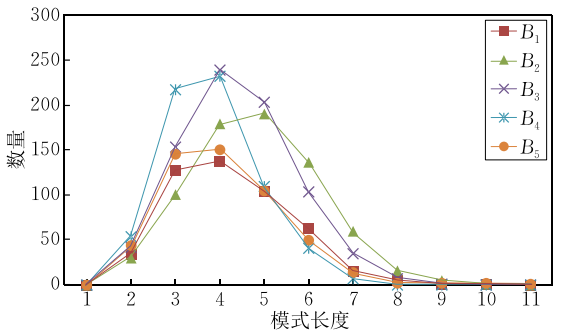


(c) 最大权重/频度

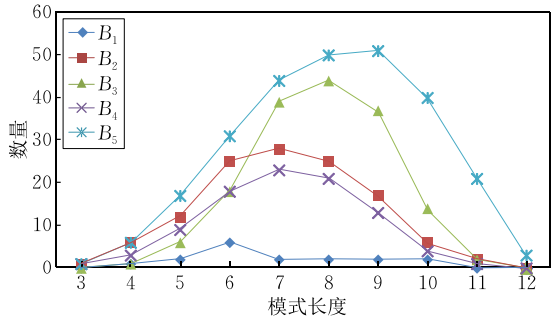
图 6 每个数据块中得到的模式信息

(4)当概念改变宽度 W 设置为 500 时,LED 的模式数量在不同数据块中表现的差异较大,如图 6(a)所示. 模式的数量、长度和权重等信息相比 Poker-hand 和 SEA 变化的比较明显. 原因除了数据本身的特征外,还在于设置的概念改变宽度是小于滑动窗口大小的,滑动窗口不能很好的解决概念

① Frank A, Asuncion A. UCI Machine Learning Repository [EB/OL]. Irvine, CA: University of California, School of Information and Computer Science. http://archive.ics.uci.edu/ml, 2010



(a) RBF(drift 0.001)



(b) LED(W=500)

图 7　数据流 LED 和 RBF 中得到的不同长度模式分布

变化问题. LED 数据块中得到的模式平均长度为 7.46,因此模式-实例长度比约为 1∶3.5. 模式的平均个数为 131,模式-实例个数比约为 1∶7.6.

(5) 当 $W<SW$ 时,数据流 LED 中得到的模式长度分布情况如图 7(b)所示,可以看出得到的模式最多的是 7、8 和 9 这 3 个长度. 其中 B_5 和 B_1 中得到的模式分布完全不同. B_5 和 B_3 中得到的模式数量多于其他 3 个数据块,B_4 和 B_2 得到的模式数量居中,B_1 中的数量最少. 总之,模式分布在不同的数据块中变化很大.

(6) 表 8 是 4 种不同特征数据流上得到的模式-实例信息比. 相比而言,RBF 得到的模式数量最多,SEA 中模式数量最少. 对比与原始的实例长度,LED、Poker-hand 生成的模式长度是相对较短,SEA 与 RBF 中得到的模式长度是相对较长.

表 8　数据流上得到的模式与实例的长度比和数量比

数据流	长度比	数量比
Poker-hand	1∶3.9	1∶3.3
SEA	1∶1.7	1∶17.5
RBF	1∶2.34	1∶1.57
LED	1∶3.5	1∶7.6

4.3.2　模拟数据流分类性能比较

具有概念漂移特征的数据流 SEA,其概念改变宽度不明确. 多个算法对其进行分类处理得到的正确率如表 9 所示. 由于 SEA 具有概念漂移特性且得到的模式频度较低,因此仅采用模式集合作为训练集合得到的数据分类效果很差. 为此,仅对 PatHT2(简记为 PatHT)算法进行比较. 从表中可以看出,PatHT 算法相比较其他方法得到的正确率有一定的提高,但提高程度不高的原因在于得到的是短模式且数量较少.

接着比较模拟数据流 RBF,对稳态(no drift)和动态 RBF(drift 0.001)进行分类处理,得到的结果如表 9 所示. 从表中分析,本文提出的 PatHT 算法的正确率在不同特征的 RBF 上均是最优的. 相比较而言,与已有方式相比,PatHT 算法在稳态 RBF 上提高的正确率较明显. 时间和内存的消耗与已有方式(除 RC 算法之外)相比,PatHT 算法的时间消耗会明显增加,内存消耗增加不明显.

表 9　算法在 SEA 与 RBF 上的性能

	SEA			RBF					
	Drift			No Drift			Drift 0.001		
	时间	正确率	内存	时间	正确率	内存	时间	正确率	内存
NB	**1.01**	82.80	**0.76**	**13.40**	72.90	**0.58**	**13.60**	49.30	**0.58**
RC	2.29	85.65	29.76	1820.00	83.19	8.42	9690.00	52.60	2.78
VFDT	36.64	82.50	4.98	18.53	91.60	8.48	20.67	53.70	6.13
HAT	1.47	86.30	0.99	68.00	92.40	10.77	36.79	67.70	3.08
HOT5	2.62	90.40	0.99	40.59	91.90	12.04	34.96	54.80	10.67
HOT50	2.39	86.00	2.10	87.00	92.00	23.68	82.00	60.90	15.89
AdoHOT5	2.38	86.00	2.10	40.09	92.30	11.91	29.76	59.60	10.71
AdoHOT50	2.34	86.01	2.11	89.00	92.10	20.43	80.00	69.60	18.89
ASHT	2.33	86.01	2.12	18.64	91.60	8.43	18.05	69.70	8.42
PatHT	8.47	**91.70**	1.01	350.00	**97.70**	11.61	150.70	**72.80**	2.45

最后比较模拟数据流 LED. 使用 3 类具有不同概念改变宽度的数据集合,特征是:无概念漂移、概念漂移宽度 $W=500$(小于窗口宽度)和 $W=2000$(大于窗口宽度). 算法在 LED 上进行分类得到的正确率如表 10 所示,可以看出 PatHT 得到的正确率相比较其他算法有一定的增加. 但在 $W=500$ 时表现稍差,这是因为概念改变的宽度小于滑动窗口的宽度,得到的模式信息分布变化很大.

表 10 比较 LED 的正确率

	LED								
	No drift			$W=500$			$W=2000$		
	时间	正确率	内存	时间	正确率	内存	时间	正确率	内存
NB	**24.23**	73.30	3.51	**25.70**	73.40	5.83	**22.21**	73.81	5.79
RC	1438.00	51.80	4.03	1442.00	52.60	4.00	1444.10	52.80	4.60
VFDT	30.25	72.80	**2.04**	32.27	72.90	**1.47**	30.25	73.30	**1.47**
HAT	61.00	73.20	5.94	78.60	**73.90**	3.09	62.89	74.12	4.02
HOT5	97.00	72.80	5.73	105.10	72.80	4.71	101.10	73.30	6.00
HOT50	99.00	72.80	5.64	106.60	72.80	4.72	101.54	73.30	6.06
AdoHOT5	101.00	72.82	5.73	96.50	72.92	4.81	83.60	73.31	3.83
AdoHOT50	103.00	72.82	5.73	97.00	72.92	4.82	85.00	73.31	3.83
ASHT	31.57	72.81	2.13	31.57	72.90	1.56	29.91	73.31	2.43
PatHT	35.84	**73.97**	5.80	32.20	72.20	3.30	35.11	**75.20**	7.22

4.3.3 真实数据流分类性能比较

比较多种算法在真实数据流 Poker-hand 上的分类正确率. 本组实验采用两类 PatHT 算法. PatHT1 仅采用模式集做训练实例,采用的模式数量大约是实际数量的 30%. 算法 PatHT2 采用模式集与原始数据集的组合集合做训练实例,采用的模式数量大约是实际数量的 20%. 由于 DTNB 采用 Leave-one-out 方式评估分类算法,运行速度相比较而言非常慢,因此仅对 50000 条实例进行分析.

PatHT1 得到的分类正确率明显优于 NB、DTNB 和 RC,与其他决策树分类算法得到的正确率几乎一致,如表 11 所示. RC 算法的时间消耗最多,PatHT1 消耗的时间最少,大约比其余算法的平均时间减少了 80%. 这是由于参与训练决策树的模式数量很少,但是额外的时间消耗会出现在生成频繁模式的过程中,使 PatHT1 消耗时间的优势会减低一些.

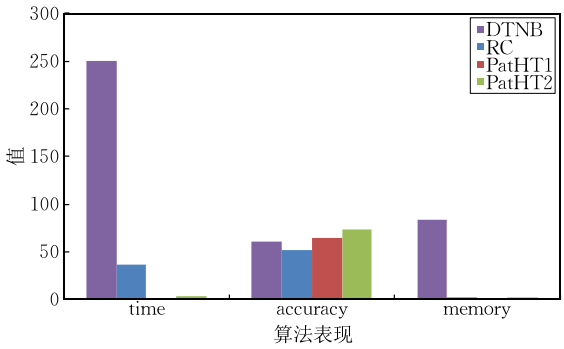
表 11 算法在 Poker-hand 上的性能

	Poker-hand					
	$\sharp I=50000$			$\sharp I=1000000$		
	时间	正确率	内存	时间	正确率	内存
NB	1.87	51.40	**0.76**	**14.65**	46.20	**2.59**
DTNB	250.00	60.85	83.89	—	—	—
RC	36.00	51.40	2.10	365.00	56.70	4.23
VFDT	2.22	63.60	1.20	20.26	79.70	3.39
HAT	3.48	65.20	1.00	41.81	77.40	3.40
HOT5	2.89	62.30	1.33	39.26	79.70	3.50
HOT50	2.89	62.30	1.33	36.11	79.70	3.52
AdoHOT5	2.98	62.30	1.34	36.32	79.71	3.36
AdoHOT50	2.98	62.30	1.34	35.86	79.71	3.40
ASHT	2.23	63.60	1.07	20.14	79.80	14.06
PatHT1	**0.53**	63.80	0.90	—	—	—
PatHT2	3.01	**73.30**	1.45	48.30	**87.80**	3.42

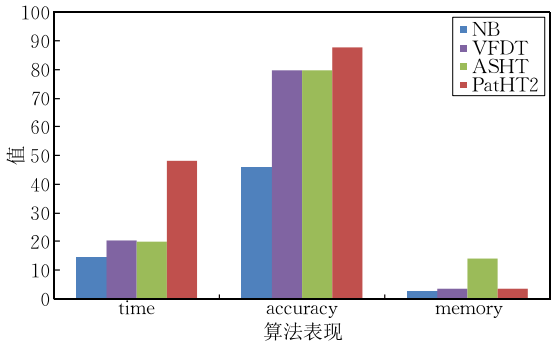
PatHT2 采用的是模式与原始数据同时做训练数据的方式. 从表 11 中可以看出,它消耗的时间和内存与其他算法相比没有明显的增加,但是正确率提高了约 20%. 消耗无明显增加是由于取 top-k 个模式,使得模式-实例个数比大约为 1∶5,没有明显增加训练消耗. 但是由于需要生成模式,因此时间和内存消耗会有一定的增加.

图 8(a)是 PatHT 算法与两种规则分类方法 DTNB 和 RC 的比较. 从图中可以分析出 PatHT1 算法在保证正确率不降低的同时明显减少了时间消耗,而 PatHT2 明显提高了正确率,且使用的时间和内存消耗相比规则分类而言很少.

图 8(b)是 PatHT 算法与 NB 和两种决策树分类方法 VFDT 和 ASHT 的比较. 从图中可以看出 PatHT 算法可以明显增加分类正确率,相比而言时间消耗会明显增加.



(a) PatHT算法与规则分类方法的比较



(b) PatHT算法与决策树分类方法的比较

图 8 PatHT 算法与已有分类方法的比较

综合上述实验可以得出结论：

(1)针对生成模式频度较高、模式长度较长的稳态数据流，PatHT 对其进行处理可以明显提高分类的正确率.

(2)针对具有模式频度较高、模式长度较长特征的稳态数据流，如真实数据流 Poker-hand，PatHT 算法对其进行处理时，PatHT1 可以得到与常见算法相似的正确率，且可以节省大量的时间消耗.

(3)针对不同概念改变特征的动态数据流，PatHT 算法对其进行处理时正确率也会增加.但是概念改变宽度较小时(小于窗口宽度)，得到的算法正确率可能会降低.

(4)由于需要生成模式，因此 PatHT 算法会额外的增加时间和内存消耗，但是由于使用的模式数量少，时间消耗增加不明显.由于使用的是高频度的模式，使得生成的分类模型更加的紧凑，因此消耗的内存相比经典算法会有一定程度的降低.

5　总　结

常用的数据流分类流程为输入数据、训练数据、生成分类模型.这些训练实例集合中可能存有大量无用信息或噪音.因此本文提出一种新的分类流程：输入-模式-训练-模型，即在数据用于训练之前先进行模式挖掘.对每个实例进行增量更新的模式挖掘，发现具有类约束的闭合频繁模式，这样可以去除原始数据中的无用信息，且得到信息量更大的模式数据.针对不同的数据流特征，这些模式集合会以独立或组合的方式参与分类决策树模型的训练.从大量的实验结果分析可以得出，在真实数据流和不同概念改变特征的模拟数据流上，使用基于模式的分类决策树可以有效的提高分类的正确率或明显降低训练时间.

但是由于需要发现频繁模式，因此不足之处是不能对连续值的属性直接进行处理，需要离散化，且由于需要增加频繁模式生成过程，因此时间或内存消耗上会有所增加.

参　考　文　献

[1] Farid D M, Zhang L, Hossain A, et al. An adaptive ensemble classifier for mining concept-drifting data streams. Expert Systems with Applications, 2013, 40(15): 5895-5906

[2] Widmer G, Kubat M. Learning in the presence of concept drift and hidden contexts. Machine Learning, 1996, 23(1): 69-101

[3] Ikonomovska E, Gama G, Džeroski S. Online tree-based ensembles and option trees for regression on evolving data streams. Neurocomputing, 2015, 150(PB): 458-470

[4] Domingos P, Hulten G. Mining high-speed data streams// Proceedings of the 6th ACM International Conference on Knowledge Discovery and Data Mining. Boston, USA, 2000: 71-80

[5] Hulten G, Spencer L, Domingos P. Mining time-changing data streams//Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining. New York, USA, 2001: 97-106

[6] Gama J, Rocha R, Medas P. Accurate decision trees for mining high-speed data streams//Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining. New York, USA, 2003: 523-528

[7] Pfahringer B, Holmes G, Kirkby R. New options for hoeffding trees//Proceedings of the 20th Australian Joint Conference on Artificial Intelligence. Gold Coast, Australia, 2007: 90-99

[8] Bifet A, Gavaldá R. Adaptive learning from evolving data streams//Proceedings of the 8th International Symposium on Intelligent Data Analysis. Lyon, France, 2009: 249-260

[9] Bifet A, Gavaldá R. Learning from time-changing data with adaptive windowing//Proceedings of the 7th SIAM International Conference on Data Mining. Minnesota, USA, 2007: 443-448

[10] Bifet A, Holmes G, Pfahringer B. New ensemble methods for evolving data streams//Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining. New York, USA, 2009: 139-148

[11] Li Guo-Hui, Chen Hui. Mining the frequent patterns in an arbitrary sliding window over online data streams. Journal of Software, 2008, 19(19): 2585-2596(in Chinese)
(李国徽，陈辉. 挖掘数据流任意滑动时间窗口内频繁模式. 软件学报，2008, 19(19): 2585-2596)

[12] Chen H, Shu L C, Xia J L, Deng Q S. Mining frequent patterns in a varying-size sliding window of online transactional data streams. Information Sciences, 2012, 215(12): 15-36

[13] Gama J, Kosina P. Recurrent concepts in data streams classification. Knowledge and Information Systems, 2014, 40(3): 489-507

[14] Klinkenberg R. Learning drifting concepts: Example selection vs. example weighting. Intelligence Data Anal, 2004, 8(3): 281-300

[15] Kosina P, Gama J. Very fast decision rules for classification in data streams. Data Mining and Knowledge Discovery, 2015, 29(1): 168-202

[16] Gama J, Zliobaite I, Bifet A, et al. A survey on concept drift adaptation. ACM Computing Surveys, 2014, 46(4): 1-37

[17] Gama J, Medas P, Castillo G, Rodrigues P. Learning with drift detection//Proceedings of the 17th Brazilian Symposium on Artificial Intelligence. Maranhao, Brazil, 2004: 286-295

[18] Baena G M, Campo A J, Fidalgo R, et al. Early drift detection method//Proceedings of the 4th International Workshop on Knowledge Discovery from Data Streams. Berlin, Germany, 2006: 77-86

[19] Han Meng, Wang Zhi-Hai, Yuan Ji-Dong. Efficient method for mining closed frequent patterns from data streams based on time decay model. Chinese Journal of Computers, 2015, 38(7): 1473-1482(in Chinese)
(韩萌, 王志海, 原继东. 一种基于时间衰减模型的数据流闭合模式挖掘方法. 计算机学报, 2015, 38(7): 1473-1482)

[20] Yen S J, Wu C W, Lee Y S, et al. A fast algorithm for mining frequent closed itemsets over stream sliding window//Proceedings of the 2011 IEEE International Conference on Fuzzy Systems. Taiwan, China, 2011: 996-1002

[21] Hall M, Frank E. Combining naive bayes and decision tables //Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference. Florida, USA, 2008: 318-319

[22] Gama J, Kosina P. Learning decision rules from data streams //Proceedings of the 22nd International Joint Conference on Artificial Intelligence. Barcelona, España, 2011: 1255-1260

[23] Bifet A, Holmes G, Kirkby R, Pfahringer B. MOA: Massive online analysis. Journal of Machine Learning Research, 2010, 11: 1601-1604

[24] Witten I H, Frank E. Data Mining: Practical Machine Learning Tools and Techniques. 2nd Edition. Beijing: China Machine Press, 2005

HAN Meng, born in 1982, Ph. D. candidate, associate professor. Her research interests include data mining and machine learning.

WANG Zhi-Hai, born in 1963, Ph. D., professor, Ph. D. supervisor. His research interests include data mining and machine learning.

DING Jian, born in 1977, M. S., associate professor. His research interests include computer application and data mining.

Background

Data stream is a massive, unbounded sequence of data elements continuously generated at a rapid rate. Advanced analysis of evolving data streams is quickly becoming a key area of data mining research as the number of applications demanding such processing increases. An important character of evolving data stream is the concept drift, so classification methods should adapt it automatically.

Although many classification methods have been proposed, studies find that frequent patterns can be used to build high quality classification models. The advantages of pattern-based classification methods lie in: (1) un-frequent itemsets may be caused by random noises which are harmful to classification methods. But frequent patterns always carry reliable information gains to construct methods. (2) Generally, pattern has more information gain than a single attribute. (3) The discovered patterns are always simple and easy to explain. Therefore, interesting, frequent and distinguished pattern can be used for effective classification.

In this paper, we focus on mining closed frequent patterns on data streams, and study classification decision tree algorithms based on these patterns. (1) Existed data stream classification cycle includes three steps: input-learning-model. In order to improve the efficiency of building model and the accuracy rate of classification, we proposed the four steps cycle: input-pattern-learning-model. (2) Incremental mine closed frequent patterns which must have class attributes. We get top-k frequent patterns to build decision trees in order to improve the efficiency of choosing an optimal splitting attribute. (3) Sliding window models and time decay model are used to deal with new examples. (4) We use ADWIN to detect concept change in evolving data streams.

Compared with state-of-art algorithms, an evaluation study on synthetic data streams with different widths of concept change and real-world data streams shows that the proposed method performs better than them.