一种基于数据流模式表示的半懒惰式分类算法

江晶晶 王志海 原继东
(北京交通大学计算机与信息技术学院 北京 100044)
(交通数据分析与挖掘北京市重点实验室 北京 100044)

摘 要 依据从大规模数据中抽取的模式来建立分类模型是模式挖掘的重要研究问题之一。一种可行的方法是根据模式集合建立贝叶斯分类模型。然而,目前基于模式的贝叶斯分类模型大多是针对静态数据集合的,通常不能适应于高速动态变化与无限的数据流环境。对此,提出一种数据流环境下基于模式发现的贝叶斯分类学习模型,其采用半懒惰式学习策略,针对分类实例在不断更新的频繁项集合上建立局部的分类模型;为加快流数据处理的速度,提出了结构更为简单的混合树结构,同时提出了给定项限制的模式抽取机制以减少候选项集的生成;对数据流中模式抽取不完全的情况,使用平滑技术处理未被抽取的项。大量实验分析证明,相较于其他数据流分类器,所提模型具有更高的分类正确率。

关键词 数据流,频繁模式,贝叶斯,半懒惰式学习
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.07.030

Partially-lazy Learning Classification Algorithm Based on Representation of Data Stream Model

JIANG Jing-jing WANG Zhi-hai YUAN Ji-dong
(School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China)
(Beijing Key Lab of Traffic Data Analysis and Mining, Beijing 100044, China)

Abstract Utilizing patterns extracted from large scale data to build classification model is one of important research problems. Exploiting patterns to estimate Bayesian probability is a feasible approach. However, most of the existing pattern-based Bayesian classifiers aim at static data set, which cannot adapt to the dynamic data stream environment. A Bayesian classification model, named PBDS(Pattern-based Bayesian classifier for Data Stream), based on pattern discovery over data streams was proposed. PBDS constructs local model for unseen case based on continuously updated frequent item sets with partially-lazy learning method. To accelerate data processing, the simpler data structure, i.e., hybrid trees structure was proposed, and pattern extracting mechanism was proposed to reduce the generation of candidate itemsets. Smoothing technique was used to handle incomplete itemset extraction in the data stream. Extensive experiments on real-world and synthetic data streams show that PBDS is more accurate than state-of-the-art data stream classifiers.

Keywords Data stream, Frequent pattern, Bayesian, Partially-lazy learning

1 引言

基于模式的分类模型是指在大规模数据集合中抽取模式集合,再根据抽取到的模式集合建立分类模型。模式是对数据的局部描述,它是项的集合,一个项是一个"属性-值"序偶。频繁模式是指一个模式在数据中出现的次数超出了规定的阈值。基于频繁模式的贝叶斯分类模型使用模式的频繁性来估计贝叶斯理论中联合概率的值。与 Naïve Bayes(NB)[15] 相比,这种建立贝叶斯分类模型的方式在模型的建立过程中考虑了属性之间的依赖关系,使得建立的分类模型更加适用于现实生活中的数据。然而,已有的基于模式的贝叶斯分类器[2-3,7,18]大多针对静态数据集合,不适用于高度动态和无限的数据流环境。

数据流是一个高速的、理论上无限的数据元素的连续序列,并且数据的分布可能随时间而发生变化。不同于静态数据集合上分类模型的建立,基于模式的数据流分类器需要处理以下问题:1)数据流算法在任意时刻只能获取数据流的一个数据片段,数据流算法难以确定所挖掘的模式集合相对整个数据的频繁性和完整性;2)当分类请求出现时,算法所能获取的数据段中可能不包含待分类实例中所有的项;3)由于数

到稿日期:2016-05-16 返修日期:2016-09-30 本文受国家自然科学基金(61672086),北京市自然科学基金(4142042)资助。
江晶晶(1992-),女,硕士生,主要研究领域为数据挖掘和机器学习,E-mail:14120392@bjtu.edu.cn;王志海(1963-),男,博士,教授,博士生导师,主要研究领域为数据挖掘和机器学习,E-mail:zhhwang@bjtu.edu.cn;原继东(1989-),男,博士,讲师,主要研究领域为数据挖掘和机器学习,E-mail:12112078@bjtu.edu.cn。

据流具有高速和无界的特点，算法必须在有限的处理时间和内存消耗内完成数据处理；4)数据流中数据的分布可能发生变化，算法必须能够适应变化。

本文提出了基于模式的数据流贝叶斯分类模型(Pattern-based Bayesian classifier for Data Stream，PBDS)。PBDS 使用半懒惰式学习(Partially-lazy Learning)策略，半懒惰式学习是一种介于懒惰式学习和急切式学习之间的学习策略。半懒惰式分类器具有捕获局部数据的能力，同时相较于懒惰式分类器能够更快速地响应请求，因此半懒惰式分类器适用于处理高度动态和复杂的学习环境，例如数据流[22]。PBDS 分类模型在训练阶段完成对数据的初步处理，即获取数据流中的项集，并且随着数据的变化实时更新项集。当有分类请求时，PBDS 根据待分类实例在当前项集集合中抽取频繁项集并建立针对待分类实例的特定局部分类模型。

为在连续数据流上在线挖掘频繁项集，本文提出一种单次扫描算法(Find Frequent Itemsetalgorithm on data stream，FFI)。FFI 使用滑动窗口模型在数据流中捕获数据。滑动窗口模型是一种数据流处理模型，它使用固定大小的窗口，窗口会随着时间进行滑动来维护固定数目的事务。为在训练阶段加快对流数据的处理速度，本文基于 SFI-forest[17] 提出结构更为简单的混合树结构 HTS(Hybrid Trees Structure)用于存储当前窗口中数据包含的项集；同时提出一种给定项限制的模式抽取机制以减少候选项集的生成。

PBDS 分类器抽取包含尽可能多的项的互不相交的频繁项集集合，同时在这一前提下 PBDS 选择项集集合的最小集。对于数据流中项集抽取不完全的情况，PBDS 使用平滑方式处理未被抽取到的项，即在概率计算时使用平滑对未被抽取到的项集的乘积近似值进行估计。

本文第 2 节介绍涉及的基本概念和相关研究现状；第 3 节描述数据流上的频繁项集挖掘算法，具体包括混合树结构的建立、窗口的更新机制和频繁项集的抽取机制；第 4 节系统地介绍 PBDS 分类器中建立乘积近似值的方法和抽取项集的一般原则；第 5 节通过大量的实验对 PBDS 分类算法进行评价；最后总结全文和展望进一步工作。

2 相关概念及研究现状

基于模式的贝叶斯分类器使用频繁项集来估计贝叶斯理论中联合概率的值。本文使用滑动窗口模型获取定长的事务流，并抽取流数据中的频繁模式，建立基于模式的贝叶斯分类模型。本节将详细介绍涉及的基本概率和研究现状。

2.1 基本概念和定义

数据流是由高速生成的事务组成的无限的连续序列，事务是项的集合。一个项是一个“属性-值”序偶，属性是数据特征的描述，值表示与之相关的信息。具体的定义如下。

定义 1(项) A 是数据的一个标签，称之为属性，用于描述数据的特征。Ω 是属性 A 对应的离散域；一个项 (A,a_i) 是指定的属性 A 的值 a_i(其中 $a_i\in\Omega$)。通常使用单个字符 a_i 来表示项 (A,a_i)。

定义 2(数据流) 事务数据流 $DS=[T_1,T_2,\cdots,T_n,\cdots)$ 是事务的无限序列，其中 T_n 是最近的事务，事务 $T_{tid}=\{a_1,a_2,\cdots,a_k\}$ 是项的集合，k 是事务中项的个数，tid 是事务标识。

定义 3(滑动窗口) 滑动窗口 $SW_{n-|w|+1}=[T_{n-|w|+1},T_{n-|w|+2},\cdots,T_n]$ 是定义在数据流上包含 w 个最近事务的集合，其中 w 是滑动窗口的大小。

定义 4(项集支持度) 项集 X 的支持度 $sup(X)$ 是指包含项集 X 的事务数 $count(X)$ 与数据集中所有事务数 N 的比值。

$$sup(X)=\frac{count(X)}{N} \tag{1}$$

频繁项集是指给定最小支持度阈值 min_sup，若一个项集的支持度超过了阈值即 $sup(X)\geqslant min\_sup$，则项集为频繁项集。

定义 5(数据流项集的类支持度) 设定数据流中任意属性 C 为类属性，基于滑动窗口模型的数据流中项集的类支持度定义如下：

$$sup_i(X)=\frac{SW(X)|_{c_i}}{|w|},c_i\in C \tag{2}$$

其中，$SW(X)|_{c_i}$ 是当前窗口中所有类标为 c_i 且包含项集 X 的事务数；w 是窗口的大小，即当前窗口中所有事务的个数。

定义 6(数据流项集的支持度) 项集在一个窗口中的支持度是这个项集的所有类支持度的总和，即：

$$sup(X)=\sum_{\forall c_i\in C}sup_i(X) \tag{3}$$

在分类问题中，指定数据集中的一个属性作为类属性或者类标签。将已知类标的实例称为训练实例，未知类标的实例称为测试实例或待分类实例。

2.2 研究现状

分类是在已有数据的基础上构建一个分类模型，该模型能够把数据库中的数据记录映射到给定类别中的某一个，从而可以用于数据预测。贝叶斯分类器[15]是一种已有广泛研究的分类器。构造贝叶斯分类模型的一个难点是对贝叶斯理论中联合概率的计算，这通常需要借助某种简化模型。最经典的简化模型是 NB 中提出的条件独立性假设：对于给定类属性，数据集中的所有属性都相互独立。因此，基于 NB 模型对概率 $P(T,c_i)$ 的计算可以转化为式(4)。

$$\begin{aligned}P(T,c_i)&=P(a_1,a_2,\cdots,a_n,c_i)\\&\approx P(c_i)P(a_1|c_i)P(a_2|c_i)\cdots P(a_n|c_i)\end{aligned} \tag{4}$$

但是 NB 提出的条件独立性假设在现实数据中很难成立，因此许多弱化 NB 的条件独立性假设的算法被提出：一种是以研究属性之间的低阶依赖为代表的贝叶斯网络[9,14]；另一种是从属性之间的高阶依赖出发，通过在数据集中抽取频繁模式来建立联合概率 $P(T,c_i)$ 的乘积近似值[2-3,7,18]。目前存在两种建立联合概率的乘积近似值的模型，即条件独立模型和条件依赖模型。

给定的数据流 DS 包含属性 A_1,A_2,A_3,A_4,A_5 和类属性 C。c_i 是任意的类属性值，$T=\{a_1,a_2,\cdots,a_5\}$ 是待分类实例，式(5)给出了基于条件依赖模型对联合概率 $P(T,c_i)$ 的估计。

$$\begin{aligned}P(T,c_i)&=P(a_1,a_2,\cdots,a_5,c_i)\\&\approx P(c_i)P(a_1|c_i)P(a_2|a_1c_i)\cdots P(a_5|a_1a_2a_3a_4c_i)\end{aligned} \tag{5}$$

条件依赖模型显性地给出了属性之间的依赖关系。对于

联合概率的估计，也可以基于条件独立模型将其表示为式(6)的形式：

$$P(T,c_i)=P(a_1,a_2,\cdots,a_5,c_i)$$
$$\approx P(c_i)P(a_1a_2a_3\,|\,c_i)P(a_4a_5\,|\,c_i) \quad (6)$$

从式(6)可知，对于给定的类属性 C，属性集 A_1，A_2，A_3 和 A_4，A_5 相互独立。式(5)和式(6)称为联合概率 $P(T,c_i)$ 的乘积近似值或近似值。

已有的基于静态数据集合的贝叶斯估计方法可以根据建立乘积近似值的方式的不同分为两类。1)基于条件依赖模型构建乘积近似值[7,18]。当有分类请求时，抽取频繁项集，接着在抽取的项集上依据条件依赖模型建立乘积近似值，并且针对待分类实例在每一个类标值下建立结构相同的乘积近似值。抽取项集的原则是抽取尽可能多的项集来覆盖待分类实例，并且抽取的项集之间包含尽可能多的重复项。这种方法的缺点：首先，乘积近似值与类标的关联性非常弱；其次，算法添加频繁项集直到没有可用的频繁项集，以致乘积近似值中包含乘积项的个数很大程度上依赖于给定的最小支持度阈值。2)遵从条件独立性模型来建立联合概率的乘积近似值[2-3]。EnBay 分类算法[2]在遵从属性条件独立假设的前提下，选择长的、非重复的完全覆盖待分类实例的频繁项集的最小集合。对于同一待分类实例，EnBay 根据类标值分别抽取不同的频繁项集集合来构建不同结构的乘积近似值。在 EnBay 中无限循环项集抽取算法，直到抽取出的项集集合完全覆盖待分类实例。抽取过程是迭代式的，在每一次迭代的过程中都需要计算并比较项集所属属性集之间的依赖程度，从而选择局部最优项集。因此，算法需要将大量的工作集中于分类器测试阶段，分类器建立过程时间长，从而导致响应分类请求速度慢。

DSM-FI 算法[17]用于在连续的事务型数据流上增量地挖掘频繁项集。该算法使用界标窗口模型来获取流数据，提出了数据结构 SFI-forest 用于保存数据流中事务的频繁项集集合，并且提出了频繁项集查找机制用于抽取频繁项集。但是 DSM-FI 算法中，数据结构 SFI-forest 的构造过程较为复杂；另外，DSM-FI 算法在抽取频繁项集时需要进行多次的筛选，在这个过程中需要产生候选项集。

3　数据流上的模式发现算法

为建立基于模式的贝叶斯分类模型，本文提出了一种单次扫描算法 FFI，其使用滑动窗口模型在连续的记录型数据流上挖掘频繁项集；基于 SFI-forest[17] 提出了结构更为简单的混合树结构。

3.1　数据流中频繁项集的生成和动态更新

FFI 算法使用滑动窗口模型来获取流数据。对于当前窗口 $SW=[T_1,T_2,\cdots,T_w]$，FFI 读取事务 $T_k(k\in[1,w])$，并将 T_k 按其类标值进行划分，对不同划分的事务集分别建立混合树结构(Hybrid Tree Structure，HTS) $HTS=\{HTS_i\}$，$\forall c_i\in C$(设定类属性为 C)。项集的抽取也是按类标值在不同的 HTS_i 中分别进行的。算法主要包括以下 3 个步骤：

(1)FFI 算法读取当前窗口中的事务，并且按事务的类标值对事务进行划分。根据事务的划分结果将事务加入到相应的混合树结构 $HTS=\{HTS_i\}$，$\forall c_i\in C$。

(2)当窗口中的事务发生变化(新的事务到来或旧的事务被丢弃)时，FFI 算法需对混合树结构 HTS 进行修剪，删除不在当前窗口中的事务信息。

(3)当有分类请求时，FFI 算法需根据待分类实例中的项在当前混合树结构 $HTS=\{HTS_i\}$ 中分别抽取频繁项集集合。

混合树结构 $HTS=\{HTS_i\}$ 是由多个子结构组成的，每一个子结构 HTS_i 都包含 3 个组成部分：频繁项列表(a list of Frequent Items，FI-list)、频繁项树(Frequent Item Trees，FIT)和辅助频繁项列表(a list of Subsequence Frequent Itemsets，SFI-list)。

FI-list 中包含当前窗口中事务集的所有非重复项。FI-list 中的每一个结点都由 3 个部分组成，即 $\{a, a.count, a.head\text{-}link\}$。其中，$a$ 表示项；$a.count$ 表示当前混合树结构中包含项 a 的事务的个数；$a.head\text{-}link$ 用于指向 a. FIT 中的头结点，并且 FI-list 中的每一个元素都对应一个 a. FIT 和一个 a. SFI-list。a. FIT 中每一个结点都由 3 个部分组成，即 $\{a, a.count, a.node\text{-}link\}$，其中 a 表示项；$a.count$ 表示包含这个结点的分支中所有事务的个数；$a.node\text{-}link$ 用于指向分支中下一个结点，一般为项 a 所在事务的后续项。a. SFI-list 中的每一项都由两个部分组成，即 $\{a, a.count\}$，其中 a 表示项，$a.count$ 表示当前混合树结构中包含项 a 的事务的个数。

以表 1 数据集为例来说明 HTS 的构建过程。

表 1　数据集

事务	属性 A	属性 B	属性 C	属性 D	属性 E
T_1	A_1	B_1	C_1	D_2	E_2
T_2	A_1	B_1	C_1	D_1	E_2
T_3	A_2	B_1	C_1	D_2	E_1
T_4	A_3	B_2	C_1	D_2	E_1
T_5	A_3	B_3	C_2	D_2	E_1
T_6	A_3	B_3	C_2	D_1	E_2
T_7	A_2	B_3	C_2	D_1	E_1

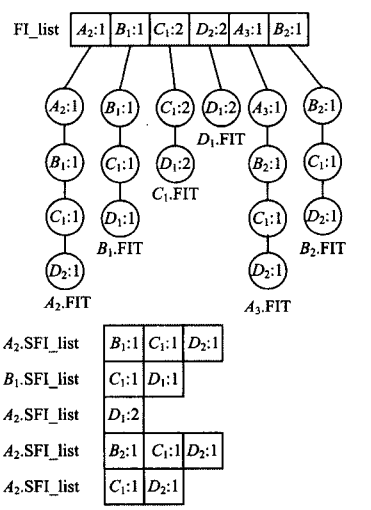

图 1　类值为 E_1 的混合树结构 HTS_1

设属性 E 为类属性，给定的窗口大小 $w=4$，则当前窗口中已有 4 个事务：T_1，T_2，T_3，T_4。根据事务的类标，将窗口中的 4 个事务分为两组，并分别建立混合树结构 $HTS=\{HTS_1, HTS_2\}$。类标值为 E_1 的 T_3 和 T_4 对应 HTS_1（见图 1）。类标值为 E_2 的 T_1 和 T_2 对应 HTS_2（见图 2）。图 1 中 FI_list 内项 D_2 所在的结点因其对应的 D_2. SFI-list 为空，所以在图中未显示。同样，图 2 中 D_2. SFI-list 和 D_1. SFI-list 为空而未在图中显示。

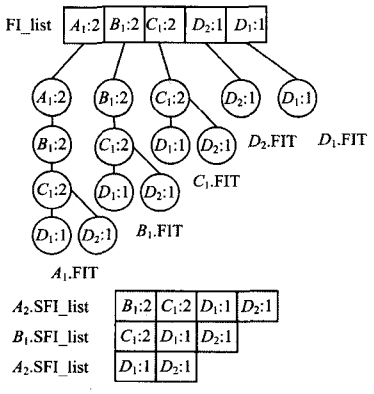

图 2 类值为 E_2 的混合树结构 HTS_2

FFI 算法采用滑动窗口模型来获取流数据，当窗口中的数据发生变化时更新相应的混合树结构，具体分为两个阶段：1）窗口初始化阶段，在这个阶段窗口中事务的个数小于窗口的大小，当新事务到来时只需要将新事务按其类标值添加到相应的混合树结构中即可；2）窗口滑动阶段，在这个阶段窗口中事务的个数等于窗口大小，当有新的事务到达窗口时：

①需要删除当前窗口中最旧的事务（相对于窗口中其他事务最先到来的事务）；

②不仅要在窗口中删除最旧的事务，还需要根据该事务的类标值在相应的混合树结构中删除包含该事务中项的结点或者更新相关结点的计数；

③在执行前两步操作之后，再将新事务加入窗口中，并且将新事务按其类标值添加到相应的混合树结构中。

举例：以表 1 中的数据集为例，设窗口大小为 4，窗口内的事务分别是 T_1，T_2，T_3，T_4。当 T_5 到达窗口时，将 T_1 从窗口和混合树结构 HTS_2 中删除，然后在窗口和 HTS_1 中添加 T_5 的相关信息。图 3 和图 4 是窗口滑动后的混合树结构 HTS。

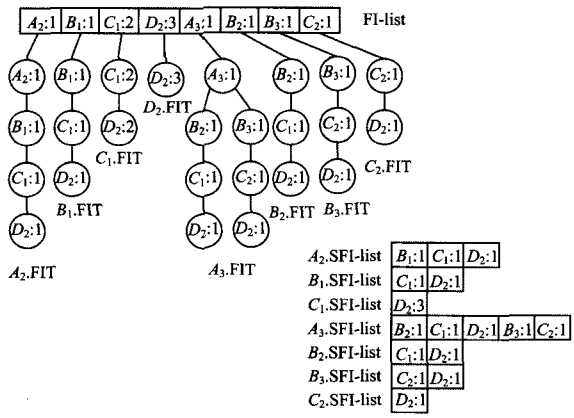

图 3 窗口滑动后类值为 E_1 的混合树结构

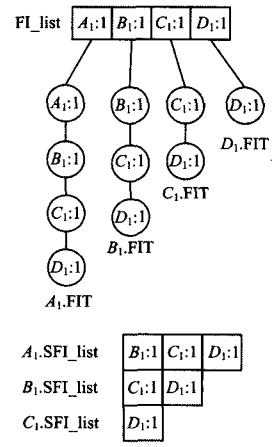

图 4 窗口滑动后类值为 E_2 的混合树结构

3.2 给定范围下数据流频繁项集的抽取

本文提出的基于频繁项集的数据流分类算法 PBDS 采用半懒惰式的学习策略，有分类请求时，根据待分类实例 T 在混合树结构 $HTS=\{HTS_i\}$ 中分别抽取频繁项集集合。本文提出了一种可抽取给定范围的频繁模式的机制。

给定数据流 DS 和待分类实例 $T_{test}=\{a_1, a_2, \cdots, a_n\}$，当有分类请求时，根据待分类实例 T_{test} 分别在混合树结构 $HTS=\{HTS_i\}$ 中抽取与待分类实例具有相同项的频繁项集集合。对于每一个子结构，项集抽取算法分为以下几步进行：

（1）在 HTS_i 中，对于 T_{test} 中的每一项 a_i，$i\in(1,n)$，在 FI_list 中找到具有相同项的结点，如果找到，则继续步骤（2）；如果未找到，则换下一个项 a_{i+1} 在 FI_list 中继续查找，直到找到具有相同项的结点；若 T_{test} 中所有项在当前 HTS_i 的 FI_list 中都不存在具有相同项的结点，则返回空项集集合。

（2）根据 FI_list 中包含项 a_i 的结点找到项 a_i 对应的 a_i. SFI_list；对于 a_i 在事务 T_{test} 中的后继项 $\{a_{i+1}, a_{i+2}, \cdots, a_{n-1}\}$，分别在 a_i. SFI_list 中查找具有相同项的元素，若找到，则这些项连同项 a_i 组成最长项集。

（3）根据 FI_list 中包含项 a_i 的结点找到项 a_i 对应的 a_i. FIT，并检查找到的最长项集在 a_i. FIT 中是否在同一分支上，若在同一分支上，则最长项集为找到的最佳项集；若存在不在同一条分支上的项，则对最长项集进行修剪，删除不在同一分支上的项，从而得到最佳项集。

（4）根据最小支持度阈值 min_sup 对最佳项集进行修剪，检查最佳项集中每一项的支持度是否超过 min_sup，从而得到最佳频繁项集。

（5）对未被选入最佳频繁项集的项重复进行步骤（1）－步骤（4）的操作，直到所有的项都被测试过。

4 PBDS 分类器

半懒惰式分类器在训练阶段建立项集形式的密集数据表达，当有分类请求时，才对待分类实例建立特定的局部分类模型。

4.1 在数据流中创建乘积近似值

PBDS 使用在数据流中抽取的频繁模式来估计贝叶斯概率，在属性独立假设下依照条件独立模型建立乘积近似值。

(1)抽取到的项集决定乘积近似值的结构。

例如，给定的数据流 DS 包含属性 A_1,A_2,A_3,A_4,A_5 和类属性 C；c_i 是任意的类属性值，$T=\{a_1,a_2,\cdots,a_5\}$ 是待分类实例。为估计概率 $P(T,c_i)$ 的值，算法需要从混合树结构 HTS_i 中抽取项集。如果抽取的项集集合为$\{\{a_1,a_2,a_3\},\{a_4,a_5\}\}$，则建立的用于估计概率的乘积近似值为 $P(T,c_i)\approx P(c_i)P(a_1a_2a_3|c_i)P(a_4a_5|c_i)$；如果抽取的项集集合为$\{\{a_1,a_2\},\{a_3\},\{a_4,a_5\}\}$，则建立的用于估计概率的乘积近似值为 $P(T,c_i)\approx P(c_i)P(a_1a_2|c_i)P(a_3|c_i)P(a_4a_5|c_i)$。

(2)对同一个待分类实例在各个类标值上分别抽取项集，即乘积近似值的结构与类标相关[2]。

例如：设定类属性 C 有属性值 c_1,c_2,c_3。为预测待分类实例 T 的类标，PBDS 需要从混合树结构 HTS_1，HTS_2 和 HTS_3 中分别抽取项集集合来建立概率 $P(T,c_1)$，$P(T,c_2)$，$P(T,c_3)$ 的乘积近似值。

(3)对于给定的类标值，乘积近似值中每一个乘积项所隐含的属性集之间相互独立。

例如，联合概率 $P(T,c_i)$ 的乘积近似值为 $P(T,c_i)\approx P(c_i)P(a_1a_2a_3|c_i)P(a_4a_5|c_i)$，其中乘积项 $P(a_1a_2a_3|c_i)$ 和 $P(a_4a_5|c_i)$ 的属性集分别为 $\{A_1,A_2,A_3\}$ 和 $\{A_4,A_5\}$，则对于给定的类标 c_i，这两个属性集相互独立。

下面将描述 PBDS 中项集抽取的原则，用于建立乘积近似值的项集必须满足下列要求。

(1)抽取的项集之间不包含重复的项

两个项集包含重复的项即表明这两个项集的所有项的集合之间存在交集。如果选择的项集之间存在重复的项，则构建的乘积近似值将不满足属性独立假设，因为乘积项的属性集之间存在交集。

(2)选择尽可能长的项集

项集的长度是指项集中包含项的个数。在抽取不包含重复项的前提下抽取的项集越长，则乘积近似值中包含的乘积项就越少。因此，在估计联合概率时，会考虑更多的属性依赖。

(3)选择尽可能覆盖待分类实例的项集集合

由于数据流是无限的，而算法使用的内存是有限的，因此当待分类实例到来时，算法不能遍历整个数据集来抽取项集。另一方面，由于数据流是动态的，数据的底层分布可能随时变化而存在概率漂移的情况，若发生概念漂移，则历史数据将不再适用于当前情况，因此近期数据的重要性要大于历史数据的重要性。使用最近的数据建立分类模型也保证了分类模型在一定程度上能避免概率漂移的发生，因为它受历史数据的干扰较小。这些情况表明，为满足数据流上数据挖掘的要求，可能存在抽取的项集集合不能够完全覆盖待分类实例的情况。本文使用 Laplace 平滑[6]来处理这种情况。

(4)项集的最小集合

最小是指集合中包含的元素(即项集)个数少。在尽可能抽取到足够多的项的情况下，本文希望抽取到的项集集合是最小集，即希望组成乘积近似值的乘积项尽可能地少，这样能最小化独立假设的数量，从而使得乘积近似值更加符合条件独立模型。

举例：$T=\{a_1,a_2,\cdots,a_5\}$ 是待分类实例，c_i 是任意类属性值。若混合树结构 HTS_i 中抽取到的项集集合为$\{\{a_1,a_2\},\{a_3\}\}$，则项集集合中包含的所有项不能够完全覆盖待分类实例中的所有项$\{a_1,a_2,a_3,a_4,a_5\}$。本文使用式(7)来估计联合概率 $P(T,c_i)$的值。

$$P(T,c_i)=P(a_1,a_2,\cdots,a_5,c_i)\approx P(c_i)P(a_1a_2|c_i)P(a_3|c_i)\cdot P(a_4a_5|c_i) \quad (7)$$

其中，函数 $attnum(A_4)$用于计算属性 A_4 中的属性值的个数，函数 $count(c_i)$用于记录当前窗口中类标为 c_i 的事务个数。使用 Laplace 平滑来估计未抽取到的项集的概率，同时将未被抽取的项合并为一个集合以减小项集集合的规模。$P(a_4a_5|c_i)=\dfrac{1}{count(c_i)+attnum(A_4)+attnum(A_5)}$。

4.2 PBDS 分类器训练阶段

PBDS 分类器训练阶段的主要工作是处理数据。这一阶段的主要任务是建立混合树结构，当有新的数据到来时更新滑动窗口和相应的混合树结构。

算法 1 描述了在训练阶段使用滑动窗口模型处理数据的完整过程。当数据流中的事务到来时，算法先判断窗口是否已满，若窗口未满，则将事务加入窗口，并根据事务的类标将事务加入相应混合树结构(第 4－5 行)；若窗口已满，则在窗口和相应的混合树结构中删除最旧事务的信息，再将新事务加入窗口和对应的混合子结构中(第 8－11 行)。

算法 1　Find frequent itemset algorithm on data stream (FFI)

输入：数据流 DS=[$T_1,T_2,\cdots,T_n,\cdots$)，滑动窗口大小 w

输出：混合树结构 HTS=$\{HTS_i\}$

1. for all T in DS
2. 　//W 是当前窗口中事务的个数
3. 　If W<w
4. W=W+1;
5. HTSBuilding(T,HTS_i);
6. 　end if
7. 　else
8. W=W−1;
9. deletefromHTS(T_{old},HTS_j);
10. W=W+1;
11. HTSBuilding(T,HTS_i);
12. 　end if
13. end for

算法 2 和算法 3 描述了当滑动窗口中事务发生变化时混合树结构的更新过程。

算法 2　建立混合树结构 HTSBuilding(T,HTS_i)

输入：事务 T 和 T 所对应的 HTS_i

输出：更新后的 HTS_i

1. for 对于事务 T 中的每一项 $x_i\in T/x_n$
2. //将 x_i 添加至 FI_list
3. addItemtoFI(x_i,FI_list);
4. for 对于$\{x_{i+1},x_{i+2},\cdots,x_{n-1}\}$中的每一项 x_j 添加到 x_i. SFI_list
5. addItemtoSFI(x_j,x_i. SFI_list);
6. 　end for
7. //将 x_i 的分支 Tbranch=$\{x_i\rightarrow x_{i+1}\rightarrow\cdots\rightarrow x_{n-1}\}$添加到 x_i. FIT

8. addTreeBranchtoFIT(Tbranch, x_i. FIT);
9. end for

算法 2 描述了在对应的混合树结构中添加新事务的过程。将事务中每一个项添加到混合子结构 FI_list 中(第 1—3 行),并将该项在事务中的后续项的集合添加到该项的 SFI_list 中(第 4—6 行)。该项连同其后续项组成树的分枝(*Tbranch*),并添加到该项的 FIT 中(第 7—8 行)。算法中,add-ItemtoFI(x_i,FI_list)即在 FI_list 中建立包含项 x_i 的结点;addItemtoSFI(x_j,x_i. SFI_list)在 x_i. SFI_list 中建立包含项 x_j 的结点;addTreeBranchtoFIT(*Tbranch*x_i. FIT)在 x_i. FIT 子树中添加分枝 *Tbranch*。

算法 3 更新混合树结构 deletefromHTS(T_{old},HTS_i)

输入:最先到达窗口的事务 T_{old} 和 T_{old} 所在分组的 HTS_i
输出:更新后的 HTS_i

1. for 对于事务 T 中的每一项 $x_i \in T_{old}/x_n$
2. deleteItemfromFI(x_i,FI_list);
3. for 对于$\{x_{i+1},x_{i+2},\cdots,x_{n-1}\}$中的每一项 x_j
4. deleteItemfromSFI(x_j,x_i. SFI_list);
5. end for
6. //对于 x_i 的分支 Tbranch=$\{x_i \rightarrow x_{i+1} \rightarrow \cdots \rightarrow x_{n-1}\}$
7. deleteTreeBranchfromFIT(Tbranch,x_i. FIT)
8. end for

算法 3 描述了在相应混合树子结构中删除旧事务的过程。对事务中的每一个项,在 HTS_i 的 FI_list 中删除包含该项的结点,并在该项的 SFI_list 中删除该项在事务中的后续项(第 2—5 行),然后在该项的 FIT 中删除该项和其后续项组成的分枝(第 6—7 行)。算法中,deleteItemformFI(x_i,FI_list)在 FI_list 中删除包含项 x_i 的结点,deleteItemformSFI(x_j,x_i. SFI_list)在 x_i. SFI_list 中删除包含项 x_j 的结点,deleteTreeBranchformFIT(*Tbranch*,x_i. FIT)在 x_i. FIT 子树中删除树的分枝 *Tbranch*。

4.3 类标预测

PBDS 分类器是一个半懒惰式的基于模式的数据流贝叶斯分类器。对于待分类实例 T_{test},在每一个类标值所对应的 HTS_i 中抽取项集,将每一次抽取到的最佳项集添加到 *finalItemset* 中。*finalItemset* 是用于最终计算概率的频繁项集的集合,其中的项集都相互独立,项集之间没有相同的项。算法 4 描述了 PBDS 分类器为待分类实例预测类标的过程。

算法 4 ClassifierClassPrediction(HTS,T_{test},*min_sup*)

输入:HTS=$\{HTS_i\}$,最小支持度阈值 min_sup 和测试实例 T_{test}
输出:待分类实例 T_{test} 的类标 c

1. for all c_i in C
2. finalItemset=∅;
3. while $T_{test} \neq$ null do
4. bestFpattern=selectBestItemset(T_{test},HTS_i,min_sup);
5. $T_{test}=T_{test}$/bestFpattern;
6. finalItemset=finalItemset∪bestFpattern;
7. end while
8. //L is the itemset in the set of finalItemset
9. $P(T,c_i)=P(c_i)\prod_{l \in finalItemset} P(l|c_i)$
10. end for
11. return the class c_i with maximal $P(T,c_i)$;

算法对待分类实例在每个类标下分别抽取项集集合(第 3—7 行),并根据抽取到的项集分别估计联合概率值(第 9 行),再通过比较各个联合概率值的大小来对待分类实例的类标进行预测(第 11 行)。

算法 5 描述了频繁项集的抽取过程。

算法 5 selectBestItemset(T_{test},HTS_i,*min_sup*)

输入:待分类实例 T_{test},混合树结构 HTS_i,最小支持度阈值 min_sup
输出:最佳频繁项集 BestFpattern

1. //定义一个新的变量来表示项;x_i in T_{test}/x_n
2. X=x_i;
3. If(null==findIteminFI(x_i,HTS_i))
4. X=x_{i+1};
5. else
6. projectItemset=$\{x_{j+1},x_{j+2},\cdots,x_{n-1}\}$
7. longpattern=findItemsetinSFI(projectItemset,X. SFI_list);
8. Bestpattern=checkPatterninFIT(longpattern,X. FIT);
9. BestFpattern=checkPatternSup(Bestpattern,min_sup)
10. Return BestFpattern;
11. End if

算法 5 对应于 3.2 节中在给定抽取项的范围下频繁模式的抽取机制,3.2 节中的内容即是对算法 5 的描述。算法中,findIteminFI(x_i,HTS_i)在 HTS_i 内的 FI_list 中寻找包含项 x_i 的结点,FindItemsetinSFI(*projectItemset*,*X*. SFI_list)在 *X*. SFI_list 中找到与 *projectItemset* 具有相同项的项集,checkPatterninFIT(*longpattern*,*X*. FIT)检查 *longpattern* 中的项在子树 *X*. FIT 中是否位于同一分枝,checkPatternSup(*Bestpattern*,*min_sup*)依据给定的最小支持度阈值修剪项集 *Bestpattern*。

5 实验分析

本文进行了大量实验,主要从分类正确率和运行时间两个方面对算法的性能进行评价,同时还研究了算法参数调整对分类器性能的影响。本文在真实数据集和合成数据集上进行实验。真实数据集是 5 个 UCI 机器学习库中的数据集,合成数据集是使用数据生成器中生成的数据集。表 2 列出了真实数据和合成数据的主要特征,属性 Attribute 不包含类属性。实验平台是 Massive OnlineAnalysis(moa)[4]。实验在 3.00GHz、Intel(R) Core(TM)2 Duo CPU、4G 内存、Windows7 系统的计算机上进行。

表 2 真实数据和合成数据

Dataset	Transaction	Attribute		Class label
		Continuous	Nominal	
agrawal	100000	5	3	2
randomRBF	100000	10	0	2
randomRBFdrift	100000	10	0	2
SEA	100000	3	0	2
STAGGER	100000	3	0	2
Chess	28056	0	6	18
Connect-4	67557	0	42	3
EEG	14980	14	0	2
Firm	10800	0	19	2
MAGIC	19020	10	0	2

表 2 中的数据集 Chess,Connect-4,EEG,Firm 和 MAG-

IC 来自 UCI 机器学习库。数据集 agrawal，random RBF，random RBF drift，SEA 和 STAGGER 分别是由数据生成器 Agrawal Generator[1]，Random RBF Generator，random RBF drift，SEA Generator[20] 和 STAGGER Generator[21] 生成的 100000 条事务组成的。

5.1 数据预处理

有连续属性值的数据不能够以项或者关联规则的形式用于分类器。为了增加可用数据，本文在 PBDS 分类器中增加对数据集的属性离散化操作，离散化采用的是最小化信息熵的启发式算法[8]。离散化代码来自 moa 平台。

表 3 列出了预处理之后数据的主要特征，属性 Attribute 不包含类属性，表中的项 Item 不包含类属性-值对。

表 3　预处理之后的数据集

Dataset	Transaction	Attribute	Item	Class label
agrawal	100000	4	49	2
randomRBF	100000	10	254	2
randomRBFdrift	100000	10	254	2
SEA	100000	3	23	2
STAGGER	100000	3	9	2
Chess	28056	6	41	18
Connect-4	67557	42	126	3
EEG	14980	14	82	2
Firm	10800	19	38	2
MAGIC	19020	10	79	2

5.2 实验模型

本文使用预测误差估计法（prequential error estimators）[10] 对算法性能进行评价。对于 PBDS 分类器，设置了两种参数配置，第一种是标准型，在标准型参数设置中将所有数据集的滑动窗口的大小 w 固定为 $w=10\%*n$，其中 n 是数据集的实例个数；将最小支持度阈值固定为 $min\_sup=0.01\%*w$，其中 w 是当前窗口的大小。第二种是调整型，由于算法在不同数据集上达到最优性能的参数设置是不同的，因此在调整型参数设置中对每一个数据集分别调整滑动窗口大小 $w=p*n$（其中 $p\in[10\%,90\%]$）和最小支持度阈值 $min\_sup=q*w$，使得算法在该数据集上达到最优性能。

5.3 分类精度

本文将 PBDS 分类器分别与数据流贝叶斯分类器、关联性分类器和懒惰式分类器进行比较；同时为了实验的全面性，将 PBDS 与其他分类器（非贝叶斯和非关联性的急切式分类器）进行比较。

表 4 列出了各个分类器的分类精度。表 4 中第 1 列描述了使用的数据集；第 2—12 列分别给出了分类器 NaiveBayes，NaiveBayes Multinomial[16]，k-NN[5]，k-NN-withPAW[5]，RuleClassifier[11] RuleClassifierNBayes[11]，HoeffdingTree[12]，HoeffdingOptionTree[19]，SGD，ORTO[23] 和 FIMTDD[13] 在各个数据集上的分类正确率；第 13 列给出了 PBDS 分类器在两种参数设置下的分类正确率。

表 4　分类精度

Dataset	NB	NBM	k-NN	PAW	Rule	Rule NB	HT	HOT	SGD	ORTO	FIMT-DD	PBDS	
												standard	tuned
agrawal	95.30	68.10	90.30	90.40	93.70	94.00	95.30	95.30	68.10	68.10	68.10	94.89	95.30
RBF	76.30	69.39	77.10	80.00	61.10	77.30	85.10	85.10	68.60	51.20	51.20	78.60	78.60
RBFdrift	76.30	69.39	77.10	80.00	61.10	77.30	85.10	85.10	68.60	51.20	51.20	78.60	78.60
SEA	87.70	66.00	79.10	80.80	83.60	87.00	89.20	89.20	76.40	34.00	34.00	89.20	89.20
STAGGER	100.00	92.70	100.00	100.00	100.00	100.00	100.00	100.00	88.50	88.50	88.50	100.00	100.00
Chess	5.00	2.10	87.70	73.10	—	40.60	22.30	22.30	—	—	—	14.70	54.40
Connect-4	48.10	53.40	66.50	65.40	53.40	64.50	61.50	66.60	54.70	53.40	53.40	59.09	59.09
EEG	83.60	82.30	91.20	91.10	90.60	90.40	79.60	86.30	90.70	90.70	90.70	86.10	90.70
Firm	99.70	99.80	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
MAGIC	47.50	68.50	100.00	100.00	100.00	100.00	100.00	100.00	100.00	—	—	100.00	100.00
Average	71.95	67.17	86.90	86.08	74.35	83.11	81.81	82.99	71.56	53.71	53.71	80.12	85.25

相较于其他算法，PBDS 算法的主要优势是：1）采用半懒惰式学习方式，针对待分类实例建立特定的局部分类模型；2）使用频繁模式，基于条件独立模型来估计联合概率的近似值；3）对待分类实例在不同的类标值上分别建立结构不同的乘积近似值。上述 3 点使得在分类模型建立的过程中能更好地获取数据中属性之间的依赖关系，进而提高分类正确率。

（1）与贝叶斯分类器进行比较

本文将 PBDS 算法与数据流贝叶斯分类器（NaiveBayes 和 NaiveBayes Multinomial[16]）在分类精度上进行比较。如表 4 所列，PBDS 分类器在分类正确率要总体优于 NaiveBayes Multinomial；PBDS 与 Naïve Bayes 相比在 randomRBF，randomRBFdrift，SEA，Chess，Connect-4，EEG，Firm 和 MAGIC 数据集上具有更高的分类精度，在 agrawal 和 STAGGER 数据集上二者持平。总的来说，PBDS 分类器在分类精度上优于其他数据流贝叶斯分类器。

（2）与基于实例的数据流分类器进行比较

将本文提出的算法与基于实例（懒惰式）的分类器 k-NN，k-NNwithPAW[5] 在分类精度上进行比较。由表 4 可知，尽管在分类正确率的均值上 PBDS 分类器略低于这两个分类器，但是在 SEA，STAGGER，Firm 和 MAGIC 数据集上其分类正确率与这两个分类器相比持平甚至略高。

（3）与关联性数据流分类器进行比较

将本文提出的算法与关联性分类器 RuleClassifier[11] 和 RuleClassifierNBayes[11] 在分类正确率上进行比较。由表 4 可知，PBDS 分类器在 agrawal，randomRBF，randomRBFdrift 和 EEG 数据集上的分类正确率优于这两个分类器，而在 STAGGER，Chess，Connect-4，Firm 和 MAGIC 数据集上的分类正确率与其持平。总的来说，PBDS 分类器优于关联性数据流分类器。

（4）与其他的数据流分类器进行比较

将 PBDS 算法与其他数据流分类器在分类正确率上进行比较，分类器包括 HoeffdingTree[12]，HoeffdingOptionTree[19]，ORTO[23]，SGD，ORTO 和 FIMTDD。由表 4 可知，与树形分类器 HoeffdingTree 和 HoeffdingOptionTree 相比，

PBDS 在数据集 agrawal，SEA，STAGGER，Firm 和 MAGIC 上的分类正确率较优。在参数设置为调整型时，PBDS 在数据集 Chess 和 Connect-4 具有较高的分类正确率。在参数设置为调整型时，PBDS 分类器的平均分类精度优于 HoeffdingTree 和 HoeffdingOptionTree 分类器。与 SGD，ORTO 和 FIMTDD 分类器相比，PBDS 在所选的 10 个数据集上都有较优的分类精度。

表 4 中 chess 数据相比于其他数据在标准型和调整型参数设置上的分类正确率差距较大。因为 chess 数据集在标准型参数设置中窗口的大小为 2805，而在调整型参数设置中窗口的大小为 14028。由表 3 可知，chess 数据集的类标个数（18 个）远多于其他数据集（以 2，3 个类标值居多）。由此可知，分类正确率相差较大的原因在于窗口大小的设置，当对测试实例进行预测时，当前窗口中各类标的事务数分布不均衡造成两种参数下分类准确度相差较大。

图 5 是 PBDS 与 NaiveBayes，HoeffdingTree，k-NN，RuleClassifier 和 RuleClassifierNBayes 在数据集 agrawal 上的分类精度比较。PBDS 采用标准型参数设置；采用的评价策略每在 10000 条数据时输出一条评价结果。

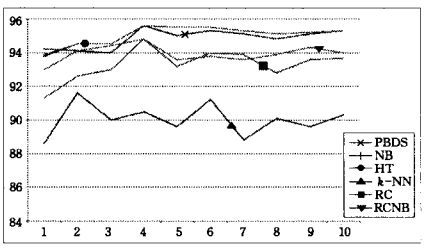

图 5 各分类器在 agrawal 数据集上的精度比较

5.4 运行时间

PBDS 是半懒惰式分类器，它具备懒惰式分类器的较高分类精度和能够处理动态复杂环境的优点；同时与懒惰式分类器相比，又有着较快的处理速度。

表 5 是 PBDS 分类器与懒惰式分类器（k-NN，k-NNwithPAW）在运行时间上的比较。

表 5 运行时间

Dataset	k-NN	k-NNwithPAW	PBDS
agrawal	18.4339	29.8866	3.7989
randomRBF	40.4763	59.0282	124.2752
randomRBFdrift	39.4282	59.1799	123.8207
SEA	11.3720	20.1007	1.5522
STAGGER	12.6606	22.0497	1.3974
Chess	7.8408	11.3098	3.4078
Connect-4	89.2032	123.4780	303.4664
EEG	7.1333	9.3697	4.4285
Firm	6.3339	8.9308	4.5657
MAGIC	7.0741	9.7144	2.6431

从表 5 中可以看出，在 agrawal，SEA，Chess，EEG，Firm 和 MAGIC 数据集上，PBDS 分类器相较于这两个分类器在运行时间上有着较大的优势，但 PBDS 分类器在 randomRBF，randomRBFdrift，Connect-4 数据集上运行时间较长。结合表 3 和表 4 可知，PBDS 运行效率会受数据集中所有项的个数的影响。数据集中所有项的个数越多，则 PBDS 分类器处理该数据集的时间越长。综上，PBDS 对所含项少的数据集在运行时间和分类精度上具有较高的性能。

结束语 本文提出了一个基于模式的数据流贝叶斯分类器 PBDS。PBDS 是一个半懒惰式分类器在训练阶段建立项集形式的数据的密集表示。当有分类请求时，PBDS 根据待分类实例选取项集集合，建立局部的分类模型。本文提出了单次扫描算法 FFI，使用滑动窗口模型用于在数据流上挖掘频繁项集；同时提出了混合树结构 HTS，用于存储在当前窗口中挖掘的项集。PBDS 在属性独立假设下抽取尽可能长的、没有重复的频繁项集集合，并且依据条件独立模型来建立联合概率的乘积近似值。这一方法在估计联合概率时考虑了属性之间的关联性，从而弱化了朴素贝叶斯的条件独立性假设。本文在实际数据和合成数据上进行了大量的实验，实验结果表明，PBDS 比其他分类器有着更高的分类性能。

参 考 文 献

[1] AGRAWAL R，SWAMI A. Database mining：A performance perspective [J]. IEEE Transactions on Knowledge and Data Engineering，1993，5(6)：914-925.

[2] BARALIS E，CAGLIERO L，GARZA P. Enbay：A novel pattern-based Bayesian classifier [J]. IEEE Transaction on Knowledge and Data Engineering，2013，25(12)：2780-2795.

[3] BARALIS E，CAGLIERO L. RIB：A robust itemset-based Bayesian approach to classification [J]. Knowledge-Based Systems，2014，71：366-375.

[4] BIFET A，HOLMES G，KIRKBY R，et al. MOA：Massive online analysis[J]. Journal of Machine Learning Research，2010，99：1601-1604.

[5] BIFET A，READ J，PFAHRINGER B，et al. Efficient data stream classification via probabilistic adaptive windows[C]// Proceedings of the 28th Annual ACM Symposium on Applied Computing. New York：ACM，2013：801-806.

[6] DOMINGOS P，PAZZANI M. On the optimality of the simple Bayesian classifier under zero-one Loss [J]. Machine Learning，1997，29(2)：103-130.

[7] FAN H J，RAMAMOHANARAO K. A Bayesian approach to use emerging patterns for classification[C]// Proceedings of the 14th Australasian Database Conference. Darlinghurst，Australia：Australian Computer Society，2003：39-48.

[8] FAYYAD U M，IRANI K B. Multi-interval discretization of continuous-valued attributes for classification learning[C]// Proceedings of the 13th Int'l Joint Conf. Artificial Intelligence. 1993：1022-1027.

[9] FRIEDMAN N，GOLDSZMIDT M. Building classifiers using Bayesian networks[C]// Proceedings of The Thirteenth National Conference on Artificial Intelligence. 1996：1277-1284.

[10] GAMA J，SEBASTIAO R，RODRIGUES P P. Issues in evaluation of stream learning algorithms[C]//Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York：ACM，2009：329-338.

[11] GAMA J，KOSINA P. Learning decision rules from data streams [C]// Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence. 2010：1255-1260.

（下转第 202 页）

Reports,2009,486(3-5):75-174.

[2] XU W,LIN B G,LIN S J,et al. Research on Community Detection Method for Social Networks Based on User Interaction and Similarity[J]. Netinfo Security,2015(7):77-83. (in Chinese)
许为,林柏钢,林思娟,等. 一种基于用户交互行为和相似度的社交网络社区现方法研究[J]. 信息网络安全,2015(7):77-83.

[3] TANG J,WANG X,LIU H. Integrating social media data for community detection[C]// Proceedings of the 2011 International Conference on Modeling and Mining Ubiquitous Social Media. Springer-Verlag,2011:1-20.

[4] TANG L,LIU H. Scalable Learning of Collective Behavior Based on Sparse Social Dimensions[C]// Proceeding of Acm Conference on Information & Knowledge Management (Cikm 09). 2009:1107-1116.

[5] GERGELY P,IMRE D,ILLÉS F,et al. Uncovering the overlapping community structure of complex networks in nature and society[J]. Nature,2005,435(7043):814-818.

[6] GIRVAN M,NEWMAN M E. Community structure in social and biological networks[J]. Proceedings of the National Academy of Sciences of the United States of America,2001,99(12):7821-7826.

[7] DEV H. A user interaction based community detection algorithm for online social networks[C]// ACM Sigmod International Conference on Management of Data. ACM,2014:1607-1608.

[8] LEE D D,SEUNG H S. Learning the parts of objects by non-negative matrix factorization[J]. Nature,1999,401(6755):788-791.

[9] WANG H,NIE F,HUANG H,et al. Nonnegative Matrix Tri-factorization Based High-Order Co-clustering and Its Fast Implementation[C]// 2011 11th IEEE International Conference on Data Mining. IEEE Computer Society,2011:774-783.

[10] CHANG Z C,CHEN H C,LIU Y,et al. Community detection based on joint matrix factorization in networks with node attributes[J]. Acta Physica Sinica,2015,64(21):0218901. (in Chinese)
常振超,陈鸿昶,刘阳,等. 基于联合矩阵分解的节点多属性网络社团检测[J]. 物理学报,2015,64(21):0218901.

[11] MAES C M. A regularization active-set method for sparse convex quadratic programming[M]. Stanford University,2010.

[12] WEBER M,RUNGSARITYOTIN W,SCHLIEP A. Perron Cluster Analysis and Its Connection to Graph Partitioning for Noisy Data:IB-Report 04-39[R]. 2004.

[13] JIN D,GABRYS B,DANG J. Combined node and link partitions method for finding overlapping communities in complex networks[J]. Scientific Reports,2015(5):08600.

[14] RUAN Y,FUHRY D,PARTHASARATHY S. Efficient Community Detection in Large Networks using Content and Links [C]// Proceedings of the 22nd international conference on World Wide Web. 2012:1089-1098.

[15] YANG J,LESKOVEC J. Defining and Evaluating Network Communities Based on Ground-Truth[J]. Knowledge and Information Systems,2012,42(1):745-754.

[16] KANUNGO T,MOUNT D M,NETANYAHU N S,et al. An Efficient k-Means Clustering Algorithm: Analysis and Implementation[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence,2002,24(7):881-892.

(上接第 174 页)

[12] HULTEN G,SPENCER L,DOMINGOS P. Mining time-changing data streams[C]// Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York:ACM,2001:97-106.

[13] IKONOMOVSKA E,GAMA J,DŽEROSKI S. Learning model trees from evolving data streams[J]. Data Mining and Knowledge Discovery,2011,23(1):128-168.

[14] KEOGH E J,PAZZANI M J. Learning augmented Bayesian classifiers:A comparison of distribution-based and classification-based approaches[C]// Proceedings of the 7th Int'l Workshop on Artificial Intelligence and Statistics. San Francisco:Morgan Kaufmann Publishers,1999:225-230.

[15] LEWIS D D. Naive(Bayes) at forty:The independence assumption in information retrieval[J]. Machine Learning:ECML-98,1998,1398:4-15.

[16] MCCALLUM A,NIGAM K. A comparison of event models for Naive Bayes text classification[C]// Proceedings of AAAI-98 Workshop on 'Learning for Text Categorization',1998.

[17] LI H F,SHAN M K,LEE S Y. DSM-FI:An efficient algorithm for mining frequent itemsets in data streams[J]. Knowledge and Information Systems,2008,17(1):79-97.

[18] MERETAKIS D,WIJTHRICH B. Extending Naive Bayes classifiers using long itemsets[C]// Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. New York:ACM,2014:165-174.

[19] PFAHRINGER B,HOLMES G,KIRKBY R. New options for hoeffding trees[C]// Procee-dings of AI 2007:Advances in Artificial Intelligence. Heidelberg,Berlin:Springer-Verlag,2007:90-99.

[20] STREET W N,KIM Y S. A streaming ensemble algorithm (SEA) for large-scale classification[C]// Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM,New York,2001:377-382.

[21] SCHLIMMER J C,GRANGER R H. Incremental learning from noisy data[J]. Machine Learning,1986,1(3):317-354.

[22] ZHANG P,GAO B J,ZHU X Q,et al. Enabling fast lazy learning for data streams[C]// Proceedings of 2011 IEEE 11th International Conference on Data Mining. New Jersey, USA:IEEE,2011:933-941.

[23] ZLIOBAITE I,BIFET A,PFAHRINGER B,et al. Active learning with evolving streaming data[J]. Lecture Note in Computer Science,2011,6913:597-612.