# 基于类集成的方式检测渐进式概念进化

王鼎晨[1]

**摘要**：在数据挖掘这一领域中，数据流挖掘是一项重要研究课题，不同于传统数据利用数据建立一个分类预测模型，对新的数据进行分类预测。数据流具有大规模，多样性，高速性等特点，对数据流的挖掘提出了更高的要求。在真实的数据流环境中，数据的分布是会随着时间的改变逐渐变化的，即发生了概念漂移，对于发生了概念漂移的数据流能否第一时间检测到，并更新分类模型，对数据预测分类的准确性有着重大影响。诱导发生概念漂移的一个重要因素就是发生了概念漂移，在概念漂移中，数据的分类倾向逐渐改变。对发生了概念进化的检测越及时，就可以更新分类模型，得到的数据结果也就越准确。为了能够检测概念进化，首先对数据流采用Kernelized Logistic Regression进行分类学习，对学习过程中分类的概念创建对应的分类模型Class-based Model，输出该概念的先验概率，在之后的学习过程中不断更行Class-based Model 和先验概率，通过检测Class-based Model 追踪概念进化的发生，及时更新学习器模型。获取准确的预测模型。

**关键词**：数据挖掘,概念漂移,概念进化,类集成,数据流

**Abstract：**In the field of data mining, data stream mining is an important research topic. Different from the traditional data utilization data, a classification prediction model is established to classify new data. The data stream has the characteristics of large-scale, diversity, high-speed, and so on, which puts forward higher requirements for data stream mining. In the real data flow environment, the distribution of data will gradually change with the change of time, that is, the concept drift occurs, whether the data flow that has undergone the concept drift can be detected at the first time, and update the classification model. The accuracy of data prediction classification has a major impact. An important factor in inducing the concept drift is the concept drift. In the concept drift, the classification tendency of data gradually changes. The more timely the detection of a conceptual evolution has taken place, the more quickly the classification model can be updated and the more accurate the resulting data. In order to detect the evolution of concepts, the data stream is first classified using the Kernelized Logistic Regression, and the corresponding classification model is created for the concept of classification in the learning process. The prior probability of the concept is output, and the prior probability is continuously The Class-based Model and Prior Probability were updated to update the learner model by detecting the evolution of the concept in the Class-based Model. Get an accurate prediction model.

**Keywords**:data mining,concept drift, class evolution, ensemble model,data stream

目录

[基于类集成的方式检测渐进式概念进化 1](#_Toc515370014)

[1.数据流挖掘介绍 3](#_Toc515370015)

[1.1.概念进化的研究现状 3](#_Toc515370016)

[2 概念进化问题描述 5](#_Toc515370017)

[2.1概念漂移和概念进化 5](#_Toc515370018)

[2.2 概念进化的检测算法 5](#_Toc515370019)

[2.3 发生概念进化的数据处理 6](#_Toc515370020)

[3 数据流分类 6](#_Toc515370021)

[3.1 决策树分类模型 6](#_Toc515370022)

[3.2 集成分类模型 6](#_Toc515370023)

[3.3数据流聚类 7](#_Toc515370024)

[4 理论过程 8](#_Toc515370025)

[4.2 Class-Based Model 介绍 9](#_Toc515370026)

[4.3 构建CBModel的基学习器 9](#_Toc515370027)

[4.4 概念进化检测 9](#_Toc515370028)

[4.5 概念检测过程 10](#_Toc515370029)

[5实际实验 11](#_Toc515370030)

[5.1 数据集准备 11](#_Toc515370031)

[5.2 实验内容 11](#_Toc515370032)

[5.3 实验数据训练结果 11](#_Toc515370033)

[5.4 真实数据训练结果 12](#_Toc515370034)

[6结论 13](#_Toc515370035)

[7进一步的研究方向 13](#_Toc515370036)

[参考文献 13](#_Toc515370037)

[附录 15](#_Toc515370038)

[致谢 20](#_Toc515370039)

1.数据流挖掘介绍

近年来数据流作为一种新型数据模型广泛出现在多种应用领域。与传统的数据集不同，数据流具有高速性、有序性、大规模和潜在无限等特性。数据流目前主要产生于网络环境，如Web网站产生的点击流量分析、网络服务产生的日志信息、交通流量的监控与管理、电力供应的管理与预测、电信数据、金融服务以及商业交易管理和分析等。专用网络同样会产生大量的数据流，如基于卫星的高分辨率测量地球测地学数据、雷达衍生的气象数据、连续的大型天文光学、调查红外线和无线电波长及大气辐射测量等。

数据流挖掘已经成为了一项新兴的智能信息处理技术。引起过内外学者的广泛研究。数据流模型不同于传统的数据库，它具有一定的约束：

(1) 数据流具有无限性，即包含的数据个数是无限的。但是存储受到限制，只能存储概要信息，其余信息在存储时被丢弃。

(2) 数据到达速度快,需要实时处理，处理后即被丢弃。

(3) 数据的分布会随着时间改变，因此历史数据可能无关甚至有害。

1.1.概念进化的研究现状

数据流的特点中观测到数据的潜在分布会随着时间改变。因此处理数据流时，数据流分类器应能适应概念改变的信息，并快速调整分类模型以适应概念的变化。研究中出现了很多处理数据流分类中的概念漂移问题的方式，包括使用滑动窗口 和实例权重检测概念改变点,监控两个不同时间窗口内分布[2] 等。如 Gama 提出基于错误率的概念检测分类方法[5] ; Baena 提出基于分类错误距离的概念检测方法[6] ;Gama 提出一种两层学习系统来解决周期性概念问题[2] 等。

经过研究者对于概念漂移问题的多年研究探索，处理数据流概念漂移的技术主要分为以下几类：

1) 自适应基学习器；

2) 基于修改训练集的学习器；

3) 集成技术；

4) 基于原型学习的方法.

自适应基学习器是解决概念漂移问题的理论上的简单方法.快速决策树算法[[5](http://html.rhhz.net/buptjournal/html/20170305.htm#b5)](VFDT，very fast decision tree)是最早处理数据流的决策树拓展算法，基于小样本足以选择最优的分裂属性这一事实，使用Hoeffding边界量化叶节点中确定最优分裂属性所需要的样本个数。自适应概念快速决策树算法[[6](http://html.rhhz.net/buptjournal/html/20170305.htm#b6)](CVFDT, concept-adapting VFDT)在缓存中流动存储固定量级滑动窗口大小的数据流实例进行Hoeffding边界的再次计算；霍夫丁自适应树[[7](http://html.rhhz.net/buptjournal/html/20170305.htm#b7)](HAT, hoeffding adaptive tree)在每个有间隔的树节点上使用自适应窗，从而允许更快、更准确地适应概念漂移；霍夫丁窗口树[[7](http://html.rhhz.net/buptjournal/html/20170305.htm#b7)](HWT, hoeffding window tree)与CVFDT相比，无须等待固定数量的流数据，在计算Hoeffding边界更新方面更加快速，并且减少了人为设定的参数；有效自适应概念快速决策树算法[[8](http://html.rhhz.net/buptjournal/html/20170305.htm#b8)] (E-CVFDT, efficient concept-adapting very fast decision tree)引入了缓存机制，将相似特征的缓存实例分批计算信息增益，而非通过CVFDT序列，改善了CVFDT的效率。以上算法都是基于一定的窗口处理方法用最新的数据对Hoeffding边界进行动态地重新计算或更新，从而判断概念漂移是否产生以及决策树是否重构。在标准决策树的基础上，Buntine等[[9](http://html.rhhz.net/buptjournal/html/20170305.htm#b9)]提出了选项树作为决策树的扩展。在选项树中，新添加了选项节点，路径可能会在多重选项节点处产生分解。相继结合了Hoeffding树与选项树的概念，提出了霍夫丁选项树算法[[10](http://html.rhhz.net/buptjournal/html/20170305.htm#b10)](HOTs，Hoeffding option trees)，当新数据中产生了更好的分裂节点时，作为选项节点加入，2个节点都进行保留。在此基础上，Bifet等[[11](http://html.rhhz.net/buptjournal/html/20170305.htm#b11)]将HOTs算法拓展为自适应霍夫丁选项树算法(AHOTs，adaptive Hoeffding option trees)，每个叶节点都赋予一个指数加权的移动平均估计值。

第2类广泛用于解决概念漂移的技术是修正分类算法的训练集。普遍采用的方法有窗口技术以及实例加权。窗口技术主要通过一定方法来选取一定数量的最新数据代表当下概念。主要的算法有FLORA[[12](http://html.rhhz.net/buptjournal/html/20170305.htm#b12)]与ADWIN[[13](http://html.rhhz.net/buptjournal/html/20170305.htm#b13)]。 FLORA算法利用析取范式(DNF)来表示窗口中的正反例，并且根据分类性能对窗口尺寸进行扩大或压缩。作为FLORA的拓展，FLORA3[[14](http://html.rhhz.net/buptjournal/html/20170305.htm#b14)]算法引入了自适应窗口概念，尝试自适应地改变窗口的大小.。ADWIN算法通过统计方法比较当下窗口中2个“足够大”的子窗口，如果两者“足够不同”，则表明产生概念漂移。而ADWIN2[[13](http://html.rhhz.net/buptjournal/html/20170305.htm#b13)]算法，则是引入了一个内存相对高效的数据结构对滑动窗口进行存储。实例加权技术适用于实例的相对重要性有益于分类的场景。典型算法是自适应加权KNN[[15](http://html.rhhz.net/buptjournal/html/20170305.htm#b15)]算法，主要原理是基于与当前概念的相似程度，对所有实例进行加权处理，当概念持续漂移时，对旧数据分配较小权值，从而弱化对新概念的价值与影响。

第3类方法是结合多个弱分类器，每个分类器训练一部分数据，对后续的实例能够获得较准确的拥有较好鲁棒性的分类器模型。流集成算法[[16](http://html.rhhz.net/buptjournal/html/20170305.htm#b16)](SEA)主要思路是将数据流分解为连续的、不重复的窗口，用每个窗口内的数据训练一个弱分类器，采用投票机制进行预测与分类。如果弱分类器个数达到上限，则删除“最差”的分类器。另一个典型的集成算法是动态加权的DWM[[17](http://html.rhhz.net/buptjournal/html/20170305.htm#b17)]算法，主要思想是通过对每个弱分类器根据分类性能进行动态加权来处理概念漂移问题。自适应Boosting集成算法[[18](http://html.rhhz.net/buptjournal/html/20170305.htm#b18)](ABE, adaptive boosting ensemble)使用限制深度的决策树来开发Boosting的性能特点，检测到概念漂移后，丢弃当前模型进行重构新的模型。随着集成技术的发展，提出将基于分块的集成技术与在线增量集成方法两种方法结合的思想，实现集成技术的进一步完善。代表性算法有精度更新集成算法[[19](http://html.rhhz.net/buptjournal/html/20170305.htm#b19)](AUE2，accuracy updated ensemble)以及在线精度更新集成算法[[20](http://html.rhhz.net/buptjournal/html/20170305.htm#b20)](OAUE，online accuracy updated ensemble)。其中AUE2结合了基于分块集成技术的基于准确度的加权机制以及霍夫丁树的增量本质。OAUE算法使用加权投票机制来计算并保存各成员分类器的权值，并且实现增量更新。

以上是处理概念漂移的较为经典的3类数据流分类方法，但是性能方面都存在相应的问题：

1) 自适应基学习器仅对某个算法进行了概念漂移的适应，受数据限制，普适性差；

2) 基于修改训练集的学习器与分类器无关，普适性强，但是很难界定窗口大小以及加权方法；

3) 集成技术可以有效处理再产生概念，但是不能及时处理突然概念漂移。

随着概念漂移问题引起了研究者越来越多的关注，新的技术也在不断发展，SyncStream[[4](http://html.rhhz.net/buptjournal/html/20170305.htm#b4)]是新技术原型学习的代表。其研究核心在于选取并更新能够有效代表当前概念的原型。此算法主要通过动态保持一系列原型进行概念趋势捕获。算法中的原型主要通过错误驱动表示学习以及同步启发式约束聚类方法共同构建实现。运用主成分分析方法和基于统计的启发式方法进行突然发生的概念漂移检测。作为现有代表算法，SyncStream能够动态构建并更新原型代表最新的数据分布。通过与经典权威算法的一系列实验，证明基于原型学习的方法更能有效处理数据流概念漂移问题。但是，在处理概念漂移方面，SyncStream在时间复杂度、概念漂移响应速度以及分类准确性等方面存在很多问题亟待改进优化。

2 概念进化问题描述

2.1概念漂移和概念进化

数据流的概念分布会随着时间动态的发生变化，这种情况被认为数据随时间发生了概念漂移，而对于数据流挖掘算法来说，能够识别出这一变化，并且能够快速调整自身的分类模型，能够适应这种改变的算法就能够更加精确的预测和建立数据模型。

对于概念漂移来说，给定预测概念y，和条件变量x，在某一时间存在x，使得即发生了概念漂移。

而概念进化则是另一种诱发概念漂移的因素，概念进化是某一确切的概念的先验概率发生改变导致概念漂移的产生。即给定预测概念y，在某一时刻即发生了概念进化，通常表现是出现了新的概念。或者是某一存在的概念消失。

2.2 概念进化的检测算法

概念进化的检测一般是通过检测和识别变化发生的时间点，对发生变化的时间间隔来进行标识和量化发生的改变，常用的概念进化检测方法有：1.基于序列的分析方法，2.基于控制图的分析方法，3.分析两个分布之间的不同的方法.

PH方法是一种基于序列分析的检测方法，这种方法可以检测模型建立过程中正常行为到异常行为的改变。PH方法主要用于检测高斯信号平均值的突变；

SPC方法是一种基于统计过程的检测方式，是一种基于控制图的检测概念进化的方式，SPC检测整个学习过程的演变，对于发生概念漂移的学习过程中可以检测出三种状态，一种检测出演变过程发生了不可控的变化，即发生了概念漂移，一种是检测演变过程依旧可控，即没有发生概念漂移，还有一种的警告状态,即学习过程中发生了变化，但不足以决定该学习结果发生了概念进化。

现在最常用的检测方式应该就是基于检测两个分布之间不同的方法进行概念进化检测方式了，AdWIN就是使用这一种检测方式的算法，AdWIN可以很好的捕获流平均数。该算法保存一个可变长度的实例，该实例大小“足够大”，使得该实例不发生概念进化，对于每一次的数据演变对比默认实例，如果通过对比两个实例窗口，发现足够明显的实例变化就可以检测是否发生概念漂移了。

2.3 发生概念进化的数据处理

如何处理数据发生改变，即在生成属性的过程中，哪些数据被分类为某一类属性时？引入一个变量：EGR即类生成率。在每一次对数据进行分类时，统计被分类为某一属性的数据占总数据的百分比。这样对于原始数据中概念分类过程中属性变化就可以被直观的描述出来，对于原始数据出现新的属性时，那么该属性的EGR在生成过程中应该由0渐进接近1的过程。同样对于消失的属性，EGR应该是由1渐进变为0的过程。

3 数据流分类

数据流分类模型主要分为单分类模型和集成分类模型。单分类模型分类方式维护和更新一个分类模型，一旦发生概念漂移，分类模型就会随之更新，可以很好的适应概念漂移，但是对于更新模型这一操作会变得非常复杂和困难。集成分类则是可以通过对某一些部分进行剪枝或者整合某些部分实现快速适应概念漂移的模型，因此集成分类模型要比单分类模型的训练速度要快的多。数据流具有大规模，快速等特征。因此对数据流的分类模型一般采用增量更新方法，即按照数据一个接一个或者一批接一批的处理数据，每一个或者每一批处理之后更新模型。

3.1 决策树分类模型

决策树模型广泛用于创建分类器处理数据,目前最常用的决策树分类模型是Hoeffding的决策树算法，VFDT则是基于Hoeffding不等式针对数据流挖掘环境建立分类决策树的方法。他通过不断的将叶节点替换为分支节点而生成，在每一个决策点保留一个重要的决策量，当该节点的统计量达到一定的阈值是进行分裂测试，该算法只需要对数据流扫描一次，拥有很好的时间复杂度。

3.2 集成分类模型

集成学习使得分类器具有更高精度的分类方法，能够很好的适应概念的变化。常用的集成方式有Boosting和Bagging方法。Bagging 方式是将原始数据集通过T次随机采样，从而得到T个与原始数据集大小相同的子数据集，再分别训练得到T个单分类器，然后通过一些结合方式(如投票)，结合为一个集成分类器。Boosting 也是通过重采样得到多个单分类器，最后得到一个集成分类器。区别在于 Bagging 是基于权值的分类器集成。

Bifet[31] 提出了一种基于 ADWIN 算法对多个单分类器投票而得到的集成分类器方式，用于处理概念漂移数据流。数据随着时间发生变化，分类器自动调整。 Grossi[32] 通过对每个子分类方法的权重进行衡量，生成集成分类方法。由于没有处理过期事务的方法得到的权重较大，因此每个权重起到了处理概念漂移的关键作用。Farid[33]提出了自适应集成方法来进行概念漂移数据流的分类和新类检测等。Brzezinski[34]提出了一种基于数据块的在线处理概念漂移数据流的集成分类方法。它将数据流划分成多个数据块，每个块训练的分类器具有不同的权重，然后对块的特征进行分析来产生新类型的集成分类器。Czamowski[35]同样对数据块进行分块处理，每块生成基分类器，而后赋予不同权重进行分类器集成。 Ikono movska[36]采用了在线权重集成方法和在线随机森林方法来设计集成分类方法，其可用于分类和回归。Abdallah[37]设计了一种可变的轻量框架用于移动用户行为分类。这种框架使用一种集成分类模型，它对每个窗口内的预测值进行投票，从而得到最终的分类结果。Hosseini[38]提出一种半监督学习的集成分类方法来处理可变数据流。设定缓冲池存储多个分类算法，每个分类器处理独立的概念。每个分类器赋予不同的权重，集成方法根据权重函数最终给出分类结果。 ZareMoodi[39]将集成分类方法用于发现数据流中出现的新类。其采用局部集成方式，每个局部集成分类器都是针对某一个类值的。

3.3数据流聚类

聚类问题可以认为是将数据集分成若干个相似对象构成的子集的过程，这里的子集称为簇或类别。聚类的结果是使簇内的对象尽量相似，同时与其它簇的对象尽量不同。由于聚类时的数据对象没有类别标签，因此聚类是无监督的学习过程。聚类分析是重要的和基本的数据挖掘方法。它可以单独用来分析获取数据的分布情况，将数据分成不同的簇，观察每一个簇的特性，然后对特定的目标簇做进一步的处理。聚类还可以作为其它数据处理方法的预处理技术，比如产生类别标签给分类提供支持，提取特征以支持相关分析、频繁项挖掘、预测异常值检测等。研究者针对数据流提出了许多聚类算法。如算法Clustream[41]是采用界标窗口，并以应用中心请求为导向的数据流聚类方法。它把聚类过程分成在线部分和离线部分。在线部分周期地存储概要统计信息，离线部分仅使用这些概要信息。分析者利用离线部分的各种输入来快速理解数据流中的广泛簇群。如何有效选择、存储和使用这些统计数据是难题。为了解决这个问题，该算法设计使用椎体时间框架(Pyramidal Time Frame)并结合微聚类方法。Clustream算法是基于k-means的，不能很好地发现任意形状的簇，且不能处理异常，需要先验知识和用户定义的时间窗口。DGClust[42]用于处理全网络产生的数据流的聚类问题。它是一种分布式算法，可以减少维度和通信负担，允许每个本地传感器保持其数据流的在线离散化。DGClust使用网格存储数据，在界标窗口中处理最新数据，它也是基于k-means的。D-Stream[43]算法能解决基于k-means算法的不足。它是一种基于密度的数据流聚类算法，分为在线和离线两部分，在线部分将每个输入数据映射到一个网格，而离线部分计算网格密度并依据密度对网格聚类。D-Stream使用密度衰减技术来捕获数据流中的动态改变。该算法利用衰减因子、数据密度和簇结构之间的复杂关联可以实时生成和调整簇。ClusTree[44]采用衰减窗口自动适应数据流速度，是无参数的单遍扫描数据流的聚类方法。它总是维持最新的聚类模型，并且报告概念漂移以及新的和异常的值。该算法不是采用先验知识假设聚类模型尺寸，而是自适应调整的。StreamKM++[45]是采用欧氏距离处理数据流聚类问题的算法。它使用点树存储数据信息，也是一种基于k-means的方法。该算法使用界标窗口处理最新的数据。StreamKM++算法对数据流的小权重抽样数据进行处理，为此提出了两种新的结构。首先使用自适应的非均匀采样方法从数据流中获得一些小的点。其次，该算法设计数据结构树用于加快自适应的时间和非均匀采用点的过程。

4 理论过程

随着科学技术的发展，网络对于生活的影响也越来越大，网络活动产生的数据不同于传统的数据模型，数据流作为一种新的数据模型产生于各种网络活动中：网站点击，朋友圈分享动态，移动电信产生的流量等。同样的，对于数据流的挖掘成为一项新的智能信息处理技术，引起广泛关注。

不同于传统的数据库信息，数据流信息具有一些特别的属性：数据流的概念分布会随着时间发生改变。这给数据流的数据挖掘带来了新的挑战。

对于数据， 表示在t时刻产生的数据， 表示在t 时刻，由 进行分类生成的属性（标签，概念）。对于这样的数据对应处理方式，在进行概念进化检测时，就是检测原始数据 的数据发生了改变的过程，即，发生概念进化就是原始数据分类出现了新的属性或者某些属性在分类过程中消失了。本文主要针对于渐进式的概念进化，对于渐进式的概念进化，数据的改变也是渐进的。

概念进化导致数据流变得动态不平衡，每个分类的先验概率则会剧烈波动，数据集中的数据趋向于被分为主要类别，而少数类则难以被识别，而下次发生概念进化时，可能从这些少数数据中发生，但是在处理之前的概念进化时对少数类的学习导致该少数类很有可能被完全忽略，为了减少这种影响，为类添加权重： ,计算过程为： ；处理后计算被分类为标签的概率为：

（1）

表示数据集被分类为的先验概率

为了最大化学习效率，对分类概率进行最小化处理

（2）

继续对公式进行简化，在t时刻，所有的分类标签的对的先验概率都相同，公式2等同于

（3）

(4)

对所有单位同时除以

(5)

最小化问题转变为计算最大的先验概率问题

(6)

4.2 Class-Based Model 介绍

Class-Based Model（CBModel） 是由数据集正确分类的数据的后验概率组成的集合，即，对t时刻的数据进行集成分类，被分类为 的标签，对于的类标记为+1，对于其他的类 标记为-1，根据贝叶斯理论，的后验概率概率为

对于数据流处理的得出的+1和-1的数量往往相差极大，而且通常-1的数量远远大于+1，所以需要对-1的数据进行欠采样，计算采样概率：

(9)

代表t时刻，数据集的先验概率，计算的公式为：

(10)

是学习过程中的衰减因子，手动赋予，随着实验不断调整衰减因子以达到最好的学习效果。

4.3 构建CBModel的基学习器

使用Kernelized Logistic Regression作为基学习器：

(11)

是中被用来作为i训练数据的样本数，

是第j项的系数

t 是时间t

是基学习器的核函数

的后验概率计算方式为：

(12)

通过多次的学习过程，CBModel可以用来在线性时间复杂度情况下预测分类的结果。

4.4 概念进化检测

概念进化有3种基本特征：1，新类的出现，2旧类的消失，3类消失又出现，

在概念进化检测算法里，数据流随时间不断加载进学习器里，首先估计数据的先验概率，在初始化对应的CBModel，对于第一第二个数据(),默认将他们的样本数量作为样本分类满足 被标记为-1的数量，所以，()计算先验概率公式为

(13)

ep:example size 表示样本数量

根据前两个数据分类来初始化创建CBModel，用来对随后加载的数据进行分类。

在随后的数据分类中，如果该分类没有对应的CBModel，则说明出现了新的类，第一种新类出现的情况被检测到了。

对于第二种情况：旧类的消失，在类集合（）中没有办法辨别出已经消失的类， 保存着从第一个数据开始的所有分类出的类标签，为了检测类的消失，

在分类过程中给定一个阈值，若低于该阀值，则认为该类消失。阀值制定标准为：在分类过程数据集在t时间序列上一直未被分类为某一固定的类超过100个时间序列，就认为该类消失。

对于第三种情况：类的消失又出现，在本论文的检测算法中，对于已有的类总有一个CBModel与之对应，在每次分类过程中，都会相应的创建或者更新该类的CBModel，而当该类在第二种情况下消失后，并不会将这个类的CBModel删除，而是标记为灭活状态，而当重新出现了该类也就重新创建了一个CBModel，对应也就可以对比已经灭活的CBModel就可得知该类又重新出现了。

4.5 概念检测过程

1.对数据集进行集成分类，产生类集合，，

2.将t-1时刻的类集合复制给当前t时刻的类集合

3.判断是否有与之对应的，没有则出现了情况1的概念出现，更新类集合=，如果是第一第二个数据，则创建该数据的先验概率和该数据类的

4.如果有与之对应的并且=0，这种条件下出现情况3概念消失之后该概念又再次出现，即出现了概念消失后又出现的概念进化情况，更新类集合=

5.检测中的每个类如果有满足某个类已经满足类消失的阀值，则更新类集合=，=0，灭活该类的

6．使用，，更新每一个已存在的CBModel

7. 如果有与之对应的，更新先验概率，使用+1标记的类更新CBModel

8. 如果没有对应的，更新先验概率，使用-1标记的类更新CBModel

5实际实验

5.1 数据集准备

本论文准备了一组测试实验数据，和一份真实数据，实验数据分别是有一类标签没有发生概念进化的，有一类标签发生概念出现的数据，有一类标签发生概念消失的数据，有一类标签发生概念消失又出现的数据，真实数据来自 UDI TwitterCrawl Dataset[42]。数据包含了五千万条从2008年到2011年twitter用户发布的话题数据。该数据拥有的hashtag可以用来作为该条数据的标签，同时又天然拥有时间戳，正好用来作为流数据处理。

5.2 实验内容

首先构建Kernelized Logistic Regression基学习器，使用使用Gauss Kernel作为基学习器的核函数：

(14)

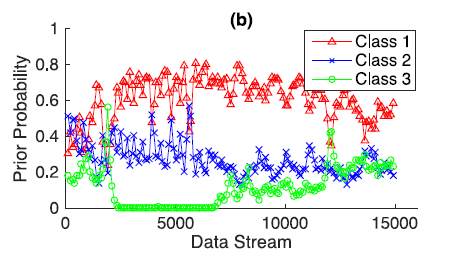
5.3 实验数据训练结果

由于实验数据有一类标签没有发生概念进化，也就不存在类的出现和消失，那么对于该类的类生成概率（EGR）一直是1。有一类标签发生概念进化，该类的类生成概率从0逐渐变为1，一类标签发生概念消失，该类的类生成概率从1逐渐变为0，一类标签发生概念出现消失，该类的类生成概率从0逐渐变为1又从1逐渐变为0。

在实验数据中，一共出现了四个类，该算法对于该类的发生的概念进化准确的识别了出来，对于上述的实验数据得出的图像为



5.4 真实数据训练结果



6结论

7进一步的研究方向

概念漂移数据流的挖掘方法在过去的十多年中得到了一定的发展，但是现有的方法仍然存在着不足之处，这为研究者提供了进一步的研究方向。

(1) 大规模分布式数据流的计算模型、连续计算理论和随机数据算法的研究。其包括针对TCP/IP网络监控的算法和计算模型研究，电信通信网络数据流的计算，虚拟天文等科学数据流的存储、查询和挖掘，为了提高搜索引擎效率对Web数据流的挖掘研究等。科学和技术数据收集的性质正在迅速演变，例如数据量和数据产生速度呈指数方式增长，伴随着不断增加的复杂性和信息内容，因此有趣的或者异常显现必须被快速地测量和处理。

(2) 大规模数据流中模式发现问题研究。其面临的主要问题是发现的模式数量巨大，其中存在大量无用的模式。当事务最小支持度阈值低时，这个问题尤其严重。为了减少模式的数量，需要挖掘压缩模式，进一步研究压缩的频繁模式，并在压缩模式的基础上再次筛选，得到更加精简的模式集合。

(3) 数据流中的概念漂移问题是研究的热点，虽已有大量研究工作及成果，但缺少有效的概念漂移检测及处理方法，因此可以对模糊周期或模糊可预测的概念漂移问题进行研究。大多数概念漂移是假设在隐藏背景下发生的，是不可观测到的。但是，在某些应用中概念漂移是可以预测的，它沿时间轴或在不同对象中的模型化区域可能重新出现。例如粮食需求预测，可以用模糊周期性季节的影响为对象设定特定于群。

(4) 基于模式的分类方法研究。数据流中包含无限的数据，这些数据包含大量的冗余信息甚至是噪声，而模式发现可以去除数据中的冗余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。因此可以进一步对基于模式的数据流分类方法进行研究。

参考文献

[1] Garofalakis M, Gehrke J, Rastogi R. Querying and mining data streams:you only get one look a tutorial[C]//Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. ACM, 2002:635-635.

[2] Rosch E H. Natural categories[J]. Cognitive Psychology, 1973, 4(3): 328–350. doi: 10.1016/0010-0285(73)90017-0

[3] Hoens T R, Polikar R, Chawla N V. Learning from streaming data with concept drift and imbalance:an overview[J]. Progress in Artificial Intelligence, 2012, 1(1): 89–101. doi: 10.1007/s13748-011-0008-0

[4] Shao J, Ahmadi Z, Kramer S. Prototype-based learning on concept-drifting data streams[C]//Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2014:412-421.

[5] Domingos P, Hulten G. Mining high-speed data streams[C]//Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data mining. ACM, 2000:71-80.

[6] 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. ACM, 2001:97-106.

[7] Bifet A, Gavaldà R. Adaptive learning from evolving data streams[C]//International Symposium on Intelligent Data Analysis. Berlin Heidelberg:Springer, 2009:249-260.

[8] Liu G, Cheng H, Qin Z, et al. E-CVFDT:An improving CVFDT method for concept drift data stream[C]//Communications, Circuits and Systems (ICCCAS), 2013 International Conference on. IEEE, 2013, 1:315-318.

[9] Buntine W. Learning classification trees[J]. Statistics and Computing, 1992, 2(2): 63–73. doi: 10.1007/BF01889584

[10] Pfahringer B, Holmes G, Kirkby R. New options for hoeffding trees[C]//Australasian Joint Conference on Artificial Intelligence. Berlin Heidelberg:Springer, 2007:90-99.

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

[12] Kubát M. Floating approximation in time-varying knowledge bases[J]. Pattern Recognition Letters, 1989, 10(4): 223–227. doi: 10.1016/0167-8655(89)90092-5

[13] Bifet A, Gavalda R. Learning from Time-Changing Data with Adaptive Windowing[C]//SDM. 2007, 7:2007.

[14] Widmer G, Kubat M. Effective learning in dynamic environments by explicit context tracking[C]//European Conference on Machine Learning. Berlin Heidelberg:Springer, 1993:227-243.

[15] Alippi C, Boracchi G, Roveri M. Just in time classifiers:managing the slow drift case[C]//2009 International Joint Conference on Neural Networks. Atlanta:IEEE, 2009:114-120.

[16] 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. San Francisco:ACM, 2001:377-382.

[17] Kolter J Z, Maloof M A. Dynamic weighted majority:A new ensemble method for tracking concept drift[C]//Proceedings of the Third IEEE International Conference on Data Mining. Washington DC:IEEE, 2003:123-130.

[18] Chu F, Zaniolo C. Fast and light boosting for adaptive mining of data streams[C]//Pacific-Asia Conference on Knowledge Discovery and Data Mining. Berlin Heidelberg:Springer, 2004:282-292.

[19] Brzezinski D, Stefanowski J. Reacting to different types of concept drift:The accuracy updated ensemble algorithm[J]. IEEE Transactions on Neural Networks and Learning Systems, 2014, 25(1): 81–94. doi: 10.1109/TNNLS.2013.2251352

[20] Brzezinski D, Stefanowski J. Combining block-based and online methods in learning ensembles from concept drifting data streams[J]. [Information Sciences, 2014, 265(5): 50–67.](https://www.mendeley.com/research-papers/combining-blockbased-online-methods-learning-ensembles-concept-drifting-data-streams/)

[21] 孙岳, 毛国君, 刘旭, 等. 基于多分类器的数据流中的概念漂移挖掘[J]. 自动化学报, 2008, 34(1): 93-97.

[22] 陈照阳, 黄上腾. 流数据分类中的概念漂移问题研究[J]. 计算机应用与软件, 2009, 26(2): 254-256.

[23] 郭佳宏. 基于概念空间理论的概念进化[J]. 学术研究, 2009 (2): 49-53.

[24] 文益民, 强保华, 范志刚. 概念漂移数据流分类研究综述[J]. 智能系统学报, 2013, 8(2): 95-104.

[25] 张杰, 赵峰. 流数据概念漂移的检测算法[J]. 控制与决策, 2013, 28(1): 29-35.

[26] 张杰, 孙曰瑶. 数据流分类中的概念漂移转移估计方法研究[J]. 统计与信息论坛, 2011, 26(12): 19-25.

[27] 富春岩, 葛茂松. 一种能够适应概念漂移变化的数据流分类方法[J]. 智能系统学报, 2007, 2(4): 86-91.

[28] Gaber M M, Zaslavsky A, Krishnaswamy S. Mining data streams: a review[J]. ACM Sigmod Record, 2005, 34(2): 18-26.

[29] Gama J, Žliobaitė I, Bifet A, et al. A survey on concept drift adaptation[J]. ACM computing surveys (CSUR), 2014, 46(4): 44.

[30] Wang S, Minku L L, Yao X. Resampling-based ensemble methods for online class imbalance learning[J]. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(5): 1356-1368.

[31] Bifet A, Gavalda R. Learning from time-changing data with adaptive windowing[C]//Proceedings of the 2007 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2007: 443-448.

[32] Gama J, Medas P, Castillo G, et al. Learning with drift detection[C]//Brazilian symposium on artificial intelligence. Springer, Berlin, Heidelberg, 2004: 286-295.

附录

Main.m

clear all;

close all;

clc;

%第一类数据

mu1=[0 0 0]; %均值

S1=[0.3 0 0;0 0.35 0;0 0 0.3]; %协方差

data1=mvnrnd(mu1,S1,100); %产生高斯分布数据

%%第二类数据

mu2=[1.25 1.25 1.25];

S2=[0.3 0 0;0 0.35 0;0 0 0.3];

data2=mvnrnd(mu2,S2,100);

%第三个类数据

mu3=[-1.25 1.25 -1.25];

S3=[0.3 0 0;0 0.35 0;0 0 0.3];

data3=mvnrnd(mu3,S3,100);

%第四个类数据

mu4=[2 2 2];

S4=[0.3 0 0;0 0.35 0;0 0 0.3];

data4=mvnrnd(mu4,S4,100);

%显示数据

plot3(data1(:,1),data1(:,2),data1(:,3),'+');

hold on;

plot3(data2(:,1),data2(:,2),data2(:,3),'r+');

hold on;

plot3(data3(:,1),data3(:,2),data3(:,3),'g+');

hold on;

plot3(data4(:,1),data4(:,2),data4(:,3),'b+');

grid on;

%三类数据合成一个不带标号的数据类

data=[data1;data2;data3;data4]; %这里的data是不带标号的

%k-means聚类

[u re]=KMeans(data,4); %最后产生带标号的数据，标号在所有数据的最后，意思就是数据再加一维度

[m n]=size(re);

%最后显示聚类后的数据

figure;

hold on;

for i=1:m

if re(i,4)==1

plot3(re(i,1),re(i,2),re(i,3),'ro');

elseif re(i,4)==2

plot3(re(i,1),re(i,2),re(i,3),'go');

else

plot3(re(i,1),re(i,2),re(i,3),'bo');

end

end

grid on;

figure(3)

load('testdata.mat')

plot(data(:,1),data(:,2),':y')

hold on

plot(data(:,1),data(:,3),':r')

hold on

plot(data(:,1),data(:,4),':g')

hold on

plot(data(:,1),data(:,5),':k')

legend('class1','class2','class3','class4')

xlabel('Data Stram')

ylabel('类生成概率EGR')

KMeans.m

%N是数据一共分多少类

%data是输入的不带分类标号的数据

%u是每一类的中心

%re是返回的带分类标号的数据

function [u re]=KMeans(data,N)

[m n]=size(data); %m是数据个数，n是数据维数

ma=zeros(n); %每一维最大的数

mi=zeros(n); %每一维最小的数

u=zeros(N,n); %随机初始化，最终迭代到每一类的中心位置

for i=1:n

ma(i)=max(data(:,i)); %每一维最大的数

mi(i)=min(data(:,i)); %每一维最小的数

for j=1:N

u(j,i)=ma(i)+(mi(i)-ma(i))\*rand(); %随机初始化，不过还是在每一维[min max]中初始化好些

end

end

while 1

pre\_u=u; %上一次求得的中心位置

for i=1:N

tmp{i}=[]; % 公式一中的x(i)-uj,为公式一实现做准备

for j=1:m

tmp{i}=[tmp{i};data(j,:)-u(i,:)];

end

end

quan=zeros(m,N);

for i=1:m %公式一的实现

c=[];

for j=1:N

c=[c norm(tmp{j}(i,:))];

end

[junk index]=min(c);

quan(i,index)=norm(tmp{index}(i,:));

end

for i=1:N %公式二的实现

for j=1:n

u(i,j)=sum(quan(:,i).\*data(:,j))/sum(quan(:,i));

end

end

if norm(pre\_u-u)<0.1 %不断迭代直到位置不再变化

break;

end

end

re=[];

for i=1:m

tmp=[];

for j=1:N

tmp=[tmp norm(data(i,:)-u(j,:))];

end

[junk index]=min(tmp);

re=[re;data(i,:) index];

end

end

Muticoremain.m

%下面是多维 k-mean演示部分,包括2维，3维度，高维度

clear all

close all

t = 2000;

d = 3;

data = rand(d,t);

k = 6;

[res, record] = FunK\_meanPolyD(data,k);

[h, w] = size(res);

if h/k == 2

hold on

for i = 1:k

plot(res(i\*2-1,1:record(i)),res(i\*2,1:record(i)),'\*')

plot(mean(res(i\*2-1,1:record(i)),2),mean(res(i\*2,1:record(i)),2),'Marker','square','Color','k','MarkerFaceColor','k','LineStyle','none')

end

hold off

elseif h/k == 3

for i = 1:k

plot3(res(i\*3-2,1:record(i)),res(i\*3-1,1:record(i)),res(i\*3,1:record(i)),'\*')

plot3(mean(res(i\*3-2,1:record(i)),2),mean(res(i\*3-1,1:record(i)),2),mean(res(i\*3,1:record(i)),2),'Marker','square','Color','k','MarkerFaceColor','k','LineStyle','none')

hold on%注意：hold on 要写在plot3之后，这样三维图形才会正常绘制

end

hold off

else

disp(['结果维度大于3维，不能进行绘制'])

end

FunK\_meanPolyD.m

function [ res, record] = FunK\_meanPolyD(data,k )

% 功能：

% 实现多维空间k-mean聚类算法

% 输入：

% data是d\*n规格的矩阵，其中d代表维度，n代表样本的数量

% k 是分成的类别的数量

% 输出：

% res 是行数为(d\*k), 列数为record中最大元素值

% 对于res的行数为d\*k的解释：

% 1:d 是对应着第一类别元素

% d+1:2\*d 是对应着第二类别元素

% ···

% d\*(k-1)+1:d\*k 是对应着第k类别元素

%

% record规格为1\*k,记录着每一类别的有效元素的个数

j = 1;

% 下面是预分配一些空间

% seedX 和 seedY 中存放着所有种子

[h w] = size(data);

cnt = w; % 输入元素的数量

cntOfDimension = h; % d 中存放着本次处理数据的维度

%seed 中存放种子，每一行代表种子所在的一个维度，每一列是一个种子向量

seed = zeros(cntOfDimension,k);

oldSeed = zeros(cntOfDimension,k);

% 结果矩阵res中，数据存放规则：

% 以d行为一个单位，总共k个d行

% 第一个d行数据存放着第一类元素集合,其他同理

res = zeros(k\*cntOfDimension,cnt);

% 用来记录resX中每一行有效元素的个数

record = zeros(1,k);

r = 0;

for i = 1:k % 产生k个随机种子, 注意： 随机种子是来自元素集合

t = round(rand()\*cnt);

% 为保证种子不重叠

if i > 1 && t == r

i = i - 1;

continue;

end

seed(:,i) = data(:,t);

r = t;

end

while 1

record(:) = 0; % 重置为零

res(:) = 0;

for i = 1:cnt % 对所有元素遍历

% 下面是判断本次元素应该归为哪一类，这里我们是根据欧几里得距离进行类别判定

% k-mean算法认为元素应该归为距离最近的种子代表的类

distanceMin = 1; % distanceMin 中存放着最短欧几里得距离的种子点的下标

for j = 2:k

% 计算高维度的欧几里得距离

a = 0;

b = 0;

for row = 1:cntOfDimension

a = a + power(data(row,i)-seed(row,distanceMin),2);

b = b + power(data(row,i)-seed(row,j),2);

end

if a > b

distanceMin = j;

end

end

% 将本次元素点进行类别归并

row = (distanceMin-1)\*cntOfDimension + 1;

res(row:row+cntOfDimension-1,record(distanceMin)+1) = data(:,i);

record(distanceMin) = record(distanceMin)+1;

end

%record

oldSeed = seed;

% 移动种子至其类中心

for col = 1:k

if record(col) == 0

continue;

end

% 计算新的种子位置

row = (col-1)\*cntOfDimension + 1;

seed(:,col) = sum(res(row:row+cntOfDimension-1,:),2)/record(col);

end

% 如果本次得到的种子和上次的种子一致，则认为分类完毕。

if mean(seed == oldSeed) == 1

break;

end

end

maxPos = max(record);

res = res(:,1:maxPos);

end

致谢