

隐私保护的高效安全三方稀疏数据计算

周丹钰¹⁾ 阎允雪¹⁾ 张建栋¹⁾ 蒋瀚^{1,2)} 徐秋亮^{1,2)}

¹⁾(山东大学软件学院 济南 250101)

²⁾(山东省软件工程重点实验室(山东大学) 济南 250101)

摘要 如今,大数据与人工智能技术发展迅猛,基于海量数据进行精确训练的机器学习模型及其应用推动了生产力的提升,但同时也带来了严重的数据安全与隐私泄露问题,这一问题促进了隐私保护机器学习的研究。在实际应用中,机器学习算法常常在稀疏数据集上进行运算,明文下的模型训练存在高效计算方法,可以充分利用数据稀疏性,提高计算效率。为了保护数据隐私而引入的密码技术,将稀疏数据转化为稠密数据,从而使高效的稀疏数据运算变得复杂。现有的对于安全稀疏数据计算的相关研究都涉及大量公钥密码操作,计算效率不高,并且主要考虑两方的场景。实际上,稀疏数据的计算可简化为非零位置上相应元素的计算。为了充分利用这一特性以提高效率,本文将稀疏向量乘法问题分为了过滤和乘法计算两个模块来处理,并在三方联合计算的场景下进行协议设计。首先,基于三方加法复制秘密分享以及伪随机置换技术构建过滤协议,该协议能够实现对向量元素的过滤,筛选出向量中非零位置对应的元素。随后,在过滤协议的基础上引入加法同态加密技术,对非零元素进行安全乘法计算,实现一个隐私保护的安全三方稀疏向量乘法协议,并在半诚实对手模型下,使用理想现实模拟范式证明了协议的安全性。最后,将隐私保护稀疏向量乘法协议应用到逻辑回归模型中,验证了其可用性。通过实验以及效率分析表明,相对于隐私保护稀疏矩阵乘法协议 CAESAR,本文所提出的协议将主要计算开销由 $O(n)$ 的密文运算次数,降低为 $O(m)$ 次,其中 n 是向量的维数, m 是向量中非零元素数量;在小批量的逻辑回归模型训练中,文本协议与通用安全多方计算框架 ABY3 相比有 10%~30% 的效率提升。

关键词 安全多方计算;隐私保护机器学习;秘密分享;稀疏向量乘法;隐私计算

中图法分类号 TP309 DOI 号 10.11897/SP.J.1016.2024.01179

Efficient Privacy-Preserving Secure Three-Party Sparse Data Computing

ZHOU Dan-Yu¹⁾ YAN Yun-Xue¹⁾ ZHANG Jian-Dong¹⁾ JIANG Han^{1,2)} XU Qiu-Liang^{1,2)}

¹⁾(School of Software, Shandong University, Jinan 250101)

²⁾(Key Laboratory of Software Engineering of Shandong Province (Shandong University), Jinan 250101)

Abstract Nowadays, with the rapid development of big data and artificial intelligence (AI) technology, machine learning models trained precisely on vast amounts of data and their applications have propelled an increase in productivity. However, such model training often necessitates collaboration among multiple data holders, which poses serious risks to data security and privacy, including the potential for data breaches. With the increasing awareness of private data privacy protection among the public, such collaboration encounters significant obstacles. Addressing the imperative need to establish connections among data islands while prioritizing privacy becomes a pressing concern. This challenge has spurred extensive research and development in the domain of privacy-preserving machine learning. In practical application scenarios, machine learning algorithms

typically perform operations on sparse datasets. This implies that the majority of elements in the data matrix are zeros, with a relatively small number of non-zero elements actively contributing to the calculation process. Taking advantage of this data sparsity during model training reduces the data processing load, thereby enhancing computational efficiency. Cryptographic techniques have been introduced to protect data privacy. While generic secret sharing techniques address privacy concerns, the process often converts sparse data into denser forms, complicating the initially efficient computations. The existing study on secure sparse data computation involves extensive use of public-key cryptographic operations, resulting in low computational efficiency. Also, they primarily focus on scenarios involving only two parties. In fact, the computation of sparse data can be simplified to the calculations of corresponding elements at their non-zero positions. To fully exploit this characteristic for efficiency gains, we divide the problem of sparse vector multiplication into two modules: filtering and multiplication. Consequently, we propose a privacy-preserving multiplication protocol for sparse vectors in a scenario involving three participants. Firstly, a filtering protocol is constructed based on the techniques of three-party additive replication secret sharing and pseudo-random permutation. Given specific filtering rules and the vectors to be processed, this protocol can filter the vectors according to the rules, sieving out the required vector elements. Therefore, filtering rules can be formulated based on the sparsity characteristics of vectors to sieve out elements at non-zero positions. Subsequently, we just need to perform multiplication operations on these selected elements to complete the multiplication calculations for sparse vectors. By introducing additive homomorphic encryption technology, a privacy-preserving three-party sparse vector multiplication protocol is efficiently implemented based on the filtering protocol. We have proven the security of the protocol using the ideal/real simulation technology under the semi-honest adversary model, which ensures that apart from the output, no other private information is leaked throughout the entire process. Finally, the privacy-preserving sparse vector multiplication protocol is applied to the logistic regression model to verify its feasibility. The experiments and efficiency analysis show that, compared with the privacy-preserving sparse matrix multiplication protocol CAESAR, the proposed protocol in this paper reduces the main computing overhead from $O(n)$ ciphertext operations to $O(m)$ times, where n is the dimension of the vector, and m is the number of non-zero elements in the vector. In mini-batch logistic regression model training, our protocol demonstrates an efficiency improvement of 10%–30% compared to the general secure multi-party computation framework ABY3.

Keywords secure multi-party computation; privacy-preserving machine learning; secret sharing; sparse vector multiplication; privacy computing

1 引 言

随着《中华人民共和国数据安全法》、《网络数据安全管理条例》等法规的相继出台,社会对于信息安全与隐私保护的意识也逐步提升。基于海量数据进行精确模型训练的传统机器学习,往往需要多个数据持有方的合作,并且在训练过程中数据都是直接以明文的形式进行传递,存在严重的隐私泄露风险。随着大众对私有数据隐私保护意识的增强,这样的

合作面临巨大的阻碍。如何在合法合规的前提下为数据孤岛搭建桥梁成为了一个亟需解决的问题。

针对上述问题,隐私保护机器学习技术应运而生,研究者使用安全多方计算、差分隐私、同态加密和联邦学习等技术,在不同实体之间实现更广泛的数据共享,在保证数据信息安全性的同时,构建面向敏感数据的训练模型,从而在医疗保健、金融和社交媒体分析等领域带来新的见解和突破。

在许多机器学习的现实应用场景中,用于训练的数据集在表示上普遍具有稀疏性的特征,尤其是

在图像分析和自然语言处理等领域。在图像分析领域,稀疏的图像可以通过一个稀疏矩阵简洁地描述,其中非零元素直接表示出了其本质属性,方便之后的图像处理操作。在自然语言处理领域,稀疏性在数据集中更为常见,Pang 和 Lee^[1]公开的 Movie Review 数据集中包含了 2000 条影评,每条数据有超过 9.5 万维的特征,但其中仅有 100 维左右的特征元素非零,占比约 0.15%;20Newsgroups 数据集^[2]由 9051 个具有 10^5 个维度的特征向量组成,但平均每个特征向量中的非零元素占比不到 0.1%;Scikit-Learn 库中也自带了很多实用的数据集,包括用于文本分类的数据集,而其中非零特征占比也在 5% 之下。

用于机器学习训练和分析的数据大多是稀疏数据,这意味着数据矩阵中的大部分元素都是零元素,真正在计算过程中发挥作用的非零元素的数量占比较小。因此,在现实数据集上的机器学习可以利用数据稀疏性,减少需要处理的数据量,提高计算效率。

传统的隐私保护机器学习算法旨在保护敏感数据的隐私,允许多方联合训练或分析模型,通常忽略了数据集的稀疏性特征,对零元素和非零元素进行相同的处理,这样将产生额外的计算开销。比如在基于秘密分享^[3-4]实现的隐私保护机器学习中,联合计算的各个参与方将自己持有的隐私数据分成多份,并将份额分配给其他参与方,各方所有的计算都在份额上进行,整个过程除了输出外,不泄露任何其他信息。秘密分享技术解决了隐私保护问题,但该技术的引入也使得原本稀疏的数据经过分享变得稠密,如图 1 所示,初始向量是一个仅含有 3 个非零元素的稀疏向量,经过在 \mathbb{Z}_2^3 上的两方秘密分享,得到了两个稠密的向量份额,也就使得原来高效的稀疏向量计算变得更复杂。

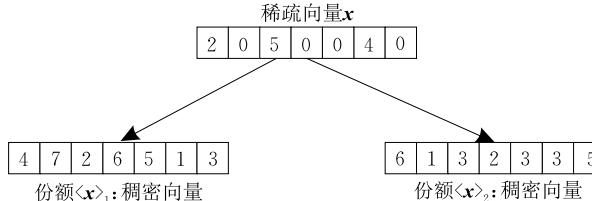


图 1 稀疏向量的秘密分享示意图

近几年,研究者开始关注隐私保护应用下的数据稀疏性问题,对该问题的探索也逐渐得到重视,成为隐私计算相关研究中不可或缺的一部分。然而,现有的安全稀疏矩阵计算协议都涉及大量公钥密码操作,计算效率并不高,并且以往的方案主要考虑两方的场景,对于更多参与方的研究有待拓展。

针对这种情况,本文研究的目标是在保护各个参与方数据隐私安全性的前提下,利用数据稀疏性的特点,降低计算复杂度,提升协议效率。本文提出了一个安全三方稀疏向量乘法协议,通过将联合计算的参与方数量扩展至三方,较为高效地实现一些功能函数,而这些功能函数在两方场景下可能会产生较大的开销甚至难以实现。

本文的贡献主要分为三个方面:

(1) 在三方场景下对稀疏数据的安全计算进行研究,基于三方加法复制秘密分享技术以及伪随机置换技术构建过滤协议,实现对向量元素的过滤,筛选出在稀疏向量乘法计算中实际具有意义的元素。

(2) 在过滤协议的基础上构建安全三方稀疏向量乘法协议,对于稀疏度(非零元素数量)为 m 的 n 维向量,充分利用其具有的稀疏性特征,将乘法协议的主要计算开销从 $O(n)$ 相关降低到了 $O(m)$ 相关,节省了计算资源。

(3) 将稀疏矩阵乘法协议应用到机器学习逻辑回归模型中,通过实验验证了其可用性。

本文第 2 节介绍与本文内容相关的研究工作;第 3 节介绍本文方案设计中涉及到的主要基础知识;第 4 节介绍两个核心协议;第 5 节对协议进行安全分析和证明;第 6 节分析本文协议的效率并做了对比;第 7 节将协议应用到逻辑回归上进行实验验证;第 8 节进行总结与展望。

2 相关工作

在本节中,我们将回顾经典的基于安全多方计算的隐私保护机器学习框架,并对现有的隐私保护背景下的数据稀疏性相关研究做简要介绍及总结。

2.1 多方安全机器学习框架

在现代机器学习中,各个领域各个程序都涉及个人数据。理论上,如果拥有不同数据的公司或者组织能够联合进行机器学习训练,将能得到更好的模型。但实际上,由于隐私保护、企业竞争、市场监管等原因,这样的合作难以推进,而安全多方计算^[5]的出现为此带来了转机。安全计算使相互不信任的各方能够在其私有输入上联合计算函数,并且除了函数本身的输出外,不会泄露过程中的其他信息。

2015 年,Demmler 等人^[6]提出了 ABY 混合协议框架,该框架有效地结合了基于算术共享、布尔共享和 Yao 混淆电路的安全计算方案,为通用机器学习模型的安全两方计算提供了可用的实践解决方案,且能够支持半诚实安全以及恶意安全要求。

ABY 方案中将相关密码操作进行了预算算, 使用基于扩展的不经意传输协议实现安全计算方案之间高效的转换, 并结合并行处理、多线程等技术进行了效率优化。

SecureML 由 Mohassel 和 Zhang^[7] 在 2017 年提出。在已有工作的基础上, 该框架研究了面向秘密共享的向量化计算, 在向量化场景下进一步优化了 ABY, 以提高共享向量和矩阵的乘法计算效率。同时, 该框架引入了秘密截断技术, 分析了秘密分享形式下的定点数截断问题, 实现对共享定点数的安全算术计算。此外, 该框架还对激活函数的计算进行了优化, 提出了适用于 MPC 场景的非线性函数替代方案。SecureML 实现了一种新的隐私保护机器学习的两方高效协议, 结合随机梯度下降思想进行安全的线性回归、逻辑回归以及神经网络的训练计算。

Mohassel 和 Rindal^[8] 在 2018 年提出了 ABY3, 将 ABY 和 SecureML 扩展到三服务器模式。数据所有者将数据秘密共享在三台服务器之间, 利用三方诚实大多数场景下的安全计算技术, 设计和优化新的协议, 实现算术共享、布尔共享以及 Yao 混淆电路之间的高效转换。针对三方共享定点小数运算问题, ABY3 提供了两种新的方案, 分别在通信量和通信轮数上进行了不同的权衡。三个参与方的引入打破了两方安全计算场景下的技术壁垒, 使得该框架具有更好的效率。

2019 年, Li 等人^[9] 提出了在四服务器模式下工作的 PrivPy, 这是一个为大规模数据挖掘任务设计的安全多方计算框架。PrivPy 使用可扩展的接口来解耦前端和后端, 可以适应多种语言和后端技术。通过将灵活的 Python 接口与 MPC 后端相结合, 其实现了与基于 Python 的机器学习和数据分析工作的集成, 使开发人员不需要掌握完备的安全多方计算技术, 也可以进行隐私保护机器学习的开发。同时, 添加扩展接口可以支持多个 MPC 后端, 在不同的安全假设和性能之间进行权衡, 以充分利用性能优化的 MPC 技术。

上述方案的基本思想都是在多个参与方(服务器)之间秘密共享数据(模型)。在这种情况下, 即使数据集具有稀疏特性, 数据共享技术的使用也将破坏其稀疏结构, 使得系统无法利用数据稀疏的特点进行优化并扩展到高维数据。

2.2 隐私保护应用中的数据稀疏性研究

数据稀疏性是大部分现实世界数据集的共同特征, 隐私保护机器学习已得到广泛研究, 对稀疏数据的探索也在逐渐跟上步伐, 国内外学者开始考虑这

一问题, 并展开了相关研究。

Nikolaenko 等人^[10] 针对推荐系统算法问题提出了一种隐私保护协同过滤协议, 推荐者可以在不了解用户评级的情况下完成推荐。该协议公开了用户评价的电影数量的上限, 使用不经意的排序网络来利用数据的稀疏性特征, 以实现效率提升; GraphSC^[11] 是一个支持图形并行计算模型的安全计算框架, 该框架的设计关键同样依赖于不经意的排序, 能够在隐藏处理器之间通信模式的同时, 高效地执行稀疏图的计算; Laud^[12] 给出了另一种在稀疏图上使用不经意排序的安全多方计算的应用, 该方案适用于三方诚实大多数的模型场景。上述这些工作都依赖于不经意的排序网络^[13] 并且仅适用于特定的任务优化。

Schoppmann 等人^[14] 第一次系统地考虑了隐私保护机器学习背景下的数据稀疏性问题。明文数据上的稀疏线性代数算法依赖于适当的数据结构, 例如 CSR 结构。借鉴这种思想, 在安全多方计算场景下, Schoppmann 等人定义了一个类似的抽象表示: ROOM^[14], 用来访问稀疏结构中的元素。在利用 ROOM 作为构建块的基础上, 设计了基本的稀疏线性代数运算协议, 如 Gather、Scatter、稀疏矩阵乘法及多种变体, 并在其之上构建更高级别的非参数模型(k 近邻和朴素贝叶斯分类)和参数模型(逻辑回归)的两方安全协议, 实现对高维数据集的安全分析。该方案侧重于设计一种底层的安全稀疏线性运算协议, 使得其可应用于所有需要线性运算的机器学习模型。然而, 这篇文章的解决方案仅适用于列稀疏或行稀疏的数据。此外, ROOM 的设计依赖通用安全多方计算(MPC)协议来实现所需的安全计算, 通常会带来大量的计算开销, 并且不可避免地会导致不可忽略的通信轮数。当涉及矩阵乘法时, ROOM 仍然需要密码学技术来生成 Beaver^[15] 预计算三元组, 也限制了这种方案的训练效率。

Chen 等人^[16] 提出了一种稀疏矩阵与稠密矩阵相乘的安全方案 CAESAR。考虑到工业上的实用性, 该文的工作专注于数据垂直分布的场景, 各个参与方拥有共同的样本实体(如不同的公司可能拥有相同的客户群体), 但各自持有的样本特征不同(如银行拥有客户的账户数据, 而电商平台拥有该客户的交易数据), 并以此为背景来设计协议。通过巧妙结合同态加密和秘密分享这两个密码学技术, Chen 等人设计了一个安全两方稀疏矩阵乘法协议, 并在此基础上构建安全的大规模稀疏逻辑回归模型。其核心思想是利用加法同态加密计算稀疏明文矩阵与

来自客户端的任意加密矩阵的乘法,然后将密文结果秘密地共享给各个参与方,以进行后续的计算.其方案的实现采用了分布式集群计算,用来分摊单个设备的计算开销.

Cui 等人^[17]将安全计算技术引入社交推荐,并提出了 S³Rec,一个稀疏感知的安全跨平台社交推荐框架.该框架可以整合社交平台上的稀疏社交数据以提高评分平台的推荐性能,同时保护两个平台的数据隐私.他们将同态加密和现有的私有信息检索^[18](PIR)技术结合起来,根据数据集的稀疏度是否敏感,设计了两种安全稀疏矩阵乘法协议:当数据集稀疏度可公开时,S³Rec 依赖 Beaver 预计算三元组实现快速矩阵乘法;当数据集稀疏度是隐私数据时,S³Rec 使用私有信息检索(PIR)作为底层技术来获得两个矩阵之间的交集,结合同态加密技术,实现两个稀疏矩阵之间的乘法.由于 S³Rec 致力于探索安全跨平台社交推荐中的数据稀疏性,因此它具有高度的场景特定性,并不适用于通用的机器学习算法模型.

Xu 等人提出了新名词:安全稀疏内积^[19](S-SIP),并设计了两种具体的方案来实现.其主要思想是以混淆布隆过滤器^[20](GBF)和私有信息检索(PIR)作为基石,仔细构建协议模块将这两种密码学技术进行融合,以实现高效的安全稀疏运算.该工作的第一种方案在离线情况下需要线性通信开销,而第二种方案具有次线性开销,但依赖于计算成本更高的安全计算工具.

以上的工作一部分是针对特定场景而特别设计,适用于某些具体的隐私保护应用.另一部分组合使用繁重的通用安全多方计算工具,虽然可行,但可能会产生不必要的开销,效率有待进一步提升.从现状来看,为实际应用设计一个高效的安全稀疏计算协议还有路可走.

3 预备知识

3.1 秘密分享

秘密分享(Secret Sharing)是信息安全和数据保密中的重要手段,也是安全多方计算和联邦学习等领域的一个基础应用技术.秘密共享将秘密进行分割,并在参与者中进行分享,每个参与方都拿到一份秘密的份额,以实现所有成员共同掌握秘密,但参与方自身对秘密一无所知.

两方加法秘密分享(2-out-of-2).想要在 P_A 和 P_B 之间分享环 \mathbb{Z}_{2^k} 上的一个元素 x (即 $x \in \mathbb{Z}_{2^k}$),那

么,在 \mathbb{Z}_{2^k} 上均匀随机地选取一个随机元素 r (即 $r \in \mathbb{Z}_{2^k}$),并且把 r 作为份额 $\langle x \rangle_A$ 发送给 P_A ,然后,计算 $x - r \bmod 2^k$ 作为份额 $\langle x \rangle_B$ 发送给 P_B .这样, P_A 和 P_B 分别持有份额 $\langle x \rangle_A$ 和 $\langle x \rangle_B$,并且满足:(1) 两方的份额 $\langle x \rangle_A$ 和 $\langle x \rangle_B$ 具有随机性,仅各方自己的份额并不会泄露关于秘密 x 的任何隐私信息;(2) $x = \langle x \rangle_A + \langle x \rangle_B \bmod 2^k$, P_A 和 P_B 合作可以恢复出秘密 x .

三方加法复制秘密分享[21](2-out-of-3).想要在 P_A 、 P_B 和 P_C 之间分享环 \mathbb{Z}_{2^k} 上的一个元素 x (即 $x \in \mathbb{Z}_{2^k}$),那么,在 \mathbb{Z}_{2^k} 上均匀随机地选取三个元素 x_1 , x_2 , $x_3 \in \mathbb{Z}_{2^k}$,满足 $x = x_1 + x_2 + x_3 \bmod 2^k$,将 (x_1, x_2) 发送给 P_A , (x_2, x_3) 发送给 P_B , (x_3, x_1) 发送给 P_C .这样, P_A 、 P_B 和 P_C 分别持有份额 $\llbracket x \rrbracket_A = (x_1, x_2)$, $\llbracket x \rrbracket_B = (x_2, x_3)$, $\llbracket x \rrbracket_C = (x_3, x_1)$,并且满足:(1) 从各个参与方的视角,自己持有的份额都是独立且随机的,不会泄露秘密 x 的隐私信息;(2) 三方中的任意两方合作,都可以将秘密 x 恢复出来.

小数的秘密分享.无论是两方秘密分享还是三方秘密分享,都是针对环 \mathbb{Z}_{2^k} 上的元素,而实际机器学习模型应用中,普遍是以小数进行计算,并不能直接适用于上述分享方案.一种常见的解决方案是以可接受的精度损失为代价,采用定点小数表示法,其中定点数可以映射到环 \mathbb{Z}_{2^k} 中进行计算,例如,精度为 2^{-f} (小数部分最多有 f 位)的小数 x ,通过计算 $x' = 2^f x \bmod 2^k$ 将 x 映射到环 \mathbb{Z}_{2^k} 中的元素 x' ,再进行秘密分享及后续计算.

3.2 加法同态加密

加法同态加密^[22-23](Additive Homomorphic Encryption)是一种特殊的加密方式,允许数据在加密情况下实现同态加法运算.一个完整的加法同态方案由密钥生成、加密、加法同态计算和解密共 4 个部分构成.

密钥生成(Gen).输入安全参数 λ 和其他公共参数,输出一对公私钥对 (pk, sk) , pk 为加密密钥(公钥), sk 为解密密钥(私钥).并且把公钥 pk 公开给其他参与方.

加密(Enc).输入公钥 pk ,明文 x ,输出 $C_x := \text{Enc}_{pk}(x)$, C_x 是明文 x 对应的加密密文,“ $:=$ ”表示概率算法输出,即实现概率加密,以满足选择明文攻击安全性要求.

加法同态计算(AHO).直接在密文上进行计算,保持与明文计算的同态性.(1) 同态加: $C_x \oplus C_y = C_{x+y}$,给定 C_x 和 C_y ,分别是 x 和 y 的密文,想要计算 $x+y$ 的密文 C_{x+y} ,可以直接利用 C_x 和 C_y 进行密文“同态加”运算得到;(2) 同态标量乘: $a \otimes C_y =$

C_{ay} , 其中, a 为明文标量, C_y 为 y 的密文, 进行密文“同态标量乘”运算可以得到 ay 对应的密文。明文与密文的运算满足同态性质, 但不一定是同一种具体计算操作。

解密(Dec). 输入私钥 sk , 密文 C_x , 输出解密之后的明文 $x = \text{Dec}(sk; C_x)$.

4 安全三方稀疏向量乘法协议

无论是矩阵与矩阵的乘法运算, 还是矩阵与向量的乘法运算, 在实际的计算过程中, 最终都可以抽象成向量与向量的乘法问题。因此, 在研究稀疏矩阵的乘法运算时, 我们将其概括为稀疏向量的乘法问题, 也就是向量的内积问题进行研究。

稀疏向量的乘法可以分为两种: 稀疏向量与稀疏向量的乘法, 以及稀疏向量与稠密向量的乘法。本文主要面向的应用场景是机器学习中的参数模型, 在许多实际场景中, 由于数据收集时相关特征的缺失或者特征工程技术的引入, 各数据持有者(如参与联合训练的企业机构等)拥有的数据(机器学习模型的输入)一般为稀疏数据, 而模型参数由于随机初始化, 以及训练过程中基于大量数据的迭代更新, 一般将其看作为稠密向量。由此, 本文主要解决的是稀疏向量与稠密向量的乘法运算问题。

稀疏向量与稠密向量的乘法运算, 如图 2 所示, 考虑稀疏向量 x 与稠密向量 y 相乘, 其结果实际上与稀疏向量 x 中非零位置对应的元素密切相关。以图 2 为例, 向量 x 仅有两个非零元素 x_2 和 x_5 , $x \cdot y = \sum_{i=0}^7 x_i y_i = x_2 y_2 + x_5 y_5$, 不难看出, 稀疏向量 x 中非零元素的位置起着关键作用。

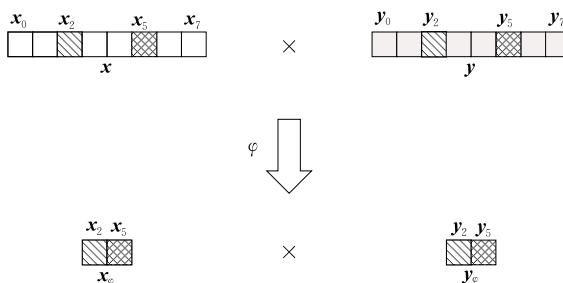


图 2 稀疏向量与稠密向量的乘法计算

由此, 可以定义一个过滤规则 φ , 使得其能够过滤掉零位置元素, 而将稀疏向量非零位置对应的信息保留。实际上, 该过滤规则 φ 是由稀疏向量持有者定义的, 并表示为向量下标索引的单射关系。如图 2

所示, φ 可以具体化为 $[2] \rightarrow [8]$ 的一种单射关系, 其中 $\varphi(0)=2, \varphi(1)=5$ 。这样, 向量 x 与 y 经过规则 φ 的过滤分别得到 x_φ 和 y_φ , 事实上, $x \cdot y = x_\varphi \cdot y_\varphi$, 初始的向量乘法 $x \cdot y$ 可以简化为计算 $x_\varphi \cdot y_\varphi$ 。很显然的, 两个向量的乘法运算从 8 维(初始向量维数)降到了 2 维(非零元素维数), 计算效率有显著提升。

因此, 针对稀疏向量和稠密向量的安全高效乘法计算问题, 本文方案主要分为两个模块: 第一个模块是过滤模块, 利用安全三方计算技术, 构建过滤协议, 实现隐私保护下的数据过滤, 筛选出对于两个向量乘法运算有意义的元素, 即得到 x_φ 和 y_φ ; 第二个模块是乘法模块, 对于过滤后的向量计算向量乘法 $x_\varphi \cdot y_\varphi$, 这里引入同态加密技术以满足隐私保护的要求。

为实现三方的安全计算, 我们延续了隐私保护机器学习方向已有工作中的秘密分享技术思想, 但在细节上做出了适应性调整:

(1) 为保持向量稀疏性特征, 我们规定不再对稀疏向量 x 进行分享, 仅由持有方私人拥有;

(2) 稠密向量 y 在三个参与方之间通过加法复制秘密分享技术进行分享。

因此, x_φ 可以由稀疏向量持有方本地计算得到。接下来, 我们需要重点解决的问题就是在保护过滤规则 φ (稀疏向量持有者方定义)和稠密向量 y (分享于三方)的隐私的前提下, 如何完成对 y 的过滤得到 y_φ 。

我们假设稀疏向量的稀疏度(非零元素数量)上限 m 可以公开, 这符合许多实际的应用场景的要求, 如文本训练数据集中文档的最大长度就是稀疏度上限, 而这是一个公开信息。

我们规定 $\langle x \rangle$ 表示 x 的两方加法分享份额, $\llbracket x \rrbracket$ 表示 x 的三方加法复制分享份额。

4.1 过滤协议

稠密向量的过滤是整个方案的重点内容。由此, 我们首先考虑稠密向量的过滤功能函数的实现。

基于三方加法复制秘密分享的份额分享形式, 我们将不经意置换网络^[24]的思想迁移到过滤函数的方案构建中。定义过滤规则 φ 为一种映射关系: 如果稀疏向量初始维度为 n , 稀疏度(非零元素数量)上限为 m , 则 φ 定义为 $[m] \rightarrow [n]$ 的一种单射, 该单射关系由稀疏向量持有方根据其所持有的稀疏向量的特征来确定。 φ 需要满足: 初始向量 $x = (x_1, \dots, x_n)$ 经过这种规则变换, 得到向量 $x_\varphi = (x_{\varphi(1)}, \dots, x_{\varphi(m)})$, 且 x_φ 是由 x 中的非零元素构成的向量。

对于复制秘密分享于三方的稠密向量 \mathbf{y} , 我们设计如下的功能函数表示向量元素的过滤.

函数 F_{filter} . 过滤功能函数.

参数: 参与方 P_A, P_B, P_C ; 向量维数 n ; 稀疏度 m

输入: P_A 输入单射 $\varphi: [m] \rightarrow [n]$ 和份额 $[\mathbf{y}]_A = (\mathbf{y}_1, \mathbf{y}_2)$,

$$\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{Z}_{2^k}^n$$

P_B 输入份额 $[\mathbf{y}]_B = (\mathbf{y}_2, \mathbf{y}_3), \mathbf{y}_2, \mathbf{y}_3 \in \mathbb{Z}_{2^k}^n$

P_C 无输入

输出: P_A 输出份额 $\langle \mathbf{y}_\varphi \rangle_A \in \mathbb{Z}_{2^k}^m$

P_B 无输出

P_C 输出份额 $\langle \mathbf{y}_\varphi \rangle_C \in \mathbb{Z}_{2^k}^m$,

其中, $\mathbf{y}_\varphi = (\mathbf{y}_{\varphi(1)}, \dots, \mathbf{y}_{\varphi(m)})$.

该理想功能函数有三个参与方 P_A, P_B, P_C , 这里, 我们假设过滤规则 φ 由 P_A 隐私持有 (即稀疏向量 \mathbf{x} 由 P_A 隐私持有), P_B, P_C 持有情况可类似分析. 待过滤的向量 \mathbf{y} 以三方复制秘密份额的形式共享于 P_A, P_B 和 P_C 之间. 功能函数得到 P_A 和 P_B 的输入分别为 $\varphi, [\mathbf{y}]_A$ 以及 $[\mathbf{y}]_B$, 可以恢复出向量 $\mathbf{y} = \mathbf{y}_1 + \mathbf{y}_2 + \mathbf{y}_3, \mathbf{y} \in \mathbb{Z}_{2^k}^n$. 然后利用规则 φ 对 \mathbf{y} 进行过滤: $\mathbf{y}_\varphi = (\mathbf{y}_{\varphi(1)}, \dots, \mathbf{y}_{\varphi(m)}), \mathbf{y}_\varphi \in \mathbb{Z}_{2^k}^m$. 接下来, 计算 \mathbf{y}_φ 的两份份额并将 $\langle \mathbf{y}_\varphi \rangle_A$ 发送给 P_A , 将 $\langle \mathbf{y}_\varphi \rangle_C$ 发送给 P_C . 最终, 功能函数以两份份额的形式输出满足过滤规则 φ 的向量 \mathbf{y}_φ . 可以看出, 除了稀疏度 m 这个信息外, 并没有其他隐私的泄露, 满足过滤功能模块的设计要求.

针对理想功能函数 F_{filter} , 我们对协议构建进行了以下分析:

P_A, P_B 和 P_C 三个参与方中, P_A 知晓过滤规则 φ , 可以将单射关系 $\varphi: [m] \rightarrow [n]$ 分解为两个关系的运算: $\varphi_1 \circ \varphi_0$, 其中, φ_0 是随机选取的 $[n] \rightarrow [n]$ 的置换关系. φ_0 选定后, 可根据关系的逆运算进一步确定 $\varphi_1 = \varphi \circ \varphi_0^{-1}: [m] \rightarrow [n]$, 之后可将 φ_0 和 φ_1 分别发送给 P_B 和 P_C . 这样, P_B 和 P_C 的组合运算, 如图 3 所示, 就可以实现关于规则 φ 的过滤.

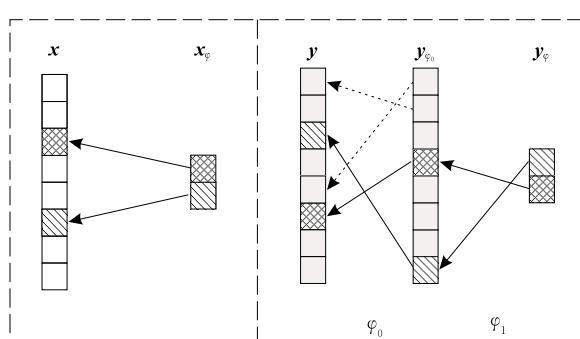


图 3 过滤示意图

据此, 过滤协议的实现过程如下所示:

协议 Π_{filter} . 过滤协议.

参数: 参与方 P_A, P_B, P_C ; 向量维数 n ; 稀疏度 m

输入: P_A 输入单射 $\varphi: [m] \rightarrow [n]$ 和份额 $[\mathbf{y}]_A = (\mathbf{y}_1, \mathbf{y}_2)$,

$$\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{Z}_{2^k}^n$$

P_B 输入份额 $[\mathbf{y}]_B = (\mathbf{y}_2, \mathbf{y}_3), \mathbf{y}_2, \mathbf{y}_3 \in \mathbb{Z}_{2^k}^n$

P_C 无输入

输出: P_A 输出份额 $\langle \mathbf{y}_\varphi \rangle_A \in \mathbb{Z}_{2^k}^m$

P_B 无输出

P_C 输出份额 $\langle \mathbf{y}_\varphi \rangle_C \in \mathbb{Z}_{2^k}^m$

协议过程:

1. P_A 均匀随机选取一个置换 $\varphi_0: [n] \rightarrow [n]$, 之后计算 $\varphi_1: [m] \rightarrow [n]$, 使得这两个映射关系的运算满足 $\varphi_1 \circ \varphi_0 = \varphi: [m] \rightarrow [n]$;
2. P_A 均匀随机选取一个随机向量 $\mathbf{r} \in \mathbb{Z}_{2^k}^n$, P_A 将 (φ_0, \mathbf{r}) 发送给 P_B , 将 φ_1 发送给 P_C ;
3. P_A 计算 $\mathbf{y}' = \mathbf{y}_1 + \mathbf{y}_2 + \mathbf{r}$, 根据 $\varphi: [m] \rightarrow [n]$ 计算得到 $\langle \mathbf{y}_\varphi \rangle_A = \mathbf{y}' = (\mathbf{y}'_{\varphi(1)}, \dots, \mathbf{y}'_{\varphi(m)})$;
4. P_B 计算 $\mathbf{y}'' = \mathbf{y}_3 - \mathbf{r}$, 之后根据 $\varphi_0: [n] \rightarrow [n]$ 对 \mathbf{y}'' 进行映射变换得到 $\mathbf{y}_{\varphi_0}'' = (\mathbf{y}_{\varphi_0(1)}'', \dots, \mathbf{y}_{\varphi_0(n)}'')$, 并将 \mathbf{y}_{φ_0}'' 发送给 P_C ;
5. 根据 $\varphi_1: [m] \rightarrow [n]$, P_C 对 \mathbf{y}_{φ_0}'' 再一次进行映射变换, 计算得到 $\langle \mathbf{y}_\varphi \rangle_C = (\mathbf{y}_{\varphi_0}''_{\varphi_1(1)}, \dots, \mathbf{y}_{\varphi_0}''_{\varphi_1(m)})$.

对于需要过滤的向量 \mathbf{y} , 由于其以三方复制秘密份额的形式被分享, 所以, 仅需 P_A 和 P_B 两份份额就可以恢复 \mathbf{y} . 如果 P_B 直接在原始份额 $[\mathbf{y}]_B$ 上进行关于 φ_0 的置换操作并将结果发送给 P_C , 而 P_C 拥有份额 $[\mathbf{y}]_C$, 那么这样将会导致 φ 和 \mathbf{y} 的信息泄露. 因此, 引入随机向量 \mathbf{r} , 在 P_A 和 P_B 之间重分享 $\mathbf{y}: \mathbf{y}' = \mathbf{y}_1 + \mathbf{y}_2 + \mathbf{r}; \mathbf{y}'' = \mathbf{y}_3 - \mathbf{r}$. P_B 对 \mathbf{y}'' 做关于 φ_0 的映射变换得到 \mathbf{y}_{φ_0}'' 并发送给 P_C , P_C 继续对 \mathbf{y}_{φ_0}'' 做关于 φ_1 的映射变换, 因为满足 $\varphi_1 \circ \varphi_0 = \varphi$, 这样两次映射变换之后, 就可以得到 \mathbf{y}' 关于 φ 的映射结果, 并将其作为 P_C 所持有的输出份额 $\langle \mathbf{y}_\varphi \rangle_C$. P_A 对 \mathbf{y}' 做关于 φ 的映射变换, 得到 P_A 的输出份额 $\langle \mathbf{y}_\varphi \rangle_A = \mathbf{y}'$.

相较于纯两方计算, 本文方案中第三方的引入不仅可以增加参与方的数量, 从原来的两方扩展到三方, 允许更多参与方联合计算, 更是充分利用了三方的安全计算优势, 以相较两方更低的计算和通信开销实现需求功能.

4.2 稀疏向量乘法协议

基于过滤功能函数的实现, 我们设计针对稀疏向量 \mathbf{x} 与稠密向量 \mathbf{y} 的乘法协议. 同样地, 不失一般性, 我们假设稀疏向量 \mathbf{x} 由 P_A 隐私持有, 则过滤规则 φ 也由 P_A 定义; 稠密向量 \mathbf{y} 被复制秘密分享在三

个参与方 P_A, P_B, P_C 中, 各方分别持有份额 $\llbracket \mathbf{y} \rrbracket_A = (\mathbf{y}_1, \mathbf{y}_2), \llbracket \mathbf{y} \rrbracket_B = (\mathbf{y}_2, \mathbf{y}_3), \llbracket \mathbf{y} \rrbracket_C = (\mathbf{y}_3, \mathbf{y}_1)$, 并且满足 $\mathbf{y}_1 + \mathbf{y}_2 + \mathbf{y}_3 = \mathbf{y}$.

首先, 定义乘法功能函数, 如下所示:

函数 $F_{S\text{-mult}}$. 稀疏向量乘法功能函数.

参数: 参与方 P_A, P_B, P_C ; 向量维数 n ; 稀疏度 m

输入: P_A 输入稀疏向量 $\mathbf{x} \in \mathbb{Z}_{2^k}^n$, 单射 $\varphi: [m] \rightarrow [n]$ 和份额 $\llbracket \mathbf{y} \rrbracket_A = (\mathbf{y}_1, \mathbf{y}_2), \mathbf{y}_1, \mathbf{y}_2 \in \mathbb{Z}_{2^k}^n$

P_B 输入份额 $\llbracket \mathbf{y} \rrbracket_B = (\mathbf{y}_2, \mathbf{y}_3), \mathbf{y}_2, \mathbf{y}_3 \in \mathbb{Z}_{2^k}^n$

P_C 无输入

输出: P_A 输出份额 $\langle z \rangle_A$

P_B 无输出

P_C 输出份额 $\langle z \rangle_C$, 其中 $z = \mathbf{x} \cdot \mathbf{y} \in \mathbb{Z}_{2^k}$

该功能函数以来自 P_A 的稀疏向量 \mathbf{x} 、单射 φ 和份额 $\llbracket \mathbf{y} \rrbracket_A$ 以及 P_B 的份额 $\llbracket \mathbf{y} \rrbracket_B$ 作为输入, 首先恢复向量 $\mathbf{y} = \mathbf{y}_1 + \mathbf{y}_2 + \mathbf{y}_3, \mathbf{y} \in \mathbb{Z}_{2^k}^n$, 然后计算 $z = \mathbf{x} \cdot \mathbf{y}, z \in \mathbb{Z}_{2^k}$, 并计算 z 的两份份额, 将 $\langle z \rangle_A$ 和 $\langle z \rangle_C$ 分别发送给 P_A 和 P_C . 最终, 该函数完成了对稀疏向量 \mathbf{x} 与稠密向量 \mathbf{y} 的乘法计算, 且将结果共享在 P_A 和 P_C 之间, 过程中没有泄露隐私信息.

如何对稀疏向量乘法理想功能函数进行协议实现? 接下来, 我们对协议设计的思路及过程进行具体分析.

正如前言所述, 稀疏向量乘法 $\mathbf{x} \cdot \mathbf{y}$ 本质上可以简化为计算 $\mathbf{x}_\varphi \cdot \mathbf{y}_\varphi$, 其实现总体上分为两个模块: 过滤模块和同态加密模块. 过滤模块可以通过调用过滤协议来实现, 完成对稠密向量 \mathbf{y} 的过滤, 过滤后的 \mathbf{y}_φ 最大限度地缩减了原始维度, 仅保留了乘法计算过程中将会用到的元素.

过滤完成之后, 接下来需要计算 $\mathbf{x}_\varphi \cdot \mathbf{y}_\varphi$, 也就是计算 $\mathbf{x}_\varphi \cdot \langle \mathbf{y}_\varphi \rangle_A + \mathbf{x}_\varphi \cdot \langle \mathbf{y}_\varphi \rangle_C$. 第一个乘积中, \mathbf{x}_φ 和 $\langle \mathbf{y}_\varphi \rangle_A$ 都是 P_A 持有的向量, P_A 本地即可完成运算; 第二个乘积中, \mathbf{x}_φ 由 P_A 持有, $\langle \mathbf{y}_\varphi \rangle_C$ 由 P_C 持有, 需要 P_A 和 P_C 两方共同进行安全乘法计算, 本方案中使用加法同态加密来实现并保证安全性.

由此, 利用第 4.1 节中的过滤协议以及加法同态加密算法, 我们设计了如下的协议对上述功能函数进行实现:

协议 $\Pi_{S\text{-mult}}$. 稀疏向量乘法协议.

参数: 参与方 P_A, P_B, P_C ; 向量维数 n ; 稀疏度 m ; 安全参数 λ

输入: P_A 输入稀疏向量 $\mathbf{x} \in \mathbb{Z}_{2^k}^n$, 单射 $\varphi: [m] \rightarrow [n]$ 和份额 $\llbracket \mathbf{y} \rrbracket_A = (\mathbf{y}_1, \mathbf{y}_2), \mathbf{y}_1, \mathbf{y}_2 \in \mathbb{Z}_{2^k}^n$

P_B 输入份额 $\llbracket \mathbf{y} \rrbracket_B = (\mathbf{y}_2, \mathbf{y}_3), \mathbf{y}_2, \mathbf{y}_3 \in \mathbb{Z}_{2^k}^n$

P_C 无输入

输出: P_A 输出份额 $\langle z \rangle_A$

P_B 无输出

P_C 输出份额 $\langle z \rangle_C$, 其中, $z = \mathbf{x} \cdot \mathbf{y} \in \mathbb{Z}_{2^k}$

协议过程:

1. P_C 调用加法同态密钥生成算法 $\text{Gen}(1^\lambda)$ 得到同态密钥对 (pk_C, sk_C) ;
2. P_A, P_B, P_C 三方调用过滤协议 $\Pi_{\text{filter}}(\varphi, \llbracket \mathbf{y} \rrbracket_A, \llbracket \mathbf{y} \rrbracket_B)$, P_A 和 P_C 分别得到 Π_{filter} 的输出 $\langle \mathbf{y}_\varphi \rangle_A$ 以及 $\langle \mathbf{y}_\varphi \rangle_C$;
3. P_A 本地计算 $\mathbf{x}_\varphi = (x_{\varphi(1)}, \dots, x_{\varphi(m)})$ 并均匀随机选取一个随机数 $r_z \in \mathbb{Z}_{2^k}$;
4. P_A 本地计算 $\langle z \rangle_A = \mathbf{x}_\varphi \cdot \langle \mathbf{y}_\varphi \rangle_A - r_z$;
5. P_C 使用同态加密密钥 pk_C 对 $\langle \mathbf{y}_\varphi \rangle_C$ 进行加密: $\mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C} := \text{Enc}_{pk_C}(\langle \mathbf{y}_\varphi \rangle_C)$, 并将 $\mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C}$ 发送给 P_A ;
6. P_A 在 \mathbf{x}_φ 和 $\mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C}$ 上进行内积乘法以及同态加运算, 得到 $\mathbf{C}_{\langle z \rangle_C} = \mathbf{x}_\varphi \otimes \mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C} \oplus \mathbf{C}_{r_z}$, 再将 $\mathbf{C}_{\langle z \rangle_C}$ 发送给 P_C ;
7. P_C 使用同态解密密钥 sk_C 对 $\mathbf{C}_{\langle z \rangle_C}$ 进行解密: $\langle z \rangle_C = \text{Dec}_{sk_C}(\mathbf{C}_{\langle z \rangle_C})$;
8. (当向量元素为定点小数时) P_A 和 P_C 分别对 $\langle z \rangle_A$ 和 $\langle z \rangle_C$ 进行安全截断.

P_C 用自己的加密密钥对 $\langle \mathbf{y}_\varphi \rangle_C$ 进行加密得到密文 $\mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C}$, 并发送给 P_A . P_A 在该密文上进行同态标量乘运算, 即计算 $\mathbf{x}_\varphi \otimes \mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C}$, 如果将这个结果直接返回给 P_C 解密, 在稀疏度 m 较小的情况下, P_C 有可能根据解密结果推理出 \mathbf{x}_φ 的相关信息. 因此, P_A 引入一个随机数 r_z 对结果进行随机化处理, 增加一次同态加运算: $\mathbf{x}_\varphi \otimes \mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C} \oplus \mathbf{C}_{r_z}$, 并在 $\mathbf{x}_\varphi \cdot \langle \mathbf{y}_\varphi \rangle_A$ 上减去 r_z , 其作用相当于对结果进行了重分享, 保证了正确性的同时也提高了安全性.

4.3 安全截断

在机器学习背景下, 通常向量元素都是小数形式. 对于小数部分最多有 f 位的定点小数 x 和 y , 我们将其转化为整数使之成为环 \mathbb{Z}_{2^k} 中的元素: $x' = 2^f x, y' = 2^f y$. 在 \mathbb{Z}_{2^k} 中计算乘积会得到 $z' = x' y'$, 注意到这时 z' 中有 $2f$ 位表示的是乘积结果的小数部分, 需要对 z' 的最后 f 位进行截断以保证其表示的是最多有 f 位小数的乘积结果.

SecureML 中的关于定点小数截断的定理证明了当 z' 以秘密份额的形式存在时, 这种截断技术依然有效. 参与方可以分别对自己的份额进行截断, 并且能够保证在足够大的环中, 截断后的份额重构结果最多只会对最低位造成 1 位的结果偏差.

因此, 协议 $\Pi_{S\text{-mult}}$ 中的乘法向量元素为 f 位定点小数时, P_A 和 P_C 最后还需要分别对自己的份额 $\langle z \rangle_A$ 和 $\langle z \rangle_C$ 的最后 f 位进行截断.

4.4 稀疏矩阵乘法协议

目前, 我们完成了对稀疏向量的乘法协议设计, 更进一步地, 如果 P_A 的输入是 $(d \times n)$ 维的稀疏矩

阵 \mathbf{X} 而非稀疏向量 \mathbf{x} . 这时, 对于稀疏矩阵的安全乘法计算, 以稀疏向量乘法协议为基础, 我们可以较容易地将其拓展为稀疏矩阵乘法协议. 我们假设矩阵为列稀疏矩阵, 矩阵的稀疏度 m 表示的是稀疏矩阵 \mathbf{X} 的非零列数.

协议过程及细节如下所示:

协议 $\Pi_{\text{SM-mult}}$. 稀疏矩阵乘法协议.

参数: 参与方 P_A, P_B, P_C ; 矩阵维数 $d \times n$; 稀疏度 m ; 安全参数 λ

输入: P_A 输入稀疏矩阵 $\mathbf{X} \in \mathbb{Z}_{2^k}^{d \times n}$ 、单射 $\varphi: [m] \rightarrow [n]$ 和份额 $[\mathbf{y}]_A = (\mathbf{y}_1, \mathbf{y}_2), \mathbf{y}_1, \mathbf{y}_2 \in \mathbb{Z}_{2^k}^n$

P_B 输入份额 $[\mathbf{y}]_B = (\mathbf{y}_2, \mathbf{y}_3), \mathbf{y}_2, \mathbf{y}_3 \in \mathbb{Z}_{2^k}^n$

P_C 无输入

输出: P_A 输出份额 $\langle \mathbf{z} \rangle_A$

P_B 无输出

P_C 输出份额 $\langle \mathbf{z} \rangle_C$, 其中, $\mathbf{z} = \mathbf{X} \cdot \mathbf{y} \in \mathbb{Z}_{2^k}^d$

协议过程:

1. P_C 调用加法同态密钥生成算法 $\text{Gen}(1^\lambda)$ 得到同态密钥对 (pk_C, sk_C) ;
2. P_A, P_B, P_C 三方调用过滤协议 $\Pi_{\text{filter}}(\varphi, [\mathbf{y}]_A, [\mathbf{y}]_B)$, P_A 和 P_C 分别得到 Π_{filter} 的输出 $\langle \mathbf{y}_\varphi \rangle_A$ 以及 $\langle \mathbf{y}_\varphi \rangle_C$;
3. P_A 本地计算 $\mathbf{X}_\varphi = (\mathbf{X}_{\varphi(1)}, \dots, \mathbf{X}_{\varphi(m)})$ 并均匀随机选取一个随机向量 $\mathbf{r}_z \in \mathbb{Z}_{2^k}^d$;
4. P_A 本地计算 $\langle \mathbf{z} \rangle_A = \mathbf{X}_\varphi \cdot \langle \mathbf{y}_\varphi \rangle_A - \mathbf{r}_z$;
5. P_C 使用同态加密密钥 pk_C 对 $\langle \mathbf{y}_\varphi \rangle_C$ 进行加密: $\mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C} := \text{Enc}_{pk_C}(\langle \mathbf{y}_\varphi \rangle_C)$, 并将 $\mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C}$ 发送给 P_A ;
6. P_A 将矩阵 \mathbf{X}_φ 乘以向量 $\mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C}$ 并加上随机向量得到 $\mathbf{C}_{\langle \mathbf{z} \rangle_C} = \mathbf{X}_\varphi \otimes \mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C} \oplus \mathbf{C}_{\mathbf{r}_z}$, 再将 $\mathbf{C}_{\langle \mathbf{z} \rangle_C}$ 发送给 P_C ;
7. P_C 使用同态解密密钥 sk_C 对 $\mathbf{C}_{\langle \mathbf{z} \rangle_C}$ 进行解密: $\langle \mathbf{z} \rangle_C = \text{Dec}_{sk_C}(\mathbf{C}_{\langle \mathbf{z} \rangle_C})$;
- * 8. (当元素为定点小数时) P_A 和 P_C 分别对 $\langle \mathbf{z} \rangle_A$ 和 $\langle \mathbf{z} \rangle_C$ 进行安全截断.

具体地, 与稀疏向量乘法协议 $\Pi_{\text{S-mult}}$ 相比, 稀疏矩阵乘法的协议 $\Pi_{\text{SM-mult}}$ 进行了以下调整:

协议第 3 步中, P_A 对于矩阵 \mathbf{X} 的非零列进行过滤, 得到 $\mathbf{X}_\varphi = (\mathbf{X}_{\varphi(1)}, \dots, \mathbf{X}_{\varphi(m)})$, 其中 $\mathbf{X}_{\varphi(i)}$ 表示的是矩阵 \mathbf{X} 的第 $\varphi(i)$ 列, 这样, 筛选出的 \mathbf{X}_φ 是一个 $(d \times m)$ 维的矩阵.

协议第 4 步中, P_A 本地计算矩阵乘法: $\langle \mathbf{z} \rangle_A = \mathbf{X}_\varphi \cdot \langle \mathbf{y}_\varphi \rangle_A - \mathbf{r}_z$, 其中, \mathbf{r}_z 为 P_A 在 $\mathbb{Z}_{2^k}^d$ 中均匀随机选取的一个随机向量.

第 5 步 P_A 收到 P_C 发送的密文 $\mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C}$ 后, 协议第 6 步中, P_A 在矩阵 \mathbf{X}_φ 和向量 $\mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C}$ 上进行同态标量乘法以及 \mathbf{r}_z 的同态加运算, 得到密文 $\mathbf{C}_{\langle \mathbf{z} \rangle_C} = \mathbf{X}_\varphi \otimes \mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C} \oplus \mathbf{C}_{\mathbf{r}_z}$, 再将 $\mathbf{C}_{\langle \mathbf{z} \rangle_C}$ 发送给 P_C , P_C 解密可以得到矩阵乘法的结果份额.

5 安全性分析

5.1 理想-现实模拟范式

半诚实敌手模型下的安全性^[25]. 非正式地来讲, 对于协议中的半诚实腐化方(敌手), 如果存在一个模拟器能够仅根据腐化方的输入和输出模拟出腐化方在协议执行过程中的视图, 且该视图与腐化方在真实世界执行协议所产生的视图完全不可区分, 那么我们认为这个协议在半诚实敌手下具有完美安全性; 如果两个视图计算不可区分, 那么我们认为这个协议在半诚实敌手下具有计算安全性.

以输入为 \mathbf{x} 执行一个协议 π , P_i 在协议执行过程中的视图可以表示为 $\text{View}_i^\pi(\mathbf{x})$, 该视图信息包括其私有输入 x_i 、随机带以及协议执行期间收到的所有消息.

定义 1. 令 $f: (\{0,1\}^*)^3 \rightarrow (\{0,1\}^*)^3$ 是一个三方概率功能函数, π 是一个协议. 如果存在一个概率多项式时间的算法 S , 对于每一个腐化方 $P_i, i \in \{1, 2, 3\}$, 以及每一个输入 $\mathbf{x} \in (\{0,1\}^*)^3$, 下式都能成立:

$$S_i(x_i, f_i(\mathbf{x})) \cong \text{View}_i^\pi(\mathbf{x}) \quad (1)$$

其中, \cong 表示计算不可区分, 那么我们可以说在半诚实敌手存在的情况下, 协议 π 对函数 f 的实现是计算安全的.

5.2 安全性证明

5.2.1 过滤协议

定理 1. 在不合谋的半诚实敌手模型下, 过滤协议 Π_{filter} 安全地实现了过滤理想功能函数 F_{filter} .

证明. 首先证明协议 Π_{filter} 的正确性.

该协议实现的是规则 φ 下, 对向量 \mathbf{y} 的过滤, 需要证明协议的输出 $\langle \mathbf{y}_\varphi \rangle_A$ 和 $\langle \mathbf{y}_\varphi \rangle_C$ 相加与 \mathbf{y}_φ 相等:

$$\begin{aligned} \mathbf{y}_\varphi &= (\mathbf{y}_1 + \mathbf{y}_2 + \mathbf{y}_3)_\varphi \\ &= (\mathbf{y}_1 + \mathbf{y}_2 + \mathbf{r})_\varphi + (\mathbf{y}_3 - \mathbf{r})_\varphi \\ &= \mathbf{y}'_\varphi + \mathbf{y}''_{\varphi_1 \circ \varphi_0} \\ &= \mathbf{y}'_\varphi + (\mathbf{y}''_{\varphi_0})_{\varphi_1} \\ &= \langle \mathbf{y}_\varphi \rangle_A + \langle \mathbf{y}_\varphi \rangle_C \end{aligned} \quad (2)$$

由此, 协议的正确性得以证明.

接下来, 分别对 P_A, P_B, P_C 可能腐化的情况进行证明:

P_A 被腐化. 协议中 P_A 没有获得任何交互信息, 因此, P_A 的视图仅有自己的私有输入, 模拟很容易实现.

P_B 被腐化. P_B 的视图除了自己的私有输入外, 还包括 P_A 发送过来的 $\varphi_0: [n] \rightarrow [n]$ 和 $\mathbf{r} \in \mathbb{Z}_{2^k}^n, \varphi_0$ 和

r 是由 P_A 均匀随机选取的,为模拟该视图,模拟器 S_B 从所有 $[n] \rightarrow [n]$ 的置换关系中均匀随机地选取一个 φ_0^* ,并且从 $\mathbb{Z}_{2^k}^n$ 均匀随机地选取一个 r^* ,由此,我们可以得到:

$$\text{View}_B^{\Pi_{\text{filter}}}(\varphi, [\mathbf{y}]_A, [\mathbf{y}]_B) = ([\mathbf{y}]_B, \varphi_0^*, r) \quad (3)$$

$$S_B([\mathbf{y}]_B, \perp) = ([\mathbf{y}]_B, \varphi_0^*, r^*) \quad (4)$$

不难看出, $\text{View}_B^{\Pi_{\text{filter}}}$ 和 S_B 的概率分布一致,即 $S_B \cong \text{View}_B^{\Pi_{\text{filter}}}$.

P_c 被腐化. P_c 获得的交互信息包括 P_A 发送的 $\varphi_1 : [m] \rightarrow [n]$ 和 P_B 发送的 $\mathbf{y}_{\varphi_0}'' = (y_{\varphi_0(1)}'', \dots, y_{\varphi_0(n)}'')$. 模拟器 S_C 从所有 $[m] \rightarrow [n]$ 的单射关系中均匀随机地选取一个 φ_1^* ,并且从 $\mathbb{Z}_{2^k}^n$ 均匀随机地选取一个 \mathbf{y}^* ,由此,我们可以得到:

$$\text{View}_C^{\Pi_{\text{filter}}}(\varphi, [\mathbf{y}]_A, [\mathbf{y}]_B) = (\varphi_1, \mathbf{y}_{\varphi_0}'') \quad (5)$$

$$S_C(\perp, \langle \mathbf{y}_\varphi \rangle_C) = (\varphi_1^*, \mathbf{y}^*) \quad (6)$$

对于 φ_1 我们可以观察到,如果先均匀随机地选定了 φ_1 ,那么可以有 $(n-m)!$ 种选法来确定 φ_0 ,并且,对于每一种 φ_1 的选择,与之对应的 φ_0 的集合并不相交.因此,协议中当 P_A 均匀随机选择 φ_0 时也会使得 φ_1 保持均匀分布.对于 $\mathbf{y}_{\varphi_0}'' = (\mathbf{y}_3 - \mathbf{r})_{\varphi_0}$, \mathbf{r} 是由 P_A 选取的随机向量,因此, \mathbf{y}_{φ_0}'' 也是均匀随机的.可得: $S_C \cong \text{View}_C^{\Pi_{\text{filter}}}$.

综上所述,在不合谋的半诚实对手模型下,过滤协议 Π_{filter} 计算安全地实现了过滤理想功能函数 F_{filter} .
证毕.

5.2.2 稀疏向量乘法协议

定理 2. 假设加法同态方案是选择明文攻击下计算不可区分的,那么在不合谋的半诚实对手模型下,稀疏向量乘法协议 $\Pi_{\text{S-mult}}$ 计算安全地实现了稀疏向量乘法理想功能函数 $F_{\text{S-mult}}$.

证明. 首先证明协议 $\Pi_{\text{S-mult}}$ 的正确性.

协议的正确性需要保证协议的输出 $\langle z \rangle_A$ 和 $\langle z \rangle_C$ 是向量 x 与向量 y 的内积份额,分析如下:

$$\begin{aligned} \mathbf{x} \cdot \mathbf{y} &= \mathbf{x}_\varphi \cdot \mathbf{y}_\varphi \\ &= \mathbf{x}_\varphi \cdot (\langle \mathbf{y}_\varphi \rangle_A + \langle \mathbf{y}_\varphi \rangle_C) \\ &= \mathbf{x}_\varphi \cdot \langle \mathbf{y}_\varphi \rangle_A - r_z + \mathbf{x}_\varphi \cdot \langle \mathbf{y}_\varphi \rangle_C + r_z \\ &= \langle z \rangle_A + \langle z \rangle_C \end{aligned} \quad (7)$$

协议的正确性得证.

接下来,分别对 P_A 、 P_B 、 P_C 可能腐化的情况进行证明:

P_A 被腐化. 协议中 P_A 收到的消息是由 P_C 发送的 $\mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C}$,所以, P_A 的视图包括其输入 $\varphi, x, [\mathbf{y}]_A$ 以及收到的 $\mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C}$,给定安全参数 λ ,模拟器 S_A 随机选择一个向量 $\mathbf{y}^* \in \mathbb{Z}_{2^k}^m$,然后使用 P_C 的公钥 pk_C 对其

加密得到 \mathbf{C}_{y^*} .由此,我们可以得到:

$$\text{View}_A^{\Pi_{\text{S-mult}}}(\varphi, x, [\mathbf{y}]_A, [\mathbf{y}]_B, sk_C, \lambda) = \mathbf{C}_{\langle \mathbf{y}_\varphi \rangle_C} \quad (8)$$

$$S_A(1^\lambda, \varphi, x, [\mathbf{y}]_A, \langle z \rangle_A) = \mathbf{C}_{y^*} \quad (9)$$

因为加法同态方案是选择明文攻击下计算不可区分的,所以 P_A 的视图和模拟器 S_A 的输出也是计算不可区分的,即 $S_A \cong \text{View}_A^{\Pi_{\text{S-mult}}}$.

P_B 被腐化. P_B 在协议 $\Pi_{\text{S-mult}}$ 中收到的消息就是在调用协议 Π_{filter} 中得到的消息,因此,模拟器 S_B 将 $[\mathbf{y}]_B$ 作为输入模拟函数 F_{filter} 即可.

P_C 被腐化. P_C 在协议中收到的消息可以分为两部分,一部分是在调用协议 Π_{filter} 中得到的消息,另一部分是 P_A 发送的密文 $\mathbf{C}_{\langle z \rangle_C}$. 模拟器 S_C 可以模拟功能函数 F_{filter} 得到第一部分的消息视图.对于第二部分的消息,给定 S_C 的输入为 $(sk_C, \langle z \rangle_C)$,模拟器 S_C 使用 P_C 的公钥 pk_C 加密 $\langle z \rangle_C$ 得到 $\mathbf{C}_{\langle z \rangle_C}^*$.由此,我们可以得到:

$$\text{View}_C^{\Pi_{\text{S-mult}}}(\varphi, x, [\mathbf{y}]_A, [\mathbf{y}]_B, sk_C, \lambda) = \mathbf{C}_{\langle z \rangle_C} \quad (10)$$

$$S_C(1^\lambda, sk_C, \langle z \rangle_C) = \mathbf{C}_{\langle z \rangle_C}^* \quad (11)$$

我们观察到, $\mathbf{C}_{\langle z \rangle_C}$ 和 $\mathbf{C}_{\langle z \rangle_C}^*$ 都是 $\langle z \rangle_C$ 在公钥 pk_C 下加密得到的密文,因此,对于 P_C 而言,它们不可区分,即 $S_C \cong \text{View}_C^{\Pi_{\text{S-mult}}}$.

综上所述,假设加法同态方案是选择明文攻击下计算不可区分的,则在不合谋的半诚实对手模型下,稀疏向量乘法协议 $\Pi_{\text{S-mult}}$ 对稀疏向量乘法理想功能函数 $F_{\text{S-mult}}$ 的实现具有计算安全性. 证毕.

协议 Π_{filter} 和 $\Pi_{\text{S-mult}}$ 都是在半诚实对手模型下安全的,这两个协议的基础都是三方加法复制秘密分享协议.在基于加法秘密分享的安全多方计算协议中,存在高效的编译器^[26-27],利用认证秘密共享方案,可以将半诚实对手模型下的协议编译为恶意对手模型下的协议.本文协议 Π_{filter} 和 $\Pi_{\text{S-mult}}$ 也可以使用同样的方式编译为恶意对手模型下安全的协议.

6 方案效率分析

通信开销和计算开销是分析方案效率的两个重要指标,因此,我们将针对通信和计算两个方面对文本的协议进行比较分析.在对比方案选择上,本文方案设计要求仅数据持有者本地持有稀疏数据,不再进行分享,相关工作中,CAESAR 方案与本文设计需求相似,选择其为对比方案;并且,本文的方案涉及三方,稠密向量的分享使用了三方复制秘密分享技术,因此引入 ABY3 方案进行对比.

6.1 通信开销

如表 1 所示, 我们首先从通信方面对比了本文协议的主要开销。

表 1 协议通信开销对比

协议	方案	通信开销	通信轮数
$\Pi_{S\text{-mult}}$	ABY3	$O(nk) \times 3$	2
	CAESAR	$O(nk)$	2
	本文	$O((n+m)k)$	3
$\Pi_{SM\text{-mult}}$	ABY3	$O(ndk) \times 3$	2
	CAESAR	$O((n+d)k)$	2
	本文	$O((n+m+d)k)$	3

注: 表中 k 表示比特位数; n 表示总特征维度, m 表示稀疏度(非零特征维度), $m \ll n$; d 表示矩阵行数;

ABY3 协议的每轮通信中三个参与方都进行了数据发送, 三方都产生通信开销, 以“ $\times 3$ ”形式表示;

CAESAR 为两方协议; ABY3 和本文协议都为三方协议。

过滤协议. 在通信方面, 该协议需要常数轮通信, 产生 $O(nk)$ 的通信开销。在计算安全的要求下, 协议中的 φ_0 和 r 可以通过使用相同的伪随机数发生器种子由 P_A 和 P_B 本地生成, 无需 P_A 在线发送, 如此可以节省通信成本, 并且将通信轮数降低到了 1 轮。

稀疏向量乘法协议. 该协议初始调用了过滤协议, 会产生 $O(nk)$ 的通信开销, 接下来使用同态加密技术计算 $x_\varphi \cdot y_\varphi$, 密文的发送又将产生 $O(mk)$ 的通信开销, 则需要的通信开销一共为 $O((n+m)k)$ 。通信轮数上, 过滤协议执行过程中最低有 1 轮通信, P_C 与 P_A 之间密文的相互发送需要 2 轮通信, 因此, 最低共需要 3 轮通信。

稀疏矩阵乘法协议. 通信开销方面, 与稀疏向量乘法协议相比, 协议第 6 步 P_A 发送给 P_C 的 $C_{(z)_C}$ 是 d 维密文向量, 与矩阵行数 d 相关, 产生 $O(dk)$ 的通信开销, 因此, 稀疏矩阵乘法协议的通信开销共为 $O((n+m+d)k)$ 。通信轮数与稀疏向量乘法协议相同, 为 3 轮。

6.2 计算开销

在计算开销上, 由于各个协议中涉及的运算种类较多, 难以用统一的标准进行衡量。因此, 我们通过实验来进行说明, 利用实验的运行时间作为分析协议计算开销的主要依据。

本文的评估实验均在 CPU 上进行单机模拟, 内存为 16 GB, 操作系统为 Ubuntu。

稀疏向量乘法协议. 由于现实生活中的数据具有稀疏且高维的特征, 因此, 我们针对高维向量进行评估。实验中, 对随机设置的 100 万维向量进行安全乘法计算, 比较各协议的计算时间。

实验结果如图 4(a)~(c) 所示。

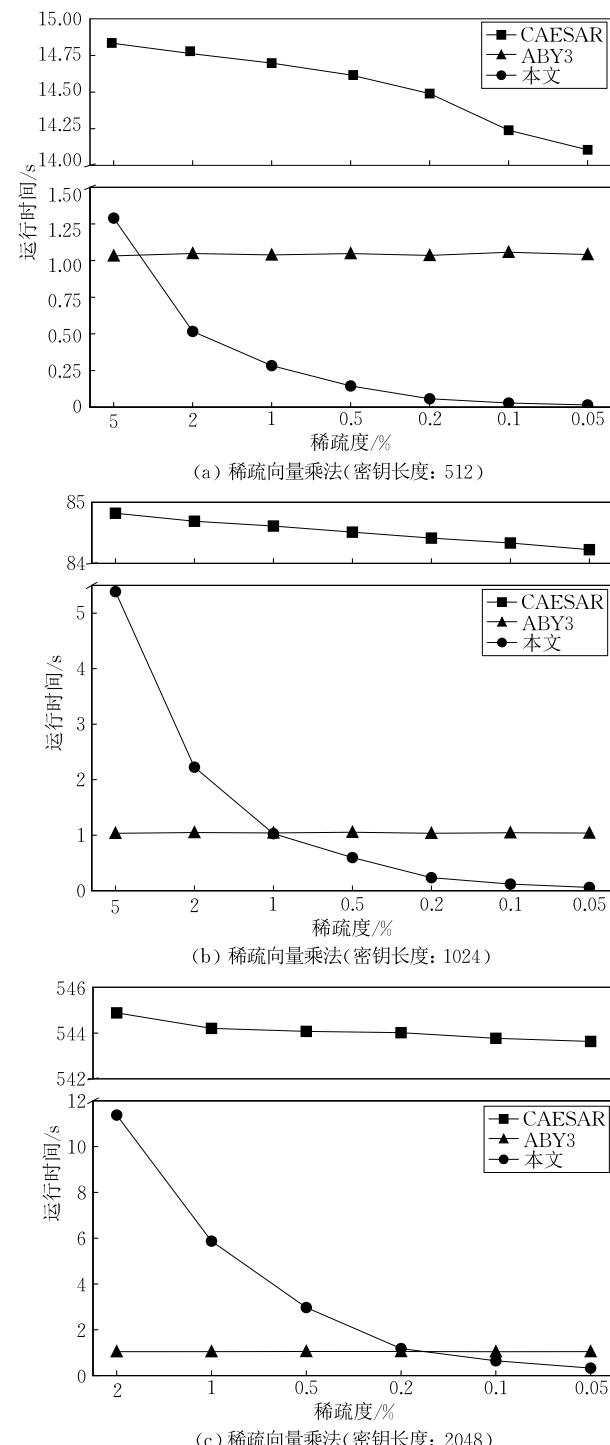


图 4 实验结果

我们对于不同稀疏度的向量以及不同的加密密钥长度(512 比特, 1024 比特, 2048 比特)都进行了实验, 对比实验结果可以发现。

(1) 本文协议的计算开销与数据稀疏度呈现正相关的关系, 计算开销随着数据稀疏度的降低而下降。理论上看, 同态加密运算构成了本文协议的主要计算开销, 而向量的稀疏程度决定了协议中同态加密的次数, 进而决定了协议的计算开销, 实验结果与

理论分析一致。

(2) 与 CAESAR 相比, CAESAR 对向量所有元素都进行了同态加密, 而本文协议仅对于过滤后的向量元素进行同态加密, 主要的计算开销由 $O(n)$ 的密文运算次数, 降低为 $O(m)$ 次 (n 是向量的维数, m 是向量的稀疏度), 实验也验证了本文协议的计算优势。但无论是本文协议还是 CAESAR 协议, 计算开销都主要由同态加密产生。事实上, 对于 CAESAR 的实现效果与其在文献[16]中的效果存在差距, 我们的实验均为单机模拟实现, 而在该论文中, 对于同态加密等计算, Chen 等人采用了分布式集群计算技术来分摊计算开销, 大幅度提升了计算效率, 由此可以推断, 如果本文协议的实现场景能够采用分布式集群计算技术, 那么计算效率还可以有进一步的提升。

(3) 与 ABY3 相比, 本文协议的计算效率在相关稀疏度下表现出优势。对于密钥长度为 512 比特以及 1024 比特的情况, 在向量稀疏度为 1% 时, 本文协议的计算时间已经优于 ABY3, 并且逐渐降低; 在密钥长度为 2048 比特时, 协议提供了更高的安全性, 相对地在计算上做出了牺牲, 但从 0.1% 的稀疏度开始, 其计算时间也逐渐优于 ABY3。

本文协议与 CASEAR 相比, 计算优势明显, 因此, 在后续的实验中, 不再与之进行对比。

稀疏矩阵乘法协议。受限于计算机内存, 我们在矩阵乘法实验中将矩阵 $\mathbf{X}(d \times n)$ 的列数 n 设定为 10 万维, 选择加密密钥长度为 1024 比特, 比较在稀疏度(矩阵非零列数)为 1% 时, 矩阵不同的行数 d 对乘法协议计算时间的影响。

实验结果如图 5 所示。

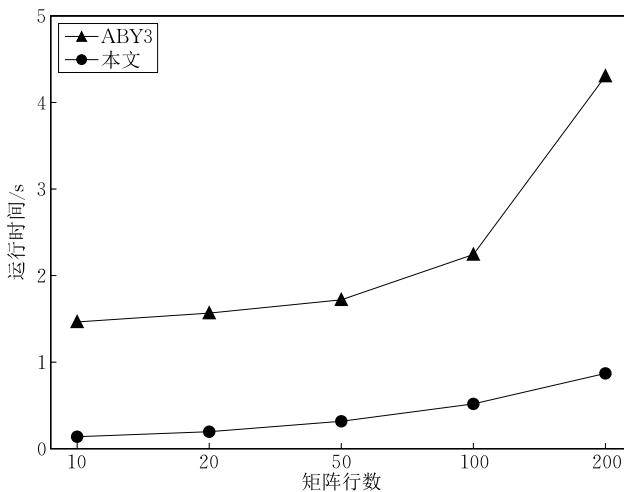


图 5 稀疏矩阵乘法

随着矩阵行数的增加, 本文协议与 ABY3 协议的计算开销都逐渐增加, 但总体上看, 在数据的稀疏

度为 1% 时, 本文矩阵乘法协议的计算时间仍然优于 ABY3 协议。

除计算优势外, 本文规定稀疏矩阵 \mathbf{X} 不再进行分享, 联系实际, 在机器学习的大规模稀疏数据模型训练中, 对于 $\mathbf{X} \cdot \mathbf{w}$ 的计算, 本文协议仅需在参与方中共享模型参数 \mathbf{w} , 三方协作更新参数 \mathbf{w} 。因此, 在存储开销方面, 除了稀疏矩阵持有者需要存储 \mathbf{X} 的完整信息以及 \mathbf{w} 的份额数据外, 其他参与方只需要存储模型参数 \mathbf{w} 的份额即可, 协议也降低了参与方的存储开销。

7 稀疏逻辑回归

本节中, 我们以逻辑回归模型为代表, 将稀疏矩阵乘法计算协议应用到该模型上, 以适应机器学习中输入数据集较为稀疏的场景。在真实数据集上开展实验, 进行安全逻辑回归计算。

7.1 逻辑回归

逻辑回归^[28]是经典的解决二分类问题的监督学习模型, 其核心思想是通过逻辑函数(Sigmoid 函数)来建模数据, 将输入特征映射到一个介于 0 和 1 之间的概率值, 并依此来进行分类决策。在实际中, 逻辑回归广泛用于医学诊断、金融风险分析、自然语言处理、图像处理等领域。

给定 d 个 n 维的训练数据样本 $\mathbf{x}_{i:i=1,\dots,d}$, 每个训练样本都有对应的样本标签为 $\mathbf{y}_{i:i=1,\dots,d}$, 监督学习的目的是学习合适的函数 f 使其能最大限度地对每个数据点进行拟合, 即意味着需要满足 $f(\mathbf{x}_i) \approx y_i$ 。逻辑回归假设输入特征与输出的对数概率之间存在线性关系, 函数 f 表示为

$$f(\mathbf{x}_i) = \text{Sigmoid}(\mathbf{x}_i \cdot \mathbf{w}) \quad (12)$$

Sigmoid 函数应用在输入特征 \mathbf{x}_i 和模型权重参数 \mathbf{w} 的内积结果上, 并将 Sigmoid 的函数输出作为模型的输出结果。

逻辑回归的训练过程, 实质上就是针对 \mathbf{w} 不断学习优化的过程, 使得模型输出 $f(\mathbf{x}_i)$ 能够不断接近真实标签值 y_i 。为了实现这一目标, 引入损失函数 $L(\mathbf{w})$, 该函数表示的是预测值 $f(\mathbf{x}_i)$ 与真实标签值 y_i 之间的“距离”, 那么整个模型训练其实就是在对 $L(\mathbf{w})$ 的最优化: $\arg \min_{\mathbf{w}} L(\mathbf{w})$, 意图找到某个 \mathbf{w} 值, 使得损失函数 $L(\mathbf{w})$ 最小。为保证 $L(\mathbf{w})$ 凸函数的特性, 逻辑回归的损失函数为交叉熵函数:

$$L_i(\mathbf{w}) = -y_i \log f(\mathbf{x}_i) - (1-y_i) \log(1-f(\mathbf{x}_i));$$

$$L(\mathbf{w}) = \frac{1}{d} \sum_{i=1}^d L_i(\mathbf{w}) \quad (13)$$

目前,对于损失函数的最优化问题,随机梯度下降法^[29]是一种普遍使用的技术方法。随机梯度下降法通过一次次的迭代能够有效地逐渐逼近函数的局部最优点,而对于只存在一个局部最优点的凸函数,则意味着能够通过迭代较快地收敛到全局最优点。随机梯度下降法的每一步迭代过程都是一次对权重 w 的更新过程:

$$w_j = w_j - \alpha \frac{\partial L_i(w)}{\partial w_j} \quad (14)$$

其中, α 为学习率,决定了每一次迭代中的更新步长。将逻辑回归的损失函数代入 w 的一般更新公式,可以得到:

$$w_j = w_j - \alpha x_{ij} (f(x_i) - y_i) \quad (15)$$

小批量训练:相较于每次采样单个样本对 w 进行迭代,对数据进行批量化处理可以优化训练过程。定义 β 为批量大小,每次迭代都进行 β 次随机采样,抽取出 β 个样本组成小批量样本 X_β (X_β 是一个 $\beta \times n$ 的矩阵),其对应的批量标签为 y_β (y_β 是一个 $\beta \times 1$ 的向量),利用该样本组中所有样本的偏导均值对 w 进行一次更新。因此,在小批量训练中, w 的更新公式表示为

$$w = w - \frac{1}{\beta} \alpha X_\beta^\top \times (\text{Sigmoid}(X_\beta \times w) - y_\beta) \quad (16)$$

利用更新公式迭代权重 w ,直到模型准确率满足设定的阈值或者模型迭代次数达到设定的上限,则停止迭代,即训练结束,输出最新的模型权重 w^* 。

7.2 稀疏逻辑回归的安全计算

逻辑回归的模型更新公式包含了矩阵乘法运算,而现实生活中的数据普遍具有高维且稀疏的特征,因此,可以使用本文提出的协议来计算其中的矩阵乘法操作以提升训练速度。

我们在公开数据集 Newsgroups 上开展实验,使用逻辑回归解决文本分类问题。

首先对文本数据集进行分词、去除停用词等预处理操作,然后使用 TF-IDF 技术将文本数据表示成词向量,文本集的总词汇表共包含超过 170 000 个单词,而每段文本拥有的单词量大约为 50~100 个。因此,单文本的词向量表示中非零元素的占比约为 0.05% 甚至更低。

Newsgroups 数据集中共有 20 个类别的文本,本文选择其中的两个类别作为逻辑回归模型的训练数据,并针对不同批量大小进行了实验比较,结果如图 6 所示。

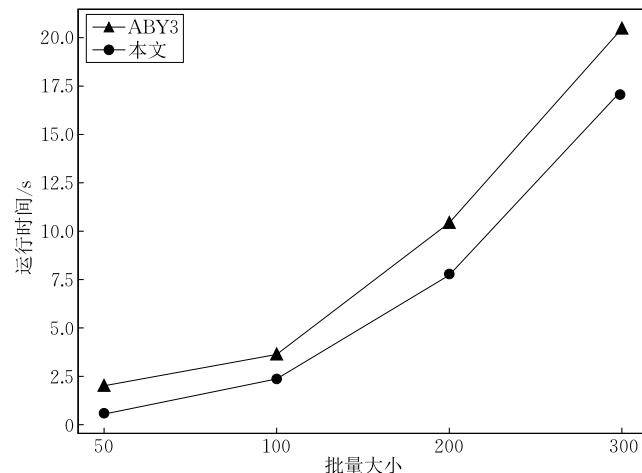


图 6 稀疏逻辑回归训练

我们统计了不同批量大小下完成一轮训练所需要的时间,从与 ABY3 的对比结果来看,本文协议应用到逻辑回归模型中依然保持着其计算优势,在时间上有 10%~30% 的提升,并且批量越小,计算性能提升的比例越高。

分析其原因:第 6.2 节中验证了矩阵稀疏度与计算开销的正相关关系,在批量较小时,矩阵 X_β 中非零列数的占比更小;随着批量的增加,矩阵 X_β 可能会加入新的非零列,增加计算开销。因此,本文稀疏乘法协议在小批量的模型训练场景下优势更加明显。

8 总结与展望

自从安全问题被社会重视以来,隐私保护机器学习一直都是热门话题。各种经典方案相继被提出,但大部分方案关注的都是模型本身,研究如何提高模型的准确性,如何降低协议的开销,而模型输入数据的特征往往被忽视。稀疏性是现实数据普遍具有的特征,将这一特点利用到模型的协议设计中去,特别是对于大型模型的计算,能够极大地节省计算开销。本文提出的方案就是在三方场景下,对稀疏数据计算的一次探索,分析说明了对于稀疏性特征的利用确实能够降低计算开销,提升效率。

但本文方案仍然存在不足,下一步将继续深入研究:(1)针对通信开销的问题,考虑在不增加计算开销的前提下,如何降低通信开销,进一步提升方案性能;(2)本文方案实现的是稀疏向量与稠密向量的乘法计算,对于更普适的稀疏向量与稀疏向量的乘法问题还有待解决;(3)本文协议要求稀疏向量被参与方中的一方所持有,即该参与方已知稀疏向

量的结构,但对于完全密态的稀疏向量的乘法问题依然没有解决,后续也将针对该问题进一步探讨研究;(4)本文仅将协议应用在了逻辑回归模型上,如何更高效地将稀疏乘法协议应用到更多机器学习模型上也是我们未来需要关注和解决的问题.

参 考 文 献

- [1] Maas A, Daly R E, Pham P T, et al. Learning word vectors for sentiment analysis//Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Oregon, USA, 2011: 142-150
- [2] Albirdre K, Albathe M, Li Y. Effective 20 newsgroups dataset cleaning//Proceedings of the 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). Singapore, 2015, 3: 98-101
- [3] Beimel A. Secret-sharing schemes: A survey//Proceedings of the Coding and Cryptology: Third International Workshop (IWCC). Qingdao, China, 2011: 11-46
- [4] Chaudhari H, Choudhury A, Patra A, et al. ASTRA: High throughput 3PC over rings with application to secure prediction //Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop. London, UK, 2019: 81-92
- [5] Chaum D, Crépeau C, Damgård I. Multiparty unconditionally secure protocols//Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. Chicago, USA, 1988: 11-19
- [6] Demmler D, Schneider T, Zohner M. ABY—A framework for efficient mixed-protocol secure two-party computation//Proceedings of the 22nd Annual Network and Distributed System Security Symposium(NDSS). California, USA, 2015: 1-15
- [7] Mohassel P, Zhang Y. SecureML: A system for scalable privacy-preserving machine learning//Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP). California, USA, 2017: 19-38
- [8] Mohassel P, Rindal P. ABY3: A mixed protocol framework for machine learning//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. Toronto, Canada, 2018: 35-52
- [9] Li Y, Xu W. PrivPy: General and scalable privacy-preserving data mining//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Anchorage, USA, 2019: 1299-1307
- [10] Nikolaenko V, Ioannidis S, Weinsberg U, et al. Privacy-preserving matrix factorization//Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. Berlin, Germany, 2013: 801-812
- [11] Nayak K, Wang X S, Ioannidis S, et al. GraphSC: Parallel secure computation made easy//Proceedings of the 2015 IEEE Symposium on Security and Privacy (SP). California, USA, 2015: 377-394
- [12] Laud P. Parallel oblivious array access for secure multiparty computation and privacy-preserving minimum spanning trees. Proceedings on Privacy Enhancing Technologies, 2015, 2015(2): 188-205
- [13] Goldreich O, Ostrovsky R. Software protection and simulation on oblivious RAMs. Journal of the ACM, 1996, 43(3): 431-473
- [14] Schoppmann P, Gascón A, Raykova M, et al. Make some ROOM for the zeros: Data sparsity in secure distributed machine learning//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. London, UK, 2019: 1335-1350
- [15] Beaver D. Efficient multiparty protocols using circuit randomization//Advances in Cryptology-CRYPTO'91: Proceedings 11. California, USA, 1992: 420-432
- [16] Chen C, Zhou J, Wang L, et al. When homomorphic encryption marries secret sharing: Secure large-scale sparse logistic regression and applications in risk control//Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. Singapore, 2021: 2652-2662
- [17] Cui J, Chen C, Lyu L, et al. Exploiting data sparsity in secure cross-platform social recommendation. Advances in Neural Information Processing Systems, 2021, 34: 10524-10534
- [18] Angel S, Chen H, Laine K, et al. PIR with compressed queries and amortized query processing//Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP). California, USA, 2018: 962-979
- [19] Xu G, Xu S, Ning J, et al. New secure sparse inner product with applications to machine learning. arXiv preprint arXiv: 2210.08421, 2022
- [20] Dong C, Chen L, Wen Z. When private set intersection meets big data: An efficient and scalable protocol//Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. Berlin, Germany, 2013: 789-800
- [21] Araki T, Furukawa J, Lindell Y, et al. High-throughput semi-honest secure three-party computation with an honest majority//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Vienna, Austria, 2016: 805-817
- [22] Acar A, Aksu H, Uluagac A S, et al. A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys , 2018, 51(4): 1-35
- [23] Paillier P. Public-key cryptosystems based on composite degree residuosity classes//Advances in Cryptology-EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques. Prague, Czech Republic, 1999: 223-238
- [24] Mohassel P, Rindal P, Rosulek M. Fast database joins and PSI for secret shared data//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. USA, 2020: 1271-1287