文章编号:1671-4598(2022)07-0201-06 DOI:10.16526/j.cnki.11-4762/tp.2022.07.030 中图分类号:TP309 文献标识码:A

一种最小化安全多方计算任务的方法

姚友罡[1], 肖 铮[1,2]
(1. 成都东软学院 保卫部, 成都 611844;
2. 四川工商职业技术学院 信息工程系, 成都 611830)

摘要: 安全多方计算 (Secure multi-party computation, MPC) 允许在不公开各参与方私有数据的情况下完成联合计算; 然而, 现有的计算任务往往涉及到多方海量数据集的分析与处理, 使得 MPC 的实际可用性显著降低; 提高 MPC 数据处理体量, 是目前研究的主要方向之一; 为提高 MPC 处理大规模数据的能力, 将 MPC 算法与数据并行分析框架相结合, 基于最小化多方计算任务的思想, 提出安全多方计算效率优化技术; 创建算法的有向无环图, 标注 MPC 节点及非 MPC 节点, 采用静态分析、查询重写转换和分区启发式等技术, 最小化 MPC 计算量, 提高计算的并发程度; 以多方线性回归为例, 讨论适应大数据分析的安全多方计算技术; 实验结果表明提出的安全多方计算优化技术在确保计算精度的条件下能够显著降低计算耗时; 算法提高了系统的效率, 增强了 MPC 的实用能力。

关键词: 安全多方计算; 私有数据; 数据集; 线性回归; 精度

A Method to Optimize the Efficiency of Secure Multiparty Computation

YAO Yougang[1], XIAO Zheng[1,2]
(1. Chengdu Neusoft Univercity, Security Dept., Chengdu 611844, China;
2. Sichuan Technology and Business College, Department of Information Engineering, Chengdu 611830, China)

Abstract: Secure multi-party computing (MPC) allows joint computing without disclosing the private data of each participant. However, the existing computing tasks often involve the analysis and processing of multi-party massive data sets, which significantly reduces the actual availability of the MPC. Improving the volume of the MPC data processing is one of the main research directions at present. In order to improve the ability of the MPC to deal with the large-scale data, the algorithm of the MPC is combined with the data parallel analysis framework. Based on the idea of minimizing multi-party computing tasks, a secure multi-party computing efficiency optimization technology is proposed. The directed acyclic graph of the algorithm is created, the MPC nodes and non MPC nodes are marked, and the techniques of static analysis, query rewriting transformation and partition heuristic are used to minimize the amount of the MPC calculation and improve the concurrency of calculation. Taking the multi-party linear regression as an example, this paper discusses that the secure multi-party computing technology is suitable for the big data analysis. The experimental results show that the proposed secure multi-party computing optimization technology can significantly reduce the computing time under the conditions of ensuring the computing accuracy. The algorithm improves the efficiency of the system and enhances the practical ability of the MPC.

Keywords: secure multi-party computation; private data; data sets; linear clustering; accuracy

0 引言

大数据分析是企业对外决策的关键环节之一, 决策的数据往往来自不同的数据源。例如, 从企业自身数据到城市数据、从基因数据到网络数据等, 跨数据实体 (数据源) 分析是数据分析的迫切需求。然而, 隐私数据或机密数据泄露已成为跨数据源分析的主要障碍, 工业界、学术界致力于寻找跨数据源分析的实用技术。安全多方计算 (MPC) 作为一种密码技术, 允许独立的参与方在不泄露自身私有输入的情况下联合执行所需的计算, 是跨数据源分析的常用技术之一。自 20 世纪 70 年代末提出 MPC 概念[1-3], 到如今的 30 多年历史里, 一直是密码学研究的活跃领域。近年来, MPC 的研究取得了重大进展[4-6], 然而, 由于以下四方面的原因, 使得 MPC 的研究仍局限于概念验证阶段[7]:

1) 因涉及密码学相关技术, MPC 应用、部署较难。

2) MPC 无法利用现有的成熟算法, 需要从头开始部署。

3) 现有的 MPC 引擎是孤立设计的, 不适应如 MapReduce 之类的云编程范式。

收稿日期:2021-12-09; **修回日期**:2022-02-23。
基金项目:教育部产学研项目基金资助(2018A03007);四川省 2020 年度教育科研重大课题(SCJG20A004-4);四川工商职业技术学院 2020 年科研创新团队阶段成果。
作者简介:姚友罡(1979-),男,黑龙江肇东人,硕士,助理研究员,主要从事高校安全管理,信息安全等方向的研究。
通讯作者:肖 铮(1983-),男,辽宁黑山人,硕士,副教授,高级工程师,主要从事人工智能、大数据等方向的研究。
引用格式:姚友罡,肖 铮.一种最小化安全多方计算任务的方法[J].计算机测量与控制,2022,30(7):201-206.

4）MPC 开发所需的编程/工程专业知识异于在敏感数据集上评估隐私泄露所需的专业知识。

与专注于 MPC 协议原语的实现不同，从 MPC 技术在云计算环境中的实际应用与企业项目开发角度考虑如何处理这些问题，即将 MPC 集成到云计算的编程范式中。更具体地说，同时扩展了 MPC 的 MapReduce 实现技术，允许多方利用自己的计算资源，支持多方计算（MPC）和 MapReduce 的混合编程。本文的创新点和主要贡献如下：

首先构建了 MPC 最小化的基本思想及实现步骤；其次，以线性聚类为例，提出了一种在确保安全性的同时最小化 MPC 实用计算的方法。该方法允许将计算划分为 MPC 计算部分和非 MPC 计算部分，其中非 MPC 计算部分在本地簇完成，无需多方通信，可并发执行，从而提高计算效率。最后，本文构建了一个实验原型系统，并以 UCI 数据集为例，验证了提出技术的可行性与正确性。

基于以上目标，全文的结构安排为。第 1 节讨论了大数据相关技术，特别是 MapReduce 编程基础，常用的 MPC 实现技术以及 MPC 与非 MPC 相关的划分技术。本文的技术创新性工作在第 2 节中展示，以线性聚类为例，分析 MPC 与非 MPC 划分方法及与 MapReduce 混合编程技术。第 3 节展示了以 UCI 机器学习数据集为基础的实验及实验结果数据。最后，第 5 节为结论以及对下一步工作的展望。

1 相关工作

文中主要讨论提高 MPC 计算效率的技术与方法，即如何有效地将 MPC 技术融入大数据处理流程，不涉及 MPC 协议原语的实现或大数据处理等技术问题。与本文相关的工作包括 Pyspark 数据分析技术、MPC 技术以及提高 MPC 运行效率的现有方法与实践等。

1.1 Pyspark 数据分析技术

Pyspark 源自 Apache 的 Spark 项目，是为实现用 Python 语言编写 Spark 应用程序而对 Spark 的封装。作为 Hadoop 中 MapReduce 的改进与替代方案，Spark 引入了弹性分布式数据集（RDD，Resilient Distributed Datasets）这一基于内存的数据结构，以适应交互式、实时、分布式与容错的低延时应用。Spark 使应用开发人员专心于业务逻辑的开发而无需担心基础架构、环境等问题。文中利用 Spark 的并行处理能力，提高非 MPC 类计算的效率（完成的主要计算任务是矩阵乘法，该部分的内容在 2.3 节）。

1.2 MPC 技术

安全多方计算作为一种通用的加密原语，在保护个人或组织数据隐私的同时实现多方私有数据的分析与挖掘。常借助电路（布尔电路或算术电路）的概念以建立 MPC 的形式化定义。通用的安全多方计算的定义如定义 1 所示。

定义 1：安全多方计算

在 n 个互不信任的参与方之间的安全多方计算（函数）F，常用是一个三元组来表示：

$$F=(C,T_i,T_o\} \tag{1}$$

其中：C 表示电路，是用户需求、功能、算法的抽象；T_i 表示输入方及其输入数据的集合；T_o 表示输出方及其输出数据的集合。在 F 的计算过程中，要求任意参与方 p_i 除获得自身的输出信息外，均不能获得其他参与方的任何输入或输出信息。MPC 的这一定义是较为严格的，实际应用中往往放宽某些限制，以提高效率。

已有一些基本子协议（技术）用于构建 MPC 协议：这些子协议（技术）主要包括：1）不经意传输（OT，oblivious transfer）协议；2）加密电路（GC，garbled circuit）技术；3）同态加密（HE，homomorphic encryption）技术以及秘密共（SS，secret sharing）协议。从这些基础协议构建 MPC 实用程序不是易事，需要了解密码学的相关知识。目前业界已提出 MPC 的实可编程式现框架，这些框架主要包括 Obliv-C、ObliVM、SPDZ、Sharemind 等。在实际应用中，借助于技术实现框架，即便是不具备密码学专业知识的应用程序开发人员也能编写实现数据安全分析的任务。另一方面，虽然这些框架都是基于 MPC 技术独立实现的，但各自拥有不同的特性及优缺点。因 ObliVM 及 SPDZ 不再维护或更新，本文的实验部分，采用了 Obliv-c 及 Sharemind 的作为 MPC 的评估版。为此，表 1 列出了这两种框架的特点差异。

表 1 常用 MPC 实现框架特点对比

	Obliv-c	Sharemind
协议	姚氏加密电路	支持多种复合协议
编程语言	与 C 兼容，面向特定的应用领域的语言	“应用服务平台”
优点	快速，易于实用	提供多种工具集，支持向量优化
缺点	两方计算协议	非开源软件

1.3 现有方法

在本小节中，先后简要描述了加速 MPC 计算的方法，以提高多方计算高效可行。目前，有两种互补的提高 MPC 计算效率的方法。其一是直接提高 MPC 计算协议并发度；其二采用混合 MPC 计算。前者从 MPC 协议角度出发，通过并发实现茫然计算。这种对 MPC 效率的改进有限，特别是不适应大数据处理需求。后者将安全计算按照一定的规则划分为不安全的非 MPC 类计算及少量的安全 MPC 类计算，通过提高非 MPC 类计算的效率间接提高计算的整体效率。两种方式都基于到 MapReduce 编程范式的基本思想，文中着眼于第二种实现方式。

文献［9－10］先后提出并证明“并行有助于不经意”以及流式 MapReduce 的概念以及任何在流式 MapReduce 抽象中编写的程序都可以被转换为高效的不经意的算法，并利用这一想法涉及了网络 ORAM 方案。Liu 等人[11]等人则基于 MapReduce 编程抽象构建 MPC 实现框架，即 ObliVM

同年，Kartik 等[4]设计并实现的 GraphSC 框架都属于提高 MPC 计算协议并发度的范畴。

Rastogi[12]等首先提出了混合 MPC 计算方法，其中非 MPC 的计算，各参与方在本地独立完成作业，类似普通的应用程序。MPC 计算部分由各参与方协同，共同完成。然而，在 Rastogi 等人的设计中，需要程序设计人员手动标注非 MPC 类及 MPC 类计算。虽然这些标注是精细化的，但程序设计人员需要具备专业的 MPC 知识，使其应用受到限制。在此基础上，文献[12]进一步将 MapReduce 编程范式融入到本地非 MPC 计算中。在他们的设计中，参与计算的节点可以分为两种类型：控制节点和工作节点。其中，控制器节点作为任务分配器、作业监督器以及任务同步器并作为节点间共享公共数据的存储介质；工作节点跨参与方完成 MPC 任务，如访问参与方自身的私有数据，充当本地 MapReduce 任务的执行器。文献[14－15]借助有向无环图（DAG）研究与查询相关的 MPC 计算及其可扩展性问题。与上述工作不同，文中以线性回归问题及其扩展为主要研究目标。

2　实现方法

本节首先介绍了最小化安全多方计算的基本思想及步骤，然后以线性回归模型为例具体讨论相关技术及方法。

2.1　基本思想及步骤

本文提出的最小化安全多方计算（下文常称为计算）的基本思想是使 MPC 类的计算尽可能少，只实现必要的 MPC，即能不用 MPC 实现的尽可能不采用。如何能最小化 MPC 类的计算，本文提出将计算分解、转换为有向图，进而简化 MPC，重构 MPC 算法。步骤总结如下：

1）确定数据的属主。根据算法所属数据属主的差异，确定数据的“主人”。

2）确定信任标注（可选）。根据数据属主，确定参与者之间的信任关系。这种信任关系可能繁殖，或者终止，这些工作采用自动以及交互的方式来实现。

3）算法重写。即获得与当前算法等效的算法，但其需要的 MPC 类计算更少。区分本地计算以及多方计算等；采用的技术包括信任繁衍、组合、分割。

4）MPC 优化。MPC 优化技术包括：MPC 前端（MPC 操作与本地明文操作之间的边界）下移、MPC 前端上移以及混合操作（如混合联接、公共联接和混合聚合）等。

5）算法优化。除了对单个计算节点的优化，算法的整体性能也应优化。常用的方法是减少对 MPC 底层算法的依赖。

本节的余下部分以线性回归预测模型为例，讨论最小化 MPC 计算的实现方法。

2.2　线性回归模型

线性回归预测模型是一种有监督的学习算法，是许多机器学习算法的基本组成部分。它通过拟合训练数据集的线性曲线来获得目标模型并实现预测，该模型的通用公式如式（2）所示。

$$Y = X\theta + \varepsilon \tag{2}$$

式中，$Y \in \boldsymbol{R}^n$，是由 n 个元素的列向量组成的目标变量，也称为标签或因变量；$X \in \boldsymbol{R}^{n\times d}$，是 n 行 d 维的输入向量，即因变量；$\theta \in \boldsymbol{R}^n$，称为拟合参数或模型参数，$\varepsilon$ 是为了平衡模型两边的值而添加的部分，称为误差项。

求解方程（1）中的模型参数的过程称为模型的学习，常通过最小化均方误差来求解，如式（3）所示。

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{n}\|Y - X\theta\|^2 \tag{3}$$

式（3）称为线性回归的目标函数。更一般地，线性回归模型的目标函数可写为如式（4）所示。

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{n}\|Y - X\theta\|^2 + \lambda\|\theta\|^2 \tag{4}$$

在式（4）中 $\lambda\|\theta\|^2$ 为目标函数的惩罚项，λ 称为惩罚系数。特别地，当 $\|\theta\|$ 为 $l2$ 范数时，式（4）所对应的模型为岭回归模型；若为 $l1$ 范数，则表示 LASSO 回归模型。

令 $J(\theta) = \frac{1}{n}\|Y - X\theta\|^2 + \lambda\|\theta\|^2$，下面给出目标函数（4）的求解过程。

首先展开平方项得等待如下的式子：

$$J(\theta) = \frac{1}{n}(Y - X\theta)^T(Y - X\theta) + \lambda\theta^T\theta =$$

$$\frac{1}{n}(Y^T - \theta^T X^T)(Y - X\theta) + \lambda\theta^T\theta =$$

$$\frac{1}{n}(Y^TY - Y^TX\theta - \theta^TX^TY + \theta^TX^TX\theta) + \lambda\theta^T\theta \tag{5}$$

式（5）对 θ 的一阶偏导数为：

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{n}(0 - X^TY - X^TY + 2X^TX\theta) + 2\lambda\theta =$$

$$2(\frac{1}{n}X^TX\theta + \lambda\theta) - \frac{2}{n}(X^TY) \tag{6}$$

在式（6）中，令导数等于 0，可知回归系数满足式（7）：

$$\left(\frac{1}{n}X^TX + \lambda I\right)\theta = \frac{1}{n}X^TY \tag{7}$$

如第 1 节所述，提高大数据分析的安全计算效率的基本思路是，尽量降低 MPC 类计算的需求，为此文中采用如下两阶段的优化措施。

2.3　数据计算优化

该阶段从计算的内在需求出发，找出 MPC 类计算与非 MPC 类计算部分。为计算式（7）需考虑两种不同的数据分布情况：1）数据在参与方间水平分割，即一个参与者有一条完美的；2）数据在参与方间垂直分割。

2.3.1　数据水平分割

该阶段从计算的内在需求出发，找出 MPC 类计算与非 MPC 类计算部分。为计算式（7）需考虑两种不同的数据

分布情况：1）数据在参与方之间水平分割；2）数据在参与方之间垂直分割。

在水平分割下，数据 X 以记录为单位分布在各参与方，即 $X=[x_1,x_2,\ldots,x_i]^T$，且 $x_i\in\mathbf{R}^{\alpha\times d}(\alpha\subseteq n)$ 是第 i 方的输入数据。数据或信息 x_i、x_i^T 以及 $y_i(y_i\in\mathbf{R})$ 都属于参与方 $i(i\leqslant n)$，因而针对式(7)中的因子 $X^TX=\sum_i x_i^Tx_i$，$X^TY=\sum_i x_i^Ty_i$ 的计算不涉及多方参与（但对方程（7）的求解需要各方秘密共享数据）。

2.3.2 数据垂直分割

在垂直分割下，数据 X 按属性划分，并由各参与方持有。即 $X=[x_1,x_2,\ldots,x_i]$，$x_i\in\mathbf{R}^{n\times\beta}(\beta\subseteq d)$ 由参与方 i 持有。特别地，考虑两方参与的情况，即数据 X 被划分成 X_1，X_2 两个子集，并分别由参与方 p_1 和 p_2 方持有。变量 Y 由其中一方持有，这里假设 Y 由 p_2 持有。则式（7）中的因子 X^TX 可以用式（8）表示：

$$X^TX=\begin{bmatrix}X_1^TX_1 & X_1^TX_2\\ X_2^TX_1 & X_2^TX_2\end{bmatrix} \tag{8}$$

X^TY 的结果如式（9）所示：

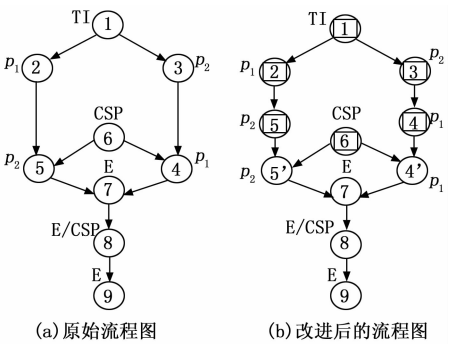
$$\begin{bmatrix}X_1^TY\\ X_2^TY\end{bmatrix} \tag{9}$$

式（7）等式右端矩阵的主对角线上元素为同一方所拥有，其计算属于非 MPC 类，次对角线上的元素为两方共有，输入 MPC 类计算。同理，式（9）等式右端矩阵中的 X_1^TY 是 MPC 类计算，X_2^TY 是非 MPC 类运算。

2.4 计算流程优化

该阶段从计算需求出发，使用 DAG（Directed Acyclic Graph）图来表示计算流程，并利用 DAG 图的内在联系，进一步降低 MPC 类的计算需求。在利用共轭梯度下降法求解式（7）的解时，参考文献[8]可得如算法 1 所示求解思路。

使用 DAG 图表示程序流程及模板的通用性，如图 1（a）所示。图中圆形内的阿拉伯数字对应算法 1 中的序号，圆形旁的字符是参与方的缩写。



(a)原始流程图 (b)改进后的流程图

图 1 安全计算流程及其改进措施

算法 1：优化的安全多方（以两方为例）计算求解算法

参与方：数据提供者 p_1、p_2，可信初始值设定者 TI/ 加密服务提供商 CSP、电路评估者 evaluator

输入：p_1、p_2 的原始数据 $\vec{x}$、$\vec{y}$

输出：数据提供者共同学习到的模型参数 θ

1）生成随机向量 $\vec{a}$、$\vec{b}$、以及随机值 r，计算 $z=<\vec{a},\vec{b}>-r$，并将（$\vec{a}$，r）发送给者 p_1，将（$\vec{b}$，r）发送给 p_2。

2）p_1 将 $\vec{x}+\vec{a}$ 发送给 p_2。

3）p_2 将 $\vec{y}+\vec{b}$ 发送给 p_1。

4）p_1 计算其加性共享值 $s_1=\vec{x}$，$\vec{y}-\vec{b}>-r$。

5）p_2 计算其加性共享值 $s_2=\vec{x}+\vec{a}$，$\vec{b}>-z$。

6）CSP 生成重构及求解式（6）的混淆电路 C，并将其发送给评估者 evaluator。

7）数据提供者与加密服务提供商 CSP 之间利用茫然传输协议获取其在混淆电路 C 中的共享输入，并将该共享值转发给电路评估者 evaluator。

8）评估者 evaluator 评估电路 C，并与数据提供者共享加密的模型参数 $\hat{\theta}$。

9）CSP 向数据提供者发送解密映射以便从 $\hat{\theta}$ 中解密出模型参数 θ。

首先可信初始值设定者 TI，生成随机向量 $\vec{a}$、$\vec{b}$ 以及随机值 r，计算 $z=<\vec{a},\vec{b}>-r$，并将（$\vec{a}$，r）发送给者 p_1，将（$\vec{b}$，z）发送给 p_2。其中的一个参与方（如本文中的）p_1 将自身的数据 $\vec{x}$ 加 $\vec{a}$，之后将信息发送给另一参与方 p_2。然后，p_2 将其数据减去 $\vec{b}$。p_1 与 p_2 计算各自的加性共享值（s_1，s_2）。服务提供商 CSP 生成重构及求解式（6）的混淆电路。数据提供者与加密服务提供商 CSP 之间利用茫然传输协议获取其在混淆电路 C 中的共享输入各参与方忠实地执行协议，在完成与其它方通信的基础上管理本地非 MPC 类计算以及 MPC 类计算。为提高文献［8］数据计算处理速度，本文采取的主要措施如下：

对于处理的数据，为提高安全计算的效率，增加三方面的额外信息：1）增加计算数据所有者域及数据所有者传递规则（某计算节点的数据所有者域由该节点的数据提供者及其输入弧传递进来的数据所有者共同确定，数据所有者沿图结构向下传递，受信任标识集的影响而可能终止）。2）增加计算结果数据的接收者域（由弧的终点确定）。3）增加数据安全处理需求，用于表示该数据是否需要在 MPC 下完成计算。例如，图 1（a）中，计算节点①的没有输入弧（入度为 0），即该节点没有输入数据，其数据所有者域仅包含该节点自身，即可信初始值设定者 TI。该节点有两条输出弧（出度为 2），表示该节点有两条输出信息，其接收者分别是两条弧指向的节点，即数据提供方 p_1 和 p_2。同时该节点数据不包含隐私信息，仅需通过安全通道传递给接收方即可，可见针对该节点的计算无需在 MPC 下执行。

增加数据信任标识集，即由数据提供者授权的可在明

文状态下（非 MPC 计算类）使用其数据的使用方构成的集合。例如公共数据或多方共享数据的所有者构成了这些数据的信任标识集，同时信任标识集也可以是权威数据管理机构，他们拥有参与方的原始数据。信任标识集里的各参与方在明文状态下使用数据，降低了计算成本。

节点分裂与繁殖，即将复杂的 MPC 类计算节点细分成 MPC 类计算节点与非 MPC 类计算节点。目前，存在 3 种方式实现节点分裂与繁殖。其一是利用上述两项措施，即当节点计算执行参与方是节点计算数据的所有者或者执行方属于节点数据的信任标识集，该类计算可以划分为非 MPC 类计算。其二是通过添加混合协议，即将部分高负载的 MPC 类计算在可信方完成，从而分离出非 MPC 计算类与轻负载的 MPC 计算类，其三是利用数据已有计算结果及数据混淆技术减少计算茫然性的需求。节点分裂与繁殖进一步减轻了 MPC 计算负载。

图 1（a）的 DAG 图，在经过上述处理后步骤后的 DAG 图如图 1（b）所示。其中带方框的节点属于非 MPC 计算类。从中可以看出 MPC 安全计算结点的数量有所降低。

3　实验与评估

3.1　实验设置

本节建立了以 Python 编程语言为基础的实验开发环境。为突出实验效果，实验共设置了 4 个参与节点，各节点都配置有 4 核 CPU（2.4G）以及 8G 的内存。其中 2 个作为数据提供者；一个作为可信初始值设定者 TI 及加密服务提供商；另一个作为数据评估者。各参与节点都配备有本地 Pyspark 组件、MPC 组件、私有数据访问与存储接口以及用于与控制器节点通信的网络接口。其中，评估节点以及两个数据提供者节点组成了含 3 节点的 Spark 独立集群。评估节点作为 Spark 的主节点，数据提供者节点是 Spark 的工作者节点。MPC 组件的实现采用的是 Obliv-c，通过 Python 的模板替换工具 Pystache，以及算法流程的 DAG 图自动生成 MPC 与非 MPC 代码。评估用的数据集来自 UCI 机器学习数据集，数据集及其主要特性如表 2 所示。各数据集按 7∶3 的比例随机选取作为训练数据集与测试数据集，并将数据按 1∶1 的比例随机分配给数据提供者 p_1 和 p_2。

表 2　实验数据集及其主要特性

编号	数据集名称	记录数(n)	维度(d)
1	Student Performance [16]	649	30
2	Wine Quality [17]	4 898	11
3	Adult [18]	48 842	13
4	CT slices [19]	53 500	384
5	Gas sensor array [20]	417 850 4	16

基于以上的实验设置，主要完成了以下两方面的评估指标。

运行时间：针对不同数据集，在不同 MPC 优化层次下（包括在 Pyspark 下的全明文技术）运行时间不同的优化参数对 MPC 运行时间的影响。

预测精度：即在各种设置下所获得的预测值与真实值之间的根均方误差（RMSE）。

3.2　实验评估

本节从运行时间及精度两方面讨论算法的性能。

3.2.1　运行时间

针对不同的数据集，在相同的设置（如 3.1 节所述）不同的安全配置（全明文 Pyspark、现有安全多方计算 Obliv-c 以及文中的优化设置）下，各重复实验 10 次，记录其平均值，实验结果如表 3 所示。

表 3　不同设置下各数据集所需运行时间

编号	Pyspark/s	Obliv-c	本文/s
1	32	45 s	41
2	36	450 s	43
3	40	620 s	50
4	46	25 m	62
5	50	——	72

实验结果表明，所有不同的安全配置下运行时间都随记录数的增加而增加，同时运行时间也受数据维度的影响。总体而言，全明文分析所需的时间最少，现有安全多方计算（Obliv-c）所需时间最多，本文提出的方法与全明文分析所需时间十分接近。另外，可以注意到当记录数达到万级以上时，Obliv-c 的运行耗时迅速增加，如本例中的 CT slices 数据集，其运行时间在 25 分钟。当记录数继续增加时，该方案甚至不可接受，如本例的 Gas sensor 数据集。可见文中提出的方法显著降提高了运行效率，提高了应用在时间的可行性。接下来分析该方案在精度上能否满足要求。

3.2.2　精度

与 3.2.1 节运行时间设置相同，同时记录了现有安全多方计算 Obliv-c 以及文中的优化设置两种情况下的根均方误差，如表 4 所示。

表 4　不同设置下各数据集的根均方误差

编号	Pyspark	Obliv-c	本文
1	4.72±0	4.60±0.01	4.62±0.006
2	1.0±0	0.82±0.04	0.80±0.03
3	132±0.003	131.5±0	132.2±0.01
4	9.11±0.004	9.05±0.05	9.04±0.02
5	87±0.052	——	90.05±0.03

可见，各种方法在精度上的差异并不显著，3 种安全配置下的根均方误差大致相同，即文中设计的方法（在现有技术条件下）满足精度需要。

4 结束语

本文以数据分析中的常用方法——线性回归为例讨论在安全多方计算中如何提高数据分析的效率。提出的优化措施包括：计算优化以及执行流程优化。在实验部分，针对 UCI 真实数据集完成的实验验证。实验结果表明，文中提出的方案在精度、时间上都是可行的。下一步准备将混合 MPC 以及 MPC 并行化相结合，进一步验证文中提出的方法的可行性。另一方面，如何将混合 MPC 技术推广，以适应更多的真实应用场景也是未来工作方向之一。同时如何减少对可信初始值设定者、加密服务提供商、电路评估者的信任和依赖，从而提高系统可信性，也是未来应考虑的主要工作之一。以去信任为中心的区块链技术提供了相应的解决思路。

参考文献：

[1] ADI Shamir. How to share a secret [J]. Communications of the ACM, 1979 (11): 612 - 613.

[2] YAO A C. Protocols for secure computations [C] // Proc. of the 23rd Annual IEEE Symposium on Foundations of Computer Science, 1982.

[3] ROGER Colbeck. Quantum And Relativistic Protocols For Secure Multi - Party Computation [D]. Phd Thesis, 2011.

[4] NAYAK Kartik, WANG X S, IOANNIDIS Stratis, et al. GraphSC: Parallel Secure Computation Made Easy [C] // 2015 IEEE Symposium on Security and Privacy (SP). IEEE, 2015.

[5] EVANS David, VLADIMIR Kolesnikov, ROSULEK Mike. A Pragmatic Introduction to Secure Multi-Party Computation [J]. Foundations & Trends? in Privacy & Security, 2018 (1): 70 - 246.

[6] ARCHER David W, DAN Bogdanov, YEHUDA Lindell, et al. From Keys to Databases—Real-World Applications of Secure Multi-Party Computation [J]. The Computer Journal, 2018 (12): 1749 - 1771.

[7] VOLGUSHEV Nikolaj; LAPETS Andrei; BESTAVROS; AZER Scather: Programming with Multi-party Computation and MapReduce [R]. Technical Report BU-CS-TR 2015-010, Computer Science Department, Boston University, August 29, 2015.

[8] ADRIA Gascón, SCHOPPMANN P, BALLE B, et al. Privacy-Preserving Distributed Linear Regression on High-Dimensional Data [J]. Proceedings on Privacy Enhancing Technologies, 2017, 2017 (4): 345 - 364.

[9] 孙玉强，李媛媛，陆 勇. 基于 MapReduce 的 K-means 聚类算法的优化 [J]. 计算机测量与控制，2016，24 (7)：272 - 275，279.

[10] 李亚如，刘建华. 大数据环境下 MapReduce 准入控制的设计与实现 [J]. 计算机测量与控制，2016，24 (2)：114 - 117.

[11] DANA Dachman-Soled, Chang LIU, Charalampos PAPAMANTHOU et al. Oblivious Network RAM and Leveraging Parallelism to Achieve Obliviousness [J]. Journal of Cryptology, 2019 (3): 941 - 972.

[12] GOODTICH M T, MITZENMACHER M. Privacy-Preserving Access of Outsourced Data via Oblivious RAM Simulation [C] // Proceedings of the 38th international conference on Automata, languages and programming-Volume Part II. Springer-Verlag, 2010.

[13] LIU C, WANG X S, NAYAK K, et al. ObliVM: A Programming Framework for Secure Computation [C] //Security & Privacy. IEEE, 2015.

[14] RASTOGI Aseem, Matthew A. HAMMER, HICKS Michael. Wysteria: A Programming Language for Generic, Mixed-Mode Multiparty Computations [C] // 2014 IEEE Symposium on Security and Privacy. IEEE, 2014.

[15] VOLGUSHEV N, LAPETS A, BESTAVROS A. Programming Support for an Integrated Multi-Party Computation and MapReduce Infrastructure [C] // 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb). IEEE, 2015.

[16] BATER J, ELLIOTT G, EGGEN C, et al. SMCQL: Secure Querying for Federated Databases [J]. Proceeding of the vidb erdowment, 2016 (10): 673 - 684.

[17] VOLGUSHEV N, SCHWARZKOPF M, GETCHELL B, et al. Conclave: secure multi-party computation on big data (extended TR) [R]. 2019 (3): 38 - 56.

[18] 李顺东，杜润萌，杨颜璟，等. 安全多方多数据排序 [J]. 计算机学报，2020，43 (8)：1448 - 1462.

[19] 方炜炜， 任江，夏红科. 异构分布的多元线性回归隐私保护模型 [J]. 计算机研究与发展，2011，48 (9)：1685 - 1692.

[20] 姚亦飞. 保护私有信息的统计计算问题研究 [D]. 合肥：中国科学技术大学，2008.

(上接第 186 页)

[16] 毕庆生. 基于 Onboard SDK 的无人机自主飞行控制技术研究 [D]. 西安：西安石油大学，2017.

[17] 饶颖露，邢金昊，张 恒，等. 基于视觉的无人机板载自主实时精确着陆系统 [J]. 计算机工程，2021，47 (10)：290 - 297.

[18] REDMON J, FARHADI A. YOLOv3: an incremental improvement [C] //2018 IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[19] GIRSHICK R. Fast R-CNN [C] //Proceedings of the IEEE international conference on computer vision. 2015: 1440 - 1448.

[20] SANDLER M, HOWARD A, ZHU M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks [C] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4510 - 4520.

[21] 姚明海，杨 圳. 基于轻量级卷积神经网络的实时缺陷检测方法研究 [J]. 计算机测量与控制，2019，27 (6)：22 - 25.