软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(4):1637−1664 [doi: 10.13328/j.cnki.jos.007237] [CSTR: 32375.14.jos.007237]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

机器遗忘综述*

李梓童, 孟小峰, 王雷霞, 郝新丽

(中国人民大学 信息学院, 北京 100872)

通信作者: 孟小峰, E-mail: xfmeng@ruc.edu.cn

摘　要: 近年来, 机器学习在人们日常生活中应用愈发广泛, 这些模型在历史数据上进行训练, 预测未来行为, 极大地便利了人们生活. 然而, 机器学习存在隐私泄露隐患: 当用户不希望个人数据被使用时, 单纯地把其数据从训练集中删去并不够, 已训练好的模型仍包含用户信息, 可能造成隐私泄露. 为了解决这一问题, 让机器学习模型“遗忘”该用户个人数据, 最简单的方法是在不包含其数据的训练集上重新训练, 此时得到的新模型必定不包含个人数据的信息. 然而, 重新训练往往代价较大, 成本较高, 由此产生“机器遗忘”的关键问题: 能否以更低的代价, 获取与重新训练模型尽可能相似的模型. 对研究这一问题的文献进行梳理归纳, 将已有机器遗忘方法分为基于训练的方法、基于编辑的方法和基于生成的方法这 3 类, 介绍机器遗忘的度量指标, 并对已有方法进行测试和评估, 最后对机器遗忘作未来展望.

关键词: 机器学习; 机器遗忘; 深度学习; 隐私保护

中图法分类号: TP18

中文引用格式: 李梓童, 孟小峰, 王雷霞, 郝新丽. 机器遗忘综述. 软件学报, 2025, 36(4): 1637–1664. http://www.jos.org.cn/1000-w9825/7237.htm

英文引用格式: Li ZT, Meng XF, Wang LX, Hao XL. Survey on Machine Unlearning. Ruan Jian Xue Bao/Journal of Software, 2025, 36(4): 1637–1664 (in Chinese). http://www.jos.org.cn/1000-9825/7237.htm

Survey on Machine Unlearning

LI Zi-Tong, MENG Xiao-Feng, WANG Lei-Xia, HAO Xin-Li

(School of Information, Renmin University of China, Beijing 100872, China)

Abstract: Machine learning has become increasingly prevalent in daily life. Various machine learning methods are proposed to utilize historical data for making predictions, making people’s life more convenient. However, there is a significant challenge associated with machine learning-privacy leakage. Mere deletion of a user’s data from the training set is not sufficient for avoiding privacy leakage, as the trained model may still harbor this information. To tackle this challenge, the conventional approach entails retraining the model on a new training set that excludes the data of the user. However, this method can be costly, prompting the exploration for a more efficient way to “unlearn” specific data while yielding a model comparable to a retrained one. This study summarizes the current literature on this topic, categorizing existing unlearning methods into three groups: training-based, editing-based, and generation-based methods. Additionally, various metrics are introduced to assess unlearning methods. The study also evaluates current unlearning methods in deep learning and concludes with future research directions in this field.

Key words: machine learning; machine unlearning; deep learning; privacy protection

机器学习是指采用一定数学模型, 从大量数据 (训练集) 中学习模式或知识, 并将这些模式或知识用于预测未知情况的方法. 作为一项新兴技术, 它已深深融入人们的日常生活, 如基于机器学习的推荐系统可以为不同用户进行个性化推荐, 从而提高推荐成功率. 然而, 这一技术的广泛使用, 也产生了隐私泄露的隐患.

* 基金项目: 国家自然科学基金 (61941121, 91846204, 6217242)
收稿时间: 2023-03-17; 修改时间: 2024-04-29; 采用时间: 2024-06-11; jos 在线出版时间: 2024-11-18
CNKI 网络首发时间: 2024-11-20

机器学习的重要特点之一是所训练的模型会“记住”训练集信息. 由于机器学习借助历史数据预测未来行为或倾向, 其模型难免包含训练集中用户个人信息, 存在信息泄露的风险[1]. 以基于协同过滤的推荐系统为例[2], 假设某推荐系统根据用户群体的书籍阅读历史来建模计算用户之间的相似度, 从而为各用户推荐相似用户阅读的书籍. 在这种情况下, 若某用户 A 试图删除自己与艾滋病主题相关的阅读记录, 而对应的用户相似度模型没有做出相应改变, 则攻击者仍可根据与用户 A 相似的用户所拥有的阅读记录 (这些用户很可能阅读了艾滋病主题的书籍), 推断出用户 A 也曾阅读过艾滋病主题的书籍. 因此, 在机器学习场景下, 仅从数据集中删除用户数据, 并不足以保护用户隐私, 用户数据很可能被内化到了机器学习模型中, 在本例中即为用户相似度模型. 此外, 对自然语言模型进行增量训练时, 可通过训练前后模型的差异推断出模型新使用的训练数据[3]; 通过成员推理攻击, 攻击者能判断某条数据是否在模型训练集中[4], 等等. 以上现象均说明, 模型会记忆训练集信息, 且不会随着时间流逝而自行“遗忘”, 易引起隐私泄露.

机器学习隐私问题越发突出, 人们对个人信息的保护也越发重视. 欧盟委员会在 2012 年首次提出被遗忘权 (the right to be forgotten), 指出个人出于合法目的可要求数据收集者删除其数据, 强调赋予用户处理个人隐私数据的合法权利[5]. 许多国家开始重视大数据下的隐私问题, 我国于 2018 年生效的《信息安全技术个人信息安全规范》指出, 用户应有权删除个人数据[6]. 同年, 欧盟公布的《通用数据保护条例》(General Data Protection Regulation, GDPR) 正式生效, 该条例指出, 自动化算法 (包括机器学习系统) 需保证用户拥有被遗忘权[7].

为了支持被遗忘权, 模型拥有者需确定模型具有遗忘的能力[8]. 但如前文所述, 由于模型仍可能泄露用户信息, 单从训练集里删除用户个人数据并不够. 即使用户个人数据被删除, 这些数据对模型的影响仍然存在, 产生隐私泄露的隐患.

由此产生了机器遗忘 (machine unlearning) 的概念. 当遗忘某些数据的请求到来时, 机器遗忘不仅要求删除训练集中的这些数据, 还要求删除它们对模型的影响[9], 让模型表现得与从未使用这些数据进行训练的结果相近.

让模型遗忘某些数据, 最简单的方法是在不包含这些数据的训练集上重新训练模型. 然而, 很多情况下, 重新训练模型代价巨大, 如在 15G 训练集上训练一个参数量为 110M 的谷歌 BERT 模型, 花费在 2000–5 万美元之间[10], 且耗时较长. 机器遗忘的核心问题是: 在训练集中移除某些数据后, 如何以代价低于重新训练的方式, 得到与重新训练结果尽可能接近的模型[11], 其本质是获取近似模型. 若有代价更小的方法达到模型遗忘的效果, 用户希望模型遗忘个人数据的需求就更容易被满足, 其隐私也就能得到更好的保护.

最后, 除了隐私保护外, 机器遗忘还有其他应用, 如: 在机器学习模型遇到后门攻击 (backdoor attack) 或存在被污染样本时, 使用机器遗忘来除去恶意样本, 提高模型维护效率[12–14]; 在训练集存在冗余时, 用户对模型语料进行压缩, 可通过机器遗忘来高效地获得压缩训练集后的模型.

当前有少量针对机器遗忘的综述文献[15–18], 本文与现有综述的比较如表 1 所示.

表 1 本文与现有综述的比较

综述	方法介绍			评估指标			实验评估		研究前景		
	已知条件	开销分析	优缺点分析	速度	可用性	完成度	方法性能分析	数据影响分析	数据类型	多方计算	评估标准
本文	√	√	√	√	√	√	√	√	√	√	√
文献[15]	×	√	√	√	√	√	×	×	×	√	√
文献[16]	×	×	√	×	×	×	×	×	√	×	√
文献[17]	×	√	√	√	×	√	×	×	×	×	×
文献[18]	√	×	√	√	√	√	×	×	√	√	×

首先, 现有综述对机器遗忘方法使用时产生的开销分析 (包括时间开销与空间开销) 并不充分, 而本文弥补了这一不足. 其次, 目前很少有综述对机器遗忘方法做统一实验, 以对各类方法进行评估测试, 而本文进行了实验比较, 以便用户对方法性能有更直观的观察. 此外, 我们还探讨了机器遗忘未来值得研究的问题, 包括数据多样化和计算多方化等场景下的机器遗忘方法设计, 提出了具体的研究方向. 本文深入探索了机器遗忘领域的定义方式、

© 中国科学院软件研究所 http://www.jos.org.cn

现有方法和度量指标, 并由此提出了该领域研究展望, 主要贡献如下.

● 本文对当前机器遗忘方法进行系统的划分, 根据现有方法的设计思想和使用技术, 把它们分为基于训练的方法、基于编辑的方法和基于生成的方法这 3 类. 本文分别介绍了各类别下的具体方法, 并对它们的优缺点、适用场景和使用时需要的已知条件进行总结归纳.

● 本文进行了大规模实验, 对部分机器遗忘方法进行评估测试. 目前很少有工作在统一深度学习训练任务和度量指标下对机器遗忘方法进行实验比较, 对此本文设置了统一的训练任务, 从上文提到的各类方法中选择部分有代表性的方法进行实验, 填补了对各方法进行实验比较的空白.

● 本文对机器遗忘领域的未来研究方向进行总结和展望. 本文立足现有研究成果、实验结果和生活实际, 从遗忘数据多样化、遗忘计算多方化和遗忘评估标准化这 3 个角度对未来研究方向进行了总结, 为未来该领域的研究提供参考和启发.

本文第 1 节介绍问题定义. 第 2、3、4 节分别介绍机器遗忘的 3 类方法. 第 5 节介绍机器遗忘的度量指标. 第 6 节对机器遗忘方法进行测试和实验. 第 7 节介绍未来的研究方向. 第 8 节为结束语.

1 机器遗忘概述

1.1 机器遗忘定义

机器学习意图让模型“记住”数据, 其过程概括如下: 记原始训练集为包含了 n 个数据点的集合 $D=\{(x_i,y_i)\}_{i=1}^{n}$, 使用模型为 $M(\theta)$ (其中 θ 为模型参数), 损失函数为 $L=L(\theta,D)$. 机器学习通过梯度下降等优化方法寻找使损失函数值最小化的参数 θ^*, 最终得到模型 $M(\theta^*)$. 模型训练好后, 就对训练集外的数据点进行预测, 即对 $x'\notin D$, 给出相应的 $M(x'|\theta^*)=y'$.

与机器学习相反, 机器遗忘意图让模型“遗忘”数据. 令 x^* 为遗忘数据集, 它是原始数据集 D 的子集, 其特征空间与包含的样本量均为 D 的子集. 机器遗忘希望模型忘记数据 x^* 所包含的信息, 实现该目标的简单方法是在不包含 x^* 的剩余数据集 $D\backslash x^*$ 上重新训练模型, 由此得到的模型记作 M_{retrain}, 此时 M_{retrain} 固然不包含 x^* 的信息.

然而, 重新训练代价往往较大. 机器遗忘的核心问题在于: 如何避免较大的开销, 而得到与 M_{retrain} 近似的模型. 假设使用机器遗忘方法获得了新模型 M_{unlearn}, 则机器遗忘目的是使 M_{unlearn} 和 M_{retrain} 尽可能地接近, 从而得到 M_{unlearn} 的开销小于重新训练的开销. 在已有文献中, 多以获取模型的用时作为开销, 而在现实生活中, 开销还包括空间开销和金钱开销等. 机器遗忘形式化定义如下.

定义 1. 给定模型 M, 原始数据集 D, 遗忘数据集 x^*, 开销计算函数 $Cost(M)$ 和模型相似度函数 $Sim(M_1,M_2)$, $Sim(M_1,M_2)$ 恒正, M_1 和 M_2 越接近, 则 $Sim(M_1,M_2)$ 越小. 在剩余数据集 $D\backslash x^*$ 上训练得到 M_{retrain}, 通过机器遗忘方式获得模型 M_{unlearn}, 则希望 $Cost(M_{\text{unlearn}})<Cost(M_{\text{retrain}})$ 且 $Sim(M_1,M_2)\to 0$.

对于如何衡量 M_{retrain} 和 M_{unlearn} 之间的相似度, 现有文献方法不一: 有的使用模型参数距离衡量相似度, 即两个模型在参数上范数差越小, 就认为两者越接近[11,19]; 有的使用模型输出分布距离衡量相似度, 即当 M_{retrain} 和 M_{unlearn} 输出分布距离小于某个阈值时, 就认为已足够相似[20,21]. 提高相似度是机器遗忘的目标之一. 目前不同文献对相似度的定义仍不统一 (具体可见第 5 节和第 7 节), 各自提出的方法也是在各自的相似度定义下实现遗忘.

数据清洗和持续学习是与机器遗忘相关度较高的研究领域, 本文在此处对这两个领域与机器遗忘之间的关联加以辨析. 数据清洗是指从原始数据中清除含有缺失值或异常值等的“脏”数据[22], 在真实数据集中, 这些“脏”数据不仅增加了模型的训练负担, 还可能对模型的性能产生负面影响. 数据清洗与机器遗忘的最大不同在于: 数据清洗发生在模型训练之前, 通过提高数据质量, 改善模型性能; 机器遗忘发生在模型训练之后, 试图在模型训练完成后, 从模型中删除特定数据的影响. 然而, 模型训练完成后, 机器遗忘也可用于清除脏数据对模型的影响, 在这种情况下, 机器遗忘与数据清洗同样都是以改善模型性能为目的.

持续学习研究模型在一系列任务中不断学习的过程. 假设当模型学习新任务时, 它无法访问以前的任务数据, 持续学习的目标是尽可能少地遗忘先前任务的数据, 并最大限度地调整先前任务的知识以有助于学习新任

© 中国科学院软件研究所 http://www.jos.org.cn

务[23]. 机器遗忘与持续学习的主要区别如下: (1) 两者处理对象的粒度不同, 机器遗忘是针对数据样本, 而持续学习是针对学习任务; (2) 机器遗忘旨在让模型忘记特定数据隐含的知识, 而持续学习则旨在让模型记住先前数据的知识. 由于持续学习的目标是对新知识的吸收和旧知识的保留, 机器遗忘同样可用于改进持续学习算法, 例如, 人们可通过逐步遗忘一些不太重要或已过时的知识, 为模型学习新知识腾出空间, 从而在不断变化的学习任务中保持灵活性和适应性.

1.2 主要研究方向

目前机器遗忘领域的研究集中在方法上. 根据设计思想和使用技术, 本文将机器遗忘方法分为 3 类: 基于训练、基于编辑和基于生成的机器遗忘方法. 各类别代表方法见表 2.

表 2 机器遗忘主要研究方向

研究方向	示例
基于训练的方法	针对朴素贝叶斯的机器遗忘方法[9]、针对岭回归的机器遗忘方法[2]、Dare[24]、HedgeCut[25]、SISA[26]
基于编辑的方法	Class Clown[27]、针对可变SVM的机器遗忘方法[28]、CR[11]、K-priors[29]、Linear Filtration[30]、针对深度学习的遗忘方法[31]、ERM[32]
基于生成的方法	针对逻辑回归的机器遗忘方法[33]、针对深度学习的遗忘方法[31]、OptLearn[34]

基于训练的方法往往涉及模型训练过程, 如利用在原训练集上训练产生的参数和梯度等中间结果来进行遗忘. 这类方法用时较长, 但 M_{unlearn} 性能更有保证. 基于编辑的方法为了在已有模型的基础上做改动, 如利用牛顿法对数据集变动后的模型参数进行近似求解, 得到与重新训练模型接近的模型. 这类方法较为快速, 但 M_{unlearn} 性能可能不佳. 基于生成的方法是指使用对抗模型等生成 M_{unlearn} 参数. 一种机器遗忘方法并非只使用 1 种思路, 而可以是多种思路的结合, 如 Kim 等人[31]就同时使用了基于编辑和基于生成的方法, 后文将分别介绍这两类方法. 目前机器遗忘领域多数文献均为方法上的探索, 此外, 还有如何对机器遗忘进行全面定义[35], 设计刻意让模型重新训练的投毒攻击[36], 考虑机器遗忘中“过度遗忘”的潜在危害[37]和如何自行生成 $D\backslash x^*$ [38]等较为分散的研究.

在第 2、3、4 节中, 本文将分别对 3 类方法进行介绍. 表 3 是本文用到的符号及其含义.

表 3 符号表

符号	含义
D	原始数据集
x^*	遗忘数据集
$D\backslash x^*$	剩余数据集
M_0	在原始数据集上训练得到的模型
M_{retrain}	在剩余数据集上重新训练得到的模型
M_{unlearn}	在剩余数据集上通过机器遗忘得到的模型

本文将 3 类方法中已公开代码的链接整理到表 4 中, 以便后续研究者查阅.

表 4 主要方法与代码

类别	方法名称	代码链接
基于训练的方法	针对岭回归的机器遗忘方法[2]	https://github.com/schelterlabs/projects-amnesia
	Dare[24]	https://github.com/jjbrophy47/dare_rf
	HedgeCut[25]	https://github.com/schelterlabs/hedgecut
	SISA[26]	https://github.com/cleverhans-lab/machine-unlearning
	DeltaGrad[39]	https://github.com/thuwuyinjun/DeltaGrad
	Unrolling SGD[21]	https://github.com/cleverhans-lab/unrolling-sgd
	针对K-means的机器遗忘方法[40]	https://github.com/tginart/deletion-efficient-K-means

表 4 主要方法与代码 (续)

类别	方法名称	代码链接
基于编辑的方法	针对MCMC的机器遗忘方法[41]	https://github.com/fshp971/mcmc-unlearning
	CR[11]	https://github.com/facebookresearch/certified-removal
	K-priors[29]	https://github.com/team-approx-bayes/kpriors
	LCODEC[42]	https://github.com/vsingh-group/LCODEC-deep-unlearning
	ERM[32]	https://github.com/ChrisWaites/descent-to-delete
	PRU[43]	https://github.com/zleizzo/datadeletion
	SelectiveForgetting[20]	https://github.com/AdityaGolatkar/SelectiveForgetting
	Unlearnable[44]	https://github.com/HanxunH/Unlearnable-Examples
基于生成的方法	BIF[45]	https://github.com/fshp971/BIF
	SCRUB[46]	https://github.com/Meghdad92/SCRUB

2 基于训练的机器遗忘

基于训练的机器遗忘方法是指通过模型训练来实现遗忘的方法. 这类方法往往对重新训练的过程进行划分, 通过对部分数据重新训练或从某个轮次起开始重新训练, 使得训练开销低于完全重新训练, 而得到的机器遗忘模型也与重新训练模型较为相似. 该方法可细分为继续计算和模型分解两类.

2.1 继续计算

继续计算是指保存训练 M_0 时的中间结果, 在遗忘请求到来时, 从中间结果开始继续训练. 继续计算类方法保存的中间结果包括朴素贝叶斯中的样本计数和、深度学习中的梯度和参数等. 当遗忘时, 这类方法在中间结果上继续计算, 减小时间开销, 但存在额外的空间开销. 这类方法设计的关键在于从获取模型的过程中挖掘可以重用的数据, 将这些数据保存下来, 在遗忘请求到来时进行修改. 传统机器学习中的分类算法 (如朴素贝叶斯和岭回归)、聚类算法 (如 K-means 聚类) 和深度学习中都有对应的继续计算类机器遗忘方法.

2.1.1 朴素贝叶斯

朴素贝叶斯通过生成数据点属于各个标签的后验概率来对数据分类. 对于一个有 $f_1,\ldots,f_k$ 个特征的样本, 该算法先计算样本属于某个标签 y^* 的后验概率 $P(y^*|f_1,\ldots,f_k)$, 选择后验概率最大的标签作为分类结果. 其中, 后验概率的计算公式为:

$$P(y^*|f_1,\ldots,f_k)=\frac{P(y^*)\prod_{i=0}^{k}P(f_i|y^*)}{\prod_{i=0}^{k}P(f_i)} \tag{1}$$

朴素贝叶斯对应的机器遗忘方法将公式 (1) 右边出现的概率转化为由样本计数表示的计算, 若要遗忘某个样本, 则在涉及该样本的计数上减去对应的数量. 例如 $P(f_i|y^*)$, 该值可通过用具有特征 f_i 且标签为 y^* 的样本数 (记作 $N^*_{f_i,y}$) 除以标签为 y^* 的样本数 (记作 N^*_y) 得到, 即 $P(f_i|y^*)=N^*_{f_i,y}/N^*_y$. 若遗忘 1 个具有该特征和标签的样本, 对应的概率变为 $P(f_i|y^*)=N^*_{f_i,y}-1/N^*_y-1$. 其他概率如 $P(y^*),P(f_i)$ 均可转化为由样本计数表示的形式. 遗忘 x^* 时, 改变相应的计数, 重新计算的后验概率就作为遗忘后的概率[2,9].

Cao 等人[9]研究普遍场景下朴素贝叶斯的遗忘方法, 而 Parne 等人[47]则具体针对使用朴素贝叶斯检测垃圾邮件的场景, 研究如何遗忘垃圾邮件 (恶意样本), 并使用卡方检验来验证遗忘与否. 这种利用样本计数进行遗忘的方法只取决于 x^*, 遗忘得到的后验概率与重新训练相同. 该方法避免了重新训练, 但需要额外的空间来存储样本计数的中间结果.

© 中国科学院软件研究所 http://www.jos.org.cn

2.1.2 岭回归

岭回归是一种有偏估计回归方法. 在岭回归中, 给定训练集 $[X,y]$, 模型参数 θ, 则有 $X\theta=y$. 若损失函数为 $\|X\theta-y\|^2+\|\alpha\boldsymbol{I}\theta\|^2$, 其中, $\boldsymbol{I}$ 为单位矩阵, $\|\alpha\boldsymbol{I}\theta\|^2$ 为正则项, 则模型参数 θ 计算方式如下:

$$\theta=(X^{\mathrm{T}}X+\alpha\boldsymbol{I})^{-1}X^{\mathrm{T}}y \tag{2}$$

假设遗忘训练集中第 u 个数据点, 则更新后参数为:

$$\theta_u=(X^{\mathrm{T}}X-X_u^{\mathrm{T}}X_u+\alpha\boldsymbol{I})^{-1}(X^{\mathrm{T}}y-X_uy_u) \tag{3}$$

岭回归对应的机器遗忘方法[2]保存了 M_0 训练时的中间结果: $X^{\mathrm{T}}y$ 和 QR 分解结果 $qr(X^{\mathrm{T}}X+\alpha\boldsymbol{I})$. 若要遗忘训练集中某个样本, 则在中间结果上计算 M_{unlearn}.

2.1.3 K-means 聚类

K-means 聚类将数据点划分到各个聚类, 使每个点都属于离它最近的聚类. K-means 聚类具体过程为: 随机选择 k 个聚类中心 $c_1,\ldots,c_k$ 将各数据点分配给距离最小的聚类中心, 再根据聚类中所有数据点的平均值更新聚类中心 c_i, 重复上述步骤, 直到各聚类中心不再改变为止.

K-means 聚类对应的机器遗忘方法把聚类中心的计算过程转化为数据的累加[9], 当遗忘某个数据点时即从累加的中间结果上删除该数据点对应的数值. 具体做法如下: 为方便表述, 首先引入两个函数 $g_{c_{i,j}}(x)$ 和 $g'_{c_{i,j}}(x)$. 对于函数 $g_{c_{i,j}}(x)$, 当数据点 x 和 c_i 之间的距离最小时, 函数输出这个最小距离, 否则输出 0; 对于函数 $g'_{c_{i,j}}(x)$, 当 x 和 c_i 之间距离最小时, 函数输出 1, 否则输出 0. 每一轮聚类中心的迭代计算表示为 $\sum_{x\in X}g_{c_{i,j}}(x)\Big/\sum_{x\in X}g'_{c_{i,j}}(x)$. 若要遗忘某个数据点 x_u, 则从上述累加运算中删去 $g_{c_{i,j}}(x_u)$ 和 $g'_{c_{i,j}}(x_u)$, 再次迭代到聚类中心值收敛.

Lloyd's 算法是 K-means 聚类变体, 与 K-means 聚类区别在于: Lloyd's 算法输入是一组连续区域, 而 K-means 聚类输入是一组离散点. 因此, Lloyd's 算法在更新聚类中心时, 需计算各聚类质心, 而不是计算均值.

Lloyd's 聚类算法对应的机器遗忘方法是 Quantized k-means (Q-k-means)[40]. Q-k-means 在运行过程中, 根据训练时保存的中间结果判断遗忘某个数据点是否会导致质心改变. 若质心发生了改变, 则重新训练, 否则更新中间结果, 不重新计算质心. 质心计算是一种线性运算, 故在遗忘某个数据点时可以迅速判断质心是否更新.

2.1.4 深度学习

深度学习是以神经网络模型为架构, 对数据集进行表征学习的一类算法, 其过程一般可以概括为: 模型分批次读取部分训练数据或一次性读入全部训练数据, 首先正向传播模型输出, 模型输出与数据真实标签比较得到误差, 再反向传播两者间的误差, 通过计算误差和参数间的梯度来更新模型参数.

DeltaGrad[39]是适用于深度学习的继续计算类机器遗忘方法. 在训练 M_0 的过程中, DeltaGrad 保存了每一次训练的中间结果, 如当前轮次训练所用训练集、超参数、运算产生的梯度和参数等. 为了保障模型可用性, 在机器遗忘训练阶段, 小部分轮次精确计算模型在 $D\backslash x^*$ 上的梯度来更新参数, 其他轮次则利用原始训练过程中的中间结果来近似计算梯度. 近似计算梯度用时较短, 从而减少了总体用时.

对 M_0 调优 (finetune) 也是适用于深度学习的一种机器遗忘方法. 当 x^* 较小时, D 和 $D\backslash x^*$ 相差较小, 可使用 M_0 在 $D\backslash x^*$ 上继续训练, 进行机器遗忘. Unrolling SGD[34]改进了调优方法, 在继续训练阶段的损失函数上增加了正则项, 提出了标准差损失 (standard deviation loss), 使遗忘更快速. Ye 等人[48]同样改变损失函数以影响模型行为. 他们设计了两种损失函数: 一是让 M_{unlearn} 在 x^* 上的输出接近随机输出, 同时保留部分普遍特征不被移除; 二是让 M_{unlearn} 在 $D\backslash x^*$ 上的输出与 M_0 保持一致. 除修改损失函数外, Chen 等人[49]提出一个轻量级机器遗忘框架: 从 $D\backslash x^*$ 中取部分子集训练一个基准模型, 再让 M_0 向基准模型继续训练. 该方法操作难度较小, 易于理解.

2.2 模型分解

模型分解是将原本需要训练完整模型的过程划分为训练若干子模型的过程, 当遗忘请求到来时, 只训练遗忘数据集所涉及子模型, 不必重新训练完整模型, 从而缩短训练时间. 继续计算的方法并不涉及数据集划分, 而模型分解则涉及数据集划分.

© 中国科学院软件研究所 http://www.jos.org.cn

与继续计算相比, 模型分解无需额外存储空间保存中间结果, 但需考虑如何划分机器学习模型和如何聚合子模型结果等. 该思想关键在于将完整模型转化为子模型, 若划分粒度太小, 则子模型可能由于数据量太少而表现不佳, 影响整体性能; 若划分粒度太大, 则重新训练子模型时需训练数据量较大, 未必能很好地缩短训练时间. 模型分解在针对集成树模型、回归模型和聚类模型的机器遗忘方法中有所体现.

2.2.1 集成树模型

集成树模型是以决策树为基础的集成学习器, 决策树对样本特征空间进行递归划分, 希望产生最大的信息增量, 提高分类准确率.

DaRE (data removal-enabled forests) 是一种支持机器遗忘的集成树算法[24]. 它充分利用集成树的随机性和数据缓存特点来提高遗忘效率. 在构成 DaRE 的各个树中, 上层节点多为“随机节点”, 这些节点随机地选择分割属性和分割阈值. 随机节点对数据依赖度较小, 被更新频率较低. 下层节点多为“贪婪节点”, 它们完整遍历计算每个节点的基尼系数或信息熵, 以选择最优分割. 此外, DaRE 中的树会记录每个节点上的统计数据和叶子节点的训练数据, 当遗忘请求到来时, 只更新与该请求有关的树, 避免更新所有树, 降低计算量. DaRE 得到的机器遗忘模型与重新训练模型完全一致.

HedgeCut 是针对极度随机树 (extremely randomized tree, ERT) 的机器遗忘方法[25]. 与随机森林不同, ERT 对树进行划分时, 独立于目标属性而随机选择划分位点. HedgeCut 构造决策树时, 先根据允许构造树的数据点发生多大变化而不影响原有划分的特性, 将划分分为稳健划分 (robust split) 和非稳健划分 (non-robust split) 两类: 稳健划分在删除少量记录时, 不改变现有划分; 非稳健划分可能会在以后发生改变. 此处“少量记录”是针对所有数据点而言, 人为地设置允许改变的阈值, 比如, 若将阈值设置为所有数据点的 0.1%, 则当该树上要遗忘数据点小于所有数据点的 0.1% 时, 稳健划分并不改变. 在确定某处划分是否为稳健划分时, HedgeCut 先遍历数据点可能发生改变的所有情况, 以此量化该划分对数据点变化的稳健程度.

对于非稳健划分, HedgeCut 预先维护一组候选划分方法. 当机器遗忘请求到来时, HedgeCut 需要更新树, 此时稳健划分不需要改变, 只修正叶节点处统计数据; 而非稳健划分会重新计算各候选划分的基尼增益, 采纳增益最高的划分方法. 为了加快计算叶节点处增益, HedgeCut 在各稳健划分树叶节点处保存了统计数据.

2.2.2 回归模型

回归模型是根据数据集 D, 拟合出最佳刻画 x 和 y 之间的函数 f, 用于预测其他数据点 x' 的目标值 $f(x')$, 常见的回归方式有线性回归和局部加权线性回归等. 第 2.1.2 节的岭回归也是一种回归方法.

对于回归模型, Aldaghri 等人[50]设计了一种通过分解实现遗忘的方法. 他们把 D 划分成若干不相交的子集, 每个子集上训练一个子模型, 这些子模型输出结果最终聚合成总体的输出结果. 当要遗忘一个数据点时, 重新训练使用该数据点的子模型, 不必全局训练. 为了降低训练代价, Aldaghri 等人在划分训练集之前首先对其进行压缩: 随机抽取训练集中的部分数据点, 用这些数据点的聚合值代替原有数据点放入训练集. 该聚合过程可回溯, 若 x^* 被聚合了, 可在相应的聚合结果中除去其影响.

2.2.3 K-means 聚类

第 2.1 节已对 K-means 聚类做了说明, 并介绍了针对 K-means 的继续计算类机器遗忘方法 Q-k-means. DC-k-means (divide-and-conquer k-means) 是针对 K-means 的模型分解类机器遗忘方法[40]. DC-k-means 将 D 划分为若干子集, 在每个子集上独立运行 K-means 算法, 再递归地合并结果.

DC-k-means 基于树结构实现. 原始数据集被划分至树的叶子节点, 每个叶子节点均通过聚类算法求解其上数据集的质心. 在将叶子合并到父节点时, 合并后节点对应的新数据集包含了各叶子节点的所有质心. DC-k-means 继续使用聚类算法计算父节点上的新质心, 以此类推. DC-k-means 利用树的层次结构使质心和数据集依赖关系模块化. 当遗忘请求到来时, 只需重新进行从对应叶子节点到根节点的聚类运算. 根据实验结果[40], Q-k-means 运行速度高于 DC-k-means, 而 DC-k-means 聚类效果略优于 Q-k-means.

2.3 混合方法

混合方法是指同时使用继续计算和模型分解两种方法来进行机器遗忘的方法: 先将完整模型分解为子模型,

© 中国科学院软件研究所 http://www.jos.org.cn

保留子模型训练过程的中间结果, 当遗忘请求到来时, 从对应子模型中间结果开始训练, 不必重新训练整个子模型. 其中较为经典的是 2021 年提出的 SISA (sharded, isolated, sliced, and aggregated, SISA) 框架[26]. SISA 适用于深度学习算法. 深度学习与传统机器学习相比, 具有模型多样而训练过程高度统一的特点. 针对统一的训练过程, SISA 并用了继续计算与模型分解两种方法. 本节主要介绍 SISA 的基本过程、改进算法和系统应用.

2.3.1 SISA 的基本过程

在 SISA 框架 (示意图如图 1 所示) 中, 原始数据集被划分为多个不相交的数据块 (shard), 每个数据点只包含在一个数据块中. SISA 在这些数据块上分别训练子模型 (模型分解), 限制单个数据点对子模型的影响. 当遗忘请求到达时, 只需重新训练受影响的子模型. 由于数据块小于整个训练集, 训练时间较少, 但是这样做也减少了每个子模型的数据量, 导致子模型预测能力偏弱.

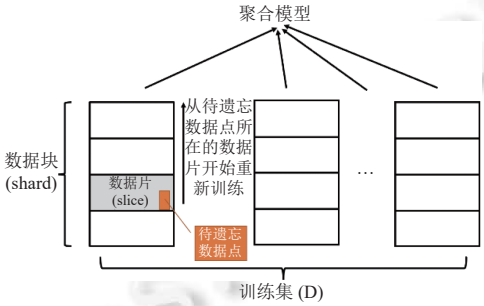


图 1 SISA 示意图

为了加速训练过程, SISA 还将每个数据块分割成数据片 (slice), 并在训练子模型时逐步加入数据片, 而不是直接在整个数据块上训练子模型. 在每次加入新的数据片之前, SISA 都会保存当前模型状态. 确定要遗忘的数据点所处的数据片之后, 模型拥有者就从该数据片参与训练前的最后模型状态开始训练 (继续计算), 进一步减少训练时间. 该做法与数据库中的检查点 (checkpoint) 策略相似. 在进行最终预测时, 模型拥有者可使用不同策略灵活地聚合各子模型的预测, 如多数投票法等.

SISA 遗忘速度与数据块和数据片大小直接相关, 数据块和数据片越小, 重训练用到的数据就越少, 用时也就越少; 但由于训练集变小, 子模型预测能力也会削弱, 从而影响模型整体性能.

2.3.2 SISA 的改进算法

ARCANE 框架是 SISA 的改进算法之一[51], 相较于 SISA 改进如下: 按标签划分数据块; 训练时保留更详细的模型状态; 人为地选出遗忘可能性较大的数据, 留到排序偏后的数据片上, 减少训练需要的数据片数.

DeepObliviate 同样在 SISA 基础上作改进, 但并非改进数据划分, 而是简化了继续计算过程[52]. 具体而言, DeepObliviate 首先将数据集 D 划分成数据块, 在训练时逐步增加数据块, 并保存这些数据块训练后的所有子模型. 接着, 对于到来的机器遗忘请求, DeepObliviate 会定位到 x^* 所在数据块, 从该数据块参与训练前的子模型开始在 $D\backslash x^*$ 上进行重新训练. 与 SISA 不同, 此处训练并非训练到用完所有数据块; 当重新训练模型参数和 M_0 中训练到某一步的子模型参数之差低于阈值时, DeepObliviate 就认为已消除该数据点的影响, 训练中止. 最后, 中止时的模型参数加上 M_0 最终步 (用到了所有数据块) 到中止步的参数差, 得到 M_{unlearn}. DeepObliviate 重用了 M_0, 与 SISA 相比, DeepObliviate 虽然对数据集进行了分块, 但最终训练步骤用到了所有数据块. Gupta 等人在 SISA 的基础上, 利用差分隐私聚合算法, 设计了一种可处理数据删除请求序列的方法[53]. 他们利用差分隐私理论, 保证了对删除请求序列进行遗忘的完成度.

2.3.3 SISA 的系统应用

本节重点介绍 SISA 在推荐系统中的应用. 推荐系统中常见的协同过滤算法根据用户的历史行为数据挖掘用户兴趣, 据此给用户推荐商品. Chundawat 等人[54]和 Chen 等人[55]研究了协同过滤算法中的机器遗忘问题, 对 SISA 遗忘框架进行改进, 使其更适用于协同过滤算法.

LASER 是一种用于推荐系统的机器遗忘方法[54]. 在 SISA 框架中, 划分数据集后, 每个用户只有与自己所在

© 中国科学院软件研究所 http://www.jos.org.cn

数据片内其他用户的协同关系, 这样的协同关系是不完整的. LASER 在给用户做划分时, 有更细致的划分策略: 事先将用户映射到一个向量空间, 用向量来表示用户信息, 并根据向量值来划分用户. 向量距离越小, 用户相似性就越高. LASER 使用优先队列进行划分, 使各个划分更加均匀. 在训练过程中, LASER 先在内部用户相似性大的数据集上训练, 再在相似性小的数据集上训练, 以此降低模型损失. Chen 等人[55]在对 SISA 方法做改进时, 除用户相似性外, 还考虑了商品相似性. 在聚合各子模型输出结果时, Chen 等人采用注意力模型来获取聚合参数, 从而提高聚合结果的准确性.

2.4 小 结

本节介绍了基于训练的机器遗忘方法. 这些方法进一步细分为继续计算、模型分解和混合方法这 3 类. 表 5 列出了各子类方法的代表方法、已知条件、所需开销和优缺点. 其中所需开销从时间和空间两部分进行说明, 已知条件是指使用这些方法时需要知道的条件.

表 5 基于训练的方法总结

类别	代表方法	已知条件	时间开销	空间开销	优点	缺点
继续计算	针对朴素贝叶斯[9]、岭回归[2]、K-means聚类[40]的机器遗忘方法	中间结果	从中间结果开始计算的开销	保存中间结果的开销	速度较快	只能针对特定算法, 对深度学习可能遗忘不完全
模型分解	针对集成树[24]、回归模型[50]、K-means聚类[40]的机器遗忘方法	数据集子集与子模型的对应关系	重新训练子模型的开销	无, 不必保存中间结果	不必保存中间结果	只能针对特定算法
混合方法	SISA[26]	数据集子集与子模型的对应关系, 中间结果	中途训练子模型的开销	保存中间结果的开销	遗忘效果较好	空间开销大

继续计算类方法需保存训练过程中的中间结果, 若模型较大, 则需保存的中间结果较大; 但这种方法允许从中间结果开始计算, 时间开销相对较小. 与继续计算类方法相比, 模型分解类方法不必保存中间结果, 但需保存子模型和子模型与数据集子集的对应关系. 继续计算类方法从中间结果上对整个模型进行训练, 而模型分解类方法是对子模型进行重新训练. 这两类方法都对模型有特定要求, 需要对中间结果或模型分解方法有所设计. 混合方法结合了继续计算和模型分解两种方法, 需要保存各个子模型的中间结果, 以更高的空间代价减少了用于遗忘的时间.

总体来看, 基于训练的方法较大程度上保真了训练过程, 得到的 M_{unlearn} 在准确率以及和 M_{retrain} 的相似度上更有保证; 缺点是需要用较多时间进行训练, 时间开销大; 以及需要额外空间保存中间结果, 空间开销大. 基于训练的方法适用于对时空开销要求较低而对 M_{unlearn} 性能要求较高的场景.

3 基于编辑的机器遗忘

基于编辑的机器遗忘方法指在已知 M_0 或 D 基础上, 通过 x^* 和 $D\backslash x^*$ 等信息对 M_0 进行编辑, 得到 M_{unlearn}. 与基于训练的方法不同, 基于编辑的方法无须对训练过程进行划分, 而是对模型输入、参数或输出进行直接修改, 使得最终模型与重新训练模型接近. 该方法根据编辑的对象可细分为输入编辑、参数编辑和输出编辑这 3 类. 当前文献以参数编辑为主.

3.1 输入编辑

输入编辑是指编辑输入数据 (包括改变输入数据的标签或特征), 用编辑后的输入数据来影响模型决策边界, 从而实现机器遗忘的方法. 输入编辑方法往往需要训练集 D, 但不必有 M_0. 输入编辑类机器遗忘方法的设计关键是挖掘训练数据和模型决策之间的关系, 通过数据影响模型, 使其在决策时与重新训练模型接近. 本节主要介绍标签修改和特征加噪两类输入编辑方法.

3.1.1 标签修改

标签修改是修改输入数据的标签. 在传统机器学习中, Parne 等人[47]研究使用决策树和随机森林对垃圾邮件进行分类的场景下遗忘某些邮件的问题. 当模型拥有者使用决策树对邮件分类时, 攻击者可在良性样本中添加垃圾信息来制造恶意样本以污染模型, 降低模型准确性. 当模型拥有者希望模型遗忘恶意样本时, 可修改含垃圾信息

© 中国科学院软件研究所 http://www.jos.org.cn

的样本标签, 削弱这些样本对模型的影响, 从而实现遗忘.

在深度学习中, Class Clown [27]通过修改输入数据标签来实现遗忘. 对于某个数据点, 模型对该点属于各个标签的可信度和这个点到决策边界距离相关, 可通过改变 x^* 周围损失函数空间分布和扭曲决策边界, 改变 x^* 对可信度, 让模型“相信” x^* 在训练时没有被使用. Class Clown 改变 x^* 的标签, 在每轮训练中都使用有错误标签的数据, 扭曲 x^* 周围决策边界. 此外, Class Clown 还增加了正确数据以缓解错误标签造成的准确率下降问题.

Kim 等人[31]同样修改了输入数据的标签, 与 Class Clown 不同的是引入了对抗学习. Kim 等人提出的方法可分为两步: 第 1 步, 中和化, 即篡改 x^* 中的正确标签, 在其上训练“学生模型”, 学生模型在 x^* 上的预测准确率接近随机猜测; 第 2 步, 在 $D\backslash x^*$ 上继续训练, 把 M_0 作为“老师模型”, 学生模型将老师模型的输出作为自己的输入, 以此来加速训练, 并使训练更稳定.

3.1.2 特征加噪

特征加噪是修改输入数据的特征, 在特征上添加噪音. Huang 等人[44]提出了一种特征加噪的方法, 适用于深度学习. Huang 等人在输入数据特征上加噪, 使在这些数据上训练的模型无法学习到数据包含的知识. Huang 等人让模型学习所加噪音和标签之间的关系, 从而生成能够混淆标签实现遗忘的噪音. 该学习过程可视为双层优化问题: 外层优化模型损失函数, 使损失函数尽可能地小, 保持模型性能; 内层优化噪音, 使噪音尽可能地小, 减少对模型的改动.

3.2 参数编辑

参数编辑的方法利用数学知识或数据集信息, 直接给出 M_{unlearn} 的参数计算方式. 参数编辑往往需要知道原模型参数 M_0, 同时需要知道数据集信息 (如原始数据集 D 或剩余数据集 $D\backslash x^*$), 必要时还需要记录训练 M_0 时的梯度等. 这类方法在 M_0 基础上进行计算, 无须通过大量数据迭代来进行训练, 速度较快. 本节根据参数计算方式, 主要介绍泰勒展开、梯度更新、优化求解和残差更新这 4 类问题.

3.2.1 泰勒展开

泰勒展开是一种用多项式来近似表示函数在某点周围情况的方法, 通过泰勒展开来近似计算 M_{unlearn} 方法多见于线性回归和深度学习. M_0 可视作损失函数在 D 上取最优值时的解, 在 x^* 较小, D 和 $D\backslash x^*$ 相差较小且 M_0 已知的情况下, 可通过泰勒展开来近似 $D\backslash x^*$ 处的最优解 M_{unlearn}.

对于线性回归模型, Guo 等人[11]提出了可验证遗忘 (certified removal, CR), 利用二阶泰勒展开来近似计算 M_{unlearn}, 用公式 (4) 来对参数进行编辑:

$$\theta_u = \theta + \boldsymbol{H}_\theta^{-1}\Delta \tag{4}$$

其中, θ 为原模型 M_0 参数, $\Delta = \alpha\theta + \nabla loss(\theta^{\mathrm{T}}X_u, y_u)$ 为模型在数据点 (X_u, y_u) 处的损失函数梯度, $\boldsymbol{H}_\theta$ 为模型参数在剩余数据集 $D\backslash x^*$ 上的海瑟矩阵 (Hessian matrix). 当损失函数是 θ 的二次函数时, 通过公式 (4) 得到的 M_{unlearn} 和 M_{retrain} 一致.

CR 需要计算损失函数在模型参数上的海瑟逆矩阵与一阶梯度向量乘积, 而海瑟逆矩阵计算复杂度较高. 对此, 有文献[20,39,56–58]在海瑟逆矩阵计算上进行改进, 如用 Fisher 矩阵来近似海瑟矩阵[20,57], 采用 BFGS 方法[39]来近似计算海瑟向量乘积等进行加速, 或用 LCODEC 来缩小计算海瑟矩阵的参数范围[42], 以及在求解海瑟矩阵时, 将求解过程转化为优化过程来近似求解[56]. 以上做法均以损失精度为代价, 缩短计算时间. CR 速度约为重新训练的 1 000 倍, 且支持多达 10 000 个数据点的遗忘[36].

当模型损失函数为凸函数时, 由于最优点唯一, 得到的 M_{unlearn} 和 M_{retrain} 一致. 而在深度学习模型中, 由于损失函数为非凸函数, 容易陷入局部最优而难以找到全局最优点, M_{unlearn} 和 M_{retrain} 之间会存在一定的误差. 这一误差可能泄露信息, 此时可通过与差分隐私[59]相似的加噪方法来避免攻击者获取误差[15]. Liu 等人[60]将在传统机器学习范式中的经典泰勒展开方法扩展到了对抗训练场景, 在对抗训练过程中, 训练数据不仅要参与最小化训练损失的外循环, 还要参与生成对抗扰动的内循环, 即存在双层优化. Liu 等人给出了该场景下遗忘后模型的解析解, 以总体海瑟矩阵对模型的影响度量为基础, 通过近似和转换降低了海瑟逆矩阵的计算开销.

© 中国科学院软件研究所 http://www.jos.org.cn

3.2.2 梯度更新

梯度更新是利用 M_0 训练过程中的梯度来更新 M_0 参数, 从而得到 M_{unlearn} 的方法, 适用于通过梯度下降来进行优化的学习过程 (如深度学习). 这种方法往往需要已知 x^* 在训练过程中对应的梯度.

Du 等人[61]提出了一种变梯度下降为梯度上升的遗忘方法 (逆训练), 参数更新方式见公式 (5). 这种方法存在损失爆炸和灾难性遗忘的问题, 对此, Du 等人使用 ReLU 函数来限制梯度的大小, 避免损失爆炸; 将更新后的模型参数和更新前的参数距离作为正则项加入梯度上升过程, 避免灾难性遗忘.

$$\theta_u = \theta + \eta \nabla loss(\theta^{\mathrm{T}} X_u, y_u) \tag{5}$$

Liu 等人[62]提出的 Forsaken 方法同样使用梯度来更新 M_0. 该方法改变了用于计算梯度的损失函数. Forsaken 首先以最小化模型在 x^* 上的输出和在随机数据集上的输出之间的分布距离为目标来设计损失函数; 对该损失函数进行优化得到梯度后, 将梯度作用于 M_0 作“掩码”, 得到 M_{unlearn}. 这种方法为使 M_{unlearn} 在 x^* 上的表现与在随机数据集上的表现接近, 需额外引入随机数据集.

Du 等人和 Liu 等人研究使用梯度下降来优化的一般性的学习问题, 而 Ganhor 等人[63]研究推荐系统泄露用户信息的具体问题. 除逆梯度方向更新参数外, 他们通过对抗训练层来抹除隐状态和用户隐私信息间的关系. 该对抗训练层从隐状态推测用户信息, 其输出作为样本标签的一部分输入主神经网络进行训练, 从而增强神经网络稳健性.

3.2.3 优化求解

优化求解类方法将求 M_{unlearn} 过程转化为优化问题, 最终求得问题的解即为 M_{unlearn}. 这类方法的关键在于优化问题构造, 构造思路可以从刻画数据集对模型的影响入手, 当遗忘请求到来时, 则从优化问题中减去 x^* 对模型的影响, 进而求解优化问题, 获得 M_{unlearn}.

在传统机器学习中, Chen 等人[28]讨论了可变 SVM (variable support vector machine, VSVM) 的机器遗忘方法. Chen 等人给出一组迭代表达式, 利用 $D\backslash x^*$ 对 VSVM 参数的影响进行优化迭代, 当参数趋于收敛时停止迭代, 得到 M_{unlearn}. Fu 等人[41]讨论了蒙特卡洛马尔可夫链 (Markov chain Monte Carlo, MCMC) 的机器遗忘方法, 它将 MCMC 过程转换为显式优化问题, 设计了数据对模型参数的影响力函数, 通过在模型参数原有分布上减去量化的影响力来实现遗忘.

Khan 等人[29]提出的 K-priors 方法既适用于传统机器学习, 也可用于深度学习. 该方法使得模型能够快速泛化到遗忘部分数据的场景. K-priors 方法的核心是 K-priors 值, 该值结合了模型参数和输出的后验概率. 在原损失函数中加上 K-priors 值后进行优化, 最终优化结果为 M_{unlearn}. Wu 等人[64]构造的优化问题与 Khan 等人不同, 它量化了训练集改变对模型参数的影响和参数改变对模型性能的影响, 当遗忘请求到来时, 先在训练集改变对模型参数的影响力函数中减去 x^* 对应的影响; 为使模型性能保持稳定, Wu 等人对最小化遗忘前后模型输出差异这一优化问题进行求解, 从而得到 M_{unlearn}.

优化求解不仅可用于传统的机器学习范式, 亦可用于强化学习范式. Ye 等人[65]利用优化求解思想设计了强化学习中的机器遗忘方法, 将待遗忘的环境数据反映到损失函数中, 尽可能地降低模型在待遗忘数据上的性能, 从而达到遗忘的目的.

3.2.4 残差更新

残差更新是一种利用合成数据来编辑参数的方法, 其编辑思路与前面介绍的泰勒展开、梯度更新和优化求解方法不尽相同. Izzo 等人[43]提出了投影残差更新法 (projective residual update, PRU), 在 M_{unlearn} 参数未知的情况下, 采用 leave-one-out 方法, 根据 M_0 在 x^* 上的预测结果合成 M_{unlearn} 在 x^* 上的预测. PRU 使用这些合成预测值和 M_0 预测值的距离来更新模型参数. PRU 时间复杂度与 x^* 大小成线性关系, 与 D 大小无关.

3.3 输出编辑

输出编辑是对模型输出进行编辑, 使模型输出与 M_{retrain} 接近. 这类机器遗忘方法较少, 目前只有 Baumhaumer 等人[30]提出的 Linear Filtration 方法. 该方法只适用于以标签为单位进行遗忘的场景和输出数据点在各标签可信度的模型. 具体而言, Linear Filtration 方法在模型输出某个数据点在各标签可信度分布向量时, 让该向量与一个过滤矩阵相乘, 使得相乘后分布向量不包含待遗忘标签对应的维度, 从而遗忘该标签. 对于过滤矩阵的构造, 有归一

© 中国科学院软件研究所 http://www.jos.org.cn

化方法和随机方法等. Linear Filtration 方法让模型输出不包含待遗忘标签, 对模型内部结构则不作改动.

3.4 小 结

本节介绍了基于编辑的机器遗忘方法. 这些方法进一步细分为输入编辑、参数编辑和输出编辑这 3 类. 表 6 是对这些方法的总结. 输入编辑类方法通过数据来影响模型, 遗忘效果较不稳定; 由于需要从头训练模型, 时间开销大; 需要的已知条件较多, 如需知道原始数据集 D. 参数编辑类方法直接给出 M_{unlearn} 计算方式, 速度较快, 时间开销小; 在 M_0 基础上进行更新, 性能更加稳定; 不同参数编辑类方法已知条件不同, 如采用梯度更新法, 则需要知道 x^* 对应的梯度等. 输出编辑类方法目前只有 Linear Filtration 这 1 种. 该方法直接在 M_0 输出上增加映射矩阵, 使输出标签不包含 x^* 的类别. 该方法运行速度快, 但使用限制大, 只适用于以标签为单位进行遗忘的场景, 且要求 M_0 输出数据点在各个标签的可信度.

表 6 基于编辑的方法总结

类别	子类	代表方法	已知条件	时间开销	空间开销	优点	缺点
输入编辑	标签修改	针对决策树[31]的机器遗忘方法、Class Clown[27]	原始数据集	重新训练模型的开销	无额外开销, 不必保存中间结果	思想易于理解, 遗忘效果较好	需要重新训练模型
	特征加噪	针对深度学习[44]的机器遗忘方法	原始数据集	重新训练模型的开销	无额外开销, 不必保存中间结果	思想易于理解	需要重新训练模型, 遗忘的效果较不稳定
参数编辑	泰勒展开	CR[11]及其改进方法[20,39,56–58]	剩余数据集和原模型	计算海瑟矩阵等的开销	计算海瑟矩阵等的开销	速度较快	遗忘的效果较不稳定
	梯度更新	Forsaken[62]	原模型, 模型训练过程中的梯度	用梯度更新参数的开销	保留训练时梯度的开销	速度较快	需要保留训练过程中的梯度
	优化求解	针对可变SVM[28]的机器遗忘方法、K-priors[29]	视优化问题构造而定, 或需要原始数据集, 或需要损失函数	求解优化问题的开销	求解优化问题的开销	速度较快, 遗忘效果较好	未必能找到最优解
	残差更新	PRU[43]	原模型, 原始数据集	合成数据的开销	合成数据的开销	时间复杂度与原始数据集大小无关	遗忘的效果较不稳定
输出编辑	—	Linear Filtration[30]	原始数据集标签分布	计算标签映射矩阵开销	计算标签映射矩阵的开销	速度较快	只适用于按标签进行遗忘的场景

总体来看, 基于编辑的方法优点是速度较快, 避免了大量训练过程; 若是编辑 M_0 得到的 M_{unlearn}, 则 M_{unlearn} 性能会有较大的保障; 缺点是可能需要额外空间来保存梯度等背景知识, 空间开销大, 以及 M_{unlearn} 性能不够稳定 (输入编辑). 基于编辑的方法适用于对时空开销要求较高但对 M_{unlearn} 性能要求较低的场景.

4 基于生成的机器遗忘

基于生成的机器遗忘方法是指通过生成模型来获得 M_{unlearn}. 常见的生成模型有传统机器学习中的朴素贝叶斯、MCMC 和深度学习中的生成对抗网络 (generative adversarial network, GAN)[66]. 生成模型会模拟数据产生的概率. 它用于机器遗忘, 则会模拟 M_{unlearn} 各参数值概率, 进而给出 M_{unlearn}. 本节分别介绍使用传统机器学习 (传统生成) 和使用深度学习生成模型 (深度生成) 来获得 M_{unlearn} 的方法.

4.1 传统生成

传统机器学习中的 MCMC 是一种从概率分布中进行采样的算法, 首先构建以目标分布为平衡分布的马尔可夫链, 通过链中状态转移来获得目标分布样本; 状态转移次数越多, 所得样的分布就越接近实际所需分布. MCMC 需要较大的状态空间, 适用于生成逻辑回归或线性回归等参数较少的模型.

Nguyen 等人[33]提出了用 MCMC 生成 M_{unlearn} 模型参数的方法: 首先使用 MCMC 生成一组 M_{unlearn} 候选参数, 并通过概率展平等方式对候选参数进行扩充, 最后将这组参数加权平均, 作为最终模型参数. 同样基于 MCMC, Ullah 等人[67]并不直接生成模型参数, 而是把数据集删除前后模型隐状态分布看作马尔可夫链出发地和目的地, 对加噪的随机梯度下降过程构造马尔可夫链进行机器遗忘, 求解从出发地到目的地的最优路线. 他们提出了 sub-

© 中国科学院软件研究所 http://www.jos.org.cn

sample-GD 和 noisy-m-A-SGD 两种算法, 前者适用于高维数据, 后者适用于低维数据, 求解过程中需要保存每个时刻的状态, 即模型梯度.

除了使用 MCMC 以外, Nguyen 等人[68]将逻辑回归看作一个变分推断 (variational inference) 问题. 变分推断通过优化简单分布和复杂分布间的距离, 使得简单分布能够拟合复杂分布. 针对逻辑回归算法的遗忘, Nguyen 等人用 M_{unlearn} 去拟合 M_{retrain}, 用 KL 散度衡量 M_{unlearn} 参数和 M_{retrain} 参数间的距离并进行优化, 继而得到 M_{unlearn}. 以上方法均生成机器遗忘后参数, 适用于参数空间小的模型.

4.2 深度生成

GAN 是在 2014 年提出的一类深度学习框架, 其中存在两个神经网络, 即生成器和判别器. 生成器生成一系列样本, 判别器则对这些样本进行评估, 生成器以提高判别器错误率为目标, 若判别器无法甄别生成样本和真实样本, 则认为生成的样本可用.

Chen 等人[69]使用 GAN 来生成 M_{unlearn}. 这种机器遗忘方法适用于各种机器学习模型, 包括损失函数为非凸深度学习模型. 在 Chen 等人提出的方法中, 生成器生成 M_{unlearn}, 判别器则要求 M_{unlearn} 在 x^* 上的输出分布和 M_{retrain} 在从未见过的第三方数据集上的输出分布接近, 以此作为优化目标. 与 Chen 等人的方法相近, Kim 等人[31] 和 Chundawat 等人[70]同样使用生成模型来获得 M_{unlearn}, 但不同之处在于, 除了对 M_{unlearn} 在 x^* 上的输出分布有要求外, 他们还考虑了 M_{unlearn} 在 $D\backslash x^*$ 上的准确率. Chundawat 等人使用两个模型作为训练基准: 一是在完整数据集上训练的模型 M_0 (smart teacher, Ts), 二是随机输出的模型 (dumb teacher, Td). Chundawat 等人将两者的输出和 M_{unlearn} 输出的距离作为损失函数进行优化, 使 M_{unlearn} 在 $D\backslash x^*$ 上的训练效果与 Ts 相似, 在 x^* 上的训练效果与 Td 相似. Kurmanji 等人提出的 SCRUB[46]同样采取了师生模式, 其中, 学生模型只记住老师模型在待删除数据以外的知识. 他们还利用“回滚”技术提高了遗忘后模型的隐私性, 避免了待删除数据被成员推理攻击窃取. Zhang 等人[71]在图像检索场景探索遗忘方法. 他们通过生成器生成噪声, 同时执行一对静态和动态训练过程, 使生成的噪声数据淡化模型对待删除数据的记忆, 同时保留剩余数据集的信息.

OptLearn 同样使用深度神经网络预测 M_{unlearn}[34], 但仅限于损失函数为凸函数的模型, 如 SVM 等. OptLearn 分为离线训练和在线估计两个阶段: 离线训练阶段, 深度神经网络从训练集中采样训练, 该过程的损失函数中加入了 KKT (Karuch-Kuhn-Tucker) 正则项和衡量模型可用性的正则项, 使得深度神经网络预测的模型参数是其损失函数上所对应的最优点; 在线估计阶段, 使用训练好的神经网络预测 M_{unlearn} 参数.

4.3 小 结

本节介绍基于生成的机器遗忘方法. 这些方法根据所用生成模型可细分为传统机器学习和深度学习两类. 表 7 列出了各子类代表方法、已知条件、时空开销和优缺点.

表 7 基于生成的方法总结

使用模型	代表方法	已知条件	时间开销	空间开销	优点	缺点
传统机器学习	使用MCMC[33]、变分推断[68]的机器遗忘方法	视算法而定, 如可能需要训练梯度等	运行生成算法的开销	保存各状态的开销	速度较快	只适用于参数量较小的模型, 遗忘效果不稳定
深度学习	SCRUB[46]、OptLearn[34]	原始数据集	训练生成模型的开销	无额外空间, 不必保存中间结果	遗忘效果较好	速度较慢

使用传统机器学习模型生成 M_{unlearn} 的常用模型有 MCMC 或变分推断等, 使用这些模型对 M_{unlearn} 参数空间有限制, 适用于参数较少的小模型. 若参数空间太大, 则将导致生成过程中状态空间太大, 开销增大. 使用深度学习模型生成 M_{unlearn} 用到的模型有 GAN 和普通神经网络, GAN 通过判别器和生成器之间的竞争, 使得生成器在 x^* 上表现趋于随机化. 普通神经网络直接生成 M_{unlearn} 参数, 但只适用于损失函数为凸函数的模型.

总体来看, 基于生成的方法优点是遗忘效果较好, 如使用 GAN 时通过判别器保证了 M_{unlearn} 在 x^* 和 $D\backslash x^*$ 上的性能表现; 缺点是时空开销较大, 如使用 MCMC 时需保存各步骤状态的空间开销、使用神经网络模型时的训练开销等. 基于生成的方法适用于对时空开销要求较低但对 M_{unlearn} 的性能要求较高的场景.

© 中国科学院软件研究所 http://www.jos.org.cn

5 度量指标

就评估性能而言, 机器遗忘和机器学习不同, 机器学习模型的性能评估可以对比该模型在预测集上的结果和真实标签的差异, 而机器遗忘则需考虑时间、M_{retrain} 和 M_{unlearn} 相似度和 M_{unlearn} 在 $D\backslash x^*$ 上的预测准确率等. 不同文献对机器遗忘进行度量的指标并不相同, 本文对机器遗忘的度量指标进行归纳, 总结为以下 3 个角度: 机器遗忘速度, 机器遗忘可用性和机器遗忘完成度.

5.1 机器遗忘速度

机器遗忘速度是得到 M_{unlearn} 所用时间相比于重新训练 M_{retrain} 所用时间的缩短程度. 评估机器遗忘速度是比较 M_{retrain} 和 M_{unlearn} 训练时间, 计算机器遗忘相对于重新训练的加速比. 加速比越高, 则遗忘速度越快. 加速比的形式化定义如下.

定义 2. 通过重新训练获得 M_{retrain} 的时间为 T_{retrain}, 通过机器遗忘获得 M_{unlearn} 的时间为 T_{unlearn}, 则加速比为 $Speedup = Tretrain/Tunlearn$.

5.2 机器遗忘可用性

机器遗忘可用性是指 M_{unlearn} 的可用性, 即 M_{unlearn} 可以作出正确判断, 预测出样本正确标签的能力. 若机器遗忘后, M_{unlearn} 在测试集上的准确率远低于 M_{retrain}, 预测能力达不到标准, 则认为模型可用性过低. 遗忘可用性往往用 M_{retrain} 和 M_{unlearn} 在测试集上的准确率来比较和衡量. 模型在测试集上准确率的形式化定义如下.

定义 3. 对于模型 M, 测试集样本总数为 n, 模型正确预测样本数为 c, 则模型在测试集上的准确率为 $ACC= c/n$.

5.3 机器遗忘完成度

机器遗忘完成度是 M_{retrain} 和 M_{unlearn} 的相似度, 两者相似度越高, 则完成度越高. 比较两者的相似度, 有多种比较方法, 本文将相似度比较方法分为 3 类: 比较输出分布、比较参数距离和数据推理验证.

5.3.1 比较输出分布

比较输出分布是指给定一组样本, 让 M_{retrain} 和 M_{unlearn} 在这组样本上分别进行预测, 比较两者输出分布的差异. 衡量分布间距离的方法有计算 KL 散度 (Kullback-Leibler divergence)[20]、计算输出分布概率比值[11]和使用 Kolmogorov-Smirnov 算法[19]等.

除直接计算概率分布的距离外, 还可引入外部分类器来判断两个模型输出是否不可分辨. Baumhauer 等人[30]使用贝叶斯分类器判断能否分辨 M_{retrain} 输出和 M_{unlearn} 输出, 若分类器分辨准确率接近随机猜测, 则认为 M_{retrain} 和 M_{unlearn} 输出难以分辨, 相似度较高.

5.3.2 比较参数距离

比较参数距离是比较 M_{retrain} 和 M_{unlearn} 在参数上的差异. 比较参数距离的常用方法是计算参数间的范数差, 范数差越小, 则认为参数距离越小. 如设 M_{retrain} 参数为 θ^{retrain}, M_{unlearn} 参数为 θ^{unlearn}, 则使用 L2 范数进行参数距离计算的公式为 $\left\|\theta^{\text{retrain}},\theta^{\text{unlearn}}\right\| = \sqrt{\sum_i (\theta_i^{\text{retrain}} - \theta_i^{\text{unlearn}})^2}$.

对于保有随机性的模型, 可用 KL 散度等比较参数分布间的距离[20,21,33,52]. 但是, Thudi 等人[72]提出在参数空间判断机器遗忘是否成功并不完全合理, 因为对于任意模型, 可通过在更大的数据集上采样, 构造出一个不包含 x^* 的数据集, 在其上训练出的模型参数和 M_0 相差极小, 即无法从参数层面判断模型的训练集是否包含 x^* .

5.3.3 数据验证评估

数据推理验证是指从训练数据本身出发, 设计推理方法来判断某个样本是否在训练集中, 从而进一步对机器遗忘效果进行评估判断. 若可通过推理方法推测出 x^* 位于训练集中, 则认为遗忘完成度低; 若无法推测出训练集中存在 x^* , 则认为遗忘完成度高. 数据验证评估主要包括成员推理攻击和给数据添加“触发器”两类. 两种验证方法均适用于黑盒评估, 即用户并不知道模型的具体信息.

© 中国科学院软件研究所 http://www.jos.org.cn

成员推理攻击是一种用于确定某样本是否在模型训练集中出现的攻击方法. 在训练成员推理攻击模型时, 需构建多个影子模型 (shadow model). 这些模型的行为与被攻击的模型 (目标模型) 类似, 但每个影子模型的训练集是攻击者构造的, 即能确定某条训练数据是否在其训练集中. 攻击者给影子模型输出打上“在数据集中”和“不在数据集中”标签, 作为攻击模型的训练集进行监督训练, 训练完成后, 攻击模型可以根据目标模型的输出来区分目标模型是否使用了某个样本.

Chen 等人[69]使用成员推理攻击, 若攻击模型能够推理出 x^* 为 M_{unlearn} 训练数据, 则认为遗忘的完全度较低. Huang 等人[73]进一步发展了成员推理攻击用于遗忘验证的方法, 除成员推理攻击外还使用统计只是加以完善. 假设用户想查询数据集 Q 中的数据点是否在训练集中, 则首先对 Q 中的数据点分别使用成员推理攻击, 将成员推理攻击准确率等作为依据, 判断这些数据点是否在训练集中. 接着, 使用统计工具统计单个数据点上的判断结果, 继而判断整个 Q 是否在训练集中. 这种先个别判断再整体判断的方法增强了判断的稳健性, 既不需要训练额外评估模型和高质量的校准数据集, 又支持数据集 Q 与训练集相似的情况.

给数据添加“触发器”是指对数据做一定程度的改动, 在改动后数据上训练的模型会有辨识度较高的预测行为, 如会把改动后数据都分到某个特定类别上. 若模型没有在对应数据上训练过, 则模型在这些数据上就不会有期望的预测行为, 从而可以判断出训练集中是否包含了改动后数据. 若要评估遗忘完成度, 可对 x^* 添加触发器, 比较 M_{unlearn} 在 x^* 或与 x^* 同样添加了触发器的数据上的预测行为. Goel 等人[74]应用了触发器思想, 强调模型从 D 和 $D\backslash x^*$ 中获取信息的差异, 针对以类为单位进行遗忘的场景, 提出了类间混淆 (interclass confusion) 测试方法: 首先在原始数据集中要遗忘的多类数据里错误标记一些样本 (添加触发器), 形成 x^*, 根据 M_{unlearn} 在标签错误和标签正确的样本上的预测情况量化机器遗忘完成度.

5.4 小 结

本节介绍了速度、可用性和完成度这 3 个机器遗忘效果的评估角度, 并给出对应的度量指标. 速度是指机器遗忘相比于重新训练在减少用时上的提升, 可用性是指 M_{unlearn} 的准确率, 完成度是指 M_{retrain} 和重新训练模型的相似程度. 本文将 3 个角度的常见计算指标和已知条件归纳在表 8 中.

表 8 评估角度总结

评估角度	常见指标	已知条件
速度	加速比	训练M_{retrain}和获取M_{unlearn}所用时间
可用性	M_{unlearn}在测试集上的准确率	M_{unlearn}和测试集
完成度	M_{retrain}和M_{unlearn}输出分布距离	M_{retrain}和M_{unlearn}的输出
	M_{retrain}和M_{unlearn}参数距离	M_{retrain}和M_{unlearn}的参数
	成员推理攻击准确度, 在添加了触发器的数据上的准确度	M_{retrain}和M_{unlearn}的输出

从本节提出的 3 个角度来看传统机器学习中机器遗忘的效果, 传统机器学习中机器遗忘方法大多保存了中间结果以便进行遗忘[75](如针对朴素贝叶斯和 K-means 的机器遗忘方法), 或根据传统机器学习的机理, 给出其遗忘后模型的解析解 (如针对线性回归中的机器遗忘方法). 由于传统机器学习方法本身复杂度较低, 涉及参数量较小, 其对应的遗忘方法复杂度低于深度学习, 具有更快的速度和较高的可用性与完成度.

对于深度学习的机器遗忘方法, 目前的工作缺少从本节提出的 3 个角度对机器遗忘方法进行统一评估, 对此, 本文在第 6 节进行拓展实验, 探究不同机器遗忘方法在各评估角度的表现差异和不同度量指标之间的联系.

6 实验评估

针对传统机器学习的方法往往根据各机器学习模型的特点来设计, 一种机器学习模型只对应特定机器遗忘方法, 不具有普适性. 相比之下, 针对深度学习模型的方法往往与具体模型无关, 一个深度学习模型存在多种机器遗忘方法. 在这种情况下, 本节进行了实验, 探究在相同深度学习场景下不同机器遗忘方法的效果.

© 中国科学院软件研究所 http://www.jos.org.cn

本文结合覆盖方法的多样性和论文提供代码的情况, 在基于训练的方法、基于编辑 (包括输入、参数和输出编辑) 和基于生成的方法中分别选取具有代表性的方法进行实验, 探究以下 4 个问题.

● RQ1: 遗忘方法性能比较

从遗忘速度、可用性和完成度来看, 不同的机器遗忘方法性能如何? 目前缺少统一的实验来对不同机器遗忘方法效果进行比较. 针对这一空缺, 本文选择了 7 种不同类别的机器遗忘方法进行测试, 并从第 5 节提出的速度、可用性和完成度这 3 个角度对这些方法进行评估. 通过实验, 前文对这些方法优缺点的理论分析可以得到验证, 人们在具体场景下选择机器遗忘方法时也参考实验结果.

● RQ2: 标签总数影响分析

对单个标签进行遗忘时, 数据集标签总数对机器遗忘方法的效果影响如何? 为了探究该问题, 本实验设置了两种遗忘场景, 一是以样本为单位进行的遗忘, 即样本遗忘; 二是以标签为单位进行的遗忘, 即标签遗忘. 样本遗忘可用样本遗忘率来刻画, 标签遗忘则可用标签遗忘数 (x^* 包含标签数) 和标签遗忘率 (x^* 包含标签数与标签总数的比值) 来刻画. 标签遗忘场景下, 本文除了在不同标签遗忘数场景下进行实验外, 还针对标签遗忘率进行实验: 固定遗忘标签数, 改变标签总数, 探究不同标签遗忘率对遗忘效果的影响. 直观假设是, 标签总数越大, 则认为机器学习任务难度越大, 相同条件下遗忘复杂度也增大, 从而对遗忘可用性和完成度产生负面影响.

● RQ3: 度量指标关联分析

机器遗忘算法不同度量指标之间有什么联系? 目前缺少统一指标来衡量评估学习算法效果, 若能设计一个统一指标, 与目前各种度量指标均有相关性, 则可用该统一指标来对机器遗忘进行评估, 不必计算多个指标. 以此为出发点, 本实验探究不同度量指标的相关性, 如模型遗忘可用性与完成度是否成正相关等, 推动统一指标的构建.

● RQ4: 遗忘适用场景分析

机器遗忘方法适用场景是什么? 机器遗忘的目的是以尽可能少的时间获取和重新训练模型尽可能接近的模型, 在这一目的下, 部分场合就不必使用机器遗忘, 例如, 若重新训练模型本身用时很短, 或重新训练模型与原模型本身差别很小, 就不必使用机器遗忘. 本文立足实验结果, 分析机器遗忘模型和重新训练模型在各场景下的性能差异, 就机器遗忘适用场景这一问题进行详细阐述.

实验在 Nvidia A40 GPU 上进行, GPU 显存为 48 GB; 所用 CPU 为 Intel (R) Xeon (R) Platinum 8268 CPU, 共 96 核. 实验在 Ubuntu 18.04 LTS 64 位操作系统上运行, CUDA 版本为 11.4, PyTorch 版本为 1.11.0, Python 版本为 3.7.

6.1 实验设置

6.1.1 实验数据

本文在 MNIST、CIFAR-10、Purchase、ImageNet 子集、FashionMNIST 和 SVHN 这 6 个数据集上进行实验, 具体数据集信息如表 9 所示. 其中, ImageNet 子集选取了 ImageNet 前 100 个类. Purchase 数据集来自 Kaggle “Acquire Valued Shoppers”竞赛数据集, 包含了若干用户 1 年内的所有购买记录, 每笔记录包含商品名、连锁店、数量、日期等信息. 本文对 Purchase 数据集的处理方法与文献 [26] 相同: 对于每条购买记录, 选择前 600 个购买量最多的商品作为特征, 根据该条记录的顾客是否购买了该商品来设定特征取值为 1 或 0, 再采用聚类算法对这些购买记录进行聚类 (聚为 2 类), 把聚类结果作为该条记录标签, 表征一类购买行为.

表 9 实验数据集

数据集	维度	训练集	测试集	标签数
MNIST[76]	28×28	60 000	10 000	10
CIFAR-10[77]	32×32×3	50 000	10 000	10
Purchase[78]	600	249 215	62 304	2
ImageNet子集[79]	84×84×3	48 000	11 640	100
FashionMNIST[80]	28×28	60 000	10 000	10
SVHN[81]	32×32×3	73 257	26 032	10

© 中国科学院软件研究所 http://www.jos.org.cn

6.1.2 实验模型

实验所用模型均为神经网络模型, 对于较复杂的 CIFAR-10、SVHN 和 ImageNet 子集数据集, 使用较大的 ResNet 神经网络, 对于较简单的 MNIST、FashionMNIST 数据集和数据集 Purchase, 使用较小的神经网络. 具体模型结构如表 10 所示.

表 10 实验模型

数据集	模型结构
MNIST、FashionMNIST	2层卷积层, 2层全连接层
CIFAR-10、SVHN	ResNet-18
Purchase	2层全连接层
ImageNet子集	ResNet-18

6.1.3 实验对象

本文将继续计算的方法 DeltaGrad 和 UnrollingSGD、输入编辑的方法 Unlearnable、参数编辑的方法 CR 和 Sekhari、输出编辑的方法 Linear Filtration, 以及部分论文中出现过的 Finetune 方法作为实验对象, 比较以上 7 种方法在相同条件下的遗忘效果. 具体方法的设置如表 11 所示.

表 11 实验使用机器遗忘方法

方法名称	方法类别	说明
Retraining	重新训练(基准)	得到 M_{retrain}
Finetune	继续计算	在原模型基础上调优, 调优阶段训练参数与原始训练阶段相同
Unrolling SGD[21]	继续计算	使用standard deviation loss作为遗忘阶段损失函数
SISA[26]	混合方法	数据块数设为10, 子模型训练过程超参数与其他机器遗忘方法正常训练的超参数相同
DeltaGrad[39]	继续计算	对于ResNet-18, 将其拆分为特征提取层(卷积层)和全连接层, 机器遗忘时仅针对全连接层进行遗忘. 数据读取方式沿用文献[39]给出的方式
Unlearnable[44]	输入编辑	通过双层优化生成噪音
Linear Filtration[30]	输出编辑	只适用于标签遗忘场景
CR[11]	参数编辑	使用LCODEC方法选择要编辑的参数, 为提高模型可用性不添加噪声
Sekhari[82]	参数编辑	使用LCODEC方法选择要编辑的参数, 为提高模型可用性不添加噪声
Chundawat[70]	深度生成	参考文献[70], 机器遗忘阶段生成模型调整的轮次数设为10

在 CIFAR-10 数据集、SVHN 和 ImageNet 子集数据集上使用 DeltaGrad 方法时, 由于训练过程中需要记录每一轮次的参数和梯度, 而 ResNet-18 网络较大, 若对整个网络进行训练和记录, 则占用存储空间过大. 因此本次实验采用和文献 [39] 相同的模型处理方法, 将 ResNet-18 模型拆分为特征提取层和全连接层, 仅对全连接层使用 DeltaGrad 进行训练和记录.

在本文实验中, 模型由神经网络构成, 实现方法均基于 Python 以及 PyTorch 框架, 其中有多项超参数. 本文在 MNIST、FashionMNIST 和 Purchase 所用神经网络模型和 ResNet-18 超参数设定上, 与文献 [26] 保持一致. 实验统一使用 SGD 算法进行优化. 所有方法均已达到收敛.

6.1.4 实验场景

对于 RQ1、RQ3 和 RQ4, 本文设置两类, 共 5 种遗忘场景进行实验. 两类遗忘场景: 1) 样本遗忘, 从样本中随机抽样待遗忘样本, 遗忘数量用样本遗忘率来刻画; 2) 标签遗忘, 以标签为单位, 遗忘 1 个或多个标签的样本. 在本文针对 RQ1 的实验中, 对于样本遗忘, 采用 1%、5% 和 10% 的样本遗忘率进行实验; 对于标签遗忘, 分别遗忘 1 个标签和 3 个标签数据进行实验.

对于 RQ2, 本文设置了一组标签遗忘数不变、标签总数改变的场景进行实验, 以探究标签遗忘率对机器遗忘的影响. 具体而言, 本文在 ImageNet 子集数据集上做调整, 每次固定遗忘 1 个标签 (遗忘标签数为 1), 而通过取不同子集来控制 D 的标签总数. 实验中, 分别取 D 标签总数为 20、30、50、70、100, 在这些遗忘场景下, 同样对实

© 中国科学院软件研究所 http://www.jos.org.cn

验对象中的 7 种机器遗忘方法进行实验.

在实验对象中, Linear Filtration 方法只适用于标签遗忘场景. 在实验数据中, Purchase 数据集不适用于标签遗忘场景. Purchase 数据集一共只有 2 个标签, 在标签遗忘场景下不具有可实验性.

6.1.5 评估方法

本文以在 $D\backslash x^*$ 上训练的模型 M_{retrain} 为基准, 各机器遗忘方法得到的模型 M_{unlearn} 与之进行比较, 计算的指标有: 机器遗忘方法加速比 *Speedup*, M_{unlearn} 在测试集上的准确率 ACC_{test}, M_{unlearn} 在 $D\backslash x^*$ 上的准确率 ACC_{remained}, M_{unlearn} 在 x^* 上的准确率 ACC_{deleted}, M_{unlearn} 与 M_{retrain} 的输出距离 $DIST_{\text{output}}$, M_{unlearn} 与 M_{retrain} 的参数距离 $DIST_{\text{para}}$.

机器遗忘速度用加速比 *Speedup* 衡量, 其计算公式同定义 2. *Speedup* 越大, 速度就越快; 可用性用 M_{unlearn} 在测试集上的准确率 ACC_{test} 来衡量, 其计算公式同定义 3, 为 $ACC_{\text{test}} = c/n_{\text{test}}\times 100$. ACC_{test} 越高, 可用性就越高; 遗忘完成度用 M_{unlearn} 与 M_{retrain} 间的输出距离 $DIST_{\text{output}}$ 和参数距离 $DIST_{\text{para}}$ 来衡量. $DIST_{\text{output}}$ 使用 Kullback-Leibler 散度来计算, 计算方法如下.

定义 4. 设 M_{unlearn} 输出为 p, M_{retrain} 输出为 q, 则 $DIST_{\text{output}} = \sum_{i=1}^{|q|} q_i \log q_i/p_i$.

$DIST_{\text{para}}$ 使用 L1 范数距离计算, 计算方法如下.

定义 5. 设 M_{unlearn} 和 M_{retrain} 模型参数为 θ, x_i^{unlearn} 和 x_i^{retrain} 分别为 M_{unlearn} 和 M_{retrain} 的第 i 个参数值, 则 $DIST_{\text{para}}= \sum_{i=1}^{|\theta|} |x_i^{\text{unlearn}} - x_i^{\text{retrain}}|$.

$DIST_{\text{output}}$ 和 $DIST_{\text{para}}$ 越小, 完成度就越高. 实验同时记录 M_{unlearn} 在剩余数据集 $D\backslash x^*$ 上的准确率 ACC_{remained} 和在遗忘数据集 x^* 上的准确率 ACC_{deleted}, 用于 RQ4 中的机器遗忘适用场景分析. ACC_{remained} 和 ACC_{deleted} 计算方法如下.

定义 6. 对于模型 M, $D\backslash x^*$ 样本总数为 n_{remained}, 模型正确预测样本数为 c, 则 $ACC_{\text{remained}} = c/n_{\text{remained}}$.

定义 7. 对于模型 M, x^* 样本总数为 n_{deleted}, 模型正确预测样本数为 c, 则 $ACC_{\text{deleted}} = c/n_{\text{deleted}}$.

以下情况不计算 $DIST_{\text{para}}$. 1) 使用 DeltaGrad 对 ResNet-18 进行遗忘的场景. 在使用 DeltaGrad 对 ResNet-18 进行遗忘时, 由于将 ResNet-18 模型拆分为特征提取层和全连接层两部分, 仅对全连接层使用机器遗忘方法进行训练和记录, 故用于训练的 M_{unlearn} 与 M_{retrain} 结构不同, 不计算 $DIST_{\text{para}}$. 2) 使用 Linear Filtration 进行遗忘的场景. Linear Filtration 需要在原模型后接标签映射矩阵, 所得 M_{unlearn} 与 M_{retrain} 结构不同, 故不计算 $DIST_{\text{para}}$. 3) 使用 SISA 进行遗忘的场景. SISA 将原始模型分割成若干个子模型进行训练, 子模型的参数与原始模型不具有相同的训练条件, 故不具有可比性, 不计算 $DIST_{\text{para}}$.

对速度、可用性和完成度比较方法如下: 1) 当比较速度时, 对各遗忘场景下各机器遗忘方法在不同数据集上的 *Speedup* 求均值, 将均值按降序排列; 2) 当比较可用性时, 对各遗忘场景下各机器遗忘方法在不同数据集上的 ACC_{test} 求均值, 将均值按降序排列; 3) 当比较完成度时, 对各遗忘场景下各机器遗忘方法在不同数据集上的 $DIST_{\text{output}}$ 和 $DIST_{\text{para}}$ 分别求均值, 将均值累加后按升序排列. 根据以上方法得到排序序列后, 前 3 位划分为“高”, 中 3 位划分为“中”, 后 3 位划分为“低”.

6.2 结果分析

本节结合实验结果, 就前文提出的 4 个研究问题进行分析. 本文分析不同研究问题所用的度量指标不同, 具体分析各问题时使用的度量指标见表 12.

表 12 研究问题使用的度量指标

研究问题	*Speedup*	ACC_{test}	ACC_{remained}	ACC_{deleted}	$DIST_{\text{output}}$	$DIST_{\text{para}}$
RQ1	√	√	—	—	√	√
RQ2	√	√	—	—	√	√
RQ3	√	√	√	√	√	√
RQ4	√	—	√	√	—	—

© 中国科学院软件研究所 http://www.jos.org.cn

6.2.1 遗忘方法性能比较

从遗忘速度、可用性和完成度来看, 不同的机器遗忘算法性能如何? 为了探究该问题, 本文将不同的机器遗忘方法在相同遗忘场景下进行对比实验. 图 2 为各方法在样本遗忘 (第 1 行) 和标签遗忘 (第 2 行) 场景下的 *Speedup*、ACC_{test}、$DIST_{\text{output}}$、$DIST_{\text{para}}$, 图中所示为各方法在所有数据集上实验结果的均值. 本文按照第 6.1.5 节的评估方法对测试的机器遗忘方法进行评估, 结果如表 13 所示.

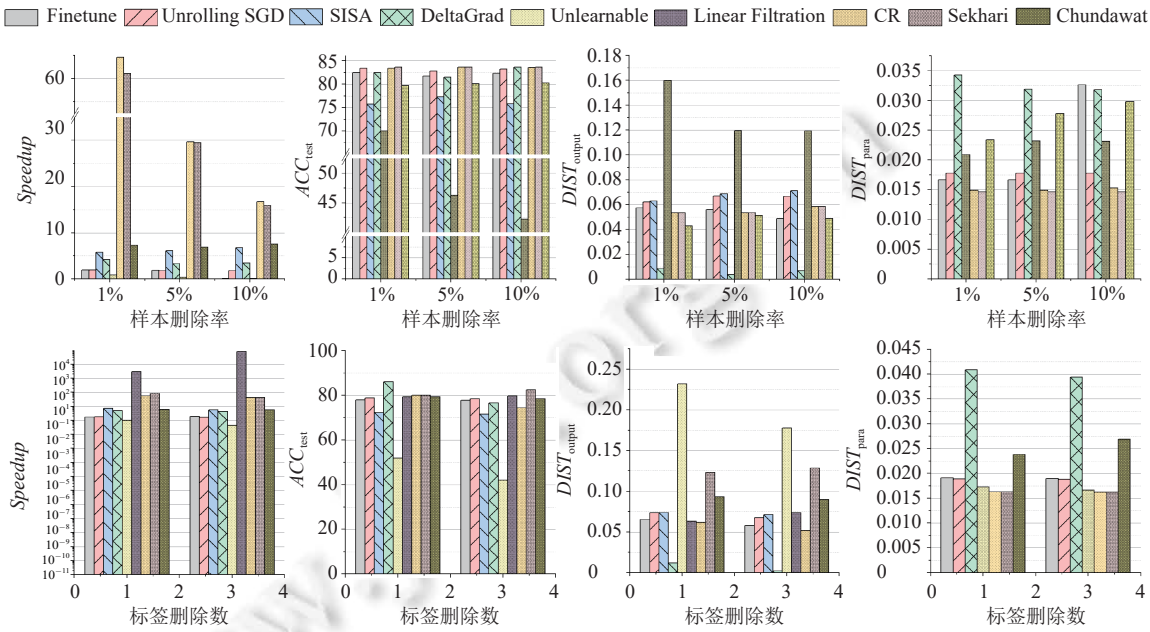


图 2 样本遗忘和标签遗忘场景各方法的性能表现

表 13 机器遗忘方法性能比较

方法名称	类型	速度	可用性	完成度
Finetune	继续训练	低	中	中
Unrolling SGD	继续训练	中	高	中
SISA	混合方法	中	低	高
DeltaGrad	继续训练	低	高	高
Unlearnable	输入编辑	低	低	低
Linear Filtration	输出编辑	高	低	低
CR	参数编辑	中	中	高
Sekhari	参数编辑	高	高	中
Chundawat	深度生成	高	中	低

从速度来看, 继续训练和输入编辑的方法速度较慢, 参数编辑和输出编辑方法速度较快, 具体见“速度”列. Finetune、Unrolling SGD、DeltaGrad 方法的用时与遗忘过程中的训练轮次及超参数设定等有关, 若要提高机器遗忘速度则可减少机器遗忘过程中的训练轮次 (如在 Finetune 和 Unrolling SGD 中减少继续训练轮次), 但模型性能也会受到影响. 混合方法 SISA 将原本在规模较大的数据集上训练模型的过程分割成在若干个小数据块上分别训练子模型的过程, 其重训练子模型的时间与原本在大数据集上的时间之比在理论上与数据块的个数相同, 在实验中, 该比例略低于数据块的个数, 推测是训练子模型的时间除与数据集大小有关外, 还受硬件条件和训练轮次等因素的影响. 输入编辑方法 Unlearnable 需要通过双层优化来生成扰动, 对数据集进行扰动后再在其上训练模型, 耗时较久. 参数编辑方法的耗时集中在计算海瑟矩阵, 输出编辑方法的耗时集中在计算输出映射矩阵, 计算任务相对较小, 速度较快. 深度生成方法 Chundawat 涉及向老师模型学习的过程. 该过程与继续训练方法相似, 用时与遗忘

© 中国科学院软件研究所 http://www.jos.org.cn

过程中的训练轮次有关, 可由用户权衡选择遗忘的用时.

从可用性来看, 继续训练和参数编辑的方法可用性较高, 具体见“可用性”列. 继续训练方法 DeltaGrad 在某些轮次计算精确梯度来更新参数, 在其他轮次采用近似方法来计算梯度以减少用时, 通过训练较大程度地保障了模型的可用性. Unrolling SGD 在对模型进行微调时, 考虑模型在剩余数据集和遗忘数据集上的性能差异来设计损失函数, 使得模型保有较高的可用性. 实验中使用 CR 和 Sekhari 方法时, 均用 LCODEC 来缩小要编辑的参数范围, 故 CR 和 Sekhari 方法得到的遗忘后模型与原模型在测试集上表现有较高的一致度.

从完成度来看, 继续训练、混合方法和参数编辑的机器遗忘方法完成度较高, 具体见“完成度”列. 完成度比较 $DIST_{\text{output}}$ 和 $DIST_{\text{para}}$, 参数编辑的方法 CR 和 Sekhari 均只选择了一部分参数进行更新, $DIST_{\text{para}}$ 较小, 但从输出分布上看, CR 得到的 M_{unlearn} 更接近 M_{retrain}. 在基于训练的方法中, DeltaGrad 基本还原了模型重新训练过程, 与 M_{retrain} 更接近, 而 Finetune 和 Unrolling SGD 均为在 M_0 基础上继续训练, 它们得到的 M_{unlearn} 在本文评估的所有方法中处于中下水平. 混合方法 SISA 会对涉及遗忘数据集的子模型进行重训练, 在与之相同的数据分块训练模式下, 其遗忘后的模型与重训练模型理论上一致, 只是因分块时数据分布的随机性而在性能上有细微差异.

结合以上 3 个方面, 继续训练类方法速度较慢, 但可用性和完成度较高, 且表现比较稳定. 输入编辑和输出编辑类方法可用性与完成度不高. 参数编辑的方法速度较快, 可用性和完成度也属于中上, 适用于需要快速得到 M_{unlearn} 而对模型性能要求较低的场景, 与第 2.4 节和第 3.4 节的结论相符.

6.2.2 标签总数影响分析

对单个标签进行遗忘时, 数据集标签总数对机器遗忘方法的效果影响如何? 为了探究该问题, 本文在一组标签遗忘数不变, 标签总数改变的场景下进行了实验 (具体实验设置见第 6.1.4 节), 图 3 为实验结果, 图中所示为各方法在所有数据集上实验结果的均值.

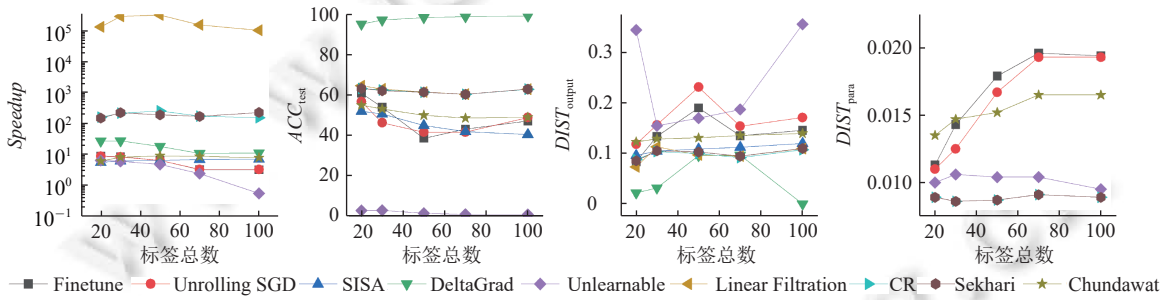


图 3 标签总数改变时各方法性能表现

为了更准确地探究标签总数与机器遗忘方法性能之间的关系, 本文计算了不同标签总数与不同评价指标间的 Spearman 相关系数, 结果如表 14 所示. Spearman 相关系数在 [−1, 1] 之间, Spearman 相关系数为负表示负相关, 反之为正相关, 绝对值越大, 则相关性越强. 如对于机器遗忘方法 Finetune, 其 *Speedup* 和标签总数之间的相关系数为−1.0000, 说明两者成明显的负相关关系.

表 14 不同标签总数与不同评价指标间的 Spearman 相关系数

机器遗忘方法	*Speedup*	ACC_{test}	ACC_{remained}	ACC_{deleted}	$DIST_{\text{output}}$	$DIST_{\text{para}}$
Finetune	−1.0000	−0.9000	−0.7000	−0.9000	0.7000	0.9000
Unrolling SGD	−1.0000	−0.8000	−0.3000	−0.8000	0.5000	0.9747
SISA	−0.8000	−0.9000	−0.3000	−0.6000	0.5000	—
DeltaGrad	−0.8000	−0.9000	−0.2887	−0.9000	0.0000	—
Unlearnable	−1.0000	−0.7071	−1.0000	−0.7071	0.4000	−0.3591
Linear Filtration	−0.3000	0.0000	−0.8944	0.0000	0.3000	—
CR	−0.3000	−0.8944	−0.8660	−0.8944	0.6000	0.4104
Sekhari	0.6000	−0.3536	−0.8660	−0.3536	0.6000	0.4104
Chundawat	−0.3000	−0.9000	−0.3667	−0.2593	0.4000	−0.2108

© 中国科学院软件研究所 http://www.jos.org.cn

根据表 14, 机器遗忘速度与标签总数总体成负相关关系. 各机器遗忘方法的 *Speedup* 与标签总数间的 Spearman 相关系数平均值为−0.5429, 故总体来看, 标签总数越大, *Speedup* 就越小. 实验中基于训练的机器遗忘方法的相关系数的绝对值都较大 (Finetune: −1.0000, Unrolling SGD: −1.0000, DeltaGrad: −0.8000, SISA: −0.8000), 故这种负相关关系在基于训练的机器遗忘方法上尤为明显. 推测其原因, 基于训练的机器遗忘方法所用时间与样本总量密切相关, 当标签总数较大时, 样本总量较大, 训练用时就较长, 从而削弱了机器遗忘的优势. 相比之下, 参数编辑的方法随标签总数变化较小, 原因可能在于计算 M_{unlearn} 时仅为单步计算 (如 CR 使用了牛顿法计算), 不必对数据进行迭代训练, 受数据集大小的影响较弱, 因此受标签总数影响较小. 基于生成的方法受标签总数的影响亦不明显, 推测其原因是在生成模型的训练过程中, 作为训练基准的教师模型鲁棒性较强, 受待删除数据的影响不明显, 因此最终得到的 M_{unlearn} 与标签总数之间的联系较弱.

遗忘可用性与标签总数总体成负相关关系. ACC_{test} 与标签总数间的 Spearman 相关系数平均值为−0.7061, 故随着标签总数增大, ACC_{test} 总体下降. 即标签总数越大, 学习任务难度就越高, 提高 ACC_{test} 的难度也就随之增大.

遗忘完成度与标签总数总体成负相关关系. $DIST_{\text{output}}$、$DIST_{\text{para}}$ 与标签总数间的 Spearman 相关系数平均值分别为 0.4444、0.3542, 说明 $DIST_{\text{output}}$、$DIST_{\text{para}}$ 与标签总数总体成正相关关系, 标签总数越大, $DIST_{\text{output}}$ 和 $DIST_{\text{para}}$ 就越大, 即当标签总数变大时, 更难得到与 M_{retrain} 相似的模型. 综合以上信息, 从遗忘速度、可用性和完成度的角度来看, 标签总数变大使机器遗忘性能有所下降.

6.2.3 度量指标关联分析

对单个标签进行遗忘时, 数据集标签总数对机器遗忘方法的效果影响如何? 在当前机器遗忘的有关研究中, 仍缺少统一评价指标对所有机器遗忘方法进行评价. 挖掘不同评价指标间的联系, 将有助于构建统一评价指标, 用以对各种机器遗忘方法进行全面而统一的评价. 本文根据不同遗忘场景下, 不同机器遗忘方法在不同评价指标上的实验结果来探究评价指标间的联系. 结合实验结果, 本文将评价指标间的联系归纳为图 4, 具体为: *Speedup* 与其他指标的独立关系, ACC_{test}、ACC_{remained}、ACC_{deleted} 三者的关系, ACC_{test}、$DIST_{\text{output}}$、$DIST_{\text{para}}$ 三者的关系. 考虑到不同数据集标签总数和数据内容的相似性, 即 FashionMNIST 和 MNIST、 SVHN 和 CIFAR-10 两组数据集标签总数相同、数据分布相似, 本节选择 MNIST、CIFAR-10、Purchase 和 ImageNet 子集的实验结果进行分析.

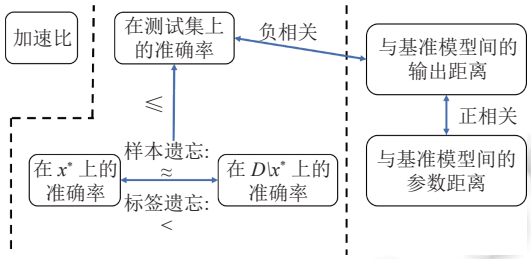


图 4 度量指标关系图

(1) *Speedup* 与其他指标的独立性

机器遗忘速度的评价指标为加速比, 也可用机器遗忘耗时代替. 该指标与机器遗忘流程和计算量有关, 独立于机器遗忘可用性和完成度.

(2) ACC_{test}、ACC_{remained}、ACC_{deleted} 三者的关系

观察 ACC_{test}、ACC_{remained} 和 ACC_{deleted}, 我们发现: 在样本遗忘场景下, ACC_{remained} 与 ACC_{deleted} 接近, 两者总体高于 ACC_{test}; 在标签遗忘场景下, ACC_{remained} 与 ACC_{deleted} 相差较大, ACC_{remained} 高于 ACC_{test}.

以上结论验证过程如下: 经计算, 在样本遗忘场景下, ACC_{remained} 与 ACC_{deleted} 之差的绝对值均值为 1.91, 而在标签遗忘场景下, 该值为 35.93, 故在标签遗忘场景下, M_{unlearn} 在 x^* 和 $D\backslash x^*$ 上的表现差异大于样本遗忘场景. 推测其原因, 在样本遗忘场景下, x^* 和 $D\backslash x^*$ 包含各种标签, 而在标签遗忘场景下, x^* 和 $D\backslash x^*$ 标签无交集. 故在样本遗忘场景中 x^* 和 $D\backslash x^*$ 数据分布差异小于标签遗忘场景, 遗忘模型在两者上的表现差异较小.

© 中国科学院软件研究所 http://www.jos.org.cn

此外, 在样本遗忘场景下, ACC_{test} 与 $ACC_{remained}$ 和 $ACC_{deleted}$ 均值之差为−9.40; 在标签遗忘场景下, 由于测试集和 $D\backslash x^*$ 标签集相同, 且均与包含 x^* 标签集无交集, 故本文计算了 ACC_{test} 与 $ACC_{remained}$ 之差, 为−12.74. 故总体来看, $M_{unlearn}$ 在 x^* 和 $D\backslash x^*$ 上的表现优于模型在测试集上的表现, 在标签遗忘场景下, 两者的差异更为明显.

(3) ACC_{test}、$DIST_{output}$、$DIST_{para}$ 三者的关系

观察 ACC_{test}、$DIST_{output}$ 和 $DIST_{para}$, 我们发现: ACC_{test} 和 $DIST_{output}$ 成负相关关系; ACC_{test} 和 $DIST_{para}$ 成负相关关系; $DIST_{output}$ 与 $DIST_{para}$ 成正相关关系.

为了验证以上结论, 本文以不同遗忘场景中, 不同机器遗忘方法得到的机器遗忘模型在 3 个评价指标上的数值为对象, 计算三者之间的 Spearman 相关系数, 结果如表 15 所示. 据计算, ACC_{test} 与 $DIST_{output}$ 的 Spearman 相关系数均值为−0.6399, 两者有较强的负相关关系; ACC_{test} 和 $DIST_{para}$ 的 Spearman 相关系数均值为−0.4867, 两者同样有负相关关系; $DIST_{output}$ 与 $DIST_{para}$ 的 Spearman 相关系数均值为 0.6359, 两者有正相关关系.

表 15 不同评价指标的 Spearman 相关系数

遗忘场景	数据集	ACC_{test}和$DIST_{output}$	ACC_{test}和$DIST_{para}$	$DIST_{output}$与$DIST_{para}$
样本遗忘率为1%	MNIST	−0.7818	−0.8182	0.7455
	CIFAR-10	−0.5429	−0.1177	0.3825
	Purchase	−0.8929	−0.3214	0.4286
	ImageNet子集	−0.7537	−0.6377	0.7647
样本遗忘率为5%	MNIST	−0.4286	−0.5946	0.7388
	CIFAR-10	0.0286	−0.1160	0.8117
	Purchase	−0.6429	−0.7857	0.4286
	ImageNet子集	−0.7714	−0.6179	0.8827
样本遗忘率为10%	MNIST	−0.7818	−0.8182	0.7455
	CIFAR-10	−0.5429	−0.1177	0.3825
	Purchase	−0.8929	−0.3214	0.4286
	ImageNet子集	−0.7714	−0.6377	0.7537
遗忘1个标签	MNIST	−0.2857	−0.6307	0.7388
	CIFAR-10	−0.6571	−0.1160	0.6088
	ImageNet子集	−0.7143	−0.6377	0.7537
遗忘3个标签	MNIST	−0.7143	−0.8929	0.7500
	CIFAR-10	−0.4857	0.1449	0.3479
	ImageNet子集	−0.8857	−0.7247	0.7537
均值		−0.6399	−0.4867	0.6359

推测这种相关性出现的原因, 我们认为: 参数距离和输出距离均用于以衡量 $M_{unlearn}$ 与 $M_{retrain}$ 的相似度, 参数上与 $M_{retrain}$ 接近的模型更有可能输出与 $M_{retrain}$ 相同的值. 当 $DIST_{output}$ 和 $DIST_{para}$ 较小时, 认为 $M_{unlearn}$ 的表现接近 $M_{retrain}$ 的表现, 而由于 $M_{retrain}$ 在测试集上的表现较为稳定, 故 $M_{unlearn}$ 在测试集上的表现也相对较好, 因此参数距离或输出距离越小, 在测试集上的表现就越好.

6.2.4 机器遗忘适用的场景分析

机器遗忘方法适用的场景是什么? 机器遗忘的目标是以低于重新训练的代价, 得到与重训练模型尽可能相似的模型, 故存在遗忘速度、可用性与完成度的权衡. 以此为出发点, 本节结合实验结果, 探究在哪些场景下适合应用机器遗忘方法, 结论如下: 输入编辑方法不适合应用于实际机器遗忘场景; 当模型较大时, 适合使用机器遗忘方法; 当 $M_{retrain}$ 在 $D\backslash x^*$ 和 x^* 上表现差异较大时, 进行机器遗忘必要性较大. 接下来将具体分析以上结论.

输入编辑方法不适合应用于实际机器遗忘场景. 输入编辑方法是指编辑原训练集 D, 从而影响在其上训练的模型行为的方法. 就机器遗忘应用而言, 若要编辑 D, 在其上训练模型, 则不如直接将 D 中遗忘数据集 x^* 删去, 在剩余数据集 $D\backslash x^*$ 上训练模型. 就训练流程来说, 后者的数据集小于前者的数据集, 在超参数设置一致的情况下, 后者用时少于前者; 且后者本身就是 $M_{retrain}$, 与 $M_{retrain}$ 表现一致, 已经实现了目标.

© 中国科学院软件研究所 http://www.jos.org.cn

当模型较大时, 适合使用机器遗忘方法. 当模型较小 (如多层感知机) 时, 重新训练模型本身用时较少, 机器遗忘方法加速比较小. 在本次实验中, Purchase 和 MNIST 所用模型较小, Purchase 使用由 2 层全连接层构成的模型, MNIST 使用由 2 层卷积层和 2 层全连接层构成的模型; CIFAR-10 和 ImageNet 子集所用模型较大, 为 ResNet-18. 经计算, 在样本遗忘场景下, 不同机器遗忘方法在 Purchase、MNIST、CIFAR-10、ImageNet 子集数据集上的平均加速比分别为 1.35、3.84、8.91、37.75, 可见在模型较小时, 机器遗忘方法的加速效果并不明显.

当 M_{retrain} 在 $D\backslash x^*$ 和 x^* 上表现差异较大时, 进行机器遗忘必要性较大. 直观来看, 若从 D 中遗忘一部分数据, 则在 $D\backslash x^*$ 上训练的模型应当在 x^* 上表现较差, 才有"遗忘"的效果. 但在实验过程中, 我们发现 M_{retrain} 未必在 x^* 上表现不佳: 在样本遗忘场景下, Purchase 和 MNIST 数据集上, ACC_{remained} 和 ACC_{deleted} 之差的平均值为 0.271, 而在 CIFAR-10 和 ImageNet 子集数据集上, 该差值为 29.182. 可见存在这样的场景, M_{retrain} 在 $D\backslash x^*$ 和 x^* 上表现相近. 在这种场景下, 我们认为遗忘数据集包含的信息本身较少, 遗忘这部分数据集并不能让模型包含的信息产生较大的差别, 故机器遗忘必要性较小. 相反, 若 M_{retrain} 在 $D\backslash x^*$ 和 x^* 上表现差异较大, 则认为遗忘这部分数据集确实能消除模型中的一部分信息, 进行机器遗忘必要性较大.

以上结论是对机器遗忘方法实用性的分析, 可见判断机器学习场景下机器遗忘方法是否值得使用, 与训练集、模型和训练过程均有关. 此外, 结合第 3 点结论, 如何在不必重新训练模型的情况下判断 M_{retrain} 在 $D\backslash x^*$ 和 x^* 上的表现差异, 作为预先判断该场景是否需要进行机器遗忘的依据, 留待人们未来去研究.

7 未来的工作

机器遗忘研究当前集中于遗忘方法, 未来可从遗忘数据多样化、遗忘计算多方化、遗忘评估标准化等角度进一步探索.

7.1 遗忘数据多样化

目前机器遗忘针对的大多是表格数据, 而在其他数据类型 (如图数据和时序数据等) 进行遗忘的工作较少, 仍有较大发掘空间. 以图数据为例, 图数据以点和边为具体存储单元, 在图数据中, 数据间关系和数据点同样重要. Chen 等人[83]研究了 SISA 方法应用于图数据时的划分不平衡问题, 提出了更适应图数据特点的划分方法. Zhu 等人[84]考虑了异质图场景下利用联邦学习来训练图表征以及对其进行遗忘的问题, 提出了联邦知识图谱表示学习与对应的机器遗忘框架 FedLU. 此外, 人们可考虑怎样改进其他机器遗忘方法在图数据上的使用, 如设计输入编辑类方法时, 考虑数据分布的特性, 若 x^* 恰好位于某个子图中, 则只对这一部分子图数据进行加噪, 等等.

对于时序数据, Li 等人[85]研究了在时序数据上应用线性回归模型, 遗忘时间窗口以外的数据. Mirzasoleiman 等人[86]则从流数据出发, 设计了支持遗忘的流数据摘要方法. 时序数据和流数据都按一定顺序排列, 数据间关联程度强, 且具有较强的规律性. 对此, 研究者们可探究在时序数据或流数据上执行不同机器学习任务时, 如何利用这些数据特点改进机器遗忘. 此外, 对于流数据不断到来的特点, 研究者们可探究如何进行在线遗忘, 以及在流数据上使用机器学习时, 若用户的遗忘请求与新数据同时到来, 如何平衡遗忘旧数据和记住新数据的问题.

随着大语言模型的兴起, 当前也出现了少量在大模型场景下, 针对文本数据的机器遗忘工作[87–91]. 这类工作中的"遗忘"往往以规范模型行为为目的, 即避免让模型输出有害、过时、带有偏见或涉及用户隐私的信息, 使用方法可归为两类: 一类是采用优化求解的思想, 设计损失函数来调整模型行为; 另一类是采用继续计算思想, 通过引入新数据, 在新数据上微调模型以修正模型行为.

不同数据组织方式, 其上的计算任务也不尽相同. 对此, 研究者们可挖掘不同数据类型的计算任务, 从机器学习以外的任务中解决遗忘问题. 例如, Mirzasoleiman 等人[86]研究了流数据的摘要计算, 此外, 还有聚合与划分等计算任务, 研究者们可进一步思考这些任务上的遗忘问题.

7.2 遗忘计算多方化

联邦学习是一种分布式学习框架, 通过多轮客户端计算和与中心服务器端通信, 训练共享模型. 和中心化模型不同, 联邦学习计算任务分配给多个客户端, 服务器端不能获取所有人的数据. 在联邦学习中, 若某用户最初愿意

© 中国科学院软件研究所 http://www.jos.org.cn

参与联邦学习, 但在后期训练过程中决定撤回其客户端数据的授权, 并希望从共享模型中删除个人数据, 则会发出遗忘请求. 与中心化模型的机器遗忘相比, 联邦学习中的机器遗忘需考虑客户端和服务器端的交互.

目前有部分文献针对联邦学习框架下的部分算法进行了探索[92–95]. 例如, 对于传统机器学习算法, Liu 等人[92]针对联邦学习框架下随机森林算法提出了遗忘方法 RevFRF. Gong 等人[95]针对贝叶斯推断场景提出了 Forget-SVGD. 对于深度学习, Wang 等人[96]研究了使用卷积神经网络做图像分类场景下, 如何遗忘某个特定标签的数据, Che 等人[97]考虑边缘计算训练局部机器学习模型的场景, 利用非线性函数分析技术来修正局部模型, 使得遗忘更快速.

对于联邦学习框架下机器遗忘的未来研究, 首先可就其针对的机器学习算法继续扩展. 对于传统机器学习模型, 除了已有的随机森林算法外, 还可考虑联邦学习框架下的逻辑回归和聚类算法等; 对于深度学习, 还可考虑循环神经网络等模型. 其次, 联邦学习自身存在的问题在机器遗忘中仍需注意. 比如, 联邦学习中的通信开销, 在机器遗忘中, 当遗忘请求到来时, 应当尽可能减少需要更新的数据量; 遗忘过程中, 可能存在客户端突然宕机的情况, 在该情况下如何处理中断的遗忘, 以及客户端重新连接后, 是否应恢复到遗忘前的状态, 等等, 均有待讨论.

7.3 遗忘评估标准化

前文已经提到, 机器遗忘多数文献都是在自定义的理论体系下对遗忘方法进行探索, 甚至对“遗忘”的定义都有差异. 对此, 目前仍缺少一个标准化评估框架, 如给定统一测试数据集和评估流程, 评估不同定义下机器遗忘方法的优劣. 由于机器遗忘场景的多样性, 在设计该评估框架时, 需对问题进行全面的考量. 比如, 测试数据集需包含删除多样本和删除多标签的场景, 以及需考虑多样本来自不同标签或相同标签的场景; 在进行机器遗忘评估时, 需统一计算各种指标, 如加速比和准确率等. 整个评估框架需全面考虑遗忘的场景, 并对模型遗忘性能有综合展现.

此外, 研究者可进一步思考, 设计统一指标来对机器遗忘方法进行评估. 在第 5 节中, 本文提出了 3 个评估角度, 即遗忘速度、可用性和完成性, 未来若有统一指标, 应与这 3 个角度都有所关联, 即当遗忘模型可用性高, 参数距离小或输出距离小时, 该指标也会相应地提高. 如果存在这样的统一指标, 人们在评估机器遗忘方法或比较大量机器遗忘方法时, 即可以此为标准, 筛选出最适合应用的算法, 这对于机器遗忘从理论投入实际应用具有重大意义.

本节主要从遗忘数据多样化、遗忘计算多方化、遗忘评估标准化这 3 个角度展望未来的工作, 此外, 研究者也可以考虑机器遗忘本身是否会侵犯人们隐私[98]等问题. 机器遗忘仍处于蓬勃发展阶段, 未来工作的开展应重视其落地应用, 使其与生活实际更好地结合.

8 结束语

由于机器学习模型会记住用户信息, 单从数据集中删除用户数据并不能消除数据对模型的影响, 由此引出了机器遗忘这一问题. 机器遗忘研究如何以较小的代价获得与重新训练模型相似的模型. 本文首先对机器遗忘目标进行了定义, 随后介绍了基于训练、基于编辑和基于生成的机器遗忘方法, 概括了目前机器遗忘中用以衡量遗忘效果的度量指标, 并对深度学习中的机器遗忘方法进行了统一实验, 比较这些方法的性能, 得出参数编辑类方法速度较快, 可用性和完成度也处于中上; 继续训练类方法遗忘可用性和完成度较高, 但速度较慢等结论. 最后, 结合当前发展情况, 本文提出了遗忘数据多样化、遗忘计算多方化和遗忘评估标准化这 3 个未来的研究方向. 在机器学习得到广泛应用的当下, 让模型拥有“遗忘”用户信息的能力, 是信息时代个人隐私保护的重要组成部分.

References:

[1] Shokri R, Shmatikov V. Privacy-preserving deep learning. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security. Denver: ACM, 2015. 1310–1321. [doi: 10.1145/2810103.2813687]

[2] Schelter S. “Amnesia”—Machine learning models that can forget user data very fast. In: Proc. of the 10th Conf. on Innovative Data Systems Research. 2020.

[3] Zanella-Béguelin S, Wutschitz L, Tople S, Rühle V, Paverd A, Ohrimenko O, Köpf B, Brockschmidt M. Analyzing information leakage of updates to natural language models. In: Proc. of the 2020 ACM SIGSAC Conf. on Computer and Communications Security. Virtual

© 中国科学院软件研究所 http://www.jos.org.cn

Event: ACM, 2020. 363–375. [doi: 10.1145/3372297.3417880]
[4] Shokri R, Stronati M, Song CZ, Shmatikov V. Membership inference attacks against machine learning models. In: Proc. of the 2017 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2017. 3–18. [doi: 10.1109/SP.2017.41]
[5] Newman AL. What the "right to be forgotten" means for privacy in a digital age. Science, 2015, 347(6221): 507–508. [doi: 10.1126/science.aaa4603]
[6] General Administration of Quality Supervision, Inspection and Quarantine of the People's Republic of China, Standardization Administration. GB/T 35273-2020 Information security technology—Personal information security specification. Beijing: Standards Press of China, 2018 (in Chinese). https://std.samr.gov.cn/gb/search/gbDetailed?id=A0280129495AEBB4E05397BE0A0AB6FE
[7] European Commission. Regulation of the European Parliament and of the Council on the protection of individuals with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). 2012. https://gdpr.eu/article-17-right-to-be-forgotten/
[8] Kwak C, Lee J, Lee H. Forming a dimension of digital human rights: Research agenda for the right to be forgotten. In: Proc. of the 50th Hawaii Int'l Conf. on System Sciences. Hilton Waikoloa Village: IEEE, 2017. 982–989.
[9] Cao YZ, Yang JF. Towards making systems forget with machine unlearning. In: Proc. of the 2015 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2015. 463–480. [doi: 10.1109/SP.2015.35]
[10] Sharir O, Peleg B, Shoham Y. The cost of training NLP models: A concise overview. arXiv:2004.08900, 2020.
[11] Guo C, Goldstein T, Hannun A, van der Maaten L. Certified data removal from machine learning models. In: Proc. of the 37th Int'l Conf. on Machine Learning. 2020. 3832–3842.
[12] Liu Y, Fan MY, Chen C, Liu XM, Ma Z, Wang L, Ma JF. Backdoor defense with machine unlearning. In: Proc. of the 2022 IEEE Conf. on Computer Communications (INFOCOM 2022). London: IEEE, 2022. 280–289. [doi: 10.1109/INFOCOM48880.2022.9796974]
[13] Cao YZ, Yu AF, Adat A, Stahl E, Merwine J, Yang JF. Efficient repair of polluted machine learning systems via causal unlearning. In: Proc. of the 2018 Asia Conf. on Computer and Communications Security. Incheon: ACM, 2018. 735–747. [doi: 10.1145/3196494.3196517]
[14] Wang BL, Yao YS, Shan S, Li HY, Viswanath B, Zheng HT, Zhao BY. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: Proc. of the 2019 IEEE Symp. on Security and Privacy. San Francisco: IEEE, 2019. 707–723. [doi: 10.1109/SP.2019.00031]
[15] Nguyen TT, Huynh TT, Nguyen PL, Liew AWC, Yin HZ, Nguyen QVH. A survey of machine unlearning. arXiv:2209.02299, 2022.
[16] Xu J, Wu ZH, Wang C, Jia XH. Machine unlearning: Solutions and challenges. IEEE Trans. on Emerging Topics in Computational Intelligence, 2024, 8(3): 2150–2168. [doi: 10.1109/TETCI.2024.3379240]
[17] Zhang HB, Nakamura T, Isohara T, Sakurai K. A review on machine unlearning. SN Computer Science, 2023, 4(4): 337. [doi: 10.1007/s42979-023-01767-4]
[18] Xu H, Zhu TQ, Zhang LF, Zhou WL, Yu PS. Machine unlearning: A survey. ACM Computing Surveys, 2023, 56(1): 9. [doi: 10.1145/3603620]
[19] Liu X, Tsaftaris SA. Have you forgotten? A method to assess if machine learning models have forgotten data. In: Proc. of the 23rd Int'l Conf. on Medical Image Computing and Computer Assisted Intervention (MICCAI 2020). Lima: Springer, 2020. 95–105. [doi: 10.1007/978-3-030-59710-8_10]
[20] Golatkar A, Achille A, Soatto S. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In: Proc. of the 2020 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Seattle: IEEE, 2020. 9301–9309. [doi: 10.1109/CVPR42600.2020.00932]
[21] Thudi A, Deza G, Chandrasekaran V, Papernot N. Unrolling SGD: Understanding factors influencing machine unlearning. In: Proc. of the 7th IEEE European Symp. on Security and Privacy (EuroS&P). Genoa: IEEE, 2022. 303–319. [doi: 10.1109/EuroSP53844.2022.00027]
[22] Chai CL, Wang JY, Luo YY, Niu ZP, Li GL. Data management for machine learning: A survey. IEEE Trans. on Knowledge and Data Engineering, 2023, 35(5): 4646–4667. [doi: 10.1109/TKDE.2022.3148237]
[23] Liu B, Liu Q, Stone P. Continual learning and private unlearning. arXiv:2203.12817, 2022.
[24] Brophy J, Lowd D. Machine unlearning for random forests. In: Proc. of the 38th Int'l Conf. on Machine Learning. 2021. 1092–1104.
[25] Schelter S, Grafberger S, Dunning T. HedgeCut: Maintaining randomised trees for low-latency machine unlearning. In: Proc. of the 2021 Int'l Conf. on Management of Data. Virtual Event: ACM, 2021. 1545–1557. [doi: 10.1145/3448016.3457239]
[26] Bourtoule L, Chandrasekaran V, Choquette-Choo CA, Jia HR, Travers A, Zhang BW, Lie D, Papernot N. Machine unlearning. In: Proc. of the 2021 IEEE Symp. on Security and Privacy. San Francisco: IEEE, 2021. 141–159. [doi: 10.1109/SP40001.2021.00019]
[27] Felps DL, Schwickerath AD, Williams JD, Vuong TN, Briggs A, Hunt M, Sakmar E, Saranchak DD, Shumaker T. Class clown: Data redaction in machine unlearning at enterprise scale. arXiv:2012.04699, 2020.

© 中国科学院软件研究所 http://www.jos.org.cn

[28] Chen YT, Xiong J, Xu WH, Zuo JW. A novel online incremental and decremental learning algorithm based on variable support vector machine. Cluster Computing, 2019, 22(S3): 7435–7445. [doi: 10.1007/s10586-018-1772-4]

[29] Khan ME, Swaroop S. Knowledge-adaptation priors. In: Proc. of the 35th Int'l Conf. on Neural Information Processing Systems. Curran Associates Inc., 2021. 19757–19770.

[30] Baumhauer T, Schöttle P, Zeppelzauer M. Machine unlearning: Linear filtration for logit-based classifiers. Machine Learning, 2022, 111(9): 3203–3226. [doi: 10.1007/s10994-022-06178-9]

[31] Kim J, Woo SS. Efficient two-stage model retraining for machine unlearning. In: Proc. of the 2022 IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops. New Orleans: IEEE, 2022. 4360–4368. [doi: 10.1109/CVPRW56347.2022.00482]

[32] Neel S, Roth A, Sharifi-Malvajerdi S. Descent-to-delete: Gradient-based methods for machine unlearning. In: Proc. of the 32nd Int'l Conf. on Algorithmic Learning Theory. 2021. 931–962.

[33] Nguyen QP, Oikawa R, Divakaran DM, Chan MC, Low BKH. Markov chain Monte Carlo-based machine unlearning: Unlearning what needs to be forgotten. In: Proc. of the 2022 ACM on Asia Conf. on Computer and Communications Security. Nagasaki: ACM, 2022. 351–363. [doi: 10.1145/3488932.3517406]

[34] Zeng YY, Wang TH, Chen S, Just HA, Jin R, Jia RX. Learning to refit for convex learning problems. arXiv:2111.12545, 2022.

[35] Gao J, Garg S, Mahmoody M, Vasudevan PN. Deletion inference, reconstruction, and compliance in machine (un)learning. Proc. on Privacy Enhancing Technologies, 2022(3): 415–436. [doi: 10.56553/popets-2022-0079]

[36] Marchant NG, Rubinstein BIP, Alfeld S. Hard to forget: Poisoning attacks on certified machine unlearning. In: Proc. of the 36th AAAI Conf. on Artificial Intelligence. Virtually: AAAI, 2022. 7691–7700. [doi: 10.1609/aaai.v36i7.20736]

[37] Hu HS, Wang S, Chang JM, Zhong HN, Sun RX, Hao S, Zhu HJ, Xue MH. A duty to forget, a right to be assured? Exposing vulnerabilities in machine unlearning services. arXiv:2309.08230, 2024.

[38] Yoon Y, Nam J, Yun H, Kim D, Ok J. Few-shot unlearning by model inversion. arXiv:2205.15567, 2023.

[39] Wu YJ, Dobriban E, Davidson S. DeltaGrad: Rapid retraining of machine learning models. In: Proc. of the 37th Int'l Conf. on Machine Learning. 2020. 10355–10366.

[40] Ginart AA, Guan MY, Valiant G, Zou J. Making AI forget you: Data deletion in machine learning. In: Proc. of the 33rd Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019. 3518–3531.

[41] Fu SP, He FX, Tao DC. Knowledge removal in sampling-based Bayesian inference. arXiv:2203.12964, 2022.

[42] Mehta R, Pal S, Singh V, Ravi SN. Deep unlearning via randomized conditionally independent Hessians. In: Proc. of the 2022 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. New Orleans: IEEE, 2022. 10412–10421. [doi: 10.1109/CVPR52688.2022.01017]

[43] Izzo Z, Smart MA, Chaudhuri K, Zou J. Approximate data deletion from machine learning models. In: Proc. of the 24th Int'l Conf. on Artificial Intelligence and Statistics. 2021. 2008–2016.

[44] Huang HX, Ma XJ, Erfani SM, Bailey J, Wang YS. Unlearnable examples: Making personal data unexploitable. arXiv:2101.04898, 2021.

[45] Fu SP, He FZ, Xu Y, Tao DC. Bayesian inference forgetting. arXiv:2101.06417, 2021.

[46] Kurmanji M, Triantafillou P, Hayes J, Triantafillou E. Towards unbounded machine unlearning. In: Proc. of the 37th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2023. 1957–1987.

[47] Parne N, Puppaala K, Bhupathi N, Patgiri R. An investigation on learning, polluting, and unlearning the spam emails for lifelong learning. arXiv:2111.14609, 2021.

[48] Ye JW, Fu YF, Song J, Yang XY, Liu SH, Jin X, Song ML, Wang XC. Learning with recoverable forgetting. In: Proc. of the 17th European Conf. on Computer Vision. Tel Aviv: Springer, 2022. 87–103. [doi: 10.1007/978-3-031-20083-0_6]

[49] Chen KY, Wang YW, Huang Y. Lightweight machine unlearning in neural network. arXiv:2111.05528, 2021.

[50] Aldaghri N, Mahdavifar H, Beirami A. Coded machine unlearning. IEEE Access, 2021, 9: 88137–88150. [doi: 10.1109/ACCESS.2021.3090019]

[51] Yan HN, Li XG, Guo ZY, Li H, Li FH, Lin XD. ARCANE: An efficient architecture for exact machine unlearning. In: Proc. of the 31st Int'l Joint Conf. on Artificial Intelligence (IJCAI 2022). Vienna: Morgan Kaufmann, 2022. 4006–4013. [doi: 10.24963/ijcai.2022/556]

[52] He YZ, Meng GZ, Chen K, He JW, Hu XB. DeepObliviate: A powerful charm for erasing data residual memory in deep neural networks. arXiv:2105.06209, 2021.

[53] Gupta V, Jung C, Neel S, Roth A, Sharifi-Malvajerdi S, Waites C. Adaptive machine unlearning. In: Proc. of the 35th Int'l Conf. on Neural Information Processing Systems. Curran Associates Inc., 2021. 16319–16330.

[54] Chundawat VS, Tarun AK, Mandal M, Kankanhalli M. Zero-shot machine unlearning. arXiv:2201.05629, 2023.

[55] Chen C, Sun F, Zhang M, Ding BL. Recommendation unlearning. In: Proc. of the 2022 ACM Web Conf. Virtual Event: ACM, 2022. 2768–2777. [doi: 10.1145/3485447.3511997]

© 中国科学院软件研究所 http://www.jos.org.cn

[56] Golatkar A, Achille A, Ravichandran A, Polito M, Soatto S. Mixed-privacy forgetting in deep networks. In: Proc. of the 2021 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Nashville: IEEE, 2021. 792–801. [doi: 10.1109/CVPR46437.2021.00085]

[57] Peste A, Alistarh D, Lampert CH. SSSE: Efficiently erasing samples from trained machine learning models. arXiv:2107.03860, 2021.

[58] Golatkar A, Achille A, Soatto S. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In: Proc. of the 16th European Conf. on Computer Vision (ECCV 2020). Glasgow: Springer, 2020. 383–398. [doi: 10.1007/978-3-030-58526-6_23]

[59] Dwork C, Lei J. Differential privacy and robust statistics. In: Proc. of the 41st Annual ACM Int'l Symp. on Theory of Computing. Bethesda: ACM, 2009. 371–380. [doi: 10.1145/1536414.1536466]

[60] Liu JX, Xue MS, Lou J, Zhang XY, Xiong L, Qin Z. MUter: Machine unlearning on adversarially trained models. In: Proc. of the 2023 IEEE/CVF Int'l Conf. on Computer Vision. Paris: IEEE, 2023. 4869–2879. [doi: 10.1109/ICCV51070.2023.00451]

[61] Du M, Chen Z, Liu C, Oak R, Song D. Lifelong anomaly detection through unlearning. In: Proc. of the 2019 ACM SIGSAC Conf. on Computer and Communications Security. London: ACM. 2019. 1283–1297. [doi: 10.1145/3319535.3363226]

[62] Liu Y, Ma Z, Liu XM, Liu J, Jiang ZY, Ma JF, Yu P, Ren K. Learn to forget: Machine unlearning via neuron masking. arXiv:2003.10933, 2021.

[63] Ganhör C, Penz D, Rekabsaz N, Lesota O, Schedl M. Unlearning protected user attributes in recommendations with adversarial training. In: Proc. of the 45th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. Madrid: ACM, 2022. 2142–2147. [doi: 10.1145/3477495.3531820]

[64] Wu G, Hashemi M, Srinivasa C. PUMA: Performance unchanged model augmentation for training data removal. In: Proc. of the 36th AAAI Conf. on Artificial Intelligence. Virtually: AAAI, 2022. 8675–8682. [doi: 10.1609/aaai.v36i8.20846]

[65] Ye DY, Zhu TQ, Zhu CC, Wang DR, Shi ZW, Shen S, Zhou WL, Xue MH. Reinforcement unlearning. arXiv:2312.15910, 2024.

[66] Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In: Proc. of the 27th Int'l Conf. on Neural Information Processing Systems. Montreal: MIT Press, 2014. 2672–2680.

[67] Ullah E, Mai T, Rao A, Rossi RA, Arora R. Machine unlearning via algorithmic stability. In: Proc. of the 34th Conf. on Learning Theory. 2021. 4126–4142.

[68] Nguyen QP, Kian B, Low H, Jaillet P. Variational Bayesian unlearning. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 16025–16036.

[69] Chen KY, Huang Y, Wang YW. Machine unlearning via GAN. arXiv:2111.11869, 2021.

[70] Chundawat VS, Tarun AK, Mandal M, Kankanhalli M. Can bad teaching induce forgetting? Unlearning in deep networks using an incompetent teacher. arXiv:2205.08096, 2023.

[71] Zhang PF, Bai GD, Huang Z, Xu XS. Machine unlearning for image retrieval: A generative scrubbing approach. In: Proc. of the 30th ACM Int'l Conf. on Multimedia. Lisboa: ACM, 2022. 237–245. [doi: 10.1145/3503161.3548378]

[72] Thudi A, Jia HR, Shumailov I, Papernot N. On the necessity of auditable algorithmic definitions for machine unlearning. In: Proc. of the 31st USENIX Security Symp. (USENIX Security 22). Boston: USENIX Association, 2022. 4007–4022.

[73] Huang Y, Li XX, Li K. EMA: Auditing data removal from trained models. In: Proc. of the 24th Int'l Conf. on Medical Image Computing and Computer Assisted Intervention (MICCAI 2021). Strasbourg: Springer, 2021. 793–803. [doi: 10.1007/978-3-030-87240-3_76]

[74] Goel S, Prabhu A, Kumaraguru P. Evaluating inexact unlearning requires revisiting forgetting. arXiv:2201.06640, 2023.

[75] Warnecke A, Pirch L, Wressnegger C, Rieck K. Machine unlearning of features and labels. arXiv:2108.11577, 2023.

[76] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc. of the IEEE, 1998, 86(11): 2278–2324. [doi: 10.1109/5.726791]

[77] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. Technical Report, University of Toronto, 2009. https://www.cs.toronto.edu/%7Ekriz/cifar.html

[78] Sakar CO, Polat SO, Katircioglu M, Kastro Y. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks. Neural Computing & Applications, 2019, 31(10): 6893–6908. [doi: 10.1007/s00521-018-3523-0]

[79] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. ImageNet: A large-scale hierarchical image database. In: Proc. of the 2009 IEEE Conf. on Computer Vision and Pattern Recognition. Miami: IEEE, 2009. 248–255. [doi: 10.1109/CVPR.2009.5206848]

[80] Xiao H, Rasul K, Vollgraph R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747, 2017.

[81] Goodfellow IJ, Bulatov Y, Ibarz J, Arnoud S, Shet V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. arXiv:1312.6082, 2014.

[82] Sekhari A, Acharya J, Kamath G, Suresh AT. Remember what you want to forget: Algorithms for machine unlearning. In: Proc. of the

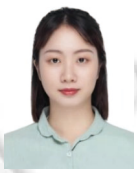
© 中国科学院软件研究所 http://www.jos.org.cn

35th Int'l Conf. on Neural Information Processing Systems. Online: Curran Associates Inc., 2021. 18075–18086.
[83] Chen M, Zhang ZK, Wang TH, Backes M, Humbert M, Zhang Y. Graph unlearning. In: Proc. of the 2022 ACM SIGSAC Conf. on Computer and Communications Security. Los Angeles: ACM, 2022. 499–513. [doi: 10.1145/3548606.3559352]
[84] Zhu XR, Li GY, Hu W. 2023. Heterogeneous federated knowledge graph embedding learning and unlearning. In: Proc. of the 2023 ACM Web Conf. Austin: ACM, 2023. 2444–2454. [doi: 10.1145/3543507.3583305]
[85] Li YT, Wang CH, Cheng G. Online forgetting process for linear regression models. In: Proc. of the 24th Int'l Conf. on Artificial Intelligence and Statistics. 2021. 217–225.
[86] Mirzasoleiman B, Karbasi A, Krause A. Deletion-robust submodular maximization: Data summarization with "the right to be forgotten". In: Proc. of the 34th Int'l Conf. on Machine Learning. 2017. 2449–2458.
[87] Yu C, Jeoung S, Kasi A, Yu PF, Ji H. Unlearning bias in language models by partitioning gradients. In: Findings of the Association for Computational Linguistics. Toronto: ACL, 2023. 6032–6048. [doi: 10.18653/v1/2023.findings-acl.375]
[88] Yao YS, Xu XJ, Liu Y. Large language model unlearning. arXiv:2310.10683, 2024.
[89] Chen JA, Yang DY. Unlearn what you want to forget: Efficient unlearning for LLMs. arXiv:2310.20150, 2023.
[90] Liu Z, Kalinli O. Forgetting private textual sequences in language models via leave-one-out ensemble. arXiv:2309.16082, 2023.
[91] Ni SW, Chen DW, Li CM, Hu XP, Xu RF, Yang M. Forgetting before learning: Utilizing parametric arithmetic for knowledge updating in large language models. arXiv:2311.08011, 2024.
[92] Liu GY, Ma XQ, Yang Y, Wang C, Liu JC. FedEraser: Enabling efficient client-level data removal from federated learning models. In: Proc. of the 29th IEEE/ACM Int'l Symp. on Quality of Service. Tokyo: IEEE, 2021. 1–10. [doi: 10.1109/IWQOS52092.2021.9521274]
[93] Liu Y, Xu L, Yuan XL, Wang C, Li B. The right to be forgotten in federated learning: An efficient realization with rapid retraining. In: Proc. of the IEEE INFOCOM 2022—IEEE Conf. on Computer Communications. London: IEEE, 2022. 1749–1758. [doi: 10.1109/INFOCOM48880.2022.9796721]
[94] Liu Y, Ma Z, Liu XM, Ma JF. Learn to forget: User-level memorization elimination in federated learning. arXiv:2003.10933, 2021.
[95] Gong J, Kang J, Simeone O, Kassab R. Forget-SVGD: Particle-based Bayesian federated unlearning. In: Proc. of the 2022 IEEE Data Science and Learning Workshop. Singapore: IEEE, 2022. 1–6. [doi: 10.1109/DSLW53931.2022.9820602]
[96] Wang JX, Guo S, Xie X, Qi H. Federated unlearning via class-discriminative pruning. In: Proc. of the 2022 ACM Web Conf. Virtual Event: ACM, 2022. 622–632. [doi: 10.1145/3485447.3512222]
[97] Che TS, Zhou Y, Zhang ZJ, Lyu LJ, Liu J, Yan D, Dou DJ, Huan J. Fast federated machine unlearning with nonlinear functional theory. In: Proc. of the 40th Int'l Conf. on Machine Learning. 2023. 4241–4268.
[98] Chen M, Zhang ZK, Wang TH, Backes M, Humbert M, Zhang Y. When machine unlearning jeopardizes privacy. In: Proc. of the 2021 ACM SIGSAC Conf. on Computer and Communications Security. Virtual Event: ACM, 2021. 896–911. [doi: 10.1145/3460120.3484756]

附中文参考文献:

[6] 中华人民共和国国家质量监督检验检疫总局, 中国国家标准化管理委员会. GB/T 35273-2020 信息安全技术 个人信息安全规范. 北京: 中国标准出版社, 2018. https://std.samr.gov.cn/gb/search/gbDetailed?id=A0280129495AEBB4E05397BE0A0AB6FE

李梓童(1999－), 女, 硕士生, 主要研究领域为隐私保护.



王雷霞(1994－), 女, 博士生, 主要研究领域为数据隐私保护.

孟小峰(1964－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为云数据管理, 网络数据管理, 隐私保护.

郝新丽(1995－), 女, 博士生, 主要研究领域为大数据分析.

© 中国科学院软件研究所 http://www.jos.org.cn