

文章编号:1007-5321(2017)03-0043-08

DOI:10.13190/j.jbupt.2017.03.005

一种基于原型学习的自适应概念漂移分类方法

苏 静¹, 裘晓峰¹, 李书芳¹, 刘道伟², 张春红¹

(1. 北京邮电大学 网络体系构建与融合北京市重点实验室, 北京 100876; 2. 中国电力科学研究院, 郑州 450052)

摘要: 为了更准确快速地处理或适应概念漂移,提出了基于原型学习的数据流分类算法,基于发掘并优化现有方法存在的问题,提出了新的方法模型 SyncPrototype,在预测方法、原型判定与更新方法等处理概念漂移问题的关键部分做出了新的尝试与优化。实验结果证明,相较于现有方法,SyncPrototype 模型在分类性能、概念漂移的响应速度以及时间性能等方面都有明显提高,能够更加有效处理并适应数据流概念漂移问题。

关键词: 数据流; 概念漂移; 分类

中图分类号: TN992.53

文献标志码: A

A Prototype-Based Adaptive Concept Drift Classification Method

SU Jing¹, QIU Xiao-feng¹, LI Shu-fang¹, LIU Dao-wei², ZHANG Chun-hong¹

(1. Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications, Beijing 100876, China; 2. China Electric Power Research Institute, Zhengzhou 450052, China)

Abstract: As a frequent problem that needs to be mainly dealt with in supervised learning scenario of streaming data, the concept drift, primarily, occurs when the data distribution or the target variable changes over time. As typical data streams, the research method which real-time solves or adapts to the concept drift of data streams can provide strong support for grid security dispatch and stable control of real-time decision-making. For accurate and quick dealing with or adapting to concept drift, a prototype-based learning algorithm of data streams classification is discussed. Based on improving the problems which have been explored in existing algorithm, a new algorithm SyncPrototype was proposed, which makes new optimization in terms of methods of classification method, prototype construction and updating. Experiment shows that SyncPrototype can outperforms the existing algorithm in terms of classification performance, time performance and response rate.

Key words: data streams; concept drift; classification

在当今信息时代,通信、计算机和网络技术的快速发展导致数据出现了爆炸性增长。海量的数据以高速有序的形式到达,学者将此类数据形式称为数据流^[1](Data Stream),即大量且连续的和潜在无限的数据有序序列。数据流具有不同于传统数据的特征^[2]:时间有序性、快速变化性、海量的和潜在无限

性,数据流的这些特性向传统数据挖掘提出了新的挑战,其中最主要的技术挑战或问题为概念漂移的处理。

概念漂移^[3]指数据流的目标概念或规则随时间推移产生不可预见的变化这一现象。这种情况下,如若不处理概念漂移,则会导致已有的模型时效

收稿日期: 2016-07-23

基金项目: 国家电网公司科技项目(XT71-15-056)

作者简介: 苏 静(1991—),女,硕士生, E-mail: sj199137@126.com; 裘晓峰(1969—),女,副教授,硕士生导师。

性下降、性能降低,无法适应新的数据分布。

基于原型^[2]的分类算法,其研究核心在于如何选取并更新能够有效代表当前概念的原型。为解决并优化基于原型学习现有算法 SyncStream^[4]存在的问题,提出了新的算法模型 SyncPrototype,在预测方法、原型判定与更新方法等方面做出了基于上述问题的改进;提出了评分策略邻居预测方法提高了分类预测性能;提出了评分策略与错误驱动表示值更新方法提高了适应概念漂移的相应速度,并且实现了原型表示值全面更新;提出了基于等级策略的原型层更新方法,从而优化了更新逻辑与时间性能,能够适应类别分布不均衡情况;此外,针对冗余结构进行了简化,进一步提高了时间性能。

1 相关工作

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^[3]:

- 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] (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] (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] (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] (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] (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] 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] (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] 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] 与 ADWIN^[13]。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] 算法引入了自适应窗,尝试自适应改变窗口大小。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。相继提出了 ADWIN2^[13] 算法,引入了一个内存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] 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,对后续的实例获得准确的鲁棒性的分类性能。流集成算法^[16] (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] 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] (ABE, adaptive boosting ensemble) 使用限制深度的决策树来开发 boosting 的性能特点,检测漂移后,丢弃当前模型进行重构。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] (AUE2, accuracy updated ensemble) 以及在线精度更新集成算法^[20] (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] 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 SyncPrototype 算法

2.1 概述

为解决并优化基于原型现有算法的问题, 提出了新的算法模型 SyncPrototype, 在预测方法、原型更新方法等方面做出了改进. 原型更新主要包括原型表示值更新以及原型层达到容量上限时的原型层压缩更新; 表示值更新的意义在于区分原型价值从而服务于预测邻居筛选以及原型层更新; 原型层更新的意义在于能够持续接收新数据, 从而动态追踪并代表最新概念. 对于现有算法, SyncPrototype 主要改进思路分析如下.

1) 分类预测方面, 现有算法采用最近邻方法, 对于存在数据噪声的数据, 会严重影响分类性能, 抗

噪性能差, 数据普适性较低. 针对此问题, SyncPrototype 提出了评分策略邻居预测方法, 设计了以距离为主的邻居评分策略, 利用分数阈值范围内的邻居原型进行预测, 并且加入了原型等级约束, 取消了阈值范围内低等级邻居的预测资格, 从而实现了抗噪性能、数据普适性以及分类预测性能的综合提升.

2) 原型表示值更新方面, 现有算法更新率较低, 通过一个训练实例的预测对错仅对其最近邻居进行表示值更新, 限制了适应概念漂移的响应速度. 为解决此问题, SyncPrototype 提出了评分策略与错误驱动表示值更新方法, 通过一个训练实例的预测结果更新分数阈值上的多个原型, 通过增加更新率提高了适应概念漂移的响应速度.

3) 现有算法没有考虑类别不平衡情况, 原型层中各类原型比例非常悬殊甚至经常出现没有某类别的情况, 无法实现类别全面分布, 严重影响召回率. 针对此问题, SyncPrototype 综合考虑了原型等级(依据表示值划分)与各类别比例分布, 优化了类别分布不平衡情况下的原型层压缩.

4) 算法的时间复杂度方面, 现有算法存在结构冗余问题. 现有算法 SyncStream 主要包括 3 部分: ①进行分类和原型表示值更新; ②追踪突然概念漂移的产生; ③原型层达到上限时进行原型压缩. 后两部分都存在冗余问题: 首先, 追踪突然概念漂移的检测结果对数据块尺寸、漂移程度阈值以及数据特性的敏感性很强, 检测结果缺乏稳定性与可靠性, 并且当突然概念漂移产生后, 由于原型更新, 极有可能在数据块缓存结束并进行检测之前, 原型层便已适应漂移后的规则; 其次, 聚类压缩旧概念不仅严重损耗了时间性能, 也并没有实质意义. SyncPrototype 摒弃了冗余结构, 在分类方法与表示值学习方法方面与 SyncStream 时间复杂度相同, 所以在时间性能方面实现了明显提高.

SyncPrototype 的模型示意图如图 1 所示, 主要流程如下.

1) 利用原型层(容量上限为 maxP)动态存储保留代表当前数据分布的原型, 最初存入部分训练样本实现初始化.

2) 处理连续进入的数据流实例时, 利用评分策略邻居预测方法进行分类, 根据实例中训练样本预测结果通过评分策略与错误驱动表示值学习方法对原型表示值进行更新, 然后将此样本加入原型层.

3) 为实现原型层持续动态更新, 当达到原型层

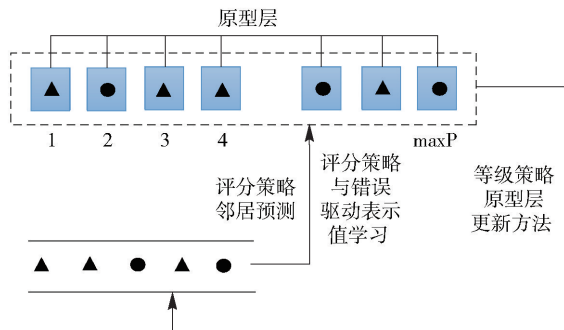


图1 SyncPrototype 算法模型示意图

容量上限时,利用等级策略原型层更新方法对原型层进行删除与保留判定,保证一定的压缩规模;并且在原型层更新部分考虑了类别分布不均衡情况,在压缩更新时保证了原型层在各类别分布的全面性. 然后继续第2)步.

2.2 分类预测与原型表示值更新方法

为能够更有效捕获到数据流中有意义的训练集样本(原型),SyncPrototype 在分类预测方面提出了评分策略邻居预测方法;在原型表示值更新方面提出了评分策略与错误驱动表示值学习. 主要原理是将预测与更新的邻居范围扩展到大于1的数值,即用于预测以及每次更新表示值的原型个数大于1,如图2所示,其中的点为原型层中原型的值分布,其中虚线圆圈范围内的原型为新训练集样本(即圆圈内中心三角)通过评分策略确定出的邻居集合,图中情况下如果仅利用最近邻(图中圆圈实例)分类,会导致分类错误,所以通过圆圈范围内的原型对进行预测,试图尽可能降低噪声影响,获得更稳定准确的预测结果;然后通过预测结果对错更新圆圈范围内原型的表示值,从而获得了更高的更新率. 此方法的核心在于邻居集合的判定. 原理主要如下.

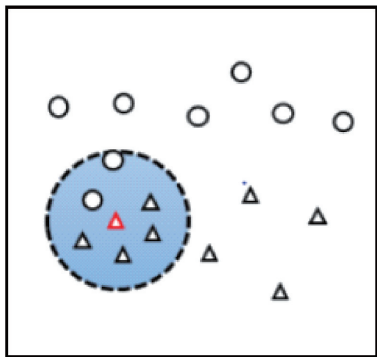


图2 评分策略邻居预测与错误驱动表示值学习

给定 $x \in R^d$, d 维实数向量,作为数据流实例, y

为原型层 P 的原型,定义2个邻居集合:

$$N_p(x) = N_r(x) = \{y \in P \mid \text{score}(y) \geq \phi\} \quad (1)$$

其中: $N_p(x)$ 和 $N_r(x)$ 分别为预测邻居集合和更新表示值邻居集合, $\phi \in [0, 1]$ 为设定的分数阈值, $\text{score}(y)$ 函数代表的是原型 y 对于实例 x 的价值分数,定义如下:

$$\text{score}(y) = \frac{e^{\text{sim}(x,y)} - 1}{e - 1} \quad (2)$$

其中: $\text{sim}(x, y)$ 为两者的一种归一化相似度,则 $\text{score}(y) \in [0, 1]$, 分数越高的邻居,对于分类预测的意义越大,所以采用分数值与 $\text{sim}(x, y)$ 呈非线性关系的方法,放大凸显程度随分值上升而增加,从而细化并着重突出对当前实例 x 高分原型的价值. $\text{sim}(x, y)$ 定义为

$$\text{sim}(x, y) = \frac{\max D - \text{dist}(x, y)}{\max D - \min D} \quad (3)$$

其中: $\max D$ 、 $\min D$ 与 $\text{dist}(x, y)$ 分别为实例与原型层中原型的最大距离、最小距离与当前距离,此处指的是欧氏距离.

确定出邻居范围之后,利用范围内原型对 x 进行预测, x 的预测类别为邻居范围内最高分数类别,方法如下:

$$x. \text{prelabel} = \arg \max \text{score}(l) \quad (4)$$

其中 $\text{score}(l)$ 为 $N_p(x)$ 范围内的类别分数:

$$\text{score}(l_m) = \frac{1}{|N_{l_m}|} \sum_{y \in N_{l_m}, |N_{l_m}| \geq 1} \text{score}(y) \quad (5)$$

$$N_{l_m} = \{y \in N_p(x) \mid y. \text{label} = l_m, \text{rank}(y) \geq 0\} \quad (6)$$

则实例 x 的预测类别为预测邻居范围内分数最高的类别,其中 $l_m \in \{l_1, l_2, \dots, l_M\}$, 共有 M 个类别. 此处需要注意的是,在预测时邻居范围除了受评分阈值限制,也受到了等级(定义见下节)约束,定义小于0等级的邻居属于噪声邻居,即无效邻居,进行剔除.

分类预测之后,在此基础上,根据预测结果,基于局部错误驱动方法用于进行原型表示值的更新,原理主要如下.

$$\text{Rep}(y) = \text{Rep}(y) + \text{sign}(x, y) \text{score}(y) \quad (7)$$

其中: $y \in N_r(x)$, 每个原型在最初进入原型层时,其表示值初始化为0. 根据预测结果对更改表示值邻居范围内的原型进行表示值更新,此时可以实现更高的更新率,从而实现原型全面更新,即在相邻两次原型层更新间隔内,通过新的数据流实例能够尽可能全面地更新到原型层的全部原型的表示值,从而

实现原型价值层次更加分明。

2.3 原型层更新部分

当原型层达到容量上限时,需要对原型层现有原型进行一定程度的压缩,从而进行后续新实例对原型层的流式更新。提出了等级策略原型层更新方法,根据原型等级对原型进行删除与保留判断,保证一定的更新删减规模,并且在原型层更新部分考虑类别分布不均衡问题,尽可能保证原型层在各类别分布的全面性。

根据原型表示值对原型进行等级分类,等级评定规则定义如下:

$$\text{rank}(y) = \lceil \text{Rep}(y) - 0.5 \rceil \quad (8)$$

其中 $\lceil \cdot \rceil$ 分别表示对 y 进行上取整。删除逻辑如下:

1) 根据原型等级,先删除小于 1 等级的原型,笔者认为在原型全面更新的前提下,此部分原型属于无价值原型;

2) 如果删除个数小于原型层容量的倍时,则按照等级由低到高对原型继续进行删除,直至删除数目达到倍为止;

3) 删除过程中加入对类别分布不均衡进行的处理,对原型层类别进行追踪记录与删除判断,综合考虑原型等级与原型层类别分布比例,保证原型层具备全面的类别分布。

2.4 算法

基于上述原理,SyncPrototype 算法流程如下。

SyncPrototype 算法

Input: $D, D_{\text{init}}, \max P, \phi$ where ϕ is the threshold of neighborhood, is the threshold of updating rate, β is the threshold for class maintain, num C is number of classes

Output: Prototype Level as P

1. $P.$ insert(D_{init}); // P -Tree initialization

2. for each new example $x \in D$ do

3. // Prediction and representation updating

4. for each prototype $y \in P$ do

5. if $\text{score}(y) \geq \phi$ and $\text{rank}(k) \geq 0$ then

6. Calculate $\text{score}(l_m)$, $l_m = y.$ label,

7. $x.$ prelabel = arg max $\text{score}(l)$

8. if $\text{score}(y) \geq \phi$ then

9. $\text{Ref}(y) = \text{Rep}(y) + \text{sign}(x, y) \text{score}(l)$

10. $P.$ insert(x)

11. // maximum boundary in prototype level

12. if $P.$ prototypesSize == $\max P$ then

13. rank = 0

14. While DeleteNum < $\alpha \cdot \max P$ do

15. For each prototype $y \in P$ do

16. if $(y.$ label). num $\geq \lceil \max P / (\beta \cdot \text{num} C) \rceil$ and $\text{rank}(y) \leq \text{rank}$ and DeleteNum < $\alpha \cdot \max P$ then

17. $P.$ delete(y)

18. rank ++

19. else then Break

3 实验评估

3.1 数据集

对多项数据集进行了相关实验评估。数据集介绍如下:

1) 数据流仿真漂移数据集 (Synthetic Data): 在二维空间中仿真了随时间变化的存在突然概念漂移的仿真数据流数据,从而可以直观地呈现算法处理概念漂移并适应新概念的效果;

2) 开源用电数据:包括 45 321 个数据流实例,是澳大利亚新南威尔士州电力市场以 5 min 为间隔采集的用电数据。其中电力价格并非固定,受市场的供应与需求影响并随之发生变化;

3) 电网 39 节点暂态仿真故障数据:进行了 10 机 39 节点的电力数据流仿真。在电力系统分析综合程序 (PSASP, power system analysis software package) 仿真过程中,在母线 28 上设置三相短路故障。故障持续时间:0.7、0.9、1.1、1.3 s,仿真步长为 0.01 s,主要包括节点、支路、电源、负荷等仿真数据。

3.2 概念漂移处理

SyncPrototype 算法,通过原型更新来追踪适应变化的概念。突然概念漂移的变化幅度大于逐渐概念漂移,所以可通过评估突然概念漂移的适应情况来衡量算法性能。SyncPrototype 通过基于评分策略的错误驱动表示值更新方法能够更快速有效地适应漂移,首先运用二维仿真数据流进行实验评估,从而呈现并证明算法的性能与效果。

仿真突然概念漂移前后 2 个概念的二维数据流数值分布,如图 3、图 4 所示,可以看出,图中二分类的决策函数发生了明显变化。通过突然概念漂移后不同时间间隔 T (取 50 个实例点与 100 个实例点) 原型层分布 (以下实验参数设置为 $\max P = 200, \alpha = 0.3, \beta = 5$) 体现 SyncPrototype 对于概念漂移的适应

情况,测试算法在不同分数阈值 ϕ 下的效果与性能,实验结果如图 5 所示。

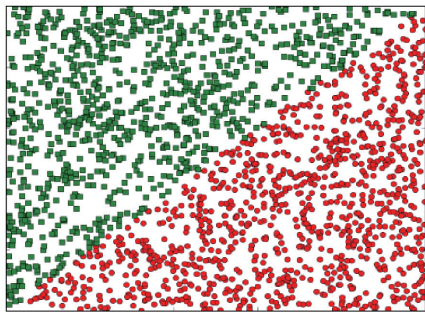


图 3 突然漂移之前

由于算法提高了原型更新率,根据图 6 实验结果,可以看出算法 SyncPrototype 可以在突然概念漂移之后较短时间间隔内适应新概念,如图 5(e) 中当原型更新邻居分数阈值 ϕ 为 0.95 时,100 个实例之后原型层基本上已经适应了新的概念,体现出了 SyncPrototype 在适应突然概念漂移方面的性能优势。此外,通过实验结果可以看出,分数阈值 ϕ 对原型层更新存在权衡效果,在最佳阈值(图 5 中为 0.95)之前,即 1.0 至最佳阈值,适应新概念的效果随阈值降低而增强,然而小于最佳阈值时,两者成正

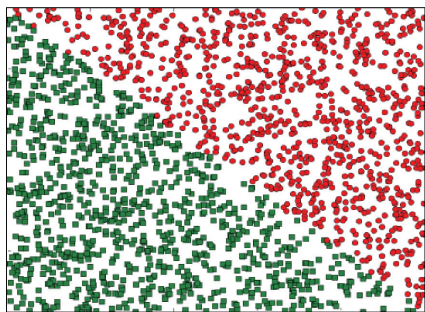


图 4 突然漂移之后

比,并非更新的邻居范围越大越有效果,因为当阈值过小,范围过大时,上一个概念中的代表性原型表示值也越高,产生新概念时,删减之前累积的过高的表示值则越不易。而此最佳阈值因数据特性而异,需要具体情况具体调参实验确定。

3.3 类性能分析

在此部分,使用开源用电数据集以及电网 39 节点暂态仿真数据集进行性能对比实验,其他参数设置见上节。数据集特征介绍如表 1 所示,其中 39-Node-Bus、39-Node-Line、39-Node-Generator、39-Node-Load 分别代表仿真数据中的母线、线路、电源与负荷数据。

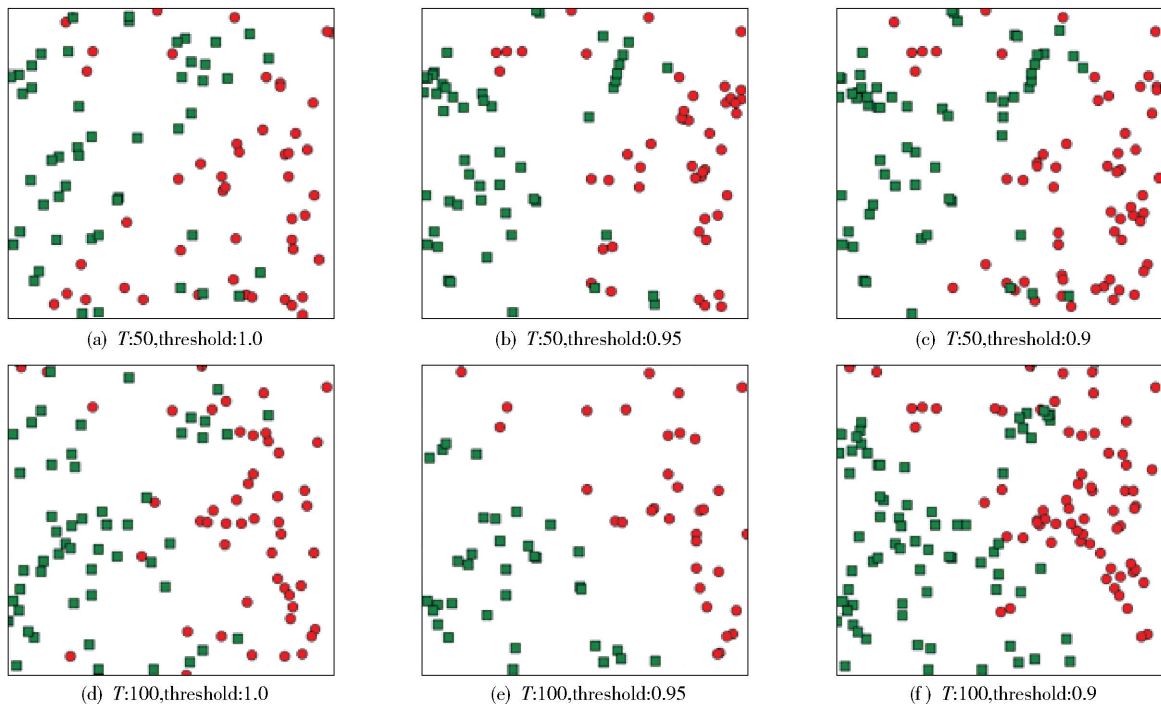


图 5 突然概念漂移后原型分布

表 1 实验数据集介绍

数据	实例数	维度	类别
Electricity Data	45 312	8	2
39-Node-Bus	156 156	3	2
39-Node-Line	184 184	6	2
39-Node-Generator	40 040	11	2
39-Node-Load	76 076	2	2

在数据流分类算法中,SyncStream 是近期提出的性能较为突出的基于原型的方法,相关实验^[4]证明 SyncStream 在时间性能和分类性能方面皆优于经典的数据流分类算法。作为基于原型学习的方法,笔者主要进行 SyncPrototype 与 SyncStream 2 个算法的性能对比实验。

表 2 SyncStream 与 SyncPrototype 算法实验结果

数据集	算法	准确率/%	精确率/%	召回率 l/%	F1-measure	t/ms
Electricity Data	SyncStream	81. 53	81. 08	81. 12	81. 09	2 881
	Sync-Prototype	82. 62	82. 27	82. 06	82. 17	1 264
39-Node-Bus	SyncStream	96. 36	88. 51	89. 74	89. 12	7 648
	Sync-Prototype	97. 21	91. 65	91. 94	91. 80	4 228
39-Node-Line	SyncStream	97. 25	91. 21	92. 31	91. 76	8 601
	Sync-Prototype	97. 76	92. 53	94. 58	93. 54	4 526
39-Node-Generator	SyncStream	99. 65	98. 72	98. 98	98. 85	2 919
	Sync-Prototype	99. 74	98. 88	99. 62	99. 24	1 436
39-Node-Load	SyncStream	97. 83	92. 67	94. 01	93. 33	2 838
	Sync-Prototype	98. 27	94. 25	95. 71	94. 97	2 025

4 结束语

研究了原型学习,基于优化已有算法存在的问题,提出了 SyncPrototype 算法,在预测方法、原型判定与更新方法、类别不均衡、时间复杂度等处理概念漂移问题的关键部分做出了优化。主要利用评分策略邻居预测方法进行分类,根据实例中训练样本预测结果通过评分策略与错误驱动表示值学习方法对原型表示值进行更新,然后利用等级策略原型层更新方法进行原型层的持续动态存储。对数据流分类研究常用开源数据集以及仿真故障产生的电网 39 节点暂态仿真数据进行了相关实验,实验结果表明,SyncPrototype 在分类性能、概念漂移的响应速度以及时间性能等方面都有明显提高,能够更加有效处理并适应数据流概念漂移问题。

原型构建的效果直接影响数据集的分类性能,更有效表示当前概念的原型构建有益于更好的分类性能。表 2 所示为 SyncPrototype 与 SyncStream 算法在不同实验数据集下不同分类指标方面的分类性能。根据算法性能结果对比可以看出,对于不同的数据集,相较于 SyncStream,由于 SyncPrototype 去除了冗余结构,实现了时间性能的较大提高,从而更符合数据流分类在实时性与处理速度方面的要求,此外,由于 SyncPrototype 提出的评分策略邻居预测方法以及原型等级约束,算法在分类各性能指标上也实现了明显提升。实验结果证明,相较于 SyncStream,SyncPrototype 在分类性能、概念漂移的响应速度以及时间性能等方面都明显提高,能够有效处理并适应数据流概念漂移问题。

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.

[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.

[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 Interna-

- tional 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, Gavalda 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.
- [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] Kubat M. Floating approximation in time-varying knowledge bases [J]. Pattern Recognition Letters, 1989, 10(4): 223-227.
- [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.
- [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.