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

三品 気吹[1,a)]　濱田 浩気[1]　五十嵐 大[1]　菊池 亮[1]

概要：データを暗号化したまま計算する「秘密計算」は，統計や機械学習といった様々なデータ分析を「安全」に行うことを可能にする．データ分析手法には様々あるが，実際のデータ分析で頻繁に用いられ，それを秘密計算上で行う研究も多くなされてきた手法が「ロジスティック回帰」である．秘密計算ロジスティック回帰では多くの手法が提案されおり，精度の面では平文と同等，速度の面でも比肩しつつある．本稿では，そこから更に「実用性」の面でも平文と同等を目指して，秘密計算ロジスティック回帰，及びモデル評価指標 (AIC) の秘密計算，高精度な秘密標準化処理の実現に取り組む．秘密計算ロジスティック回帰では 10 属性 1000 件のデータを約 2 秒を達成し，予測精度も平文と一致した．秘密標準化処理と評価指標の秘密計算についても実用的な性能を示した．

キーワード：秘密計算，機械学習，ロジスティック回帰

Fast and Accurate Logistic Regression and Data Standardization Using Secure Real Number Operations

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

Abstract: Secure Computation makes it possible to perform various data analyses such as statistics and machine learning in a secure manner, since the data is still encrypted. There are various data analysis methods, and logistic regression is one of the most frequently used methods in data analysis, and has been studied in secure computations. Various methods of secure logistic regression have been proposed, which are comparable to those of plain text in terms of accuracy and speed. In this paper, we aim to further improve the "practicality" of this paper by using secret logistic regression and model evaluation indexes (AIC), and a highly accurate secure standardization process. The secure logistic regression achieved about 2 seconds for 1000 data sets of 10 attributes, and the prediction accuracy was consistent with the plaintext. We also achieved high speed and high accuracy in the secret standardization process and the secret computation of evaluation indices.

Keywords: Secure Computation, Machine Learning, Logistic Regression

1. はじめに

秘密計算は，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，データ分析などに利用できる．その分析手法として特に有望なのが機械学習 (AI) である．筆者らは，平文の AI ライブラリ [3] が提供するような数多の AI 分析を，データを安全に守ったまま行う「秘密計算 AI ライブラリ」の実現を目指している．それに向け，筆者らはこれまでに秘密計算上のロジスティック回帰分析に取り組んできた [11][14]．本稿では秘密計算上ロジスティック回帰のさらなる実用性向上を目指す．

[1] NTT セキュアプラットフォーム研究所
NTT Secure Platform Laboratories
[a)] ibuki.mishina.br@hco.ntt.co.jp

1.1 関連研究

Mohassel と Zhang [8] が秘密分散に garbled circuit [10] や紛失通信 [9] といった暗号を組み合わせたものや，筆者らが秘密分散を用いた手法を提案したものなどがある [11]. これらの先行研究の主眼は，ロジスティック回帰分析の肝となる非線形関数 (シグモイド関数) の計算を如何にして秘密計算上で実現するかということであり，[11] では自作の Python スクリプトに近い性能と精度を両立するに至った．さらに濱田ら [14] の提案手法では，シグモイド関数の計算に秘密一括近似という新たな手法を用いることで非常に高い精度と高速な処理を実現している．

1.1.1 課題

これまでの秘密計算ロジスティック回帰の研究では，ロジスティック回帰分析を秘密計算で行うことにのみ着目されており，秘密計算ロジスティック回帰で計算したモデルの良し悪しが評価できるか，といったことには着目されてこなかった．しかしながら実際の分析においては，推定したモデルが良いものかどうかを評価する作業は必ず行われるため，そのような処理もロジスティック回帰とセットで出来なければ実用的とは言えない．

また秘密計算ロジスティック回帰に入力するデータに対して，事前に秘密計算で扱いやすくするような処理を加えたりしており，この点についても実用的なシチュエーションでは，そのような前処理はできない場合があるため，改善が必要である．

1.2 本稿の貢献

本稿の貢献は以下である

- 秘密実数演算を用いて高速・高精度を両立した秘密計算ロジスティック回帰
- AIC などモデル評価指標の計算を秘密計算上で実現
- 高精度なデータの標準化処理を秘密計算上で実現

特に新しいのは 2,3 点目である．AIC はロジスティック回帰の結果を評価するための重要な指標であるが，従来手法では計算されてこなかった．AIC の計算には学習データが必要であるため，ロジスティック回帰と合わせて秘密計算で計算できる必要がある．

3 点の標準化については，筆者らが以前に [12] でも少し触れているが本稿では新たな工夫を取り入れた．

2. 準備

2.1 記法

a を b で定義することを $a := b$ と書き，ベクトルを $\vec{a} := (a_0, \ldots, a_{n-1})$ と書き，同じ要素数の 2 つのベクトルの内積は $(\cdot)$ で表し，要素ごとの積は $(\circ)$ で表す．

加減乗算において入力がベクトル $\vec{a}$ もしくは行列 A とスカラー b の場合，$\vec{a}, A$ の全ての要素に対して b との演算を行うものとする．行列 A と列ベクトル $\vec{b}$ の場合は，行列の各列ベクトルに対して $\vec{b}$ との要素ごとの演算を実施し，行列 A と行ベクトル ${}^t\vec{b}$ 場合は，行列の各行ベクトルに対して ${}^t\vec{b}$ との要素ごとの演算を実施するものとする．

特に記載が無いベクトルは列ベクトルであり，横ベクトルの場合は ${}^t\vec{a}$ のように左上に t を付けることで区別する．

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 [6], [18]. 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．

2.2.2 算術演算

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．

総和

ベクトル $[\![\vec{a}]\!]$ の総和を求める処理を $\mathsf{sum}([\![\vec{a}]\!])$ と記述する．また $\mathsf{sum}([\![A]\!])$ のように $m \times n$ の行列が入力の場合は，列方向の総和を計算し，長さ n のベクトル $[\![\vec{c}]\!]$ を出力するものとする．

積和

長さが等しい 2 つのベクトル $[\![\vec{a}]\!], [\![\vec{b}]\!]$ の積和を求める処理を $\mathsf{psum}([\![\vec{a}]\!], [\![\vec{b}]\!])$ と記述する．また大きさの等しい $m \times n$ 行列 $[\![A]\!], [\![B]\!]$ を入力した場合，$\mathsf{psum}([\![A]\!], [\![B]\!])$ は列ごとの積和を計算し，長さ n の行ベクトル $[\![\vec{c}]\!]$ を出力するものとする．行列 $[\![A]\!], [\![B]\!]$ の行ごとの積和を計算する場合は，$\mathsf{hpsum}([\![A]\!], [\![B]\!])$ と書き，$\mathsf{hpsum}([\![A]\!], [\![B]\!])$ は長さ m の列ベクトル $[\![{}^t\vec{c}]\!]$ を出力する．

2.2.3 実数演算

逆数

暗号文 $[\![a]\!]$ の逆数 $1/[\![a]\!]$ を計算することを $[\![c]\!] \leftarrow \mathsf{reciprocal}([\![a]\!])$ のように書く．入力がベクトルの場合も同じ記法とする．

指数

暗号文 $[\![a]\!]$ を入力とし，ネイピア数 e の $[\![a]\!]$ 乗を計算することを $[\![c]\!] \leftarrow \mathsf{exp}([\![a]\!])$ のように書く．入力がベクトルの場合も同じ記法とする．

対数

暗号文 $[\![a]\!]$ を入力とし，自然対数を計算することを $[\![c]\!] \leftarrow \mathsf{log}([\![a]\!])$ のように書く．入力がベクトルの場合

も同じ記法とする．

2.2.4 統計演算

平均

長さ n のベクトル $[\![\vec{a}]\!]$ の平均値 $[\![\bar{a}]\!]$ を求める処理は式 (1) で実現し，これ以降は $[\![\bar{a}]\!] \leftarrow \mathsf{average}([\![\vec{a}]\!])$ のように書く．

$$[\![\bar{a}]\!] \leftarrow \mathsf{sum}([\![\vec{a}]\!])/n \tag{1}$$

入力が $m \times n$ 行列の場合は列ごとに average を適用し，長さ n の行ベクトル $[\![{}^t\vec{a}]\!]$ を出力するものとする．

分散

長さ n のベクトル $[\![\vec{a}]\!]$ の分散 $[\![a_{var}]\!]$ は，以下のように実現し，これ以降は $[\![\bar{a}]\!] \leftarrow \mathsf{var}([\![\vec{a}]\!])$ のように書く．

$$[\![\bar{a}]\!] \leftarrow \mathsf{average}([\![\vec{a}]\!]) \tag{2}$$

$$[\![\vec{a'}]\!] \leftarrow [\![\vec{a}]\!] - [\![\bar{a}]\!] \tag{3}$$

$$[\![a_{var}]\!] \leftarrow \mathsf{average}([\![\vec{a'}]\!] \times [\![\vec{a'}]\!]) \tag{4}$$

入力が行列だった場合のは average と同様列ごとに適用し，$[\![{}^t\vec{a}_{var}]\!]$ を出力する．

2.2.5 その他の処理

要素がすべて $[\![a]\!]$ で長さ b のベクトルを作成する処理を $[\![\vec{c}]\!] \leftarrow \mathsf{fill}([\![a]\!], b)$ と書く．

2.2.6 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，前述したような演算を組み合わせて自由にプログラムできる [19]．本稿の実験で用いたプログラムは，MEVAL を用いて実装している．

2.3 ロジスティック回帰

ロジスティック回帰 [7] は，ある変数 x(説明変数) と別の変数 y(目的変数) の関係式 (モデル)$y = f(x)$ を求める統計・機械学習の手法の一種で，目的関数が「病気にかかったか否か」のような 2 値であるのが特徴である．例えば患者の健康状態 (年齢，体重，健康診断のデータなど) を説明変数 x，ある病気に罹患したか否かを目的変数 y として $f(x)$ を推定することで，どの説明変数が病気の罹患に強く影響を与えているかを分析したり，患者の健康状態から罹患リスクを予測することに用いる．

$$f(\vec{x}) = \sigma(\vec{w} \cdot \vec{x}) \tag{5}$$

σ は (6) に示すシグモイド関数である．

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{6}$$

これ以降，実際に観測された目的変数 (1or0 の正解ラベル) と式 (5) を用いて計算した目的変数を，それぞれ t, y として区別する．ロジスティック回帰では観測された x, t から式 (5) のパラメータ $\vec{w}$ を推定する．パラメータ推定には最尤推定という方法が用いられ，式 (7) に示す対数尤度関数と呼ばれる関数を最小化する $\vec{w}$ を求める．m はレコード数である．

$$E(\vec{w}) = -\sum_{i=1}^{m} t_n \log y_m + (1 - t_m) \log(1 - y_m) \tag{7}$$

イメージとしては，式 (5) で計算した予測値 y を実際に観測された正解 t にできるだけ近づける $\vec{w}$ を求める計算である．そして式 (7) を最小化する手法として広く用いられるのが Newton 法である．Newton 法は，$\vec{w}$ を変数とする関数 $f(\vec{w})$ の 1 階微分 $\vec{g}$ と 2 階微分 H(ヘシアン) を用いて式 (8) を計算し，これを反復することで，その関数 $f(\vec{w})$ を最小化する $\vec{w}$ を求める手法である．およそ 5 回ほどの反復で収束する．

$$\vec{w} = \vec{w} - H^{-1}\vec{g} \tag{8}$$

ロジスティック回帰では式 (7) に対して式 (8) を適用する．非常に有名な方法であるため導出は割愛するが，式 (7) の g, H はそれぞれ式 (9)(10) になる．

$$\vec{g} = X^\top(\vec{y} - \vec{t}) \tag{9}$$

$$H = X^\top \mathsf{diag}(\vec{y} \circ (1 - \vec{y}))X \tag{10}$$

2.4 モデル評価指標

AIC(赤池情報量基準) は統計モデルの良さを評価する際に最もよく用いられる指標である [5]．R など多くの統計ソフトでも採用されており，R でロジスティック回帰分析を行ったときも AIC が出力される．できるだけ説明変数の数が少なく，かつ y を高い精度で推定できるようなモデルが良いとされるが，具体的にどれくらいの数の説明変数をモデルに組み込めば良いかを判断するのは難しいため，AIC が最小になるモデルを選択するというのが 1 つの評価基準とされる．多くの場合は AIC が最小になるモデルを選択すれば良いモデルが得られる．AIC は式 (11) で表される．

$$\mathsf{AIC} = 2\ln L + 2k \tag{11}$$

L は最大尤度，k はパラメータの数を表す．ロジスティック回帰の場合 k は基本的に説明変数の数＋ 1(定数項の分) で計算する．

また AIC と共に用いられるモデル評価指標として，Residual Deviance(残差逸脱度) や Null Deviance(ヌル逸脱度) があり，それぞれ式 (12)(13) で計算される．

$$\mathsf{residualDeviance} = 2\ln L \tag{12}$$

$$\mathsf{nulllDeviance} = 2\ln L^* \tag{13}$$

L^* は式 (7) の尤度関数を計算する際に，y_n の代わりに正解ラベル t の平均値を入力することで求められる．

2.5 共役勾配法

式 (8) からも分かるように，Newton 法ではヘシアンの逆行列の計算が必要になる．逆行列の計算はコストが大きいため，回避する方法として共役勾配法が知られている．共役勾配法では $I(\vec{\beta})^{-1}$ を計算せずに，$I(\vec{\beta})$ と $U(\vec{\beta})$ から直接 $I(\vec{\beta})^{-1}U(\vec{\beta})$ を求める手法である．共役勾配法のアルゴリズムを **Algorithm** 1 に示す．ϵ は収束判定のハイパーパラメータであり，0^n は 0 が n 個並んだベクトルを表す．

Algorithm 1 共役勾配法

Require: n 次の正定値対称行列 H，n 次のベクトル $\vec{g}$
Ensure: $H\vec{d} = \vec{g}$ となる n 次のベクトル $\vec{d}$

```
d ← 0^n
r ← g
p ← g
while r^T r > ε do
    ρ ← r^T r
    α ← ρ / (p^T H p)
    d ← d + αp
    r ← r − αHp
    β ← r^T r / ρ
    p ← r + βp
end while
```

2.6 データの標準化

データの標準化とは，あるデータ $\vec{x}$ の分散が 1，平均が 0 になるように変形する処理である．データ $\vec{x}$ の平均を x_{ave}，分散を x_{var} とすると，標準化したデータ x_{std} は式 (14) のように計算される．

$$x_{std} = \frac{x - x_{ave}}{\sqrt{x_{var}}} \tag{14}$$

ロジスティック回帰モデルで単位の異なる複数の説明変数をモデルに組み込む場合，データをそのまま分析に用いてしまうと，パラメータの大きさが各説明変数の単位に依存してしまうため，「目的変数に対する影響力の強い説明変数はどれか」といった解釈が正しくできなくなってしまう．そのためパラメータ推定を行う前に，全ての説明変数に対して標準化を適用することが広く行われる

3. 提案手法

3.1 固定小数点数によるアルゴリズム設計

秘密計算では浮動小数点数の処理コストが大きいため，本稿では固定小数点数を用いてアルゴリズム設計を行う．固定小数点数の計算では乗算などによって小数点位置が変わってしまうため，適宜右シフトによって小数点位置を調節している．本稿の実装では，定数ラウンドの高速な右シフト [16] を採用している．アルゴリズムが煩雑になるのを防ぐため，以降に示すアルゴリズム上では右シフトを行う箇所を明記しない．

3.2 高速な秘密実数演算を用いた実装

ロジスティック回帰を秘密計算で行う際，難関となるのはネイピア数を底する指数関数や除算を含むシグモイド関数の計算である．従来手法ではシンプルな関数でシグモイド関数を近似したり [13]，参照表を用いた手法で計算されてきた [11][14]．本稿では [16] で提案されている高速な秘密実数演算群を用いることで，従来手法よりも更なる高速化と，平文と同程度の精度を目指す．

またデータの標準化では平方根や除算，AIC の計算では自然対数の計算が含まれるため，これらの処理にいても同様に [16] を用いる．

3.3 秘密計算によるデータの標準化

標準化のような前処理を高速かつ高精度に秘密計算で行うのは難しい．その難しさの要因となる秘密計算の性質は主に下記の 2 つである．

- 固定小数点数で処理するため，オーバーフローやアンダーフローが起きやすい
- データの値の正確な bit 数は把握できない (把握してはいけない)

このような秘密計算の性質に加え，標準化処理に入力されるデータは，値の小さいものや大きいもの，バラつきの小さいものや大きいものなど様々な性質を持つ．何故ならば，そのような様々な性質のものを同じ性質になるように整えて，その後の処理をやりやすくするのが標準化の目的だからである．

本稿では次の 4 つを全て満たす秘密計算の標準化処理方法を提案する．

- 入力データの値の大きさが分からない
- 固定小数点数なので表現できる値の範囲が狭い
- オーバーフローやアンダーフローを起こさない
- 精度を保ったまま高速に計算する

3.3.1 ベクトル msb 合わせ

秘密計算による標準化の難しさの最も根本的な要因は「入力データの値の大きさが分からない」ことである．

そこで本稿では，ある公開情報の msb 位置があったときに，入力データの msb 位置と公開情報の msb 位置の差を秘匿したまま，入力データの msb 位置を公開情報の msb 位置に合わせる手法を取る [17].

ここで，シェアベクトル $[\![\vec{a}]\!]$ の中で，絶対値が最大である値 $[\![a_{absmax}]\!]$ の msb 位置を「ベクトル msb」と定義し，ベクトル msb を秘匿したまま公開情報の msb 位置に合うようにシェアベクトル全体をシフトする処理を「ベクトル msb 合わせ」と定義する．またこの処理を式 (15) のように書く．入力がベクトルではなく行列の場合は，各々の列ベクトルに対して式 (15) を適用する．

$$[\![\vec{a'}]\!] \leftarrow \mathsf{adjustVmsb}([\![\vec{a}]\!], b) \tag{15}$$

ベクトル msb 合わせを適用することで「実際の値は分からないが，最大値のは msb 位置は分かる」という状態になり処理しやすい．

またベクトル msb あわせのようなベクトル全体を n 倍するスケーリング行っても，標準化の結果は変わらないことは明らかである．$\vec{x}$ 全体を n 倍した $\vec{x'}$ に式 (14) の標準化を適用する場合を考える．$\vec{x'}$ の平均 x'_{ave} は x_{ave} の n 倍，分散 x'_{var} は x_{var} の n^2 倍となる．分子の $\vec{x'} - x'_{ave}$ は $\vec{x} - x_{ave}$ の n 倍で，分母の $\sqrt{x'_{var}}$ も $\sqrt{x_{var}}$ の n 倍になっているため，標準化した結果 $\vec{x}_{std}$ は同じとなる．

3.3.2 ベクトル msb 合わせを適用した秘密標準化

ベクトル msb 合わせを適用した秘密標準化アルゴリズムを **Algorithm** 2 に示す．$[\![X^+]\!]$ などの「$+$」は adjustVmsb でのシフトによって元の値の n 倍になっていること，「$++$」は n^2 倍になっていること，「$-$」は $1/n$ 倍になっていることを表す．

処理の効率化のため，分散 x_{var} の平方根を計算してから除算ではなく，平方根の逆数を求める処理 sqrtinv を採用している [16]．平方根，逆数，平方根の逆数の計算にかかる処理時間は全て同程度であるため，平方根の逆数を直接求めるのが良い．

Algorithm 2 ベクトル msb 合わせを適用した秘密標準化

Require: 行列 $[\![X]\!]$，msb 位置 b
Ensure: 列ごとに標準化された行列 $[\![X_{std}]\!]$
$[\![X^+]\!] \leftarrow \mathsf{adjustVmsb}([\![X]\!], b)$
$[\![{}^t\vec{x}^+_{ave}]\!] \leftarrow \mathsf{average}([\![X^+]\!])$
$[\![{}^t\vec{x}^{++}_{var}]\!] \leftarrow \mathsf{var}([\![X^+]\!])$
$[\![U^+]\!] \leftarrow [\![X^+]\!] - [\![{}^t\vec{x}^+_{ave}]\!]$
$[\![{}^t\vec{w}^-]\!] \leftarrow \mathsf{sqrtinv}([\![{}^t\vec{x}^{++}_{var}]\!])$
$[\![X_{std}]\!] \leftarrow [\![U^+]\!] \times [\![{}^t\vec{w}^-]\!]$

3.4 秘密計算ロジスティック回帰

ロジスティック回帰のパラメータ推定にはいくつか方法があるが，少ない反復回数で良好なパラメータが得られ，かつ非常に重い処理である逆行列の計算を回避できる Newton 共役勾配法を用いる．

式 (10) のヘシアン H の計算に含まれる $\mathsf{diag}(\vec{y} \circ (1^m - \vec{y}))$ は直接計算すると $m \times m$ の行列になってしまう．ロジスティック回帰では通常 m は n より遥かに大きな数である．説明変数の数 n は数個〜数十個程度が一般的であるのに対し，分析に用いるデータ数が 10 万件であれば，この行列は 10 万× 10 万となってしまうため，処理効率やメモリ効率の観点から計算を避けたい．$\mathsf{diag}(\vec{y} \circ (1 - \vec{y}))$ が対角行列であることを利用し，直接この行列は計算せずにヘシアン H を計算する方法を **Algorithm** 3 に示す．

Algorithm 3 の方法を用いることで，$n \times m$ 行列と $m \times m$ 行列の積の計算が，長さ $n \times m$ のベクトル同士の積で済む．この方法はロジスティック回帰でのヘシアンの計算に限らず，巨大な対角行列との積を計算する場合であれば同様に計算できる．以降，**Algorithm** 3 を calcHesse で表す．

Algorithm 3 ロジスティック回帰のヘシアンの秘密計算

Require: $m \times n$ 説明変数行列 $[\![X]\!]$，m 次予測値ベクトル $[\![\vec{y}]\!]$
Ensure: $n \times n$ のヘシアン $[\![H]\!]$
$[\![\vec{r}]\!] \leftarrow [\![\vec{y}]\!] \times ([\![1]\!] - [\![\vec{y}]\!])$
$[\![\vec{x}]\!] \leftarrow \mathsf{flatten}([\![X]\!])$
$[\![\vec{x}_t]\!] \leftarrow \mathsf{transpose}([\![\vec{x}]\!])$
$[\![\vec{r}_{copy}]\!] \leftarrow \mathsf{copyVec}([\![\vec{r}]\!], n)$
$[\![\vec{h}]\!] \leftarrow [\![\vec{r}_{copy}]\!] \times [\![\vec{x}_t]\!]$
$[\![\vec{h}]\!] \leftarrow \mathsf{matmul}([\![\vec{h}]\!], [\![\vec{x}_t]\!], m)$
$[\![H]\!] \leftarrow \mathsf{matrixify}([\![\vec{h}]\!], n)$

秘密計算ロジスティック回帰のアルゴリズムを **Algorithm** 4 に示す．**Algorithm** 4 の CG は共役勾配法の計算を表し，別途 **Algorithm** 5 にアルゴリズムを記載した．

Algorithm 4 秘密計算ロジスティック回帰

Require: $m \times n$ の説明変数行列 $[\![X]\!]$，長さ m の正解ラベルベクトル $[\![\vec{t}]\!]$，学習回数 α
Ensure: 長さ n のパラメータベクトル $[\![\vec{w}]\!]$
$[\![\vec{w}]\!] \leftarrow \mathsf{fill}([\![0]\!], n)$
for $i = 0, 1, \cdots, \alpha - 1$ **do**
  $[\![\vec{y}]\!] \leftarrow \mathsf{Sigmoid}(\mathsf{hpsum}([\![X]\!], [\![{}^t\vec{w}]\!]))$
  $[\![\vec{g}]\!] \leftarrow [\![X]\!]^\top([\![\vec{y}]\!] - [\![\vec{t}]\!])$
  $[\![H]\!] \leftarrow \mathsf{calcHesse}([\![X]\!], [\![\vec{y}]\!])$
  $[\![\vec{d}]\!] \leftarrow \mathsf{CG}([\![H]\!], [\![\vec{g}]\!])$
  $[\![\vec{w}]\!] \leftarrow [\![\vec{w}]\!] - [\![\vec{d}]\!]$

Algorithm 5 秘密計算共役勾配法

Require: n 次の正定値対称行列 $[\![H]\!]$，n 次のベクトル $[\![\vec{g}]\!]$
Ensure: $H\vec{d} = \vec{g}$ となる n 次のベクトル $[\![\vec{d}]\!]$
$[\![\vec{d}]\!] \leftarrow \mathsf{fill}([\![0]\!], n)$
$[\![\vec{r}]\!] \leftarrow [\![\vec{g}]\!]$
$[\![\vec{p}]\!] \leftarrow [\![\vec{g}]\!]$
$[\![\rho]\!] \leftarrow [\![\vec{r}]\!] \cdot [\![\vec{r}]\!]$
while $[\![\rho]\!] > \epsilon$ **do**
  $[\![\alpha]\!] \leftarrow \frac{\vec{r}^\top \vec{p}}{\vec{p}^\top H \vec{p}}$
  $[\![\alpha]\!] \leftarrow ([\![\vec{r}]\!] \cdot [\![\vec{p}]\!]) \times \mathsf{reciprocal}([\![\vec{p}^\top]\!][\![H]\!][\![\vec{p}]\!])$
  $[\![\vec{d}]\!] \leftarrow [\![\vec{d}]\!] + [\![\alpha]\!] \times [\![\vec{p}]\!]$
  $[\![\vec{r}]\!] \leftarrow [\![\vec{r}]\!] - [\![\alpha]\!][\![H]\!][\![\vec{p}]\!]$
  $[\![\beta]\!] \leftarrow ([\![\vec{r}]\!] \cdot [\![\vec{r}]\!]) \times \mathsf{reciprocal}([\![\rho]\!])$
  $[\![\vec{p}]\!] \leftarrow [\![\vec{r}]\!] + [\![\beta]\!] \times [\![\vec{p}]\!]$
  $[\![\rho]\!] \leftarrow [\![\vec{r}]\!] \cdot [\![\vec{r}]\!]$
end while

3.4.1 モデル評価の秘密計算

式 (7) の計算を秘密計算で行う．この処理の中で比較的コストが大きいのは対数の計算である．式の通りに実装すると $2m$ 回の log を計算することになるが，t_m が 0 か 1 であることを利用すると，log の計算は m 回で済む $t_m = 0$ の場合 $t_m \log y_m$ は 0 になるため $\log y_m$ を計算する必要はなく，同様に $t_m = 1$ の場合 $(1 - t_m)\log(1 - y_m)$ は 0 に

なるため $\log(1-y_m)$ を計算する必要はない．したがって，$t_m=0$ の場合は $y'_m=1-y_m$，$t_m=1$ の場合は $y'_m=y_m$ のような y' を並べた長さ m のベクトル $\vec{y'}$ を計算できれば，あとは log と総和を求めるだけである．

Algorithm 6 秘密計算での対数尤度の計算

Require: 長さ m のベクトル $[\![\vec{t}]\!], [\![\vec{y}]\!]$
Ensure: 対数尤度 $[\![lh]\!]$
$[\![\vec{y'}_1]\!] \leftarrow [\![\vec{t}]\!] \times [\![\vec{y}]\!]$
$[\![\vec{y'}_2]\!] \leftarrow ([\![1]\!] - [\![\vec{t}]\!]) \times ([\![1]\!] - [\![\vec{y}]\!])$
$[\![\vec{y'}]\!] \leftarrow [\![\vec{y'}_1]\!] + [\![\vec{y'}_2]\!]$
$[\![\vec{l}]\!] \leftarrow \mathsf{log}([\![\vec{y'}]\!])$
$[\![lh]\!] \leftarrow \mathsf{sum}([\![\vec{l}]\!])$

対数尤度の計算方法を **Algorithm** 6 のようにしても加減乗算の回数は式 (7) を素直に実装した場合と比較して同じであり，単純にデータ数 m 個ぶんの log の計算コストが抑えられる．

4. 実験

4.1 設定

4.1.1 実験環境

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

4.1.2 データセット

本稿の実験では，実データとして以下の 3 つを用いた

- Pima Indians Diabetes Database(diabetes)[1]
- Wine Quality Data Set(wine quality)[4]
- birthwt[2]

diabetes は説明変数が 8 個でレコード数が 768 件，wine quality は説明変数が 12 個でレコード数が 6497 件，birthwt は説明変数が 6 個でレコード数が 189 件のデータセットである．birthwt データは，数字の大小に意味がないカテゴリ値の説明変数は除外して 6 個とした．

4.2 秘密標準化

4.2.1 処理時間の評価

ダミーデータを用いて，提案手法の処理時間を測定した結果を表 2 に示す．

表 2 秘密標準化の処理時間 [s]

	10 属性	100 属性
1000 件	0.433	4.564
10 万件	1.195	10.928

10 万× 100 という大きなデータセットも約 11 秒で処理でき，1000 × 10 程度のデータセットであれば 0.4 秒程度でき非常に高速であることを示した．

4.2.2 精度の評価

標準化処理の最初の msb 位置合わせで何 bit にあわせるかを変え，秘密計算で標準化した結果と平文 (浮動小数点数) で標準化した場合での差を評価した結果を表 3 に示す．

データ X を平文で標準化したものを X_{std}，提案手法で標準化したものを $[\![X_{std}]\!]$ としたとき，2 つの行列の全要素に対して $|X_{std}-[\![X_{std}]\!]|$ を計算し，その中で最も値の大きいものを最大誤差とした．

表 3 ベクトル msb 合わせを用いた秘密標準化の最大誤差

	17	20	24
dieabetes	1.08e-3	1.29e-4	8.02e-6
wine quality	5.38e-3	5.78e-4	1.06e-5
birthwt	1.00e-3	1.20e-4	7.65e-6

データセットによって多少の差はあるものの概ね誤差の大きさは同程度となり，また bit 数と誤差は比例の関係となることが分かった．24bit の設定では，平文で処理したものと比較しても小数点第 5 位まで一致するような結果となり，高い精度で標準化できている．

最初のベクトル msb 合わせで p[bit] となるようにした場合，標準化処理はなるべく有効桁数 p[bit] を維持したまま行うように，適宜小数点位置の調節を行っている．しかし計算過程でまるめ誤差が積み重なっていくため，標準化結果の有効桁数はそれ以下となる．表 3 の結果では，標準化処理を行うことで最初の p[bit] から 10 進数で下から 2〜3 桁目までまるめ誤差の影響がでたが，4 桁目までは出なかった．

4.3 実数演算を用いた秘密計算ロジスティック回帰

4.3.1 処理時間

秘密標準化の実験で用いたものと同じダミーデータで，秘密ロジスティック回帰の処理時間を測定した結果を表 4 に示す．この実験では，Newton 法の反復回数を 5 回で固定し，共役勾配法の中の反復処理も収束判定を行わず，5 回で固定して実施した．

表 4 秘密計算ロジスティック回帰の処理時間 [s]

	10 属性	100 属性
1000 件	2.117	2.433
100 万件	13.988	70.924

1000 件× 10 属性の場合は約 2 秒となっており非常に高速である．100 万件× 100 属性という非常に大きなデータセットに対しても 1 分強で処理できているため，十分に実用的である．

4.3.2 実データでの実験

また提案手法に対して 3 つの実データを入力し，それぞれ処理時間，提案手法で得られたパラメータと平文で計算したパラメータの相関係数，提案手法で得られたパラメータで計算した予測結果の精度を評価したものを表 5 に示す．比較対象の平文は scikit-learn[3] である．

表 5 実データでの処理時間，相関係数，予測精度

	diabetes	winequality	birthwt
処理時間 [s]	2.133	9.270	2.662
相関係数	0.99999	0.99999	0.99999
予測精度 [%]	78.3	99.5	70.4

相関係数は 2 つのデータが近いほど 1 に近い値をとる．提案手法で得られたパラメータは平文で計算したものとほぼ一致していた．また予測精度についても，どのデータでも平文で計算したものと一致することが確認でき，実データに対しても高い実用性を示した．

4.3.3 秘密対数尤度計算の評価

3 つの実データに対して，**Algorithm** 6 で提案した秘密計算での対数尤度の計算アルゴリズムを用いて，AIC, residualDeviance.nulllDeviance を計算し，平文で計算したものと比較したものを表 6 に示す．対数尤度の処理時間については [15] で示しており，1000 万件を 4.6 秒を処理時でき非常に高速である．

表 6 AIC, Residual Deviance, Null Deviance

		diabetes	wine quality	birthwt
AIC	平文	741.45	450.23	222.77
	提案手法	741.45	431.19	222.77
RD	平文	723.45	424.23	208.77
	提案手法	723.45	405.49	208.77
ND	平文	993.48	7250.98	234.67
	提案手法	993.48	7250.98	234.67

diabetes と birthwt では完全に一致したものの，wine quality では AIC, residualDeviance で誤差が生じてしまった．原因を探ったところ log の入力が非常に 0 に近い値 (10^{-11} 程度) になるレコードが 1 つあり，大きな誤差になっていた．自然対数関数は 0 に近づくほど傾きが急激に大きくなる関数であるため，このような思わぬ誤差が生じてしまった．そのようなレコードは，実際の値が 1 であるのに対して予測値がほぼ 0 になっていたり，その逆の場合に生じてしまう．提案手法でも 10^{-8} 程度までなら問題ないが，実データではそれよりも小さな値が生じることもあると分かった．

単純な対策は入力の精度を高くすることだが，その前の予測値 $\vec{y}$ の計算で，シグモイド関数の exp を計算する際にオーバーフローを防ぐために入力の bit 数を落としていることが精度低下の原因につながっていた．この問題に対しては，シグモイド関数の入力が正の値になるような線形変換を施してからシグモイド関数に入力することで，現状よりも高い精度が確保できるようになると考えられる．

5. おわりに

本稿の成果は以下の 3 つである．

- 秘密実数演算を用いて高速・高精度を両立した秘密計算ロジスティック回帰
- AIC などモデル評価指標の計算を秘密計算上で実現
- 高精度なデータの標準化処理を秘密計算上で実現

本稿で提案した秘密計算ロジスティック回帰では，全ての実データに対して数秒で処理が終わり，結果も平文で計算したものとほぼ一致した．また AIC などの重要なモデル評価指標も合わせて計算できるようになったため，秘密計算ロジスティック回帰が更に実用的なものとなった．そしてベクトル msb 合わせを用いた標準化処理は高い処理性能と精度，そしてスケーラビリティを示した．提案手法の標準化処理は，ロジスティック回帰に限らず様々な統計分析や機械学習において役立つだろう．

5.1 今後の展望

様々な条件での実験を通して，対数尤度の計算やシグモイド関数の計算にはまだ改善の余地があることが分かったため，引き続き改良を行うと同時に，AIC 以外のモデル評価指標 (標準誤差など) も秘密計算で行えるように検討していく．

参考文献

[1] diabetes. https://www.kaggle.com/uciml/pima-indians-diabetes-database.

[2] R. https://cran.r-project.org/.

[3] scikit-learn. https://scikit-learn.org/stable/.

[4] Wine quality data set. https://archive.ics.uci.edu/ml/datasets/wine+quality.

[5] Hirotogu Akaike. Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike*, pp. 199–213. Springer, 1998.

[6] 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.

[7] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 215–242, 1958.

[8] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *IEEE Symposium on Security and Privacy, SP 2017*, pp. 19–38, 2017.

[9] Michael O. Rabin. How to exchange secrets by oblivious transfer. In *Technical Report TR-81*, 1981.

[10] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pp. 162–167, 1986.

[11] 三品気吹, 濱田浩気, 五十嵐大. 秘密計算によるロジスティック回帰は本当に使えるか？ In *SCIS*, 2019.
[12] 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.
[13] 三品気吹, 五十嵐大, 濱田浩気, 菊池亮. 高精度かつ高効率な秘密ロジスティック回帰の設計と実装. In *CSS*, 2018.
[14] 濱田浩気, 五十嵐大, 三品気吹, 菊池亮. 秘密計算上の一括近似とそれを使った正確度の高いロジスティック回帰. In *CSS*, 2019.
[15] 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.
[16] 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.
[17] 五十嵐大. 高効率なシフト量秘匿シフトプロトコルの構成による, 速度と精度を両立する秘密計算上の浮動小数点数の実現. In *CSS*, 2020.
[18] 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.
[19] 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.