

分类号: 按中国图书分类法, 学位办网上可查

单位代码: 10335

密 级: 注明密级与保密期限

学 号: \_\_\_\_\_

# 浙江大学

## 博士学位论文



中文论文题目: 隐私保护机器学习中的  
效率和隐私研究

英文论文题目: Research on the Efficiency and Privacy  
in Privacy-Preserving Machine Learning

申请人姓名: 郑非

指导教师: 郑小林

合作导师: 陈超超

学科(专业): 计算机科学与技术

研究方向: 隐私保护机器学习

所在学院: 计算机科学与技术学院

论文递交日期 2023.3.1



# 隐私保护机器学习中的

## 效率和隐私研究



论文作者签名: \_\_\_\_\_

指导教师签名: \_\_\_\_\_

论文评阅人 1: \_\_\_\_\_ 姓名 \_\_\_\_\_

评阅人 2: \_\_\_\_\_ 姓名 \_\_\_\_\_

评阅人 3: \_\_\_\_\_ 姓名 \_\_\_\_\_

评阅人 4: \_\_\_\_\_ 姓名 \_\_\_\_\_

评阅人 5: \_\_\_\_\_ 姓名 \_\_\_\_\_

答辩委员会主席: \_\_\_\_\_

委员 1: \_\_\_\_\_

委员 2: \_\_\_\_\_

委员 3: \_\_\_\_\_

委员 4: \_\_\_\_\_

委员 5: \_\_\_\_\_

答辩日期: \_\_\_\_\_



**Research on the Efficiency and Privacy  
in Privacy-Preserving Machine Learning**



**Author's signature:** \_\_\_\_\_

**Supervisor's signature:** \_\_\_\_\_

External reviewers: \_\_\_\_\_ Name \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_ Name \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_ Name \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_ Name \_\_\_\_\_  
\_\_\_\_\_

Examining Committee Chairperson:  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Examining Committee Members:  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Date of oral defence: \_\_\_\_\_



# 浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 浙江大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名： 签字日期： 年 月 日

## 学位论文版权使用授权书

本学位论文作者完全了解 浙江大学 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 浙江大学 可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

(保密的学位论文在解密后适用本授权书)

学位论文作者签名： 导师签名：  
签字日期： 年 月 日 签字日期： 年 月 日



致谢



## 摘要

随着机器学习的广泛应用，其隐私保护问题也越来越突出，因此许多**隐私保护机器学习方法**被提出，旨在保护数据隐私和模型隐私的同时，实现机器学习模型的高效推断和训练。当前的隐私保护机器学习方法主要分为基于密码学的方法和拆分学习两类。基于密码学的方法拥有较高的安全性，但是其计算量和通信量较高，运行时间长，导致了实际应用中的效率问题。拆分学习仅需交换模型计算的中间结果，无需复杂的加解密运算，因此效率相对密码学方法较高，但是相比于中心化训练依然存在较高的通信开销。同时，在拆分学习过程中，部分模型以及模型中间结果的暴露也会带来多种隐私泄漏问题。

上述效率和隐私问题对隐私保护机器学习的实际应用带来了重大的挑战。因此，本文围绕着隐私保护机器学习中的效率和隐私问题，进行了全面的分析研究，并且从拆分学习以及密码学和随机排列的混合方法两条路线对隐私保护机器学习中的效率和隐私问题进行优化。本文的主要贡献点包括：

**(1) 针对拆分学习的通信量大问题，提出了基于随机 top- $k$  稀疏化的通信压缩方法。**尽管拆分学习相对于密码学方法效率较高，但是在拆分层表征尺寸大的情况下依然会带来一定的通信开销，尤其是在分类类别数目众多的情况下。而当前少有研究对拆分学习的通信进行优化。本文研究了稀疏、量化、拆分层缩小、L1 正则化等经典的通信压缩手段，从理论上说明了在同等压缩率下稀疏化相比于缩小拆分层能够带来更大的表征空间从而提高模型泛化能力，同时展示了传统的 top- $k$  稀疏化会带来部分神经元难以被训练的收敛性问题。基于以上分析，本文提出了随机 top- $k$  稀疏化方法，在提高训练和推断的通讯效率的同时，尽可能减少了模型性能的损失。

**(2) 针对拆分学习面临的模型补全攻击风险，提出了基于势能损失的隐私保护方法。**拆分学习中，样本的拆分层表征和其类别标签往往有着较高的关联性，从而导致了模型补全攻击的隐患，即：攻击者可以利用少量泄漏的样本标签或者直接对拆分层表征上进行聚类，从而得到表征到标签的映射，窃取顶部模型以及样本标签。为了防御模型补全攻击，本文将攻击过程看作攻击者的有监督或无监督学习过程，从泛化误差的角度说明当样本表征分布在决策边界时，攻击效果会降低。受到电磁学现象的启发，本文提出了

基于势能损失函数的拆分学习隐私保护方法，通过在同类表征之间加入排斥力，使得样本表征分布在决策边界附近，相比于已有方法，显著降低了模型补全攻击的效果，同时实现最少的模型性能损失。

(3) 针对密码学方法非线性函数计算的性能瓶颈，提出了基于密码分享和随机排列的高效隐私保护神经网络框架。

(3) 针对大预言模型隐私推断难题，融合优化秘密分享、随机排列和同态加密，实现秒级别的隐私推断。

## **Abstract**



# 目录

致谢 .....	I
摘要 .....	III
Abstract .....	V
目录 .....	VII
1 绪论 .....	1
1.1 研究背景和意义 .....	1
1.2 研究内容 .....	3
2 相关研究综述 .....	7
2.1 基于拆分学习的隐私保护机器学习 .....	7
2.1.1 拆分学习中的隐私问题 .....	8
2.1.2 拆分学习的隐私保护方法 .....	12
2.1.3 拆分学习的效率提升 .....	13
2.2 基于密码学的隐私保护机器学习 .....	14
2.2.1 基于单一密码学技术的隐私保护机器学习 .....	17
2.2.2 基于混合技术的隐私保护机器学习 .....	17
2.2.3 其他相关研究 .....	19
2.3 密码学与非密码学混合的隐私保护机器学习 .....	19
3 基于随机 Top- $k$ 算法的拆分学习效率优化 .....	21
3.1 研究背景 .....	21
3.2 压缩方法初步研究 .....	22
3.3 随机 top- $k$ 算法 .....	24
3.3.1 Top- $k$ 算法和缩小拆分层算法的分析 .....	24
3.3.2 Top- $k$ 算法训练收敛问题 .....	27
3.3.3 随机 top- $k$ 算法定义 .....	29
3.3.4 隐私分析 .....	30
3.4 实验分析 .....	30
3.4.1 压缩比率和模型准确率对比 .....	31

3.4.2 训练速度分析 .....	32
3.4.3 随机参数 $\alpha$ 分析 .....	33
3.4.4 隐私分析 .....	35
3.5 本章小结 .....	36
4 基于势能损失的隐私保护拆分学习 .....	37
4.1 研究背景 .....	37
4.2 问题描述 .....	38
4.2.1 拆分层的隐私泄露 .....	39
4.2.2 威胁模型 .....	39
4.2.3 问题定义 .....	40
4.3 势能损失函数 .....	41
4.3.1 数据分布导致的学习误差 .....	41
4.3.2 势能损失函数定义 .....	44
4.3.3 与距离相关性的关系 .....	47
4.3.4 输入特征的隐私保护 .....	48
4.4 实验分析 .....	48
4.4.1 微调攻击 .....	50
4.4.2 聚类攻击 .....	51
4.4.3 在拆分层之前攻击 .....	52
4.4.4 拆分层维度的影响 .....	53
4.4.5 拆分层表征分布分析 .....	54
4.4.6 在大型模型上的补充实验 .....	55
4.5 本章小结 .....	56
5 基于秘密分享和随机排列的高效隐私保护机器学习 .....	57
5.1 研究背景 .....	57
5.2 问题描述 .....	58
5.2.1 隐层表征的隐私泄露 .....	58
5.2.2 本文问题定义 .....	60
5.3 方法描述 .....	61

5.3.1 基于秘密分享的线性运算 .....	61
5.3.2 基于随机排列的激活函数计算 .....	65
5.3.3 基于关联随机性的通讯优化 .....	67
5.3.4 隐私保护神经网络框架实现 .....	68
5.4 安全性分析 .....	69
5.4.1 距离相关性 .....	69
5.4.2 随机线性变换的距离相关性 .....	70
5.4.3 随机排列的距离相关性分析 .....	73
5.4.4 数值模拟 .....	74
5.5 实验分析 .....	75
5.5.1 实验设置 .....	76
5.5.2 基准测试 .....	78
5.5.3 隐层维度效果 .....	78
5.5.4 真实数据集实验 .....	78
5.5.5 准确率 .....	79
5.5.6 隐层隐私泄漏情况 .....	80
5.6 本章小结 .....	81
6 基于随机排列的高效大语言模型推理框架 .....	83
6.1 研究背景 .....	83
6.2 初步研究 .....	84
6.2.1 大模型结构 .....	84
6.2.2 针对大模型拆分学习的攻击 .....	85
6.3 PermLLM 框架 .....	89
6.3.1 隐私推断的三个阶段 .....	90
6.3.2 优化的秘密分享乘法 .....	91
6.3.3 基于随机排列的非线性函数计算 .....	94
6.3.4 基于同态加密的下标抽取 .....	94
6.3.5 对 PermLLM 进行优化 .....	97
6.4 安全性分析 .....	98

6.4.1 随机排列 .....	98
6.4.2 浮点数秘密分享 .....	99
6.5 实验分析 .....	100
6.5.1 基准测试 .....	100
6.5.2 ChatGLM-6B 测试.....	102
6.6 本章小结 .....	102
参考文献 .....	105
作者简历 .....	113

# 1 绪论

本章将介绍隐私保护机器学习相关的研究背景、研究现状，梳理本文的研究目的和研究问题，最后总结本文的主要工作和贡献，并呈现本文的组织结构。

## 1.1 研究背景和意义

近年来，以深度神经网络（Deep Neural Network）为代表的人工智能（Artificial Intelligence）技术在多个领域被广泛应用，包括金融领域的智能推荐、风险控制；媒体领域的智能翻译、语音识别和生成；工业或医学领域的图识别；机器人和自动驾驶领域的智能控制；以及近年来新出现的内容生成领域的文本、图像和视频的生成<sup>[1-3]</sup>。人工智能技术作为近年来开始广泛发展的新兴技术，在各个领域显示出了其潜力，因此各国企业和政府都在不断加大对人工智能的投入。我国国务院于 2017 年发布《新一代人工智能发展规划》<sup>[4]</sup>，提出“人工智能是引领未来的战略性技术”。美国于 2016 年提出《国家人工智能研发计划》，并于 2019 年、2023 年进行了两次更新，强调联邦政府必须继续支持人工智能的长期研究<sup>[5]</sup>。根据 2023 年世界人工智能大会，目前我国人工智能产业规模以及达到 5000 亿人民币，企业数量超过 4300 多家<sup>[6]</sup>。据 Statista 数据库的估计，全球人工智能的产业规模在 2023 年达到 1400 亿美元，并且将在 2030 年前后增长到 8000 亿美元。

数据是人工智能应用的关键一环。训练人工智能模型，往往需要采集大量的数据。而使用人工智能模型，也需要提供数据才能产生模型的输出。因此，随着人工智能的广泛应用，数据隐私的问题也越来越突出。从用户角度而言，数据被搜集用于人工智能模型的应用将会暴露用户自身的隐私。比如，电商平台的 APP 为了训练和使用其推荐系统模型，必须采集用户的浏览、点击等行为信息，甚至用户的地址、生活习惯等额外信息；使用类似 ChatGPT 的聊天机器人时，用户的数据也会被发送到模型所在的服务器中。从公司角度而言，如果多个公司想要互相利用对方的数据用于训练更强大的模型，使用一般的人工智能模型的训练方法将数据集中在某个服务器上进行集中式训练，则有数据泄露的风险。因此，各个公司难以将数据进行共享以训练更强大的模型，形成了“数据孤

岛”问题。近年来各国政府也相继出台了有关保护数据隐私的相关法律法规，旨在加强数据的安全使用，防止出现数据滥用和泄露问题。例如：2021年，我国通过了《个人信息保护法》<sup>[7]</sup>，规定信息处理者有“采取相应的加密、去标识化等安全技术措施”的义务。2016年欧盟提出的《一般资料保护规则（GDPR）》法案<sup>[8]</sup>也要求，数据处理时需要考虑使用匿名化、加密等技术保证数据的隐私。

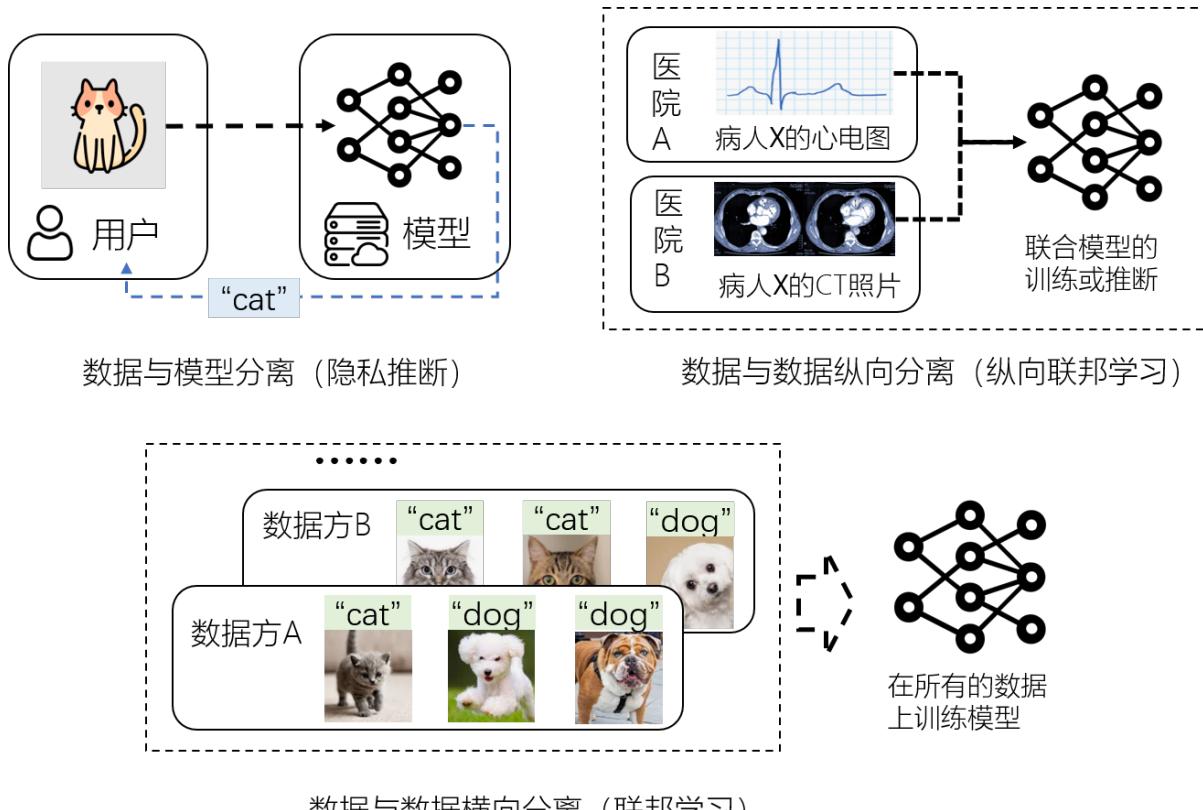


图 1.1 人工智能应用中的三大隐私保护场景

人工智能应用中的隐私问题可以分为3个场景，分别是数据与模型分离的“隐私推断（Private Inference）”场景<sup>[9]</sup>、数据与数据纵向分离的“纵向联邦学习（Vertical Federated Learning）”场景<sup>[10]</sup>、以及数据与数据横向分离的“(横向)联邦学习（Federated Learning）”场景<sup>[11]</sup>。我们将其呈现在图1.1中。其中数据与数据横向分离场景中的数据隐私问题已经有联邦学习这一较为成熟的解决方案<sup>[11]</sup>。联邦学习的基础范式为联邦平均（Federated Learning）算法<sup>[12]</sup>，其通过多个参与方在本地训练模型再在服务器上以参数平均的方式聚合，从而保护各个参与方的数据隐私。虽然联邦学习的实现较为简单高效，但是其应用场景局限在数据横向分布（各方拥有具有相同格式的不同样本）的模型训练阶段，且无法保护模型的隐私，因此具有一定的局限性。而隐私保护机器学习（Privacy-Preserving

Machine Learning) 是解决人工智能应用中的隐私问题的一种通用解决方案<sup>[13-14]</sup>。其通过密码学、拆分学习等多种方法，实现对人工智能模型训练或推断过程中的隐私保护，可以适配多种人工智能模型的训练和推断场景。与联邦学习的场景互补，本文中所讨论的隐私保护机器学习方案往往考虑在单次计算中参与方较少的场景，对应数据与模型分离以及数据与数据纵向分离的场景。对于包含大量参与方的联邦学习场景，可以通过各个参与方将数据加密地上传至少数几个服务器，从而将其转化为隐私保护机器学习问题。

隐私保护机器学习的核心问题在于隐私和效率。基于密码学的隐私保护机器学习，能够达到很高的安全性，并且可以对安全性进行严格的数学证明，但是其需要复杂的多方交互计算以及昂贵的密码学加解密运算。随着人工智能模型的规模日益庞大，许多类似 ChatGPT 的大规模模型被提出<sup>[15-17]</sup>，其对计算能力的要求逐步提升，导致基于密码学方法的效率往往难以达到实用的标准<sup>[18]</sup>。而拆分学习相比密码学方法有着较高的效率，但是其安全性缺少严格的证明，许多研究也提出了针对拆分学习的攻击方法；同时，当前对于其效率优化的研究也较为缺少。上述的隐私和效率问题制约了隐私保护机器学习的应用，因此如何更好地平衡隐私保护机器学习的隐私和效率问题，是当前人工智能隐私领域亟待解决的问题。通过优化隐私保护机器学习的效率和性能，可以使得人工智能的应用更加广泛，并且更好地保护用户的隐私权和人工智能公司的数据及模型资产，推动人工智能产业可持续发展。

## 1.2 研究内容

当前隐私保护机器学习的关键问题是隐私和效率问题。综合已有研究，我们可以对当前的隐私保护机器学习领域总结出如下几条挑战：

- **拆分学习的隐私和效率的进一步优化：**拆分学习是纵向联邦学习的一种范式，其通过将模型切分划分给各个参与方的形式来实现一定程度的隐私保护。在拆分学习模型的推断和训练中，中间结果需要频繁交换，因此也带来了隐私和效率的问题。隐私方面，已有许多研究调查了拆分学习中间结果泄露输入特征和标签信息的问题，但是对应的防护措施却鲜有提出。效率方面，不同于联邦学习中通讯压缩的广泛使用，对于拆分学习的效率研究依然处于起步阶段。因此，如何对拆分

学习的隐私和效率进行优化，是拆分学习走向更广泛应用的一大挑战。

- **更好地混合密码学和非密码学方法来实现效率与隐私的平衡：**虽然基于密码学算法的隐私保护机器学习系统有着可证明的安全性，但是其带来的巨大的计算和通讯量使其在许多人工智能模型上难以实用。当前的将密码学方法和拆分学习等非密码学方法的研究往往对安全性做出了较大的妥协，有一些研究甚至对其安全性提出了严重的质疑。因此，探究如何更好地将密码学方法和非密码学方法结合，同时尽可能保证其安全性影响最小，是让隐私保护机器学习系统走向实用的一大挑战。
- **将隐私保护机器学习应用到更大规模的模型中：**随着人工智能和深度学习的发展，模型的规模日益加大，尤其是 ChatGPT 等大语言模型的兴起，许多模型的参数量已经增长到数十亿甚至百亿、千亿量级。在此如此大规模的模型上采用密码学方法会带来巨大的计算和通讯开销，难以达到实用水平；而由于大规模模型的中间结果也十分庞大，导致基于拆分学习的方法的隐私泄露大大增加。因此，大规模模型的出现和流行给隐私保护机器学习带来了新的挑战。

本文针对上述的三大挑战，围绕着拆分学习以及密码学和其他方法混合的隐私保护机器系统进行研究，提出了一系列创新的解决方案，对隐私保护机器学习的效率和隐私进行了更好的优化和平衡，并且提出了在大语言模型上可以实用的、高安全性的隐私保护机器学习系统。本文的研究内容具体如下：

1. **基于随机 top- $k$  算法的拆分学习的通讯效率优化：**拆分学习要求每一轮训练或推断都传输拆分层的隐层表征，因此也具有一定的通讯开销，在联邦学习中，稀疏和量化等通讯压缩方法已经被广泛使用，但是其在拆分学习中的应用依然鲜有研究。本文考虑标签数量大且标签信息不敏感的拆分学习场景，对缩小拆分层维度、top- $k$  稀疏化、拆分层 L1 正则化、拆分层量化等集中基础的通讯压缩方法进行了研究。通过对收敛性和泛化性的理论分析，最终提出了随机 top- $k$  稀疏化，在保证模型准确率的基础上大幅降低了拆分学习的通讯和部署开销。
2. **基于势能损失的拆分学习隐私保护：**拆分学习过程中暴露的中间结果带来的隐私泄露问题已经被大量研究提出，如何对此进行防护是一个亟待解决的问题。我们

针对拆分学习的拆分层表征于模型预测值相关性高的问题，以及随之而来的模型补全攻击进行理论分析，将攻击问题转化为一个有监督或无监督的机器学习问题。通过对泛化误差和聚类误差进行分析，并受到静电平衡的物理现象启发，我们提出了基于势能损失的拆分层扰动方法，对拆分层的标签信息提供了有力防护。

3. **基于秘密分享和随机排列的隐私保护神经网络：** 基于密码学的隐私保护机器学习框架的开销主要集中在非线性激活函数的计算上，而对于矩阵乘法之类的线性运算则有较成熟的解决方案。而现有的基于密码学和其他方法的混合框架，往往需要暴露中间层结果，在安全性上面临较大的质疑。为此，本文利用神经网络中非线性函数的逐元素（Element-Wise）性质，基于半可信的辅助第三方，提出了基于随机排列的算法来计算非线性函数，并与秘密分享的线性计算融合，实现了高效的三方隐私保护神经网络。本文还对随机排列的安全性进行了细致的理论和实验分析，表明随机排列带来的隐私泄露小到忽略不计。
4. **高效的大语言模型安全推理：** 大语言模型即使是明文也需要极高的计算资源，因此目前的基于密码学的隐私保护机器学习系统应用在大模型上往往会产生更为巨大的通讯和计算开销。本文基于此前研究的基于秘密分享和随机排列的隐私保护神经网络框架，结合同态加密技术，针对大模型推理进行了进一步优化。考虑到推理过程的权重是不变的，本文将秘密分享的乘法计算过程分为三个阶段，并且减少了重复的秘密分享环节。同时，采用了安全的排列协议来计算随机排列，将第三方的所有工作转移到离线阶段进行，进一步提高在线阶段的效率。本文提出的大模型隐私推断框架，实现了在常规网络环境下的数秒内两方大模型在线推理，相比于已有算法产生了数百倍的提升。



## 2 相关研究综述

隐私保护机器学习 (Privacy-Preserving Machine Learning) 指的是在保护数据以及模型隐私的前提下进行机器学习模型的训练和推断。隐私保护机器学习的研究目前分为两大类，分别是拆分学习 (Split Learning) 和基于密码学的隐私计算，拆分学习通过切分模型，以及各方交换中间结果实现隐私的训练和推断，实现简单，额外计算和通讯开销小。但是由于部分模型和中间结果的暴露，存在一定的隐私泄露风险，并不能做到可证明的安全。基于密码学的隐私计算通过各种密码学底层技术实现可证明安全的隐私保护机器学习，但是由于牵涉到大量密码学计算，会带来较高的通讯和计算开销。另外还有一些混合了密码学隐私计算和纵向联邦学习思想的方法，旨在更好地平衡效率和隐私。除此之外，还有一些方法旨在保护机器学习中的隐私，如联邦学习通过多个参与方聚合模型来保护各个参与方的本地数据隐私。但是由于其只适用于数据横向分割、训练阶段的场景，与本文讨论的通用的隐私保护机器学习的训练和推断有所区别，因此不再进行赘述。本章节对以上的三类方法逐一进行介绍。

### 2.1 基于拆分学习的隐私保护机器学习

拆分学习<sup>[19-20]</sup> (Split Learning) 是纵向联邦学习 (Vertical Federated Learning) 的主流范式，指的是在纵向联邦学习中，把模型拆分成多个部分划分给不同的参与方，参与方之间相互交换中间结果和梯度，从而实现模型的正向传播和反向传播。在此过程中，各方的输入数据并未直接泄露，因此拆分学习被认为在一定程度上可以保护隐私。

一个简单的拆分学习例子如图 2.1 所示。此时纵向联邦学习有 3 个参与方（记作  $P_1, P_2, P_3$ ），分别拥有样本特征第一部分 ( $\mathbf{x}_1$ )、样本特征第二部分 ( $\mathbf{x}_2$ ) 和样本标签 ( $\hat{y}$ )。相应地，模型也被划分成了 3 个部分，分别是底部模型第一部分 ( $M_{b1}$ , 参数记为  $\Theta_{b1}$ )，底部模型第二部分 ( $M_{b2}$ , 参数记为  $\Theta_{b2}$ )，以及顶部模型 ( $M_t$ , 参数记为  $\Theta_h$ )。在前向传播的过程中， $P_1$  和  $P_2$  分别本地计算得到各自的中间结果： $(\mathbf{h}_1 = M_{b1}(\mathbf{x}_1; \Theta_{b1}), \mathbf{h}_2 = M_{b2}(\mathbf{x}_2; \Theta_{b2}))$ ，然后将其发送给  $P_3$ 。标签拥有方接收到  $\mathbf{h}_1, \mathbf{h}_2$  之后，将其输入到顶部模型，计算得到模型输出  $y = M_t(\mathbf{h}_1, \mathbf{h}_2; \Theta_h)$ 。这样便完成了一次前向传播过程。在反向传播过程中， $P_3$

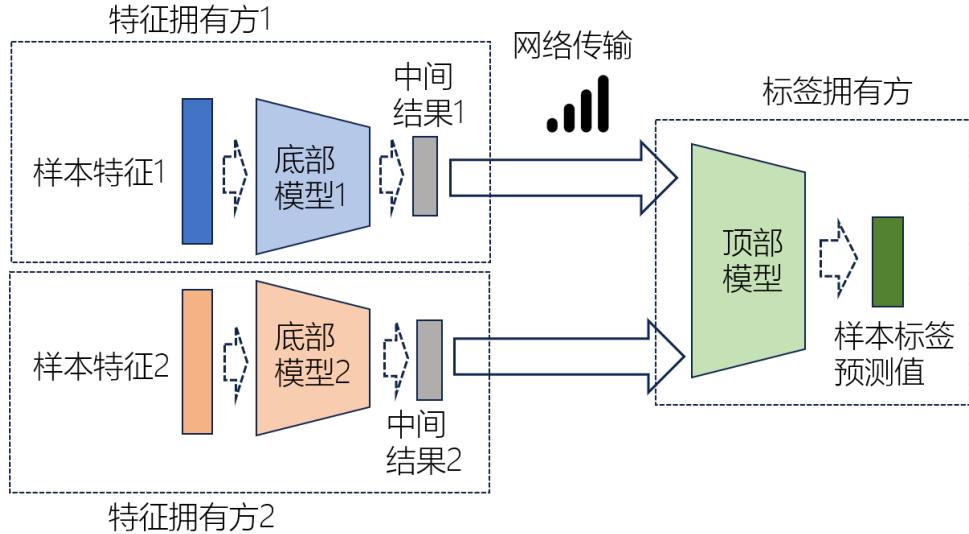


图 2.1 拆分学习示意图

通过模型的预测标签 ( $\hat{y}$ ) 和真实标签 ( $y$ )，计算出损失函数  $L(\hat{y}, y)$ ，并且将其对于顶部模型和中间结果的参数： $\partial L(\hat{y}, y)/\partial \Theta_t$ ,  $\partial L(\hat{y}, y)/\partial \Theta_{b1}$  和  $\partial L(\hat{y}, y)/\partial \Theta_{b2}$ . 其中，前者被标签拥有方用于更新顶部模型参数，而后两者则发给相应的特征拥有方用于计算损失函数相对于底部模型参数的梯度。该计算通过链式求导法则进行，以  $P_1$  为例：

$$\frac{\partial L(\hat{y}, y)}{\partial \Theta_t} = \frac{\partial L(\hat{y}, y)}{\partial \mathbf{h}_1} \cdot \frac{\partial \mathbf{h}_1}{\partial \Theta_{b1}}. \quad (2-1)$$

注意，上式中的  $\partial \mathbf{h}_1 / \partial \Theta_{b1}$  是雅可比矩阵，链式法则中的乘法也是对应的矩阵乘法运算。

拆分学习具有原理简单、实现方便、计算和通信开销小、可以适用于不同深度学习的模型的优势，因此其已经被应用于很多需要考虑数据隐私的领域，如：医疗影像分割<sup>[21]</sup>、边缘设备中的图像分类<sup>[22-23]</sup>、毫米波接收功率预测<sup>[24]</sup>等。

### 2.1.1 拆分学习中的隐私问题

虽然拆分学习拥有实现简单、效率高等诸多优势，但是在训练和推断过程中，各方直接交换了中间结果和中间梯度的明文，从而存在一定的隐私泄漏风险。拆分学习的隐私泄露可以分为两类：特征拥有方的隐私泄露和标签拥有方的隐私泄露。下面将对两种隐私泄露进行介绍。

### 2.1.1.1 特征拥有方的隐私泄露

在拆分学习的过程中，特征拥有方将底部模型产生的中间结果，也就是神经网络的隐层表征（Hidden Representation），直接发送给了标签拥有方。作为输入特征经过变换的结果，隐层表征必然包含了输入特征的信息。假设攻击者腐化了标签拥有方，便可以通过各种手段从隐层表征中恢复出输入特征的信息。因此，近年来也有许多工作研究了拆分学习中特征拥有方的隐私泄露问题，现总结如下：

1. 在卷积神经网络中，隐层表征直接包含了输入的信息<sup>[25]</sup>。由于卷积层的局域运算特性，其会保留图像本身的大致形状、轮廓等信息。即便经过多个卷积层后，隐层表征依然能够保留图像的整体轮廓，从而使得原始图像的数据出现一定程度的泄露。
2. 攻击者可能训练一个网络从隐层表征中重构出原始输入特征<sup>[26-27]</sup>。一种简单的情况可以是，攻击者能够获取一部分泄漏的样本特征  $(\mathbf{x}_1, \dots, \mathbf{x}_n)$  和对应的隐层表征  $(\mathbf{h}_1, \dots, \mathbf{h}_n)$ ，则重构函数  $R$  可以定义为：

$$\operatorname{argmin}_R \sum_{i=1}^n \|R(\mathbf{h}_i) - \mathbf{x}_i\|^2 = \operatorname{argmin}_R \sum_{i=1}^n \|R(M_b(\mathbf{x}_i)) - \mathbf{x}_i\|^2, \quad (2-2)$$

其中， $\|\cdot\|^2$  表示二范数平方，也可以按照具体情况换成其他的函数来度量重构效果。一般可以采用一个和底部模型相对应的神经网络作为重构函数，比如原始模型是多层卷积，则  $R$  可以主要由反卷积（ConvTranspose）构成。

3. 如果攻击者本身拥有底部模型的参数信息，则可以采用白盒攻击的形式，优化  $\mathbf{x}'$  使得  $M_b(\mathbf{x}')$  尽可能接近  $\mathbf{h} = M_b(\mathbf{x})$ :<sup>[27-28]</sup>

$$\operatorname{argmin}_{\mathbf{x}'} \|M_b(\mathbf{x}') - \mathbf{h}\|^2 = \operatorname{argmin}_{\mathbf{x}'} \|M_b(\mathbf{x}') - M_b(\mathbf{x})\|^2. \quad (2-3)$$

注意到式 2-3 可能有无穷多个解，影响重构效果。为了提高重构效果，也可以在损失函数中加入关于输入特征的先验知识。比如输入特征是图像，则可以在损失函数中加入  $\mathbf{x}'$  的总变差（Total Variance），使得重构出来的结果更接近实际图像；如果攻击者有同一样本的额外特征  $\mathbf{z}$ ，则可以将  $\mathbf{x}'$  替换为  $g(\mathbf{z})$ ，这里假设想要攻击的输入特征和攻击者获取的额外特征有关。

4. 在训练过程中，攻击者可以对训练目标进行修改，使得中间结果包含更多关于输入特征的信息。比如假设攻击者拥有一部分泄漏的样本特征  $\{\mathbf{x}'\}$ ，则他可以先训练一个自编码器<sup>[29-30]</sup> ( $f, f^{-1}$ ) 使得  $f^{-1} \circ f(\mathbf{x}') \approx \mathbf{x}'$ 。在拆分学习训练过程中，攻击者采用对抗训练的方式，使得待攻击样本的隐层表征  $\mathbf{h} = M_b(\mathbf{x})$  与自编码器编码的泄漏样本的特征  $f(\mathbf{x}')$  尽可能接近。于是攻击者可以直接通过  $f^{-1}(\mathbf{h})$  来重构原始的输入特征<sup>[31]</sup>。

值得注意的是，第 4 种方法需要攻击者改变正常的拆分学习训练过程，而前面介绍的 3 种方法则不需要改变训练过程。一般将主动改变训练过程的攻击者称为“主动攻击者（Active Attacker）”，反之则称为“被动攻击者（Passive Attacker）”。主动攻击者往往能取得更强大的攻击效果，但是由于其对训练本身的改变，可能导致训练效果变差、训练速度变慢等问题，会更容易被检测出来，且无法在模型推断阶段进行攻击。与此相反，被动攻击者的攻击效果会更低，但是具有更好的隐蔽性。

### 2.1.1.2 标签拥有方的隐私泄露

对于标签拥有方的隐私泄露，我们考虑的是攻击者腐化特征拥有方的情况。标签拥有方的隐私泄露主要来源于两个方面：

1. 隐层表征（中间结果）导致的标签信息泄露。深度学习模型学习的过程，可以看做一个将输入特征逐渐转化成标签的过程。因此，随着训练的进行，隐层表征会逐渐变得与标签更为相关。许多将神经网络隐层表征可视化的工作也显示出，随着训练的进行，隐层表征逐渐按照对应的标签聚类成不同的簇；且越靠近输出的隐层表征和标签关联度越大<sup>[32-34]</sup>。在这种情况下，攻击者就可以根据隐层表征来输入数据的标签。即使在没有任何额外信息的情况下，攻击者也可以对隐层表征进行聚类，从而获取输入样本的类别关系<sup>[35-36]</sup>。如果攻击者能够获取少量泄漏的隐层表征和对应的标签，也可以自己训练一个顶部模型，并且获得很好的分类效果<sup>[37]</sup>。
2. 隐层表征的梯度导致的标签信息泄露。与隐层表征类似，隐层的梯度和标签信息也呈现出非常强的相关性。比如在二分类模型中，同类样本的梯度间的余弦相似度接近 1，而异类样本的梯度间的余弦相似度则接近 -1，直接暴露了样本的类别信

息<sup>[38]</sup>。此外，也可以通过优化替代样本的梯度，使其尽可能接近攻击者获取到的梯度，从而复原出样本的标签<sup>[39]</sup>。具体方法如下：

$$\operatorname{argmin}_{\mathbf{x}', \hat{y}'} \left\| \frac{\partial L(\hat{y}', M_h(M_b(\mathbf{x}); \Theta'_h))}{\partial M_b(\mathbf{x})} - \frac{\partial L(\hat{y}, M_h(M_b(\mathbf{x}); \Theta_h))}{\partial M_b(\mathbf{x})} \right\|^2, \quad (2-4)$$

其中， $\mathbf{x}, \hat{y}$  是原样本的特征和标签，而  $\mathbf{x}', \hat{y}'$  是随机初始化的替代样本的特征和标签； $\Theta_h, \Theta'_h$  分别表示原始的顶部模型参数和替代的顶部模型参数。通过优化替代的样本和顶部模型参数，可以恢复出样本的标签。

标签泄露不仅仅带来了样本标签的隐私问题，同时还会泄露模型本身。如上文所述，当特征拥有方腐化（Corrupted）后，它可以利用少量标签（或直接聚类生成标签），训练出顶部模型。由于在模型训练的后期，底部模型的表征提取能力已经很强，固定底部模型后，攻击者很容易就可以训练出一个表现良好的顶部模型，从而得到整个完整的模型。因此，这种攻击也被称为“模型完整攻击（Model Completion Attack）”<sup>[37]</sup>。在这种情况下，腐化的特征拥有方可以窃取几乎标签拥有方的所有财产（标签和顶部模型），从而使得拆分学习的意义几乎不复存在。

### 2.1.1.3 总结

无论是输入特征的泄露还是标签的泄露，都与许多因素有关，包括模型结构和拆分的层数。显而易见的是，如果分割点靠近模型输入，则隐层表征泄漏的特征信息更多；而分割点接近模型输出，则会泄露更多关于标签的信息。这也是著名的数据处理不等式（Data Processing Inequality）的推论。模型的结构也对泄漏的信息有很大的影响。从输入特征的角度而言，对于卷积神经网络，卷积层的隐层表征与输入十分相关，且很多网络的隐层维度也很高，因此很容易被攻击者重构出输入特征<sup>[25]</sup>；对于全连接网络，如果隐层维度小于数据维度，则会存在无穷多组输入拥有同样的隐层表征，这导致没有先验知识的情况下，攻击者难以获取输入特征信息；基于 Transformer 结构的语言模型，包括时下流行的大语言模型（Large Language Model），拥有很高的隐层表征维度，并且保留了序列结构，因此攻击者也可以轻易地从隐层表征中重构出原始的输入文本<sup>[40]</sup>。从标签的角度而言，更低的隐层表征维度往往意味着更容易恢复出标签，因为高维度的特征中可能包含许多与标签无关的信息，使得攻击者更难从中提取出标签<sup>[38,41]</sup>。

### 2.1.2 拆分学习的隐私保护方法

为了解决前文所述的拆分学习中的隐私泄露问题，一些研究也提出了保护拆分学习中输入特征和标签的隐私的方法，当前主要采用的方法是对隐层表征或隐层梯度进行扰动。

首先介绍对隐层表征进行扰动，使其与输入特征或标签特征不相关的方法。由于隐层表征、输入特征、标签的维度可以是任意的，因此常用的皮尔逊相关性（Pearson Correlation）并不适用。因此现有工作主要采用距离相关性来衡量隐层表征和输入特征、标签的相关性<sup>[26,41]</sup>。距离相关性（Distance Correlation）<sup>[42-43]</sup>是一种特殊的相关性度量，可以度量不同维度的随机向量之间的相关性，并且可以度量非线性的相关性，距离相关性为 0 当且仅当两个变量是无关的（互信息为 0）。使用距离相关性来保护输入特征或标签信息的损失函数可以写为：

$$L' = L_0 + \alpha \text{Dcor}(H, X) + \beta \text{Dcor}(H, Y), \quad (2-5)$$

其中， $\text{Dcor}(\cdot, \cdot)$  表示距离相关性， $X, H, Y$  分别表示当前批次的输入特征、隐层表征和标签， $\alpha, \beta$  则用于控制扰动程度。注意到，此处的距离相关性计算是通过批样本的经验分布（Empirical Distribution）进行估计的，因此需要较大的批大小，否则估计值会出现较大的误差。

针对隐层梯度泄露标签信息的情况，Marvell 方法<sup>[38]</sup>通过最小化正样本和负样本之间的隐层梯度间的 KL 散度来保护样本的标签信息。当标签拥有方计算出隐层梯度  $g = \partial L / \partial h$  时，它会对其加入一个正态分布的噪声  $D \sim \mathcal{N}(0, \Sigma_D)$ ，然后将其发给特征拥有方。具体的优化问题可以写做：

$$\begin{aligned} & \min_{\Sigma_D^+, \Sigma_D^-} \text{KL} [\mathcal{N}(\mu_g^+, \Sigma_g^+ + \Sigma_D^+) \| \mathcal{N}(\mu_g^-, \Sigma_g^- + \Sigma_D^-)], \\ & \text{s.t. } \text{ptr}(\Sigma_D^+) + (1-p)\text{tr}(\Sigma_D^-) \leq P, \end{aligned} \quad (2-6)$$

这里假设正样本和负样本的原始的隐层梯度也分别满足正态分布  $\mathcal{N}(\mu_g^+, \Sigma_g^+)$  和  $\mathcal{N}(\mu_g^-, \Sigma_g^-)$ ，而  $\Sigma_D^+, \Sigma_D^-$  则分别是给正样本和负样本的隐层梯度家的噪声的协方差矩阵。约束条件中的  $p$  和  $1-p$  分别代表正样本和负样本的频率， $P$  表示设定的噪声大小的平均值上限，用于控制噪声的规模，防止添加的噪声太大。

上述的基于对隐层表征和梯度的扰动方法都存在一些缺陷。在隐层表征维度高的情况下，距离相关性的估计会变差，从而使得其保护效果大幅度降低<sup>[39]</sup>。而 Marvell 方法<sup>[41]</sup>虽然对梯度进行了扰动，并不能防止从隐层表征直接推出样本标签的情形，同时该方法也仅适用于二分类场景。因此，如何保护拆分学习过程中的特征和标签隐私，依然是个极具挑战性的问题。

### 2.1.3 拆分学习的效率提升

相对于基于密码学方法的纵向联邦学习，拆分学习具有较小的开销，因为拆分学习仅仅是在明文计算的基础上，传输隐层表征和隐层梯度，并未引入额外的计算。但是考虑到许多神经网络的隐层尺寸较大，因此在通讯效率方面，拆分学习依然存在可以优化的空间。一种简单的思路是将稀疏（Sparsification）和量化（Quantization）方法引入拆分学习，对隐层表征和隐层梯度进行压缩，从而减少拆分学习训练和推断过程中的通讯量。

**稀疏化：**对一个向量（或矩阵、张量）进行稀疏化，指的是将其（绝对值）较大的元素保留，而将其绝对值较小的元素丢弃，即设置为 0。稀疏化有效性基于“较大的元素在计算过程中较为重要，而较小的元素在计算过程中可以忽略”这一假设。稀疏化之后，数据的传输格式也会发生变化。一种简单做法是将数据以“(下标，值)”的形式进行存储。假设原有的数据位数为  $L$ （如对于 32 位浮点数  $L = 32$ ），数据的总量为  $N$ ，稀疏率为  $p$ ，则简单的下标-值压缩方法可以实现压缩比率（压缩后大小/压缩前大小）为：

$$\text{压缩比率} = \frac{pN(L + \lceil \log_2 N \rceil)}{NL} = p\left(1 + \frac{\lceil \log_2 N \rceil}{L}\right). \quad (2-7)$$

为了提高压缩比率，可以采用更加先进的数据压缩方法，如经典的 Huffman 编码<sup>[44]</sup>可以直接作用于稀疏化后的数据（的二进制表示）上。更为适合稀疏化的编码为对下标的差值（Run-length）序列进行 Golomb 编码<sup>[45]</sup>。如果在一批数据中，每个下标被稀疏化的概率是均等的，则 Golomb 编码具有最小的期望压缩比率，大约可以在下标-值压缩的基础上再减少一半<sup>[46]</sup>。值得注意的是，采用编码方案虽然减少了通讯开销，但是也会带来一定的计算开销。

**量化：**量化指的是降低数据的精确度，使用较少的比特位保留数据，在尽可能保证数据的准确性的情况下，压缩其存储空间。一般的机器学习模型的训练或推断采用的都

是 32 位浮点数，通过量化的方式，可以将模型的权重、模型计算的中间结果等压缩到更低位数（如 8 位，4 位，甚至 1 位），从而减少模型大小或降低模型推断过程中的内存开销和计算开销<sup>[47-49]</sup>。

稀疏化和量化用于减轻通讯量的方法，在分布式计算、横向联邦学习领域被广泛应用<sup>[46,50]</sup>。此时，稀疏化和量化也是被应用在梯度上，从而减少了每轮训练过程中的通讯量开销。虽然稀疏化和量化导致梯度不准确，由于使用随机梯度下降（Stochastic Gradient Descent）优化时，批样本梯度本身有一定噪声，甚至被认为对收敛（Convergence）和泛化（Generalization）是有益的<sup>[51-53]</sup>。同时，相关研究也表明，稀疏化和量化并未对模型最终的结果有显著影响，而训练到一个较高准确度的通讯量也显著小于常规训练的通讯量<sup>[46,50,54]</sup>。

但是拆分学习作为一个较新的领域，其通讯效率提升的研究相对较少。Castiglia 等人<sup>[55]</sup>研究了基本的稀疏、（标量）量化以及向量量化（Vector Quantization）在拆分学习中的应用，并且证明了其收敛性。具体而言，拆分学习的前向传播过程被更改为

$$Y = M_t(\text{Compress}[M_b(X)]) \quad (2-8)$$

其中， $M_t$  表示顶部模型， $M_b$  表示底部模型， $X, Y$  表示输入特征和模型预测值， $\text{Compress}$  表示压缩操作（如稀疏、量化）。尽管收敛性得到证明，但是该文采用的压缩方法最多只能将压缩率降低到 1/16，且常规的稀疏和量化（除了向量量化外）对模型最终的效果有较明显的降低。此外，该方法只适用于训练场景，并且要求公开的标签信息和顶部模型让各个参与方执行块坐标下降法（Block Coordinate Descent）对自身的参数进行优化。

也有部分研究研究了对拆分学习采用异步更新的方式提高通讯效率，其方法包括底部模型多轮本地更新<sup>[56]</sup>或是顶部模型多轮本地更新<sup>[57]</sup>。这些方法同样针对的是拆分学习训练的场景，并不能应用于推断阶段。此外，Ayad 等提出使用自编码器压缩中间表征<sup>[58]</sup>，也就是将(2-8)中的  $\text{Compress}$  替换为一个自编码器，从而实现训练和推断过程中的通讯优化。但是该方法需要针对特定模型定制自编码器，并非一种通用的方法。

## 2.2 基于密码学的隐私保护机器学习

纵向联邦学习可以看作是特定的一类隐私计算问题，即：在保护数据和模型参数的情况下进行模型的训练和推断，因此也可以通过基于密码学的安全多方计算（Secure

Multiparty Computation) 来实现。

**定义 2.2.1 (安全多方计算)** 有  $n$  个参与方  $P_1, \dots, P_n$  与各自的输入  $X_1, \dots, X_n$ , 以及一个公共函数  $f$ , 安全多方计算指的是各方根据指定的协议交互计算出  $Y = f(X_1, \dots, X_n)$  ( $Y$  可以是公开的或是被指定的参与方获得, 两种定义是等价的), 同时保护各方输入的隐私。

注意安全多方计算的定义隐含了安全性设定 (Security Setting), 也就是对于各个参与方行为的限制条件, 包括: 各个参与方忠实执行计算协议但是同时利用自己获取到的一切信息, 称为半可信 (Semi-honest) 安全性设定, 以及参与方可能存在不遵守计算协议的情况, 称为恶意 (Malicious) 安全性设定, 以及在上述两种情况下部分参与方共谋 (Collusion) 的情况。复杂的安全性设定可能导致极为复杂的协议设计和安全性证明。本文遵循一般隐私保护机器学习中的半可信安全性设定。

安全多方计算协议往往基于几种特定的密码学原语来实现, 包括秘密分享 (Secret Sharing)、同态加密 (Homomorphic Encryption)、混淆电路 (Garbled Circuits) 等。下面对此进行简要介绍。

**秘密分享:** 秘密分享<sup>[59]</sup>指的是把一个值分享给多个参与方, 其中一定数量的参与方合作才能恢复出该值。具体而言,  $(t, n)$ -秘密分享指的是将一个值  $x$  分享给  $n$  方  $P_0, \dots, P_{n-1}$ , 其中  $P_i$  得到的值记作  $\langle x \rangle_i$ 。至少要  $t$  个参与方一起, 才能恢复出  $x$  的值。低于  $t$  个参与方则无法得到任何关于  $x$  的信息。常用的秘密分享包括两方加法分享 (Additive Sharing) 以及两方布尔分享 (Boolean Sharing), 这两种分享分别把一个数  $x$  拆分成两个随机数相加 ( $x = \langle x \rangle_0 + \langle x \rangle_1$ ) 或是两个随机数 (按位) 异或 ( $x = \langle x \rangle_0 \oplus \langle x \rangle_1$ )。在秘密分享的情况下, 进行加法 (异或) 只需各方本地运算, 但是进行乘法 (异与) 较为复杂, 一般可以采用离线 (Offline) 计算的 Beaver 三元组进行<sup>[60]</sup>, 即: 在获得具体的输入前, 两方通过一定方法计算出一组秘密分享的随机数  $(u, v, w)$  满足  $uv = w$ 。一般可以使用同态加密、不经意传输 (Oblivious Transfer) 等手段, 在三方场景下, 可以使用一个半可信第三方分发从而提高离线计算的效率。注意秘密分享一般在有限域上进行, 如: 加法分享在模  $N$  的整数环  $\mathbb{Z}_N$  进行, 布尔分享按照定义在  $\mathbb{Z}_2$  进行)。此时各方所获取到的值都是在有限域上均匀分布的随机数, 与实际值无关, 因此实现信息论安全 (Information-Theoretic Security)。

**同态加密：**同态加密是一类特殊的加密算法，其支持在密文上进行一定的计算，使得解密的结果和明文计算一致。令某种加密系统的加密函数为  $\text{Enc}$ ，解密函数为  $\text{Dec}$ 。如果对于某个定义在明文上的运算符  $\text{OP}_P$ ，存在密文上的运算符  $\text{OP}_C$ ，使得对于任意明文  $X, Y$  都满足：

$$\text{OP}_P(X, Y) = \text{Dec}[\text{OP}_C(\text{Enc}[X], \text{Enc}[Y])] \quad (2-9)$$

则称该加密系统可以同态地计算  $\text{OP}_P$  的运算。注意  $\text{OP}_P$  往往不等于  $\text{OP}_C$ 。比如，Paillier 同态加密支持明文加法，即： $\text{OP}_P = '+'$ ，但是对应的密文运算是乘法，也就是  $\text{OP}_C = ' \times '$ 。同态加密按照支持的运算的不同，一般可以分为半同态加密（Partial Homomorphic Encryption）和全同态加密（Fully Homomorphic Encryption）。半同态加密支持一种同态运算，如加法或乘法，分别可以称为加同态加密和乘同态加密。全同态加密同时支持加法或乘法。除此之外，还有分级（Leveled）全同态加密，其只支持一定次数多乘法运算。当前常用的同态加密包括 Paillier 加同态加密<sup>[61]</sup>，其定义域为大整数；BFV/BGV 全同态加密<sup>[62-64]</sup>，其定义域为整系数多项式；以及 CKKS 全同态加密<sup>[65]</sup>，其定义域是实系数多项式，并且运算过程会产生一定的误差。

**混淆电路：**混淆电路由姚期智院士于 1986 年提出<sup>[66]</sup>，是一个多方安全计算协议，可以用于计算任意布尔电路。混淆电路通过对原始电路的逻辑门进行“混淆”得到，具体而言，每个门的输入和输出都变成了随机数，因此无法根据混淆后的电路的输入、输出以及中间值，获取关于原始电路输入输出值的任何信息。混淆电路的执行过程可以表示为：给定一个公开函数  $f$ ，某方（这里设为  $P_0$ ）产生一个布尔电路  $C$ ，然后对其进行混淆得到混淆电路  $C^G$ ，并发送给另一方（这里设为  $P_1$ ）；同时， $P_0$  自身的输入  $X_0$  混淆后得到  $X_0^G$  同时， $P_1$  和  $P_0$  执行不经意传输协议，获取  $P_1$  输入的混淆值  $X_1^G$ 。然后  $P_1$  可以根据混淆输入计算出混淆输出： $Y_G = C_G(X_0^G, X_1^G)$ ，将其返还给  $P_0$  后， $P_0$  根据自身维护的混淆表得到实际的输出值。混淆电路只需要常数轮的通信即可计算任何门电路的值，但是因为电路中的每个布尔值都转化成了高位的随机数，会消耗大量通讯流量；由于涉及密码学计算，也有较大的计算开销。混淆电路发展至今有多种优化方案提出，包括 Free-XOR<sup>[67]</sup>，HalfGate<sup>[68]</sup>等，降低了通讯和计算开销。

除去以上介绍的几种密码学技术，多方安全计算还可以包含其他技术，以及一些定制化的计算协议。下文将对部分主要的基于密码学的隐私保护机器学习研究进行介绍。

### 2.2.1 基于单一密码学技术的隐私保护机器学习

本节对采用单一密码学技术来实现隐私保护机器学习的相关工作进行介绍。这些工作基于两方的隐私保护机器学习场景，如 MLaaS（Machine Learning as a Service），即：用户提供数据，服务方提供机器学习模型。

**基于同态加密：**2016 年微软提出 CryptoNets<sup>[69]</sup>，率先通过同态加密实现神经网络的推断。由于神经网络的权重和输入都被加密，因此这种方法可以实现安全的神经网络推断。CryptoNets 采用了 YASHE 加密算法<sup>[70]</sup>，这是一种分级同态加密算法，可以支持一定次数的乘法运算、由于同态加密仅仅支持加法和乘法运算，因此 CryptoNets 将神经网络中的非线性激活函数替换成平方函数，然后再明文上训练得到模型权重。该方法需要花费数分钟和 300Mb 左右的流量对 MNIST 数据集中的一张图片进行分类。一些未正式发表的工作<sup>[71-72]</sup>对 CryptoNets 进行了改进，包括使用更加精确的多项式拟合激活函数、将同态加密算法换为 BGV 或 CKKS 等更高效的算法，以及优化同态加密的密文打包技术等。Zhou 等人通过将卷积神经网络的权重二值化，实现了更高效的基于全同态加密的神经网络推理<sup>[73]</sup>。

**基于混淆电路：**DeepSecure 框架<sup>[74]</sup>将神经网络模型转化为布尔电路，从而实现隐私神经网络推断。为了提高效率，该框架减少了各类电路中的非异或门数目，从而适配 Free-XOR 算法。该框架运算速度高于 CryptoNets，但是带来了更高的通讯开销。实验表明，一个简单的全连接网络的推断消耗 800MB 左右的流量。XONN 框架<sup>[75]</sup>将二值神经网络（Binary Neural Network）与混淆电路结合，将内积运算转变为异或运算，从而极大地提高了安全推断的效率。

### 2.2.2 基于混合技术的隐私保护机器学习

为了提高隐私保护机器学习的效率，现有的隐私保护机器学习框架大多将多种密码学技术混合，为机器学习中不同的算子找到最佳的密码学技术。

**两方框架：**ABY 框架<sup>[76]</sup>提出将算数秘密分享、布尔秘密分享以及 Yao 分享（基于混淆电路<sup>[66]</sup>的两方分享，一方持有混淆值和真实值的对照表，另一方持有混淆电路和混淆值）融合，并且涉及协议对三种分享状态的值进行相互转化，从而为特定的计算任务找到最适合的混合计算协议。SecureML 框架<sup>[77]</sup>将算数秘密分享和混淆电路结合，实

现了逻辑回归的推断和训练。其还使用分级同态加密和不经意传输对算数秘密分享的 Beaver 三元组<sup>[60]</sup>的离线生成进行了优化，并且将 Sigmoid 函数表示为分段线性函数，通过混淆电路（计算当前值落在哪一段）和算数秘密分享融合来提高计算效率。MiniONN 框架<sup>[78]</sup>将秘密分享和混淆电路进一步应用于神经网络推断中，并且使用分段的 Spline 插值来拟合激活函数，以及采用 SIMD (Single Instruction Multiple Data) 技术加速了 Beaver 三元组的生成，从而提高了隐私神经网络的模型准确率和效率。Gazelle 框架<sup>[79]</sup>使用优化的基于格密码的同态加密 (BFV 加密<sup>[62-63]</sup>) 直接加速安全矩阵乘法计算，而非用于产生三元组，从而极大提高了通讯效率；同时采用混淆电路来计算 ReLU、最大池化 (Max Pooling) 等非线性运算。实验结果表明 Gazelle 的性能甚至大幅度优于需要依赖第三方的 Chameleon 框架。Delphi 框架<sup>[80]</sup>基于 Gazelle 框架进行了改进，将同态加密的安全乘法计算用于离线计算 Beaver 三元组，并利用了神经网络推断过程中权重不变的特性减少了在线的开销；同时将 ReLU 激活函数替换成二次函数以降低开销。CryptFlow2 框架<sup>[81]</sup>基于秘密分享，采用同态加密和不经意传输实现安全乘法，并且新设计了高效的基于不经意传输的安全比较和除法协议，将两方的隐私保护机器学习推广到 ResNet 等大型模型中。Cheetah 框架<sup>[82]</sup>进一步对基于同态加密的神经网络线性运算以及基于不经意传输的 ReLU 激活函数等非线性运算进行优化。SIRNN 框架<sup>[83]</sup>通过查真值表的方式计算指数函数，进一步对神经网络中 Softmax 等非线性函数进行了优化。

**三方框架：**Chameleon 框架<sup>[84]</sup>混合秘密分享、GMW 协议和混淆电路协议进行神经网络计算，在离线阶段使用了可信硬件作为第三方，并采用相关随机性 (Correlated Randomness) 在各方生成相同的随机数，提高了 Beaver 三元组产生的效率。ABY3 框架<sup>[85]</sup>基于 ABY 框架<sup>[76]</sup>，将秘密分享、布尔分享和混淆电路推广到三方计算的场景以提高效率，并对乘法截断进行了改进。SecureNN 框架<sup>[86]</sup>采用了两方秘密分享并且使用第三方来产生 Beaver 三元组，同时基于比较的布尔运算电路设计了高效的三方安全比较协议，实现了高效的隐私神经网络推断。CryptFlow<sup>[87]</sup>是一个支持两方和三方协议的开源框架，其两方协议采用的是 ABY 协议，而三方协议基于 SecureNN 框架，对其卷积操作的 Beaver 三元组进行了优化，同时使用相关随机性来进一步降低通信量；该框架还提供了基于可信执行环境 (Trusted Execution Environment, TEE) 的恶意安全转换机制，可以将半可信安全的协议转换为恶意安全的协议。Crypten 框架<sup>[88]</sup>基于秘密分享和 GMW 协议实现安全的深度学习，并且提供了类似 Pytorch<sup>2019\_pytorch</sup>风格的接口，使得非密码

学专业研究者也可以进行隐私保护机器学习。Falcon 框架<sup>[89]</sup>基于 ABY3 和 SecureNN 进行了改进，优化了比较协议。AriaNN 框架<sup>[90]</sup>提出使用函数秘密分享（Function Secret Sharing）来计算比较函数，将其降低到仅需一轮通信轮次。

### 2.2.3 其他相关研究

除了设计安全多方计算框架之外，也有一些其他关于提高基于密码学的隐私保护机器学习效率的研究。比如，Dalskov 等<sup>[91]</sup>研究了将量化（Quantization）技术应用于隐私保护机器学习，将神经网络量化到 8 比特，并在多个现有的隐私保护机器学习框架上进行了测试。此外，也有许多对于非神经网络的专用机器学习模型进行隐私保护的研究，如树模型<sup>[92-94]</sup>、聚类模型<sup>[95-96]</sup>、矩阵分解<sup>[97-98]</sup>等。这些方法往往基于已有的同态加密、混淆电路、多方安全计算协议等技术，对于特定的机器学习模型进行设计和优化。

2023 年起，随着 ChatGPT<sup>[15]</sup>的出现，大语言模型在工业界和学术界产生了巨大的影响。而大模型的参数量和计算量都极为巨大，即使是明文推断的情况下也对硬件有较高要求，因此对基于密码学的隐私保护机器学习提出了新的挑战。Iron Transformer<sup>[99]</sup>，MPCFormer<sup>[100]</sup>，SecureTLM<sup>[101]</sup>，PrivFormer<sup>[102]</sup>等框架在之前研究的基础上，采用同态加密、秘密分享、多项式拟合 ReLU、使用特定结构的模型等方法，实现了 Transformer 模型的隐私保护推断。针对参数量高达数十亿的大语言模型，最近也有研究基于已有的 ABY3 协议或 Cheetah 框架进行<sup>[18,103-104]</sup>。目前这些方案在生成单个输出单词时需要至少进行数十 GB 的流量传输，在理想环境下也需要数分钟的计算时间，因此依然难以用于实际应用。

## 2.3 密码学与非密码学混合的隐私保护机器学习

考虑到基于密码学的隐私保护机器学习框架往往开销巨大且实现困难，也有许多研究尝试将密码学和非密码学方法进行融合，从而实现更高效的隐私保护机器学习。

一种方法是将同态加密和其他方法结合，其计算流程大致可以表示为

$$\llbracket \mathbf{y} \rrbracket = W \otimes \llbracket \mathbf{x} \rrbracket + \llbracket \mathbf{b} \rrbracket, \quad (2-10)$$

其中， $\llbracket \cdot \rrbracket$  表示加同态加密的密文， $\otimes$  表示明文和密文的同态乘法，其结果为乘积的密文。GELU-Net 算法<sup>[105]</sup>利用同态加密实现了两方的安全神经网络推断。该算法假设用

户拥有同态加密的私钥。对于神经网络的每一层，用户将数据上传至服务器，服务器执行同态乘法运算，得到加密的结果后返回给用户，用户再计算非线性的激活函数值。该方法虽然保护了用户的隐私，但是若用户多次进行查询，即可通过对多个输入  $\mathbf{x}_i$  和输出  $\mathbf{y}_i$  解线性方程组 ( $W\mathbf{x}_i + b = \mathbf{y}_i$ )，恢复出神经网络的权重。BAYHENN 算法<sup>[106]</sup>类似 Gelu-Net，区别在于其则采用贝叶斯神经网络来保证模型的隐私。在贝叶斯神经网络中，模型的权重是随机变量，使得根据输入和输出学习权重转换成一个“噪声学习（Learning With Errors）”问题，而该问题被证明是一个困难问题。然而后续研究指出<sup>[107]</sup>，BAYHENN 和 GELU-Net 的安全性都存在问题，很容易被攻击者通过选取特定输入的方式窃取模型参数。

此外，还有一些研究通过结合拆分学习和密码学方法，提高了隐私保护机器学习的效率。Zhou 等<sup>[108]</sup>提出在多方的纵向联邦学习中使用秘密分享计算神经网络的第一个全连接层，此后在服务器上明文计算。Chen 等<sup>[109]</sup>将该方法进一步扩展到图神经网络中。BlindFL 框架<sup>[110]</sup>将纵向联邦学习的底部模型拆分层使用密码学方法计算，同时明文计算底部模型的其他层和顶部模型。这些方法在一定程度上权衡了效率和隐私，相比于传统的拆分学习，其一般只暴露多个底部模型产生的联合表征而非单一参与方底部模型的输出。但是由于中间结果的暴露，其依然面临着较大的隐私泄漏风险，存在着被逆向攻击的可能性<sup>[25,27-28,39,111]</sup>。

综上所述，当前密码学和非密码学方法混合实现隐私保护机器学习的研究仍然处于起步阶段。

### 3 基于随机 Top- $k$ 算法的拆分学习效率优化

拆分学习是一种简单的隐私保护机器学习方法，其适用于纵向联邦学习和隐私推断的场景。在拆分学习中，各方交换模型的中间表征或梯度，实现多方的推断和训练。尽管拆分学习的效率远高于基于密码学的隐私保护机器学习方法，但是其效率依然存在进一步优化的空间。本章探索了拆分学习通信效率的优化方法，对稀疏、量化、缩小拆分层等方法进行了研究，从收敛和泛化两个角度进行优化，最终提出随机 top- $k$  稀疏化方法，提高了拆分学习的通信效率。

#### 3.1 研究背景

随着社会的数据隐私保护意识的提升和相关法律法规的出台，许多隐私保护机器学习方法被提出。拆分学习<sup>[19-20]</sup>作为纵向联邦学习<sup>[10]</sup>的一种主要方法，有着效率较高、实现简单的优势，因此获得了工业界和学术界的广泛关注<sup>[21-24]</sup>。

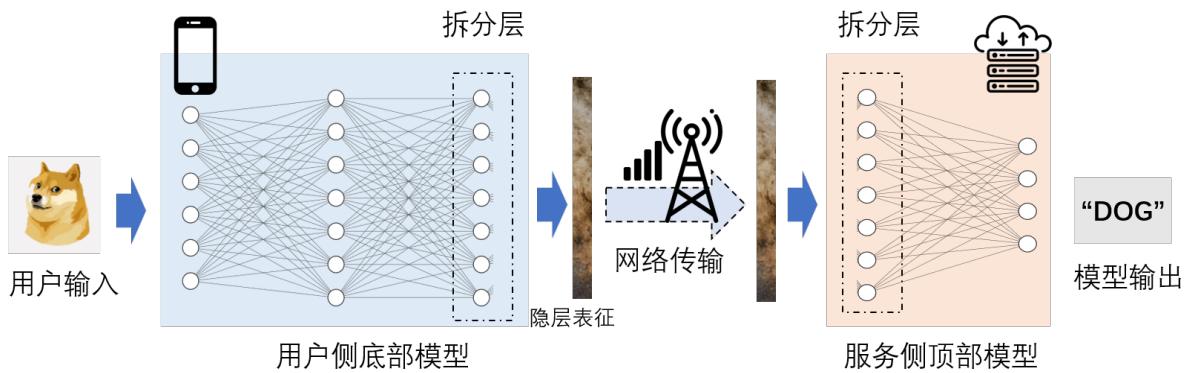


图 3.1 两方拆分学习推断示意图

拆分学习的思想是把模型拆分成多个部分分发给各个参与方，各方以交换中间结果的方式进行模型训练和推断，从而实现一定程度上的隐私保护。图 3.1 显示了一个典型的两方拆分学习的推断场景，我们假设用户具有输入数据，而服务器拥有一个分类模型。进行拆分学习的推断时，用户拥有底部模型，而服务器拥有顶部模型。用户将输入输入底部模型获得隐层表征后，通过网络传输给服务器，然后将其输入顶部模型，从而得到最终的模型预测值。在此过程中，用户只暴露了输入数据的隐层表征，因此其原始

输入获得了一定的保护；而服务器只暴露了底部模型而非完整模型，因此模型的隐私也在一定程度上得到了保护。虽然拆分学习的效率相对于密码学方法很高，但是由于拆分学习训练过程中依然需要在每一轮传输隐层特征和梯度，考虑到许多深度学习模型的隐层空间较大，因此拆分学习在通讯效率上依然存在可以优化的空间。

本文主要关注类别数量多的深度学习模型场景。分类模型往往可以拆分为特征抽取器和分类器两个部分，其中，特征抽取器的结构较为复杂，可以包含卷积层、长短期记忆层等；而分类器可能是一层简单的全连接层。但是在类别数量众多的情况下，分类器往往包含了模型的大部分参数，如：推荐系统模型<sup>[112-113]</sup>的最后一层可能包含了所有商品的嵌入向量（Embedding Vector）；人脸识别模型<sup>[114]</sup>的最后一层可能包含了所有用户的人脸对应的嵌入向量。在这种情况下，若简单地将整个模型部署到用户侧，既会占用大量的用户侧存储空间，也会带来模型隐私泄露的风险。因此，在这种情况下，就需要采用拆分学习的方式来提高效率，同时保护用户输入和模型参数的隐私。

当前关于纵向联邦学习通讯压缩的研究相对较少，仅有对基础压缩算法的收敛性分析<sup>[55]</sup>、拆分层嵌入自编码器<sup>[58]</sup>等，前者仅考虑了基本的压缩方法且只应用于训练阶段，而后者只针对拆分学习的一个特定场景。关于提升横向联邦学习的通讯效率的工作则较多，包括了加快收敛速度<sup>[115-116]</sup>、压缩每一轮传输的梯度值<sup>[46,50,54]</sup>等。前者并不适用于拆分学习，因此本文主要研究如何压缩拆分学习训练和推断过程中传递的中间结果或中间梯度信息。

## 3.2 压缩方法初步研究

本节介绍了几种基础的可以对拆分学习的通讯量进行压缩的方法，包括缩小拆分层、top- $k$  稀疏、量化拆分层、使用  $L1$  损失函数诱导稀疏，并对各方法进行初步分析。在以下的分析中，我们将拆分学习的底部模型记作  $M_b$ ，顶部模型记作  $M_t$ ，输入、中间结果、输出分别记作  $X, H, Y$ 。我们使用  $d$  表示初始的拆分层的维度， $k$  表示拆分层压缩后的维度。

**缩小拆分层：**在拆分学习中，每一轮训练或推断中都会传输拆分层的值以及（训练时）梯度。因此，缩小拆分层的大小可以直接减少拆分学习每一轮的通讯量。假设 1 个全连接网络的隐层维度分别是 1000（输入）-100（第 1 层）-100（第 2 层/拆分层）-10

(输出层)，则每一轮前向传播需要传输 100 个数值；若把拆分层维度缩小到 10，则只需要传输 10 个数值。同样地，在反向传播过程中，也需要传输 10 个梯度值。因此，这种方法可以削减 90% 的每轮通讯量。

**量化拆分层：**量化指的是将较高比特位的值（如 32 位浮点数）转化为较小比特位的近似值（如 16 位浮点数，4 位整数，甚至 1 位的 0-1 布尔值）<sup>[47-49]</sup>。其在神经网络中被广泛应用于减少计算和存储开销。通过对拆分层的值以及梯度进行量化压缩，也可以减少拆分学习训练和推断过程中的通讯量。本文考虑最常用的等距均匀量化（Uniform Quantization），对于一个隐层向量  $\mathbf{h} = (h_1, \dots, h_d)$ ，将其量化为  $b$  比特的量化方法为：

$$\mathbf{h}^C = \text{Compress}(\mathbf{h}) = \left( \left\lfloor \frac{h_1 - h_{\min}}{h_{\max} - h_{\min}} \cdot 2^b \right\rfloor, \dots, \left\lfloor \frac{h_n - h_{\min}}{h_{\max} - h_{\min}} \cdot 2^b \right\rfloor \right) \quad (3-1)$$

其中， $h_{\min}$  和  $h_{\max}$  分别表示向量  $\mathbf{h}$  中的最小值和最大值。而解压缩的过程则可以表示为：

$$\mathbf{h}' = \text{Decompress}(\mathbf{h}^C) = \left( \dots, h_{\min} + (h_i^C + \frac{1}{2})(\frac{h_{\max} - h_{\min}}{2^b}), \dots \right). \quad (3-2)$$

尽管量化压缩可以应用于前向传播的隐层值和反向传播的梯度值，但是在实验中我们发现，如果将量化压缩同时应用于前向传播和反向传播，会带来严重的模型效果损失。考虑到本文主要关注拆分学习推断过程中的通讯压缩，因此只将量化压缩应用于反向传播的隐层梯度中。

**Top- $k$  稀疏化：**Top- $k$  稀疏化指在向量中只保留  $k$  个绝对值最大的元素，而将其他元素设为 0。在这种情况下，只需要传输  $k$  个元素的值和下标。如果在前向传播中应用了 top- $k$  稀疏化，则较小的  $d - k$  个元素没有参与顶部模型的计算，因此，在反向传播过程中也无需传播这  $d - k$  个元素的信息。因此，top- $k$  稀疏化可以自然地同时应用于前向传播和反向传播。

**L1 正则化：** $L1$  正则化通过  $L1$  损失来诱导稀疏性，被广泛应用于不同的机器学习领域<sup>[117-120]</sup>。具体来说，在拆分学习的训练过程中，我们在原有的损失函数上加入一个对于拆分层的隐层向量的  $L1$  损失： $L' = L + \lambda \sum_{i=1}^d |h_i|$ 。其中， $\lambda$  用于控制稀疏化的强度，大的  $\lambda$  会导致更高的稀疏性，但是同时也会降低模型训练的效果。由于  $L1$  正则化需要在训练过程中逐渐实现稀疏化，因此其稀疏化只适用于模型的推断阶段，此时我们可以将隐层中接近 0 的元素删除，类似 top- $k$  稀疏化来传播绝对值较大的值和对应的下标。

我们将各种方法对应的压缩比率（压缩后大小/压缩前大小）总结在表 3.1 中。对于 top- $k$  即 L1 稀疏化，我们未考虑对下标进行压缩。表中  $N$  表示初始值的比特位数，一般为 32， $k$  表示保留的元素的个数。

压缩方法	压缩比率	
	前向传播	反向传播
拆分层缩小	$k/d$	$k/d$
$b$ 比特量化	$2^b/N$	1
Top- $k$ 稀疏化	$k/d \cdot (1 + \lceil \log_2 d \rceil / N)$	$k/d$
L1 正则化	$k/d \cdot (1 + \lceil \log_2 d \rceil / N)$	1

表 3.1 不同压缩方法的压缩比率

### 3.3 随机 top- $k$ 算法

本章节将提出随机 top- $k$  算法。随机 top- $k$  的优势来自于两个方面：

1. 从模型泛化 (Generalization) 角度，top- $k$  稀疏化优于缩小拆分层，因为同等压缩比率下，top- $k$  稀疏化能够提供更大的表征空间和决策边界大小。
2. 从模型训练角度，top- $k$  会导致部分神经元训练困难，导致收敛变慢，并且无法完全利用表征空间，从而降低其相对于缩小拆分层的优势。

基于以上两个观察，我们通过在 top- $k$  算法中加入随机扰动的方式，解决了第二点的部分神经元训练困难问题，从而提高了模型的收敛速度和泛化效果。

#### 3.3.1 Top- $k$ 算法和缩小拆分层算法的分析

首先，我们注意到在输出类别数较多的分类模型中，如果缩小拆分层的神经元数目，模型将会产生更大的泛化误差。而 top- $k$  算法因为在同等压缩率下有更大的样本空间，在理论上可以解决这个问题。

**增大类间距离可以提高模型泛化性能：**许多神经网络泛化误差的研究都与模型的光滑度 (Smoothness) 密切相关。简单而言，模型越光滑，其泛化效果越好<sup>[121-123]</sup>。考虑拆

分层在最后一层，激活函数是 Softmax 的情况。此时模型的输出值可以写成：

$$\mathbf{y} = \text{Softmax}(\mathbf{h} \cdot \mathbf{w}_1, \dots, \mathbf{h} \cdot \mathbf{w}_n) \quad (3-3)$$

其中， $\mathbf{h}$  表示拆分层的隐层表征， $\mathbf{w}_i$  表示最后一层权重矩阵的第  $i$  行， $n$  是类别个数。当模型训练到一个较高准确率后，假设  $\mathbf{h}$  对应某个第  $i$  类样本的拆分层表征，则  $\mathbf{h}$  和  $\mathbf{w}_i$  会比较接近，而和其他的权重  $\mathbf{w}_{j \neq i}$  距离较远。令最短类间距离 (Inter-Class Distance)  $d_W = \min_{i \neq j} \|\mathbf{w}_i - \mathbf{w}_j\|_2$ ，则我们可以估算顶部模型的光滑度为  $\|\nabla_{\mathbf{h}} M_t(\mathbf{h})\|_2 \approx c/d_W$ ，其中  $M_t$  表示顶部模型， $c$  是某个常数。直接理解， $d_W$  是两个不同类别对应的权重的最短距离。当拆分层表征从一个类别的表征  $\mathbf{w}_i$  变化到另一个类别的表征  $\mathbf{w}_j$  时，其最短的移动距离为  $d_W$ ，而模型的输出也应该从  $\text{onehot}(i)$  变化到  $\text{onehot}(j)$ 。可以看出， $d_W$  越小，要实现准确的分类效果，模型的输出就需要变化得越剧烈，也就是模型更不光滑；反之，模型的输出变化可以更加缓慢，模型可以更加光滑。由于光滑性和泛化误差是直接相关的，因此，我们可以使用  $d_W$  作为一个模型泛化的指标。

**同等压缩率下，top- $k$  比缩小拆分层有更大的类间距离：**首先注意到，类间距离和表征空间 (Feature Space) 的体积呈正相关—表征空间越大，则可以有更大的类间距离。但是一般神经网络的隐层表征的值域是无限大的，即  $\mathbb{R}^d$ ，导致类间距离无法计算。但是注意到，对于 Softmax 的多分类层，其分类结果仅仅和隐层表征的方向有关，而和其大小无关。因此，我们可以将拆分层表征空间加上一个范数约束，即： $\|\mathbf{h}\|_2 = 1$ ，而不影响模型输出结果。此时整个表征空间可以划分为  $n$  个类别的决策区域 (Decision Region)，假设第  $i$  个类别与其他所有类别的最短类间距离  $d_i$  和该类别的决策区域大小  $s_i$  呈现单调递增关系，则很显然得到最短类间距离最大时，各个类别应该有相同的决策区域大小，因为：

$$\min(s_1, \dots, s_n) \quad \text{s.t. } \sum_{i=1}^n s_i = S, s_i > 0 \quad (3-4)$$

在  $s_1 = \dots = s_n$  时取得最小值  $S/n$ 。其中， $S$  表示表征空间的总面积。

假设表征空间维度为  $k$ ，考虑到  $\|\mathbf{h}\|_2^2 = 1$  的约束，则表征空间构成一个  $k$  维超球面，其面积为：

$$S = 2\pi^{k/2}/\Gamma(k/2). \quad (3-5)$$

我们可以假设样本的决策区域面积约为对应的权重向量的附近的  $k-1$  维圆形区域。该圆形落在  $k-1$  维的球面空间，但是在类别数量多的情况下各个类别的决策区域

较小，因此可以近似为欧氏空间，也就是决策区域的大小可以用圆面积近似表示：

$$s_i \approx \frac{\pi^{(k-1)/2}}{\Gamma((k+1)/2)} \cdot r_i^{k-1}, \quad (3-6)$$

其中， $r_i = 1/2d_i$  表示决策区域的半径。此时，我们可以分别计算缩小拆分层和 top- $k$  稀疏化后的类间距离。

- 缩小拆分层：假设缩小后的拆分层维度为  $k$ ，则表征空间大小为  $k$  维球体的表面积大小： $S = 2\pi^{k/2}/\Gamma(k/2)$ 。而单个类别对应的决策区域为  $k-1$  维半径为  $r$  的圆形，其面积为  $2r^{k-1}\pi^{(k-1)/2}/\Gamma[(k+1)/2]$ 。考虑到总共有  $n$  个类别，可以得到以下关系：

$$nr^{k-1} \cdot 2\pi^{(k-1)/2}/\Gamma[(k+1)/2] \approx 2\pi^{k/2}/\Gamma(k/2). \quad (3-7)$$

于是可以求出决策边界半径大小：

$$r^{k-1} \approx \frac{2\sqrt{\pi}}{n} \cdot \frac{\Gamma[(k+1)/2]}{\Gamma[k/2]}. \quad (3-8)$$

注意到伽马函数（Gamma Function）的特性  $\Gamma(x + 1/2) \approx \Gamma(x) \times x^{1/2}$ ，上式可以进一步化简为

$$r \approx \left( \frac{2}{n} \sqrt{\frac{k\pi}{2}} \right)^{1/(k-1)}. \quad (3-9)$$

- Top- $k$  稀疏化：假设缩小后的拆分层维度为  $k'$ （在同等压缩率下， $k' < k$ ，因为需要传输额外的下标信息）。注意到此时的表征空间由  $\binom{d}{k'}$  个  $k$  维球面组成，因为 top- $k$  稀疏化每次从  $d$  维中选出  $k'$  维保留。因此，类似缩小拆分层的推导，可以得到决策区域半径为：

$$r' \approx \binom{d}{k'} \left( \frac{2}{n} \sqrt{\frac{k'\pi}{2}} \right)^{1/(k'-1)}. \quad (3-10)$$

对两个决策区域半径相除，得到：

$$\frac{r'}{r} = \frac{d'_W}{d_W} \approx \tilde{c} := \binom{d}{k'} \left( \frac{2}{n} \sqrt{\frac{k\pi}{2}} \right)^{1/(k'-1)-1/(k-1)} \left( \frac{k'}{k} \right)^{1/(k'-1)}. \quad (3-11)$$

其中，我们用  $\tilde{c}$  表示估计的决策区域半径（类间距离的一半）比值。现在我们考察  $k'$  和  $k$  的关系。很显然注意到，神经网络的隐层大小一般不超过  $2^{16} \approx 250K$ ，因此我们最多只需要 16 个比特来表示每个元素的下标；同时考虑一般的浮点数本身也是 32 位的，要

约束 top- $k$  的压缩率不高于缩小拆分层，只需要满足  $k' + 1/2k' \leq k$ ，其中  $k'$  表示  $k'$  个元素的值自身需要  $k'$  个浮点数的空间，而  $1/2k'$  表示下标需要  $k'/2$  个浮点数的空间（因为 16 位比特足以表示下标）。因此，可以取  $k' = 2/3k$  使得式(3-11)最大。

考虑指数项：

$$f(k) = \frac{1}{k'-1} - \frac{1}{k} = \frac{1}{2/3 \cdot k - 1} - \frac{1}{k-1} \leq 1/2. \quad (3-12)$$

对  $k$  求导得到：

$$f'(k) = \frac{1}{(k-1)^2} - \frac{1}{(3/2)(2/3 \cdot k - 1)^2} \quad (3-13)$$

注意到在  $k \geq 3$  时有：

$$(k-1)^2 = k^2 - 2k + 1 > 3/2 \cdot (2/3 \cdot k - 1)^2 = 2/3 \cdot k^2 - 2k + 3/2 \quad (3-14)$$

于是， $f'(k) \leq 0$  对  $k \geq 3$  都成立，同理得到  $k \geq 3$  时， $f(3) = 1/2 \geq f(k)$ 。

假设  $\sqrt{2k\pi} < n$ ，则  $2/n\sqrt{k\pi/2} \in (0, 1)$ ，又考虑到  $a^x$  在  $a \in (0, 1)$  时单调递减，代入式(3-11)，可以得到：

$$\tilde{c} \geq \frac{2}{3} \binom{d}{k'} \left( \frac{2k\pi}{n^2} \right)^{1/4}. \quad (3-15)$$

在  $\binom{d}{k'} > \sqrt{n}$  的情况下，该式大于 1。举例来说，比如原始的特征维度  $d = 1000$ ，类别数目  $n = 1000$ ，稀疏化后的  $k = 6, k' = 4$ ，则有  $\tilde{c} \approx 10^{10} \gg 1$ 。若将特征维度缩小到  $d = 100$ ，则  $\tilde{c} \approx 10^6 \gg 1$ 。

通过以上推导可以看出，使用 top- $k$  稀疏算法时，虽然保留的维度变少了一些，但是由于引入了  $\binom{d}{k'}$  这一组合项，使得表征空间远大于同等压缩率下直接缩小拆分层的方法。因此，从理论上来说，top- $k$  稀疏化将使得模型有更好的泛化性能。

### 3.3.2 Top- $k$ 算法训练收敛问题

尽管上文的推导证明 top- $k$  稀疏化具有更好的泛化性能，但是其依然可能存在收敛性上的问题。因为将 top- $k$  稀疏化运用在训练过程中时，存在“赢家通吃”的问题，导致非 top- $k$  的神经元可能难以得到训练。具体而言，假设一个神经元对应的输入权重都较低，那么这个神经元很可能在不同的输入样本上都会获得较小的激活值，从而导致其一直无法得到训练。在这种情况下，top- $k$  可能陷入局部最优。下文将用一个简单的例子来说明 top- $k$  陷入局部最优的情况。

考虑如下的目标函数：

$$f : (x_1, x_2) \rightarrow \text{Sign}(x_1 - x_2) \quad (3-16)$$

以及如下的拆分逻辑回归模型：

$$\begin{cases} M_b : (x_1, x_2) \rightarrow (h_1, h_2) = (w_1 x_1, w_2 x_2) \\ M_t : (h_1, h_2) \rightarrow \text{Tanh}(h_1 + h_2). \end{cases} \quad (3-17)$$

这里我们用  $x_1, x_2 \in \mathbb{R}$  表示二维的输入特征， $h_1, h_2 \in \mathbb{R}$  表示拆分层的表征，而  $w_1, w_2$  表示底部模型的权重，初始值为  $w_1 = 1, w_2 = -0.1$ 。

假设有以下两个训练样本：

$$\begin{cases} \mathbf{x}_1 = (-1, 0), \quad y_1 = 1 \\ \mathbf{x}_2 = (0.5, 1), \quad y_2 = -1. \end{cases} \quad (3-18)$$

很容易可以看出，在 top- $k$  ( $k = 1$ ) 稀疏的情况下， $w_1 \rightarrow +\infty, w_2 \rightarrow -\infty$  是最优的权重。我们在图 3.2 中显示了这个例子的损失函数曲面和梯度下降方向。

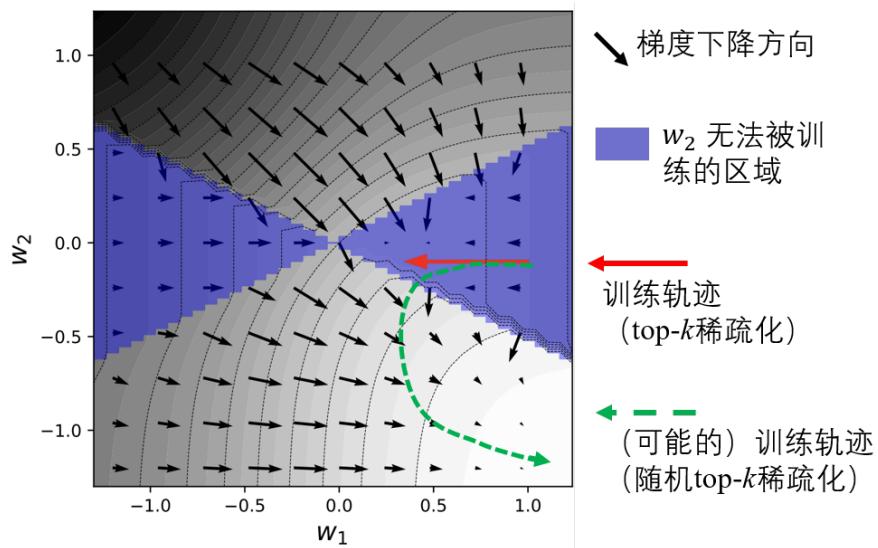


图 3.2 使用 Top- $k$  稀疏化时的损失函数曲面和梯度下降方向

然而如果在训练过程中使用 top- $k$  稀疏化（假设使用平方损失函数），由于  $h_2$  的值在各个样本上都很小，则  $h_2$  总是无法被选为 top- $k$ ，从而导致其对应的权重  $w_2$  一直无法被训练（对应图中的蓝色区域）。在这种情况下，模型会收敛到一个远离最优点的区

域，使得  $w_1$  降低到 0.4 左右，平衡两个样本的误差，而  $w_2$  则未被更新而停留在 0.1 处，对应图 3.2 中的红色训练轨迹。但是如果在 top- $k$  的过程中加入一点随机性，使得  $h_2$  也有一定的几率被选中，则可以让  $w_2$  被训练到并且更新，从而摆脱局部最优。图中的绿色轨迹表明了在 top- $k$  稀疏中加入随机性的一条可能的训练轨迹。

Top- $k$  稀疏化的收敛问题也会进一步导致其泛化性能降低。如上文所述，top- $k$  稀疏化带来的泛化优势是因为其表征空间包含了  $\binom{d}{k}$  个超球面，其中  $d$  是原始的拆分层维度（神经元个数）， $k$  是稀疏化后保留的维度。但是由于 top- $k$  稀疏化带来某些神经元（维度）一直难以得到训练的问题，导致某些神经元在各个样本都无法被选中为 top- $k$ ，从而导致这些神经元被在模型中无法发挥作用，缩小了表征空间的大小。具体而言，如果采用 top- $k$  算法进行训练，假设有  $d'$  个神经元处于“废弃”状态，则最终得到的表征空间中的超球面数目就会从  $\binom{d}{k}$  减少到  $\binom{d-d'}{k}$ 。同样地，如果在 top- $k$  稀疏化中加入随机性，将使得各个神经元都有一定的机会被训练，提高表征空间的利用率，从而提高泛化效果。

### 3.3.3 随机 top- $k$ 算法定义

基于以上分析，我们可以定义如下的随机 top- $k$  算法：选择  $k$  个要保留的神经元时，每一次选择某个神经元的概率为：

$$P(\text{选择该神经元}) = \begin{cases} (1 - \alpha)/N_1 & \text{如果该神经元是 top-}k \text{ 的,} \\ \alpha/N_2 & \text{反之,} \end{cases} \quad (3-19)$$

其中， $N_1$  和  $N_2$  分别表示当前未被选择的 top- $k$  神经元数目和非 top- $k$  神经元数目。 $\alpha$  显示了随机 top- $k$  算法的可能执行结果。可以看出， $\alpha$  表示选择非 top- $k$  神经元的概率。当  $\alpha = 0$  时，随机 top- $k$  算法退化为普通的 top- $k$  稀疏化；而当  $\alpha = 1$  时，则转变为完全随机的 Dropout 算法。注意到，随机 top- $k$  仅仅被应用于模型的训练阶段，在推断阶段，我们依然采用确定性的 top- $k$  稀疏化。通过实验，我们发现  $\alpha \in [0.05, 0.1]$  时模型能取得较好的训练效果。



图 3.3 随机 top-k 算法示意图

### 3.3.4 隐私分析

由于大部分的拆分层表征元素被丢弃，因此 top- $k$  稀疏化或随机 top- $k$  稀疏化都能够比原始的拆分学习更多地保护输入特征的隐私。这一点在已有的研究工作中也得到了证明。我们也在后续实验章节中提供了通过拆分层表征对输入特征进行重建攻击（Reconstruction Attack）的结果。尽管如此，研究表明拆分学习对于标签恢复攻击（Label Inference Attack）或模型补全攻击（Model Completion Attack）较为脆弱，随机 top- $k$  算法也并不能解决标签的隐私问题。因此本文考虑的应用场景为标签类别数量大且标签推断攻击不可行的深度学习应用。比如，包含了数千类的人脸识别模型、包含了上万种商品的推荐系统模型。对于这些模型，使用者难以获取其他类别对应的输入特征，因此也难以进行标签推断攻击或是模型补全攻击。

## 3.4 实验分析

为了验证本章所提出的随机 top- $k$  算法的性能，我们在 4 个不同类型数据集上使用不同的模型进行了实验：

- CIFAR-100<sup>[124]</sup>: 一个常用的包含了 100 类总共 5 万张大小为  $32 \times 32$  的彩色图片。我们使用 ResNet-20 模型<sup>[125]</sup>进行分类，拆分层设置为最后的隐层，大小为 128。
- YooChoose (1/64)<sup>[126]</sup>: 一个推荐系统的数据集，包含了大约 15 万条用户的点击序列，总共有约 1.8 万个商品类别。我们使用 GRU4Rec 模型<sup>[112]</sup>进行分类，隐层大小设置为 300。拆分层设置为最后的隐层，大小为 300。

- DBpedia<sup>[127]</sup>: 一个文本分类数据集，包含了 219 种类别，总共有大约 34 万条文本。我们采用 TextCNN 模型<sup>[128]</sup>进行分类，卷积核的大小为 (3, 4, 5)，并且使用 Glove 预训练词向量<sup>[129]</sup>。拆分层设置为最后的隐层，大小为 300。
- Tiny-Imagenet<sup>[130]</sup>: 一个图像分类数据集，包含了 200 类的 10 万张彩色图片，图片尺寸为  $64 \times 64$ 。我们采用 EfficientNet-b0 模型<sup>[131]</sup>进行分类。拆分层设置为最后的隐层，大小为 1280。额外地，我们对权重采用 ImageNet 的预训练权重和随机初始化两种情况分别进行了实验并汇报结果。

对于每个任务，我们都测试了不同的模型压缩方法和压缩比率，并且对每一个设定都重复了 5 次实验取平均值汇报。实验的代码基于 Pytorch 框架编写，在带有 NVIDIA RTX 3090 的服务器上进行。实验时我们按照 8:1:1 的比例划分训练集、验证集合测试集（对于 Yoochoose 数据集按照先后时间划分），使用 Early Stop 策略获得验证集上最佳的模型，然后在测试集上进行测试。

### 3.4.1 压缩比率和模型准确率对比

我们测试了模型在测试集上的准确率（对于 Yoochoose 数据集，我们用前 20 准确率代替），汇报在。我们把随机 top- $k$  的随机参数在 CIFAR-100, DBpedia, Tiny-Imagenet 三个任务上  $\alpha$  设置为 0.1，在 Yoochoose 任务上设置为 0.05。我们把实验结果汇报在表 3.2 中的。表内每一项的格式为“准确率/压缩比率 \*100”，任务名称下方的准确率表示无压缩的普通拆分学习的准确率（压缩比率 =100）。表内空白项表示该方法无法达到对应的压缩比率。我们用粗体表示同等压缩比率下最高的准确率，用下划线表示次高的准确率。

实验结果表明，随机 top- $k$  算法几乎在所有任务上都取得了最好的准确率和最低的压缩比率，且大幅领先于其他方法，在很低的压缩比率情况下依然保持了和原始无压缩拆分学习相近的表现。在 YooChoose 任务中，随机 top- $k$  算法甚至超过了无压缩的拆分学习，我们认为这可能归功于随机 top- $k$  的正则化效果。同时，我们注意到，量化方法和  $L1$  正则化方法在无法达到一些较低的压缩比率。这是因为量化方法最低智能达到  $1/32$  的压缩比率，并且此时拆分层表征被压缩为 2 值，往往会导致模型无法收敛。而  $L1$  正则化在系数过大时也会导致模型无法收敛。

表 3.2 实验结果：准确率和压缩比率

任务	压缩	随机 top- $k$	Top- $k$	缩小拆分层	量化	L1 正则化
CIFAR-100 67.20	高	<b>65.25 / 2.86</b>	<u>62.23 / 2.86</u>	55.52 / 3.13	-	-
	中	<b>65.83 / 5.71</b>	<u>61.56 / 5.71</u>	60.43 / 6.25	53.56 / 6.25	<u>62.11 / 8.41</u>
	低	<u>65.98 / 12.4</u>	62.11 / 12.3	62.93 / 12.4	<b>66.01 / 12.5</b>	63.87 / 19.5
YooChoose 63.57	高	<b>60.29 / 0.85</b>	<u>60.28 / 0.85</u>	50.71 / 1.00	-	-
	中	<b>64.55 / 1.71</b>	<u>63.81 / 1.71</u>	62.20 / 2.00	-	-
	低	<b>66.88 / 3.84</b>	<u>66.12 / 3.84</u>	<u>66.12 / 4.00</u>	64.69 / 3.13	61.48 / 3.01
DBpedia 93.11	很高	<b>84.88 / 0.44</b>	<u>83.04 / 0.44</u>	64.80 / 0.50	-	-
	高	<b>88.01 / 0.88</b>	<u>85.49 / 0.88</u>	78.57 / 1.00	-	81.35 / 1.08
	中	<b>90.50 / 1.97</b>	87.74 / 1.97	86.42 / 2.00	-	<u>87.88 / 0.93</u>
	低	<u>91.59 / 3.06</u>	90.05 / 3.06	88.38 / 3.00	91.20 / 6.25	<b>93.11 / 5.31</b>
Tiny-ImageNet 随机初始化 53.11	高	<b>50.83 / 0.21</b>	48.36 / 0.21	35.46 / 0.23	-	-
	中	<b>51.75 / 0.42</b>	<u>47.24 / 0.42</u>	45.66 / 0.47	-	-
	低	<b>51.16 / 0.94</b>	45.50 / 0.94	48.87 / 0.14	-	-
Tiny-ImageNet 预训练 75.18	高	<b>71.09 / 0.21</b>	<u>70.86 / 0.21</u>	59.23 / 0.23	-	-
	中	<b>72.15 / 0.42</b>	<u>71.19 / 0.42</u>	66.52 / 0.47	-	-
	低	<b>73.83 / 0.94</b>	<u>72.52 / 0.94</u>	68.67 / 0.94	-	67.82 / 1.24

### 3.4.2 训练速度分析

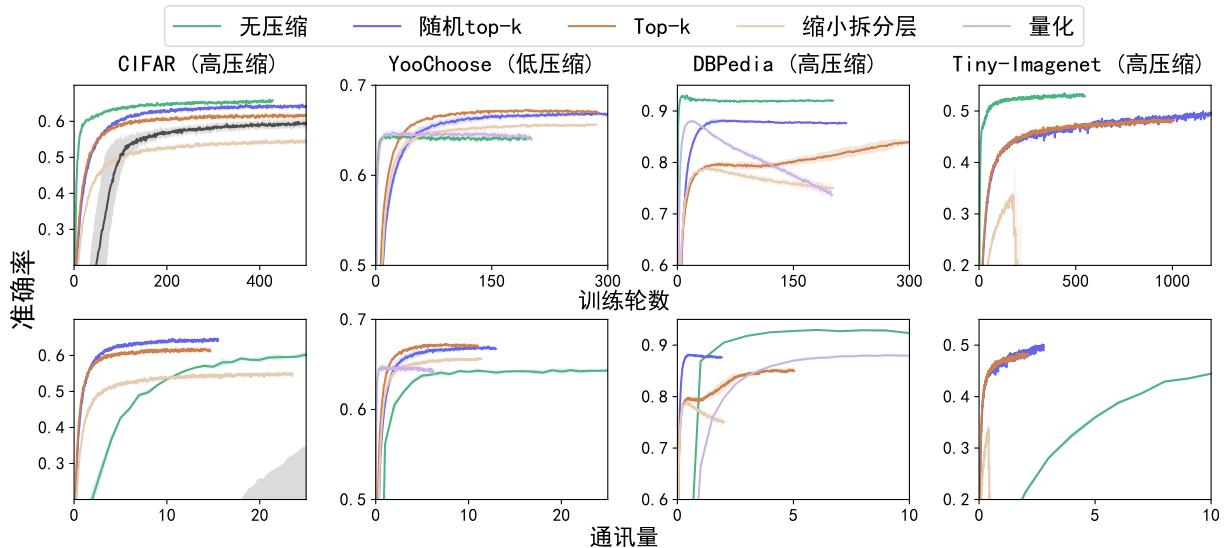


图 3.4 训练轮次/通讯量和准确率

我们在图 3.4 中汇报了训练过程中准确率和训练轮次以及通信量的关系。图的第一行是训练轮次和准确率的对比，第二行是通信量和训练轮次的对比，其中我们把一轮无

压缩的拆分学习的通信量设为 1。实验结果表明，无压缩的拆分学习收敛所需要的训练轮次数最少。但是以通信量衡量时，几乎所有压缩方法的收敛所需的通信量都不无压缩的拆分学习少，并且相比于其他方法，随机 top- $k$  的收敛速度和准确率都是最高的。

### 3.4.3 随机参数 $\alpha$ 分析

本节我们汇报了变化随机参数  $\alpha$  时的实验结果，包括了  $\alpha$  和准确率、收敛速度、泛化误差，top- $k$  神经元分布、以及输入特征重构攻击效果的关系，为选取合适的  $\alpha$  提供参考，并且印证前文关于收敛性、泛化效果以及隐私保护的理论分析。

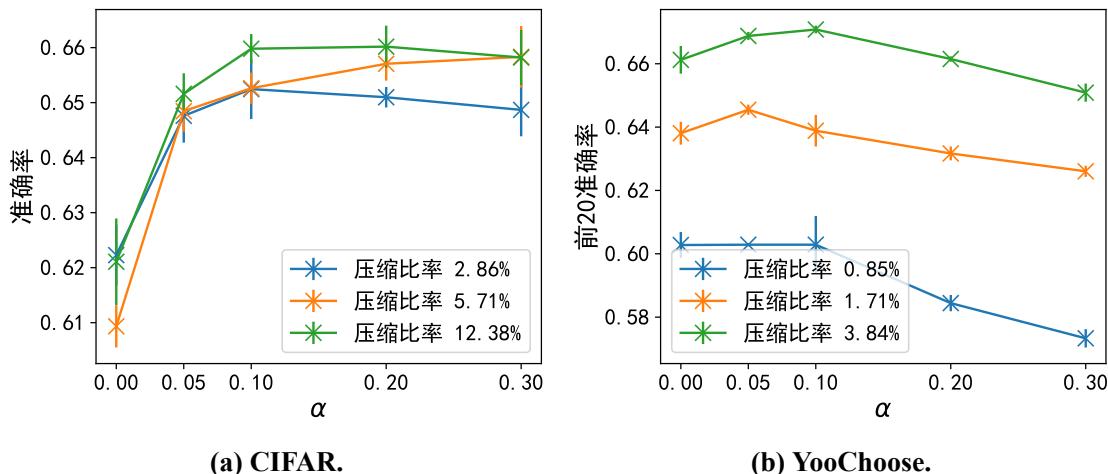


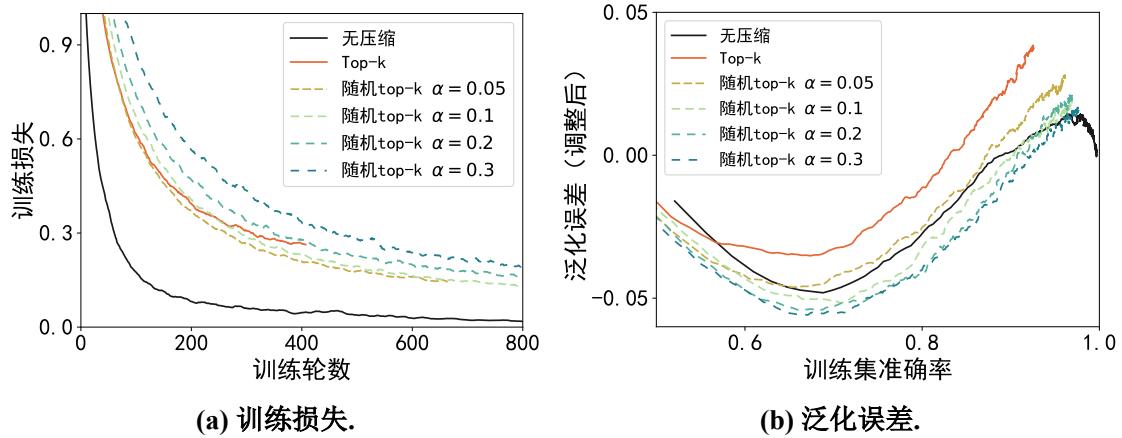
图 3.5  $\alpha$  与准确率的关系

**准确率：**图 3.5 展示了在 CIFAR-100 任务中  $k = 3$  情况下，随机参数  $\alpha$  与模型测试准确率的关系。可以看出，在 CIFAR-100 任务中，无论怎样选取  $\alpha$  都可以比无压缩情况下显著提高准确率，而在 YooChoose 任务中，提高的幅度则相对有限。同时，在  $\alpha$  提高到 0.1 之后，准确率随着  $\alpha$  的进一步提高呈现出下降趋势。分析表明， $\alpha \in [0.5, 1]$  可以使得模型达到较高准确率；而过大的  $\alpha$  会引入过多的噪声，从而损害模型的效果。

**收敛性和泛化误差：**图 3.6 展示了在 CIFAR-100 任务中  $k = 3$  情况下，随机参数  $\alpha$  与模型训练损失和泛化误差的关系。为了更清晰呈现泛化误差的变化，我们再对泛化误差进行了调整，将图 3.6a 调整为的 Y 轴调整为如下：

$$y = \text{泛化误差} (\text{训练集准确率} - \text{测试集准确率}) - 0.5 \times \text{训练集准确率} + 0.2. \quad (3-20)$$

从图中可以看出，top- $k$  稀疏化在训练开始时损失下降较快，但是随着训练轮数的增长，随机 top- $k$  的损失下降变快，并且最终收敛的损失低于 top- $k$ 。另外，太大的  $\alpha$  也导致损

图 3.6  $\alpha$  与训练损失以及泛化误差的关系

失下降变慢。同时，相对于无压缩的情况， $\text{top-}k$  稀疏化显著增加了泛化误差。而增大  $\alpha$  也显著降低了泛化误差。这和我们在节 3.3 中对于随机  $\text{top-}k$  算法的收敛性和泛化性的理论分析相对应。

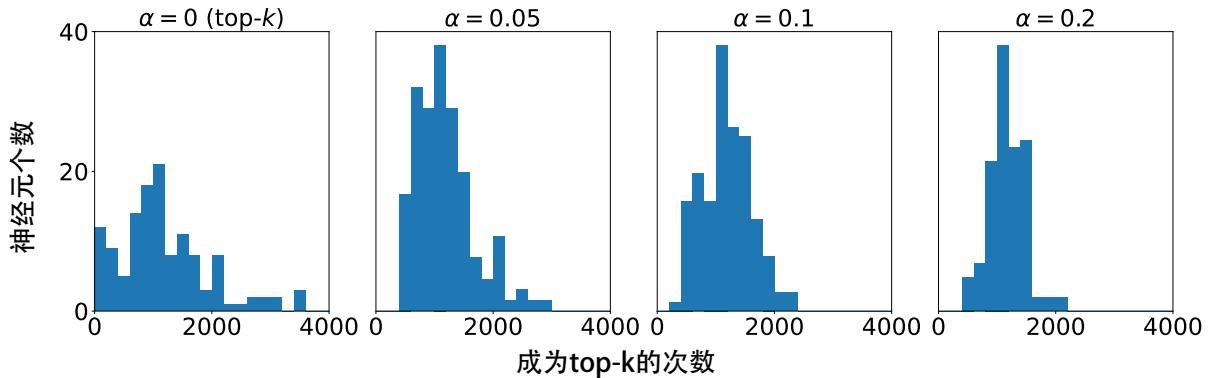


图 3.7 两方拆分学习推断示意图

**Top- $k$  神经元的分布** 图 3.7 显示了采用不同的  $\alpha$  训练后，在推断阶段使用测试集样本时，神经元被选为  $\text{top-}k$  次数的分布。具体而言，隐层的第  $i$  个神经元被选为  $\text{top-}k$  的次数按照如下公式计算：

$$C_i = \sum_{j=1}^N [M_b(X_j) \text{ 的 } \text{top-}k \text{ 神经元包含了其第 } i \text{ 个神经元 (是 =1, 否 =0)}], \quad (3-21)$$

其中， $N$  表示测试集样本数， $X_j$  表示第  $j$  个样本。可以看出，仅使用  $\text{top-}k$  稀疏化时，神经元被选为  $\text{top-}k$  次数的分布不均匀，部分神经元几乎从未被选为  $\text{top-}k$ ，而某些神经元则总是被选中；而使用随机  $\text{top-}k$  有效地解决了这一问题， $\text{top-}k$  次数的分布变得均匀，说明各个神经元被选为  $\text{top-}k$  的概率更加均等。即使是一个较小的  $\alpha$  (0.05)，也能显著

使得 top- $k$  次数的分布变得均匀。这也说明，随机 top- $k$  更好地利用了表征空间，从而提高了泛化性能。

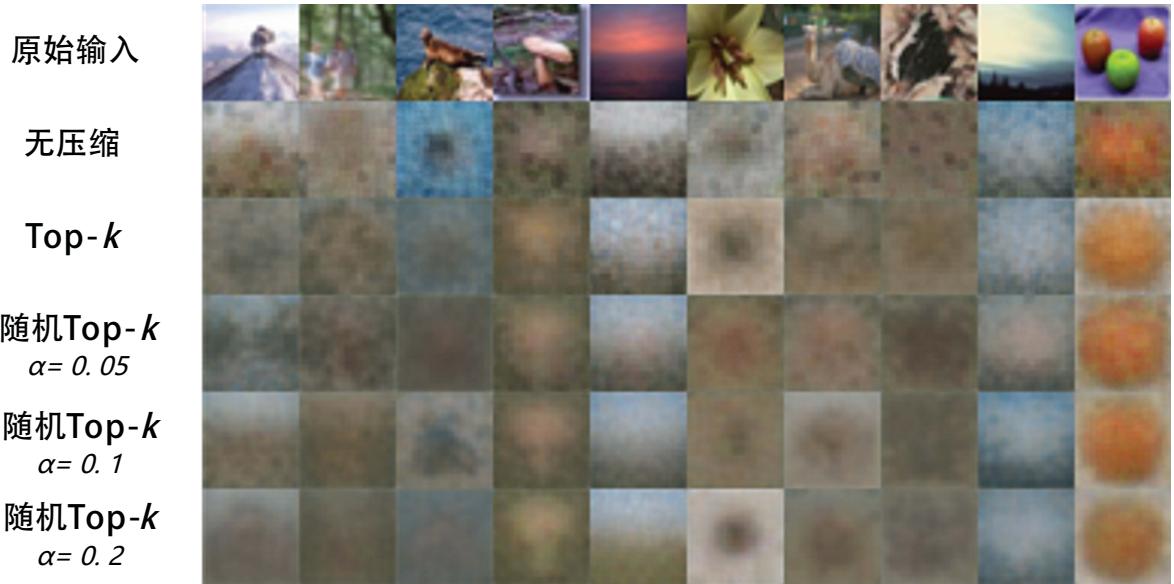


图 3.8 CIFAR-100 输入重建攻击效果

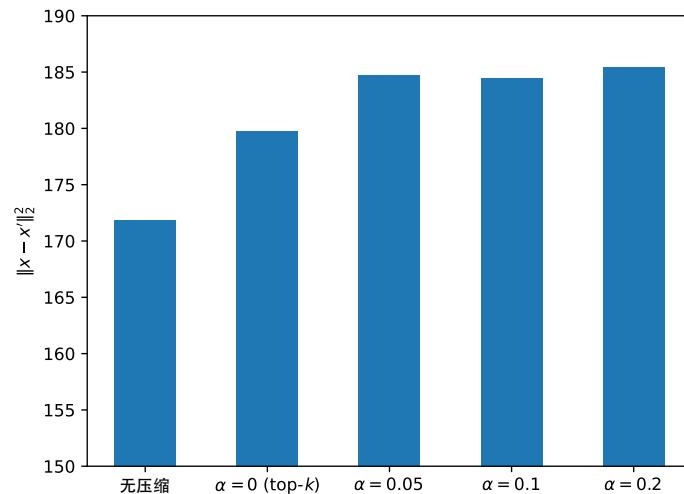


图 3.9 CIFAR-100 输入重建攻击误差

### 3.4.4 隐私分析

为了表明随机 top- $k$  压缩方法对于输入数据的隐私的影响，我们在 CIFAR-100 数据集上进行了输入特征重建攻击的实验。输入重建攻击是一种从隐层恢复出输入数据的手段，其方法是通过泄漏的数据，训练一个重构神经网络将拆分层数据逆向回原始输入特

征<sup>[26-27]</sup>。本实验中，重构神经网络的结构为（全连接，卷积，反卷积，卷积），其中全连接层将 128 维拆分层投影为  $4 \times 16 \times 16$ （4 表示频道数），此后三层的输出大小分别为  $16 \times 16 \times 16$ ,  $32 \times 32 \times 32$ ,  $3 \times 32 \times 32$ 。激活函数为 LeakyReLU<sup>[132]</sup>。

图 3.8 展示了在 CIFAR-100 任务中  $k = 3$  情况下，在模型推断阶段，攻击者根据拆分层表征对原始输入图片进行恢复攻击的效果。从图中可以看出，由于拆分层被设置在最后一层线性层，因此即使是无压缩的情况下，攻击者依然难以还原图片的主要信息，只能呈现出模糊的类似于类别的“平均图”的图片。而 top- $k$  和随机 top- $k$  进一步使得原始图片中的色彩信息几乎也被丢失。

为了更加清晰地现实随机 top- $k$  和其他方法的输入特征隐私保护效果，我们测量了图片恢复攻击的平方误差，即  $\mathbb{E}[\|\text{恢复值} - \text{原始值}\|_2^2]$ ，并汇报在图 3.9 中。从图中可以看出，top- $k$  稀疏相对于原始无压缩的拆分学习显著提高了攻击者的重建损失，增强了隐私保护效果。而随机 top- $k$  比 top- $k$  拥有更高的重建损失，说明其进一步提高了对于输入特征的隐私保护。

### 3.5 本章小结

本章节针对多分类拆分学习中的通信效率问题，提出了随机 top- $k$  稀疏方法，并且从泛化误差、收敛性以及隐私性三个角度进行了理论和实验分析，证明了随机 top- $k$  算法的优越性。我们通过近似计算表征空间大小，表明了 top- $k$  算法在同等压缩率下拥有更大的表征空间，从而能够使得各个类别的决策区域更大，带来更低的泛化误差。同时，通过一个二维例子说明了 top- $k$  算法在收敛性上可能面临部分神经元无法被训练到的问题，而在 top- $k$  中加入随机性可以有效地防止该类问题。基于以上分析，我们提出随机 top- $k$  算法对传统的 top- $k$  进行了改进。通过与 top- $k$  算法、拆分层量化、L1 正则化、缩小拆分层等方法对比，随机 top- $k$  显示出了其在拆分学习训练与推断过程中的优越性，包含了更高的模型准确率和训练/推断速度。此外，实验结果也表明稀疏化后的拆分层表征也能降低对于输入特征的隐私泄漏问题。总之，随机 top- $k$  算法显著提高了类别数量众多时拆分学习训练和推断过程中的通讯效率，为拆分学习模型在实际应用中的部署提供了有力支持。

## 4 基于势能损失的隐私保护拆分学习

拆分学习是一种简单高效的隐私保护机器学习方法，其通过将模型分成与输入特征相连的底部模型和与预测标签相连的顶部模型，各方运算自己对应的部分并交换中间结果来实现推断和训练。虽然拆分学习无需暴露原始的输入数据，但是其中间结果的交换也带来了隐私泄漏风险。其中一个主要的风险是模型补全攻击（Model Completion Attack），即在分类任务中，攻击者可以很容易根据中间结果和少量泄露的标签训练出高准确率的顶部模型，从而实现窃取模型或标签信息的目的。本文将模型补全攻击转化为攻击者的有监督或无监督学习问题，从泛化误差角度对其进行了分析，并由电磁学现象启发，提出了势能损失函数，对模型补全攻击实现了有效的防护。

### 4.1 研究背景

拆分学习是一种高效的隐私保护机器学习手段<sup>[19-20]</sup>。在拆分学习中，模型被分割为数个部分分发给各个参与方，各个参与方在模型的训练或推断过程中只需要交换中间结果而非原始数据，在一定程度上保护了数据的和模型参数的隐私。相比于基于密码学的隐私保护机器学习方法，拆分学习具有非常高的通讯和计算效率。如今拆分学习已经被应用在机器学习的多个领域中<sup>[22-23,133]</sup>。

由于第3章中已经对拆分学习进行了基本的介绍，本节对此不再赘述。虽然拆分学习拥有实现简单、效率高等诸多优势，但是在训练和推断过程中，各方直接交换了中间结果和中间梯度的明文，从而存在一定的隐私泄漏风险。本章主要考虑的是拆分学习中的模型和标签数据的隐私问题。不失一般性地，我们考虑两方拆分学习的场景，其中用户拥有输入特征（ $X$ ），而模型拥有方拥有模型参数并把底部模型（ $M_b$ ）下发给用户。一个两方拆分学习推断的流程可以表示为：

$$X \xrightarrow{M_b} H \xrightarrow{M_t} Y, \quad (4-1)$$

其中， $X$  是输入特征， $H$  是拆分层的隐层表征， $Y$  是预测的样本标签，而  $M_b, M_t$  表示底部模型和顶部模型。可以看出，拆分层表征  $H$  同输入特征  $X$ 、预测值  $Y$  都具有一定相关性，因此有一定的隐私泄漏风险。

目前已经有一些研究考察了从拆分层表征逆推输入特征的攻击，并提出了对应的防御手段，如 Vepakomma 等人<sup>[26]</sup>提出了在训练过程中加入额外的基于距离相关性<sup>[42]</sup>的损失函数  $D_{cor}(H, X)$  来减少表征中包含的输入信息，并取得了一定的效果。而拆分层表征泄露标签隐私的问题则更为严重。Fu 等人<sup>[37]</sup>提出通过少量泄漏的带标签样本即可从随机初始化训练出顶部模型，从而窃取样本的标签以及整体模型，并将这种攻击称之为模型补全攻击（Model Completion Attack）。针对这种攻击，Sun 等人<sup>[41]</sup>提出使用距离相关性对表征和输出标签进行解耦，并取得了一定的效果。除此之外，拆分学习的训练过程中的梯度也会泄露标签隐私。Li 等人<sup>[38]</sup>表明的拆分层的梯度和样本标签的相关性很高，并提出了一些对于二分类模型的防御方法。Sanjay 等人<sup>[134]</sup>在训练时采用替代的头部模型对梯度进行匹配从而实现盗取标签的目的。本文研究的是针对分类模型的拆分学习推断阶段，尽可能地对  $H \rightarrow Y$  这一隐私泄漏链条进行保护，防止模型补全攻击。

直觉而言，拆分层表征泄露模型输出信息似乎是不可避免的。因为在模型训练过程中，其逐渐丢弃隐层表征中关于输入特征的无关信息，只保留与标签相关的信息。为了模型能产生准确的预测，隐层表征中必须包含关于标签的信息。因此，本文并不尝试将标签相关信息从拆分层表征中彻底删除，而是把攻击问题转化成一个监督学习或无监督学习问题，研究如何使得攻击者难以从拆分层表征中获取标签信息。

## 4.2 问题描述

本节我们对拆分学习中底部模型产生的拆分层表征带来的对顶部模型以及标签的隐私泄漏问题（模型补全攻击）进行描述和定义，并且将该隐私泄漏问题转化为一个攻击者的有监督或无监督的学习问题。

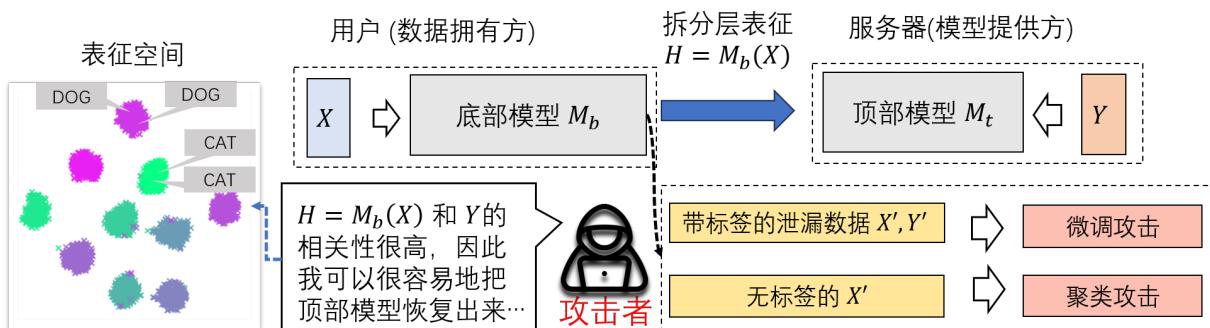


图 4.1 拆分学习中底部模型带来的隐私泄漏问题和模型补全攻击示意图

### 4.2.1 拆分层的隐私泄露

许多对神经网络隐层进行可视化的研究都揭露了如下的规律：在神经网络的训练过程中，隐层表征与样本标签的关联性逐渐增加，相同标签对应的隐层表征会逐渐聚集形成簇，而不同标签对应的隐层表征会逐渐分离开<sup>[32-34]</sup>。并且位置越靠后的隐层的表征，与样本标签的关联性越高。尽管上述的“同类别隐层表征靠近，不同类别隐层表征分离”的规律是神经网络具备学习能力的一个必需原因，这个规律给拆分学习带来了很大的隐私泄漏风险。

理想的两方拆分学习  $X \xrightarrow{M_b} H \xrightarrow{M_t} Y$  应该满足下述的两个条件：

- 从  $H$  推出  $X$  是困难的，从而保证用户的输入数据的隐私。
- 从  $H$  推出  $Y$  是困难的，因此用户必须和模型拥有方合作，才能获取模型预测值，而不能仅仅通过  $M_b$  获取预测值，从而保证模型的隐私。

然而，前面所述的神经网络隐层表征的性质，恰恰说明  $H$  推出  $Y$  可能是十分容易的，与我们期望的理想拆分学习的性质所矛盾。假设攻击者腐化了用户，它可以轻松地通过少量带标签样本  $(\dots, (X_i, Y_i), \dots)$ ，然后通过用户侧的底部模型  $M_b$  来产生拆分层表征  $(\dots, H_i, \dots)$ ，并通过拆分层表征-标签对  $(\dots, (H_i, Y_i), \dots)$  来训练顶部模型  $M_t$ ，从而窃取整个模型；同时，攻击者也可以直接通过无标签的样本产生大量表征，并对其进行聚类得到类别标签，从而得到样本标签  $Y$  或训练出顶部模型  $M_t$ 。我们将这两种攻击统称为模型补全攻击（Model Completion Attack），与现有研究一致。图 4.1 是模型补全攻击的示意图。

### 4.2.2 威胁模型

为了更清晰地定义上述的隐私泄漏问题，我们定义两种攻击方法：

- **微调攻击：**假设攻击者具有底部模型  $M_b$ ，并且拥有少部分泄露的带标签样本  $Z = (\dots, (X'_i, Y'_i), \dots)$ ，其中每一种类别的样本都有  $k$  个。攻击者也知道顶部模型  $M_t$  的结构，但是不知道具体参数。攻击者通过  $(\dots, (X'_i, Y'_i), \dots)$  和固定的  $M_b$  来微调  $M_t$ ，从而实现窃取模型的目的。

- **聚类攻击：**假设攻击者具有底部模型  $M_b$ ，并且拥有大量泄漏的无标签样本  $X = (\dots, X_i, \dots)$ 。攻击者通过底部模型产生中间表征  $H = (\dots, H_i, \dots)$ ，然后对其进行聚类产生类别标签，使得聚类模型实现顶部模型  $M_t$  的功能，从而实现窃取整体模型的目的。

注意到，我们的攻击场景设定为拆分学习的推断阶段而非训练阶段，因为本文的目标是降低拆分层表征的隐私泄露。如果要对拆分层梯度进行保护，可以使用非敏感数据集进行训练，或者使用基于密码学的隐私保护神经网络训练方法<sup>[77,86]</sup>。除此之外，一些研究也指出对于无标签的样本进行无监督学习也能产生一个表现良好的分类模型<sup>[135-136]</sup>。但是这些方法与拆分学习无关，因为它们不需要任何关于底部模型或是拆分层表征的知识。本文不对这些方法进行讨论。

### 4.2.3 问题定义

表面上看来，上述的攻击和模型正常的训练过程的目标是类似的：两者都会学习从拆分层表征  $H$  到标签  $Y$  的映射。然而攻击者的训练过程和正常的模型训练过程存在着一个本质区别，即两者的训练数据不同。正常的模型训练工程需要大量的带标签数据，而在模型补全攻击中，我们假设攻击者只能获取少量的带标签样本。因此，我们可以把模型补全攻击转化成一个机器学习问题。具体而言，防御的目标转换为：控制底部模型  $M_b$ ，使得整体模型  $(M_b, M_t)$  有较高的准确率，但是使用少量带标签的  $H = M_b(X)$  对随机初始化的  $M_t$  进行微调，则会产生较高的泛化误差。

**定义 4.2.1 (底部模型优势)** 我们定义底部模型优势为攻击者利用底部模型  $M_b$  进行攻击产生的额外优势。考虑一个攻击者  $\mathcal{A}$  和其输入数据  $D$  以及底部模型  $M_b$ （可以为空），其输出则为一个完全模型  $M = (M_b, M_t) : X \rightarrow Y$ ，底部模型优势可以定义为：

$$Adv(M_b; \mathcal{A}) = \mathbb{E}_D\{R[\mathcal{A}(D; null)] - R[\mathcal{A}(D; M_b)]\}, \quad (4-2)$$

其中， $R[\cdot]$  表示攻击者产生的完全模型在测试集上的误差， $\mathcal{A}(D; null)$  表示攻击者在没有底部模型时的产生的完全模型。

例如，在微调攻击时， $D = (\dots, (X_i, Y_i), \dots)$  是泄漏的带标签的样本；而在聚类攻击时， $D = (\dots, X_i, \dots)$  是泄漏的无标签的样本。而  $\mathcal{A}(D; null)$  相当于攻击者在没有任何底

部模型信息，仅有泄露数据的情况下产生的完全模型，即使用带标签数据/无标签数据从零开始训练模型（Train from Scratch）。

**定义 4.2.2 (完美保护)** 对于某种攻击  $\mathcal{A}$ ，如果底部模型  $M_b$  满足底部模型优势  $Adv(M_b, \mathcal{A}) \leq 0$ ，则称  $M_b$  对  $\mathcal{A}$  产生了完美保护，因为  $M_b$  的存在对攻击效果的提升没有任何作用。

基于以上定义，本文目标可以描述为训练一个拆分学习模型  $(M_b, M_t)$  使得：

- 模型本身的准确率越高越好。
- 底部模型针对于微调攻击和聚类攻击的底部模型优势越小越好。

## 4.3 势能损失函数

如上节所述，我们把模型补全攻击考虑为攻击者基于泄露数据的机器学习过程。本节我们研究攻击者在泄露样本上学习得到的顶部模型的泛化误差。我们通过二分类例子表明当拆分层表征分布在决策边界附近时，攻击者的泛化误差较大。同时，数据分布在决策边界时，类间距离减小，类内距离加大，因此聚类攻击的效果也下降。受到静电平衡理论以及库仑定律的启发，我们提出了势能损失，使得拆分层表征满足上述的分布，从而实现对于模型补全攻击的保护。

### 4.3.1 数据分布导致的学习误差

我们从微调攻击的泛化误差以及聚类误差两个角度来说明同类样本分布在决策边界附近时可以使得攻击者的学习误差变大。

#### 4.3.1.1 泛化误差

直观地说，当同类表征分布在决策边界附近时，表明同类表征之间的距离加大，从而使得少量该类别的表征无法代表整个类别的表征，因此会提高攻击者在少量样本上训练顶部模型的泛化误差。下面我们使用一个二分类的例子对其进行说明。

考虑表征空间为一个  $d$  维超球体 ( $d$ -sphere)  $\{\mathbf{x} : \sum_{i=1}^d h_i^2 = 1\}$ ，所有表征被划分为两个类别（正样本和负样本）。令假设集 (Hypothesis Set, 即所有可能的分类函数的集

合) 为所有的半球面(表示正样本)集合:

$$\mathcal{H} = \{h : h(\mathbf{x}) = \text{Sign}(\mathbf{w} \cdot \mathbf{x}), \|\mathbf{w}\|_2 = 1\}. \quad (4-3)$$

不失一般性, 我们假设目标分类函数为  $f(\mathbf{x}) = \text{Sign}(x_1)$ 。同时, 我们对表征的分布做出如下假设:

- 表征分布的概率密度仅仅和其坐标的第一维有关, 也就是仅和  $x_1$  有关, 而在其他维度上呈现出各向同性 (Isotropic) 性质。
- 给定一批正样本数据  $S = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ , 学习算法简单地产生其归一化 (Normalized) 的平均值作为学到的分类模型的权重  $\mathbf{w}$ 。即:

$$f^{(S)}(\mathbf{x}) = \text{Sign} \left[ \mathbf{x} \cdot \sum_{i=1}^n \mathbf{x}_i / \left\| \sum_{i=1}^n \mathbf{x}_i \right\|_2 \right]. \quad (4-4)$$

考虑当学习到的参数  $\mathbf{w} = \sum_{i=1}^n \mathbf{x}_i / \left\| \sum_{i=1}^n \mathbf{x}_i \right\|_2$  与目标分类模型的参数  $\mathbf{e}_1$  有一点偏差时的情况。由于我们假设表征的分布在  $\mathbf{e}_1$  之外的维度时各向同性的, 我们可以假设  $\mathbf{w}$  位于前两维展开的超平面上, 也就是

$$\mathbf{w} = \mathbf{e}_1 \cos \epsilon + \mathbf{e}_2 \sin \epsilon, \quad (4-5)$$

其中  $\epsilon$  表示  $\mathbf{w}$  和  $\mathbf{e}_1$  之间的一个小角度。此时我们可以得到泛化误差为:

$$\begin{aligned} 1/2 \cdot R[\mathbf{w}] &= \mathbb{E}_{\mathbf{x} \sim S} \text{Sign}(x_1) \cdot I[x_1 \cos \epsilon + x_2 \sin \epsilon \leq 0] \\ &= \int_{\substack{x_1 > 0 \\ x_1 \cos \epsilon + x_2 \sin \epsilon \leq 0 \\ x_1^2 + \dots + x_d^2 = 1}} p(x_1, \dots, x_d) dS \leq \int_{\substack{x_1^2 + \dots + x_d^2 = 1 \\ 0 < x_1 \leq \tan \epsilon}} p(x_1, x_2, \dots, x_d) dS \end{aligned} \quad (4-6)$$

注意到当  $\epsilon \rightarrow 0$  时,  $\tan \epsilon \rightarrow \epsilon$ , 并且令  $p_1$  为表征在  $\mathbf{e}_1$  上的边缘概率密度, 即  $p_1(x_1) = \int_{x_1^2 + \dots + x_d^2 = 1} p(x_1, \dots, x_d) dx_2 \cdots dx_d$ , 上式右侧可以进一步化为

$$\int_{\substack{x_1^2 + \dots + x_d^2 = 1 \\ 0 < x_1 \leq \tan \epsilon}} p(x_1, x_2, \dots, x_d) dS \approx \int_{x_1=0}^{\epsilon} p_1(x_1) dx_1 \approx \epsilon p_1(0). \quad (4-7)$$

从上式可以看出, 当  $\epsilon$  很小时, 泛化误差的上界与  $p_1(0)$  呈近似线性关系。因为  $p_1(0)$  就是正样本和负样本交界处的表征边缘概率密度, 所以也可以说, 此时泛化误差的大小与表征在决策边界附近的概率密度成比例的。越多的表征分布在决策边界上, 则分类函数的误差导致的泛化误差就越大。

在上面的讨论中，我们假设  $\epsilon$  是一个固定的小量。现在我们来讨论  $\epsilon$  与表征分布的关系。对于任何一个随机变量  $X$ ，如果  $X_1, \dots, X_m$  是  $m$  个独立的采样，则有：

$$\mathbb{E} \left[ \left( \frac{1}{m} \sum_{i=1}^m h_i - \mathbb{E}[X] \right)^2 \right] = \frac{1}{m} \mathbb{E} [(X - \mathbb{E}X)^2]. \quad (4-8)$$

也就是说，如果随机变量的分布远离其均值，则多次采样后得到的采样均值相对于实际分布的均值的误差就越大。虽然在先前的讨论中，我们假定表征空间是一个超球面，但是上述结论依然是近似成立的，也就是  $\mathbb{E}[\epsilon^2] \propto \mathbb{E}[(X - \mathbb{E}X)^2]$ 。为了使得决策区域误差  $\epsilon$  尽可能变大，同一类别的表征应该尽量分布在远离其分布均值的区域。同时我们也可以注意到，离某一类别表征分布均值最远的区域也正是该类别的决策边界。已有的从数据分布直接推导泛化误差上界的研究与上文的分析较为类似<sup>[137]</sup>，该研究使用数据的类内距离和类间距离对泛化误差上界进行估计。

#### 4.3.1.2 聚类误差

当表征分布于决策边界附近时，对其进行聚类也将变得困难。因为聚类的基本思想是把距离较近的一群数据点归入一个簇中，也就是说，聚类之后的每个类别里面的数据点应该彼此靠近<sup>[138]</sup>。但是通过让表征分布到决策边界上，与其分布均值远离，同类表征之间的距离变得更大了，而不同类表征之间的距离变得更小了。这便与聚类的基本要求相违背，因此让表征分布于决策边界的做法很显然可以降低对其聚类的效果。

#### 4.3.1.3 总结

综上所述，让表征分布到自身类别的决策边界，可以同时提高对其进行微调或聚类的学习误差，其原因可以总结为以下三点：

- 学习得到的决策区域的误差  $\epsilon$  变大。
- 同样的决策区域误差  $\epsilon$  导致的模型泛化误差变大。
- 类间距离变小，类内距离变大。

我们将上述的分析直观地在图 4.2 中显示。

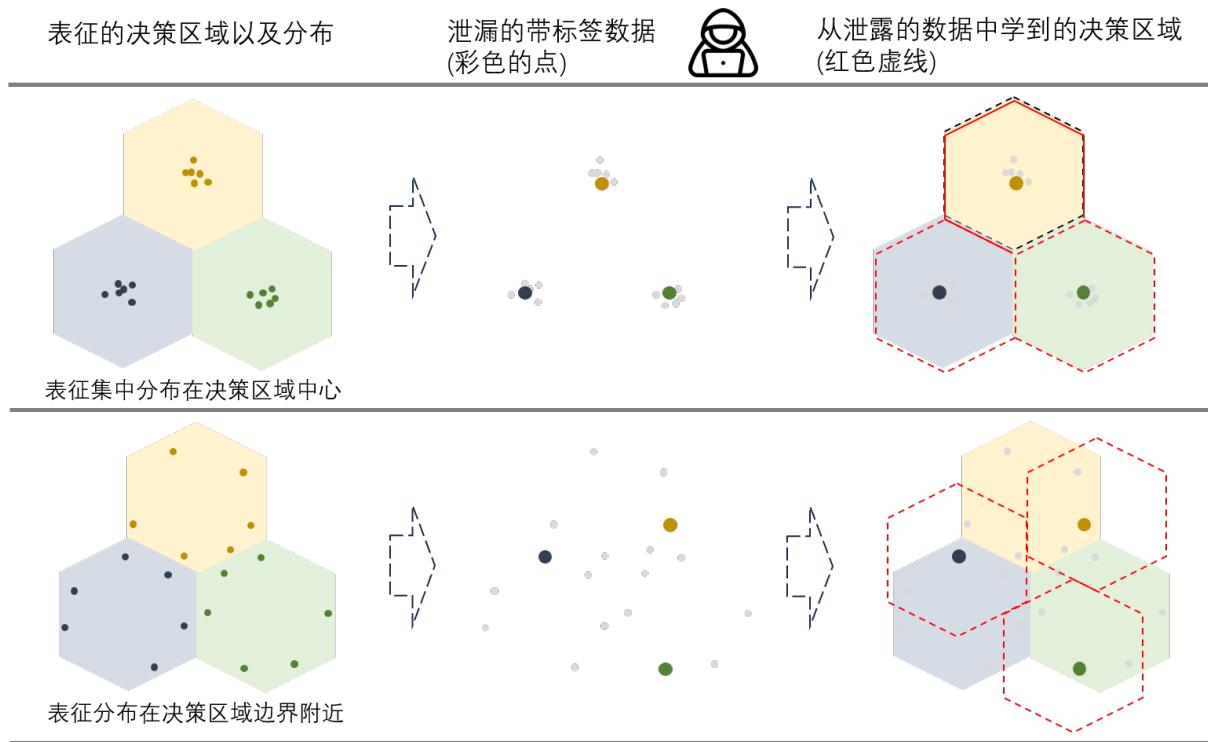


图 4.2 隐层表征分布导致的攻击者误差示意图

### 4.3.2 势能损失函数定义

物理学中的静电平衡 (Electrostatic Equilibrium) 现象告诉我们，当一个导体建立静电平衡时，其所有净电荷 (Net Charge) 将会分布在导体表面<sup>[139]</sup>。这个现象的一部分原因是库仑定律 (Coulomb's Law)，即：同性电荷之间相互排斥，而异性电荷之间相互吸引。受到该物理现象的启发，我们可以把相同类别的表征看作同性电荷，使得它们之间存在互相排斥对方的斥力。因此，这些同类别的表征就会相互远离直到决策区域的边界处。

库仑定律的公式为

$$\mathbf{F} = kq_1q_2(\mathbf{r}_1 - \mathbf{r}_2)/\|\mathbf{r}_1 - \mathbf{r}_2\|_2^3, \quad (4-9)$$

其中， $k$  表示一个常数， $q_1, q_2$  时带符号的电荷大小（正电荷为正号，负电荷为负号）， $\mathbf{r}_1, \mathbf{r}_2$  表示两个电荷的位置， $\|\cdot\|$  表示欧几里得范数， $\mathbf{F}$  则表示第二个电荷对第一个电荷产生的力（库仑力）的大小。我们忽略常数以及电荷的大小，只考虑同性电荷，则可以把式 4-9 化为  $\mathbf{F} = (\mathbf{r}_1 - \mathbf{r}_2)/\|\mathbf{r}_1 - \mathbf{r}_2\|_2^3$ 。由于库仑力也可以表示为电势能的梯度，我们可以进一步将其表示为梯度的形式  $F = \nabla_{\mathbf{r}_1} 1/\|\mathbf{r}_1 - \mathbf{r}_2\|_2$ 。这种表示方法与深度学习

中所使用的梯度下降方法不谋而合。基于以上分析，我们定义势能损失函数为：

$$L_{pe} = \sum_{c \in \mathcal{C}} \sum_{\mathbf{h} \in H_c} \sum_{\mathbf{h}' \in H_c, \mathbf{h}' \neq \mathbf{h}} \frac{1}{\|\mathbf{h} - \mathbf{h}'\|_2}, \quad (4-10)$$

其中  $\mathcal{C}$  表示标签的集合， $H_c$  表示类别为  $c$  的表征集合。

通过把  $L_{pe}$  加入损失函数中，在拆分模型的训练过程中，同类别的表征互相之间会排斥并且逐渐移动到该类别的决策区域边界处。在三维的情况下，汤姆森定理（Thomson's Theorem）证明电势能最小的情况对应于电荷全部分布在导体的表面，也就是表征的概率质量全部在决策边界<sup>[139]</sup>。但是考虑到拆分层的表征空间的维度一般远大于 3 维，因此其概率质量并不能完全集中在决策边界。但是我们依然能证明，在势能损失函数最小的情况下，决策边界处必然有概率质量分布。

**定理 4.3.1 (决策边界的表征分布非零)** 考虑一个  $d$  维有界区域  $\Omega \subset \mathbb{R}^d$  以及定义在  $\Omega$  内部的概率密度泛函  $f^*$  为最小化势能损失的概率密度：

$$f^* = \underset{f}{\operatorname{argmin}} \text{PE}(f) = \underset{f}{\operatorname{argmin}} \int_{\mathbf{x} \in \Omega} \int_{\mathbf{y} \in \Omega} \frac{f(\mathbf{x})f(\mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|_2} dU dV, \quad (4-11)$$

其中， $f$  表示概率密度函数且满足  $f \geq 0$ （非负性）以及  $\int_{\Omega} f(\mathbf{x}) dV = 1$ （归一性）。令  $\Delta_{\epsilon}\Omega = \{\mathbf{x} : \mathbf{x} + \epsilon\mathbf{r} \notin \Omega, \mathbf{x} \in \Omega, \|\mathbf{r}\|_2 = 1\}$  表示距离  $\Omega$  边界距离小于  $\epsilon$  的区域，那么  $f^*$  满足对于任意  $\epsilon > 0$ ，有  $\int_{\mathbf{x} \in \Delta_{\epsilon}\Omega} f^*(\mathbf{x}) dV > 0$ 。

**证明 4.3.1** 为了证明的方便，此处我们用  $|\cdot|$  来表示欧几里得范数。首先考虑中心在  $\mathbf{x}_0$ ，半径为  $r$  的球  $B_r(\mathbf{x}_0)$ 。令

$$r_0 = \inf_r \left\{ \int_{\mathbf{y} \in \Omega - B_r(\mathbf{x}_0)} f(\mathbf{y}) dV = 0 \right\}, \quad (4-12)$$

即： $f(x)$  在  $B_{r_0}(\mathbf{x}_0)$  之外几乎处处为 0，也就是说  $B_0 = B_{r_0}(\mathbf{x}_0)$  是以  $\mathbf{x}_0$  为中心且包含了几乎所有概率质量的最小半径球。因此我们可以找到一个靠近  $B_0$  边界的小球  $B_1 = B_{\epsilon/4}(\mathbf{x}_0 + \mathbf{r}')$  使得  $\int_{\mathbf{y} \in B_1} f(\mathbf{y}) dV = \delta$ ，其中  $\delta > 0$ ,  $|\mathbf{r}'| = r_0 - \epsilon/4$ 。也就是说， $B_1$  内部的概率质量非零。

现在我们考虑将  $f$  在  $B_1$  里面的概率质量移动到  $B_2 = B_{\epsilon/4}(\mathbf{x}_0 + \mathbf{r}'(1 + \frac{3\epsilon}{4|\mathbf{r}'|})) = B_{\epsilon/4}(\mathbf{x}_0 + \mathbf{r}'')$  中。由于  $B_2 \subset \Delta_{\epsilon}\Omega$  且  $B_1 \subset \Omega - \Delta_{\epsilon}\Omega$ ，所以上述的移动是可行的。我们

把概率质量移动后的概率密度函数记作

$$f'(\mathbf{t}) = \begin{cases} f(\mathbf{t} - \mathbf{r}'' + \mathbf{r}') & \text{当 } \mathbf{t} \in B_2, \\ 0 & \text{当 } \mathbf{t} \in B_1, \\ f(t) & \text{其他.} \end{cases} \quad (4-13)$$

此时我们计算  $f'$  和  $f$  的势能差异 (把  $\Omega - B_1 - B_2$  记作  $\Omega'$ ):

$$\begin{aligned} PE(f) - PE(f') &= \int_{\mathbf{x} \in \Omega} \int_{\mathbf{y} \in \Omega} \frac{f(\mathbf{x})f(\mathbf{y}) - f'(\mathbf{x})f'(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} dU dV \\ &= \left( \int_{\mathbf{x} \in B_1 \cup B_2} \int_{\mathbf{y} \in \Omega'} \frac{f(\mathbf{x})f(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} dU dV + \int_{\mathbf{x} \in \Omega'} \int_{\mathbf{y} \in B_1 \cup B_2} \frac{f(\mathbf{x})f(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} dU dV + \int_{\mathbf{x} \in B_1 \cup B_2} \int_{\mathbf{y} \in B_1 \cup B_2} \frac{f(\mathbf{x})f(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} dU dV \right) \frac{f(\mathbf{x})f(\mathbf{y}) - f'(\mathbf{x})f'(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} dU dV \\ &= 2 \int_{\mathbf{x} \in B_1} \int_{\mathbf{y} \in \Omega'} \frac{f(\mathbf{x})f(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} dU dV - 2 \int_{\mathbf{x} \in B_2} \int_{\mathbf{y} \in \Omega'} \frac{f'(\mathbf{x})f(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} dU dV + \\ &\quad \int_{\mathbf{x} \in B_1} \int_{\mathbf{y} \in B_1} \frac{f(\mathbf{x})f(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} dU dV - \int_{\mathbf{x} \in B_2} \int_{\mathbf{y} \in B_2} \frac{f'(\mathbf{x})f'(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} dU dV. \end{aligned} \quad (4-14)$$

注意到  $f'$  是把  $f$  在  $B_1$  处的值直接移动到了  $B_2$ , 因此上式最后两项可以相消。此时可以把式 4-14 写成:

$$\frac{1}{2}[PE(f) - PE(f')] = \int_{\mathbf{x} \in B_1} \int_{\mathbf{y} \in \Omega'} \frac{f(\mathbf{x})f(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} dU dV - \int_{\mathbf{x} \in B_2} \int_{\mathbf{y} \in \Omega'} \frac{f'(\mathbf{x})f(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} dU dV. \quad (4-15)$$

令  $g(\mathbf{s}) = f(\mathbf{x} - \mathbf{r}') = f'(\mathbf{x} - \mathbf{r}'')$ ,  $\mathbf{s} \in B_{\epsilon/4}(0)$ , 则可以将上式进一步简化为

$$\int_{\mathbf{x} \in B_{\epsilon/4}(0)} \int_{\mathbf{y} \in \Omega'} g(\mathbf{x})f(\mathbf{y}) \left[ \frac{1}{|\mathbf{x} + \mathbf{r}' - \mathbf{y}|} - \frac{1}{|\mathbf{x} + \mathbf{r}'' - \mathbf{y}|} \right] dU dV. \quad (4-16)$$

由于  $\mathbf{r}'$  和  $\mathbf{r}''$  方向一致 (我们将其对应的单位向量记作  $\mathbf{e}_r$ ), 因此我们可以将  $bvecx$  和  $\mathbf{y}$  分解为与  $\mathbf{e}_r$  相同的方向和与其正交的方向:

$$\begin{cases} \mathbf{x}_r = (\mathbf{x} \cdot \mathbf{e}_r)\mathbf{e}_r \\ \mathbf{x}_v = \mathbf{x} - \mathbf{x}_r \end{cases} \quad \text{以及} \quad \begin{cases} \mathbf{y}_r = (\mathbf{y} \cdot \mathbf{e}_r)\mathbf{e}_r \\ \mathbf{y}_v = \mathbf{y} - \mathbf{y}_r \end{cases} \quad (4-17)$$

于是我们有:

$$\begin{cases} |\mathbf{x} + \mathbf{r}' - \mathbf{y}|^2 = |\mathbf{x}_r + \mathbf{y}_r|^2 + |\mathbf{x}_r + \mathbf{r}' - \mathbf{y}_r|^2 = |\mathbf{x}_v + \mathbf{y}_v|^2 + |\mathbf{x}_r + \mathbf{r}' - \mathbf{y}_r|^2, \\ |\mathbf{x} + \mathbf{r}'' - \mathbf{y}|^2 = |\mathbf{x}_v + \mathbf{y}_v|^2 + |\mathbf{x}_r + \mathbf{r}'' - \mathbf{y}_r|^2 = |\mathbf{x}_r + \mathbf{y}_r|^2 + |\mathbf{x}_r + \mathbf{r}'' - \mathbf{y}_r|^2, \end{cases} \quad (4-18)$$

上式最后一项的左侧是相等的, 因此只需考虑右侧。我们只需考虑考虑如下两种情况:

- $|\mathbf{x}_r + \mathbf{r}'| \geq |\mathbf{y}_r|$ : 同时注意到  $|\mathbf{r}''| > |\mathbf{r}'|$ , 因此  $|\mathbf{x} + \mathbf{r}'' - \mathbf{y}| < |\mathbf{x} + \mathbf{r}' - \mathbf{y}|$ 。
- $|\mathbf{x}_r + \mathbf{r}'| < |\mathbf{y}_r|$ : 因为  $|\mathbf{y}_r| < r_0$ ,  $|\mathbf{x}_r + \mathbf{r}'| \geq r_0 - \epsilon/2$ , 以及  $|\mathbf{x}_r + \mathbf{r}''| > r_0 + \epsilon/2$ , 因此有

$$|\mathbf{x}_r + \mathbf{r}'' - \mathbf{y}_r| > \epsilon/2 > |\mathbf{y}_r - (\mathbf{x}_r + \mathbf{r}')|. \quad (4-19)$$

综上, 当  $x \in B_{\epsilon/4}(0), y \in \Omega'$  时, 就有

$$|\mathbf{x} + \mathbf{r}' - \mathbf{y}| < |\mathbf{x} + \mathbf{r}'' - \mathbf{y}|, \quad (4-20)$$

因此,

$$1/|\mathbf{x} + \mathbf{r}' - \mathbf{y}| - 1/|\mathbf{x} + \mathbf{r}'' - \mathbf{y}| > 0. \quad (4-21)$$

带入式 4-16, 可以得到  $PE(f') < PE(f)$ 。至此定理 4.3.1 得到证明。

定理 4.3.1 表明, 当势能损失最小时, 表征在决策边界上的分布必然不为 0, 否则我们便可以通过移动一小部分的概率质量到边界处使得势能损失更小, 与势能损失最小的前提相矛盾。

**加入层归一化:** 势能损失方法有一个隐含条件: 某一类别的表征的决策区域需要是有界的, 并且要与其他类别的表征决策区域相邻。否则, 势能损失可以把同类表征推向无穷远处或是与其他类别决策区域不相邻的“边界”, 使得其保护能力降低。这要求表征空间是一个有界 (Bounded) 并且无边界 (Borderless) 的流形。我们简单地通过层归一化 (Layer Normalization) 来实现该要求。经过层归一化后, 拆分层的表征被投影到  $d$  维超球体上。为了保持一致性, 我们也需要改变其距离的计算方式, 将欧氏距离 (Euclidean Distance) 转化为超球面上的测地距离 (Geodesic Distance), 此时势能损失函数被修改为:

$$L_{pe} = \sum_{c \in \mathcal{C}} \sum_{\mathbf{h} \in H_c} \sum_{\mathbf{h}' \in H_c, \mathbf{h}' \neq \mathbf{h}} \frac{1}{\arccos \langle \mathbf{h}, \mathbf{h}' \rangle}. \quad (4-22)$$

最终, 在模型的训练过程中, 我们把势能损失加入到原有的损失函数 (如交叉熵分类损失) 中, 并设置权重为  $\alpha$ 。总的损失函数为  $L' = L + \alpha L_{pe}$ 。

### 4.3.3 与距离相关性的关系

已有研究采用了距离相关性将拆分层表征对输入特征<sup>[26]</sup>或预测标签<sup>[41]</sup>进行解耦, 从而实现中拆分学习中保护数据输入特征或标签的目的。我们在这里讨论对标签进行保

护的情况，此时距离相关性损失可以表示为

$$L_{\text{dcor}} = \sum_{i=1}^n d_{i,j} d'_{i,j} / \sqrt{\sum_{i,j=1}^n d_{i,j}^2 \sum_{i,j=1}^n d'^2_{i,j}}, \quad (4-23)$$

其中， $d_{i,j}$  表示样本  $i$  和样本  $j$  的表征之间的双中心 (Doubly-Centered) 距离，即：

$$d_{i,j} = \|\mathbf{h}_i - \mathbf{h}_j\|_2 - \mathbb{E}_{\mathbf{h}' \in H} \|\mathbf{h}_i - \mathbf{h}'\|_2 - \mathbb{E}_{\mathbf{h}' \in H} \|\mathbf{h}' - \mathbf{h}_j\|_2 + \mathbb{E}_{\mathbf{h}', \mathbf{h}'' \in H} \|\mathbf{h}' - \mathbf{h}''\|_2. \quad (4-24)$$

类似地， $d'_{i,j}$  则是样本  $i$  和样本  $j$  的标签的双中心距离，以独热 (One-Hot) 标签为例，则  $d_{i,i} = 0, d_{i,j \neq i} = \sqrt{2}$ 。将其带入式 4-23，提取出其分母，得到

$$\sum_{\substack{c,c' \in \mathcal{C} \\ c \neq c'}} \sum_{\substack{\mathbf{h} \in H_c \\ \mathbf{h}' \in H'_c}} \sqrt{2} \left( \|\mathbf{h} - \mathbf{h}'\|_2 - \overline{\|\mathbf{h} - \cdot\|_2} - \overline{\|\cdot - \mathbf{h}'\|_2} + \overline{\|\cdot - \cdot\|_2}, \right) \quad (4-25)$$

其中，我们用  $\overline{\|\mathbf{h} - \cdot\|}$  表示  $\mathbb{E}_{\mathbf{h}' \in H} \|\mathbf{h} - \mathbf{h}'\|_2$ ， $\overline{\|\cdot - \cdot\|}$  表示  $\mathbb{E}_{\mathbf{h}, \mathbf{h}' \in H} \|\mathbf{h} - \mathbf{h}'\|_2$ ， $H$  表示当前批样本表征集合， $H_c$  表示当前批样本中类别为  $c$  的表征集合。可以看出最小化距离相关性损失类似于最小化表征的类间距离，而本文提出的势能损失则旨在最大化类内距离，两者之间有一定的相似性。然而直观地来讲，不同类别的决策边界分布在不同的方向上，导致距离相关性的计算有更高的误差且难以优化。[??](#)中的实验结果也表明势能损失相比于距离相关性更加优越。

#### 4.3.4 输入特征的隐私保护

尽管势能损失并未直接保护输入特征的隐私，但是间接地对其进行了保护。这是因为在拆分学习中，可以通过拆分层的选择来调节输入特征和标签的隐私泄露程度。具体而言，拆分层靠近模型输入，则输入特征隐私泄露加大；而拆分层靠近模型输出，则输出特征隐私泄露加大<sup>[25]</sup>。而通过势能损失函数，我们可以将拆分层设置到更靠近输出的位置，此时特征的隐私泄露自然也就减少了。对于类似卷积神经网络的模型，其中间层会保留大量的输入数据的信息，因此通过势能损失，把拆分层设置为最后一层，可以极大提高对于输入特征的保护，同时也保护标签的隐私。

### 4.4 实验分析

为了验证本文所提出的势能损失函数对于模型补全攻击的有效性，本文在多个数据集上进行了实验，并且与基于距离相关性<sup>[26,41]</sup>以及标签差分隐私 (Label Differential

Privacy)<sup>[140]</sup>的方法进行对比。各个数据集和对应的模型介绍如下：

- MNIST<sup>[141]</sup>: 一个常用的手写数字数据集，包含了 7 万张  $28 \times 28$  大小的黑白图片，根据手写数字的值划分为 10 类 (0-9)。对应的模型是一个隐层大小为 [128, 32] 的全连接神经网络，激活函数采用 LeakyReLU。
- Fashion-MNIST<sup>[142]</sup>: 一个包含了 10 个类别的图片数据集，包含了 7 万张服饰相关的  $28 \times 28$  大小的黑白图片。对应的模型是一个双层卷积加上双层全连接的模型，卷积层的通道数分别为 [32, 64]，卷积核大小分别为 [5, 3]，每一个卷积层后面跟随 LeakyReLU 激活函数和大小为 2 的最大池化层。全连接层的维度为 2304 (卷积层提供的输入) -128-10 (输出)。
- CIFAR-10<sup>cifar</sup>: 一个包含了 10 个类别的 6 万张图片的彩色图片数据集，图片大小为  $3 \times 32 \times 32$ 。对应的模型是 ResNet-20 模型。
- DBpedia<sup>[127]</sup>: 一个文本分类数据集，包含了多级的标签，我们选择最上级的标签，共有 9 个类别，总共有大约 34 万条文本。我们采用 TextCNN 模型<sup>[128]</sup>进行分类，卷积核的大小为 (3, 4, 5)，并且使用 Glove 预训练词向量<sup>[129]</sup>。拆分层设置为最后的隐层，大小为 300。

攻击者采用微调攻击和聚类攻击两种方法对训练完毕的拆分学习模型进行攻击。对于聚类攻击，我们采用经典的  $k$ -平均 ( $k$ -Means) 聚类算法<sup><empty citation></sup>，并且给定总类别个数。

所有实验都在装配有 NVIDIA RTX3090 GPU 的服务器上进行，并使用不同的随机种子重复 5 次以上。对于所有的任务，我们最多训练 100 轮模型，并在 90-100 轮中采用“早停 (Early Stopping) 策略获取验证集最佳的模型，以保证损失函数被充分优化。对于聚类攻击，我们采用 Scikit-Learn 库中的  $k$ -平均聚类算法，并采用默认参数。我们将势能损失的权重从 0.25 变化到 32，每次翻倍；类似地，对于距离相关性损失，我们也把权重从 1 变化到 32，每次翻倍；对于标签差分隐私方法，我们将翻转的标签比例从 0.01 提高到 0.16，每次翻倍。我们默认将拆分层选为模型的最后一层，因为其最贴近模型的输出，但我们也汇报了拆分层选为其他层的实验结果。

#### 4.4.1 微调攻击

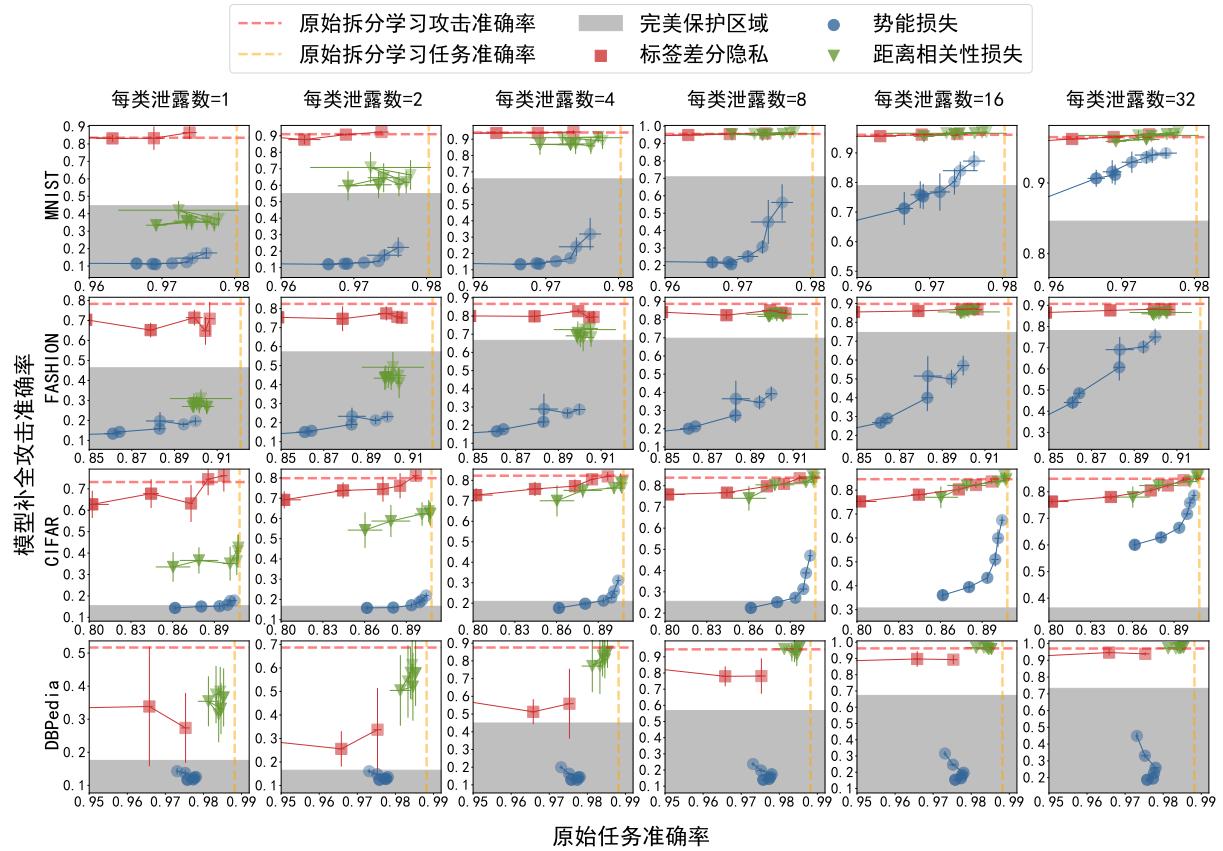


图 4.3 拆分层为最后一层时原始任务准确率与模型补全攻击准确率对比图

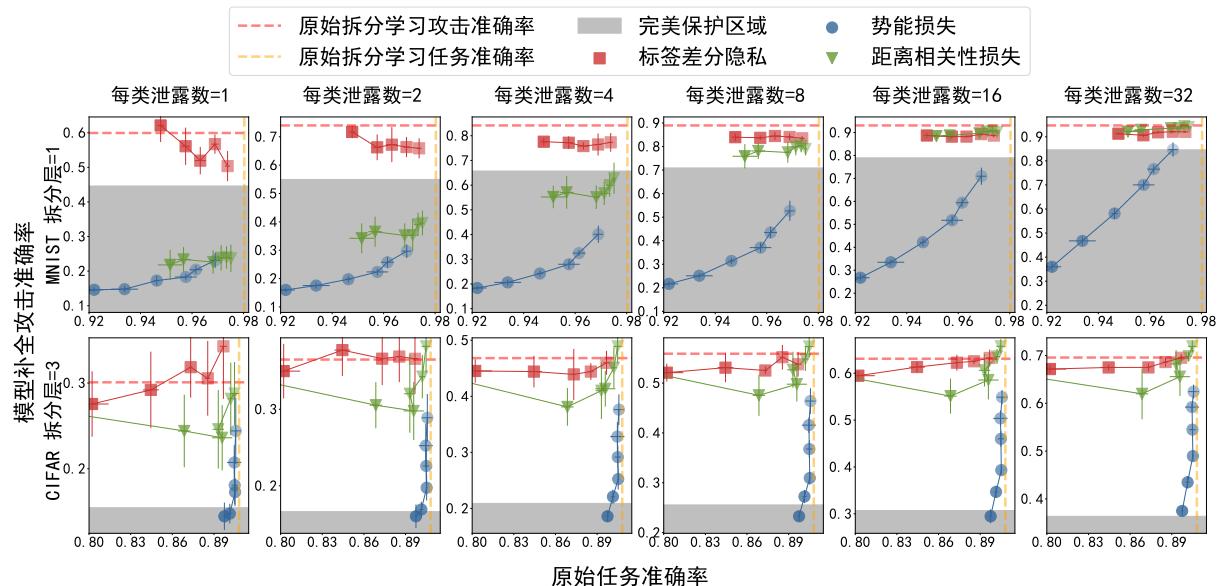


图 4.4 拆分层为中间层时原始任务准确率与模型补全攻击准确率对比图

我们测量了模型在测试集上的准确率，以及攻击者在泄漏的带标签数据上进行微调

攻击得到的模型的准确率。我们将拆分层为最后一层的结果和拆分层为中间其他层的结果分别汇报在图 4.3 和图 4.4 中。在拆分其他层的实验中，我们将 MNIST 任务的全连接神经网络的拆分层设置在第一个隐层处，将 CIFAR-10 任务的拆分层设置在第二个残差块（Residual Block）之后。

图中的数据点的深浅对应实验中损失函数系数或标签翻转概率，颜色越深代表超参数的设定更加偏向于保护隐私，越浅则代表超参数的设定偏向于保留原始任务的准确率。落在灰色区域的数据表示训练好的拆分学习模型实现了完美保护，也就是底部模型或隐层表征对于拆分者没有任何额外作用，拆分者对其实行攻击的效果未达到在泄漏的数据集上直接训练的效果。如果数据点落在图的左下角，说明该实验降低了攻击准确率的同时保持了原始任务的准确率。

实验结果表明，如果不对拆分学习加以防护，则攻击者可以获得很高的攻击效果。标签差分隐私、距离相关性损失和势能损失都能一定程度上降低攻击准确率，且势能损失的数据点大多落在对比方法的右下方，表明势能损失取得了更好的保护效果的同时又有更高的原始任务准确率。同时，势能损失数据点连成的曲线较为平滑，可以很清楚展现出通过改变损失系数来权衡原始任务准确率和攻击效果的过程。与此同时，距离相关性损失的数据点则难以连成曲线，表明其效果难以控制。此外，势能损失有最多的数据点落在完美保护区域，而其他方法落在完美保护区域的数据点则很少。综上，对于微调攻击，在同等的原始任务准确率下，势能损失的防御效果显著高于其他方法。

#### 4.4.2 聚类攻击

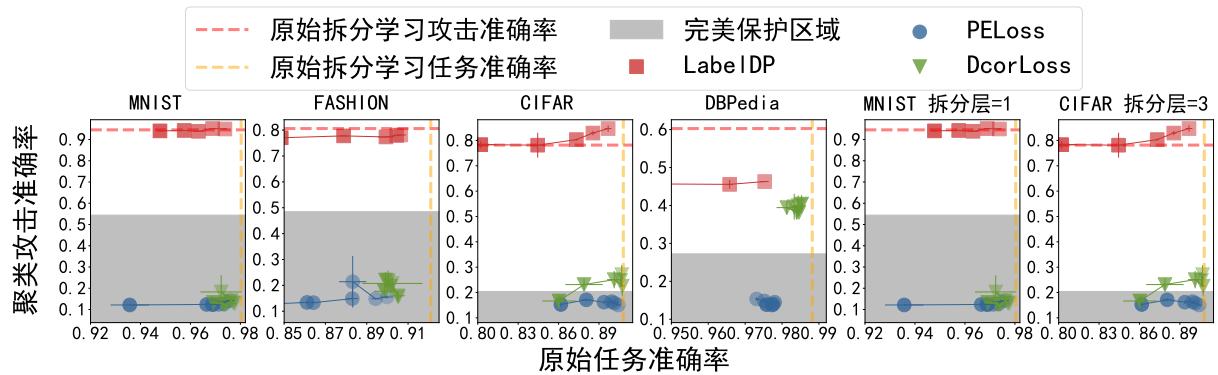


图 4.5 拆分层为中间层时原始任务准确率与模型补全攻击准确率对比图

我们测量了模型在测试集上的准确率，以及攻击者无标签数据（整个测试集，均

超过 1 万个样本) 上进行聚类攻击得到的模型的准确率。全部结果汇报在图 4.5 中。由于聚类结果本身并不与实际的标签类别相对应, 我们通过匈牙利匹配算法 (Hungarian Matching Algorithm) 找到准确率最高的类别映射关系并汇报该准确率。该准确率计算公式可以表示为:

$$\max_{k_1, \dots, k_C \in \text{Permutation}(1, \dots, C)} \frac{\sum_{i=1}^C c_{i, k_i}}{\sum_{i,j=1}^C c_{i, j}}, \quad (4-26)$$

其中  $C$  表示样本总数,  $c_{i,j}$  表示第  $i$  个簇中实际标签是  $j$  的样本数量,  $1 < i, j \leq C$ 。

实验结果表明, 势能损失在所有的情况下均实现了完美保护, 大幅领先于其他对比方法。也就是说攻击者直接在泄漏的无标签数据上聚类得到的效果高于在拆分层表征上聚类得到的效果。而标签差分隐私在对抗聚类攻击上效果不佳, 相比于原始的拆分学习, 未呈现出明显的保护作用。

#### 4.4.3 在拆分层之前攻击

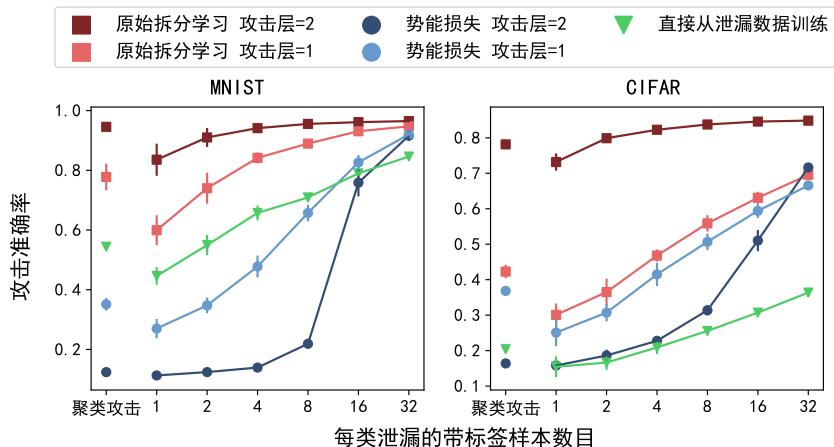


图 4.6 拆分层为中间层时原始任务准确率与模型补全攻击准确率对比图

在之前的模型补全攻击实验中, 我们假设攻击者使用了拆分层的表征进行攻击。如果攻击者可以获取并操纵整个底部模型, 则他也可以在拆分层之前的层进行攻击, 也就是获取拆分层之前的隐层表征来微调模型或聚类。因此, 我们在 MNIST 和 CIFAR 任务中考察了攻击者在拆分层之前的层攻击的情况。我们将拆分层设置为模型最后一层之前, 然后类似地, 将 MNIST 任务的攻击层设置在第一个隐层处, 将 CIFAR-10 任务的攻击层设置在第二个残差块 (Residual Block) 之后。对于该实验, 势能损失的系数被选取为  $\alpha = 4$ 。我们在图 4.6 呈现在拆分层之前攻击的实验结果。

可以看到，在拆分层之前攻击可以让攻击效果有所提升，但是依然比无保护的拆分学习的攻击效果要低很多。这是由于势能损失在训练过程中作用于拆分层的表征上，因此拆分层之前的层的表征分布可能与标签关联性更大。

#### 4.4.4 拆分层维度的影响

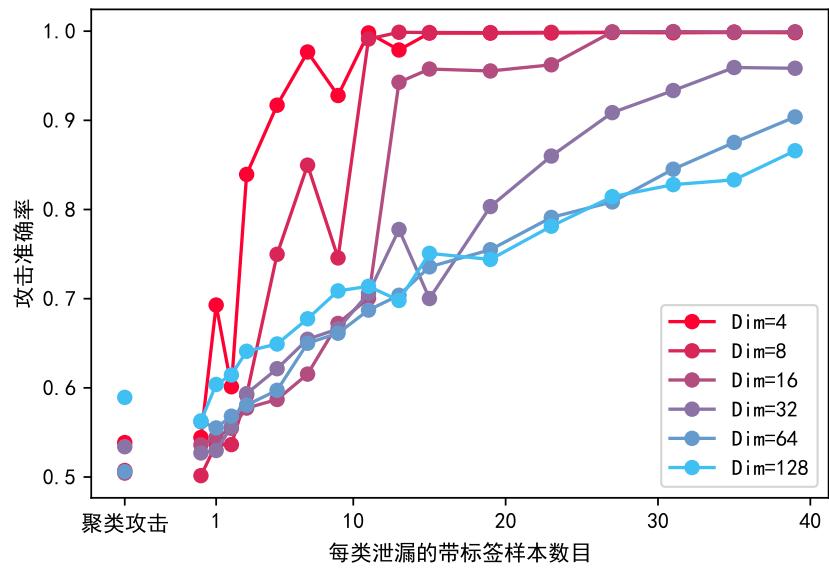


图 4.7 拆分层为中间层时原始任务准确率与模型补全攻击准确率对比图

为了研究拆分层的维度对于攻击效果的影响，我们对 MNIST 数据集的 01 子集上（只保留 0 和 1 类别的样本）进行了实验，其结果被汇报在图 4.7 中。此时我们设置势能损失的系数  $\alpha = 1$ 。

从图中可以看出，较高的拆分层维度大体上让微调攻击变得更难了，特别在每类样本泄露数较多的情况下。例如，拆分层仅仅为 4 维的时候，只需要每个类别 10 个样本就可以让攻击准确率达到 100；而拆分层维度为 128 时，每类泄露 40 个样本后攻击准确率也不到 90%。这与我们的直接相对应，因为对更高维度的数据进行分类时，自然需要更多的样本。

作为对比，在聚类攻击时，更高的拆分层维度并未降低攻击的效果，甚至反倒有所提高。这可能与聚类攻击时样本数量庞大有关。

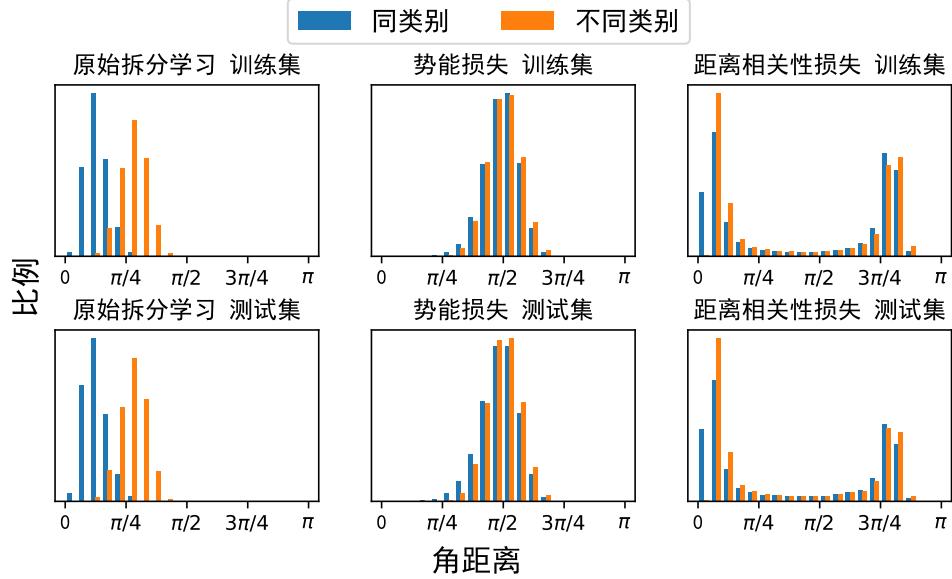


图 4.8 同类别和不同类别的隐层表征之间的角距离分布

#### 4.4.5 拆分层表征分布分析

为了分析拆分层表征的分布，我们在 MNIST 数据集上计算了同类别表征和不同类别的表征的角距离的分布，并且呈现在图 4.9 中。对于势能损失和距离相关性损失，我们分别把系数设置为  $\alpha = 1$  和  $\alpha = 0.5$ ，这两者的原始任务效果基本相同。图中可以看出，无论是否是同类别，抑或是属于训练集或者测试集，势能损失方法的样本表征间的角距离都接近  $\pi/2$ ，也就是表征之间相互正交。对于距离相关性损失，角距离则呈现出一个双峰分布，峰值靠近 0 和  $\pi$ 。并且同类别样本表征对的角距离在靠近 0 的分布明显增加，导致保护效果变差。而原始拆分学习中，同类别表征之间的角距离则明显小于不同类别表征之间的角距离，因此攻击准确率较高。

为了更直观地呈现拆分层表征的分布，我们使用了 t-SNE 降维<sup>[143]</sup>对拆分层表征进行可视化图 4.9。可以看到，原始拆分学习的拆分层表征呈现出明显的聚类性质，各个类别之间的表征可以很容易地被划分开。而加入了势能损失后，不同类别的表征的分布的 t-SNE 投影融为一体，难以区分。而距离相关性损失则呈现出了较为奇怪的分布，其分布被分为多个小块，而每个小块内都对应某种类别，且单个类别可能包含多个块。由于同类别表征间仍有一定程度的集聚效应，因此其安全性低于势能损失。

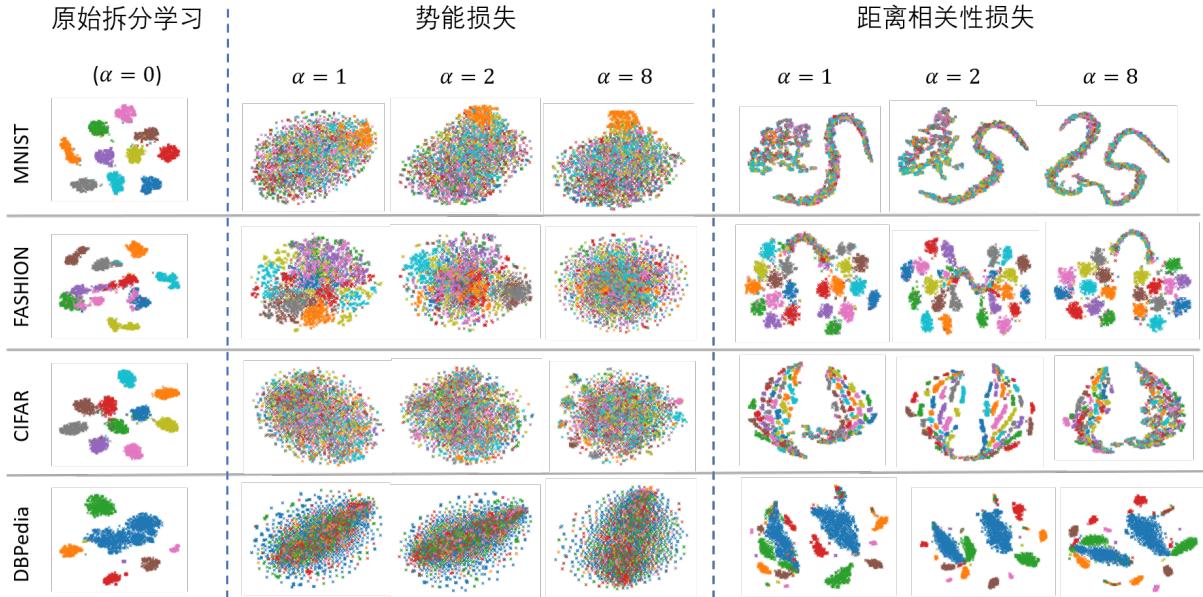


图 4.9 势能损失和距离相关性损失的隐层分布 t-SNE 降维

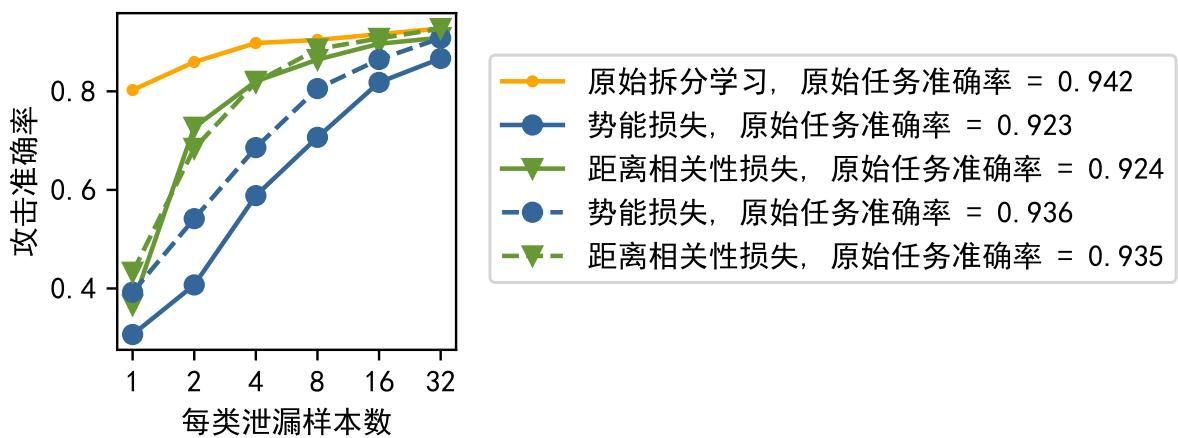


图 4.10 Vegetable Images 数据集下不同泄露样本数和攻击准确率的对比

#### 4.4.6 在大型模型上的补充实验

为了进一步验证势能损失在大规模模型上的效果，我们在蔬菜图像<sup><empty citation></sup>数据集上使用视觉 Transformer (ViT, Vision Transformer) 模型进行了部分实验。蔬菜图像数据集包含 21000 张图片，分为 15 个类别，其图片大小为  $128 \times 128 \times 3$ 。我们将 ViT 模型的分块大小 (Patch Size) 设置为  $16 \times 16$ ，隐层大小为 256，层数为 4，注意力头 (Attention Heads) 个数为 8。实验结果呈现在图 4.10 中，可以看出，势能损失在 ViT 模型上依然显著领先于对比的距离相关性损失。

## 4.5 本章小结

本章节针对拆分学习模型推断过程中的标签隐私泄露以及模型补全攻击问题，提出了受物理学现象启发的势能损失函数。我们将模型补全攻击转化成一个攻击者的有监督或无监督的学习问题，通过分析泛化误差、聚类误差，表明拆分层表征分布在决策区域边缘可以增加攻击者的攻击难度，而势能损失函数恰好可以帮助样本的拆分层表征产生如此分布。实验结果表明，势能损失函数在多个数据集、不同的泄漏标签数量上均优于对比算法，同时保证了原始任务准确率和攻击误差。总之，势能损失降低了拆分学习推断过程中的隐私泄露问题，为拆分学习模型的实际应用提供了有力保护。

## 5 基于秘密分享和随机排列的高效隐私保护机器学习

基于密码学隐私保护机器学习框架面临着非线性激活函数开销大的问题，而拆分学习和其他混合方法往往面临着中间结果产生的隐私泄漏问题。为此，本文混合秘密分享技术和随机排列技术，使用秘密分享协议计算矩阵乘法灯线性运算，同时基于随机排列进行逐元素的非线性激活函数计算，提出了高效易用的隐私保护神经网络框架。本文也对随机排列后的中间结果的隐私泄露程度进行了量化分析，表明随机排列可以有效地保护数据隐私。

### 5.1 研究背景

随着机器学习在各个领域的广泛应用，数据隐私问题日益严重。训练机器学习模型所需的数据可能分散在多个参与方中，比如两家医院分别有患者的 CT 图像数据和患者的心电图数据；银行和电商平台分别有用户的金融数据和消费数据。直接把数据汇聚到一个服务器上进行中心化的模型训练，会直接暴露参与方的数据隐私。类似地，在机器学习模型的部署应用阶段，用户需要将数据发送给服务器来获得模型的预测结果，这也暴露了用户的隐私数据。

为了解决隐私问题，许多隐私保护机器学习的方案被提出。比如（横向）联邦学习（Federated Learning）通过各方训练本地模型再聚合的方式，保护各方本地数据不出域。但是其仅支持数据横向分割（各个参与方有不同的训练样本，但是样本特征列是相同的）的情况。对于更加复杂的数据纵向分割（各个参与方有同样的样本，但是特征列不同的）的情况，或是用户使用模型对自身的输入进行推断的情形，传统的横向联邦学习就无法被应用。本文前面两章所研究的拆分学习技术，在一定程度上可以解决这些问题。但是由于在拆分学习的训练和推断过程中，各方需要交换中间结果，因此其隐私泄漏问题依然存在。尽管前两章提出的方案能够缓解拆分学习隐私泄露的问题，但是拆分学习交换中间结果的特性注定其无法完全保护数据或模型的隐私，因此在某些隐私保护要求较高的领域难以应用。

为了实现计算过程中完全的、可证明的隐私保护，必须采用密码学的安全计算技

术。近年来，许多基于密码学的隐私保护机器学习框架被提出。这些框架一般假设有 2 方或 3 方参与隐私保护的机器学习计算，采用秘密分享、同态加密、混淆电路等密码学基础技术，来实现安全的神经网络的推断和训练。尽管基于密码学的隐私保护机器学习框架拥有可证明的安全性，其也存在着严重的效率问题。相比于中心化的明文计算，密码学方法的通讯和计算开销往往超出几个数量级。由于使用密码学方法进行算数运算（加法、乘法）的技术已经相对成熟，密码学方法的性能瓶颈往往在于神经网络中的激活函数。神经网络中的激活函数是非线性的，其无法用算数运算简单表示，因此需要使用开销相对较高的混淆电路协议<sup>[66]</sup>、GMW 协议<sup><empty citation></sup>或是其他的定制化协议进行安全计算，并以分段线性<sup>[77]</sup>或多项式<sup><empty citation></sup>等方法对非线性函数进行拟合。另一方面，近年来也有一些研究通过将密码学与其他方法结合，设计更加高效的隐私保护机器学习框架<sup>[105-106,108]</sup>。但是其往往类似拆分学习思想，暴露了中间结果的信息，因此存在一定程度的数据或模型的隐私泄露<sup>[25,27,107]</sup>。因此，如何更好地将密码学方法与非密码学方法结合以提高效率，同时尽可能地保持安全性，成为了隐私保护机器学习中亟待解决的问题。

## 5.2 问题描述

本节我们先探究暴露模型中间结果带来的隐私泄露问题，然后定义本章所解决的三方协作的隐私保护机器学习问题。

### 5.2.1 隐层表征的隐私泄露

在拆分学习或部分密码学和拆分学习等其他方法混合的隐私保护机器学习框架中，模型的部分隐层表征会暴露<sup><empty citation></sup>。这些研究对于安全性的论证往往基于中间表征的不可逆性质，也就是在没有辅助信息的情况下，从拆分表征恢复出原始的输入特征是不可能的。比如考虑全连接神经网络的第一层表征  $\mathbf{h} = W\mathbf{x}$ ，其中  $W \in \mathbb{R}^{d_1 \times d_0}$  是第一层的权重（这里忽略偏置项）， $\mathbf{X} \in \mathbb{R}^{d_0}$  表示输入表征。即使攻击者收集到了大量隐层

表征  $H = (\mathbf{h}_1, \dots, \mathbf{h}_n)$ , 只需要隐层表征维度  $d_1$  小于输入特征维度  $d_0$ , 线性方程组

$$\begin{cases} \mathbf{h}_1 = W\mathbf{x}_1 \\ \dots \\ \mathbf{h}_n = W\mathbf{x}_n \end{cases} \quad (5-1)$$

就是有无穷多解的。这是因为其已知量的个数  $nd_1$  小于未知量的个数  $d_0d_1 + nd_0$ 。尽管一些研究在特定条件下提出了对隐层表征进行攻击从而恢复训练特征的办法, 但是这些方法往往需要一些额外信息, 比如底部模型权重  $W$ , 或是部分泄露的样本-表征对  $(\mathbf{x}_i, \mathbf{h}_i)$ 。接下来我们将说明, 即使攻击者只搜集隐层表征, 而没有模型的额外信息, 依然存在隐私泄露的问题。

**样本间距离信息的隐私泄露:** Johnson-Lindenstrauss 定理<sup><empty citation></sup>告诉我们, 对高维数据点的低维线性投影可以很大程度上保留数据点之间的距离关系。也就是说,

$$\|W\mathbf{x}_1 - W\mathbf{x}_2\|_2 \approx c\|\mathbf{x}_1 - \mathbf{x}_2\|, \quad (5-2)$$

其中  $c$  是一个和  $W$  的分布以及维度相关的常数。考虑到神经网络的全连接层也可以视为一个随机线性投影, 因此隐层表征之间的距离也可以反应原始输入特征之间的距离, 从而使得整个数据集的拓扑结构信息被暴露。以 MNIST 数据集为例, 我们将 784 维的输入图片通过一个按照正态分布随机初始化的矩阵 (类似于神经网络的权重初始化) 投影到 128 维, 然后对低维投影计算欧氏距离, 找到最相似的样本, 结果呈现在图 5.1 中。

可以看出, 低维的投影相似的图片, 其原始图片也几乎相似。因此, 攻击者可以通过分析隐层表征, 得到特征拥有方的数据分布、数据之间的关系, 乃至数据分布随着时间变化的信息。以疫情防控期间的口罩检测为例, 假设某公司 A 采用了某公司 B 的“口罩检测”模型来判断员工是否按照规定佩戴了口罩, 并采用拆分学习 (或其他需要发送中间表征给公司 B 的方式) 在公司内部监控摄像头的控制中心进行了模型部署。公司 B 可以仅仅通过分析中间表征, 而无需其他任何复杂的攻击操作, 来判断公司 A 的员工出现状况。若某一天公司 A 传输的中间表征的相似度与之前的中间表征的相似度降低, 则可以判断公司 A 出现了较大的人事变动情况。这显然对公司 A 的隐私信息造成了严重泄露。

**隐层表征的重用攻击:** 在纵向联邦学习开始前, 各参与方需要进行样本对齐操作, 即按照统一的样本 ID 对各参与方数据库中的样本进行统一, 保证在训练中各方采用的

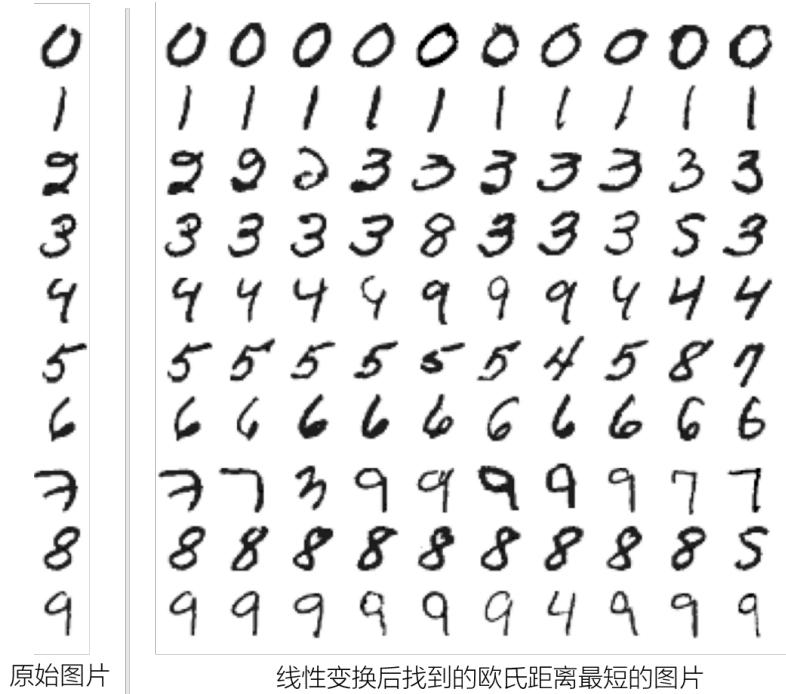


图 5.1 通过对比低维投影得到的最相似样本

是相同的样本。因此，在纵向联邦学习过程中，某一个恶意的参与方也可以将样本的隐层表征信息记录下来，并对这些进行滥用。比如某电商公司 **A** 为了训练更好的推荐系统，与某社交平台 **B** 达成协议，让社交平台 **B** 提供一部分用户数据并且以纵向联邦的方式进行训练。两者采用用户的手机号作为 ID 进行样本对齐。在训练过程中，**A** 公司可以存储隐层表征信息。训练完成后，**A** 公司可以将这些隐层表征信息，连同对应的手机号售卖给其他公司。虽然这些隐层表征信息主要是通过 **B** 公司的数据以及底部模型产生的，但是在拆分学习的场景下，**B** 公司并没有有效的技术手段检测或防止 **A** 公司对其滥用的行为。因此，这种隐层表征的重复利用攻击也对拆分学习的安全性带来了不利影响。

综上所述，使用拆分学习或是其他暴露中间结果的隐私保护机器学习方法，必然要面对中间结果与输入特征的关联性以及中间结果可能被再次利用的问题。

### 5.2.2 本文问题定义

本文考虑如下的隐私保护机器学习问题。两方  $P_0, p_1$  分别持有部分数据  $X$  和模型参数  $\Theta$ ，并且要进行模型的推断或训练，具体如下：

1. 模型推断：计算  $Y = f(X; \Theta)$ ，其中  $f$  表示模型的推断函数，是公开的。要求在推

断过程中，模型参数  $\Theta$  和各方数据  $X$ ，以及其他中间结果尽可能不暴露， $Y$  只暴露给设定的结果获取方，可能是  $P_0$  或  $P_1$ ，也可以是其他方；

2. 模型训练：计算  $\Theta' = g(X; \Theta)$ ，其中  $g$  表示模型的训练函数，比如对于梯度下降法， $g(X; \Theta) = \Theta - \alpha \partial L(X; \Theta)/\Theta$ ，其中  $L$  表示损失函数， $\alpha$  是学习率。要求在训练过程中，模型参数  $\Theta, \Theta'$ ，各方数据  $X$ ，以及其他中间结果尽可能不暴露。

在上述的隐私保护机器学习过程中，我们新增一个第三方  $P_2$ ，其自身不带任何模型参数或数据，仅仅为  $P_0$  和  $P_1$  的计算提供辅助作用。此外，我们假设各个参与方在执行特定的算法（协议）时，会遵守协议进行计算，但是其可能尽可能地利用其在执行协议中接收到的信息来获取其他参与方的隐私数据。同时，任何参与方之间不会通过共谋（Collusion）来获取其他参与方的数据。这个设定在密码学中被称为半可信安全性（Semi-Honest Security）设定或诚实但好奇（Honest-but-Curious）设定，在许多隐私保护机器学习算法的设计中被采用`<empty citation>`。

## 5.3 方法描述

### 5.3.1 基于秘密分享的线性运算

由于本章的设定是两方拥有数据和模型，外加第三方辅助计算，因此本章采用两方加法（Additive Sharing）分享作为秘密分享方案。秘密分享一般在有限域上进行，在本章中我们将其定义在整数环  $\mathbb{Z}_N$  上。 $\mathbb{Z}_N$  表示模  $N$  的整数环，其元素为  $\{0, 1, \dots, N-1\}$ 。所有的计算都通过模  $N$  进行，也就是将自然数运算的结果对  $N$  取余数。在  $\mathbb{Z}_N$  上两方加法分享的基本运算如下：

- **Share( $x$ )**: 将一个数  $x \in \mathbb{Z}_N$  分享给  $P_0$  和  $P_1$ 。具体而言，分享方（可以是  $P_0, P_1$  或其他外部参与方）产生随机数  $r \xleftarrow{\$} \mathbb{Z}_N$  ( $\xleftarrow{\$}$  表示等概率地从集合中选取)，然后将  $\langle x \rangle_0 := r$  发送给  $P_0$ ，将  $\langle x \rangle_1 := x - r$  发送给  $P_1$ 。我们用  $\langle x \rangle$  表示  $x$  处于分享状态。
- **Reconst( $\langle x \rangle, P_i$ )**: 将分享状态的  $\langle x \rangle$  恢复给  $P_i$  方（默认  $P_i = \{P_0, P_1\}$ ）。 $P_0$  和  $P_1$  分别把  $\langle x \rangle_0, \langle x \rangle_1$  发送给  $P_i$ ，然后  $P_i$  计算  $x = \langle x \rangle_0 + \langle x \rangle_1$ 。
- **Add( $\langle x \rangle, y$ )**: 将一个分享状态的数  $\langle x \rangle$  与一个公开的数  $y$  相加。将其和记为  $\langle z \rangle$ ， $P_0$  本地计算  $\langle z \rangle_0 = \langle x \rangle_0 + y$ ， $P_1$  本地计算  $\langle z \rangle_1 = \langle x \rangle_1$ 。

- $\text{Add}(\langle x \rangle, \langle x \rangle)$ : 将分享状态的  $\langle x \rangle, \langle y \rangle$  相加。将其和记为  $\langle z \rangle$ ,  $P_i$  ( $i \in \{0, 1\}$ ) 本地计算  $\langle z \rangle_i = \langle x \rangle_i + \langle y \rangle_i$ 。
- $\text{Mul}(\langle x \rangle, y)$ : 将一个分享状态的数  $\langle x \rangle$  与一个公开的数  $y$  相乘。将其乘积记为  $\langle z \rangle$ ,  $P_i$  ( $i \in \{0, 1\}$ ) 本地计算  $\langle z \rangle_i = \langle x \rangle_i y$ 。
- $\text{Mul}(\langle x \rangle, \langle y \rangle)$ : 将两个分享状态的数  $\langle x \rangle, \langle y \rangle$  相乘。该过程较为复杂, 需要借助 Beaver 三元组才可以进行。本章使用辅助第三方  $P_2$  来产生 Beaver 三元组。具体步骤如下:
  - $P_2$  产生  $u, v \xleftarrow{\$} \mathbb{Z}_N$ , 并计算  $w = uv$ 。
  - $P_2$  执行  $\text{Share}(u), \text{Share}(v), \text{Share}(w)$ , 将  $u, v, w$  分享给  $P_0, P_1$ 。
  - $P_0$  和  $P_1$  执行  $\text{Reconst}(\langle x \rangle - \langle u \rangle), \text{Reconst}(\langle y \rangle - \langle v \rangle)$ 。
  - $P_0$  计算  $\langle z \rangle_0 = (x - u)(y - v) + (x - u)\langle v \rangle_0 + \langle u \rangle_0(y - v) + \langle w \rangle_0$ ;  $P_1$  计算  $\langle z \rangle_1 = (x - u)\langle v \rangle_1 + \langle u \rangle_1(y - v) + \langle w \rangle_1$ 。

我们可以简单地验证  $z = xy$ 。

可以看到, 秘密分享的所有运算中, 仅有两个分享状态的数的乘法需要参与方之间进行信息传输。且对于任何一个参与方  $P_i$ , 如果其没有关于  $x$  的信息, 则  $x - u$  对其来说也是一个均匀分布的随机变量; 同理, 如果没有关于  $y$  的信息, 则  $y - v$  对其来说也是一个均匀分布的随机变量。因此, 使用上述方法计算两个分享状态的数的乘法并不会暴露任何隐私信息。事实上, 上述秘密分享的运算均满足信息论安全性 (Information-Theoretic Security), 也就是在各个参与方不共谋的情况下, 即使攻击者拥有无穷的计算机, 他依然无法获得任何原始数据的信息<sup><empty citation></sup>。

尽管上述的秘密分享运算只定义在单独一个整数上, 很容易可以看出我们也可以将其定义到任何向量、矩阵, 乃至张量 (Tensor) 的运算上, 乘法也可以拓展为矩阵乘法、张量乘法等一切满足乘法分配律的运算上, 且安全性得到完全保留。

### 5.3.1.1 定点数编码

一般的神经网络的计算都定义在实数上, 并且以浮点数的格式进行实际的计算。但是上述的秘密分享运算只适用于整数。因此, 我们需要定义实数与浮点数点相互转换

机制。我们把整数环  $\mathbb{Z}_N$  分为两个部分:  $[0, \lfloor N/2 \rfloor)$  用于表示正数, 而  $(\lfloor N/2 \rfloor, N)$  用于表示负数。对于一个实数  $x$ , 我们首先计算其正整数部分  $\lfloor x \cdot P \rfloor$ , 其中  $P$  表示放缩系数, 用于控制小数部分的精度。然后对于负数, 我们则用  $N - 1$  对其进行相减, 得到  $N - x - \lfloor x \cdot P \rfloor$ 。综上所述, 将实数编码为整数的函数为:

$$x_Z := \text{Encode}(x) = x \cdot P \bmod N. \quad (5-3)$$

相应的解码函数为:

$$x := \text{Decode}(x_Z) = \begin{cases} x_Z/P & \text{if } x < N/2 \text{ (}x\text{ 是正数)} \\ N - x_Z/P & \text{else (}x\text{ 是负数)} \end{cases} \quad (5-4)$$

同时, 为了保证编码和解码的准确性, 实数  $x$  必须处于区间  $[-\frac{N}{2P}, \frac{N}{2P})$  中。

### 5.3.1.2 乘法截断

当两个被编码的实数  $x_Z, y_Z$  进行乘法后, 其对应的放缩系数也会被平方:

$$x_Z y_Z = (xP \bmod N)(yP \bmod N) = xyP^2 \bmod N = [(xy)_F \bmod N \cdot P] \bmod N. \quad (5-5)$$

若乘法执行了  $L$  次, 则对应的放缩系数会变为  $P^L$ , 此时是  $xyP^L$  很可能已经超出了  $[-N/2, N/2)$  这一个有效的解码范围, 导致解码错误。为此, 我们需要在每次乘法之后进行截断操作, 即

$$z = xy \Rightarrow z_Z = \left\lfloor \frac{(x_Z y_Z) \bmod P}{P} \right\rfloor. \quad (5-6)$$

在明文状态下, 该截断操作可以直接执行。对于秘密分享状态下的正整数, 则截断操作会更为复杂。SecureML 论文的作者采用了两方直接本地截断 (Local Truncation) 的办法, 也就是:

$$\langle z_Z \rangle_i \leftarrow \langle x_Z y_Z \rangle_i / P. \quad (5-7)$$

注意这里的除法需要考虑符号位。此处我们直接用 “ $/$ ” 来表示整数除法 ( $\lfloor \cdot / \cdot \rfloor$ )。可以证明, 该方法能够以很大的概率产生和在明文上截断最多相差 1 的结果。

**定理 5.3.1 (本地截断)** 对于在  $\mathbb{Z}_N$  下分享状态的数  $\langle x \rangle_0, \langle x \rangle_1$ , 且  $|x| \leq x_{max} < N/2$ , 则最多有  $\frac{2x_{max}}{N}$  的概率使得  $\langle x \rangle_0 / P + \langle x \rangle_1 / P \notin [x/P - 1, x/P + 1]$ 。

**证明 5.3.1** 首先考虑  $\langle x \rangle_0, \langle x \rangle_1$  符号相反的情况，此时在  $\mathbb{Z}$  上有  $\langle x \rangle_0 + \langle x \rangle_1 = x$ 。注意到因为四舍五入的误差不超过 0.5，因此对于任何  $a+b=c \in \mathbb{R}$ ，有  $[a]+[b] \in [c]-1, [c]+1$ 。将其带入  $\langle x \rangle_0/P + \langle x \rangle_1/P = x/P$ ，即可得到  $\langle x \rangle_0, \langle x \rangle_1$  符号相反对该结论成立。接下来考虑  $\langle x \rangle_0, \langle x \rangle_1$  符号相同的概率。假设  $-x_{max} \leq x \leq 0$ ，则需要满足

$$\begin{cases} 0 \leq \langle x \rangle_0 < N/2 \\ 0 \leq x - \langle x \rangle_0 < N/2 \end{cases} \quad (5-8)$$

第二行可以改为  $N/2 < \langle x \rangle_0 - x \leq N$ ，只有  $\langle x \rangle_0 \in (N/2 - x, N/2)$  时，上述条件成立。再考虑  $-x_{max} \leq x \leq 0$ ，得到其概率小于  $2x_{max}/N$ 。对于  $0 \leq x \leq x_{max}$  的情况，我们也可以得到类似结果  $\langle x \rangle_0 \in [N/2, N/2 + x]$ ，证明完毕。

上述定理表明，采用简单的本地截断方法，其截断出错的概率也是很小的。但是小概率的截断错误在神经网络训练中也是不可接受的，下面我们举例说明。比如我们令  $N = 2^{64}$ ，放缩系数  $P = 2^{20}$ ， $|x|, |y| \leq 2^8$ ，则经过一次乘法  $z = xy$ ， $|z| \leq 2^{56}$ 。通过定理 5.3.1 可以得到，此时截断出错的概率小于  $2 \times 2^{56}/2^{64} = 1/128$ 。考虑到神经网络的计算中会涉及到较大的矩阵，则截断出错很容易在计算中出现。不仅如此，我们可以看到该截断产生的误差是非常大的，可以到接近  $\frac{N}{2P}$  的量级。如此巨大的误差将会给神经网络的训练或推断带来重大的影响，使得其结果毫无意义。因此，我们必须采取措施避免简单的“本地截断”带来的误差。

根据定理 5.3.1 的分析，可以看到截断错误是否发生与  $\langle x \rangle_0$  有关。具体而言，当  $|\langle x \rangle_0| \in (N/2 - x_{max}, N/2]$  时，才有可能产生截断错误。因此，对  $x$  进行截断时，我们只需要让  $P_0$  检查  $|\langle x \rangle_0|$  是否在可能截断错误的范围内，再对其进行缩小处理。如果  $|\langle x \rangle_0|$  在该范围内，若  $\langle x \rangle_0$  为正数（在  $[0, N/2]$  区间），则  $P_0$  首先计算  $\langle x \rangle_0 \leftarrow \langle x \rangle_0 - N/4$ ，并发送“+”给  $P_1$ ， $P_1$  执行  $\langle x \rangle_1 \leftarrow \langle x \rangle_1 + N/4$ 。注意到这种方法会暴露关于  $x$  的范围信息，但是其暴露的概率较小（等价于定理 5.3.1 中截断出错的概率），且即使暴露，很大概率也只暴露一个宽度为  $x_{max}$  左右的范围。考虑到神经网络的计算过程中的变量元素都很多，暴露其中很少部分元素的范围不会造成实质性的隐私泄露。总而言之，本章所提出的方法是将小概率的截断错误转化为小概率的部分元素范围泄露。因为在神经网络的训练和推断中往往涉及到数量庞大的大规模的矩阵运算，即使是小概率的截断错误，由于其错误的误差极大，也会导致神经网络的训练崩溃或输出无意义结果。而相比

之下，泄露少量元素的范围并不会让攻击者窃取有意义的隐私数据。因此，本章提出的截断方法更适合神经网络的安全计算。

### 5.3.1.3 优化的秘密分享乘法

为了进一步提高效率，我们注意到上述截断过程可以与秘密分享的乘法过程绑定，从而实现减少通讯量的目的。在秘密分享的乘法中， $P_0$  和  $P_1$  之间要进行两次通信来重构  $x - u$  和  $y - v$ ，分别是  $P_0$  发送自身对应的分享值和  $P_1$  发送自身对应的分享值。假设  $P_1$  先发送自身的秘密分享值，则  $P_0$  收到分享值后，可以先计算乘积分享值  $\langle z \rangle_0$ ，同时利用该值检查是否有需要进行缩小，再把是否需要缩小的信息连同  $\langle x \rangle_0 - \langle u \rangle_0, \langle y \rangle_0 - \langle v \rangle_0$  一同发给  $P_1$ 。这样不会在秘密分析乘法中带来任何额外的通讯。我们将优化后的秘密分享张量乘法定义在算法 5.1 中。

### 5.3.2 基于随机排列的激活函数计算

神经网络中除了矩阵的线性运算（加法、乘法），另一种不可或缺的运算是非线性激活函数（Activation Function）。若没有激活函数，则整个神经网络将退化为一个线性映射。由于激活函数是非线性的，甚至无法使用多项式表示，因此其对密码学方案不友好，当前许多研究采用混淆电路或其他精心设计的 MPC 协议计算，产生了大量的开销，同时由于其中采用了近似计算，可能还会带来神经网络的精确度损失<sup><empty citation></sup>。另一方面，如果将神经网络中间结果发送给第三方来算激活函数，则第三方可以根据中间结果来反推神经网络的权重。为此，本章提出利用随机排列来安全地解决激活函数的计算。

对  $n$  个元素进行随机排列（Random Permutation），表示将  $n$  个元素打乱顺序，总共有  $n!$  种不同的排列顺序。注意到，排列的数目随着  $n$  的增长呈现出超指数（Super-Exponential）增长，因此即使是较少的元素，其可能的排列顺序也是天文数字。比如，仅仅 10 个元素有 360 多万种排列 ( $\approx 2^{22}$ )；而当元素个数增长到 100，则可能的排列数超出  $2^{524}$ 。如此大量的排列个数，使得从随机排列中猜测出原排列的概率几乎为 0。考虑到神经网络中间结果往往是包含大量元素的张量，我们可以认为对其进行随机排列后可以几乎消除所有原始张量的信息。并且注意到，神经网络的激活函数是作用于每个元素上的（Element-wise），因此打乱顺序后执行激活函数与执行激活函数后再打乱顺序是等价的。

**算法 5.1 秘密分享乘法  $\text{SSMul}(\langle X \rangle, \langle Y \rangle, f_{\text{mul}})$** 

**输入:** 秘密分享的张量  $\langle X \rangle, \langle Y \rangle$ , 乘法函数  $f_{\text{mul}}$ , 放缩系数  $P$ , 结果范围  $z_{\max}$

**输出:** 秘密分享的张量  $\langle Z = \frac{XY}{P} \rangle$

- 1: 各方将默认乘法设置为  $f_{\text{mul}}$  // 视具体情况, 可以是逐元素乘法、矩阵乘法等
- 2:  $P_2$  生成 Beaver 三元组  $UV = W$  ( $U, V$  形状和  $X, Y$  相同, 其元素从  $\mathbb{Z}_N$  中均匀采样), 并将其分享给  $P_0, P_1$
- 3:  $P_1$  发送  $\langle X \rangle_1 - \langle U \rangle_1, \langle Y \rangle_1 - \langle V \rangle_1$  给  $P_0$
- 4:  $P_0$  恢复出  $X - U = \langle X \rangle_0 - \langle U \rangle_0 + \langle X \rangle_1 - \langle U \rangle_1, Y - V = \langle Y \rangle_0 - \langle V \rangle_0 + \langle Y \rangle_1 - \langle V \rangle_1$
- 5:  $P_0$  计算出  $\langle Z \rangle_0 = (X - U)(Y - V) + (X - U)\langle V \rangle_0 + \langle U \rangle_0(Y - V) + \langle W \rangle_0$
- 6:  $P_0$  找出  $\langle Z \rangle_0$  中满足  $\langle Z \rangle_0[i] \in (N/2 - z_{\max}, N/2)$  的下标  $i$  集合, 记作  $I_{\text{large}}$ ; 同理找到  $\langle Z \rangle_0[i] \in [-N/2, -N/2 + z_{\max})$  的下标  $i$  集合, 记作  $I_{\text{small}}$   
/\* 此处下标为把张量  $\langle Z \rangle_0$  转化为 1 维向量的下标 \*/
- 7: **for**  $i \in I_{\text{large}}$  **do**
- 8:      $P_0$  设置  $\langle Z \rangle_0[i] \leftarrow \langle Z \rangle_0[i] - N/4$
- 9: **for**  $i \in I_{\text{small}}$  **do**
- 10:      $P_0$  设置  $\langle Z \rangle_0[i] \leftarrow \langle Z \rangle_0[i] + N/4$
- 11:  $P_0$  发送  $\langle X \rangle_0 - \langle U \rangle_0, \langle Y \rangle_0 - \langle V \rangle_0, I_{\text{small}}, I_{\text{large}}$  给  $P_1$
- 12:  $P_1$  恢复出  $X - U, Y - V$
- 13:  $P_1$  计算出  $\langle Z \rangle_1 = (X - U)\langle V \rangle_1 + \langle U \rangle_1(Y - V) + \langle W \rangle_1$
- 14: **for**  $i \in I_{\text{large}}$  **do**
- 15:      $P_1$  设置  $\langle Z \rangle_1[i] \leftarrow \langle Z \rangle_1[i] + N/4$
- 16: **for**  $i \in I_{\text{small}}$  **do**
- 17:      $P_1$  设置  $\langle Z \rangle_1[i] \leftarrow \langle Z \rangle_1[i] - N/4$

因此, 我们可以让  $P_0$  和  $P_1$  对秘密分享的神经网络中间结果执行随机排列后, 再恢复到  $P_2$  上。 $P_2$  随即在随机排列后的明文上计算出激活函数的值, 再将其重新分享给  $P_0$  和  $P_1$  两方。 $P_0$  和  $P_1$  可以通过逆向排列, 恢复出最终结果的分享值。注意到, 在此过程中,  $P_0$  和  $P_1$  必须采用同样的随机排列, 否则会导致恢复错误。此外, 我们还可以通过 (在元素数量不足够大时) 添加扰动元素; 对 Tanh、Sigmoid 等对称激活函数在随机

表 5.1 不同算法 ReLU 开销对比

算法	通讯轮次	通讯量（比特）
基于随机排列（本方法）	3	$3L$
SecureNN <sup>[86]</sup>	11	$8L \log_2 p + 32L + 2$
ABY3 <sup>[85]</sup>	$6 + \log L$	$105L$
基于混淆电路 <sup>[74]</sup>	4	$k(3L - 1)$

排列中加入随机翻转来进一步提高安全性。上述方法实现简单，本文不再赘述。我们将该方法在算法 5.2 中进行具体描述。

---

#### 算法 5.2 基于随机排列的激活函数计算方法 $\text{PermNonlinear}(\langle X \rangle, f)$

---

**输入：**秘密分享的张量  $\langle X \rangle$ , 激活函数  $f$

**输出：**秘密分享的张量  $\langle Y = f(X) \rangle$

- 1:  $P_0$  生成一个随机排列  $\pi \leftarrow \text{Perm}(\text{Size}(X))$  //  $\text{Size}(X)$  表示张量  $X$  的元素个数
  - 2:  $P_0$  将  $\pi$  发送给  $P_1$
  - 3:  $P_0, P_1$  分别将  $\pi[\langle X \rangle_0], \pi[\langle X \rangle_1]$  发送给  $P_2$
  - 4:  $P_2$  恢复出  $\pi[X] = \pi[\langle X \rangle_0] + \pi[\langle X \rangle_1]$ , 并计算出  $\pi[f(X)] = f(\pi[X])$
  - 5:  $P_2$  将  $\pi[f(X)]$  分享给  $P_0, P_1$
  - 6:  $P_0$  和  $P_1$  分别计算得到  $\langle Y \rangle_0 = \pi^{-1}\langle \pi[f(X)] \rangle_0, \langle Y \rangle_1 = \pi^{-1}\langle \pi[f(X)] \rangle_1$
- 

通过该方法，我们可以极大的降低激活函数的计算开销。我们在表 5.1 中对本方法和对比算法的 ReLU 激活函数计算的通讯轮次/通讯量进行对比。表中的  $p$  是一个质数（一般为三位数）， $k$  是混淆电路的安全性参数（Security Parameter），一般大于 128。注意到，本方法的开销已经考虑了下一节所用基于关联随机性（Correlated Randomness）的通讯优化算法。

### 5.3.3 基于关联随机性的通讯优化

两方具有关联随机性，可以理解为两方有一个同样的随机数生成器，可以使得其每次采样时产生同样的随机数。通过关联随机性，我们可以进一步降低上述的秘密分享乘法（算法 5.1）以及基于随机排列的激活函数计算方法（算法 5.2）的通讯复杂度<sup>[84]</sup>。

**优化 SSMul:** 具体而言,  $P_0$  和  $P_2$  分享一个随机数生成器  $G_{\text{beaver}}$ 。在算法 5.1 第 2 行中,  $P_2$  使用  $G_{\text{beaver}}$  产生 Beaver 三元组给  $P_0$  的分享:  $\langle U \rangle_0, \langle V \rangle_0, \langle W \rangle_0 \leftarrow G_{\text{beaver}}$ 。同时,  $P_0$  自身也可以产生这些分享, 因此  $P_2$  无需再将这些分享值发送给  $P_0$ 。

**优化 PermNonlinear:**  $P_0$  和  $P_1$  分享一个随机数生成器  $G_{\text{perm}}$ , 用于在算法 5.2 中产生同样的随机排列。因此, 算法 5.2 第 2 行可以改为  $P_1$  通过  $G_{\text{perm}}$  产生随机排列, 从而减少一次通讯。此外,  $P_0$  和  $P_2$  分享一个随机数生成器  $G_{\text{share}}$ , 用于  $P_2$  产生  $\langle \pi[f(X)] \rangle_0$  时。此时在算法 5.2 第 5 行时,  $P_2$  无需再将  $\langle \pi[f(X)] \rangle_0$  发送给  $P_0$ , 从而减少一次通讯。

### 5.3.4 隐私保护神经网络框架实现

基于上述的秘密分享乘法和基于随机排列的激活函数, 我们已经能够实现基本的全连接神经网络的推断和训练。在神经网络中有如下的操作:

- 线性运算: 加法、减法、乘法 (包含矩阵乘法和逐元素相乘);
- 激活函数运算: 比如 ReLU, Sigmoid, Tanh 等;
- 本地计算, 包括了矩阵转置 (Transpose) 等。

对于加法和乘法, 我们采用秘密分享以及本章提出的算法 5.1 来实现。对于减法  $x - y$ , 我们将其转换为  $x + (-y)$ , 而负数在秘密分享中也只需要双方本地将该数取负号 (在  $\mathbb{Z}_N$  中,  $-x = N - x$ )。对于激活函数, 采用上述的算法 5.2 实现。综上, 我们将全连接神经网络的推断和训练在 ?? 中进行描述。

---

#### 算法 5.3 隐私保护神经网络推断 PriavteNNInfer( $\langle X \rangle, (\langle W \rangle_i, \langle b \rangle_i, f_i)$ )

---

**输入:** 秘密分享的输入  $\langle X \rangle$ , 秘密分享的第  $i$  层权重  $\langle W \rangle_i$ , 偏置  $\langle b \rangle_i$ , 激活函数  $f_i$ , 总层数  $L$

**输出:** 秘密分享的神经网络输出  $\langle Y \rangle$

- 1:  $\langle H \rangle_0 \leftarrow \langle X \rangle$
  - 2: **for**  $i = 0$  to  $L - 1$  **do**  $// L$  为总层数
  - 3:      $Z_{i+1} = \text{SSMul}(\langle H_i \rangle, W_i, \text{矩阵乘}) + \langle b \rangle_i$
  - 4:      $A_{i+1} = \text{PermNonlinear}(\langle Z \rangle_{i+1}, f_i)$
  - 5: **return**  $\langle A_L \rangle$
-

**算法 5.4 隐私保护神经网络训练**  $\text{PrivateNNTrainStep}(\langle X \rangle, \langle Y \rangle, (\langle W \rangle_i, \langle b \rangle_i, f_i), \alpha)$ 

**输入:** 秘密分享的输入特征  $\langle X \rangle$  和标签  $\langle Y \rangle$ , 秘密分享的第  $i$  层权重  $\langle W \rangle_i$ , 偏置  $\langle b \rangle_i$ , 激活函数  $f_i$ , 学习率  $\alpha$

**输出:** 一次梯度下降后更新的权重和偏置

- 1: 各方执行  $\langle Y' \rangle \leftarrow \text{PrivateNNInfer}(\langle X \rangle, (\langle W \rangle_i, \langle b \rangle_i, f_i))$ , 并且在执行过程中保留中间变量  $Z_1, \dots, Z_L$  和  $A_1, \dots, A_L$
- 2:  $G_L \leftarrow 2 \cdot (\langle Y \rangle - \langle Y' \rangle)$
- 3: **for**  $i = L - 1$  to 0 **do**  $// L$  为总层数
- 4:    $G_{i+1} \leftarrow \text{PermNonlinear}(Z_{i+1}, \frac{df_i}{dx})$   $// \frac{df_i}{dx}$  也是逐元素函数
- 5:    $G_{i+1} = \text{SSMul}(\langle G_{i+1} \rangle, \langle W_i^T \rangle)$   $//$  对权重进行了转置
- 6:    $b_i \leftarrow b_i - \alpha \cdot \text{sum}(G_i, \text{axis} = 0)$
- 7:    $W_i \leftarrow W_i - \alpha \cdot \text{SSMul}(A_i^T, G_i)$

## 5.4 安全性分析

本章提出的隐私保护神经网络的线性计算部分采用秘密分享进行设计，在半可信安全性设定下满足信息论安全。因此，本节讨论基于随机排列的非线性激活函数计算的安全性。尽管前文已经展示了可能的随机排列个数随着元素个数  $n$  的增长而超指数增长，本节采用了更加精确的量化手段基于分析。具体而言，本节采用了距离相关性 (Distance Correlation) 作为安全性指标，对于随机排列以及简单的线性变换（拆分学习中直接暴露中间结果的情况）进行了分析，显示了随机排列带来的隐私泄露极小。

### 5.4.1 距离相关性

距离相关性 (Distance Correlation)<sup>[42-43]</sup> 是一种统计学上的相关性度量，可以度量任意两个高维向量之间的相关性。相对于常用的皮尔逊相关性系数，距离相关性可以对任意两个不同维度的高维随机向量进行建模，并且可以捕获非线性的相关性，只有在两个随机向量完全无关时才会为 0。

**定义 5.4.1 (距离相关性)** 对于两个随机向量  $X \in \mathbb{R}^p, Y \in \mathbb{R}^q$ , 距离相关的定义为

$$DCOR(X, Y) = \frac{DCOV(X, Y)^2}{\sqrt{DCOV(X, X)^2 DCOV(Y, Y)^2}}, \quad (5-9)$$

其中  $DCOV$  表示距离协方差 (*Distance Covariance*), 定义如下

$$DCOV(X, Y)^2 = \frac{1}{c_p c_q} \int_{\mathbb{R}^{p+q}} \frac{|f_{X,Y}(t, s) - f_X(t)f_Y(s)|^2}{\|c_p\|^{p+1}\|c_q\|^{q+1}} dt ds, \quad (5-10)$$

其中,  $c_p, c_q$  为两个常数,  $\|\cdot\|$  表示欧几里得范数 (*Euclidean Norm*)。

除去上述的定义外, 距离协方差还可以通过

$$\begin{aligned} DCOV(X, Y)^2 = & \mathbb{E}\|X - X'\|\|Y - Y'\| + \mathbb{E}\|X - X'\|\mathbb{E}\|Y - Y'\| \\ & - \mathbb{E}\|X - X'\|\|Y - Y''\| - \mathbb{E}\|X - X''\|\|Y - Y'\| \end{aligned} \quad (5-11)$$

来计算, 其中  $(X, Y), (X', Y'), (X'', Y'')$  是相互独立的。通过上式可以以采样的方式来估算距离相关性。

#### 5.4.2 随机线性变换的距离相关性

现在我们研究随机线性变换后的向量与随机变换之前的向量的距离相关性。

**定理 5.4.1** 令  $X \in \mathbb{R}^n$  是一个随机的  $n$  维随机向量, 令  $Y = AX$ , 其中  $A \in \mathbb{R}^{n \times d}$  是一个随机线性变换, 并且满足在单位正交变换下概率不变, 即  $P(A) = P(AT)$ ,  $T$  是  $n$  维单位正交矩阵 (*Orthonormal Matrix*)。则

$$\mathbb{E}_A DCOV(X, AX)^2 = a DCOV(X, X)^2 \quad (5-12)$$

以及

$$\mathbb{E}_A DCOV(AX, AX)^2 = a^2 S_1 + b^2 S_2 - 2S'_3, \quad (5-13)$$

其中  $S_1 = \|X - X'\|^2$ ,  $S_2 = (\mathbb{E}\|X - X'\|)^2$ ,  $S_3 = \mathbb{E}\|X - X'\|\|X - X''\|$ ,  
 $S' = \mathbb{E}[g(\angle[X - X', X - X'']) \cdot \|X - X'\|\|X - X''\|]$ , 以及  $g_A(\theta) = \mathbb{E}_A \frac{\|Ax\|\|Ay\|}{\|x\|\|y\|}$ ,  $a = \mathbb{E}_A \frac{\|Ax\|}{\|x\|}$ ,  $b = \sqrt{\mathbb{E}_A \frac{\|Ax\|^2}{\|x\|^2}}$ ,  $x, y$  为任意的  $n$  维向量, 且夹角为  $\theta$ 。

**证明 5.4.1** 我们用  $\mathbb{E}$  来简略表示  $\mathbb{E}_{X, X', X''}$ 。首先注意到  $\mathbb{E}_A \mathbb{E}\|X - X'\|\|A(X - X')\| = \mathbb{E} \left[ \|X - X'\|^2 \mathbb{E}_A \frac{\|A(X - X')\|}{\|X - X'\|} \right]$ 。可以看出  $\mathbb{E}_A \frac{\|A(X - X')\|}{\|X - X'\|}$  和  $X$  无关, 因此可以进一步写成  $\mathbb{E}\|X - X'\|^2 \cdot \mathbb{E}_A \left[ \frac{\|A(X - X')\|}{\|X - X'\|} \right] = a \mathbb{E}\|X - X'\|^2$ 。同理可得式 5-12 的推导。另

外

$$\begin{aligned} \mathbb{E}_A \mathbb{E} \|AX - AX'\| \|AX - AX''\| &= \mathbb{E} \mathbb{E}_A \|A(X - X')\| \|A(X - X'')\| \\ &= \mathbb{E} \left[ \|X - X'\| \|X - X''\| \mathbb{E}_A \frac{\|A(X - X')\| \|A(X - X'')\|}{\|X - X'\| \|X - X''\|} \right]. \end{aligned} \quad (5-14)$$

下面我们证明  $\mathbb{E}_A \frac{\|Ax\| \|Ay\|}{\|x\| \|y\|}$  只和  $x, y$  之间的夹角有关。考虑两对  $n$  维向量  $x, y$  与  $x', y'$  满足  $\|x\| = \|x'\| = \|y\| = \|y'\| = 1$ , 且  $x, y$  和  $x', y'$  的夹角均为  $\theta$ 。令  $e_1 = x/\|x\|, e_2 = (y - y \cdot x/\|x\|)/\|y - y \cdot x/\|x\|\|$ , 且  $e_3, \dots, e_n$  是与  $e_1, e_2$  正交的任意  $n-2$  组单位正交向量组。此时, 有  $x = e_1, y = \cos \theta e_1 + \sin \theta e_2$ 。同理, 我们也可以把  $x'$  和  $y'$  表示为  $e'_1$  与  $\cos \theta e'_1 + \sin \theta e'_2$ 。此时  $(e_1, \dots, e_n)$  与  $(e'_1, \dots, e'_n)$  是  $\mathbb{R}^n$  的两组单位正交基, 因此存在单位正交矩阵  $T$  使得  $Te_i = e'_i$ 。也就是  $\frac{\|Ax\| \|Ay\|}{\|x\| \|y\|} = \frac{\|ATx'\| \|ATy'\|}{\|x'\| \|y'\|}$ 。注意到  $P(A) = P(AT)$ , 因此  $\mathbb{E}_A f(A) = \mathbb{E}_A f(AT)$ , 令  $f(A) = \frac{\|Ax\| \|Ay\|}{\|x\| \|y\|}$  可以得到  $f(A)$  仅仅和  $\theta$  有关。于是有  $\mathbb{E}_A DCOV(AX, AX) = b^2 S_1 + a^2 S_2 - 2S'_3$ , 也就是式 5-13。

定理 5.4.1 显示出, 当  $A$  是一个具有旋转不变性 (Rotation Invariance) 分布的随机线性变换时,  $DCOV(X, AX)/DCOV(X, X)$  只和  $A$  有关, 而和  $X$  的具体分布无关。同时,  $\mathbb{E}_A DCOV(AX, AX)$  随着  $g_A(\theta)$  的提升而降低。我们可以进一步证明,  $g_A(\theta)$  在  $[0, \pi/2]$  上是单调递减的。

**引理 5.4.1**  $g_A(\theta)$  在  $[0, \pi/2]$  上是单调递减的。

**证明 5.4.2** 由于  $A$  的分布具有旋转不变性, 不妨令  $x = \begin{bmatrix} \cos \theta/2 \\ \sin \theta/2 \\ \vdots \\ 0 \end{bmatrix}$  和  $y = \begin{bmatrix} \cos \theta/2 \\ \sin \theta/2 \\ \vdots \\ 0 \end{bmatrix}$ , 此时有  $g_A(\theta) = \mathbb{E}_A \|Ax\| \|Ay\|$ 。我们可以对  $A$  在前两个维度构成的平面上进行旋转角度  $\alpha$ :

$$AT = A \begin{bmatrix} \cos \alpha & \sin \alpha & O \\ -\sin \alpha & \cos \alpha & O \\ O & O & E \end{bmatrix} \quad (5-15)$$

, 此时有:

$$\begin{aligned} \mathbb{E}_A \|Ax\| \|Ay\| &= \mathbb{E}_A \|ATx\| \|ATy\| = \mathbb{E}_A \mathbb{E}_T \|ATx\| \|ATy\| \\ &= \mathbb{E}_A \mathbb{E}_\alpha [\|\cos(\alpha - \theta/2) \mathbf{a}_0 - \sin(\alpha - \theta/2) \mathbf{a}_1\| \cdot \|\cos(\alpha + \theta/2) \mathbf{a}_0 - \sin(\alpha + \theta/2) \mathbf{a}_1\|] \end{aligned} \quad (5-16)$$

其中,  $\mathbf{a}_0, \mathbf{a}_1$  是  $A$  的前两列。对  $\theta$  进行求导, 得到:

$$\begin{aligned} \frac{d\mathbb{E}_A\|Ax\|\|Ay\|}{d\theta} &= -(\|\mathbf{a}_0\|^2 + \|\mathbf{a}_1\|^2) \sin \theta. \\ \int_{\alpha=0}^{2\pi} \text{Sign}[(\|\mathbf{a}_0\|^2 - \|\mathbf{a}_1\|^2) \cos 2\alpha] - (\mathbf{a}_0 \cdot \mathbf{a}_1) \sin 2\alpha + (\|\mathbf{a}_0\|^2 + \|\mathbf{a}_1\|^2) \cos \theta d\alpha \end{aligned} \quad (5-17)$$

注意到积分号中的前两项积分后为 0, 整个式子为负数, 因此我们可以得出  $g(\theta)$  在  $[0, \pi/2]$  中单调递减。

直观角度理解,  $S'$  项反应了分布  $X$  的“集中度”对距离方差带来的影响。当  $X$  中的各个点分布在很接近的方向上时,  $\angle[X - X', X - X'']$  变小, 因此  $S'$  变大, 从而导致  $\text{DCOV}(AX, AX)$  变小, 从而导致  $\text{DCOR}(X, AX)$  变大。也就是说,  $X$  的分布的方向越集中, 则与其随机线性投影灯距离相关性有更大可能变大。在极端情况下,  $X$  分布在一条线上, 则  $S' = S_3$  取得最大值。同时, 变换  $A$  若能尽可能保留数据点之间的距离, 则也能提高变换前后的距离相关性。如果  $A$  是保距变换 (Distance-Preserving Transformation), 则  $\text{DCOR}(X, AX) = 1$ 。

在神经网络中, 我们一般采用  $\mathcal{N}(0, \sigma^2)$  来初始化全连接权重。考虑  $W \sim \mathcal{N}(0, \sigma^2)^{n \times d}$ , 此时考虑对向量  $\mathbf{e}_1$  计算  $\mathbf{E}_A\|We_1\|$  与  $\sqrt{\mathbb{E}_A\|We_1\|^2}$ , 对应卡方分布 (Chi-Square Distribution) 的均值以及其平方根的均值。因此可以得到  $a = \mathbb{E}\sqrt{w_1^2 + \dots + w_d^2} = \sigma\sqrt{2}\Gamma(d + 1/2)/\Gamma(d) \approx \sigma\sqrt{d - 1/2}$  对于较大的  $d$ ,  $b = \mathbb{E}[w_1^2 + \dots + w_d^2] = \sigma^2 d$ 。再考虑  $\mathbf{x} = \mathbf{e}_1, \mathbf{y} = \cos \theta \mathbf{e}_1 + \sin \theta \mathbf{e}_2$ , 可以得到

$$g_A(\theta) = \mathbb{E}_A\|Ax\|\|Ay\| = \int_{x,y \in \mathbb{R}^d} e^{-\frac{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2}{2\sigma^2}} \|\mathbf{x}\| \|\cos \theta \mathbf{x} + \sin \theta \mathbf{y}\| dx dy. \quad (5-18)$$

通过数值模拟, 我们也注意到:

$$\begin{aligned} \mathbb{E}_A \text{DCOR}(X, AX) &= \mathbb{E}_A \frac{\text{DCOV}(X, AX)^2}{\text{DCOV}(X, X)\text{DCOV}(AX, AX)} \\ &\approx \frac{\mathbb{E}_A \text{DCOV}(X, AX)^2}{\text{DCOV}(X, X)\mathbb{E}_A \text{DCOV}(AX, AX)} \\ &= \sqrt{\frac{a^2(S_1 + S_2 - 2S_3)}{a^2S_1 + b^2S_2 - 2S'_3}} \end{aligned} \quad (5-19)$$

也就是我们可以用距离协方差的均值来估算距离相关性的均值(尽管一般情况下  $\mathbb{E}x/y \neq \mathbb{E}x/\mathbb{E}y$ ), 这给我们提供了新的估算距离相关性的手段。

### 5.4.3 随机排列的距离相关性分析

我们首先注意到，对于  $\mathbf{x} \in \mathbb{R}^d$  和随机排列  $\pi$ ，可以得到如下特性：

- $\mathbb{E}_\pi \pi[\mathbf{x}] = [\mathbb{E}\mathbf{x}, \dots, \mathbb{E}\mathbf{x}] = M(\mathbf{x})$ ，其中我们定义  $\mathbb{E}\mathbf{x} = 1/d \cdot \sum_{i=1}^d x_i$ 。
- 对于任意排列  $\pi'$ ，有  $\pi'[\mathbf{x}] - \mathbb{E}_\pi \pi[\mathbf{x}]^T \mathbf{1} = 0$ 。
- 对于任意排列  $\pi'$ ，有  $\|\pi'[\mathbf{x}] - M(\mathbf{x})\| = \|\mathbf{x} - M(\mathbf{x})\|$ 。

通过以上特性可以归纳出，随机排列后的向量分布在以  $M(\mathbf{x})$  为中心， $\|\mathbf{x} - \mathbb{E}_\pi \pi[\mathbf{x}]\|$  为半径的一个  $d - 1$  维超球面 (Hypersphere) 上，且该超球面所在的子空间为  $\{\mathbf{x} : \mathbf{x} \in \mathbb{R}^d, \mathbf{1}^T \mathbf{x} = 0\}$ 。我们将  $\pi[\mathbf{x}] - M(\mathbf{x})$  记为  $r(\pi[\mathbf{x}])$  (误差向量)，并考察其在子空间  $\mathbf{1}^T \mathbf{y} = 0$  上面的投影。

**定理 5.4.2** 对于任何一个子子空间  $\mathbf{1}^T \mathbf{y} = 1$  上的单位向量  $\mathbf{y}$ ，我们有： $\mathbb{E}_\pi [r(\pi[\mathbf{x}])] = 0$ ，以及  $Var_\pi [r(\pi[\mathbf{x}]) \cdot \mathbf{y}] = \frac{1}{n-1} \|r(\mathbf{x})\|^2$ 。

**证明 5.4.3** 首先， $\mathbb{E}_\pi [r(\pi[\mathbf{x}])] = \sum_i^n \mathbb{E}_\pi [x_{\pi[i]} - \mathbb{E}\mathbf{x}] \mathbf{y} = \sum_{i=1}^n 0 \cdot \mathbf{y} = 0$ 。然后考虑方差，我们有：

$$Var [r(\pi[\mathbf{x}]) \cdot \mathbf{y}] = \mathbb{E}_\pi \left[ \sum_{i,j=1}^n r(\pi[\mathbf{x}])_i \cdot y_i r(\pi[\mathbf{x}])_j \cdot y_j \right] \quad (5-20)$$

我们分为  $i = j$  和  $i \neq j$  两种情况进行讨论。

- $i = j$ :

$$\mathbb{E}_\pi \sum_{i=1}^d r(\pi[\mathbf{x}])_i^2 y_i^2 = \sum_{i=1}^d \mathbb{E}_\pi [\|r(\pi[\mathbf{x}])_i\|^2] y_i^2 = \sum_{i=1}^d \frac{1}{d} \|r(\mathbf{x})\|^2 y_i^2 = \frac{\|r(\mathbf{x})\|^2}{d} \quad (5-21)$$

- $i \neq j$ : 此时我们注意到  $\sum_{i=1}^d r(\pi[\mathbf{x}])_i = 0$  和  $\sum_{i=1}^d y_i = 0$  两个等式，代入可得：

$$\mathbb{E}_\pi \sum_{i=1}^d \sum_{j=1, j \neq i}^d r(\pi[\mathbf{x}])_i r(\pi[\mathbf{x}])_j y_i y_j = \sum_{i=1}^d \sum_{j=1, j \neq i}^d \mathbb{E}_\pi [r(\pi[\mathbf{x}])_i r(\pi[\mathbf{x}])_j] y_i y_j \quad (5-22)$$

注意到，对于任意  $i, i'$ ， $\pi[i] = i'$  都是等概率的，因此，我们可以进一步得到：

$$\begin{aligned} \mathbb{E}_\pi [r(\pi[\mathbf{x}])_i r(\pi[\mathbf{x}])_j] &= \frac{1}{d(d-1)} \sum_{\substack{i', j'=1 \\ i' \neq j'}}^d r(\mathbf{x})_{i'} r(\mathbf{y})_{j'} \\ &= \frac{1}{d(d-1)} \sum_{i'=1}^d -r(\mathbf{x})_{i'}^2 = -\frac{1}{d(d-1)} \|r(\mathbf{x})\|^2 \end{aligned} \quad (5-23)$$

代入式 5-22 中，得到

$$-\frac{\|r(\mathbf{x})\|^2}{d(d-1)} \sum_{i=1}^d \sum_{j=1, j \neq i}^d y_i y_j = -\frac{\|r(\mathbf{x})\|^2}{d(d-1)} \sum_{i=1}^d -y_i^2 = \frac{\|r(\mathbf{x})\|^2}{d(d-1)} \quad (5-24)$$

将上述两项相加，可以得到  $\text{Var}_{\pi}[r(\pi[\mathbf{x}]) \cdot \mathbf{y}] = \frac{1}{n-1} \|r(\mathbf{x})\|^2$ 。

定理 5.4.2 表明，随机排列的误差向量几乎均匀地分布在超球面上，因为其没有对任何一个方向有偏好性（在任何一个方向上的投影的均值和标准差都是一样的）。我们现在考虑随机投影  $Y = AX, A \sim \mathcal{N}(0, 1)^{n \times d}$ 。由于  $A$  中的元素符合正态分布，因此我们可以得到： $\|M(Y)\| \approx \frac{1}{n} \|Y\|^2$  和  $\|r(\pi[Y])\| \approx \|Y\|$ （因为  $\pi[Y] = M(Y) + r(\pi[Y])$ ）。这种情况下， $\pi[Y]$  的误差向量的幅度显著高于其均值向量的维度。由于误差向量的分布没有方向上的偏好性，我们可以假设  $\text{DCOV}[X, r(Y)] = 0$  以及  $\text{DCOV}(M(Y), r(Y)) = 0$ 。其次，由于误差向量的幅度远大于均值向量，我们可以假设

$$\text{DCOV}[\pi[Y], \pi[Y]] = \text{DCOV}[M(Y) + r(Y), M(Y) + r(Y)] > \text{DCOV}[M(Y), M(Y)] \quad (5-25)$$

综合上述两点假设以及距离协方差的性质，我们可以得到：

$$\begin{aligned} \text{DCOR}(X, \pi[Y]) &= \text{DCOV}(X, \pi[Y]) / (\text{DCOV}[X, X] \text{DCOV}[\pi[Y], \pi[Y]]) \\ &< \text{DCOV}(X, M(Y)) / (\text{DCOV}[X, X] \text{DCOV}[M(Y), M(Y)]) \quad (5-26) \\ &= \text{DCOR}(X, M(Y)) \end{aligned}$$

注意到  $M(Y)$  是  $X$  的一维随机投影，可以进一步得到：

$$\mathbb{E}_A \text{DCOR}(X, \pi[AX]) < \mathbb{E}_A \text{DCOR}(X, M(AX)) = \mathbb{E}_{B \sim \mathcal{N}(0, 1)^{n \times 1}} \text{DCOR}(X, BX). \quad (5-27)$$

上述推导表明， $X$  和  $\pi[AX]$  之间的距离相关性要小于  $X$  与其一维投影之间的距离相关性。也就是说，从距离相关性的角度，随机投影再加上随机排列后泄漏的信息小于一维投影泄漏的信息。这说明本章提出的基于随机排列的非线性激活函数计算基本可以认为是安全的。

#### 5.4.4 数值模拟

为了验证上述对于距离相关性的分析以及呈现随机排列与随机投影对距离相关性的影响，本节我们进行了数值模拟实验来测算符合一定分布的数据经过随机投影/随机排列后的距离相关性。

我们令数据的原始维度均为 100 维，并且符合如下分布：

- **正态分布**：数据从  $\mathcal{N}(0, 1)^{100}$  中抽取。
- **均匀分布**：数据从  $\mathcal{U}(0, 1)^{100}$  中抽取。
- **稀疏分布**：数据中每个点有 0.1 的概率为 1，否则则为 0。
- **子空间分布**：数据分布在一个 20 维的子空间附近。具体而言，数据表示为  $X = AH + E$ ，其中  $H \sim \mathcal{N}(0, 1)^{20}, A \sim \mathcal{N}(0, 1/20^2), E \sim \mathcal{N}(0, 0.1)$ 。

对于每个分布，我们采样 1 万个点（记作  $X$ ），然后使用高斯分布的矩阵投影得到  $Y = WX$ ，最后对投影后的向量进行随机排列得到  $\pi[Y]$ 。我们采用文献<sup>[43]</sup>中的方法对距离相关性进行数值计算（标记为“Brown”），同时也使用式 5-19 对随机投影的距离相关性进行估算（标记为“估计值”），并一同呈现在图 5.2 中。

上述数值模拟表明，随机投影后的数据与原始数据具有较大的距离相关性，且投影的维度越高，距离相关性越大。进一步采用随机排列极大减少了距离相关性，且无论投影维度变化，其距离相关性一直小于一维投影的距离相关性，验证了本节的理论分析结论。

## 5.5 实验分析

为了验证本章所提出的隐私保护神经网络框架的实用性，本节对本章的算法进行了实现。我们基于 Python 的 Numpy 计算库实现了秘密分享的计算，其中秘密分享的整数环被选为  $\mathbb{Z}_{2^{64}}$ ，因为 64 位整数能被计算机原生支持，可以方便地使用各种编程语言操作。我们选择放缩系数为  $2^{23}$ ，也就是对于任何实数（浮点数  $x$ ），其整数表示为  $x \cdot 2^{23}$ 。已有研究表明<sup><empty citation></sup>，在该精度下神经网络的计算几乎没有精度损失。对于秘密分享中的通讯交互，我们采用 Python 原生的 socket 接口进行编程，通过在节点之间建立持久化 TCP 连接的方式来交互数据，仅需一次三次握手建立连接，防止其他上层协议带来的通讯延迟，同时我们采用流的方式进行数据读写，使用 Pickle 对变量进行打包。

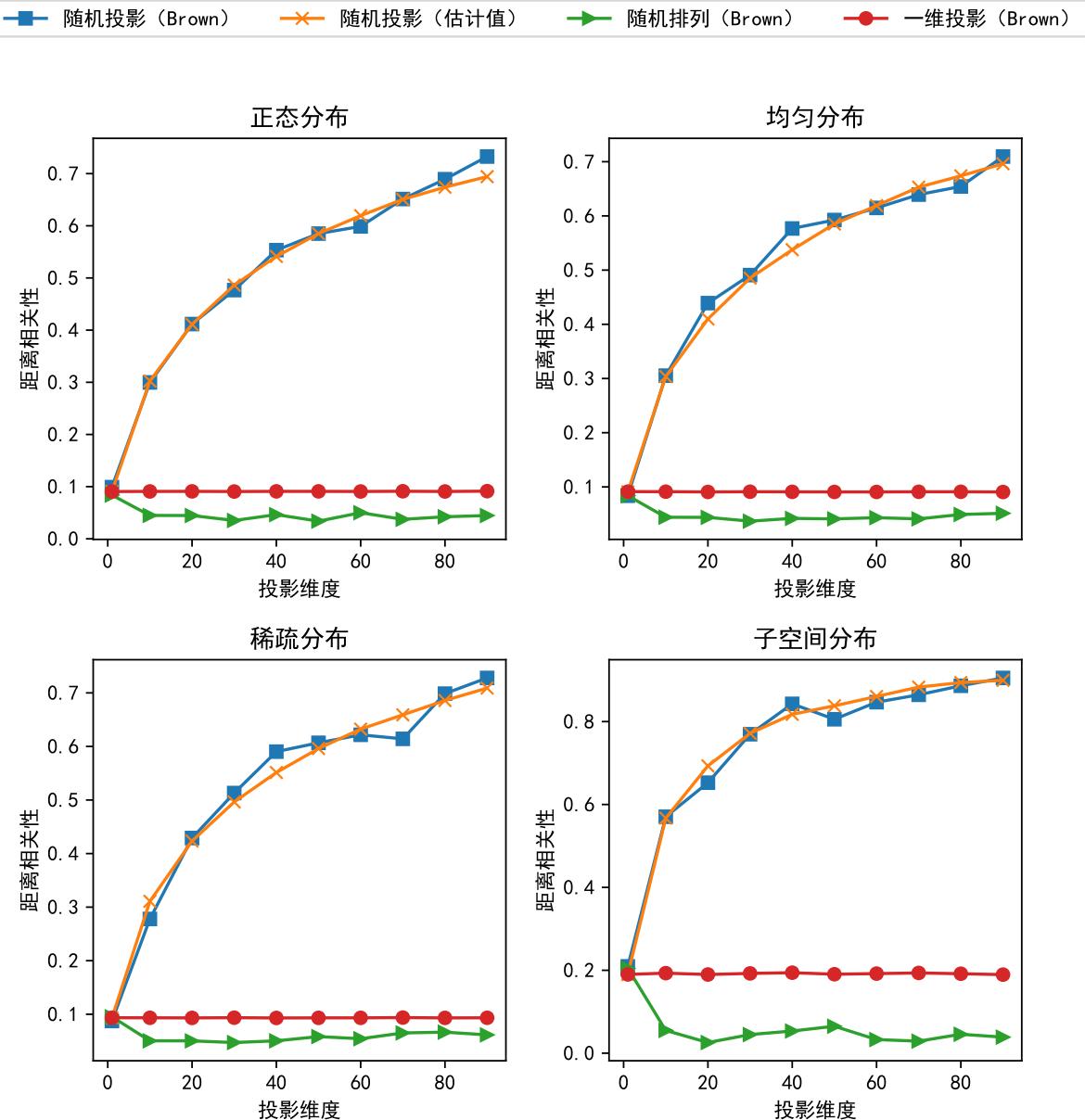


图 5.2 各种数据分布下随机投影和随机排列的距离相关性

### 5.5.1 实验设置

本节的实验在一台装备了 16 核 Intel CPU, 64GB 内存的服务器上进行。我们通过 Linux 自带的 tc (Traffic Control) 内核命令在本地网络 (Localhost) 上模拟了广域网 (WAN, Wide Area Network) 的带宽和延迟。我们将 WAN 的带宽设置为 80Mbps, 将往返时间 (Round Trip Time) 设置为 40ms。这样的设置符合普通的家用宽带标准。在实验开始之前, 我们将训练数据以及初始化的神经网络权重秘密分享在  $P_0$  和  $P_1$  上。我们对比了两个较新的密码学隐私保护神经网络方案, 分别是 SecureNN 和 ABY3。由于上述

框架原始论文的开源代码未提供或无法运行，我们使用 LatticeX-Rosetta 库的 SecureNN 框架，以及 TF-Encrypted 库的 ABY3 框架。所有的实验至少重复运行了 10 次，并把实验结果取平均值。

表 5.2 逻辑回归的运行时间和通讯量

输入维度	批大小		运行时间 (秒)			通讯量 (Mb)		
			本方法	SecureNN	ABY3	本方法	SecureNN	ABY3
100	64	推断	<b>0.099</b>	0.219	0.500	<b>0.103</b>	0.226	0.372
		训练	<b>0.279</b>	0.348	0.534	<b>0.209</b>	0.391	0.385
	128	推断	<b>0.108</b>	0.228	0.500	<b>0.202</b>	0.448	0.624
		训练	<b>0.281</b>	0.367	0.539	<b>0.413</b>	0.775	0.639
1000	64	推断	<b>0.132</b>	0.358	0.511	<b>0.996</b>	1.072	1.369
		训练	<b>0.294</b>	0.698	0.831	1.988	2.581	<b>1.678</b>
	128	推断	<b>0.114</b>	0.558	0.513	1.975	2.134	<b>1.884</b>
		训练	<b>0.334</b>	1.202	0.837	3.949	5.121	<b>3.225</b>

表 5.3 神经网络的运行时间和通讯量

模型	批大小		运行时间 (秒)			通讯量 (Mb)		
			本方法	SecureNN	ABY3	本方法	SecureNN	ABY3
DNN1	64	推断	<b>0.19</b>	0.68	0.75	<b>0.39</b>	1.98	1.90
		训练	<b>0.54</b>	1.29	0.88	<b>0.78</b>	4.05	2.21
	128	推断	<b>0.20</b>	1.10	0.92	<b>0.70</b>	3.89	3.63
		训练	<b>0.59</b>	2.08	1.78	<b>1.38</b>	7.90	4.10
DNN2	64	推断	<b>1.05</b>	5.67	1.86	<b>10.69</b>	25.18	17.16
		训练	<b>1.86</b>	12.11	3.24	<b>17.97</b>	55.98	<b>35.20</b>
	128	推断	<b>1.26</b>	9.88	3.65	<b>12.54</b>	43.08	<b>36.09</b>
		训练	<b>2.58</b>	20.00	5.14	<b>24.84</b>	93.19	<b>47.71</b>

### 5.5.2 基准测试

我们在逻辑回归模型以及全连接神经网络上对本方法和 SecureNN、ABY3 框架进行了效率的评估，包含了运行时间、网络通讯量的测量。

**逻辑回归：**对于逻辑回归模型，我们把输入维度设置为 100 或 1000，并把样本批次大小（Batch Size）设置为 64 或 128。实验结果汇报在 5.2 中。可以看出，本章提出的方法相比于对比方法，训练或推断同时减少了 2-4 倍，并且在大多数情况下也取得了更低的通讯消耗。

**神经网络：**我们测试了两种全连接神经网络模型，其输入维度分别为 100 和 1000，云层大小分别为 50 和 500。样本批次大小也同样设置为 64 和 128。实验结果汇报在 5.3 中。实验结果表明，相比于逻辑回归模型，本方法在神经网络上具有更高的优越性。本方法相比于对比方法，在训练时间上有 1.5-5.5 倍提升，同时减少了 40%-80% 的通讯量。这是因为神经网络的计算中有更多的非线性激活函数计算，使得本方法更多地发挥了在激活函数上的优势。

### 5.5.3 隐层维度效果

为了测试本方法在不同大小的隐层上的效果，本节对本方法和对比方法在不同大小的神经网络隐层的运行时间以及通讯量进行了测试，其中隐层的输入维度固定为 1000。本实验不仅测试了总的通讯量，也测试了  $P_0$  和  $P_1$  之间的通讯量和  $P_2$  相关的通讯量。实验结果呈现在 5.3 中。可以看出，随着隐层维度逐步增扎根，本方法的优势逐渐扩大。当隐层维度达到 64 左右时，本方法的所有通讯量指标都是最低的，本方法尤其减少了辅助第三方 ( $P_2$ ) 的通讯量，相比于其他方法的通讯量减少了大概 1 个数量级。由于三种方法均采用秘密分享进行线性运算，因此  $P_0$  和  $P_1$  之间的通讯量差距较小。

### 5.5.4 真实数据集实验

为了测试本章提出的方法在实际场景下的效果，本节对本章方法在真实数据下进行了实验，并且将其和明文训练进行对比，考察了训练准确率以及隐层经过随机排列后的隐私泄露程度。

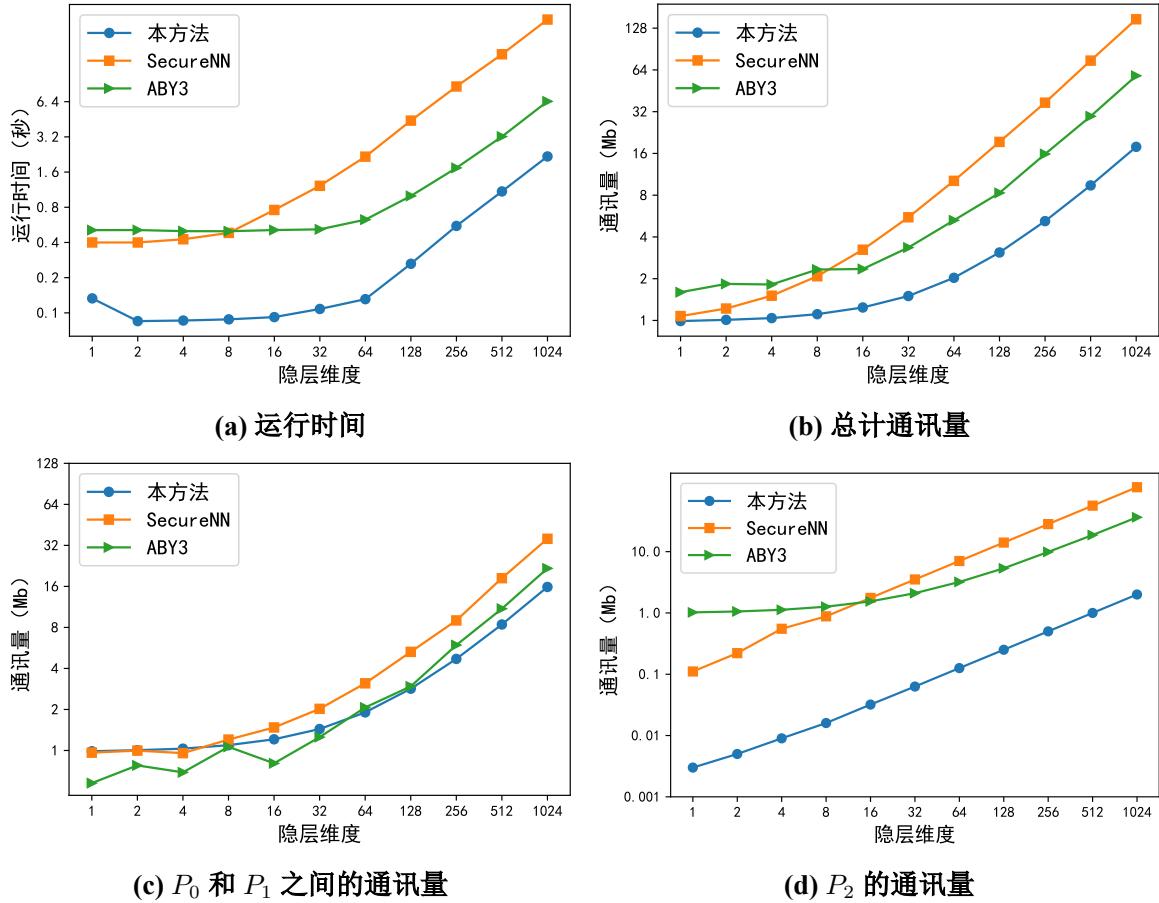


图 5.3 不同隐层维度时的运行时间和通讯量

### 5.5.5 准确率

为了验证本方法的训练准确率，我们在 Gisette 数据集和 MNIST 数据集上分别使用本方法进行了训练，同时与明文对比，并绘制出验证集准确率随着训练轮数变化的曲线图在图 5.4a-c 中。

Gisette 数据集<sup><empty citation></sup>是一个二分类数据集，包含了 6000 个训练样本和 1000 个测试样本，特征维度为 5000。我们用逻辑回归模型对其进行训练。MNIST 数据集<sup>[141]</sup>是一个 10 分类的手写数字分类数据集，包含了 50000 个训练样本和 10000 个验证样本。我们用两个全连接神经网络（DNN1、DNN2）对其进行训练，其结构分别为 784-128-10 和 784-128-32-10，DNN2 的第一个隐层激活函数为 ReLU，其余隐层采用 Sigmoid 作为隐层激活函数。

实验结果表明，采用本方法的隐私保护逻辑回归和神经网络的训练曲线与明文神经网络的训练曲线几乎重合，说明本方法带来的训练精度损失可以忽略不计。

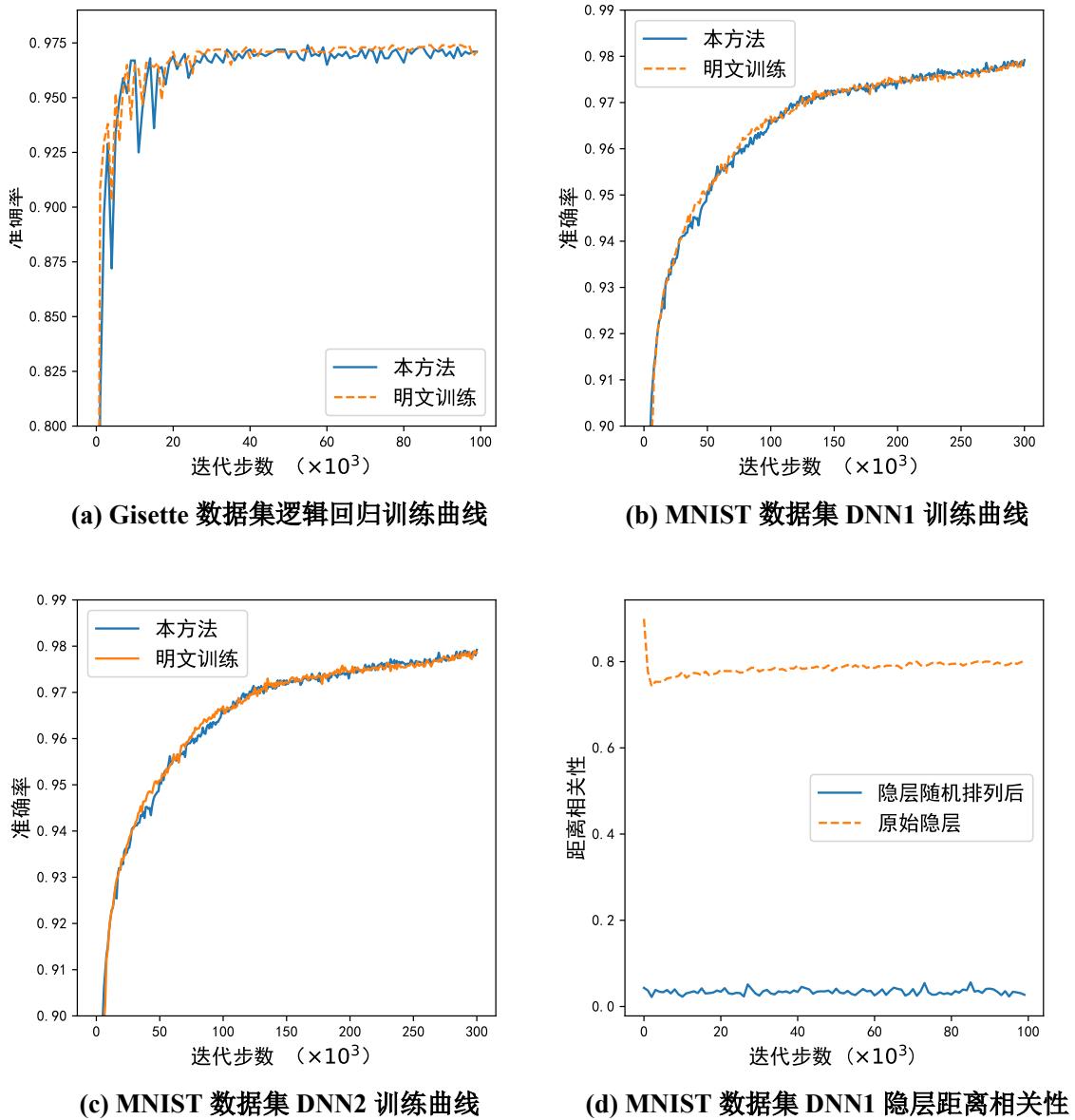


图 5.4 在 Gisette 和 MNIST 数据集上的训练结果

### 5.5.6 隐层隐私泄漏情况

我们在图 5.4c 中展现了 DNN1 隐层与输入的距离相关性，以及本方法暴露的数据（随机排列后的隐层）与输入的距离相关性。可以看出，隐层与输入的距离相关性较高，在整个训练过程中维持在 0.8 左右的水平。而随机排列后的隐层数据，其距离相关性大幅度降低到 0.1 以下，表明随机排列后的数据几乎不会带来隐私泄露。

另外，我们也测试了 MNIST 数据集中，对隐层进行相似性攻击（??）的结果。为了考察在特殊情况下的安全性，提高隐私保护的难度，我们令一批样本全部为同一类别，

因此随机排列并不能引入其他类别样本的信息。实验结果呈现在图 5.5 中。其中最左边的列表示原始样本，蓝色方框表示框内的样本被合并为单个向量来进行随机排列。

图 5.5 <caption>

实验结果表明。如果没有进行随机排列，也就是直接暴露隐层表征时，攻击者很容易根据隐层表征找到相似的样本。而采用了随机排列之后，则攻击者找到的相似样本和原始样本无关。无论原始样本属于哪个类别，找到的相似样本都是一些笔画较细的“1”等图像。这是因为两个隐层表征  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$  之间的距离可以用表示为  $\mathbb{E}_i(x_i - y_i)^2 = \mathbb{E}_i(x_i^2 + y_i^2 + x_i y_i)$ 。当  $x_i, y_i$  之间没有相关性时，则上式可以进一步表示为  $\mathbb{E}_i x_i^2 + \mathbb{E}_i y_i^2 + \mathbb{E}_i x_i \mathbb{E}_i y_i$ 。注意到  $x_i, y_i \geq 0$ 。因此，若  $\mathbb{E} y_i$  越小，就会导致  $\mathbb{E}_i(x_i - y_i)^2$  越小。而较小的  $\mathbb{E} y_i$  对应是长度较小的  $\mathbf{y}$ ，其输入大概率是黑色像素较少的图片，因为我们把白色背景编码为 0，黑色像素编码为 1。

## 5.6 本章小结



## 6 基于随机排列的高效大语言模型推理框架

大语言模型（Large Language Models，简称大模型）的出现给隐私保护机器学习带来了新的挑战。一方面，由于大模型的参数量、计算量庞大，基于密码学的隐私推断方法面临着极为严重的性能问题；另一方面，由于大模型的结构，拆分学习等方法无法保证隐私。为了实现更为高效和安全的大模型隐私推断，本章在第5的基础上，优化秘密分享的乘法协议和基于随机排列的非线性函数协议，同时采用同态加密实现预测层，实现了高效的大模型隐私推断框架 PermLLM。PermLLM 框架能够在现实网络环境下实现秒级别的大模型隐私推断，比已有的基于密码学的方案提升了超过两个数量级。

### 6.1 研究背景

2023 年 ChatGPT 的出现，标志着人工智能的发展到达了一个新的阶段<sup>[15]</sup>。ChatGPT 以及类似的大语言模型（Large Language Models），通过利用大量的语料进行自回归训练以及人类反馈强化学习，从而实现了与人类进行自然对话的能力，并且在大量任务上都达到了最先进水平<sup><empty citation></sup>。通过提示词，大模型可以完成多种多样的任务，其效果往往超过此前研究者针对该任务精心设计的模型<sup><empty citation></sup>。因此，当前大语言模型成为了学术界和业界的焦点，基于大语言模型的应用也层出不穷，包括了各种领域的问题回答（Question Answering）、阅读理解（Reading Comprehension）、文本生成（Text Generation）、检索增强（Retrieval Augmented Generation）等<sup><empty citation></sup>。

在大模型被广泛应用的同时，其隐私问题也愈加突出。大模型的训练成本极大，据估计，一个 GPT-3 模型训练的计算成本已经高达数百万美金。因此，大模型的模型参数是公司的重要资产，不能直接以开源的方式提供给用户。在这种情况下，用户必须调用模型拥有方（如 OpenAI 公司）提供的接口（API）来使用大模型。用户将自身的输入文本发送给模型拥有方的服务器，服务器上执行明文的大模型推断，然后将结果返还给用户。在这个过程中，用户输入文本的隐私就被完全暴露给了模型拥有方。这带来了极大的隐私泄漏风险。例如，三星公司的员工使用 ChatGPT 分析了公司内部的机密数据，使得三星公司开始禁止员工使用类似的大模型工具<sup><empty citation></sup>。因此，上述的隐私泄露

风险给大语言模型的应用产生了一定程度的阻碍。

为了解决大模型的隐私问题，在应用大模型时同时保护用户输入和模型参数的隐私，一些研究基于密码学方法提出了初步的解决方案。这些方法一般采用已有的秘密分享或同态加密方案来实现大模型中的线性运算，然后采用高阶多项式拟合的方式来实现大模型中的非线性运算，如 GeLU、Softmax 等。但是由于采用密码学的隐私保护机器学习框架自身会带来巨大的额外开销，加之大模型的参数规模和运算量十分庞大，上述方法即使在理想的计算资源与网络环境下，也至少需要消耗几分钟时间、数十 GB 流量才能产生一个输出单词。因此，基于密码学方法的大模型隐私推断框架的实用性依然欠缺，并且可以预见到使用已有的密码学底层算法很难实现具备实用性的大模型隐私推断。如何在保护输入的模型隐私的基础上，实现高效安全的实现大模型推断，依然是一个亟待解决的问题。

## 6.2 初步研究

本节我们对大语言模型的模型结构进行初步的介绍和分析，并从理论和实验两方面分析使用拆分学习进行大模型隐私推断时的严重隐私泄漏问题。

### 6.2.1 大模型结构

当前的大语言模型都是基于 Transformer 架构实现的，其模型可以分为 3 层，分别是

1. 输入词向量 (Input Word Embedding): 即一个嵌入向量矩阵  $E \in \mathbb{R}^{n \times d}$ ，其中  $n$  表示词汇表大小， $d$  表示词向量的维度，一般为 4096。当输入  $n$  个词语 (Token) 后，其输出为  $n \times d$  大小的矩阵。
2. Transformer 层: 包含了多层 Transformer 结构，对隐层表征进行多次变换，其输入和输出的矩阵大小均为  $n \times d$  (此处忽略批样本情况，进考虑三个样本)。
3. 输出词向量: 与输入词向量相同，包含了一个矩阵  $E' \in \mathbb{R}^{n \times d}$ ，且一般来说  $E' = E$ 。当获得 Transformer 层的输出后，对输出的最后一行计算分数  $s = E'h$ ，表示输出词语的概率分布。

其中 Transformer 层的关键模块是多头自注意力机制 (Multi-Head Self-Attention, 简称 MHA)。该模块会融合文本中不同位置词语 (上下文) 的信息产生输出，而其他模块仅仅对当前位置的表征进行变换。具体而言，对于隐层表征  $H = n \times d$ ，多头自注意力机制的运算包含如下步骤：

1. 产生查询 (Query)、键 (Key)、值 (Value): 对于任意一个位置的隐层表征  $\mathbf{h} \in H$ ，通过线性投影产生对应的  $\mathbf{q} = W_q \mathbf{h}, \mathbf{k} = W_k \mathbf{h}, \mathbf{v} = W_v \mathbf{h} \in \mathbb{R}^d$ 。
2. 将所有的查询、键、值向量分割为维度为  $d'$  的子向量 (注意力头)，总共  $d/d'$  个，并对其进行位置嵌入 (Positional Embedding) 变换，一般而言是对向量逐元素地加或乘一个固定并和当前词语相关的向量。
3. (对于每个注意力头) 对于第  $i$  个查询  $\mathbf{q}'_i$ ，将其与当前存在的所有的键作内积得到分数  $s'_{i,j} = \mathbf{q}'_i \cdot \mathbf{k}_j, j = 1 \dots n$ 。对这些分数进行 Softmax 归一化后再计算加权的值向量和  $\mathbf{h}'_i = \sum_{j=1}^n s'_{i,j} \in \mathbb{R}^{d'}$ ，这是第  $i$  个注意力头的输出。
4. 将各个注意力头的输出拼接起来，得到多头注意力的输出  $\mathbf{h}_i \in \mathbb{R}^d$ 。

从上述分析中可以看出，注意力机制将不同位置 (也就是不同输入词语) 对应的表征进行了相互融合，从而捕捉文本中上下文的关系。大语言模型在生成文本时，需要采用迭代生成的方式，即将前一轮的输出词语拼接到当前的输入语句中，然后再次输入模型预测下一个词语。由于我们仅仅需要最后一个词语对应的表征，我们可以把之前 MHA 中计算出来的键向量和值向量缓存下来，从而只需要把最后一个词语输入模型中即可计算，从而避免重复计算。这种技术被称为键值缓存 (KV Cache)。Transformer 的其他模块包括全连接层 (Fully-Connect Layer)、层归一化 (Layer Normalization) 模块，以及残差连接，是普通神经网络中的常用模块，在此不再赘述。图 6.1 展现了一个典型的大语言模型的网络结构。

### 6.2.2 针对大模型拆分学习的攻击

从上节的大模型结构描述中可以看出，大模型的隐层表征在每一层 Transformer 的变换中都保持着同样的大小，即  $n \times d$  ( $n$  表示输入语句长度， $d$  表示表征维度)。因此从直观角度理解，大模型隐层表征包含了大量的输入信息。且由于其隐层表征的尺寸随

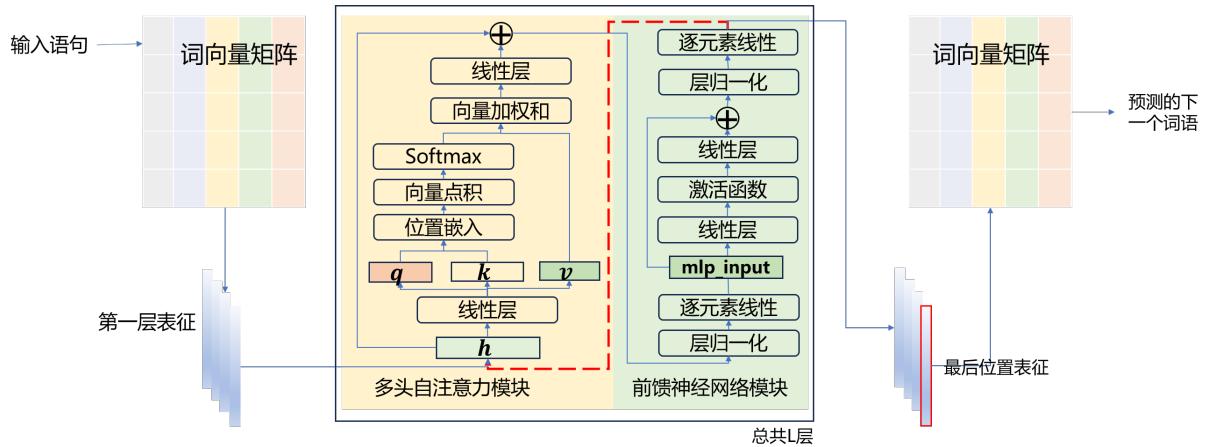


图 6.1 大语言模型结构示意图

着层数增加保持不变，可以猜测即使是高层的隐层表征，依然包含着足够多的输入数据的信息。

在拆分学习的大模型推断场景中，大模型被划分为一个底部模型  $M_b$  和一个顶部模型  $M_h$ 。我们假设模型拥有方最初拥有整个模型  $(M_b, M_h)$ ，这也符合当前大模型训练是中心化的实际情况。为了实现拆分学习的推断，模型拥有方将  $M_b$  发送给用户。这种情况下，用户需要发送底部模型产生的表征  $\mathbf{h} = M_b(\mathbf{x})$  给模型拥有方，其中  $\mathbf{x}$  表示用户的输入文本对应的下标向量。

### 6.2.2.1 表征逆向攻击

当模型拥有方获取用户输入的表征后，就可以通过如下的优化问题来猜测用户的输入：

$$\min \|M_b(\mathbf{x}') - M_b(\mathbf{x})\|^2, \quad (6-1)$$

其中  $\mathbf{x}'$  是模型拥有方猜测的用户输入， $\mathbf{x}$  是用户的实际输入。注意到直接对  $\mathbf{x}'$  进行优化是较为困难的，因为用户的输入是离散的词语下标，虽然可以转化成独热向量（One-Hot Vector），但是其依然具有离散的约束从而导致优化难以进行。为此，模型拥有方可以首先优化初始表征  $H'$ ，也就是词向量矩阵与（猜测的）用户输入对应的独热向量乘积： $H'_i = E \text{onehot}(x'_i)$ 。我们让  $T_b$  为底部模型除去词向量层（第一层）产生的模型，则我们可以把式 6-1 改写为

$$\min \|T_b(H') - M_b(\mathbf{x})\|^2. \quad (6-2)$$

由于此时优化变量  $H'$  属于连续变量，同时注意到  $T_b$  是模型拥有方已知的，因此我们可以直接用梯度下降等方法对式 6-2 进行优化。此时上述优化问题的未知数个数 ( $H'$  中的元素个数) 与方程个数相等 ( $M_b(\mathbf{x})$  的元素个数)，均为  $nd$  个元素，因为我们认为这个优化问题很可能存在唯一的最优解，也就是  $H' = H := (E \text{ onehot}(x_1), \dots, E \text{ onehot}(x_n))$ 。当得到猜测的表征  $H'$  之后，对于其中的第  $i$  行，使用余弦相似度计算出其最相似的词向量：

$$\text{最相似单词下标} = \underset{i}{\operatorname{argmax}} \frac{\mathbf{h}'_i \cdot \mathbf{e}_i}{\|\mathbf{h}'_i\| \|\mathbf{e}_i\|}, \quad (6-3)$$

其中  $\mathbf{e}_i$  表示词向量矩阵的第  $i$  行。

综上所述，从大模型隐层表征恢复出用户输入文本的过程可以分为恢复第一层表征和从第一层表征恢复单词两步。通过上述提出的两步方法，攻击者可以从隐层表征逆向推理出输入文本中每一个位置的单词，从而窃取用户的输入数据。

### 6.2.2.2 可能的防御机制

对数据加噪声是一种常用的隐私保护方法，并且在特定情况下可以实现差分隐私。因此，本节我们考虑在隐层表征中加入噪声进行防御的手段。当用户算出底部模型产生的表征  $H$  后，对其进行扰动得到

$$\tilde{H}_{ij} = H_{ij} + \lambda \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1), \quad (6-4)$$

其中， $\lambda$  表示噪声的大小。越大的噪声一般可以实现更好的保护效果。

### 6.2.2.3 实验分析

为了进一步测试上述攻击方案的可行性，我们在 ChatGLM-6B 模型上进行了实验。我们选用了 Squad<sup>[144]</sup> 和 FindSum<sup>[145]</sup> 两个数据集进行实验，分别对应较短的文本和较长的文本。对于每个实验，我们都对 200 个输入进行测试并且求其平均值。攻击准确率的定义如下：

$$\text{准确率} = \frac{1}{n} \sum_{i=1}^n \left( \frac{1}{l_i} \sum_{j=1}^{l_i} 1_{t'_{i,j} = t_{i,j}} \right), \quad (6-5)$$

其中， $n = 200$  表示总共的输入文本数， $l_i$  表示第  $i$  个文本的长度， $t_{i,j}$  和  $t'_{i,j}$  表示攻击得到的和实际的第  $i$  个文本的第  $j$  个词语。由于 ChatGLM-6B 模型总共包含了 28 个

Transformer 层，我们设置拆分层分别为第 10 层和第 20 层。我们使用梯度下降算法迭代优化式 6-2 直到  $H'$  和  $H$  的余弦相似度达到 0.98 或迭代步数达到 1000 步为止。我们使用半精度浮点数进行运算。

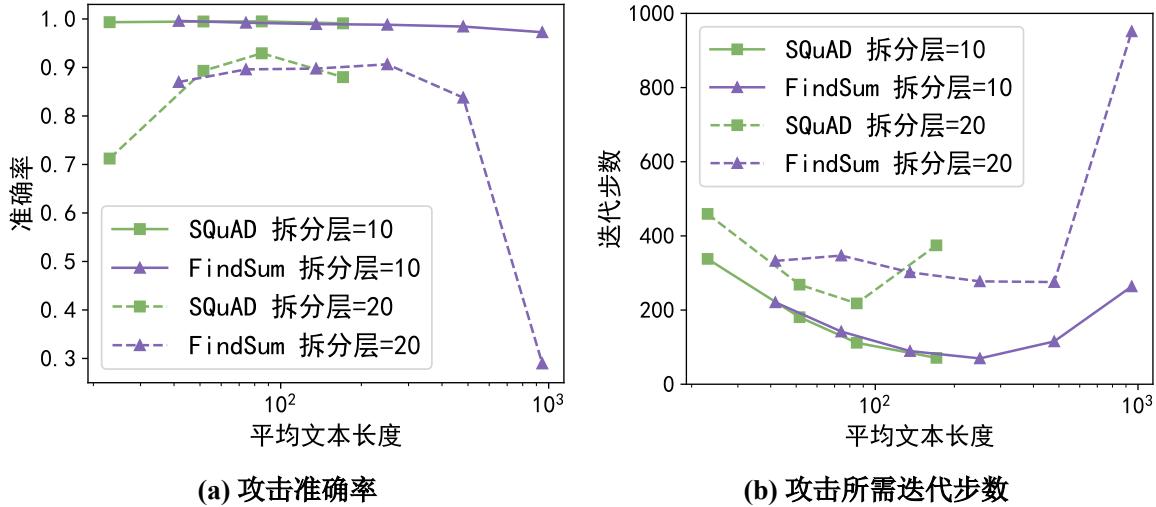


图 6.2 攻击效果与文本长度的关系

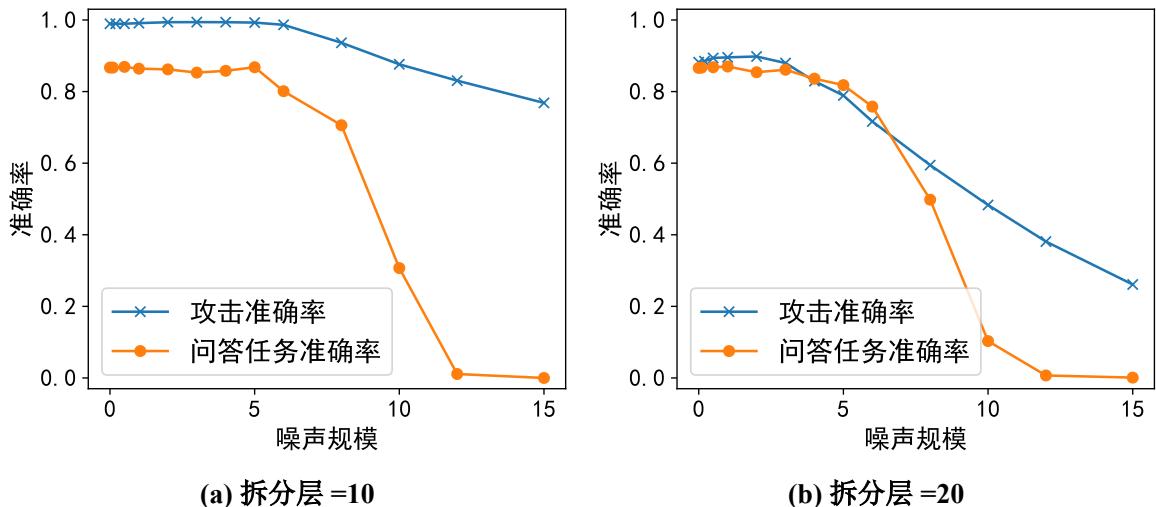


图 6.3 攻击效果与噪声大小的关系

我们首先测试了对于不同输入语句长度的逆向攻击效果，测量了窃取用户输入文本的准确率以及迭代所用次，并呈现在图 6.2 中。从图中可以看出，在拆分层为第 10 层时，无论文本长度多长，攻击都能达到接近 100% 的准确率；而拆分层在 20 层时，攻击准确率略有下降，大概为 90% 左右。同时我们注意到，文本长度特别长（1000 左右）或特别短（个位数）时，攻击效果也会下降。特别是文本长度达到 1000 时，攻击效果下降明显，但是此时迭代步数也达到了我们预设的最高值 1000 步，因此可能在更多的迭

代步数下会达到更高的效果。此外，提高浮点运算的精度（比如从 16 位半精度提升到 32 位单精度）也可能会提升攻击效果。

我们也在 Squad 数据集上，测试了噪声规模  $\lambda$  的大小对攻击效果以及原始任务效果的影响。其中原始任务的准确率指的是问答任务的准确率，按照大模型返回的字符串中是否包含参考答案的字符串计算，若包含则为回答正确，反之则为回答错误。我们将结果汇报在图 6.3 中。此处可以看到，随着噪声规模的增加，攻击准确率呈现出下降趋势。然而原始的问答任务准确率也同步开始下降，并且下降速度更快。在拆分层为第 10 层时，当问答任务的准确率已经变成 0 之后，攻击准确率依然有 80% 以上。在拆分层为第 20 层时，两者的下降幅度更加接近，但是问答任务准确率依然在  $\lambda \geq 5$  后下降的更快并且更快归零。

上述的实验分析表明，攻击者可以通过大模型拆分层的表征，有效地以很高精度还原出用户的原始输入。且通过对拆分层表征添加噪声的办法并不能有效地避免此种攻击。这也说明拆分学习不适用于大模型隐私推断的场景。

### 6.3 PermLLM 框架

经过前文的梳理和分析，我们可以看到拆分学习应用于大模型的隐私推断存在严重的隐私泄漏问题，与此同时，密码学方法则存在着严重的计算通讯开销问题。我们注意到，密码学在进行大模型隐私推断时，其主要的开销也在于非线性激活函数 GeLU 上。由于 GeLU 的非线性较强，各种密码学方案往往采用高阶多项式对其进行拟合，从而带来了严重的计算开销。现有研究的分析表明，GeLU、层归一化等的通讯开销甚至可以达到线性计算（矩阵乘法等）的一百多倍。因此，优化基于密码学的大模型隐私推断的关键在于优化非线性函数。

而上一章提出的基于秘密分享和随机排列的安全神经网络框架正是对神经网络的非线性激活函数进行优化，极大提高了神经网络隐私推断和训练的效率，同时尽可能地维持了安全性。在此基础上，本章我们对此针对大模型进行进一步优化，提出融合密码学方法和随机排列的基础上，实现高效安全的大模型隐私推断框架 PermLLM。PermLLM 框架面向的场景为两方推断的场景，且存在一个辅助第三方用于离线计算。 $P_0$  为模型拥有方（如 OpenAI），其拥有模型的所有参数信息。 $P_1$  为用户，其目标是获取大模型的

关于自身的文本输入的回复。 $P_2$  为辅助第三方，其参与离线计算阶段，用于生成安全乘法和安全排列的预计算值。我们采用了半可信的安全性设定，同??一致。

我们将大语言模型的计算分为三个部分：线性部分、非线性部分和下标选择阶段。其中线性阶段包含了线性层、注意力分数计算、层归一化后的逐元素线性映射等；非线性阶段包含了 GeLU 激活函数、注意力分数的 Softmax 计算、层归一化的计算等；而下标选择阶段则是根据预测的分数安全地产生预测词语的下标的阶段。对于线性阶段，我们通过改进的秘密分享协议进行高效计算；对于非线性阶段，我们采用基于安全随机排列协议的逐元素计算协议进行计算；对于下标选择阶段，我们基于随机排列和同态加密设计了安全的下标选择协议，不仅支持常用的选取最大分数的词语，也支持按照概率采样等复杂的下标选择策略。

### 6.3.1 隐私推断的三个阶段

许多基于秘密分享技术的隐私保护机器学习框架会按照两个阶段运行，分别是离线（Offline）阶段和在线（Online）阶段。考虑  $P_0$  和  $P_1$  要安全地计算  $f(X)$  的情况，其中  $X$  被秘密分享在两方之间，而  $f$  则是一个固定的函数。在离线阶段， $X$  还未出现， $P_0$  和  $P_1$  并不需要知道  $X$  的具体值。但是其可以对未来的  $f(X)$  进行一些准备工作，比如生成 Beaver 三元组来辅助安全乘法计算。在线阶段， $P_0$  和  $P_1$  得到了  $X$  的具体值，再进行后续的工作。通过这种方法，参与方可以在还未进行具体的隐私计算任务时提前准备，从而可以减少具体输入出现后的安全计算的开销。

而在 PermLLM 中，我们采用类似的思想，将隐私推断分为三个阶段，分别是初始化阶段、离线阶段、在线阶段。其中初始化阶段和对应的大模型权重有关，在大模型权重不改变的情况下，初始化阶段仅需执行一次。离线阶段和在线阶段则和一般隐私保护机器学习框架中的定义相同。每一次离线阶段会产生一些预计算值供在线阶段使用，而每一次在线阶段都会消耗一次离线阶段产生的预计算值。比如，一个可能的运行序列可以是“初始化-离线-在线-离线-在线”，也可以是“初始化-离线-离线-在线-在线”。但是“初始化-离线-在线-在线-离线”是一个不合法的顺序，因为第二个在线阶段时已经没有预训练的值供其使用。

### 6.3.2 优化的秘密分享乘法

一般的秘密分享乘法在每一轮都要针对两个乘数生成对应的 Beaver 三元组元素  $U$  和  $V$ ，并且在线阶段恢复出  $X - U$  和  $Y - V$ 。当乘数的尺寸很大时，这种方法会带来巨大的通讯开销。在大语言模型线性层的计算中，乘数的尺寸分别为  $d \times d$  和  $d$ ，且  $d > 10^3$ 。以 ChatGLM-6B 模型为例， $d = 4096$ ，此时恢复出  $X - U$  需要传输大约  $2 \times 4096^2$  个浮点数，折合流量  $128M$ 。即普通秘密分享乘法仅仅计算一个线性层就消耗  $128M$  的流量，导致一次推断中矩阵乘法的总计流量开销超过  $10Gb$ ，极大降低了大语言模型隐私推断的侠侣。因此，本节考虑到大语言模型推断过程的特性，对秘密分享乘法进行优化。

#### 6.3.2.1 针对线性层的秘密分享乘法

我们首先考虑大预言模型中的线性层。线性层的计算可以表示为  $W\mathbf{x} + \mathbf{b}$ ，其中  $W \in \mathbb{R}^{d \times d}$  是一个大矩阵，表示权重； $\mathbf{x} \in \mathbb{R}^d$  则是输入文本相关的隐层表征。在秘密分享中，共有两次通讯，分别是离线时  $P_2$  分发 Beaver 三元组  $(U\mathbf{v} = \mathbf{w})$  给  $P_0$  和  $P_1$ ，和在线时  $P_0$  和  $P_1$  恢复出  $W - U$  和  $\mathbf{x} - \mathbf{v}$ 。我们注意到，由于大模型推断过程中权重  $W$  是固定不变的，我们可以产生一次 Beaver 三元组中的  $U$ ，同时  $W - U$  也只需恢复一次。因此我们可以把这些计算安排到初始化阶段进行。而输入相关的  $\mathbf{x}$  则会每次变化，因此三元组中的  $\mathbf{v}$  和对应的  $\mathbf{w} = U\mathbf{v}$  会被每一次在线运算阶段消耗，需要在每次离线阶段重新生成。因为  $W$  的大小是  $\mathbf{x}$  的  $d$  倍（在大语言模型中  $d > 10^3$ ），因此将有关  $W$  的计算提前到初始化阶段单次计算，可以极大地减少在线阶段和离线阶段的通讯开销。

我们在算法 6.1 中具体描述上述算法。同时，在图 6.4 中，我们也对比了原始的秘密分享乘法和优化的  $X$  固定的秘密分享乘法。利用  $X$  是  $P_0$  已知的特性，我们可以把  $P_0$  计算的  $(X - U)(Y - V) + \langle U \rangle_0(Y - V)$ ，以及  $P_1$  计算的  $\langle U \rangle_1(Y - V)$  合并，变成仅需  $P_0$  计算  $X(Y - V)$ 。此时  $P_1$  无需知道  $Y - V$ ，只需将自身的  $\langle Y \rangle_1 - \langle V \rangle_1$  发送给  $P_0$  即可。因此我们进一步将在线阶段的通讯次数从 2 轮减少到 1 轮。

**算法 6.1**  $X$  为  $P_0$  拥有的常数的安全乘法  $\text{SecureMul}_F(X, Y)$ 

**输入:**  $P_0$  持有常数  $X$ . 秘密分享的输入  $\langle Y \rangle$

**输出:** 秘密分享的乘积  $\langle Z \rangle = \langle XY \rangle$

初始化阶段:

1:  $P_2$  产生随机的  $U$  (和  $X$  形状相同) 并发送给  $P_0$

2:  $P_0$  发送  $X - U$  给  $P_1$

离线阶段:

3:  $P_2$  产生随机的  $V$  (和  $Y$  形状相同), 并且分享  $V, W \leftarrow UV$  给  $P_0$  和  $P_1$

在线阶段:

4:  $P_1$  发送  $\langle Y \rangle_1 - \langle V \rangle_1$  给  $P_1$ ,  $P_1$  恢复  $X - U$

5:  $P_0$  计算  $\langle Z \rangle_0 \leftarrow X(Y - V) + (X - U)\langle V \rangle_0 + \langle W \rangle_0$

6:  $P_1$  计算  $\langle Z \rangle_1 \leftarrow (X - U)\langle V \rangle_1 + \langle W \rangle_1$

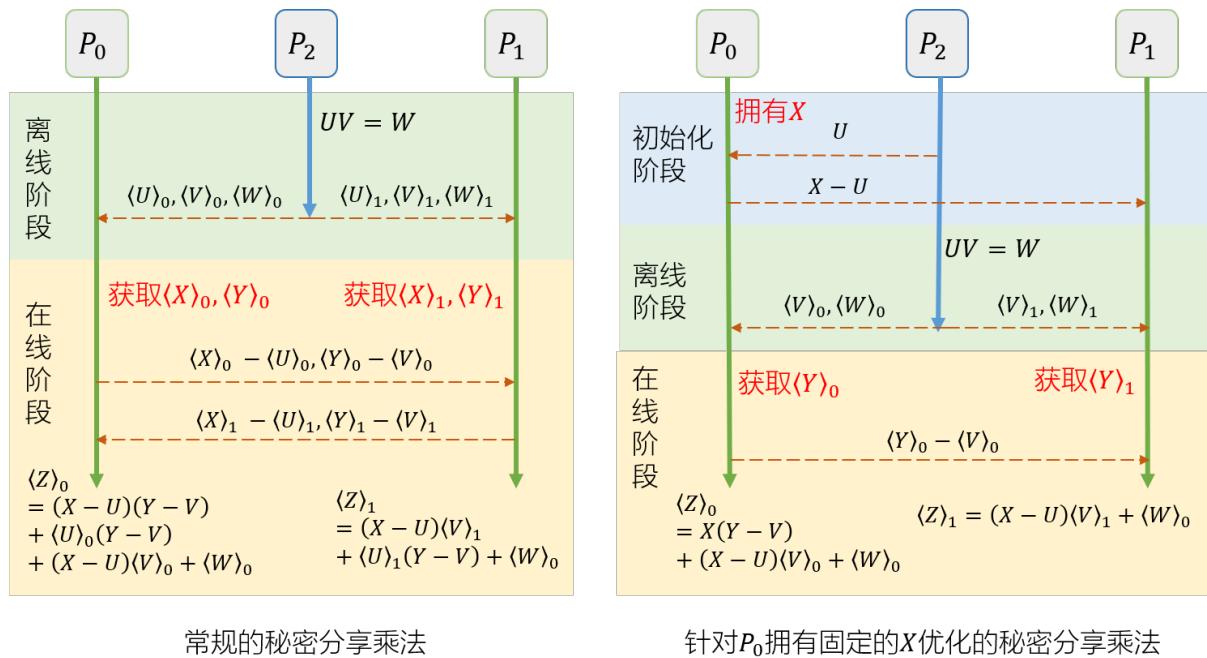


图 6.4 优化的秘密分享乘法与原始秘密分享乘法的对比

### 6.3.2.2 针对注意力的秘密分享乘法

在计算注意力分数时, 我们需要将当前的查询向量  $\mathbf{q}_i$  与多个键向量  $\mathbf{k}_1, \dots, \mathbf{k}_n$  求内积, 且注意到在单次文本生成任务中, 每轮都会有一个新的键向量加入。如果把键向量构成的矩阵当作  $X$ , 我们可以发现情况与线性层有所类似。乘数  $X$  虽然不是固定的,

**算法 6.2**  $X$  每一轮会拼接的安全乘法  $\text{SecureMul}_G$ 

**输入:** 分享值  $\langle X' \rangle$ , 原有的  $\langle X \rangle$  以及  $\langle Y \rangle$ 。每一次在线执行时,  $X$  会拼接上  $X'$ , 即:  $X \leftarrow \text{concat}(X, X')$

**输出:** 秘密分享的乘积  $\langle Z \rangle = \langle XY \rangle$

初始化:

- 1:  $P_0$  设置  $X - U \leftarrow \text{null}$ ,  $\langle U \rangle_0 \leftarrow \text{null}$   $P_1$  设置  $X - U \leftarrow \text{null}$ ,  $\langle U \rangle_1 \leftarrow \text{null}$
- 2:  $P_2$  设置  $U \leftarrow \text{null}$ .

离线阶段:

// 我们默认  $\text{concat}(\text{null}, U) = U$

- 3:  $P_2$  随机生成  $U', V$  (形状与  $X', Y$  一致), 并设置  $U \leftarrow \text{concat}(U, U')$ ,  $W \leftarrow UV$
- 4:  $P_2$  分享  $U', V, W$  给  $P_0$  和  $P_1$

在线阶段:

- 5:  $P_0$  和  $P_1$  恢复  $X' - U'$  和  $Y - V$ .
- 6:  $P_0$  和  $P_1$  计算  $X - U \leftarrow \text{concat}(X - U, X' - U')$ .
- 7:  $P_0$  计算  $\langle U \rangle_0 \leftarrow \text{concat}(\langle U \rangle_0, \langle U' \rangle_0)$ .  $P_1$  计算  $\langle U \rangle_1 \leftarrow \text{concat}(\langle U \rangle_1, \langle U' \rangle_1)$ .
- 8:  $P_0$  计算  $\langle Z \rangle_0 \leftarrow (X - U)(Y - V) + (X - U)\langle V \rangle_0 + \langle U \rangle_0(Y - V) + \langle W \rangle_0$ .
- 9:  $P_1$  计算  $\langle Z \rangle_1 \leftarrow (X - U)\langle V \rangle_1 + \langle U \rangle_1(Y - V) + \langle W \rangle_1$ .

但是其每一轮只会加入一个新的向量, 而其余部分保持不变。因此, 我们可以借用针对线性层优化的安全秘密分享乘法中的思想, 在每一轮在线阶段的运算中, 只考虑  $X$  新增加的部分 (此处记作  $X'$ ) 恢复出  $X' - U'$ , 从而更新原有的  $X - U$  上。我们在算法 6.2 中对算法进行具体描述。

在计算注意力机制时,  $X$  对应键向量矩阵, 第  $i$  行是  $\mathbf{k}_i$ 。 $X' = \mathbf{k}_n$  对应的是根据当前用户输入的最后一个词语新计算的键向量。 $Y$  对应的就是查询向量  $\mathbf{q}$ 。可以看出, 在每轮交换  $X' - U'$  而非  $X - U$  的情况下, 在线的通讯开销可以从  $nd + d + n$  减小为  $d + d + n$ 。由于  $d > 10^3$ , 减少的幅度可以高达 99%。

同时, 在计算注意力机制的加权和  $V^T K \mathbf{q}$  时 (此时  $V^T = [\mathbf{v}_1, \dots, \mathbf{v}_n]$ ),  $V$  每一轮也会拼接上新的值向量, 因此我们同样可以采用上述优化的秘密分享矩阵乘法。

### 6.3.3 基于随机排列的非线性函数计算

现有的基于密码学的大模型隐私推断框架的性能瓶颈主要在非线性激活函数，本节基于章 5 提出的基于随机排列的非线性函数计算协议，对其进行改进，实现了在线阶段仅需两方参与的非线性函数计算协议。

为此，对于一个秘密分享的输入向量  $\langle \mathbf{x} \rangle$ ，我们首先让  $P_0$  产生一个随机排列  $\pi$ ，然后通过安全的排列协议，在  $P_1$  上恢复出  $\pi[\mathbf{x}]$ 。对于逐元素非线性函数  $f$ ， $P_1$  计算出  $\pi[f(\mathbf{x})] = f(\pi[\mathbf{x}])$ ，然后将其分享得到  $\langle \pi[f(\mathbf{x})] \rangle$ 。然后  $P_0$  和  $P_1$  再次使用逆排列  $\pi^{-1}$  执行安全排列协议获得  $f(\mathbf{x})$ 。此处我们选择<sup><empty citation></sup>所提供的秘密分享的安全排列协议。由于其原始协议是两方的，因此我们可以将其离线计算阶段交给  $P_2$  执行以提高效率，保留其在线阶段的执行。我们在算法 6.3 中描述该安全排列协议。整体的非线性函数计算协议在算法 6.4 中描述。

---

#### 算法 6.3 安全排列协议 SecurePerm

---

**输入：** $P_0$  持有一个排列  $\pi$ . 秘密分享的向量  $\langle \mathbf{x} \rangle$ .

**输出：**排列后的秘密分享向量  $\langle \mathbf{y} \rangle = \langle \pi[\mathbf{x}] \rangle$ .

离线阶段：

- 1:  $P_0$  发送  $\pi$  给  $P_2$
- 2:  $P_2$  产生随机向量  $\mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathbb{Z}_N$ , 并计算  $\Delta \leftarrow \pi[\mathbf{r}_0] - \mathbf{r}_1$
- 3:  $P_2$  发送  $\Delta$  给  $P_0$ , 发送  $\mathbf{r}_0, \mathbf{r}_1$  给  $P_1$

在线阶段：

- 4:  $P_1$  发送  $\langle \mathbf{x} \rangle_1 - \mathbf{r}_0$  给  $P_0$ , 并计算  $\langle \mathbf{y} \rangle_1 \leftarrow \mathbf{r}_1$
  - 5:  $P_0$  计算  $\langle \mathbf{y} \rangle_1 \leftarrow \pi[\langle \mathbf{x} \rangle_0] + \pi[\langle \mathbf{x} \rangle_1 - \mathbf{r}_0] + \Delta$
- 

注意到，对于一批向量，且必需要进行向量内部的随机排列的场景（如 Softmax、层归一化模块），我们可以首先对向量内部元素进行随机排列，然后再对向量顺序进行随机排列。在逆排列时，先对向量顺序进行恢复，再对向量内部元素进行恢复。

### 6.3.4 基于同态加密的下标抽取

当大语言模型计算出最后一层隐层表征后，最后一行向量被用于和输出词向量计算内积，得到预测的下一个词语的可能性分数。有很多方法可以根据分数解码 (Decoding)

**算法 6.4 安全非线性计算 SecureNonlinear**

**输入:** 秘密分享的输入  $\langle \mathbf{x} \rangle$ , 一个公开的非线性函数  $f$  满足  $f(\pi[\mathbf{x}]) = \pi[f(\mathbf{x})]$  对于某一类随机排列  $\pi$  都成立

**输出:** 秘密分享的输出  $\langle \mathbf{y} \rangle = \langle f(\mathbf{x}) \rangle$

- 1:  $P_0$  产生一个随机排列  $\pi$
- 2:  $P_0$  和  $P_1$  执行  $\langle \pi[\mathbf{x}] \rangle \leftarrow \text{SecurePerm}(\pi, \mathbf{x})$ , 然后恢复出  $\pi[\mathbf{x}]$  on  $P_1$
- 3:  $P_1$  计算  $\langle \mathbf{y}' \rangle_1 \leftarrow f(\pi[\mathbf{x}])$ ,  $P_0$  设置  $\langle \mathbf{y}' \rangle \leftarrow \mathbf{0}$
- 4:  $P_0$  和  $P_1$  执行  $\langle \mathbf{y} \rangle \leftarrow \text{SecurePerm}(\pi^{-1}, \langle \mathbf{y}' \rangle)$

预测的下一个词语，最简单的方法贪心生成算法 (Greedy Generation)，即选择分数最大的词语，这会使得大语言模型的推断呈现出确定性。如果要使得大语言模型的输出有一定的随机性或多多样性，可以按照根据分数使用 Softmax 等方法算出各个词语的采样概率再进行采样。目前的基于密码学的大模型隐私推断框架，只能支持贪心生成，采用特定密码学协议来计算最大分数的下标 (也就是 argmax 函数)。

如果我们直接把分数在  $P_1$  (用户) 上恢复出来，则会导致隐私泄露。这是因为预测分数  $\mathbf{s} = W\mathbf{h}$  是词向量矩阵与隐层表征的乘积，当  $P_1$  积累了多个预测分数后，就可以得到

$$S = (\mathbf{s}_1, \dots, \mathbf{s}_n) = (W\mathbf{h}_1, \dots, W\mathbf{h}_n) = WH = \begin{bmatrix} \mathbf{w}_1^T H \\ \vdots \\ \mathbf{w}_N^T H, \end{bmatrix} \quad (6-6)$$

其中， $n$  表示预测分数向量的数量， $N$  表示词汇表大小。可见  $S$  中的每一行都是经过线性变换  $H$  后的词向量。虽然从线性变换后的词向量难以还原出原始的词向量，但是其很可能保留了原始词向量的功能，依然可以用来辅助其他语言模型的训练。由于词向量在语言模型中的地位十分重要，因此我们认为大语言模型的隐私推断必须保护词向量矩阵不被泄露。

为了解决这个问题，我们同样采用随机排列的方法。通过随机排列协议，我们可以让  $P_1$  得到随机排列的分数  $\pi[\mathbf{s}]$ ，然后通过一定的策略产生排列后的下个词语下标  $\pi[i]$ 。此时， $P_1$  再和  $P_0$  执行隐匿查询协议，将  $\pi[i]$  还原为实际的下标  $i$ 。我们采用 BFV 同态加密算法来实现上述隐匿查询协议。BFV 是一种同态加密算法，其明文空间为 4096 维的整数向量 (以多项式形式表示)，支持密文和明文的向量点积。为此，我们让  $P_1$  持有

同态加密的私钥，然后加密  $i$  对应的独热向量并发送给  $P_0$ :

$$\llbracket \mathbf{p} \rrbracket = \text{Encrypt}[\mathbf{p}], \quad \mathbf{p} = (p_1, \dots, p_N) \text{ 满足 } p_i = 1, p_{j \neq i} = 0. \quad (6-7)$$

$P_1$  然后可以执行同态的密文-明文内积运算，求出

$$\llbracket i \rrbracket = \llbracket \mathbf{p} \rrbracket \odot (\pi^{-1}[1], \dots, \pi^{-1}[n]) \quad (6-8)$$

并发送给  $P_0$  解密，此时  $P_0$  得到实际的预测下标  $i = \text{Decrypt}[\llbracket i \rrbracket]$ 。在此过程中， $P_0$  只能接触到加密的下标，而  $P_1$  只能接触到随机排列后的分数。由于每次采用不同的随机排列，因此  $P_1$  也无法从分数中获取词向量的信息。额外地，由于词汇表大小  $N$ （词语总数）大于单个密文所能装载的整数数目（4096），我们需要把明文的独热编码和逆排列向量拆分成  $L = 4096$  大小的片，然后分别内积，最后相加得到最终结果：

$$\llbracket i \rrbracket = \sum_{j=1}^{N/L} \llbracket p[jL : (j+1)L] \rrbracket \odot (\pi^{-1}[jL], \dots, \pi^{-1}[jL + L - 1]), \quad (6-9)$$

其中的求和也是同态求和。我们把上述算法步骤在算法 6.5 中正式描述。

---

#### 算法 6.5 安全预测 SecurePrediction

---

**输入:** 秘密分享的分数向量  $\langle \mathbf{s} \rangle$ ,  $P_1$  拥有解码策略  $D$

**输出:**  $P_0$  获得预测的下标  $i = \text{argmax}_i \mathbf{s}[i]$

- 1:  $P_0$  和  $P_1$  执行安全排列协议:  $\langle \pi[\mathbf{s}] \rangle \leftarrow \text{SecurePerm}(\pi, \langle \mathbf{s} \rangle)$ , 然后恢复  $\pi[\mathbf{s}]$  给  $P_1$  // 只有  $P_0$  知道  $\pi$
  - 2:  $P_1$  解码出排列后的下标  $\pi[i] \leftarrow D(\pi[\mathbf{s}])$
  - 3:  $P_1$  计算  $\mathbf{p} \leftarrow \text{onehot}(\pi[i])$ , 然后把  $\mathbf{p}$  切分成  $\lceil N/L \rceil$  个长度为  $L$  的子向量, 即  $\mathbf{p}_1, \dots, \mathbf{p}_{\lceil N/L \rceil}$
  - 4:  $P_1$  加密得到  $\llbracket p_j \rrbracket \leftarrow \text{Enc}(\mathbf{p}_j), j = 1.. \lceil N/L \rceil$ , 然后发送给  $P_0$
  - 5:  $P_0$  将向量  $(\pi^{-1}[1], \dots, \pi^{-1}[N])$  切分成  $\lceil N/L \rceil$  份  $\mathbf{q}_1, \dots, \mathbf{q}_{\lceil N/L \rceil}$ , 然后执行同态的向量内积运算  $\llbracket i \rrbracket \leftarrow (\llbracket \mathbf{p}_1 \rrbracket \odot \mathbf{q}_1) \oplus \dots \oplus (\llbracket \mathbf{p}_{\lceil N/L \rceil} \rrbracket \odot \mathbf{q}_{\lceil N/L \rceil})$   $P_0$  发送  $\llbracket i \rrbracket$  给  $P_1$
  - 6:  $P_1$  解密得到  $i \leftarrow \text{Dec}(\llbracket i \rrbracket)$
-

### 6.3.5 对 PermLLM 进行优化

通过上述的三个协议，我们已经可以实现整个大语言模型。具体而言，所有的矩阵（向量、张量）乘法运算可以由秘密共享的乘法实现，而线性层的偏置项可以由  $P_0$ （模型拥有方）本地加入自身的分享值；而所有的非线性函数，包括 GeLU、Softmax，层归一化（Layer Normalization）都由基于随机排列的非线性计算协议实现；获得了模型产生的分数向量后，再交给安全预测协议产生解码后的单词下标。本节，我们对 PermLLM 的具体实现提出一些优化，包括将一些参数公开来减少秘密分享乘法的次数，以及采用浮点数秘密分享来提高运算效率。

#### 6.3.5.1 公开部分参数

当秘密分享的数乘以一个公开的数时，只需各方将自己本地的分享乘以该数即可：

$$z = xy \Leftrightarrow \langle z \rangle_i = \langle x \rangle_i y. \quad (6-10)$$

因此，我们可以在大语言模型中选取一部分不敏感的参数进行公开，从而减少秘密分享乘法的数目，降低通讯开销。我们选取层归一化模块之后的逐元素乘法（Element-Wise Affine）的权重作为公开的参数。这些权重的参数量（元素个数）在全体参数中的占比低于 0.01%，因此几乎可以忽略不记。但是在不公开的情况下，进行秘密分享的矩阵乘法，则依然会带来通讯冗余，因此在高延迟的网络场景下会带来明显的性能损失。同时，我们也公开位置嵌入的权重，因为这些权重的计算方法往往是公开的，因此无需对其进行隐私保护。我们认为其他的参数是比较重要的，它们不仅参数量占比较大，还和模型的特定功能关系较大。例如，词向量可以用来微调其他的语言模型；而注意力机制中的线性层权重也包含了模型的一些特性，对其进行低秩扰动（LoRA）就可以让大语言模型产生不同特色和功能的文本`<empty citation>`。在实际情况中，也可以选择更多的公开参数，实现隐私保护和性能之间的权衡。

#### 6.3.5.2 浮点数秘密分享

秘密分享起初是在有限域（Finite Domain）上定义的，比如整数环  $\mathbb{Z}_N$ 。在这种情况下，秘密分享可以实现信息论的安全，因为任何一方在协议执行过程中接收到的信息都是均匀分布在有限域上的。但是当前的 GPU 主要面向浮点数的运算，整数运算的效果

率较低。因此也有一些工作探究了在浮点数上进行秘密分享`<empty citation>`。因此我们可以将（加法）秘密分享推广到浮点数上以提高大语言模型隐私推断的效率。在这种情况下，我们定义秘密分享的分享过程为  $x = \langle x \rangle_0 + \langle x \rangle_1$ ，其中  $\langle x \rangle_0 \sim \mathcal{D}$ 。 $\mathcal{D}$  为一个  $\mathbb{R}$  上的随机分布，其规模需要比  $x$  大很多，从而实现对  $x$  的充分隐藏。我们假设  $x \sim \mathcal{X}$ ，则可以让  $\text{Var}[\mathcal{D}] = \lambda^2 \text{Var}[\mathcal{X}]$ ，其中  $\lambda$  控制随机分布的大小，越大的  $\lambda$  可以越好地隐藏  $x$ 。类似地，用于计算乘法  $XY = Z$  的 Beaver 三元组也需要按照同样的分布生成：

$$\begin{cases} \text{Var}[\langle U \rangle_0] = \text{Var}[\langle U \rangle_1] = \lambda^2 \text{Var}[X], \\ \text{Var}[\langle V \rangle_0] = \text{Var}[\langle V \rangle_1] = \lambda^2 \text{Var}[Y], \\ \text{Var}[\langle W \rangle_0] = \lambda^2 \text{Var}[Z]. \end{cases} \quad (6-11)$$

在这种情况下，我们可以保证在秘密分享的计算过程中暴露的值都是原始值加上一个很大的噪声，因此可以几乎与原始值无关。但是在某些情况下，如同样的值被多次重复分享，则攻击者可以通过求平均来恢复原始值，从而带来隐私泄漏风险。注意到，浮点数秘密分享对于 PermLLM 来说并不是必需的，其目的仅在于提高 GPU 计算的适配性和效率，可以简单地将其更改为整数秘密分享。

## 6.4 安全性分析

本节我们对 PermLLM 进行的安全性进行分析，表明其泄漏的隐私是忽略不计的。我们分别对随机排列和浮点数秘密分享的安全性进行分析。

### 6.4.1 随机排列

在 PermLLM 中，随机排列用于注意力分数的 Softmax 计算、激活函数的计算，以及最终的词语分数解码中。首先考虑注意力分数，假设有  $h$  个注意力头，键值向量个数为  $n$ ，则总共有  $h!(n!)^h$  总随机排列的方式。即使  $n = 1$ ，也有  $h!$  种。在大语言模型中， $h \geq 32$ ，因此  $h! > 2^{117}$ ，因此猜测出实际排列的概率可以忽略不计。而对于激活函数，则排列元素个数为隐层大小，超过 1000；对于词语分数，则其元素个数为词汇表大小，超过 10 万。因此这些情况恢复出原始排列的可能性更加接近 0。

**暴力搜索攻击：**一种针对随机排列的攻击是暴力搜索（Brute-Force Search）攻击。对于随机排列后的结果  $\mathbf{y}' = \pi[f(\mathbf{x})]$ ，攻击者可以选取所有可能的输入  $\mathbf{x}' \in X$ ，并检查

$f(\mathbf{x}')$  是否与  $f(\mathbf{x})$  的元素相同。这是因为随机排列只会改变向量中元素的顺序，但是还是会保留原始的元素集合。在不考虑计算资源的情况下，若函数的输入  $\mathbf{x}$  的取值是有限的或是可数（Countable）的，则攻击者理论上可以通过暴力搜索攻击从随机排列的输出推出（可能的）原始的输入。

由于大语言模型输入是离散的，因此也存在暴力搜索攻击的风险。但是 PermLLM 巧妙地回避了此种攻击。在 PermLLM 中，只有  $P_1$ （用户）才能获得随机排列后的明文，而  $P_1$  本身也是离散输入的拥有方，因此不存在暴力破解的可能性。

#### 6.4.2 浮点数秘密分享

在浮点数秘密分享中，任何一个数  $x$  的分享值满足

$$\begin{cases} \langle x \rangle_0 = D & \text{其中 } \mathbb{E}[D^2] = \lambda^2 \mathbb{E}[x^2] \\ \langle x \rangle_1 = x - D \end{cases} \quad (6-12)$$

此处为了方便，我们假设  $\mathbb{E}x = \mathbb{E}D = 0$ 。可见  $\langle x \rangle_0$  与  $x$  无关，而  $\langle x \rangle_1$  是  $x$  加入一个规模是输入的  $\lambda$  倍的巨大噪声的结果。同时我们也注意到，在执行秘密分享的乘法时， $x - u$  会暴露给双方，此时  $\mathbb{E}[u^2] = 2\lambda^2 \mathbb{E}[x^2]$ 。对于秘密分享乘法的结果  $z = xy$ ，在产生 Beaver 三元组的时候我们已经选取了  $\mathbb{E}(W)_0 = \lambda^2 \mathbb{E}[z^2]$ ，因此  $\langle z \rangle_0$  与  $z$  无关，而  $\langle z \rangle_0$  是  $z$  加上一个规模为其  $\lambda$  倍的噪声。综上所述可以得出，对于任何一个值  $x$ ，采用浮点数秘密分享时，各方获取到的数据  $x'$  只有两种情况：

1.  $x'$  与  $x$  无关；
2.  $x' = x + D$ ，且  $\mathbb{E}[D^2] = \lambda^2 \mathbb{E}[x^2]$ 。

当  $\lambda$  很大时， $x'$  暴露的关于  $x$  的信息暴露可以忽略不计。但是如果使用同样的输入变量进行多次浮点秘密分享计算，则攻击者可以通过多次平均的方式来获取原始的值。假设  $x$  是原始值，攻击者获取的加噪值为  $x_1, \dots, x_n$ ，则有：

$$\mathbb{E} \left[ \left( \frac{x_1 + \dots + x_n}{n} - x \right)^2 \right] = \mathbb{E} \left[ \frac{(x_1 - x)^2 + \dots + (x_n - x)^2}{n^2} \right] = \frac{\lambda^2}{n}. \quad (6-13)$$

若选取  $\lambda = 100$ ，则攻击者（如  $P_1$ ）采用同样的输入进行一万次隐私推断，并对上述秘密分享值进行平均，可以将误差缩小到原始数据规模的  $1/10$  水平： $\sqrt{\mathbb{E}[(\bar{x} - x)^2]} = \mathbb{E}[x^2]/10$ 。

此时  $P_1$  可以根据这些估计的秘密分享值，也就是隐私推断中的中间结果，反推出模型权重，从而实现窃取模型的目的。因此，采用浮点数秘密分享时，必须要采取额外的手段来防止上述攻击的产生。可能的方案有：采用第三方对  $P_1$  的输入文本进行验证； $P_0$  设计一定的手段检测恶意行为；以及当隐私推断达到一定次数后，对模型进行恒等变换来改变权重值<sup><empty citation></sup>，再重新执行初始化阶段分享权重等，

## 6.5 实验分析

为了验证所提出的 PermLLM 方法，我们在不同的设定下进行了实验，包括了不同的网络环境以及不同的 Transformer 模型大小。我们的当前最新的基于密码学的大语言模型或 Transformer 模型隐私推断框架进行了对比，包括 MPCFormer<sup><empty citation></sup>和 Puma<sup><empty citation></sup>。这些实验在搭载 4 张 NVIDIA RTX 3090(24G) 显卡的服务器上进行。此外，我们使用 PermLLM 框架实现了 ChatGLM-6B 模型<sup><empty citation></sup>，并在搭载有 3 台 NVIDIA L20(48G) 的显卡的服务器上进行了实验。

我们考察了以下的网络环境：

- LAN (Local Area Network): 此时我们直接采用本机 IP 地址 (127.0.0.1) 进行通讯。
- WAN1: 往返时延为 10ms，相当于 400 千米距离的网络时延，如东京到大阪；网络带宽为 1Gbps，对应商用水平的网络连接。
- WAN2: 往返时延为 20ms，相当于 1000 千米距离的网络时延，如柏林到伦敦；网络带宽为 100Mbps，对应家用网络。

### 6.5.1 基准测试

我们首先对非线性函数计算和单层的 Transformer 推断进行了基准测试，并且测量了通讯量、通讯轮数和总的执行时间。我们将非线性函数的结果汇报在表 6.1 中，将单层 Transformer 结果汇报在表 6.2 中。

可以看到，对于非线性运算而言，在任意输出规模或网络环境下，PermLLM 框架都达到了最高的效率，其通信轮数、通信量以及时间消耗都显著低于对比算法。不同于

表 6.1 非线性函数的基准测试

$f$	$n$	方法	通信 (Mb)	轮数	LAN	1Gbps/10ms	100Mbps/20ms
Argmax	1K	MPCFormer	1.24	101	0.09 (0.01)	2.11 (0.00)	4.12 (0.01)
		Puma	$\approx 0.80$	$\approx 1700$	0.15 (0.03)	3.29 (0.03)	6.33 (0.05)
		<b>PermLLM</b>	<b>0.26</b>	<b>4</b>	<b>0.03 (0.00)</b>	<b>0.05 (0.00)</b>	<b>0.06 (0.01)</b>
	100K	MPCFormer	110.9	153	0.78 (0.04)	4.04 (1.41)	10.74 (0.50)
		Puma	$\approx 80$	$\approx 2500$	1.17 (1.13)	6.23 (0.08)	12.88 (0.06)
		<b>PermLLM</b>	<b>4.02</b>	<b>4</b>	<b>0.12 (0.01)</b>	<b>0.17 (0.01)</b>	<b>0.2 (0.00)</b>
Element-wise	1K	Puma: GeLU	$\approx 1.7$	$\approx 200$	0.04 (0.02)	0.38 (0.02)	0.71 (0.01)
		Puma: LN	$\approx 0.20$	$\approx 190$	0.02 (0.00)	0.35 (0.01)	0.67 (0.01)
		<b>PermLLM: any</b>	<b>0.01</b>	<b>3</b>	<b>0.00 (0.00)</b>	<b>0.02 (0.00)</b>	<b>0.04 (0.01)</b>
	100K	Puma: GeLU	$\approx 170$	$\approx 200$	1.32 (0.11)	1.69 (0.23)	12.37 (0.08)
		Puma: LN	$\approx 20$	$\approx 200$	0.13 (0.03)	0.15 (0.02)	0.83 (0.03)
	100×1K	<b>PermLLM: any</b>	<b>1.15</b>	<b>3</b>	<b>0.01 (0.00)</b>	<b>0.03 (0.01)</b>	<b>0.05 (0.00)</b>

表 6.2 单层 Transformer 的基准测试

模型大小	方法	通信 (Mb)	轮数	LAN	1Gbps/10ms	100Mbps/20ms
Large ( $d = 4096$ )	MPCFormer	3073.79	53	$1.37 \pm 0.01$	$26.15 \pm 0.35$	$259.65 \pm 0.36$
	Puma	$\approx 33$	$\approx 1500$	8.82 (0.09)	11.45 ± 0.42	14.40 ± 0.65
	<b>PermLLM</b>	<b>0.49</b>	<b>20</b>	<b>0.04 ± 0.01</b>	<b>0.12 ± 0.01</b>	<b>0.24 ± 0.00</b>
Small ( $d = 768$ )	MPCFormer	108.32	53	$1.29 \pm 0.19$	$1.71 \pm 0.01$	$10.29 \pm 0.20$
	Puma	$\approx 6$	$\approx 1000$	0.46 (0.03)	2.24 ± 0.03	3.97 ± 0.04
	<b>PermLLM</b>	<b>0.1</b>	<b>20</b>	<b>0.034 ± 0.00</b>	<b>0.15 ± 0.01</b>	<b>0.23 ± 0.02</b>

对比算法需要针对特定的非线性函数设计安全计算协议，PermLLM 框架通过随机排列的方法，任何逐元素非线性函数都能在 4 轮通信中完成计算。同时，其通信量也只是明文大小的 4 倍。对于 Argmax 函数，我们使用的基于同态加密的安全下标抽取算法，只需要进行数十次同态乘法和同态加法，即可完成计算，其速度也领先于对比算法。

同样地，对于单层 Transformer 推断，PermLLM 也达到了最大的效率。且对于更大的 Transformer，PermLLM 的优势更大，说明了其在大语言模型中会具有更大的优势。当 Transformer 的维度从 768 提升到 4096 后，其优势从一个数量级提升至两个数量级。我们也注意到，在 MPCFormer 中，其采用 Crypten 框架进行秘密分享乘法，没有考虑到大语言模型的权重固定情况。每一次秘密分享乘法时，都会对权重产生新的 Beaver 三元组并分发给  $P_0$  和  $P_1$ ，以及恢复遮罩过的权重 ( $X - U$ )，带来了巨大的额外性能损耗。

### 6.5.2 ChatGLM-6B 测试

我们使用 PermLLM 框架实现了 ChatGLM-6B 模型，并且在两个 WAN (Wide Area Network) 网络环境下进行了实验。我们选择了 6, 15, 37 和 62 长度的输入文本，测量了每生成一个词语所需的时间和流量，绘制在图 6.5 中。可以看到，在 WAN2 (20ms/100Mbps) 环境下，每个词语生成的时间大概在 7 秒左右；而在更快的 WAN1 (10ms/1Gbps) 下，词语生成的时间仅需 3 秒左右。注意到第一个词语产生的时间较长，这是因为在第一个词语生成时，要把整个输入文本放入模型进行推断；但是因为我们采用了键值缓存 (Key-Value Cache) 技术，之后只需要放入新生成的文本即可。这个实验证明了 PermLLM 可以在普通的网络环境下把大模型隐私推断的时间提升到秒级别，相比于密码学方案提升了两个数量级以上，从而使其具备一定的实用性。

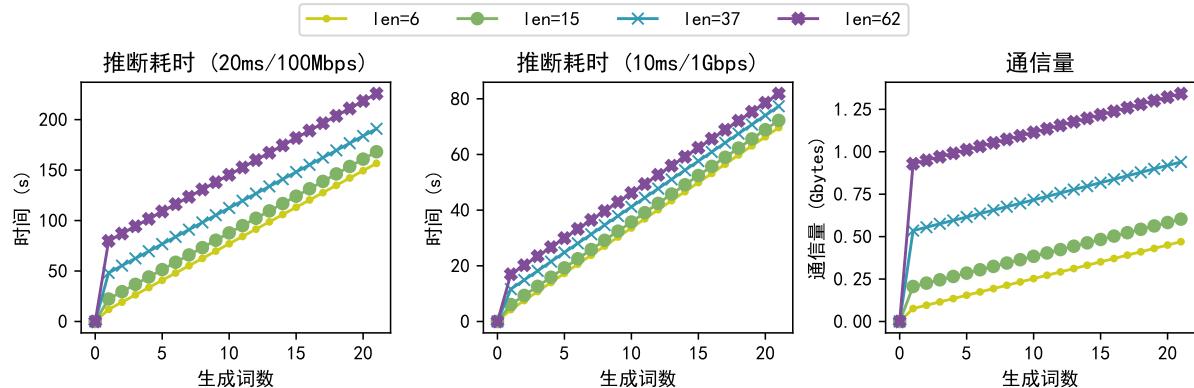


图 6.5 ChatGLM-6B 实验结果

此外，实验也观察到 PermLLM 框架几乎不会有准确率的损失。这是因为浮点秘密分享的情况下，误差来源仅为秘密分享计算中的浮点计算舍入误差 (Rounding Error)。即使将其转化为整数计算，只需要设置精度位，同样可以实现几乎无误差的隐私推断。对比之下，基于密码学的方法由于采用了多项式拟合 GeLU 等技术，会带来除去舍入误差之外的更大误差。

## 6.6 本章小结

本章我们研究了大语言模型的隐私推断问题，提出了拆分学习在此问题上的不可行性。针对密码学方法的大语言模型隐私推断中非线性激活函数开销大的问题，我们基于第5章提出的隐私保护神经网络框架进行改进，并且对秘密分享进行进一步优化，结合

同态加密技术，实现了高效的大模型隐私推断框架 PermLLM。我们利用 PermLLM 框架实现了 ChatGLM-6B 模型，在现实的网络环境下实现了秒级别的隐私推断速度，为大语言模型隐私推断的实用打下了基础。



## 参考文献

- [1] LECUN Y, BENGIO Y, HINTON G. Deep learning[J]. nature, 2015, 521(7553): 436-444.
- [2] ZHANG C, LU Y. Study on artificial intelligence: The state of the art and future prospects[J]. Journal of Industrial Information Integration, 2021, 23: 100224.
- [3] WANG Y, PAN Y, YAN M, et al. A survey on ChatGPT: AI-generated contents, challenges, and solutions[J]. IEEE Open Journal of the Computer Society, 2023.
- [4] 国务院. 新一代人工智能发展规划的通知[EB]. 2017. [https://www.gov.cn/zhengce/content/2017-07/20/content\\_5211996.htm](https://www.gov.cn/zhengce/content/2017-07/20/content_5211996.htm).
- [5] SCIENCE N, on ARTIFICIAL INTELLIGENCE T C ( S C. The National Artificial Intelligence Research and Development Strategic Plan: 2023 Update[M]. National Science, 2023.
- [6] 新华社. 我国人工智能蓬勃发展核心产业规模达 5000 亿元[EB]. 2023. [https://www.gov.cn/yao/wen/liebiao/202307/content\\_6890391.htm](https://www.gov.cn/yao/wen/liebiao/202307/content_6890391.htm).
- [7] 国务院. 中华人民共和国个人信息保护法[EB]. 2021. [https://www.gov.cn/xinwen/2021-08/20/content\\_5632486.htm](https://www.gov.cn/xinwen/2021-08/20/content_5632486.htm).
- [8] VOIGT P, VON DEM BUSSCHE A. The EU general data protection regulation (GDPR)[J]. A Practical Guide, 1st Ed., Cham: Springer International Publishing, 2017, 10(3152676): 10-5555.
- [9] MANN Z Á, WEINERT C, CHABAL D, et al. Towards practical secure neural network inference: the journey so far and the road ahead[J]. ACM Computing Surveys, 2023, 56(5): 1-37.
- [10] LIU Y, KANG Y, ZOU T, et al. Vertical Federated Learning: Concepts, Advances, and Challenges [J]. IEEE Transactions on Knowledge and Data Engineering, 2024.
- [11] YANG Q, LIU Y, CHEN T, et al. Federated machine learning: Concept and applications[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2019, 10(2): 1-19.
- [12] MCMAHAN B, MOORE E, RAMAGE D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial intelligence and statistics. 2017: 1273-1282.
- [13] AL-RUBAIE M, CHANG J M. Privacy-preserving machine learning: Threats and solutions[J]. IEEE Security & Privacy, 2019, 17(2): 49-58.
- [14] XU R, BARACALDO N, JOSHI J. Privacy-preserving machine learning: Methods, challenges and directions[J]. arXiv preprint arXiv:2108.04417, 2021.
- [15] OpenAI. Introducing ChatGPT[J]. OpenAI blog, 2022. <https://openai.com/blog/chatgpt>.
- [16] TOUVRON H, LAVRIL T, IZACARD G, et al. Llama: Open and efficient foundation language models[J]. arXiv preprint arXiv:2302.13971, 2023.
- [17] DU Z, QIAN Y, LIU X, et al. GLM: General Language Model Pretraining with Autoregressive Blank Infilling[C]//Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2022: 320-335.
- [18] HOU X, LIU J, LI J, et al. Ciphergpt: Secure two-party gpt inference[J]. Cryptology ePrint Archive, 2023.
- [19] VEPAKOMMA P, GUPTA O, SWEDISH T, et al. Split learning for health: Distributed deep learning without sharing raw patient data[J]. arXiv preprint arXiv:1812.00564, 2018.
- [20] POIROT M G, VEPAKOMMA P, CHANG K, et al. Split learning for collaborative deep learning in healthcare[J]. arXiv preprint arXiv:1912.12115, 2019.
- [21] ROTH H R, HATAMIZADEH A, XU Z, et al. Split-u-net: Preventing data leakage in split learning for collaborative multi-modal brain tumor segmentation[C]//International Workshop on Distributed, Collaborative, and Federated Learning. 2022: 47-57.
- [22] FAGBOHUNGBE O, REZA S R, DONG X, et al. Efficient Privacy Preserving Edge Intelligent Computing Framework for Image Classification in IoT[J]. IEEE Trans. Emerg. Top. Comput. Intell., 2022, 6(4): 941-956. <https://doi.org/10.1109/TETCI.2021.3111636>. DOI: 10.1109/TETCI.2021.3111636.
- [23] PALANISAMY K, KHIMANI V, MOTI M H, et al. SplitEasy: A Practical Approach for Training ML models on Mobile Devices[C]//HotMobile '21: The 22nd International Workshop on Mobile Computing Systems and Applications, Virtual Event, United Kingdom, February 24-26, 2021. ACM,

- 2021: 37-43. <https://doi.org/10.1145/3446382.3448362>. DOI: 10.1145/3446382.3448362.
- [24] KODA Y, PARK J, BENNIS M, et al. Communication-Efficient Multimodal Split Learning for mmWave Received Power Prediction[J]. IEEE Commun. Lett., 2020, 24(6): 1284-1288. <https://doi.org/10.1109/LCOMM.2020.2978824>. DOI: 10.1109/LCOMM.2020.2978824.
- [25] ABUADBBA S, KIM K, KIM M, et al. Can We Use Split Learning on 1D CNN Models for Privacy Preserving Training?[C]//ASIA CCS '20: The 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, October 5-9, 2020. ACM, 2020: 305-318. <https://doi.org/10.1145/3320269.3384740>. DOI: 10.1145/3320269.3384740.
- [26] VEPAKOMMA P, SINGH A, GUPTA O, et al. NoPeek: Information leakage reduction to share activations in distributed deep learning[C]//2020 International Conference on Data Mining Workshops (ICDMW). 2020: 933-942.
- [27] HE Z, ZHANG T, LEE R B. Model inversion attacks against collaborative inference[C]//BALENSON D. Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, December 09-13, 2019. ACM, 2019: 148-162. <https://doi.org/10.1145/3359789.3359824>. DOI: 10.1145/3359789.3359824.
- [28] LUO X, WU Y, XIAO X, et al. Feature Inference Attack on Model Predictions in Vertical Federated Learning[C]//37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021. IEEE, 2021: 181-192. <https://doi.org/10.1109/ICDE51399.2021.00023>. DOI: 10.1109/ICDE51399.2021.00023.
- [29] KRAMER M A. Nonlinear principal component analysis using autoassociative neural networks[J]. AIChE journal, 1991, 37(2): 233-243.
- [30] BALDI P. Autoencoders, unsupervised learning, and deep architectures[C]//Proceedings of ICML workshop on unsupervised and transfer learning. 2012: 37-49.
- [31] PASQUINI D, ATENIESE G, BERNASCHI M. Unleashing the Tiger: Inference Attacks on Split Learning[C]//CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021. ACM, 2021: 2113-2129. <https://doi.org/10.1145/3460120.3485259>. DOI: 10.1145/3460120.3485259.
- [32] RAUBER P E, FADEL S G, FALCÃO A X, et al. Visualizing the Hidden Activity of Artificial Neural Networks[J]. IEEE Transactions on Visualization and Computer Graphics, 2017, 23(1): 101-110. DOI: 10.1109/TVCG.2016.2598838.
- [33] PEZZOTTI N, HÖLLT T, VAN GEMERT J, et al. Deepeyes: Progressive visual analytics for designing deep neural networks[J]. IEEE transactions on visualization and computer graphics, 2017, 24(1): 98-108.
- [34] CANTAREIRA G D, ETEMAD E, PAULOVICH F V. Exploring neural network hidden layer activity using vector fields[J]. Information, 2020, 11(9): 426.
- [35] LIU J, LYU X. Clustering label inference attack against practical split learning[J]. arXiv preprint arXiv:2203.05222, 2022.
- [36] LIU J, LYU X. Distance-Based Online Label Inference Attacks Against Split Learning[C]//ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2023: 1-5. DOI: 10.1109/ICASSP49357.2023.10096955.
- [37] FU C, ZHANG X, JI S, et al. Label Inference Attacks Against Vertical Federated Learning[C]//BUTLER K R B, THOMAS K. 31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022. USENIX Association, 2022: 1397-1414. <https://www.usenix.org/conference/usenixsecurity22/presentation/fu-chong>.
- [38] LI O, SUN J, YANG X, et al. Label Leakage and Protection in Two-party Split Learning[C]//The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022. <https://openreview.net/forum?id=cOtBRgsf2fO>.
- [39] ERDOGAN E, KÜPÇÜ A, ÇIÇEK A E. UnSplit: Data-Oblivious Model Inversion, Model Stealing, and Label Inference Attacks against Split Learning[C]//HONG Y, WANG L. Proceedings of the 21st Workshop on Privacy in the Electronic Society, WPES2022, Los Angeles, CA, USA, 7 November 2022. ACM, 2022: 115-124. <https://doi.org/10.1145/3559613.3563201>. DOI: 10.1145/3559613.3563201.
- [40] MORRIS J X, KULESHOV V, SHMATIKOV V, et al. Text Embeddings Reveal (Almost) As Much

- As Text[C]//EMNLP 2023. 2023.
- [41] SUN J, YANG X, YAO Y, et al. Label Leakage and Protection from Forward Embedding in Vertical Federated Learning[J]. CoRR, 2022, abs/2203.01451. arXiv: 2203.01451. <https://doi.org/10.48550/arXiv.2203.01451>. DOI: 10.48550/ARXIV.2203.01451.
- [42] SZÉKELY G J, RIZZO M L, BAKIROV N K. MEASURING AND TESTING DEPENDENCE BY CORRELATION OF DISTANCES[J]. The Annals of Statistics, 2007, 35(6): 2769-2794.
- [43] SZÉKELY G J, RIZZO M L. Brownian distance covariance[J]. The annals of applied statistics, 2009: 1236-1265.
- [44] HUFFMAN D A. A method for the construction of minimum-redundancy codes[J]. Proceedings of the IRE, 1952, 40(9): 1098-1101.
- [45] GALLAGER R, VAN VOORHIS D. Optimal source codes for geometrically distributed integer alphabets (corresp.)[J]. IEEE Transactions on Information theory, 1975, 21(2): 228-230.
- [46] SATTLER F, WIEDEMANN S, MÜLLER K R, et al. Sparse binary compression: Towards distributed deep learning with minimal communication[C]//2019 International Joint Conference on Neural Networks (IJCNN). 2019: 1-8.
- [47] ZHOU S, WU Y, NI Z, et al. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients[J]. arXiv preprint arXiv:1606.06160, 2016.
- [48] BANNER R, HUBARA I, HOFFER E, et al. Scalable methods for 8-bit training of neural networks [J]. Advances in neural information processing systems, 2018, 31.
- [49] YANG J, SHEN X, XING J, et al. Quantization networks[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 7308-7316.
- [50] WEN W, XU C, YAN F, et al. Terngrad: Ternary gradients to reduce communication in distributed deep learning[J]. Advances in neural information processing systems, 2017, 30.
- [51] HARDT M, RECHT B, SINGER Y. Train faster, generalize better: Stability of stochastic gradient descent[C]//International conference on machine learning. 2016: 1225-1234.
- [52] GOYAL P, DOLLÁR P, GIRSHICK R, et al. Accurate, large minibatch sgd: Training imagenet in 1 hour[J]. arXiv preprint arXiv:1706.02677, 2017.
- [53] CHAUDHARI P, SOATTO S. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks[C]//6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. Open-Review.net, 2018. <https://openreview.net/forum?id=HyWrIgW0W>.
- [54] AJI A, HEAFIELD K. Sparse Communication for Distributed Gradient Descent[C]//EMNLP 2017: Conference on Empirical Methods in Natural Language Processing. 2017: 440-445.
- [55] CASTIGLIA T J, DAS A, WANG S, et al. Compressed-VFL: Communication-Efficient Learning with Vertically Partitioned Data[C]//CHAUDHURI K, JEGELKA S, SONG L, et al. Proceedings of Machine Learning Research: Proceedings of the 39th International Conference on Machine Learning: vol. 162. PMLR, 2022: 2738-2766. <https://proceedings.mlr.press/v162/castiglia22a.html>.
- [56] FU F, MIAO X, JIANG J, et al. Towards communication-efficient vertical federated learning training via cache-enabled local updates[J]. Proceedings of the VLDB Endowment, 2022, 15(10): 2111-2120.
- [57] CHEN X, LI J, CHAKRABARTI C. Communication and computation reduction for split learning using asynchronous training[C]//2021 IEEE Workshop on Signal Processing Systems (SiPS). 2021: 76-81.
- [58] AYAD A, RENNER M, SCHMEINK A. Improving the communication and computation efficiency of split learning for iot applications[C]//2021 IEEE Global Communications Conference (GLOBECOM). 2021: 01-06.
- [59] SHAMIR A. How to share a secret[J]. Communications of the ACM, 1979, 22(11): 612-613.
- [60] BEAVER D. Efficient multiparty protocols using circuit randomization[C]//Advances in Cryptology —CRYPTO' 91: Proceedings 11. 1992: 420-432.
- [61] PAILLIER P. Public-key cryptosystems based on composite degree residuosity classes[C]//International conference on the theory and applications of cryptographic techniques. 1999: 223-238.
- [62] BRAKERSKI Z. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP[C]//SAFAVI-NAINI R, CANETTI R. Lecture Notes in Computer Science: Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, Au-

- gust 19-23, 2012. Proceedings: vol. 7417. Springer, 2012: 868-886. [https://doi.org/10.1007/978-3-642-32009-5\\_50](https://doi.org/10.1007/978-3-642-32009-5_50). DOI: 10.1007/978-3-642-32009-5\_50.
- [63] FAN J, VERCAUTEREN F. Somewhat Practical Fully Homomorphic Encryption[J]. IACR Cryptol. ePrint Arch., 2012: 144. <http://eprint.iacr.org/2012/144>.
- [64] BRAKERSKI Z, GENTRY C, VAIKUNTANATHAN V. (Leveled) fully homomorphic encryption without bootstrapping[J]. ACM Transactions on Computation Theory (TOCT), 2014, 6(3): 1-36.
- [65] CHEON J H, KIM A, KIM M, et al. Homomorphic Encryption for Arithmetic of Approximate Numbers[C]//TAKAGI T, PEYRIN T. Lecture Notes in Computer Science: Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I: vol. 10624. Springer, 2017: 409-437. [https://doi.org/10.1007/978-3-319-70694-8\\_15](https://doi.org/10.1007/978-3-319-70694-8_15). DOI: 10.1007/978-3-319-70694-8\_15.
- [66] YAO A C C. How to generate and exchange secrets[C]//27th annual symposium on foundations of computer science (Sfcs 1986). 1986: 162-167.
- [67] KOLESNIKOV V, SCHNEIDER T. Improved garbled circuit: Free XOR gates and applications[C] //Automata, Languages and Programming: 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II 35. 2008: 486-498.
- [68] ZAHUR S, ROSULEK M, EVANS D. Two halves make a whole: Reducing data transfer in garbled circuits using half gates[C]//Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II 34. 2015: 220-250.
- [69] GILAD-BACHRACH R, DOWLIN N, LAINE K, et al. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy[C]//International conference on machine learning. 2016: 201-210.
- [70] BOS J W, LAUTER K, LOFTUS J, et al. Improved security for a ring-based fully homomorphic encryption scheme[C]//Cryptography and Coding: 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings 14. 2013: 45-64.
- [71] HESAMIFARD E, TAKABI H, GHASEMI M. Cryptodl: Deep neural networks over encrypted data [J]. arXiv preprint arXiv:1711.05189, 2017.
- [72] CHABANNE H, DE WARGNY A, MILGRAM J, et al. Privacy-preserving classification on deep neural network[J]. Cryptology ePrint Archive, 2017.
- [73] ZHOU J, LI J, PANAOUSIS E, et al. Deep Binarized Convolutional Neural Network Inferences over Encrypted Data[C]//7th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2020 / 6th IEEE International Conference on Edge Computing and Scalable Cloud, EdgeCom 2020, New York City, NY, USA, August 1-3, 2020. IEEE, 2020: 160-167. <https://doi.org/10.1109/CSCLOUD-EDGECom49738.2020.00035>. DOI: 10.1109/CSCLOUD-EDGECom49738.2020.00035.
- [74] ROUHANI B D, RIAZI M S, KOUSHANFAR F. Deepsecure: Scalable provably-secure deep learning[C]//Proceedings of the 55th annual design automation conference. 2018: 1-6.
- [75] RIAZI M S, SAMRAGH M, CHEN H, et al. {XONN}:{XNOR-based} oblivious deep neural network inference[C]//28th USENIX Security Symposium (USENIX Security 19). 2019: 1501-1518.
- [76] DEMMLER D, SCHNEIDER T, ZOHNER M. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation[C]//22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015. The Internet Society, 2015. <https://www.ndss-symposium.org/ndss2015/aby---framework-efficient-mixed-protocol-secure-two-party-computation>.
- [77] MOHASSEL P, ZHANG Y. Secureml: A system for scalable privacy-preserving machine learning [C]//2017 IEEE symposium on security and privacy (SP). 2017: 19-38.
- [78] LIU J, JUUTI M, LU Y, et al. Oblivious neural network predictions via minionn transformations[C] //Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. 2017: 619-631.
- [79] JUVEKAR C, VAIKUNTANATHAN V, CHANDRAKASAN A. {GAZELLE}: A low latency framework for secure neural network inference[C]//27th USENIX security symposium (USENIX

- security 18). 2018: 1651-1669.
- [80] MISHRA P, LEHMKUHL R, SRINIVASAN A, et al. Delphi: A cryptographic inference system for neural networks[C]//Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice. 2020: 27-30.
- [81] RATHEE D, RATHEE M, KUMAR N, et al. Cryptflow2: Practical 2-party secure inference[C]//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020: 325-342.
- [82] HUANG Z, LU W J, HONG C, et al. Cheetah: Lean and fast secure {Two-Party} deep neural network inference[C]//31st USENIX Security Symposium (USENIX Security 22). 2022: 809-826.
- [83] RATHEE D, RATHEE M, GOLI R K K, et al. Sirnn: A math library for secure rnn inference[C]//2021 IEEE Symposium on Security and Privacy (SP). 2021: 1003-1020.
- [84] RIAZI M S, WEINERT C, TKACHENKO O, et al. Chameleon: A hybrid secure computation framework for machine learning applications[C]//Proceedings of the 2018 on Asia conference on computer and communications security. 2018: 707-721.
- [85] MOHASSEL P, RINDAL P. ABY3: A mixed protocol framework for machine learning[C]//Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. 2018: 35-52.
- [86] WAGH S, GUPTA D, CHANDRAN N. SecureNN: 3-party secure computation for neural network training[J]. Proceedings on Privacy Enhancing Technologies, 2019.
- [87] KUMAR N, RATHEE M, CHANDRAN N, et al. Cryptflow: Secure tensorflow inference[C]//2020 IEEE Symposium on Security and Privacy (SP). 2020: 336-353.
- [88] KNOTT B, VENKATARAMAN S, HANNUN A, et al. Crypten: Secure multi-party computation meets machine learning[J]. Advances in Neural Information Processing Systems, 2021, 34: 4961-4973.
- [89] WAGH S, TOPLE S, BENHAMOUDA F, et al. Falcon: Honest-Majority Maliciously Secure Framework for Private Deep Learning[J]. Proceedings on Privacy Enhancing Technologies, 2021, 1: 188-208.
- [90] RYFFEL T, THOLONIAT P, POINTCHEVAL D, et al. AriaNN: Low-Interaction Privacy-Preserving Deep Learning via Function Secret Sharing[J]. Proceedings on Privacy Enhancing Technologies, 2021, 2022(1): 291-316.
- [91] DALSKOV A, ESCUDERO D, KELLER M. Secure Evaluation of Quantized Neural Networks[J]. Proceedings on Privacy Enhancing Technologies, 2020, 2020(4): 355-375.
- [92] WU Y, CAI S, XIAO X, et al. Privacy preserving vertical federated learning for tree-based models [J]. Proceedings of the VLDB Endowment, 2020, 13(12): 2090-2103.
- [93] FANG W, ZHAO D, TAN J, et al. Large-scale secure XGB for vertical federated learning[C]//Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2021: 443-452.
- [94] LU W J, HUANG Z, ZHANG Q, et al. Squirrel: A Scalable Secure {Two-Party} Computation Framework for Training Gradient Boosting Decision Tree[C]//32nd USENIX Security Symposium (USENIX Security 23). 2023: 6435-6451.
- [95] BUNN P, OSTROVSKY R. Secure two-party k-means clustering[C]//Proceedings of the 14th ACM conference on Computer and communications security. 2007: 486-497.
- [96] WU W, LIU J, WANG H, et al. Secure and efficient outsourced k-means clustering using fully homomorphic encryption with ciphertext packing technique[J]. IEEE Transactions on Knowledge and Data Engineering, 2020, 33(10): 3424-3437.
- [97] NIKOLAENKO V, IOANNIDIS S, WEINSBERG U, et al. Privacy-preserving matrix factorization [C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. 2013: 801-812.
- [98] KIM J, KOO D, KIM Y, et al. Efficient privacy-preserving matrix factorization for recommendation via fully homomorphic encryption[J]. ACM Transactions on Privacy and Security (TOPS), 2018, 21(4): 1-30.
- [99] HAO M, LI H, CHEN H, et al. Iron: Private inference on transformers[J]. Advances in neural information processing systems, 2022, 35: 15718-15731.

- [100] LI D, WANG H, SHAO R, et al. MPCFORMER: FAST, PERFORMANT AND PRIVATE TRANSFORMER INFERENCE WITH MPC[C]//The Eleventh International Conference on Learning Representations. 2022.
- [101] CHEN Y, MENG X, SHI Z, et al. SecureTLM: Private inference for transformer-based large model with MPC[J]. Information Sciences, 2024, 667: 120429.
- [102] AKIMOTO Y, FUKUCHI K, AKIMOTO Y, et al. Privformer: Privacy-preserving transformer with mpc[C]//2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P). 2023: 392-410.
- [103] DONG Y, LU W J, ZHENG Y, et al. Puma: Secure inference of llama-7b in five minutes[J]. arXiv preprint arXiv:2307.12533, 2023.
- [104] LU W J, HUANG Z, GU Z, et al. Bumblebee: Secure two-party inference framework for large transformers[J]. Cryptology ePrint Archive, 2023.
- [105] ZHANG Q, WANG C, WU H, et al. GELU-Net: A Globally Encrypted, Locally Unencrypted Deep Neural Network for Privacy-Preserved Learning[C]//Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. ijcai.org, 2018: 3933-3939. <https://doi.org/10.24963/ijcai.2018/547>. DOI: 10.24963/IJCAI.2018/547.
- [106] XIE P, WU B, SUN G. BAYHENN: Combining Bayesian Deep Learning and Homomorphic Encryption for Secure DNN Inference[C]//KRAUS S. Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019. ijcai.org, 2019: 4831-4837. <https://doi.org/10.24963/ijcai.2019/671>. DOI: 10.24963/IJCAI.2019/671.
- [107] WONG H W H, MA J P K, WONG D P H, et al. Learning Model with Error - Exposing the Hidden Model of BAYHENN[C]//BESSIÈRE C. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020. ijcai.org, 2020: 3529-3535. <https://doi.org/10.24963/ijcai.2020/488>. DOI: 10.24963/IJCAI.2020/488.
- [108] ZHOU J, ZHENG L, CHEN C, et al. Toward scalable and privacy-preserving deep neural network via algorithmic-cryptographic co-design[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2022, 13(4): 1-21.
- [109] CHEN C, ZHOU J, ZHENG L, et al. Vertically federated graph neural network for privacy-preserving node classification[J]. arXiv preprint arXiv:2005.11903, 2020.
- [110] FU F, XUE H, CHENG Y, et al. Blindfl: Vertical federated machine learning without peeking into your data[C]//Proceedings of the 2022 International Conference on Management of Data. 2022: 1316-1330.
- [111] QIU P, ZHANG X, JI S, et al. Your Labels are Selling You Out: Relation Leaks in Vertical Federated Learning[J]. IEEE Transactions on Dependable and Secure Computing, 2023, 20(5): 3653-3668. DOI: 10.1109/TDSC.2022.3208630.
- [112] JANNACH D, LUDEWIG M. When recurrent neural networks meet the neighborhood for session-based recommendation[C]//Proceedings of the eleventh ACM conference on recommender systems. 2017: 306-310.
- [113] KANG W C, MCAULEY J. Self-attentive sequential recommendation[C]//2018 IEEE international conference on data mining (ICDM). 2018: 197-206.
- [114] PARKHI O, VEDALDI A, ZISSERMAN A. Deep face recognition[C]//BMVC 2015-Proceedings of the British Machine Vision Conference 2015. 2015.
- [115] KARIMIREDDY S P, KALE S, MOHRI M, et al. Scaffold: Stochastic controlled averaging for federated learning[C]//International conference on machine learning. 2020: 5132-5143.
- [116] REDDI S J, CHARLES Z, ZAHEER M, et al. Adaptive Federated Optimization[C]//International Conference on Learning Representations. 2020.
- [117] TIBSHIRANI R. Regression shrinkage and selection via the lasso[J]. Journal of the Royal Statistical Society: Series B (Methodological), 1996, 58(1): 267-288.
- [118] WRIGHT J, YANG A Y, GANESH A, et al. Robust face recognition via sparse representation[J]. IEEE transactions on pattern analysis and machine intelligence, 2008, 31(2): 210-227.
- [119] YIN J, LIU Z, JIN Z, et al. Kernel sparse representation based classification[J]. Neurocomputing, 2012, 77(1): 120-128.
- [120] HOEFLER T, ALISTARH D, BEN-NUN T, et al. Sparsity in Deep Learning: Pruning and growth for

- efficient inference and training in neural networks[J]. *J. Mach. Learn. Res.*, 2021, 22: 241:1-241:124. <http://jmlr.org/papers/v22/21-0366.html>.
- [121] NEYSHABUR B, TOMIOKA R, SREBRO N. Norm-Based Capacity Control in Neural Networks [C]//GRÜNWALD P, HAZAN E, KALE S. *JMLR Workshop and Conference Proceedings: Proceedings of The 28th Conference on Learning Theory, COLT 2015*, Paris, France, July 3-6, 2015: vol. 40. JMLR.org, 2015: 1376-1401. <http://proceedings.mlr.press/v40/Neyshabur15.html>.
- [122] NEYSHABUR B, BHOJANAPALLI S, MCALLESTER D, et al. Exploring Generalization in Deep Learning[C]//GUYON I, von LUXBURG U, BENGIO S, et al. *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, December 4-9, 2017, Long Beach, CA, USA. 2017: 5947-5956. <https://proceedings.neurips.cc/paper/2017/hash/10ce03a1ed01077e3e289f3e53c72813-Abstract.html>.
- [123] GOUK H, FRANK E, PFAHRINGER B, et al. Regularisation of neural networks by enforcing Lipschitz continuity[J]. *Mach. Learn.*, 2021, 110(2): 393-416. <https://doi.org/10.1007/s10994-020-05929-w>. DOI: 10.1007/s10994-020-05929-w.
- [124] KRIZHEVSKY A, HINTON G, et al. Learning multiple layers of features from tiny images[J]. Master's thesis, Department of Computer Science, University of Toronto, 2009.
- [125] HE K, ZHANG X, REN S, et al. Deep Residual Learning for Image Recognition[C]//2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. IEEE Computer Society, 2016: 770-778. <https://doi.org/10.1109/CVPR.2016.90>. DOI: 10.1109/CVPR.2016.90.
- [126] BEN-SHIMON D, TSIKINOVSKY A, FRIEDMANN M, et al. Recsys challenge 2015 and the yoochoose dataset[C]//Proceedings of the 9th ACM Conference on Recommender Systems. 2015: 357-358.
- [127] AUER S, BIZER C, KOBILAROV G, et al. DBpedia: A Nucleus for a Web of Open Data[C]//ABERER K, CHOI K, NOY N F, et al. *Lecture Notes in Computer Science: The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*: vol. 4825. Springer, 2007: 722-735. [https://doi.org/10.1007/978-3-540-76298-0\\_52](https://doi.org/10.1007/978-3-540-76298-0_52). DOI: 10.1007/978-3-540-76298-0\\_52.
- [128] KIM Y. Convolutional Neural Networks for Sentence Classification[C]//MOSCHITTI A, PANG B, DAELEMANS W. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL. ACL, 2014: 1746-1751. <https://doi.org/10.3115/v1/d14-1181>. DOI: 10.3115/v1/d14-1181.
- [129] PENNINGTON J, SOCHER R, MANNING C D. Glove: Global vectors for word representation [C]//Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014: 1532-1543.
- [130] LE Y, YANG X. Tiny imagenet visual recognition challenge[J]. *CS 231N*, 2015, 7(7): 3.
- [131] TAN M, LE Q V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks[C]//CHAUDHURI K, SALAKHUTDINOV R. *Proceedings of Machine Learning Research: Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*: vol. 97. PMLR, 2019: 6105-6114. <http://proceedings.mlr.press/v97/tan19a.html>.
- [132] MAAS A L, HANNUN A Y, NG A Y, et al. Rectifier nonlinearities improve neural network acoustic models[C]//Proc. icml: vol. 30: 1. 2013: 3.
- [133] CHEN C, ZHOU J, ZHENG L, et al. Vertically Federated Graph Neural Network for Privacy-Preserving Node Classification[C]//Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022. 2022: 1959-1965. <https://doi.org/10.24963/ijcai.2022/272>. DOI: 10.24963/ijcai.2022/272.
- [134] KARIYAPPA S, QURESHI M K. ExPloit: Extracting Private Labels in Split Learning[C]//2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML). 2023: 165-175. DOI: 10.1109/SaTML54575.2023.00020.
- [135] BERTHELOT D, CARLINI N, GOODFELLOW I J, et al. MixMatch: A Holistic Approach to Semi-Supervised Learning[C]//Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Van-

- couver, BC, Canada. 2019: 5050-5060. <https://proceedings.neurips.cc/paper/2019/hash/1cd138d0499a68f4bb72bee04bbec2d7-Abstract.html>.
- [136] XU Y, DING J, ZHANG L, et al. DP-SSL: Towards Robust Semi-supervised Learning with A Few Labeled Samples[C]//Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual. 2021: 15895-15907. <https://proceedings.neurips.cc/paper/2021/hash/854d6fae5ee42911677c739ee1734486-Abstract.html>.
- [137] JIN P, LU L, TANG Y, et al. Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness[J]. Neural Networks, 2020, 130: 85-99. <https://www.sciencedirect.com/science/article/pii/S0893608020302392>. DOI: <https://doi.org/10.1016/j.neunet.2020.06.024>.
- [138] SAXENA A, PRASAD M, GUPTA A, et al. A review of clustering techniques and developments [J]. Neurocomputing, 2017, 267: 664-681. <https://doi.org/10.1016/j.neucom.2017.06.053>. DOI: [10.1016/j.neucom.2017.06.053](https://doi.org/10.1016/j.neucom.2017.06.053).
- [139] GRIFFITHS D J. Introduction to electrodynamics[Z]. 2005.
- [140] WU R, ZHOU J P, WEINBERGER K Q, et al. Does Label Differential Privacy Prevent Label Inference Attacks?[C]//RUIZ F J R, DY J G, van de MEENT J. Proceedings of Machine Learning Research: International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain: vol. 206. PMLR, 2023: 4336-4347. <https://proceedings.mlr.press/v206/wu23a.html>.
- [141] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324. DOI: [10.1109/5.726791](https://doi.org/10.1109/5.726791).
- [142] XIAO H, RASUL K, VOLLGRAF R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms[EB]. arXiv. 2017. <https://arxiv.org/abs/1708.07747>.
- [143] VAN DER MAATEN L, HINTON G. Visualizing data using t-SNE.[J]. Journal of machine learning research, 2008, 9(11).
- [144] RAJPURKAR P, ZHANG J, LOPYREV K, et al. SQuAD: 100,000+ Questions for Machine Comprehension of Text[J]. arXiv e-prints, 2016, arXiv:1606.05250: arXiv:1606.05250. arXiv: 1606.05250.
- [145] LIU S, CAO J, YANG R, et al. Long Text and Multi-Table Summarization: Dataset and Method[C] //The 2022 Conference on Empirical Methods in Natural Language Processing. 2023: 1-16.

## 作者简历