中国科学: 信息科学 2021 年 第 51 卷 第 1 期: 1–12
SCIENTIA SINICA Informationis
论文

基于决策树模型重用的分布变化流数据学习

赵鹏, 周志华*

南京大学计算机软件新技术国家重点实验室, 南京 210023
* 通信作者. E-mail: zhouzh@lamda.nju.edu.cn

收稿日期: 2020–06–09; 接受日期: 2020–08–05; 网络出版日期: 2020–12–25

国家自然科学基金 (批准号: 61921006) 资助项目

摘要 在很多真实应用中, 数据以流的形式不断被收集得到. 由于数据收集环境往往发生动态变化, 流数据的分布也会随时间不断变化. 传统的机器学习技术依赖于数据独立同分布假设, 因而在这类分布变化的流数据学习问题上难以奏效. 本文提出一种基于决策树模型重用的算法进行分布变化的流数据学习. 该算法是一种在线集成学习方法: 算法将维护一个模型库, 并通过决策树模型重用机制更新模型库. 其核心思想是希望从历史数据中挖掘与当前学习相关的知识, 从而抵御分布变化造成的影响. 通过在合成数据集和真实数据集上进行实验, 我们验证了本文提出方法的有效性.

关键词 机器学习, 分布变化, 流数据, 模型重用, 集成学习, 动态环境

1 引言

机器学习技术在诸多领域得到了广泛应用, 包括图像、视频、语音、文本处理等 [1~3]. 传统的机器学习技术假定数据分布是恒定的, 但数据收集的环境通常是开放动态的, 因而数据分布恒定这一假设往往难以满足. 特别在诸如天气预测、股票价格预测、语音识别等真实应用场景中, 数据以流的形式不断被在线得到, 随着时间不断累积, 数据分布往往会随着收集环境的动态变化而不断变化. 传统的机器学习算法及理论依赖于数据同分布假设, 难以适用于这类分布不断变化的流数据问题. 因而, 针对分布变化的流数据, 如何设计性能良好且有理论保障的学习算法是非常重要的课题.

首先需要明确的是, 如果对流数据的分布变化一无所知, 甚至允许分布任意、敌对变化, 那么这样的问题是不可学的. 分布变化流数据学习的基本假设是: 历史数据中包含对当前预测有价值的知识. 该领域以往有一些相关工作, 代表性方法如基于滑动窗口机制的学习算法 [4~6]、基于遗忘加权机制的学习算法 [7,8] 和基于集成学习机制的学习算法 [9~11], 均建立在前述基本假设之上, 区别在于如何建模并利用当前预测和历史数据之间的相关性. 如果没有该基本假设, 分布变化的流数据学习将无从谈起.

引用格式: 赵鹏, 周志华. 基于决策树模型重用的分布变化流数据学习. 中国科学: 信息科学, 2021, 51: 1–12, doi: 10.1360/SSI-2020-0170
Zhao P, Zhou Z-H. Learning from distribution-changing data streams via decision tree model reuse (in Chinese). Sci Sin Inform, 2021, 51: 1–12, doi: 10.1360/SSI-2020-0170

基于上述认知, 本文试图显式建模当前预测和历史数据之间的关系, 自适应挖掘历史数据中对当前预测有用的知识, 并通过模型重用 (model reuse) 机制利用这些知识以辅助当前时刻的学习. 具体而言, 本文采用在线集成学习的机制, 通过维持一个模型库并对其动态调整更新以对抗流数据中的分布变化. 由于决策树模型的灵活性, 它特别适合集成学习框架, 并且决策树模型可以通过简单的伸展和收缩操作实现模型知识的重用, 因而我们选择决策树模型作为基学习器. 本文提出 CondorForest 学习算法. 在每次模型更新时刻, CondorForest 算法首先通过自适应权重调整机制, 给出历史模型相对当前数据的可重用性权重. 然后算法根据可重用性权重重用历史的决策树模型学习得到新的决策树, 并加入到模型库中进行更新. 我们通过在合成数据集以及真实数据集上进行实验, 验证了本文提出算法的有效性.

本文第 2 节从分布变化的流数据学习和模型重用两个方面介绍相关工作. 第 3 节介绍本文提出的 CondorForest 算法, 并给出了相应分析. 第 4 节通过在合成和真实数据集上的实验, 验证算法的有效性. 最后总结全文, 并对未来工作做出展望.

2 相关工作

本节从分布变化的流数据学习和模型重用两个方面介绍相关工作.

2.1 分布变化的流数据学习

在很多真实应用中, 数据随时间不断被累积. 由于数据收集环境往往不断动态变化, 数据分布因此随之不断变化, 这就要求学习模型能够快速更新以适应分布变化. 这类现象在文献中也常被称为概念漂移 (concept drift) [12]. 如前文所述, 如果分布变化是毫无规律甚至敌对的, 这样的学习任务是不可学的. 因而, 分布变化流数据学习的基本假设是: 历史数据中包含对当前预测有价值的知识. 以往的研究如基于滑动窗口 (sliding window) [4,5] 及基于遗忘机制 (forgetting mechanism) [7,8,13,14] 的方法本质上都是建立在该假设上的. 这些方法认为接近当前时刻的历史数据应该包含更多当前数据可以利用的知识. 另一类典型方法是基于集成学习技术 (ensemble method) [15], 其主要想法是维护包含多个模型的模型库, 并根据各个模型的表现对模型库动态更新调整. 这类方法的代表性算法包括 DWM (dynamic weighted majority) 算法 [9,10]、AddExp (additive expert ensemble) 算法 [16] 和 Learn.NSE (learning in non-stationary environments) 算法 [17] 等.

关于处理分布变化的流数据的更多算法及介绍, 读者可以参考综述 [11,12], 后者集中讨论了基于集成学习技术的方法. 对集成学习的理论与算法感兴趣的读者可以参考专著 [15,18].

2.2 模型重用

模型重用的基本设定是: 给定一个 (或一些) 原始模型和少量当前任务的标记数据, 希望学习算法能够有效地利用原始模型辅助当前任务的学习. 模型重用是非常重要的学习任务, 是学件 (learnware) 的基本需求之一 [19].

对于支持向量机模型, 一种常用的模型重用方法是偏差正则化 (biased regularization) 技术 [20], 具体而言, 偏差正则化技术将原始模型加入到支持向量机的正则化项中作为偏差项, 从而希望在原始模型的基础上利用当前任务的数据学到一个好的模型. 偏差正则化技术在许多领域得到了应用, 包括视频概念检测 [21]、跨模态图像分类 [22,23] 等. 此外, 最近的研究从理论层面证明了偏差正则化的有效性 [24~27]. 具体而言, Kuzborskij 等 [24] 利用算法稳定性的工具对二分类问题的偏差正则化进行分析,

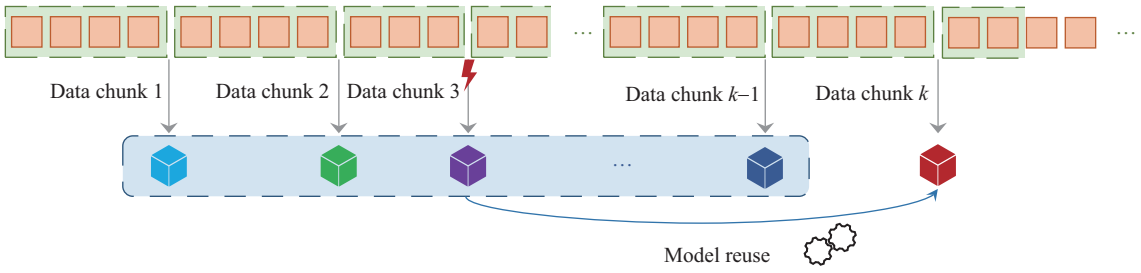

图 1 (网络版彩图) CondorForsest 算法示意图

Figure 1 (Color online) Illustration of the CondorForest algorithm

后续又利用 Rademacher 复杂度的工具进一步提升了之前的分析结果 [25]; Zhao 等 [26] 进一步弱化了已有结果对损失函数的假设, 并将理论结果从原有的二分类拓展到多分类任务; Du 等 [27] 采用转换函数的技术对偏差正则化进行了分析.

除了上述工作以外, 还有一些其他模型重用的方法及应用. 例如, Wu 等 [28] 利用领域知识重用已有模型; Segev 等 [29] 采用决策树进行模型重用; Ye 等 [30] 考虑不同特征空间的模型重用, 采用语义映射重用不同特征空间下的特征表示不变量; Wu 等 [31] 通过优化多个局部模型集成的整体表现, 为多方学习问题提出了一种新颖的异质模型重用技术; Li 等 [32] 以及 Ding 等 [33] 通过模型重用技术解决学习过程中性能评价指标可能变化的问题.

3 CondorForest 算法

决策树模型因其具有优异的性能、良好的可解释性, 以及适配于集成学习框架的特性, 在实践中被广泛使用 [34,35]. 本文采用决策树模型作为基分类器, 通过集成学习框架, 以动态调整模型库的方式对抗流数据中可能出现的分布变化.

本节主要介绍所提出的 CondorForest 算法. 我们首先简述算法框架, 然后介绍算法的两个核心组件: 决策树模型重用 (decision tree model reuse) 和自适应权重调整 (adaptive weight adjustment).

3.1 算法框架

在本文的设定中, 数据以流的形式给出. 在第 t 时刻, 学习算法首先获得样本 $\boldsymbol{z}_t = (\boldsymbol{x}_t, y_t) \in \mathcal{X} \times \mathcal{Y}$ 的属性 $\boldsymbol{x}_t$, 其中 $\mathcal{X}$ 和 $\mathcal{Y}$ 分别为样本的属性空间和标记空间. 接着, 算法根据当前模型做出预测 $\widehat{y}_t$. 然后, 算法收到样本的真实标记 y_t, 根据预测和真实标记的损失以更新模型. 如图 1 所示, 为了处理分布变化的流式数据, CondorForest 将周期性进行模型更新. 同时, 为了应对分布可能出现的突变, CondorForest 会额外在后台运行分布变化检测器 $\mathfrak{D}$. 当分布变化检测器检测到变化时, 算法也将进行模型更新. 在第 k 次模型更新时, CondorForest 会通过自适应权重调整机制从模型库中选择一个历史的决策树模型, 并在此模型基础上结合数据块 k 中的样本进行决策树模型重用, 学习得到一个新的决策树模型并加入到模型库中.

在模型更新时, CondorForest 算法一方面利用当前数据块的样本, 另一方面重用模型库中的历史模型. 为了方便叙述, 我们首先约定一些数学记号. 在第 t 时刻, 记当前的模型库为 $\mathcal{M}$, 其中包含 K 个决策树模型, 即 $\mathcal{M} = \{\mathsf{T}_1, \ldots, \mathsf{T}_K\}$, 其中 $\mathsf{T}_k : \mathcal{X} \mapsto \mathbb{R}$ 为第 k 棵决策树模型, $k = 1, \ldots, K$. 记当前数据块 $D_K = \{(\boldsymbol{x}_i^{(K)}, y_i^{(K)})\}_{i=1}^{m_K}$, 其中 m_K 为当前数据块中的样本总数. 如图 1 所示, 在第 K 次模

型更新时, 算法会利用当前数据块 D_K 的样本及模型库 $\mathcal{M}$ 中的历史模型, 学习得到一棵新的决策树 T_{new} 并加入到模型库中. 为了简化符号, 在不影响上下文理解的情况, 我们将省略数据块符号中的上下标 K, 简记当前数据块为 $D=\{(\boldsymbol{x}_i,y_i)\}_{i=1}^m$. 此外, CondorForest 算法会根据决策树模型与当前数据的相关性 (即可重用性), 对模型库中的每棵决策树赋予权重. 决策树 T_k 当前时刻对应的权重为 w_t^k, $k=1,\ldots,K$.

从上述流程我们可以看出, CondorForest 算法包含以下两个关键的组件.

- 决策树模型重用: 在模型更新时, 利用当前数据块的样本, 同时重用模型库中历史模型的知识;
- 自适应权重调整: 在模型更新时, 根据历史模型相对当前数据的 "可重用性" 进行模型权重调整.

在 CondorForest 算法中, 由于基分类器是决策树, 我们需要使用适配于决策树模型的模型重用方法, 本文采用结构伸展收缩算法 [29] 以重用决策树模型. 此外, 我们采用乘法权重更新方法 [36] 进行模型自适应权重调整. 3.2 和 3.3 小节分别介绍这两个关键组件.

3.2 决策树模型重用: 结构伸展收缩算法

在决策树模型重用部分, CondorForest 算法首先根据模型库中模型的权重采样得到一棵决策树, 记为 $\widetilde{\mathsf{T}}$. 接着, 算法将通过模型重用机制学习得到一棵新的决策树. 具体而言, 模型重用阶段的主要任务是: 给定决策树 $\widetilde{\mathsf{T}}$ 及当前数据块 $D=\{(\boldsymbol{x}_i,y_i)\}_{i=1}^m$, 学习得到一棵新的决策树 T_{new}. 下面, 我们介绍决策树模型重用算法 —— 结构伸展收缩算法 [29].

首先, 在介绍具体的算法之前, 我们通过一个简单的例子直观理解决策树的模型重用 [29]. 如图 2 所示, 图 2(a) 和 (b) 分别表示原始空间和目标空间. 考虑二分类的任务, 蓝色和黄色分别表示正负类别的概念区间. 显然, 原始空间和目标空间的概念是不同的, 但是又有一定的相似之处. 因此, 我们希望能够重用在原始空间上训练得到的决策树模型, 辅助目标空间概念的学习.

在图 2(a) 上方的原始空间概念示意图中, 设蓝色正方形边长为 1, 黄色正方形边长为 0.5, 记蓝色正方形的左下角顶点为原点 $(0,0)$, 则从原点开始按顺时针计数, 蓝色正方形的 4 个顶点位置分别为 $(0,0)$, $(0,1)$, $(1,1)$, $(1,0)$. 与之对应, 黄色正方形的 4 个顶点位置分别为 $(0.25,0.25)$, $(0.25,0.75)$, $(0.75,0.75)$, $(0.75,0.25)$. 由此, 我们可以得到原始空间概念对应的决策树模型, 如图 2(a) 下方所示.

在图 2(b) 上方的目标空间概念示意图中, 设蓝色正方形的边长为 1, 此外的两个黄色长方形长为 0.5, 宽为 0.2. 上方黄色长方形的 4 个顶点位置分别为 $(0.25,0.55)$, $(0.25,0.75)$, $(0.75,0.75)$, $(0.75,0.55)$; 下方黄色长方形的 4 个顶点位置分别为 $(0.25,0.25)$, $(0.25,0.45)$, $(0.75,0.45)$, $(0.75,0.25)$. 由此, 我们可以得到目标空间概念对应的决策树模型, 如图 2(b) 下方所示.

注意到, 原始空间和目标空间的两棵决策树具有很大程度的相似性, 原始空间的决策树只需在 (原始空间的) 叶节点 Y 上做进一步生长就可以得到目标空间的决策树. 这个例子直观地阐述了决策树模型重用的可能性.

由图 2(a) 和 (b) 启发, 当我们希望重用原始决策树模型 $\widetilde{\mathsf{T}}$ 配合少量目标空间标记数据 D 以学习目标空间概念时, 一个自然的操作是将目标空间的标记数据在原始决策树上继续生长直至完全. 我们记这种操作为伸展操作. 注意到, 如果只有伸展操作, 生成的决策树很容易对少量标记数据 D 过拟合, 而无法反映目标空间的真实概念. 因此, 我们引入收缩操作进行剪枝, 以防止过拟合, 从而更好地重用原始决策树模型. 总结起来, 本文中采用的结构伸展收缩算法, 主要包括伸展和收缩两个操作.

- 伸展操作: 利用当前数据块内的样本在原有决策树模型上继续伸展直至生长完全.
- 收缩操作: 自下而上的, 对于每个内部节点, 将维持两个错误率, 子树错误率及叶子错误率. 如果叶子错误率小于子树错误率, 则进行收缩操作, 即对决策树进行剪枝使当前节点为叶子节点; 反之, 则

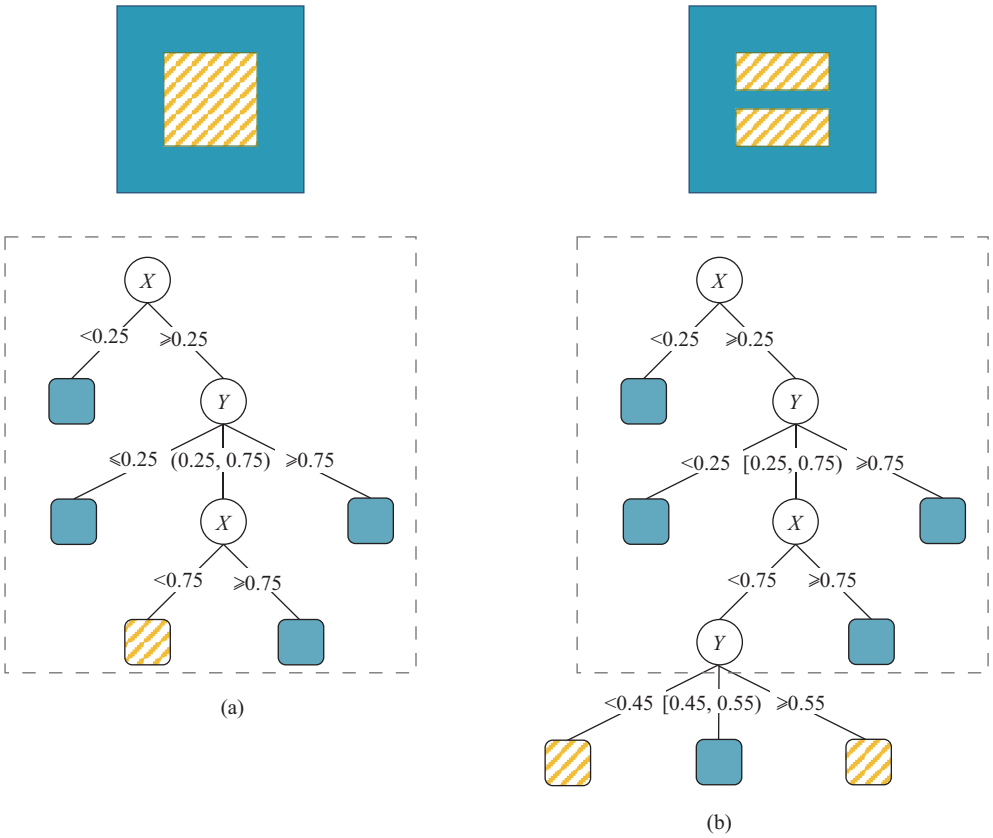

图 2 (网络版彩图) 决策树模型重用的直观示意图. (a) 原始空间的概念及对应的决策树模型; (b) 目标空间概念及对应的决策树模型

Figure 2 (Color online) Illustration of the decision tree model reuse mechanism. (a) shows the concept and decision tree model of the source domain; (b) demonstrates those of the target domain

不进行剪枝.

3.3 自适应权重调整: 乘法权重更新算法

第 3.2 小节主要探讨了当 CondorForest 算法从模型库中选定决策树之后, 如何重用此决策树模型以配合当前数据学习得到新的决策树模型. 这里遗留了一个问题: 算法应该如何在模型库选择好的模型? 实际上, 这要求算法要能够根据模型与当前数据的适配程度, 或称之为 "可重用性" (reusability), 进行权重调整. 特别是, 在流数据学习问题中, 这种权重更新需要是在线完成的, 即算法应根据每轮数据反馈进行模型权重调整.

本文采用经典的乘法权重更新 (multiplicative weight update, MWU) [36,37] 算法以进行自适应权重调整. 该技术广泛应用于在线学习、机器学习理论、理论计算机、博弈论和计算经济学等方面. 乘法权重更新算法有多种不同变种, 我们采用在线学习中常用的 Hedge 算法 [37~39]. 下面介绍具体做法.

在第 t 时刻, 模型库包含 K 个模型, 记为 $\mathcal{M} = \{\mathsf{T}_1, \ldots, \mathsf{T}_K\}$. 此时, 学习算法首先获得样本 $\boldsymbol{z}_t = (\boldsymbol{x}_t, y_t)$ 的属性 $\boldsymbol{x}_t$, 模型库中所有模型根据此做出预测 $\mathsf{T}_k(\boldsymbol{x}_t)$, $k = 1, \ldots, K$. 然后, 算法收到样本的真实标记 y_t, 从而每个模型将会遭受 $\ell(\mathsf{T}_k(\boldsymbol{x}_t), y_t)$ 的损失, 其中 $\ell : \mathbb{R} \times \mathcal{Y} \mapsto \mathbb{R}$ 为某一损失函数. 算法根据不同模型遭受的损失大小更新权重, 从而反映模型针对当前数据的可重用性, 具体更新公式

如下:

$$w_{t+1}^k = \frac{w_t^k \exp(-\eta \ell(\mathsf{T}_k(\boldsymbol{x}_t), y_t))}{\sum_{k=1}^K w_t^k \exp(-\eta \ell(\mathsf{T}_k(\boldsymbol{x}_t), y_t))}, \tag{1}$$

其中 $\eta > 0$ 为自适应权重调整算法的步长.

从上述权重更新公式, 我们可以看到每个模型的权重在每轮都会被调整, 当该模型在当前数据上表现良好 (即预测损失较小), 则权重会在原有的基础上调; 否则会下降. 注意最终所有的权重会对 K 个模型再进行归一化操作, 使其成为合法的概率分布.

上述权重更新公式非常简单有效. 事实上, 由更新公式 (1) 可知, 第 k 个模型最终的权重将正比于 $\exp(-\eta L_k)$, 其中 $L_k = \sum_{t \in S_k} \ell(\mathsf{T}_k(\boldsymbol{x}_t), y_t)$ 为模型 k 在当前数据块上的累计损失, 如下述命题.

命题1 (权重集中, weight concentration)　在数据块 k 内权重更新结束后, 模型权重将会集中到那些在当前数据块上累计损失小的历史模型.

因此, 通过乘法权重更新算法得到的权重可以表征历史模型和当前数据的适配程度, 即可重用性.

总结起来, CondorForest 算法根据第 3.1 小节描述的算法框架进行执行, 其中两个主要组件 (模型重用和自适应权重调整) 的具体流程可见 3.2 和 3.3 小节的介绍. 算法 1 给出了 CondorForest 算法的具体执行步骤, 其中第 4∼6 行, 算法收到样本属性, 并通过模型库 $\mathcal{M}$ 中的每个模型的预测进行加权从而得到最终的预测标记. 第 7 和 8 行, 算法获得样本真实标记并遭受损失, 从而利用第 3.3 小节介绍的自适应调整机制更新权重. 第 9∼13 行进行模型库更新, 其中使用第 3.2 小节介绍的模型重用方法得到新的决策树模型.

Algorithm 1 CondorForest

Input: Data stream $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T)\}$. Update period p; model pool size K; step size η; distribution change detector $\mathfrak{D}$.

Output: Predictive label $\widehat{y}_t$, $t = 1, \ldots, T$.

1: Use the initial data to initialize a decision-tree model T_1;
2: Initialize the model pool $\mathcal{M} = \{\mathsf{T}_1\}$ and weight $w_1 = 1$;
3: **for** $t = 1$ **to** T **do**
4: Receive the feature $\boldsymbol{x}_t$;
5: Each model of $\mathcal{M}$ makes the prediction: $\mathsf{T}_k(\boldsymbol{x}_t)$, where $k = 1, \ldots, |\mathcal{M}|$;
6: Make the prediction $\widehat{y}_t = \sum_{k=1}^{|\mathcal{M}|} w_t^k \mathsf{T}_k(\boldsymbol{x}_t)$;
7: Receive the ground-truth label y_t, and each model suffers $\ell(\mathsf{T}_k(\boldsymbol{x}_t), y_t)$;
8: Update the model weight according to (1);
9: **if** $(t \mod p) = 0$ **or** $\mathfrak{D}$ detects the distribution change **then**
10: Sample a decision tree model $\widetilde{\mathsf{T}}$ according to the weight distribution;
11: Learn a new model T_{new} via the decision tree model reuse, and then add it into the model pool $\mathcal{M}$;
12: Update the weight as the uniform distribution;
13: **end if**
14: **end for**

4 实验

本节通过在合成和真实数据集上进行实验, 将提出的 CondorForest 算法与多种分布变化数据流算法进行比对, 以验证本文提出算法的有效性.

数据集. 我们采用 8 个数据集, 其中包含 2 个合成数据集, CIR500G 和 SIN500G; 以及 6 个真实数据集, Luxembourg, Weather, GasSensor, Powersupply, Electricity, Covertype. 表 1 展示了所使用数据集的基本信息, 包括数据集名称、样本数目、维度和类别数目. CIR500G 合成数据集是 CIRCLE 数据集 [17] 的变种, 数据属性为 2 维, 决策边界为圆形, 并通过调整圆的半径模拟分布变化. 具体而言, 数据决策空间为 $x_1^2+x_2^2\leqslant r$, 其中 $r=\{3,2.5,2,2.5,3,3.5,4,3.5\}$, 变化周期为 500. SIN500G 合成数据集是 SINE 数据集 [17] 的变种, 数据属性为 2 维, 决策边界为正弦函数, 并通过调整 j 角度模拟分布变化. 具体而言, 数据决策空间为 $\sin(x_1+\theta)\leqslant x_2$, 其中 $\theta_0=0,\Delta\theta=\pi/60$, 变化周期为 500. 实验中所用到的真实数据集包含文本、气象、电费、森林覆盖率等多方面, 样本数目规模从 1900 到最多的 58 万多, 且包含二分类和多分类不同任务. 详细的介绍可以参考文献 [26] 的附录 1.

实验设置. CondorForest 算法中, 我们采用信息熵作为决策树划分准则, 使用第 3.2 小节中介绍的结构伸展收缩算法对决策树进行模型重用. 我们设置理论上的最大树深度为 1000 层, 在实际数据中生长出的决策树最大深度为 15 层上下. 我们采用准确率 (accuracy) 作为评价指标. 本文所提的 CondorForest 算法涉及到 3 个重要参数: 更新周期 p, 模型库大小 K, 步长 η. 在实验中, 我们设定默认更新周期 $p=100$, 模型库大小 $K=25$, 步长 $\eta=0.75$. 其他对比方法参数均使用其文章或代码中推荐的参数设置方式.

对比方法. 我们和以下 4 种对比方法进行对比. 首先是分布变化流数据学习的经典方法, DWM 算法 [9,10], 该方法是基于集成学习, 通过动态权重调整以删减更新模型库. 其次是一些利用模型迁移的方法: TIX (temporal inductive transfer) 算法 [40], 将历史模型的预测拼到当前数据作为额外的属性; DTEL (diversity and transfer-based ensemble learning) 算法 [41] 也采用决策树作为基分类器, 使用模型迁移将模型库中所有的历史决策树模型修正. 最后是和本文最相关的对比方法: CondorSVM (handling concept drift via model reuse) 算法 [26], 该算法采用支持向量机 (support vector machine, SVM) 作为基分类器, 并使用偏差正则化进行模型重用.

实验结果. 表 2 展示不同算法在 8 个数据集上的表现. 在每个数据集上, 我们均进行了 10 次测试运行, 并显示了平均正确率以及标准偏差. $\bullet$ ($\circ$) 表示在 95% 显著性水平的成对 t- 检验意义下, CondorForest 算法明显优于 (劣于) 对比方法. 表 2 显示, CondorForest 算法在 5 个数据集 (总计 8 个数据集) 上取得最优的性能, 这体现了 CondorForest 算法的有效性. 此外, 我们可以看到, 在两个非线性的合成数据集上, CondorForest 算法显著优于 CondorSVM 算法. 这是因为 CondorSVM 的基分类器是线性支持向量机, 因此无法很好地拟合这类非线性数据. 在其他 6 个真实数据集上, CondorForest 在其中 4 个数据集上取得了最优的性能. 在 Weather 和 GasSensor 这两个数据集上, CondorForest 的表现不如 CondorSVM, 这可能是因为这两个数据集的数据分布相对而言比较散, 线性可分性比较强, 因此更加适合采用基于 SVM 的方法进行分类. 值得注意的是, 总体而言, 在 6 个真实数据集上, CondorForest 和 CondorSVM 算法取得了最优, 优于其他的对比方法 (DWM, DTEL, TIX). 这两种算法均采用模型重用机制以抵御流数据中的分布变化, 这体现了模型重用机制的有效性.

鲁棒性分析. 为了进一步综合对比不同方法在不同数据集上的表现, 我们进行鲁棒性分析 (robustness analysis). 具体而言, 对于某一特定算法 $\mathcal{A}$, 在数据集 D 上的鲁棒性定义为算法 $\mathcal{A}$ 在数据集 D 上的准确率与所有方法中最低准确率的比值 [42], 即

$$r_{\mathcal{A}}(D)=\frac{\mathtt{acc}_{\mathcal{A}}(D)}{\min_{\alpha}\mathtt{acc}_{\alpha}(D)},$$

其中 $\mathtt{acc}_{\mathcal{A}}(D)$ 为算法 $\mathcal{A}$ 在数据集 D 上的准确率, $\min_{\alpha}\mathtt{acc}_{\alpha}(D)$ 为所有算法 (包括算法 $\mathcal{A}$ 及所有对比算法) 在数据集 D 上的准确率的最小值. 由上述定义可知, 在数据集 D 上表现最差的算法 $\mathcal{A}_0$ 的

表 1　实验中所用数据集的基本信息

Table 1　Basic statistics of datasets involved in the experiments

Dataset	# instance	Dim	# class	Dataset	# instance	Dim	# class
CIR500G	60000	3	2	GasSensor	4450	129	6
SINE500G	60000	2	2	Powersupply	29928	2	2
Luxembourg	1900	32	2	Electricity	45312	8	2
Weather	18159	8	2	Covertype	581012	54	2

表 2　在 2 个非线性合成数据集和 6 个真实数据集上的性能比较

Table 2　Performance comparisons on two non-linear synthetic datasets and six real-world datasets

Dataset	DWM	DTEL	TIX	CondorSVM	CondorForest
CIR500G	77.09 ± 0.71 •	79.03 ± 0.34 •	66.38 ± 0.85 •	68.41 ± 0.87	**79.60 ± 1.11**
SIN500G	66.99 ± 0.10 •	74.93 ± 0.34 ◦	62.73 ± 0.14 •	65.68 ± 0.12	73.98 ± 0.90
Luxembourg	90.42 ± 0.55 •	100.0 ± 0.00	90.99 ± 0.97 •	99.98 ± 0.03	**100.0 ± 0.00**
Weather	70.83 ± 0.49 •	68.92 ± 0.27 •	70.21 ± 0.33 •	**79.37 ± 0.26**	73.65 ± 0.66
GasSensor	76.61 ± 0.36 •	63.82 ± 3.64 •	43.40 ± 2.88 •	**81.57 ± 3.77**	76.25 ± 4.38
Powersupply	72.09 ± 0.29 •	69.90 ± 0.38 •	68.34 ± 0.16 •	72.82 ± 0.29	**73.23 ± 0.91**
Electricity	78.03 ± 0.17 •	81.05 ± 0.35 •	58.44 ± 0.71 •	84.73 ± 0.33	**87.88 ± 1.04**
Covertype	74.17 ± 0.87 •	69.43 ± 1.30 •	64.60 ± 0.89 •	89.58 ± 0.14	**91.35 ± 0.24**

鲁棒性为 1, 即 $r_{\mathcal{A}_0}(D)=1$; 其他的算法 $\mathcal{A}$ 的鲁棒性大于 1, 即 $r_{\mathcal{A}}(D)\geqslant 1$, 且值越大说明该算法在该数据集的表现越好. 因此, 假设有 n 个数据集 $D_1,\ldots,D_n$, 算法 $\mathcal{A}$ 的总体鲁棒性定义为该算法在所有数据集的鲁棒性的求和:

$$r_{\mathcal{A}}=\sum_{i=1}^{n} r_{\mathcal{A}}(D_i).$$

该鲁棒性值越大说明算法的表现越好.

图 3 展示了 CondorForest 算法与其他对比算法在 8 个数据集上的鲁棒性对比. 可以看到 CondorForest 算法的总体鲁棒性在所有算法中排名第一, 体现了本文提出的算法在解决分布变化流数据学习任务上的有效性. CondorSVM 算法的性能表现紧随其后. 值得注意的是, CondorSVM 算法与本文的 CondorForest 算法均基于集成学习框架, 并通过模型重用机制处理分布变化流数据. 因此, 这在一定程度上体现模型重用机制配合集成学习框架在分布变化流数据学习任务上的有效性. 相比而言, TIX 算法的性能表现并不尽如人意, 这可能是因为 TIX 算法没有使用集成学习框架, 因而无法很好地抵御流数据中出现的分布变化.

参数敏感度分析. 我们研究 CondorForest 算法中参数对最终性能的影响. 该算法有 3 个非常重要的参数: 更新周期 (update period or epoch size) p、模型库大小 (model pool size) K 以及步长 (step size) η. 我们选取了 4 个真实数据集 (其他数据集表现类似) 测试并汇报 CondorForest 算法在不同参数选择下的表现. 从图 4 可以看出, 随着更新周期、模型库大小和步长这 3 个参数变化, 在很大范围内我们的算法性能并没有受到较大的波动. 这体现了 CondorForest 算法的稳定性, 对参数变化不敏感. 在实验中, 我们设定默认更新周期 $p=100$, 模型库大小 $K=25$, 步长 $\eta=0.75$.

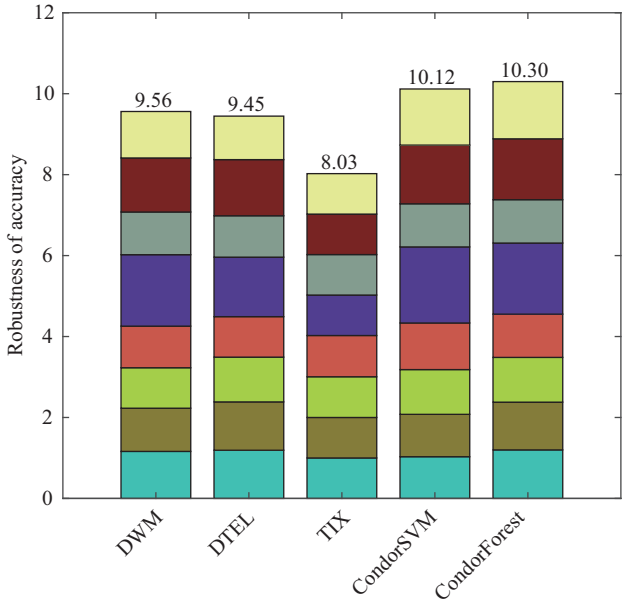

图 3 (网络版彩图) 鲁棒性分析

Figure 3 (Color online) Robustness analysis

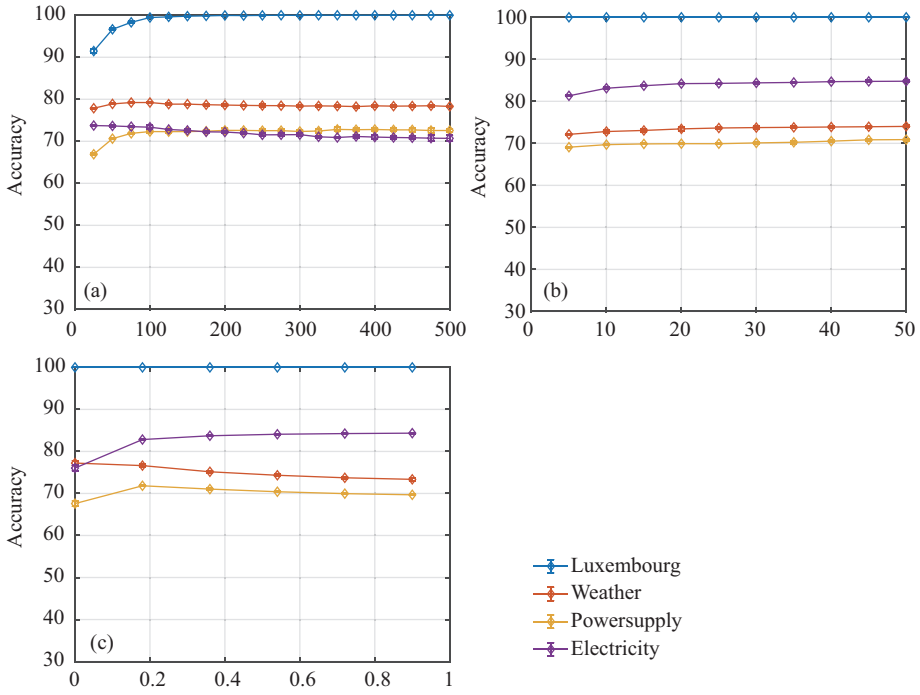

图 4 (网络版彩图) 在不同数据集上的参数敏感度分析

Figure 4 (Color online) Parameter sensitivity analysis on different datasets: (a) Update period p; (b) model pool size K; (c) step size η

5 结语

分布变化的流数据学习是非常重要的机器学习任务, 在很多真实场景中有广泛的应用. 本文首先

明确了分布变化流数据学习的基本假设, 即 “历史数据中包含对当前预测有价值的知识”. 基于此, 我们显式建模当前预测和历史数据之间的关系, 自适应挖掘历史数据中对当前预测有用的知识. 同时, 由于决策树的灵活性以及与集成学习良好的适配性, 本文采用决策树模型, 通过决策树模型重用方法利用历史数据的知识以辅助当前时刻的学习, 从而对抗流数据中可能出现的分布变化. 我们通过在合成和真实数据集上的实验验证了提出算法的有效性.

流数据学习是一个非常重要的研究课题, 在机器学习算法实践和理论方面都具有重要意义. 本文考虑分布发生变化的场景, 并设计行之有效的算法. 如何从理论上刻画分布变化流数据的可学习性质是一个非常困难且重要的课题. 此外, 在真实应用中, 流数据的属性空间可能会不断发生变化 [43, 44], 新的类别可能出现 [45, 46] 等, 这些也是亟待进一步研究的问题.

参考文献

1 LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521: 436–444
2 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 770–778
3 Devlin J, Chang M, Lee K, et al. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2019. 4171–4186
4 Bifet A, Gavaldà R. Learning from time-changing data with adaptive windowing. In: Proceedings of the 7th SIAM International Conference on Data Mining (SDM), 2007. 443–448
5 Kuncheva L I, Žliobaitė I. On the window size for classification in changing environments. Intell Data Anal, 2009, 13: 861–872
6 Ross G J, Adams N M, Tasoulis D K, et al. Exponentially weighted moving average charts for detecting concept drift. Pattern Recogn Lett, 2012, 33: 191–198
7 Koychev I. Gradual forgetting for adaptation to concept drift. In: Proceedings of ECAI 2000 Workshop on Current Issues in Spatio-Temporal Reasoning, 2000. 101–106
8 Zhao P, Wang X Q, Xie S Y, et al. Distribution-free one-pass learning. IEEE Trans Knowl Data Eng, 2019. doi: 10.1109/TKDE.2019.2937078
9 Kolter J Z, Maloof M A. Dynamic weighted majority: a new ensemble method for tracking concept drift. In: Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM), 2003. 123–130
10 Kolter J Z, Maloof M A. Dynamic weighted majority: an ensemble method for drifting concepts. J Mach Learn Res, 2007, 8: 2755–2790
11 Gomes H M, Barddal J P, Enembreck F, et al. A survey on ensemble learning for data stream classification. ACM Comput Surv, 2017, 50: 1–36
12 Gama J, Žliobaitė I, Bifet A, et al. A survey on concept drift adaptation. ACM Comput Surv, 2014, 46: 1–37
13 Anagnostopoulos C, Tasoulis D K, Adams N M, et al. Online linear and quadratic discriminant analysis with adaptive forgetting for streaming classification. Stat Analy Data Min, 2012, 5: 139–166
14 Jaber G, Cornuéjols A, Tarroux P. Online learning: Searching for the best forgetting strategy under concept drift. In: Proceedings of the 20th International Conference on Neural Information Processing (ICONIP), 2013. 400–408
15 Zhou Z-H. Ensemble Methods: Foundations and Algorithms. Boca Raton: Chapman & Hall/CRC Press, 2012
16 Kolter J Z, Maloof M A. Using additive expert ensembles to cope with concept drift. In: Proceedings of the 22nd International Conference on Machine Learning (ICML), 2005. 449–456
17 Elwell R, Polikar R. Incremental learning of concept drift in nonstationary environments. IEEE Trans Neural Netw, 2011, 22: 1517–1531
18 Schapire R E, Freund Y. Boosting: Foundations and Algorithms. Cambridge: The MIT Press, 2012
19 Zhou Z-H. Learnware: on the future of machine learning. Front Comput Sci, 2016, 10: 589–590
20 Schölkopf B, Herbrich R, Smola A J. A generalized representer theorem. In: Proceedings of the 14th Annual Conference Computational Learning Theory (COLT), 2001. 416–426
21 Yang J, Yan R, Hauptmann A G. Cross-domain video concept detection using adaptive svms. In: Proceedings of the

15th ACM International Conference on Multimedia (ACM MM), 2007. 188–197
22 Tommasi T, Orabona F, Caputo B. Safety in numbers: learning categories from few examples with multi model knowledge transfer. In: Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010. 3081–3088
23 Tommasi T, Orabona F, Caputo B. Learning categories from few examples with multi model knowledge transfer. IEEE Trans Pattern Anal Mach Intell, 2014, 36: 928–941
24 Kuzborskij I, Orabona F. Stability and hypothesis transfer learning. In: Proceedings of the 30th International Conference on Machine Learning (ICML), 2013. 942–950
25 Kuzborskij I, Orabona F. Fast rates by transferring from auxiliary hypotheses. Mach Learn, 2017, 106: 171–195
26 Zhao P, Cai L W, Zhou Z-H. Handling concept drift via model reuse. Mach Learn, 2020, 109: 533–568
27 Du S S, Koushik J, Singh A, et al. Hypothesis transfer learning via transformation functions. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 574–584
28 Wu X-Z, Zhou Z-H. Model reuse with domain knowledge. Sci Sin Inform, 2017, 47: 1483–1492 [吴西竹, 周志华. 领域知识指导的模型重用. 中国科学: 信息科学, 2017, 47: 1483–1492]
29 Segev N, Harel M, Mannor S, et al. Learn on source, refine on target: a model transfer learning framework with random forests. IEEE Trans Pattern Anal Mach Intell, 2017, 39: 1811–1824
30 Ye H J, Zhan D C, Jiang Y, et al. Rectify heterogeneous models with semantic mapping. In: Proceedings of the 35th International Conference on Machine Learning (ICML), 2018. 1904–1913
31 Wu X-Z, Liu S, Zhou Z-H. Heterogeneous model reuse via optimizing multiparty multiclass margin. In: Proceedings of the 36th International Conference on Machine Learning (ICML), 2019. 6840–6849
32 Li N, Tsang I W, Zhou Z-H. Efficient optimization of performance measures by classifier adaptation. IEEE Trans Pattern Anal Mach Intell, 2013, 35: 1370–1382
33 Ding Y-X, Zhou Z-H. Preference based adaptation for learning objectives. In: Proceedings of Advances in Neural Information Processing Systems, 2018. 7839–7848
34 Wu X D, Kumar V, Ross Q J, et al. Top 10 algorithms in data mining. Knowl Inf Syst, 2008, 14: 1–37
35 Chen T, Guestrin C. Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016. 785–794
36 Arora S, Hazan E, Kale S. The multiplicative weights update method: a meta-algorithm and applications. Theor Comput, 2012, 8: 121–164
37 Cesa-Bianchi N, Lugosi G. Prediction, Learning, and Games. Cambridge: Cambridge University Press, 2006
38 Littlestone N, Warmuth M K. The weighted majority algorithm. In: Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 1989. 256–261
39 Freund Y, Schapire R E. A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci, 1997, 55: 119–139
40 Forman G. Tackling concept drift by temporal inductive transfer. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2006. 252–259
41 Sun Y, Tang K, Zhu Z X, et al. Concept drift adaptation by exploiting historical knowledge. IEEE Trans Neural Netw Learn Syst, 2018, 29: 4822–4832
42 Vlachos M, Domeniconi C, Gunopulos D, et al. Non-linear dimensionality reduction techniques for classification and visualization. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), 2002. 645–651
43 Hou B-J, Zhang L, Zhou Z-H. Learning with feature evolvable streams. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 1417–1427
44 Hou C P, Zhou Z-H. One-pass learning with incremental and decremental features. IEEE Trans Pattern Anal Mach Intell, 2018, 40: 2776–2792
45 Mu X, Ting K M, Zhou Z-H. Classification under streaming emerging new classes: a solution using completely-random trees. IEEE Trans Knowl Data Eng, 2017, 29: 1605–1618
46 Cai X Q, Zhao P, Ting K M, et al. Nearest neighbor ensembles: an effective method for difficult problems in streaming classification with emerging new classes. In: Proceedings of the 19th International Conference on Data Mining (ICDM), 2019

Learning from distribution-changing data streams via decision tree model reuse

Peng ZHAO & Zhi-Hua ZHOU*

National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023*, China*
* Corresponding author. E-mail: zhouzh@lamda.nju.edu.cn

Abstract In many real-world applications, data are collected in the form of streams. As a result of the evolving nature of dynamic environments, the distribution of data streams generally changes over time. Such distribution changes hinder the application of conventional machine learning approaches because the fundamental assumption of independent and identical distribution does not hold in these scenarios. This paper proposes an algorithm based on the decision tree model reuse mechanism for learning from distribution-changing data streams. The proposed algorithm is essentially an online ensemble method that maintains a model pool and updates it by performing decision tree model reuse. The main idea is to exploit the useful knowledge in historical data to help resist the negative effects of distribution changes. We validate the effectiveness of the proposed approach through experiments on synthetic and real-world datasets.

Keywords machine learning, distribution change, data stream, model reuse, ensemble methods, dynamic environments

Peng ZHAO received his B.S. degree from Tongji University, Shanghai, China, in 2016. He is currently working toward his Ph.D. degree with the National Key Lab for Novel Software Technology in Nanjing University under the supervision of Prof. Zhi-Hua ZHOU. His research interest is mainly focused on machine learning. He is currently working on robust learning in nonstationary environments.

Zhi-Hua ZHOU received his Ph.D. degree in Computer Science from Nanjing University, China, in 2000. Currently, he is a professor at Nanjing University. His research interests mainly include artificial intelligence, machine learning, and data mining.