プライバシ保護ゲノム解析のための秘密計算フィッシャー正確検定の実装評価

長谷川 聡[1] 濱田 浩気[1] 三澤 計治[2,3] 千田 浩司[1] 荻島 創一[2,3] 長崎 正朗[2,3]

概要：ゲノム解析におけるプライバシ保護のため, ゲノムデータを秘匿しつつ解析可能とする研究が活発に行われている. ゲノムワイド関連解析（GWAS）の重要な検定手法としてフィッシャー正確検定が知られるが, ゲノムデータを秘匿しつつ実行するためには次の二つの技術課題がある：(1) フィッシャー正確検定の超幾何分布を効率よく求める方法が自明でない, (2) GWAS では数十万から数千万回のフィッシャー正確検定を行う場合があり全体の計算時間が膨大となる. 筆者らは 第 74 回 CSEC において, これらを解決する手法を提案した. 本研究では, その手法の効率的な実装を行い, 計算時間の評価を行った. 具体的には，フィッシャー正確検定の入出力対応表を決定木表現する手法について，メモ化を適用して, N を標本数とするとき, シンプルな実装より計算時間を $1/\log N$ 倍効率良くする実装を行った. また, 実験により, $N = 1,000$ の分割表を 100 万回処理する場合，フィッシャー正確検定の入出力対応表を単純に等号判定の秘密計算を繰り返す手法では 20 年以上かかる見積もりに対し，提案方式では実測で 8 分以内となり, 十分実用的であることを確認した.

Implementation and Evaluation of Privacy-Preserving Fisher's Exact Test for GWAS

SATOSHI HASEGAWA[1] KOKI HAMADA[1] KAZUHARU MISAWA[2,3] KOJI CHIDA[1]
SOICHI OGISHIMA[2,3] MASAO NAGASAKI[2,3]

1. はじめに

近年の ICT(Information and Communication Technology) 技術の発達に伴い, 買物履歴, 位置・移動情報, バイタル情報等のパーソナルデータ（個人に関する情報）が容易に収集できるようになった. それに伴い, それらの情報の利活用による新たな価値創造への期待が高まっている. しかしパーソナルデータには個人を特定可能な情報が含まれており, 取り扱う際はプライバシの保護に十分配慮する必要がある. このような背景から, データ提供者のプライバシを保護しつつ, 提供データをデータマイニング等に利活用可能とする**プライバシ保護データマイニング**（Privacy Preserving Data Mining: PPDM）の研究が活発に進められている [1], [2][*1].

1.1 秘密計算

PPDM の研究に先駆けて, 入力データを演算実行者に明かさず計算させることができる**秘密計算** (Secure Computation) が古くから研究されている [3]. 秘密計算は一般に, 入力データを秘匿化し提供する主体と, 入力データを復元せずに関数を実行する主体の少なくとも 2 者が存在する. 秘匿化の手法として暗号化や秘密分散 [11] 等が用いられる. Yao は, 組合せ論理回路を実行可能な状態のまま秘匿化する主体と, 秘匿化された組合せ論理回路 (Garbled Circuit) を実行する主体からなる秘密計算を提案した [12]. この方法では, 組合せ論理回路により汎用性の高い演算が可能となる. なお, 秘密計算において複数の主体がそれぞれ知られたくないデータを持ち, 協調してある関数を計算する場合は**セキュアマルチパーティ計算** (Secure Multi-Party Computation: MPC) とも呼ばれる.

加減算や乗算の秘密計算についても多くの研究結果が知

[1] NTT セキュアプラットフォーム研究所
[2] 東北大学東北メディカル・メガバンク機構
[3] 東北大学大学院医学系研究科
[*1] PPDM はデータマイニングに限らず統計解析等を含むデータ分析全般を対象とすることが多い.

られている．Ben-Or ら [13] と Chaum ら [14] は秘密分散を用いて加減算と乗算ができる MPC を提案した．Cramer ら [15] は加法準同型暗号を用いて同様の MPC を実現した．加減算と乗算の組合せで回路素子の演算を構成し，前記の MPC を組合せ論理回路による汎用性の高い秘密計算とすることもできる．

秘密計算は，前記のような汎用性高く実行可能な手法の研究が進められてきたが，必ずしも処理効率が良くないため，関数を限定した特化型の手法も研究が行われるようになった．例えば，等号判定 [16], [17], [18], 大小比較 [16], [17], [18], ランダムシャッフル [19], [20], ソート [19], [20] などの基本的なアルゴリズムに関する秘密計算の研究がある．

こういった基本的なアルゴリズムの研究が盛んになるにつれ，実用的な演算も実現されるようになってきた．例えば，疾患と遺伝子の関連等を調べる全ゲノム関連解析（Genome-Wide Association Study: GWAS）[4] を，ゲノムデータ提供者のプライバシを保護したまま実行可能にする秘密計算の研究もその 1 つである [6], [7]．GWAS において，ある特定の遺伝子と疾患との関係を調べるために，統計検定がよく利用される．Lu ら や Kamm らは，χ^2 検定を違う方式の秘密計算で，GWAS を実現する手法を提案している [6], [7]．χ^2 検定はフィッシャー正確検定 [5] と呼ばれる検定手法の近似法であり，Yates により，検定に用いる分割表の度数が小さい場合，近似誤差が大きくなることが指摘されている [10]．

1.2 秘密計算フィッシャー正確検定

本研究では，GWAS の重要な検定手法である，**フィッシャー正確検定** [5] に特化した秘密計算 (以降これを秘密計算フィッシャー正確検定と呼ぶ) に着目する．

秘密計算フィッシャー正確検定を実現するためには次の二つの技術課題がある：(1) フィッシャー正確検定の超幾何分布を秘密計算で効率よく求める方法が自明でない，(2) GWAS では数十万から数千万回のフィッシャー正確検定を行う場合があり全体の計算時間が膨大となる．筆者らは第 74 回 CSEC において，課題 (1) と課題 (2) の解決に取り組んだ [24], [25], [26]．

課題 (1) に対し，全パターンの集計表とそれらの検定結果からなる対応表を事前に決定木として表現しておき，実際の処理では集計表の値を秘匿しつつ決定木を実行するだけで済む手法を提案した (以降これを決定木法と呼ぶ)[24]．対応表を決定木として置き換えた理由として，標本数を N として固定したとき，全パターンの 2×2 集計表および当該検定結果からなる対応表のサイズは $\Theta(N^3)$ となるため，標本数が多いと作成困難となるからである．

また課題 (2) に対しては，フィッシャー正確検定よりも軽量な演算を用いて，帰無仮説を棄却できる集計表の候補を絞り込むことで，フィッシャー正確検定の実行回数を削減する手法を提案した (以降これを篩法と呼ぶ)[26]．具体的には，フィッシャー正確検定の中間出力が実際の確率よりも大きくならないことを利用し，当該中間出力が有意水準以下となる集計表のみを，帰無仮説を棄却できる候補とする．ただしどの集計表が候補となったか，また候補数だけでも，最終的な出力以外の情報を与えてしまうため，候補情報を秘匿しつつ処理できるようにした．

1.3 貢献

本研究の貢献は，以下の 3 つである．

(1) 決定木の手法のサイズが $N^{1.6}$〜$N^{1.7}$ に比例することの実験的な確認．
(2) 決定木法における計算時間を，シンプルな方法より $1/\log N$ 倍高速化する効率的な実装．
(3) 決定木法と篩法を実装し，実用的な時間での動作の確認．

[24] では，提案した決定木法における決定木のサイズが，$O(N^3)$ となることしか確認できていなかった．もしサイズを見積もることができれば，標本数に対する計算時間の見積もりが可能となる．そこで本研究では，実験的なアプローチにより，決定木のサイズが，$N^{1.6}$〜$N^{1.7}$ に比例する程度に圧縮できることを確認する．

更に，決定木法を効率的な実装で実現し，その時間計算量を評価した．具体的には，サイズが $N^{1.6}$〜$N^{1.7}$ に比例する程度に圧縮できる決定木の計算を，途中重複計算を伴わない効率の良いアルゴリズムで実装することにより，シンプルな実装より $1/\log N$ 倍高速化した．

また，決定木法および篩法を実装し，提案方式の有効性を検証した．標本数 1,000, SNP 数 100 万として性能シミュレーションを行ったところ，単純な方法では 19 年以上かかる処理が，提案方式では実測で 8 分以内となり，十分実用的であることを確認した．

本論文の構成を示す．2 節において，フィッシャー正確検定，GWAS を説明し，本論文で用いる秘密計算の演算を示す．3 節で従来の秘密計算フィッシャー正確検定を示し，4 節に，シンプルな法を効率よくした手法を示す．5 節に，提案法を実装し計算時間を評価した結果を示す．6 節に，まとめを述べる．

2. 準備

2.1 フィッシャー正確検定

フィッシャー正確検定は，2 つ以上のカテゴリーの独立性の検定を行う手法である．表 1 のような 2×2 の分割表 (度数表) を考える．ここで $X = a + b$, $Y = a + c$, $N = a + b + c + d$. すると表 1 の分割表が得られる確率 $P(a)$ は以下の超幾何分布 $P(z)$ から得られる：

表 1 2×2 の分割表

	Yes	No	Total
Category A	a	b	X
Category B	c	d	$N-X$
Total	Y	$N-Y$	N

$$P(z) = \frac{X!Y!(N-X)!(N-Y)!}{N!z!(X-z)!(Y-z)!(N-X-Y+z)!}. \quad (1)$$

フィッシャー正確検定 (両側検定) の有意確率 (p 値) は, $P(a)$ よりも極端な分割表の確率も考慮し,

$$P = \sum_{P(i) \le P(a)} P(i) \quad (2)$$

で与えられる. ただし

$$\max(0, X+Y-N) \le i \le \min(X, Y).$$

最終的に P と有意水準 α との大小関係により統計的な差の有無を得る. 具体的には,

$$P < \alpha \quad (3)$$

であれば帰無仮説が棄却され, 「カテゴリー A とカテゴリー B には統計的な差が無いとは言えない」と帰結される. α の値は通常 0.05 や 0.01 等が用いられるが, ゲノム解析のように多重に検定を行う場合は, Bonferroni 補正法 [8] 等によって α の値が非常に小さくなることもある.

式 (1) から得られる $P(a)$ の計算を効率化する工夫として, 対数をとる方法がよく知られる. 非負整数 n について $\log(n!) = \sum_{i=1}^{n} \log i$ が成り立つことから, $\ell_n := \sum_{i=1}^{n} \log i$ として式 (1) の対数

$$\log P(z) = \ell_X + \ell_Y + \ell_{N-X} + \ell_{N-Y} - (\ell_N + \ell_z + \ell_{X-z} + \ell_{Y-z} + \ell_{N-X-Y+z}) \quad (4)$$

を計算する. 特に標本数 N を固定して $\ell_1, \ell_2, \ldots, \ell_N$ を事前計算しておけば, 式 (4) を効率良く計算できる.

2.2 GWAS

GWAS では, ある集団のゲノム塩基配列中に見られる個人によって異なるものである一塩基多型 (Single Nucleotide Polymorphism: SNP) や一塩基バリアント (Single Nucleotide Variants: SNVs) に基づき, 疾患等との関連を統計的に調べることが行われる. SNP や SNVs は多数発見されており, 例えば日本人のゲノムの解析によって 2,120 万箇所におよぶ SNVs が発見されたという報告がある [9]. GWAS では通常全ての SNP や SNVs に対して網羅的に検定を行うため, すなわち数千万種類の分割表を作成して仮説検定を行う場合もある. 仮説検定は, χ^2 検定による独立性の検定や, フィッシャー正確検定等が用いられる. χ^2 検定はフィッシャー正確検定の近似であり簡便に計算可能だが, 分割表に小さい度数が存在する場合等に誤差の影響が大きくなることが指摘されている [10].

2.3 秘密計算

秘密計算でフィッシャー正確検定を実現するにあたり利用する秘密計算の演算を示す. 本研究では特に, 以下の演算を用いる.

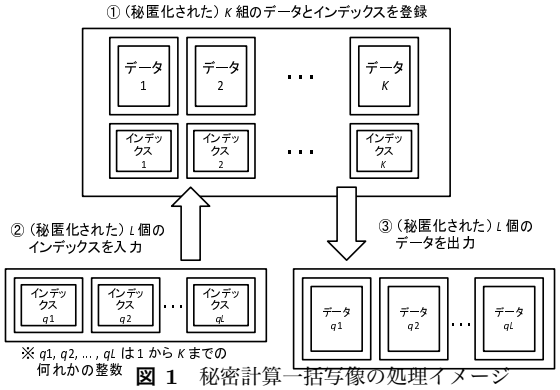

図 1 秘密計算一括写像の処理イメージ

算術演算 加減算, 乗算

比較 等号判定, 大小比較

その他 ランダムシャッフル, ソート, 一括写像

Damgård ら [17] と Nishide, Ohta[18] は秘密分散を用いて等号判定や大小比較の MPC を構成している. 濱田らはいくつかの既存の効率的なソートアルゴリズムをランダムシャッフルを用いた MPC で実現できる手法を提案している [19], [20]. 一括写像は, 図 1 に示すように, ある整数 K, L について K 組の (秘匿化された) データとインデックスがあるとき, L 個の秘匿化されたインデックスを入力すると, それらのインデックスに紐づいた L 個の秘匿化されたデータを出力する [21], [22]. これらは何れも秘密計算の部品として利用することができる. すなわち, 加減算, 乗算, 等号判定, 大小比較, ランダムシャッフル, ソート, 一括写像の結果を秘匿化したまま次の計算の入力にできる.

3. 秘密フィッシャー正確検定

1 節で述べたように, 秘密計算フィッシャー正確検定を実現するためには次の二つの技術課題がある：(1) フィッシャー正確検定の超幾何分布を効率よく求める方法が自明でない, (2) GWAS では数十万から数千万回のフィッシャー正確検定を行う場合があり全体の計算時間が膨大となる. 本節では第 74 回 CSEC にて前記課題 (1),(2) を対策した手法を紹介する [24], [26].

3.1 決定木法

3.1.1 基本アイディア

筆者らは, 式 (1),(2) を効率よく求める方法は自明でなかったため, 事前に全ての集計表パターンについてフィッシャー正確検定を実行しておき, 集計表と検定結果の対応表を作成しておくアプローチを着想した. これにより, 実際に検定を行う集計表は, 対応表と照合するだけで済む. しかし 2×2 分割表において標本数 N および有意水準 α を固定したとしても, 対応表のサイズは $O(N^3)$ となるため,

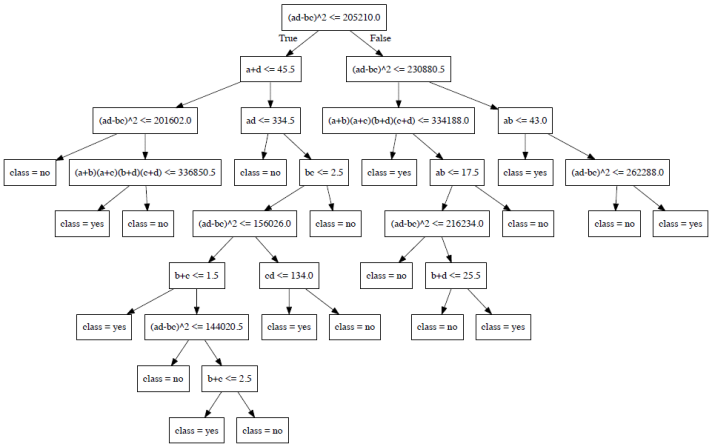

図 2 フィッシャー正確検定の決定木の例 ($N = 50, \alpha = 10^{-8}$). 決定木の作成は, Python の scikit-learn の DecisionTreeClassifier 関数を用い, 分割基準は情報量とした. 特徴量は, 分割表の度数: a, b, c, d, 2 次単項式: $a^2, b^2, c^2, d^2, ab, ac, ad, bc, bd, cd$, 1 次 2 項多項式: $a+b, a+c, a+d, b+c, b+d, c+d$, χ^2 検定の式の一部: $(ad-bc)^2, (a+b)(a+c)(b+d)(c+d)$ を用いた.

N が大きいほど実現困難となる.

そこで次に, 当該対応表を決定木として表現することでサイズを圧縮する方法を検討した. 具体例として $N = 50, \alpha = 10^{-8}$ の決定木を図 2 に示す. 図 2 の決定木は, class=yes の葉に到達すれば $P < \alpha$ となる. そして $N = 50, \alpha = 10^{-8}$ であれば 18 個の条件式（内部ノード）と 19 個の検定結果（葉ノード）からなる 37 個のノードで決定木を構成できることが分かる.

3.1.2 アルゴリズム

3.1.1 節のアイデアに基づき提案した, 決定木を用いた秘密計算フィッシャー正確検定のプロトコルを紹介する. 標本数 N および有意水準 α は固定とし, 決定木は事前に作成済みとする. 決定木について, i 段目の j 番目の内部ノード（またはその条件式）を v_{ij} と表記する. v_{ij} の条件式が yes であれば 1 を, no であれば 0 を出力するものとし, v_{ij} の出力を $b_{ij} \in \{0,1\}$ とする. k 番目の TRUE の葉に到達するパスを P_k と表記し, P_k に含まれる内部ノードの集合を $V_k \subseteq \{v_{ij}\}$ とする. P_k に含まれる枝のうち, v_{ij} とその子ノードをつなぐ枝を w_{ijk} と表記し, yes の枝であれば 0, no の枝であれば 1 とする. すなわち $w_{ijk} \in \{0,1\}$ となる. w_{ijk} の集合を W_k と表記する. なお j と k の順序は適当に定めてよい.

入力は分割表の度数 a, b, c, d を秘匿化した値とする (秘匿化関数を $E(\cdot)$, その逆関数を $D(\cdot)$ と表記する). 決定木の特徴量を $S = \{s_1, s_2, \ldots, s_n\}$ とする（n は定数）. フィッシャー正確検定の関数を $\mathrm{Fisher}(\cdot,\cdot,\cdot,\cdot,\cdot)$ と表記し, 出力は

$$\mathrm{Fisher}(\alpha, a, b, c, d) = \begin{cases} 1 & \text{if } P < \alpha \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Algorithm 1 決定木法

Input: $E(a), E(b), E(c), E(d)$, $\{v_{ij}\}$, (P_k, V_k, W_k), S, N, α.
Output: $\mathrm{Fisher}(\alpha, a, b, c, d)$.

1: $E(a), E(b), E(c), E(d)$ から, S を秘匿化した値 $E(s_1), \ldots, E(s_n)$ を秘密計算で求める.
2: 全ての v_{ij} について, $E(s_1), \ldots, E(s_n)$ から, v_{ij} の真偽を秘匿化した値 $E(b_{ij})$ を秘密計算で求める.
3: 全てのパス P_k について以下の (a),(b) を行う.
(a) V_k に含まれる全ての v_{ij} について, $E(b_{ij})$ および w_{ijk} から,
$$c_{ijk} = E(b_{ij} \oplus w_{ijk}) \tag{6}$$
を秘密計算で求める.
(b) c_{ijk} から,
$$d_k = E(\bigwedge_{i,j} D(c_{ijk})) \tag{7}$$
を秘密計算で求める.
4: 全ての d_k から,
$$E(\bigvee_{k} D(d_k)) \tag{8}$$
を秘密計算で求める.
5: $E(\bigvee_k D(d_k))$ を復号して出力する.

とする. 本プロトコルを, Algorithm1 に示す.

決定木法の安全性や詳細は, 文献 [24] を参照されたい.

3.2 篩法

3.2.1 基本アイディア

次に, 課題 (2) に対する単一のフィッシャー正確検定に対してではなく, GWAS のように膨大な回数のフィッシャー正確検定を一括で効率よく実行する場合について考える. 先ず秘密計算でなく通常のフィッシャー正確検定による GWAS を考えると, 式 (1),(2) を所定の回数（M とする）計算する. M の値は数十万から数千万が想定される. 式 (1) は超幾何分布であり, 式 (2) でそれを $\Omega(N)$ 回計算する. そこで式 (2) よりも軽量な演算により, 式 (3) を明らかに満たさない集計表を特定して篩い落とすアプローチを検討した. 詳細は, 文献 [26] を参照されたい. プロトコルを Algorithm2 に示す.

3.2.2 アルゴリズム

3.2.1 節で提案した基本アイデアに基づくプロトコルを以下に示す. 入力は M 組の分割表の度数 a_i, b_i, c_i, d_i $(i = 1, 2, \ldots, M)$ を秘匿化した値とする. また $e_i \in \{0,1\}$ を以下のように定義する.

$$e_i = \begin{cases} 1 & \text{if } P(a_i) < \alpha \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

3.3 秘匿性の強化

3.3.1 候補出力法

最後に篩法の出力を用いて, 式 (3) を満たす集計表の候補を求めるプロトコルを与える. 篩法の出力を復号した値 e_i が 0 の場合, 式 (3) を満たさないことは先に述べた. しかし e_i は集計表の情報を与えるため, そのまま復元することはできない. そこで, $e_i = 1$ となる集計表の数の上限 T

Algorithm 2 篩法

Input: $\{E(a_i), E(b_i), E(c_i), E(d_i)\}_{i=1}^{M}$, $\{(\ell_i, i)\}_{i=0}^{N}$, N, α.
Output: $E(e_1), E(e_2), \ldots, E(e_M)$.

1: 全ての i について, $n = a_i$ として式 (4) の右辺の各項のインデックスを秘匿化した値 $E(X_i)$, $E(Y_i)$, $E(N - X_i)$, $E(N - Y_i)$, $E(X_i - a_i)$, $E(Y_i - a_i)$, $E(N - X_i - Y_i + a_i)$ を秘密計算で求める. ただし $X_i = a_i + b_i$, $Y_i = a_i + c_i$.
2: 秘密計算一括写像を用いて, $8M$ 個の秘匿化されたインデックス $E(X_i)$, $E(Y_i)$, $E(N - X_i)$, $E(N - Y_i)$, $E(a_i)$, $E(X_i - a_i)$, $E(Y_i - a_i)$, $E(N - X_i - Y_i + a_i)$ を入力し, $8M$ 個の秘匿化されたデータ $E(\ell_{X_i})$, $E(\ell_{Y_i})$, $E(\ell_{N-X_i})$, $E(\ell_{N-Y_i})$, $E(\ell_{a_i})$, $E(\ell_{X_i-a_i})$, $E(\ell_{Y_i-a_i})$, $E(\ell_{N-X_i-Y_i+a_i})$ を求める.
3: 全ての i について以下を行う.
(a) 加減算の秘密計算を用いて, 式 (4) より $E(\ell_{X_i})$, $E(\ell_{Y_i})$, $E(\ell_{N-X_i})$, $E(\ell_{N-Y_i})$, ℓ_N, $E(\ell_{a_i})$, $E(\ell_{X_i-a_i})$, $E(\ell_{Y_i-a_i})$, $E(\ell_{N-X_i-Y_i+a_i})$ から $E(\log P(a_i))$ を求める.
(b) 大小比較の秘密計算を用いて α と $E(\log P(a_i))$ から $\log P(a_i) < \log \alpha$ の真偽 e_i を秘匿化した値 $E(e_i)$ を出力する.

Algorithm 3 候補出力法

Input: $\{E(a_i), E(b_i), E(c_i), E(d_i), E(e_i)\}_{i=1}^{M}$, T.
Output: e_i を降順に並べた上位 T 個の $(E(a_i), E(b_i), E(c_i), E(d_i))$.

1: ソートの秘密計算を用いて, $\{E(a_i), E(b_i), E(c_i), E(d_i), E(e_i)\}_{i=1}^{M}$ を e_i の降順に並べる.
2: 上位 T 組の $(E(a_i), E(b_i), E(c_i), E(d_i))$ を抽出する.
3: ランダムシャッフルの秘密計算を用いて, 上位 T 組の $(E(a_i), E(b_i), E(c_i), E(d_i))$ をランダムシャッフルして出力する.

を設定し, 必ず T 個の集計表を候補とすることを考える. すなわち, $\{E(a_i), E(b_i), E(c_i), E(d_i), E(e_i)\}_{i=1}^{m}$ を入力とし, e_i の降順にそれらを秘密計算でソートし, 上位 T 個の $(E(a_i), E(b_i), E(c_i), E(d_i))$ を抽出し, これらを決定木法の入力として秘密計算フィッシャー正確検定を実行する. 本プロトコルを Algorithm3 に示す.

このプロトコルの全体の計算時間は, 手続き 2 で用いる一括写像が支配的となり, $O((M+N)\log(M+N))$ となる.

このプロトコルの全体の計算時間は, 手続き 1 で用いるソートが支配的となり, $O(M \log M)$ となる. プロトコルの詳細については, 文献 [26] を参照されたい.

4. 決定木法の効率的な実装

筆者らは, [26] において, 決定木表現を用いることで, 決定木のサイズの圧縮を試みたが, 具体的にどの程度圧縮の効果があったかは定量的には示せていなかった. そのため, 標本数に対する計算時間を見積もることができていなかった.

本節では, 決定木法の計算時間を見積もるため, 標本数と決定木サイズの関係を明らかにする (4.1 節). 次に, 4.1 節の結果を用いて, 決定木法に要する計算時間の評価を行う. その後, 決定木法を, 効率よく実装する方法を示す.

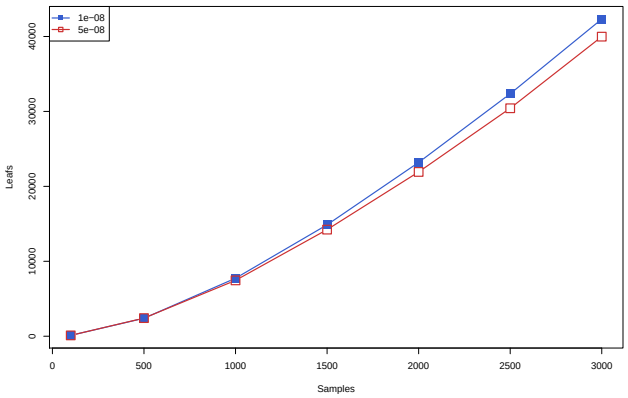

図 3 フィッシャー正確検定の決定木の葉ノード数. 横軸は標本数 N, 縦軸は決定木の葉の数 L.

4.1 標本数と決定木サイズの関係の見積もり

決定木のアルゴリズムは, データの中身や, 特徴量に依存した処理を行うため, 生成される木の構造を予測し, 解析を行うことが難しい. そこで, 標本数と決定木の葉の数の関係を数値シミュレーションすることで, 実験的にこれらの関係を明らかにする方法を検討した.

標本数 $N = 100, 500, 1000, 1500, ..., 3000$ とした際の, 標本数 N と決定木の葉の数 L の関係を図 3 に示す. なお, 決定木作成に用いた実装や特徴量は, 図 2 と同様のものを用いた. 図 3 を見る限り, 決定木表現による大きな圧縮効果が期待できる. 決定木の葉の数を L と, 標本数 N との関係が, $L = C\dot{N}^p$(C は定数) と表されると仮定し, $1 \leq p \leq 2$ で, 相関係数が最大となる p を求めた. $\alpha = 10^{-8}$ のとき,

$$L \propto N^{1.642269}, \tag{10}$$

が相関係数 $r^2 = 0.99992456$ で, $\alpha = 5 \times 10^{-8}$ のとき,

$$L \propto N^{1.623651}, \tag{11}$$

が $r^2 = 0.999922789$ で最も高かった. 決定木の葉の数 L は, 標本数 N に対し, おおよそ $N^{1.6}$〜$N^{1.7}$ に比例することがわかる. これらより, $L = C\dot{N}^p$(C は定数) と表されると仮定した場合, 葉の数 L は, $O(N^{1.7})$ と評価できる.

4.2 計算時間の見積もり

本節では, 4.1 節において見積もった, 標本数と決定木サイズとの関係を用いることで, 決定木法の計算時間を評価する. なお加減乗算, 定数サイズの大小比較, および回路素子の 1 回あたりの秘密計算の計算時間は N に依存しないため定数とする. 手続き 1 について, 決定木の特徴量の数 n は定数であり N に依存しない. 手続き 2 では, v_{ij} ごとの計算が必要となり, v_{ij} の数は節 4.1 より $O(N^{1.7})$ であり, 計算時間も $O(N^{1.7})$ と推定できる. 手続き 3(b) は, 決定木の構造に依存するが, 例えば決定木がバランスの良い平衡二分木であった場合, 深さ $O(\log N)$ であり, かつ葉の数は $O(N^{1.7})$ 程度と推定できるため, 計算時間は $O(N^{1.7} \log N)$

Algorithm 4 メモ化を用いた決定木法

Input: $E(a), E(b), E(c), E(d)$, $\{v_{ij}\}$, (P_k, V_k, W_k), S, N, α.
Output: Fisher(α, a, b, c, d).

1: $E(a), E(b), E(c), E(d)$ から, S を秘匿化した値 $E(s_1), \ldots, E(s_n)$ を秘密計算で求める.
2: 各 v_{ij} について, 以下の 2 ステップを行う
 (a) 当該 v_{ij} について, 秘匿化した値 $E(s_1), \ldots, E(s_n)$ を用いて, $E(b_{ij})$ を求める.
 (b) $E(c_{i-1,*})$ と $E(b_{ij})$ と w_{ij} から, $E(c_{ij}) = E(z_{i-1,*} \oplus b_{ij} \oplus w_{ij})$ を計算する.
3: class=yes の葉ノードを表す各 v_{ij} において, $c_{i-1,*}$ と w_{ij} から $E(d_{ij}) = E(z_{i-1,*} \oplus w_{ij})$ を計算する.
4: $E(d_{ij})$ から $E(\bigvee_{ij}(d_{ij}))$ を計算し, 結果を復号する.

と推定できる.

4.3 メモ化を用いた効率の良いプロトコル

4.3.1 基本アイディア

決定木法は, 決定木の構造に依存したプロトコルとなっており, 単純に実装すると, 決定木の葉の数それぞれに対して木の深さ ($O(\log N)$) 分の計算時間を要する. もし偏りが大きく, 深さが $\Theta(N^{1.7})$ である場合には, $\Theta(N^{3.4})$ の時間計算量となり, 計算効率が非常に悪い. このようになる理由として, 手続き 3(b) において, P_k の計算毎に, P_{k-1} で計算済みの c_{ijk} を重複して計算していることがある. 例えば, 図 2 を例に考えてみる. 決定木の右側から class=yes となるパスを辿る計算を考えてみる.

$$P_0 \quad : \quad ((A_1 > 205210) \wedge (A_1 > 230880.5) \wedge (A_2 > 43) \wedge (A_1 > 262288) \tag{12}$$

$$P_1 \quad : \quad ((A_1 > 205210) \wedge (A_1 > 230880.5) \wedge (A_2 \leq 43) \tag{13}$$

ただし, $A_1 = (ad - bc)^2$, $A_2 = ab$ とする. P_0 と P_1 の計算のうち, $A_1 > 205210$ や $A_1 > 230880.5$ の計算が重複していることがわかる.

そこで, 重複計算を省くためメモ化のテクニックを用い, 根から近い順序で木を辿ることにより, P_{k-1} で計算した c_{ijk} の再計算を省き, 葉の数分の計算のみで済むアルゴリズムを提案する.

4.3.2 アルゴリズム

決定木法の説明に利用したものと同様の記号を用いる. 加えて, v_{ij} の親ノードを $v_{i-1,*}$ と表すこととし, v_{ij} と $v_{i-1,*}$ の枝を w_{ij} と表す ($w_{ij} = 0$ の場合, True の枝を表し, $w_{ij} = 1$ の場合, False の枝を表す).

プロトコルを Algorithm4 に示す. 手続き 1 は, 決定木の特徴量の計算を, 加減乗算の秘密計算を用いて計算している. 手続き 2 は, 各 v_{ij} について, あるノードまで辿ってきた結果を, 親ノードの結果と枝の結果のみを用いて計算する. より具体的には, 対象となる v_{ij} の特徴量と条件値との比較を行い (手続き 2(a) に相当), 親ノード $E(c_{i-1,*})$ と枝の値 w_{ij} を用いて, v_{ij} までパスを辿ってきた結果 $E(c_{ij})$ を計算する (手続き 2(b) に相当). 手続き 3 は, class=yes となる葉ノードにおけるパスを辿ってきた結果を計算しており, 手続き 4 で結果を復号する. Algorithm 1 との違いは, 各ノード v_{ij} まで辿ってきた結果を保持しながら計算する点である.

本プロトコルは, 手続き 2 において $O(N^{1.7})$ の計算量で, 手続き 3 においても $O(N^{1.7})$ であるゆえ, プロトコル全体の時間計算量も $O(N^{1.7})$ となる.

5. 実験

篩法 [26] に加え, 節 4.3 の有効性を確認するため, 決定木法 (Algorithm4), 篩法 (Algorithm2), 候補出力法 (Algorithm3) を実装し, 計算時間を確認した. 加減乗算, 回路素子演算, 大小比較の秘密計算は五十嵐らの手法 [23] に基づき実装した. ランダムシャッフル, ソート, 一括写像の秘密計算は濱田らの手法 [19], [20] に基づき実装した. マルチパーティ計算で用いたマシンのスペックは以下となる.

- CPU:Intel Core i7 4930K 3.4GHz (6cores),
- RAM:32GB(DDR3 2400MHz),
- OS:CentOS 7.1.1503,
- NW:1000BASE-T switching hub.

5.1 比較対象

比較対象として, 全パターンの集計表とそれらの検定結果からなる対応表を事前に計算しておき, M 組の集計表各々について対応表を参照する方法を挙げる. 比較対象の場合, $O(N^3)$ のサイズの対応表が必要となる. より具体的には, $N = a + b + c + d$, $a, b, c, d \leq 0$ より, N を固定すれば

$$Q = \frac{N(N-1)(N-2)}{6} \tag{14}$$

行の対応表となり, 各行には a, b, c, d の値および検定結果 Fisher(α, a, b, c, d) の秘匿化された値が格納される. これらと入力の集計表を比較するため, 少なくとも $3MQ$ 回の比較が必要となる. [23] では, 10^7 回の比較の秘密計算を 12.6 s で実現した報告がある. この値を参考にすれば, $M = 10^6$, $N = 1,000$ の場合, $3MQ$ 回の比較に約 20 年かかることになる.

5.2 実行時間

$N = 100, 200, ..., 1000$, $M = 10^6$ として評価した. また, 文献 [26] より, Algorithm 3 の篩法で残す集計表の数を $T = 200$ として, 提案方式を Algorithm 2, 3, 4 の順に実行した. 図 4 にプロトコルごとの実行時間を示す.

決定木法は, N を増やすごとに計算時間が増えたが, 篩法, 候補出力法アルゴリズムはほぼ一定であった. 理由として, 篩法の時間計算量が, $O((N + M)\log(N + M))$ であり, $M \gg N$ であったため, N の実行時間に対する影響が小さかったからだといえる. 候補出力法は, そもそも時間計算量が $O(M \log M)$ であり, N に依存しないゆえ, N が実行時間に影響がなかった.

決定木法の実行時間を調査するため, N^p の p を $p =$

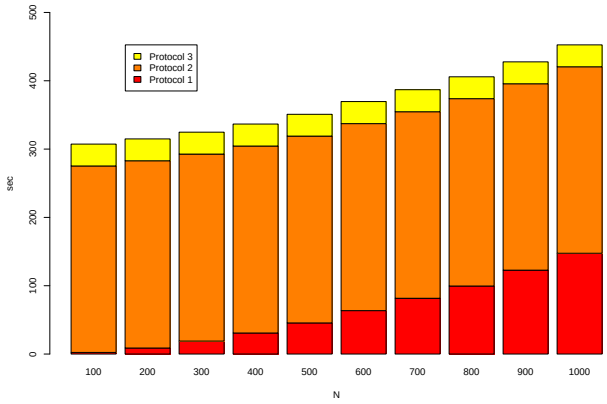

図 4 決定木法, 篩法, 候補出力法の実行時間. Protocol 1 が決定木法, 2 が篩法, 3 が候補出力法の実行時間を表す

$1, 1.00001, ..., 2$ と変化させ, 実行時間との相関係数を計測した. $\alpha = 10^{-8}$ のケースにおいて, $N^{1.65785}$ の場合相関係数が最も高くなった (相関係数 $r^2 = 0.999874495$). 結果, 実行時間も時間計算量と一致することがわかった.

比較手法が約 20 年かかると見積もられる処理が, 提案手法では, 全体で 8 分以内に処理が完了しており, 十分実用的な速度で実行できることを確認した.

6. おわりに

本研究では, GWAS 計算の重要な統計検定手法であるフィッシャー正確検定を, ゲノムデータを秘匿しつつ行うプロトコルを実装した. 特に, 提案した手法を効率よく実行可能にする実装を行い, 実行時間を評価した. 結果, 単純な方法では 20 年かかるところ, 提案方式は 8 分以内で実行可能であることを確認し, 十分実用であることを示した.

謝辞

本論文で紹介した決定木の作成は, 東北大学東北メディカル・メガバンク機構のスーパーコンピュータを利用して実行しました.

参考文献

[1] Vaidya, J., Clifton, C.W. and Zhu, Y.M.: Privacy preserving data mining, Springer-Verlag (2005)
[2] Aggarwal, C. and Yu, P.: Privacy-preserving data mining: Models and algorithms, Springer-Verlag (2008)
[3] Yao, A. C.: Protocols for secure computations, Proc. of FOCS '82, 160-164 (1982)
[4] Ozaki, K., Ohnishi, Y., Iida, A., Sekine, A., Yamada, R., Tsunoda, T., Sato, H., Hori, M., Nakamura, Y. and Tanaka, T.: Functional SNPs in the lymphotoxin-a gene that are associated with susceptibility to myocardial infarction, Nature Genetics 32, 650-654 (2002)
[5] Fisher, R.A.: On the interpretation of χ^2 from contingency tables, and the calculation of P, Journal of the Royal Statistical Society 85(1), 87-94 (1922)
[6] Kamm, L., Bogdanov, D., Laur, S., Vilo, J.: A new way to protect privacy in large-scale genome-wide association studies. Bioinformatics 29(7), 886-893 (2013), http://dx.doi.org/10.1093/bioinformatics/btt066
[7] Lu, W.J., Yamada, Y. and Sakuma, J.: Privacy-preserving genome-wide association studies on cloud environment using fully homomorphic encryption, BMC medical informatics and decision making 15(Suppl 5), S1 (2015)
[8] 瀬々潤, 濱田道昭: 生命情報処理における機械学習 多重検定と推定量設計, 講談社 (2014)
[9] Nagasaki, M., Yasuda, J., Katsuoka, F., Nariai, N., Kojima, K., Kawai, Y., Yamaguchi-Kabata, Y., Yokozawa, J., Danjoh, I., Saito, S., Sato, Y., Mimori, T., Tsuda, K., Saito, R., Pan, X., Nishikawa, S., Ito, S., Kuroki, Y., Tanabe, O., Fuse, N., Kuriyama, S., Kiyomoto, H., Hozawa, A., Minegishi, N., Douglas Engel, J., Kinoshita, K., Kure, S., Yaegashi, N.: Rare variant discovery by deep whole-genome sequencing of 1,070 Japanese individuals, Nat Commun. 6(8018), (2015)
[10] Yates, F.: Contingency tables involving small numbers and the χ^2 test, Supplement to the Journal of the Royal Statistical Society 1(2), 217-235 (1934)
[11] Shamir, A.: How to share a secret, Communications of the ACM 22(22), 612-613 (1979)
[12] Yao, A. C.: How to generate and exchange secrets, Proc. of FOCS '86, 162-167 (1986)
[13] Ben-Or, M., Goldwasser, S., and Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation, Proc. of STOC '88, 1-10 (1988)
[14] Chaum, D., Crepeau, C., and Damgård, I.: Multiparty unconditionally secure protocols, Proc. of STOC '88, 11-19 (1988)
[15] Cramer, R., Damgård, I., and Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption, EUROCRYPT 2001, LNCS 2045, Springer-Verlag, 280-300 (2001)
[16] Schoenmakers, B. and Tuyls, P.: Practical two-party computation based on the conditional gate, ASIACRYPT 2004, LNCS 3329, Springer-Verlag, 119-136 (2004)
[17] Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., and Toft, T.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation, TCC2006, LNCS 3876, Springer-Verlag, 285-304 (2006)
[18] Nishide, T. and Ohta, K.: Multiparty computation for interval, equality, and comparison without bit-decomposition protocol, PKC 2007, LNCS 4450, Springer-Verlag, 343-360 (2007)
[19] 濱田浩気, 五十嵐大, 千田浩司, 高橋克巳: 秘匿関数計算上の線形時間ソート, 第 28 回暗号と情報セキュリティシンポジウム (SCIS2011) 予稿集 (2011)
[20] Hamada, K., Kikuchi, R., Ikarashi, D., Chida, K. and Takahashi, K.: Practically efficient multi-party sorting protocols from comparison sort algorithms, ICISC 2012, LNCS 7839, Springer-Verlag, 202-216 (2012)
[21] 濱田浩気, 五十嵐大, 千田浩司: 秘密計算一括写像計算の効率化, 第 31 回暗号と情報セキュリティシンポジウム (SCIS2014) 予稿集 (2014)
[22] Laud, P.: Parallel oblivious array access for secure multiparty computation and privacy-preserving minimum spanning trees. PoPETs 2015(2), 188-205 (2015), http://www.degruyter.com/view/j/popets.2015.2015.issue-2/popets-2015-0011/popets-2015-0011.xml
[23] Dai Ikarashi, Ryo Kikuchi, Koki Hamada, Koji Chida:

Actively Private and Correct MPC Scheme in t¡n/2 from Passively Secure Schemes with Small Overhead. IACR Cryptology ePrint Archive 2014: 304 (2014)
[24] 千田浩司, 長谷川聡, 濱田浩気, 荻島創一, 三澤計治, 長崎正朗: 秘密計算フィッシャー正確検定 (1)〜標本数が少ない場合, 第 74 回コンピュータセキュリティ研究会 (CSEC74) (2016)
[25] 濱田浩気, 長谷川聡, 千田浩司, 荻島創一, 三澤計治, 長崎正朗: 秘密計算フィッシャー正確検定 (2)〜標本数が多い場合, 第 74 回コンピュータセキュリティ研究会 (CSEC74) (2016)
[26] 長谷川聡, 濱田浩気, 千田浩司, 荻島創一, 三澤計治, 長崎正朗: プライバシ保護ゲノム解析のための秘密計算フィッシャー正確検定, 第 74 回コンピュータセキュリティ研究会 (CSEC74) (2016)