秘密計算によるプライバシー保護階層型クラスタリング

三品 気吹[1,a)]　五十嵐 大[1]　濱田 浩気[1]　菊池 亮[1]

概要：秘密計算ではデータを暗号化したまま計算するため，プライバシーを保護したまま安全にデータ分析を行う方法として注目されており，特に機械学習 (AI) を秘密計算上で実現する「秘密計算 AI」の研究は近年活発に行われている．今までの秘密計算 AI の研究の多くは，ロジスティック回帰やニューラルネットワークといった「教師あり学習」に分類されるものであったが，これまであまり取り組まれてこなかった「教師無し学習」も平文のデータ分析ではよく用いられる重要な手法である．そこで本稿では，教師無し学習に分類される手法の一つである「階層型クラスタリング」の秘密計算上での実現に取り組む．階層型クラスタリングは平文でも計算量が多く，また途中計算を全て秘匿したまま行うのは難しいが，本稿では入力データ数以外を全て秘匿したまま，安全かつ効率良く計算できる秘密計算階層型クラスタリングアルゴリズムを提案する．50 件の分類を 11 秒で達成，また結果は平文と一致し，実用的な性能を示した．

キーワード：秘密計算，機械学習，教師無し学習，階層型クラスタリング

Privacy Preserving Agglomerative Clustering in Secure Computation

Ibuki Mishina[1,a)]　Dai Ikarashi[1]　Koki Hamada[1]　Ryo Kikuchi[1]

Abstract: Secure computation is a technology that calculates data while it is encrypted. It is expected as a method for safely analyzing data while protecting privacy, and in particular, research on "secure-computation AI" that realizes machine learning (AI) on secure computation has been actively conducted in recent years. Most of the research on secret computation AI so far has been categorized as "supervised learning," such as logistic regression and neural networks, but "unsupervised learning," which has not been tackled so far, is also a commonly used method. In this paper, we tackle the implementation of "hierarchical clustering," one of the unsupervised learning methods, on secure computation. Hierarchical clustering is computationally expensive even in plain text, and it is difficult to perform all the computations in secret. In this paper, we propose a secret hierarchical clustering algorithm that can securely and efficiently compute the hierarchical clustering while keeping all the data secret except the number of input data. 50 cases can be classified in 11 seconds, and the results are consistent with the plaintext, showing practical performance.

Keywords: Secure Computation, Machine Learning, Unsupervised Learning, Hierarchical Clustering

1. はじめに

秘密計算とはデータを暗号化したまま計算する技術である．秘密計算を用いることで，企業の重要な情報や顧客情報を安全に守ったままデータ分析等に利用できるため，様々な統計手法や機械学習手法を秘密計算で行うといった研究がなされている．特に秘密計算上で機械学習を行う手法については多数の提案がされている．

しかし，これまでに提案されてきた秘密計算上の機械学習手法の多くは，ロジスティック回帰やディープラーニングと言った「教師あり学習」に分類されるもので，「教師無し学習」を扱った研究は少ない．教師あり学習と教師無し学習の違いは，学習時にラベルを必要とするかどうかで，それぞれ異なる目的で用いられるため，実際のデータ分析ではどちらもできることが望ましい．

筆者らはこれまでにロジスティック回帰 [11]，線形回

[1] NTT 社会情報研究所
NTT Social Informatics Laboratories
[a)] ibuki.mishina.br@hco.ntt.co.jp

帰 [13]，ディープラーニング [8]，医療統計 [12][10] など多数の機械学習や高度な統計手法を秘密計算上で実現してきたが，それらに教師無し学習に該当するものはない．そこで本稿では教師無し学習の一手法である階層型クラスタリング [5] を秘密計算上で計算するアルゴリズムを提案し，実装・評価する．本稿で提案する秘密計算階層型クラスタリングでは学習データは勿論のこと，学習途中の情報も秘密にしまま安全に処理することを目指す．

1.1 関連研究

プライバシー保護階層型クラスタリングを扱った論文としては，MPOT19[6] などがある．MPOT19 は，分析に参加する 2 者がそれぞれ独立したデータセットを持ち，お互いにデータを明かさずに集約モデルを作る連合学習の形を取っており，筆者らが目指す最初から最後まで暗号化したまま処理するものとは異なるモデルとなっている．またクラスタ間の距離計算方法として採用されているのは単リンク法 (最短距離法) と呼ばれるもので，計算コストは低いものの，分類精度が良くないことから近年の実用ではあまり用いられない手法である．他のプライバシー保護階層型クラスタリングを扱った論文も同様に，計算の参加者が各々分割されたデータを持っているモデルとなっていた [4], [7].

1.1.1 課題

先行研究を踏まえて，プライバシー保護階層型クラスタリングには以下の課題があると考えられる．

- 学習データ，計算途中の値も全て暗号化したまま行う手法は提案されていない
- 秘密計算上での実用的なクラスタ間の距離計算方法が提案されていない

1.2 本稿の貢献

本稿の貢献は以下である．

- 学習データ，計算途中の値を全て暗号化したまま安全に階層型クラスタリングを行う手法を実現
- 実用的なクラスタ間の距離計算方法である群平均法を秘密計算上で実現

50 件のデータに対する秘密計算階層型クラスタリングを 11 秒で処理し，またクラスタリング結果は平文と完全に一致した．

2. 準備

2.1 クラスタリング

クラスタリングは教師無し学習に分類される機械学習手法の一つである．回帰分析やクラス分類等の教師あり学習では欲しい出力 (教師データ) を用意して，その出力を高い精度で再現するようなモデルの構築を目的とするのに対し，クラスタリング等の教師無し学習では事前に欲しい出力を定めない．例えばクラス分類 (教師あり学習) とクラスタリング (教師無し学習) は一見似ているが，クラス分類の場合は「このデータはクラス A に分類されるようにしたい」といったように予め分類したいクラスが決まっているのに対して，クラスタリングの場合は「どのデータとどのデータが同じクラスタに分類されるか (似ているか) を知りたい」といったように，事前にどのデータがどのクラスタに分類されるかは決まっていないという違いがある．

2.1.1 階層型クラスタリング

クラスタリングは与えられた複数のデータ間の距離を計算し，距離が近いデータ，すなわち似ているデータ同士を同じクラスタとして仲間分けする．例えばネット通販のユーザに対してクラスタリングを適用することで似ているユーザを見つけ出し，それによってレコメンドする商品を決定する等の利用方法がある．

クラスタリングの手法は大きく 2 つに分類され，k-means 法 [3] のように何個のクラスタを作るかを事前に決めておく非階層的なものと，クラスタ数は事前に定義せず最も似ているデータ同士から順番にクラスタにしていく階層的な手法がある．本稿で取り組むのは後者の「階層型クラスタリング」[5] である．

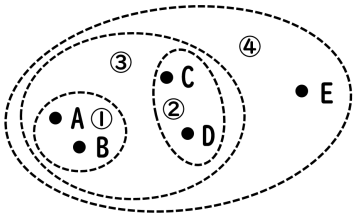

図 1: 階層型クラスタリングのイメージ図

階層型クラスタリングは図 1 のように，データを近いものから順番にクラスタとしてまとめていく分析手法である．階層型クラスタリングの大まかな計算手順を以下に示す．

(1) 全データ点の中で最も近い 2 つを 1 つのクラスタとしてくくる
(2) 新しく作ったクラスタと，他のデータ点との距離を計算する
(3) 最も近いデータ点 or クラスタ同士を新しいクラスタとする
(4) 全てのデータ点が 1 つのクラスタになるまで手順 (2) と (3) を繰り返す

このような計算手順で全てのデータ間の距離を計算し，最終的には図 2 のようなデンドログラム (樹形図) を得ることが階層型クラスタリングの目的である．図 2 のデンドログラムは図 1 のクラスタリング結果に対応しており，データ A と B が最も近く，その次に C と D が近く，その次に A

とBからなるクラスタとCとDからなるクラスタが近く，Eが最も遠いことを表す．このように各データの近さを細かく見たい場合に階層型クラスタリングは用いられる．

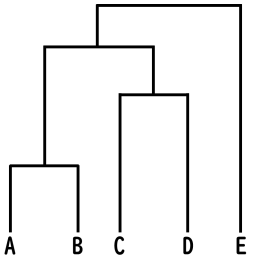

図 2: デンドログラムの例

一般的な平文の階層型クラスタリングアルゴリズムの計算量は，データ数 n に対して $n^2 \log n$ となる．また階層型クラスタリングを行う場合に重要となるのは，「データ間の距離の求め方」と「クラスタ間の距離の求め方」である．これらには複数の手法があり，本稿で用いた手法を次に説明する．

ユークリッド距離

ユークリッド距離はデータ間の距離の計算方法として最も有名な方法の1つであり，2つのデータ $\vec{x} = (x_1, x_2, \cdots, x_n)$ と $\vec{y} = (y_1, y_2, \cdots, y_n)$ の距離 $d(\vec{x}, \vec{y})$ を式 (1) で表す．

$$d(\vec{x}, \vec{y}) = \sqrt{(x_1 - y_2)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} \tag{1}$$

群平均法

群平均法はクラスタ間の距離を計算する際によく用いられる距離関数の1つである．平文の階層型クラスタリングではウォード法を用いることが多いが，分類感度が良いぶん計算量も多い．一方で最短距離法や最長距離法といったシンプルな方法は計算量が少ないぶん分類感度が低い．群平均法はウォード法より計算量が少なく，最短距離法や最長距離法と比べて分類感度が良く，計算量と精度のバランスが良好な手法である．

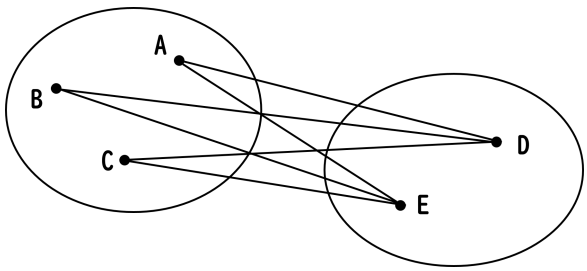

図 3: 群平均法のイメージ図

図3のような，データ点ABCを含むクラスタと，データ点DEを含む2つのクラスタ間の距離を群平均法で求める場合，データ点ABCとデータ点DEの全てのデータ間の距離の平均をクラスタ間の距離とする．各データ点間の距離は最初に全て求めているため，クラスタ間の距離を計算する際は平均値を求めるだけで済む．ウォード法の詳細な計算方法は割愛するが，クラスタを作るごとに重心を計算しなおし，クラスタ内の各データ点と重心の距離も計算しなおすため，郡平均法と比べると計算量が多い．

2.2 秘密計算

2.2.1 秘密分散を用いた秘密計算

秘密計算にはいくつか方式があり，中でも秘密情報を「シェア」という複数の断片に変換する秘密分散方式は，データの処理単位が小さく，処理が高速である [2], [15]．本稿では，n 個のシェアを生成し，k 個以上のシェアからは秘密が復元できるが，k 個未満のシェアからは秘密の情報が全く漏れない (k, n) 閾値法という秘密分散方式を用いる．平文とシェア (暗号文) を区別するため，a の暗号文は $[\![a]\!]$ と書き，括弧がついていないものは平文とする．本稿で用いる主な関数を以下に示す．

加減乗算

2つの暗号文 $[\![a]\!], [\![b]\!]$ の加算，減算，乗算は，それぞれ暗号文 $[\![a+b]\!], [\![a-b]\!], [\![a \times b]\!]$ を計算する処理である．これらの演算をそれぞれ，$[\![a]\!] + [\![b]\!], [\![a]\!] - [\![b]\!], [\![a]\!] \times [\![b]\!]$ と書き，入力がベクトル $[\![\vec{a}]\!], [\![\vec{b}]\!]$ の場合にも同じ記法とする．

積和

行列 $[\![A]\!], [\![B]\!]$ の行ごとの積和を計算する処理を $\mathsf{hpsum}([\![A]\!], [\![B]\!])$ と書き，$\mathsf{hpsum}([\![A]\!], [\![B]\!])$ は長さ m のベクトル $[\![\vec{c}]\!]$ を出力する．

Group-by common

Group-by commonは，Group-by sumやGroup-by countといった様々なGroup-by演算で共通的に用いることができる中間データを生成する処理である [9]．中間データは置換表 $[\![\pi]\!]$ と，キーの値の境目かどうかを表すフラグ $[\![\vec{e}]\!]$ を含み，これらを使いまわすことで，同じキーを用いた様々なGroup-by演算を効率良く行う．

キーのベクトル $[\![\vec{k}]\!]$ を入力してGroup-by commonを行うことを式 (2)，ソート済みのバリュー属性ベクトル $[\![\vec{a'}]\!]$ と $[\![\vec{e}]\!]$ を用いてGroup-by sumを行うことを式 (3)，Group-by countを行うことを式 (4) のように記述する．

$$[\![\pi]\!], [\![\vec{e}]\!], [\![\vec{a'}]\!] \leftarrow \mathsf{groupByCommon}([\![\vec{a}]\!], [\![\vec{k}]\!]) \tag{2}$$

$$[\![\vec{c}]\!] \leftarrow \mathsf{groupBySum}([\![\vec{a'}]\!], [\![\vec{e}]\!]) \tag{3}$$

$$[\![\vec{d}]\!] \leftarrow \mathsf{groupByCount}([\![\vec{a'}]\!], [\![\vec{e}]\!]) \tag{4}$$

その他

等号判定を行う $\mathsf{equality}([\![a]\!], [\![b]\!])$ や，秘匿シャッフルを行う $\mathsf{shuffle}([\![\vec{a}]\!])$，論理演算を行う $\mathsf{and}([\![a]\!], [\![b]\!])$，$\mathsf{or}([\![a]\!], [\![b]\!])$，$\mathsf{not}([\![a]\!])$ などがある．秘匿シャッフルの入力が行列の場

合は，各行の対応関係を保ったまま行ごと入れ替える形でシャッフルを行う．また，暗号文を復号する処理を reveal($[\![a]\!]$) と書く．

2.2.2 プログラマブルな秘密計算ライブラリ MEVAL

MEVAL は筆者らが開発する秘密分散ベースの秘密計算ライブラリで，加減乗算などの基本的な演算に加えて，公開値除算，除数秘匿除算，指数といった実数演算など 100 以上の演算を組み合わせて自由にプログラムできる [14], [16]．本稿の提案アルゴリズムは必要な関数が揃っていれば MEVAL 以外の秘密計算ライブラリにも適用できるが，実装では MEVAL を用い，MEVAL で実装したプログラムを用いて実験を行った．

3. 提案手法

秘密計算階層型クラスタリングでは，データ数及びデータ数から自明な値 (クラスタリングの手順を何回繰り返すか等) 以外は秘密とする．具体的には以下の情報を秘密にしたまま計算する．

- データの特徴量
- 各データ間・クラスタ間の距離
- どのデータ点 or クラスタを結合したか
- 各クラスタ内にどのデータが含まれるか
- 各クラスタ内に含まれるデータの数

3.1 アイデア

どのクラスタに何個のデータあるか，どのデータが含まれているかといった情報を秘匿しながら階層型クラスタリングを行うためのアイデアを示す．このアイデアは，データ間やクラスタ間の距離の計算方法に関わらず利用することができる．

3.1.1 クラスタ情報の管理

階層型クラスタリングでは「各クラスタにどのデータが含まれるか」という情報を保存しておき，データ点やクラスタを結合するごとにその情報を更新していく必要がある．提案手法では図 4 のような，データ ID とクラスタ ID を一対一で対応付けた表を用いて管理する．図 4 は，図 1 や図 2 に対応するクラスタ ID テーブルで，クラスタ ID がどのように更新されていくかを左から順番に示している．

クラスタ ID テーブルの大まかな更新手順を示す．

(1) クラスタ ID の初期値として，各データに対して 0 から順番に数字を割り当てる
(2) 結合するクラスタに含まれるデータ点のクラスタ ID を新しいものに書き換える
(3) 全てのデータが 1 つのクラスタになるまで (2) を繰り返す

3.1.2 距離情報の管理

階層型クラスタリングでは，各データ点間の距離，クラスタ間の距離を保存・更新していく必要がある．秘密計算階層型クラスタリングでは以下の 3 つのテーブルをで距離情報を管理する．

- 全データ間の距離のテーブル (図 5)
- 全クラスタ間の距離のテーブル (図 6)
- 結合したクラスタ間の距離のテーブル (図 7)

データ ID①	データ ID②	距離
$[\![A]\!]$	$[\![B]\!]$	$[\![d_{AB}]\!]$
$[\![A]\!]$	$[\![C]\!]$	$[\![d_{AC}]\!]$
$[\![A]\!]$	$[\![D]\!]$	$[\![d_{AD}]\!]$
⋮	⋮	⋮
$[\![D]\!]$	$[\![E]\!]$	$[\![d_{DE}]\!]$

図 5: 距離テーブル (データ)

クラスタ ID①	クラスタ ID②	距離
$[\![0]\!]$	$[\![1]\!]$	$[\![d_{01}]\!]$
$[\![0]\!]$	$[\![2]\!]$	$[\![d_{02}]\!]$
$[\![0]\!]$	$[\![3]\!]$	$[\![d_{03}]\!]$
⋮	⋮	⋮
$[\![3]\!]$	$[\![4]\!]$	$[\![d_{34}]\!]$

図 6: 距離テーブル (クラスタ)

クラスタ ID①	クラスタ ID②	距離
$[\![0]\!]$	$[\![1]\!]$	$[\![d_{01}]\!]$
$[\![2]\!]$	$[\![3]\!]$	$[\![d_{23}]\!]$
$[\![5]\!]$	$[\![6]\!]$	$[\![d_{56}]\!]$
$[\![7]\!]$	$[\![4]\!]$	$[\![d_{74}]\!]$

図 7: 距離テーブル (結合)

図 5 の「全データ間の距離のテーブル」は，ID1 と ID2 のデータ間の距離の情報を保存するテーブルである．たとえば ID1 = A，ID2=B であれば，データ点 A と B の距離が 3 列目に入っている．図 6 の「全クラスタ間の距離のテーブル」は ID1 と ID2 のクラスタ間の距離の情報を保存するテーブルである．初期化の際は各データ点を 1 つのクラスタとみなす．図 7 の「結合したクラスタ間の距離のテーブル」は，階層型クラスタリングの出力に相当し，最終的にはこのテーブルに基づいてデンドログラムを作成する．初期化の際は空のテーブルだが，クラスタの結合を行うごとに，結合した 2 つのクラスタの ID と距離の情報を追加していく．

「全データ間の距離のテーブル」は最初に 1 度作ったあとは使い回すが，「全クラスタ間の距離のテーブル」と「結合したクラスタ間の距離のテーブル」はクラスタの結合を行うたびに更新する必要がある．

3.1.3 ポイント

秘密計算階層型クラスタリングを実現するうえでポイントとなる処理は以下の 2 つである．

- クラスタ ID テーブルの更新
- 全クラスタ間の距離テーブルの更新

3.2 秘密計算階層型クラスタリング

秘密計算階層型クラスタリングのメインアルゴリズムを **Algorithm**1 に示したのち，各関数の詳細な処理を示していく．

calcDataDist(X) は，全データ間の距離を計算する処理である．距離の計算方法は何でも良いが，本稿では式 (1) に示すユークリッド距離を用いる．ユークリッド距離は減算と積和のみで計算できる．式 (1) には平方根の計算が含まれているが，クラスタリングでは大小関係のみに注目し，

データ ID	クラスタ ID
$[\![A]\!]$	$[\![0]\!]$
$[\![B]\!]$	$[\![1]\!]$
$[\![C]\!]$	$[\![2]\!]$
$[\![D]\!]$	$[\![3]\!]$
$[\![E]\!]$	$[\![4]\!]$

データ ID	クラスタ ID
$[\![A]\!]$	$[\![5]\!]$
$[\![B]\!]$	$[\![5]\!]$
$[\![C]\!]$	$[\![2]\!]$
$[\![D]\!]$	$[\![3]\!]$
$[\![E]\!]$	$[\![4]\!]$

データ ID	クラスタ ID
$[\![A]\!]$	$[\![5]\!]$
$[\![B]\!]$	$[\![5]\!]$
$[\![C]\!]$	$[\![6]\!]$
$[\![D]\!]$	$[\![6]\!]$
$[\![E]\!]$	$[\![4]\!]$

データ ID	クラスタ ID
$[\![A]\!]$	$[\![7]\!]$
$[\![B]\!]$	$[\![7]\!]$
$[\![C]\!]$	$[\![7]\!]$
$[\![D]\!]$	$[\![7]\!]$
$[\![E]\!]$	$[\![4]\!]$

データ ID	クラスタ ID
$[\![A]\!]$	$[\![8]\!]$
$[\![B]\!]$	$[\![8]\!]$
$[\![C]\!]$	$[\![8]\!]$
$[\![D]\!]$	$[\![8]\!]$
$[\![E]\!]$	$[\![8]\!]$

図 4: クラスタ ID テーブル

Algorithm 1 秘密計算階層型クラスタリング

Require: データ $[\![X]\!](m \times n$ の行列)
Ensure: 結合したクラスタ間の距離のテーブル $[\![D_{out}]\!]$
1: クラスタ ID の初期化 $[\![cid_{new}]\!] = [\![m]\!]$
2: クラスタ ID テーブル $[\![C_{id}]\!]$ の初期化
3: $[\![D_{out}]\!]$ を空のテーブルとして初期化
4: 全データ間の距離のテーブル $[\![D_{data}]\!]$ の計算
$[\![D_{data}]\!] = \mathsf{calcDataDist}([\![X]\!])$
5: 全クラスタ間の距離テーブル $[\![D_{clust}]\!]$ の計算
$[\![D_{clust}]\!] = \mathsf{calcClustDist}([\![X]\!])$
6: **for** $i = 0, 1, \cdots, n-2$ **do**
7: 最も近いクラスタの ID($[\![cid_1]\!], [\![cid_2]\!]$) と距離 $[\![d]\!]$ を取得
$[\![cid_1]\!], [\![cid_2]\!], [\![d]\!] = \mathsf{getClosestClust}([\![D_{clust}]\!])$
8: $[\![C_{id}]\!]$ の更新
$[\![C_{id}]\!] = \mathsf{updateCid}([\![cid_1]\!], [\![cid_2]\!], [\![C_{id}]\!], [\![cid_{new}]\!])$
9: $[\![D_{out}]\!]$ の更新
$[\![D_{out}]\!] = \mathsf{updateDout}([\![cid_1]\!], [\![cid_2]\!], [\![d]\!])$
10: $[\![D_{clust}]\!]$ の更新
$[\![D_{clust}]\!] = \mathsf{updateClustDist}([\![cid_1]\!], [\![cid_2]\!], [\![D_{data}]\!])$
11: $[\![cid_{new}]\!] = [\![cid_{new}]\!] + 1$

平方根の計算は省略しても結果は変わらないため，提案手法でも平方根の計算は省略した．

calcClustDist(X) は，全クラスタ間の距離を計算処理だが，初期状態では各データ点をクラスタとみなしているため，全データ間の距離のテーブルと同じである．

getClosestClust(D_{clust}) は，最も近い 2 つのクラスタの ID と距離を取得する処理で，D_{clust} を距離をキーにして秘密計算上でソートしたのち，先頭の要素 (距離が最小の要素) を取得する．

updateCid の計算手順は **Algorithm**2 に示す．

Algorithm 2 updateCid(クラスタ ID テーブルの更新)

Require: $[\![cid_1]\!], [\![cid_2]\!], [\![C_{id}]\!], [\![cid_{new}]\!]$
Ensure: 更新した $[\![C_{id}]\!]$
1: C_{id} からクラスタ ID の列 $[\![\vec{c_{id}}]\!]$ を取り出す
2: $[\![\vec{c_1}]\!] = \mathsf{equality}([\![\vec{c_{id}}]\!], [\![cid_1]\!])$
3: $[\![\vec{c_2}]\!] = \mathsf{equality}([\![\vec{c_{id}}]\!], [\![cid_2]\!])$
4: $[\![\vec{c}]\!] = \mathsf{or}([\![\vec{c_1}]\!], [\![\vec{c_2}]\!])$
5: $[\![\vec{c_{not}}]\!] = \mathsf{not}([\![\vec{c}]\!])$
6: $[\![\vec{c}]\!] = [\![\vec{c}]\!] \times [\![cid_{new}]\!]$
7: $[\![\vec{c_{not}}]\!] = [\![\vec{c_{not}}]\!] \times [\![\vec{c_{id}}]\!]$
8: $[\![\vec{c_{id}}]\!] = [\![\vec{c}]\!] + [\![\vec{c_{not}}]\!]$
9: $[\![C_{id}]\!]$ のクラスタ ID の列を $[\![\vec{c_{id}}]\!]$ に置き換える

updateDout($[\![cid_1]\!], [\![cid_2]\!], [\![d]\!]$) は，単純に $[\![D_{out}]\!]$ の末尾に $[\![cid_1]\!], [\![cid_2]\!], [\![d]\!]$ を追加するだけの処理である．

updateClustDist(全クラスタ間の距離テーブルの更新) の計算手順は，大きく以下の 2 つの処理からなる．

(1) クラスタの結合によって不要になった情報を $[\![D_{clust}]\!]$ から削除する
(2) 新たに結合したクラスタと他のクラスタの距離を計算して $[\![D_{clust}]\!]$ に追加する

クラスタの結合によって不要になった情報を削除する際の計算手順を **Algorithm**3 に示す．

Algorithm 3 不要になった情報の削除アルゴリズム

Require: $[\![cid_1]\!], [\![cid_2]\!], [\![D_{data}]\!]$
Ensure: 不要になった情報を削除した $[\![D_{clust}]\!]$
1: $[\![D'_{data}]\!] = \mathsf{shuffle}([\![D_{data}]\!])$
2: $[\![D'_{data}]\!]$ から ID1 の列 $[\![\vec{id_1}]\!]$ を取り出す
3: $[\![\vec{c_1}]\!] = \mathsf{equality}([\![\vec{id_1}]\!], [\![cid_1]\!])$
4: $[\![\vec{c_2}]\!] = \mathsf{equality}([\![\vec{id_1}]\!], [\![cid_2]\!])$
5: $[\![\vec{c}]\!] = \mathsf{or}([\![\vec{c_1}]\!], [\![\vec{c_2}]\!])$
6: $[\![D'_{data}]\!]$ から ID2 の列 $[\![\vec{id_2}]\!]$ を取り出す
7: $[\![\vec{c_1}]\!] = \mathsf{equality}([\![\vec{id_2}]\!], [\![cid_1]\!])$
8: $[\![\vec{c_2}]\!] = \mathsf{equality}([\![\vec{id_2}]\!], [\![cid_2]\!])$
9: $[\![\vec{c'}]\!] = \mathsf{or}([\![\vec{c_1}]\!], [\![\vec{c_2}]\!])$
10: $[\![\vec{c}]\!] = \mathsf{or}([\![\vec{c}]\!], [\![\vec{c'}]\!])$
11: $[\![\vec{c_{not}}]\!] = \mathsf{not}([\![\vec{c}]\!])$
12: $\vec{c_{not}} = \mathsf{reveal}([\![\vec{c_{not}}]\!])$
13: $\vec{c_{not}}$ のフラグが 0 となっている箇所に基づいて $[\![D'_{data}]\!]$ から不要な行を削除する

結合して 1 つのクラスタになる $[\![cid_1]\!], [\![cid_2]\!]$ に関する情報は不要になるため，それ以外の情報を残す処理を行っている．$[\![\vec{c_{not}}]\!]$ を復号しているが，事前にシャッフルしているため復号結果から分かる情報は削除される行数だけである．各ステップにおいて削除される行数は，学習データのレコード数から自明であるため問題無い．

次に，クラスタの結合によって新しくできたクラスタと，他のクラスタの距離の計算手順について述べる．具体例として，データ点が全部で 4 つあり，ID=1 と ID=2 のクラスタを結合させたあとの処理のイメージを図 8 に示す．

ID=1 と ID=2 のクラスタを結合して新しいクラスタとしたのち，新しいクラスタと他のクラスタ (ID=0,ID=3) との距離を求める．そのために，全データ間の距離テーブ

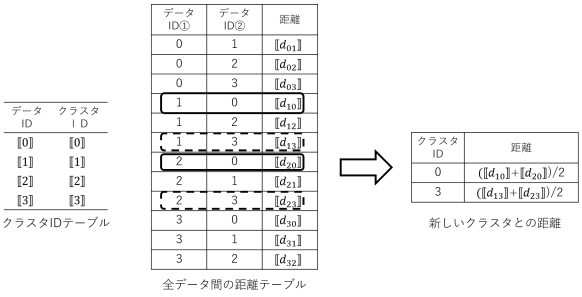

図 8: 新しい距離情報の計算イメージ

ルから持ってくる必要のあるデータは，以下の 2 つの条件を同時に満たす．

(1) データ ID1 が，結合する 2 つのクラスタに含まれる
(2) データ ID2 が，結合する 2 つのクラスタに含まれない

結合する 2 つのクラスタに含まれるデータは ID=1 と ID=2 であり，結合する 2 つのクラスタに含まれないデータは ID=0 と ID=3 であるため，ID1 と ID2 の組み合わせが (1,0),(1,3),(2,0),(2,3) の 4 つのデータを取り出せれば良い．群平均法を用いてクラスタ間の距離を求める場合，(1,0) と (2,0) の距離の平均値が新しいクラスタと ID=0 のクラスタとの距離，(1,3) と (2,3) の距離の平均値が新しいクラスタと ID=3 のクラスタとの距離になる．

全データ間の距離テーブルから，新しいクラスタと他のクラスタ間の距離を求める際に Group-By 演算を用いるが，そのためには Group-By 演算を行うための key 情報をうまく作る必要がある．具体的には，図 9 に示すような key を作れれば，所望の Group-By 演算の結果を得ることができる．

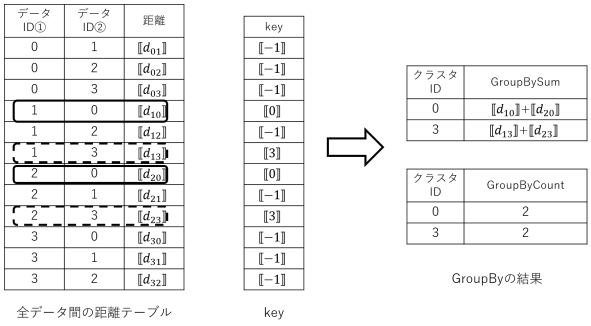

図 9: GroupBy のイメージ

実際に key を作るための処理の流れを以下に示し，処理の流れに対応する具体例を図 10 に示す．図 10 で得られた key が，図 9 で必要とする key と一致している．この key を用いて，全データの距離テーブルの 3 列目 (距離) をバリュー属性として Group-By 演算を実行すれば良い．不要な部分の情報は ID=-1 となって先頭に来るため，Group-By の計算結果から先頭要素以外を取り出せば欲しい情報だけが残る．

(1) 全データ間の距離テーブルの ID1 が，結合する 2 つのクラスタに含まれるデータは 1，それ以外は 0 となるフラグベクトルを作成する
(2) 全データ間の距離テーブルの ID2 が，結合する 2 つのクラスタに含まれるデータは 0，それ以外は 1 となるフラグベクトルを作成する
(3) (1) と (2) の論理積をとる
(4) (3) の全要素から 1 を引いたベクトルを作成する
(5) (3) にデータ ID に対応するクラスタ ID をかける
(6) (4),(5) を足す

詳細な処理の一部は割愛しているが，equality と論理演算，加減乗算といった軽量な演算の組み合わせのみで (1)〜(6) の処理を実現できる．これによって，どのクラスタに何個のデータが含まれるか，どのクラスタ同士が結合したのかといった情報を全て暗号化したまま，新しいクラスタに関する距離情報が計算できる．

Algorithm1 の updateClustDist(全クラスタ間の距離テーブルの更新) は **Algorithm**3 による古い距離情報の削除と，先述の方法による新しいクラスタに関する距離情報の計算によって実現する．

4. 実験

4.1 実験環境

秘密計算サーバとして表 1 に示すマシン 3 台を用いて実験を行った．

表 1: 測定環境

OS	CentOS Linux release 7.3.1611
CPU	Intel Xeon Gold 6144k(3.50GHz 8 コア/16 スレッド) × 2
メモリ	768GB
NW	Intel Ethernet Controller X710/X557-AT 10G リング構成

4.2 処理時間の評価

10 属性× 10〜100 件のデータを用いて，**Algorithm**1 で提案した秘密計算階層型クラスタリングの処理時間を測定した結果を図 11 に示す．

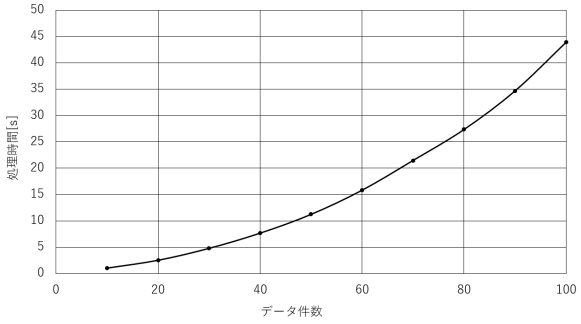

図 11: 秘密計算階層型クラスタリングの処理時間 [s]

50 件の処理時間は約 11 秒，100 件の処理時間は約 44 秒

データ ID①	データ ID②	距離
0	1	$[\![d_{01}]\!]$
0	2	$[\![d_{02}]\!]$
0	3	$[\![d_{03}]\!]$
1	0	$[\![d_{10}]\!]$
1	2	$[\![d_{12}]\!]$
1	3	$[\![d_{13}]\!]$
2	0	$[\![d_{20}]\!]$
2	1	$[\![d_{21}]\!]$
2	3	$[\![d_{23}]\!]$
3	0	$[\![d_{30}]\!]$
3	1	$[\![d_{31}]\!]$
3	2	$[\![d_{32}]\!]$

全データ間の距離テーブル

key
$[\![0]\!]$
$[\![0]\!]$
$[\![0]\!]$
$[\![1]\!]$
$[\![1]\!]$
$[\![1]\!]$
$[\![1]\!]$
$[\![1]\!]$
$[\![1]\!]$
$[\![0]\!]$
$[\![0]\!]$
$[\![0]\!]$

(1)

key
$[\![0]\!]$
$[\![0]\!]$
$[\![1]\!]$
$[\![1]\!]$
$[\![0]\!]$
$[\![1]\!]$
$[\![1]\!]$
$[\![0]\!]$
$[\![1]\!]$
$[\![1]\!]$
$[\![0]\!]$
$[\![0]\!]$

(2)

key
$[\![0]\!]$
$[\![0]\!]$
$[\![0]\!]$
$[\![1]\!]$
$[\![0]\!]$
$[\![1]\!]$
$[\![1]\!]$
$[\![1]\!]$
$[\![1]\!]$
$[\![0]\!]$
$[\![0]\!]$
$[\![0]\!]$

(3)

key
$[\![-1]\!]$
$[\![-1]\!]$
$[\![-1]\!]$
$[\![0]\!]$
$[\![-1]\!]$
$[\![0]\!]$
$[\![0]\!]$
$[\![-1]\!]$
$[\![0]\!]$
$[\![-1]\!]$
$[\![-1]\!]$
$[\![-1]\!]$

(4)

key
$[\![0]\!]$
$[\![0]\!]$
$[\![0]\!]$
$[\![0]\!]$
$[\![0]\!]$
$[\![3]\!]$
$[\![0]\!]$
$[\![0]\!]$
$[\![3]\!]$
$[\![0]\!]$
$[\![0]\!]$
$[\![0]\!]$

(5)

key
$[\![-1]\!]$
$[\![-1]\!]$
$[\![-1]\!]$
$[\![0]\!]$
$[\![-1]\!]$
$[\![3]\!]$
$[\![0]\!]$
$[\![-1]\!]$
$[\![3]\!]$
$[\![-1]\!]$
$[\![-1]\!]$
$[\![-1]\!]$

(6)

図 10: key 作成の流れ

となり，レコード数の 2 乗に比例することが分かった．また，scipy[1] の階層型クラスタリング (linkage 関数) の処理時間を測定したところ，10 件で 0.228[ms](提案手法は約 1 秒)，100 件で 0.300[ms](提案手法は約 44 秒) であった．平文と比較すると処理時間はかかっているが，十分に実用的な時間内で処理できることが確認できた．

4.3 分類精度の評価

表 2 に示す 2 属性× 10 件のデータに対して，提案手法を用いた秘密計算階層型クラスタリング，及び scipy を用いた平文での階層型クラスタリングを行った．scipy を用いた平文での階層型クラスタリングでも提案手法と同様にデータ間の距離計算方法はユークリッド距離，クラスタ間の距離計算方法は群平均法とした．

表 2: サンプルデータ

id	X_1	X_2
0	44	96
1	98	74
2	53	97
3	18	36
4	55	24
5	42	13
6	65	49
7	26	4
8	32	51
9	27	58

それぞれで得られた結果を，scipy の dendrogram 関数を用いて樹形図にした結果を図 12，13 に示す．図 12，13 の樹形図の縦軸は距離に相当する．

提案手法と平文で分類結果は完全に一致し，距離もほぼ一致していることから，提案手法の秘密計算階層型クラスタリングが正しく計算できていることが確認できた．

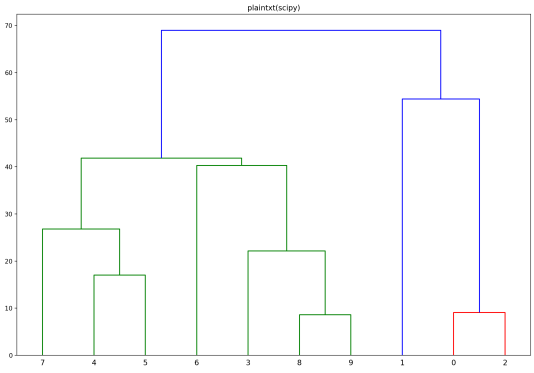

図 12: 平文 (scipy)

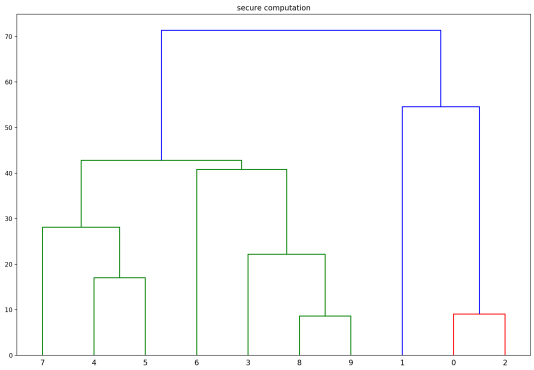

図 13: 提案手法

5. おわりに

本稿の貢献は以下である．

- 学習データ，計算途中の値を全て暗号化したまま安全に階層型クラスタリングを行う手法を実現

- 実用的なクラスタ間の距離計算方法である群平均法を秘密計算上で実現

50 件のデータに対する秘密計算階層型クラスタリングを 11 秒で処理し，またクラスタリング結果は平文と完全に一致した．

参考文献

[1] scipy. https://www.scipy.org/.

[2] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *CCS*, pp. 805–817, 2016.

[3] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, Vol. 28, No. 1, pp. 100–108, 1979.

[4] Ali Inan, Selim V Kaya, Yücel Saygın, Erkay Savaş, Ayça A Hintoğlu, and Albert Levi. Privacy preserving clustering on horizontally partitioned data. *Data & Knowledge Engineering*, Vol. 63, No. 3, pp. 646–666, 2007.

[5] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, Vol. 32, No. 3, pp. 241–254, 1967.

[6] Xianrui Meng, Dimitrios Papadopoulos, Alina Oprea, and Nikos Triandopoulos. Privacy-preserving hierarchical clustering: Formal security and e icient approximation. 2019.

[7] Mina Sheikhalishahi and Fabio Martinelli. Privacy preserving hierarchical clustering over multi-party data distribution. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, pp. 530–544. Springer, 2017.

[8] 三品気吹, 濱田浩気, 五十嵐大. 秘密計算ディープラーニングは速いだけでは使えない. In *SCIS*, 2020.

[9] 菊池亮, 濱田浩気, 五十嵐大, 高橋元, 高橋克巳. 横断的動線分析を秘密計算でやってみよう. In *SCIS*, 2020.

[10] 三品気吹, 深見匠, 濱田浩気, 五十嵐大, 菊池亮. 秘密計算によるプライバシー保護生存時間解析. In *CSS*, 2020.

[11] 三品気吹, 濱田浩気, 五十嵐大, 菊池亮. 秘密実数演算を用いた高速かつ高精度なロジスティック回帰とデータ標準化. In *CSS*, 2020.

[12] 市川敦謙, 須藤弘貴, 竹之内大地, 菊池亮, 濱田浩気, 五十嵐大. 秘密計算において高速な初等関数が機械学習や高度な統計にもたらす実用性. In *SCIS*, 2020.

[13] 深見匠, 三品気吹, 五十嵐大. 秘密計算による正則化線形回帰分析. In *CSS*, 2020.

[14] 五十嵐大. 秘密計算 ai の実装に向けた秘密実数演算群の設計と実装-o(p\) ビット通信量 o(1) ラウンドの実数向け右シフト-. In *CSS*, 2019.

[15] 五十嵐大, 濱田浩気, 菊池亮, 千田浩司. 超高速秘密計算ソートの設計と実装：秘密計算がスクリプト言語に並んだ日. In *CSS*, 2017.

[16] 桐淵直人, 五十嵐大, 濱田浩気, 菊池亮. プログラマブルな秘密計算ライブラリ MEVAL3. *SCIS*, 2018.