Computer Security Symposium 2020
26 - 29 October 2020

秘密計算による正則化線形回帰分析

深見 匠[1,a)]　三品 気吹[1]　五十嵐 大[1]

概要：多種多様なデータから知見を取り出す手法として機械学習が普及してきており, 機密性の高いデータを持ち寄って解析するニーズも生じつつある. そこで機械学習で用いられるデータの秘匿性を確保するため, 暗号化したまま学習を行う秘密計算を用いた機械学習が注目されている. 機械学習手法は様々存在するが, 高次元データにおいて, 変数選択と正則化をしつつ, 回帰による予測を提供する Lasso 回帰, ElasticNet は様々な分野で用いられている. しかし, 当手法を秘密計算上で実装した前例はなく, 本研究ではその実装評価を行った.
Lasso 回帰, ElasticNet のパラメタ更新では平文演算に比べ処理コストが高い秘密計算演算が多用されるため, 性能の確保が課題となった. 本研究では, 課題解決のため計算の最適化と Soft-Thresholding 関数を数 Bit で行えるように軽量化し, 秘密計算上での Lasso 回帰, ElasticNet の実装可能性と現実的な処理コストでの演算可能性を確認した.

キーワード：秘密計算, 秘密分散 , 正則化, 線形回帰

Regularized Linear Regression Analysis in secure computation

Takumi Fukami[1,a)]　Ibuki Mishina[1]　Dai Ikarashi[1]

Abstract: Machine learning is becoming more and more popular as a method to extract findings from a wide variety of data, There is also a need to bring in highly sensitive data for analysis. Therefore, to ensure the confidentiality of the data used in machine learning, cryptographic machine learning with cryptography has been attracting attention. Although various machine learning methods exist, Lasso regression and ElasticNet are used in various fields to provide predictions by regression while selecting and normalizing variables in high-dimensional data. However, there has been no previous secret implementation of those methods, and we evaluate the implementation in this study. Lasso regression and ElasticNet's parameter updates often use secret operations that are more expensive than plaintext operations. In this study, we optimize the computation and reduce the weight of Soft-Thresholding functions to a few bits, and confirm the feasibility of implementing Lasso regression and ElasticNet on the covert computation, as well as the computational feasibility with realistic processing costs.

Keywords: Secure Compilation, Secret Sharing, Regularization, Linear Regression

1. はじめに

秘密計算は, 情報を暗号化したまま計算できる技術である. 秘密計算を用いることにより, 複数の機関が所有する機密データを開示することなく収集, 分析することができる. 分析手法として, 特にデータの機械学習が注目されており, 秘密計算の分野において研究が進んでいる. 機械学習手法は様々存在しているが, 医療データなどの高次元データにおいて, 変数選択と正則化をしつつ回帰予測をする Lasso 回帰, ElasticNet は多くの分野で用いられている. しかし, 当手法が秘密計算上で行われた前例はない. 本研究では, Lasso 回帰 [9], ElasticNet[11] を平文と同程度の精度かつ実用的な時間で実現することを目的とする.

[1] NTT セキュアプラットフォーム研究所
NTT Secure Platform Laboratory
[a)] takumi.fukami.as@hco.ntt.co.jp

1.1 本稿の成果

本稿では, 先述した課題に取り組み下記の成果を得た.

(1) Soft-Thresholding 関数を軽量化と秘密計算用に計算式の最適化を行い, 処理の高速化をした

(2) 秘密計算での正則化を用いた線形解析を 20 属性×1000 万属性において, 平文と同じ精度で実用可能なことを示した.

1.2 正則化を用いた線形回帰

線形回帰は回帰手法の一種であり, ある説明変数 X と目的変数 Y が与えられたとき, その関係式 $Y = f(X)$ を推定する機械学習の手法である. 線形回帰の目的は, 説明変数と目的関数の関係の推定, 未知の説明変数に対応する目的変数の推定である. 例えば, 目的変数をある病気への罹患率とし, 説明変数を人の健康情報とすると, 健康状態と罹患率の関係性の推定することができる.

線形回帰の最も簡単なモデルは $Y = a + bX$ で表される. この式の a,b をパラメータと呼び, データを用いてパラメータを推定する"学習"を行うことで, モデルを推定する. 簡単な回帰分析では, 最小二乗法を用いてパラメータを学習するが, この手法では訓練データのモデルに過剰にフィットしてしまい, テストデータに用いたときに汎用性が低下する過学習という問題が発生する. 過学習が発生すると, 未知の説明変数 X に対応する目的変数 Y を正しく推定できない. この問題を解決し線形回帰を行う手法が Lasso 回帰, ElasticNet である. これらの手法は, 損失関数に重み（正則化項）を足し, 過学習を防ぎ, さらに不要と判断した説明変数を無視して学習ができる.

1.3 関連研究

秘密計算で線形回帰を行った既存研究として, R Hall ら [6] や Y Aono ら [3] が準同型暗号を用いたもの, A Gascón ら [2] が garbled circuit[10] を用いたものがある. これらの手法は, 共役勾配降下法を用いて, 線形回帰モデルを安全に推定できる. また平文で Lasso 回帰,ElasticNet を行える scikit-learn[1] などもある.

2. 準備

2.1 記法

ベクトルを $\vec{a} := (a_1, a_2, \cdots, a_n)$ と書き, 行列を $A := (\vec{a_1}, \vec{a_2}, ..., \vec{a_m})$ と書く. 行列やベクトルの内積は $(\cdot)$ で表し, 要素ごとの積は $(\times)$ で表し, 演算子がない場合はスカラー倍を表す. $[\cdot]$ は述語を表し, 例えば $[a = b]$ は $a = b$ ならば 1, そうでないなら 0 を表す.

2.2 線形回帰分析

説明変数が複数ある線形回帰分析を考える, 線形回帰モデル f は, パラメータを $\vec{w} \in \mathbb{R}^m$, 説明変数を $\vec{x} \in \mathbb{R}^{m+1}$ とすると下記の式である.

$$f(\vec{x}) = w_0 + w_1 x_1 + w_2 x_2 + ... + w_m x_m \tag{1}$$

説明変数が複数ある場合, 説明変数の大きさによらず目的変数との関係を推定するため, 説明変数を標準化してからモデルを推定するのが一般的である.

2.2.1 最小二乗法

回帰分析でパラメータを学習する際, 良く用いられるアルゴリズムの一つであり, 残差の二乗和を最小とするパラメータを推定する. 最小二乗法では, パラメータ更新で用いる損失関数を式 (2) のように定義する.

$$L = \sum_{i=1}^{n} (y_i - f(\vec{x_i}))^2 = (\vec{y} - X\vec{w})^T (\vec{y} - X\vec{w}) \tag{2}$$

このとき, $\vec{y} = (y_1, y_2, ..., y_n)$ は目的変数を表す. 損失関数を最小とするパラメータは式 (3) のようになり, 反復せずに一度計算すればよい.

$$\vec{w} = (X^T X)^{-1} X^T \vec{y} \tag{3}$$

最小二乗法を用いると一度の計算でパラメータ推定が行えるが, 前述したように過学習が発生してしまう.

2.2.2 lasso 回帰と ElasticNet

線形回帰分析を過学習を防ぐため, Lasso 回帰, ElasticNet では L1 ノルム $||\vec{w}||_1$ と L2 ノルム $||\vec{w}||_2^2$ と呼ばれる正則化項を損失関数に導入する. λ, α をハイパーパラメータとすると, 損失関数は式 (4) のようになる.

$$L = \frac{1}{2n}(\vec{y} - X\vec{w})^T (\vec{y} - X\vec{w}) + \lambda(\alpha||\vec{w}||_1 + (1-\alpha)\frac{1}{2}||\vec{w}||_2^2) \tag{4}$$

上記の式により正則化を考慮した回帰分析は ElasticNet といい, $\alpha = 1$ のときは Lasso 回帰という.

式 (4) が最小となるパラメータは, 下記の式を計算することで求められる.

$$S(a,b) = \begin{cases} a - b & (b < a) \\ 0 & (-b < a < b) \\ a + b & (a < -b) \end{cases} \tag{5}$$

$$w_0 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{d} x_{ij} w_j) \tag{6}$$

$$w_k = \frac{S(\sum_{i=1}^{n} (y_i - w_0 - \sum_{j=1(j \neq k)}^{d} x_{ij} w_j) x_{ik}, n\lambda\alpha)}{\sum_{i=1}^{n} x_{ik}^2 (1 + \lambda(1-\alpha))} \tag{7}$$

式 (6) と式 (7) は更新式であり, 更新式を反復することで適切なパラメータを推定する. Lasso 回帰と ElasticNet では, Soft-Threshoding 関数と呼ばれる式（5）があることにより, 不要と判断したパラメータを 0 にすることで, 変数選択を行うことができる.

2.3 秘密計算

2.3.1 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る．その中でも秘密情報を「シェア」という分散情報に変換し計算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 [4], [13]. 秘密分散はさまざまな方法で構成できることが知られており, Shamir の秘密分散 [8] や複製秘密分散 [5], [7] などが知られている．本稿では秘密分散の中でも (n,k) 閾値秘密分散法を用いる．これは n 個のシェアを生成し, k 個以上情報を集めると秘密情報が復元できる秘密分散法である．

2.3.2 プログラマブルな秘密計算ライブラリ MEVAL

MEVAL は我々が開発する秘密分散ベースの秘密計算ライブラリ [14] で, 加算乗算などの基本的な演算から徐算, 指数といった実数演算など 100 以上ある様々な演算を組み合わせて自由にプログラムできる．本稿では, MEVAL を用いて Lasso 回帰と ElasticNet を計算する．MEVAL は演算に応じて複数の秘密分散を用いており, Lasso 回帰と ElasticNet では, $GF(2^{127}-1)$ 上の Shamir 秘密分散と $\mathbb{Z}_2$ 上の複製秘密分散を用い, それぞれのシェアを $[\![a]\!]$, $\{a\}$ と書く．ベクトルは $[\![\vec{a}]\!](=[\![a_1]\!],[\![a_2]\!],...,[\![a_m]\!])$ のように書く．シェアの加算や定数倍は通信無しで計算可能であり, 例えば $a[\![\vec{b}]\!]+c=[\![a\vec{b}+c]\!]=(ab_1+c,ab_2+c,...,ab_m+c)$ のように書く．ベクトルの要素ごとに同じ演算を行う場合はこのように要素 1 つに演算を行う場合と同様に書く．

- equality($[\![\vec{a}]\!],[\![\vec{b}]\!]$)
  - 入力：$[\![\vec{a}]\!],[\![\vec{b}]\!]$
  - 出力：$\{[\vec{a}=\vec{b}]\}:=(\{[a_1=b_1]\},...,\{[a_m=b_m]\})$
- greater_than($[\![\vec{a}]\!],[\![t]\!]$)
  - 入力：$[\![\vec{a}]\!],[\![t]\!]$
  - 出力：$\{[\vec{a}>t]\}:=(\{[a_1>t]\},...,\{[a_m>t]\})$
- ifgate($[\![\vec{a}]\!],[\![t]\!],[\![f]\!]$)
  - 入力：分岐条件 $[\![\vec{a}]\!]$ と $[\![t]\!],[\![f]\!]$)
  - 出力：分岐条件が真の場合 $[\![t]\!]$, 偽の場合 $[\![f]\!]$
- to_P($\{\vec{a}\}$)
  - 入力：$\{\vec{a}\}:=(\{a_1\},\{a_2\},...,\{a_m\})$
  - 出力：$[\![\vec{a}]\!]:=([\![a_1]\!],[\![a_2]\!],...,[\![a_m]\!])$
- sum($[\![\vec{a}]\!]$)
  - 入力：$[\![\vec{a}]\!]:=([\![a_1]\!],[\![a_2]\!],...,[\![a_m]\!])$
  - 出力：$[\![b]\!]:=[\![a_1+a_2+...+a_m]\!]$
- rshift($[\![\vec{a}]\!],t$)
  - 入力：$[\![\vec{a}]\!]:=([\![a_1]\!],[\![a_2]\!],...,[\![a_m]\!]),t$
  - 出力：$[\![\vec{b}]\!]$

    $[\![\vec{a}]\!]$ を t [bit] 算術右シフトした $[\![\vec{b}]\!]$ を出力する．算術右シフトは左側を 0 ではなく符号ビットでパディングするシフトであり，論理右シフト rshift [12] を用いて以下のように $\mathrm{rshift}([\![A\times 2^n]\!],n-m)=[\![A\times 2^m]\!]$ を構成する．

$$[\![A'\times 2^n]\!]=[\![A\times 2^n]\!]+a\times 2^n(a\geq|A|) \quad (8)$$

$$[\![A'\times 2^m]\!]=rshift([\![A'\times 2^n]\!],n-m) \quad (9)$$

$$[\![A\times 2^m]\!]=[\![A'\times 2^m]\!]-a\times 2^m \quad (10)$$

3. 提案手法

秘密計算で Lasso 回帰, ElasticNet を計算する．秘密計算では浮動小数点数の処理コストが大きいため, 本稿では固定少数点数を用い, 少数を含む値は全て整数にエンコードして扱う．例えば, 精度が a[bit] の場合は秘密情報 $[\![b]\!]$ は $[\![b\times 2^a]\!]$ として分散する．固定小数点数で乗算を行う場合, $[\![x\times y\times 2^{2a}]\!]=[\![x\times 2^a]\!]\times[\![y\times 2^a]\!]$ となり, オーバーフローと精度が変わる問題が発生する．本稿では, 乗算を行う毎に rshift[12] を行うことで, これらの問題を回避する．扱う値の精度は全て 30[bit] になるように調整しつつ計算を行う．

3.1 Secure ElasticNet

ElasticNet ではパラメータ推定を行うために, 更新式を反復して計算する必要がある．そのため, 秘密計算での処理が大きい演算が更新式に含まれていると処理コストが増えてしまう．そこで更新式の処理コストを削減することを考える．

3.1.1 更新式の変形

更新式に含まれる行列計算はデータ数と属性数が増えると処理コストが増えてしまう．そこで, パラメータ更新式 (7) を式 (11) から (14) のように変形する．式 (6) の変形は行わない．

$$\vec{t_1}=\vec{y}-w_0-\sum_{j=2}^{d}x_{ij}w_j \quad (11)$$

$$w_1=\frac{S(\sum_{i=1}^{n}t_{1i}x_{i1},n\lambda\alpha)}{\sum_{i=1}^{n}x_{i1}^2(1+\lambda(1-\alpha))} \quad (12)$$

$$\vec{t_k}=\vec{t_{k-1}}+w_k\vec{x}_k-w_{k-1}\vec{x}_{k-1} \quad (13)$$

$$w_k=\frac{S(\sum_{i=1}^{n}t_{ki}x_{ik},n\lambda\alpha)}{\sum_{i=1}^{n}x_{ik}^2(1+\lambda(1-\alpha))} \quad (14)$$

ただし, $\vec{t_l}:=(t_{l1},t_{l2},...,t_{ld})$, $\vec{x_l}:=(x_{1l},x_{2l},...,t_{dl})$ とする．Secure ElasticNet のパラメータ更新では, 式 (12), (14) を用いることとする．これによりパラメータ数を m とすると, 行列計算を行う回数を更新 1 回につき $m+1$ 回から 2 回に減らすことができる．

3.1.2 Soft-Threshoding 関数

Soft-Threshoding 関数は $a-b>0$ または $a+b<0$ かどうかを計算することで出力する値を決定する．秘密計算では Bit 数が大きい値の比較ほど処理が重くなる．ElasticNet において a は標準化された値を元にした計算結果であり, b は事前に決められるハイパーパラメータを元にした値であるため, a と b の Bit 数を事前に計算可能である．秘密計算で $a-b>0$ または $a+b<0$ かどうかを計算するには, $a-b$

と $a+b$ の精度が 30[bit] より小さくても, 所望の精度を満たすことができるので, 右シフトし, 処理するデータのビット数を減らして高速化する. 軽量化した Soft-Threshoding 関数を Algorithm 1 に示す.

Algorithm 1 秘密計算での Soft-Threshoding 関数

Require: $[\![a]\!], [\![b]\!]$, BitNum
Ensure: $[\![ST]\!] = S([\![a]\!], [\![b]\!])$
  $[\![c]\!] \leftarrow$ rshift($[\![$a + b$]\!]$, BitNum)
  $[\![d]\!] \leftarrow$ rshift($[\![$a − b$]\!]$, BitNum)
  $\{cond1\} \leftarrow$ greater_than($[\![$c$]\!]$, 0)
  $\{cond2\} \leftarrow$ greater_than($[\![$d$]\!]$, 0)
  $[\![cond1]\!] \leftarrow$ to_P({cond1})
  $[\![cond2]\!] \leftarrow$ to_P({cond2})
  $[\![ST]\!] \leftarrow$ ifgate($[\![$cond1$]\!]$, $[\![0]\!]$, $[\![$a + b$]\!]$)
  $[\![ST]\!] \leftarrow$ ifgate($[\![$cond2$]\!]$, $[\![$a − b$]\!]$, $[\![$ST$]\!]$)

Algorithm 1 の入力である BitNum は $[\![a+b]\!]$ と $[\![a-b]\!]$ の bit 数の小さい方の値である.

3.1.3 Secure ElasticNet プロトコル

Secure ElasticNet では, Algorithm 2, 3, 4 に示すバイアス更新とパラメータ更新を反復することによりパラメータを推定する. n, m はそれぞれデータ数と属性数に対応し, また, 秘密計算で扱いやすいようにベクトルは属性ごとにもつ. ElasticNet では, 入力値を標準化してから入力するため, 式 (12), (14) において $\sum_{i=1}^{n} x_{ik}^2 = n-1$ として考えることができる.

Algorithm 2 秘密計算でのバイアス更新

Require: $[\![X]\!](= [\![\vec{x_1}]\!], [\![\vec{x_2}]\!], ..., [\![\vec{x_m}]\!])$
  $[\![\vec{w}]\!](= [\![w_1]\!], ..., [\![w_m]\!])$
  $[\![\vec{y}]\!](= [\![y_1]\!], [\![y_2]\!], ..., [\![y_n]\!])$
Ensure: $[\![w_0]\!]$
  $[\![\vec{c}]\!] \leftarrow ([\![X]\!] \cdot [\![\vec{w}]\!])$
  $[\![\vec{d}]\!] \leftarrow [\![\vec{y}]\!] - [\![\vec{c}]\!]$
  $[\![g]\!] \leftarrow$ sum($[\![\vec{\text{d}}]\!]$)
  $[\![w_0]\!] \leftarrow$ sum($[\![$g$]\!]$/n)

Algorithm 3 秘密計算での ElasticNet パラメータ更新 (式変形無)

Require: $[\![X]\!](= [\![\vec{x_1}]\!], [\![\vec{x_2}]\!], ..., [\![\vec{x_m}]\!])$
  $[\![\vec{w}]\!](= [\![w_1]\!], ..., [\![w_m]\!]), [\![w_0]\!]$
  $[\![\vec{y}]\!](= [\![y_1]\!], [\![y_2]\!], ..., [\![y_n]\!])$
  ハイパーパラメータ $[\![\lambda]\!], [\![\alpha]\!]$
Ensure: $[\![\vec{w}]\!]$
  for $j = 1$ to $j \leq m$ **do**
    $[\![\vec{tmp}]\!] \leftarrow ([\![w_1]\!], [\![w_2]\!], ..., [\![w_3]\!])$
    $[\![tmp_j]\!] \leftarrow [\![0]\!]$
    $[\![\vec{c}]\!] \leftarrow ([\![X]\!] \cdot [\![\vec{tmp}]\!])$
    $[\![\vec{t}]\!] \leftarrow [\![\vec{y}]\!] - [\![\vec{c}]\!] - [\![w_0]\!]$
    $[\![\vec{e}]\!] \leftarrow [\![\vec{t}]\!] \times [\![\vec{x_j}]\!]$
    $[\![g]\!] \leftarrow$ sum($[\![\vec{\text{e}}]\!]$)
    $[\![ST]\!] \leftarrow S([\![g]\!], [\![n \times \lambda \times \alpha]\!])$
    $[\![w_1]\!] \leftarrow [\![ST]\!]/[\![(n-1) \times \lambda \times (1-\alpha))]\!]$
  end for

Algorithm 4 秘密計算での ElasticNet パラメータ更新 (式変形有)

Require: $[\![X]\!](= [\![\vec{x_1}]\!], [\![\vec{x_2}]\!], ..., [\![\vec{x_m}]\!])$
  $[\![\vec{w}]\!](= [\![w_1]\!], ..., [\![w_m]\!]), [\![w_0]\!]$
  $[\![\vec{y}]\!](= [\![y_1]\!], [\![y_2]\!], ..., [\![y_n]\!])$
  ハイパーパラメータ $[\![\lambda]\!], [\![\alpha]\!]$
Ensure: $[\![\vec{w}]\!]$
  $[\![\vec{tmp}]\!] \leftarrow ([\![0]\!], [\![w_2]\!], ..., [\![w_3]\!])$
  $[\![\vec{c}]\!] \leftarrow ([\![X]\!] \cdot [\![\vec{tmp}]\!])$
  $[\![\vec{t}]\!] \leftarrow [\![\vec{y}]\!] - [\![\vec{c}]\!] - [\![w_0]\!]$
  $[\![\vec{e}]\!] \leftarrow [\![\vec{t}]\!] \times [\![\vec{x}]\!]$
  $[\![g]\!] \leftarrow sum([\![\vec{e}]\!])$
  $[\![ST]\!] \leftarrow S([\![g]\!], [\![n\lambda\alpha]\!])$
  $[\![w_1]\!] \leftarrow [\![ST]\!]/[\![(n-1) \times \lambda \times (1-\alpha))]\!]$
  for $j = 2$ to $j \leq m$ **do**
    $[\![\vec{t}]\!] \leftarrow [\![\vec{t}]\!] + [\![w_k]\!] \times [\![\vec{x_j}]\!] - [\![w_k]\!] \times [\![\vec{x_{j-1}}]\!]$
    $[\![\vec{e}]\!] \leftarrow [\![\vec{t}]\!] \times [\![\vec{x_j}]\!]$
    $[\![g]\!] \leftarrow$ sum($[\![\vec{\text{e}}]\!]$)
    $[\![ST]\!] \leftarrow S([\![g]\!], [\![n\lambda\alpha]\!])$
    $[\![w_j]\!] \leftarrow [\![ST]\!]/[\![(n-1) \times \lambda \times (1-\alpha))]\!]$
  end for

3.2 Secure Lasso 回帰

Lasso 回帰は最小二乗法の損失関数に L1 ノルムを導入した手法であり, パラメータ更新式は式 (6) と式 (15) になる.

$$w_k = \frac{S(\sum_{i=1}^{n}(y_i - w_0 - \sum_{j=1(j\neq k)}^{d} x_{ij}w_j)x_{ik}, n\lambda)}{\sum_{i=1}^{n} x_{ik}^2} \tag{15}$$

Secure ElasticNet と同様に, 秘密計算での処理コストを軽減するため, 式 (15) を式 (16) から式 (19) に変形する.

$$\vec{t_1} = \vec{y} - w_0 - \sum_{j=2}^{d} x_{ij}w_j \tag{16}$$

$$w_1 = \frac{S(\sum_{i=1}^{n} t_{1i}x_{i1}, n\lambda)}{\sum_{i=1}^{n} x_{i1}^2} \tag{17}$$

$$\vec{t_k} = \vec{t}_{k-1} + w_k\vec{x}_k - w_{k-1}\vec{x}_{k-1} \tag{18}$$

$$w_k = \frac{S(\sum_{i=1}^{n} t_{ki}x_{ik}, n\lambda)}{\sum_{i=1}^{n} x_{ik}^2} \tag{19}$$

式 (17), (19) で用いられる t_{1i} と t_{ki} は式 (16), (18) のベクトルの要素である.

3.2.1 Secure Lasso 回帰プロトコル

Secure Lasso 回帰では, Algorithm 2, 5 に示すバイアス更新とパラメータ更新を反復することによりパラメータを推定する. n, m はそれぞれデータ数と属性数に対応し, また, 秘密計算で扱いやすいようにベクトルは属性ごとにもつ.

Algorithm 5 秘密計算での Lasso 回帰パラメータ更新

Require: $[\![X]\!](=[\![\vec{x_1}]\!],[\![\vec{x_2}]\!],...,[\![\vec{x_m}]\!])$
$[\![\vec{w}]\!](=[\![w_1]\!],...,[\![w_m]\!]),[\![w_0]\!]$
$[\![\vec{y}]\!](=[\![y_1]\!],[\![y_2]\!],...,[\![y_n]\!])$
ハイパーパラメータ $[\![\lambda]\!]$
Ensure: $[\![\vec{w}]\!]$
$[\![\vec{tmp}]\!] \leftarrow ([\![0]\!],[\![w_2]\!],...,[\![w_3]\!])$
$[\![\vec{c}]\!] \leftarrow ([\![X]\!] \cdot [\![\vec{tmp}]\!])$
$[\![\vec{t}]\!] \leftarrow [\![\vec{y}]\!] - [\![\vec{c}]\!] - [\![w_0]\!]$
$[\![\vec{e}]\!] \leftarrow [\![\vec{t}]\!] \times [\![\vec{x}]\!]$
$[\![g]\!] \leftarrow \mathrm{sum}([\![\vec{e}]\!])$
$[\![ST]\!] \leftarrow S([\![g]\!],[\![n\lambda]\!])$
$[\![w_1]\!] \leftarrow [\![ST]\!]/[\![(n-1)\times\lambda)]\!]$
for $j = 2$ to $j \leq m$ **do**
$[\![\vec{t}]\!] \leftarrow [\![\vec{t}]\!] + [\![w_k]\!] \times [\![\vec{x_j}]\!] - [\![w_k]\!] \times [\![\vec{x_{j-1}}]\!]$
$[\![\vec{e}]\!] \leftarrow [\![\vec{t}]\!] \times [\![\vec{x_j}]\!]$
$[\![g]\!] \leftarrow \mathrm{sum}([\![\vec{e}]\!])$
$[\![ST]\!] \leftarrow S([\![g]\!],[\![n\lambda]\!])$
$[\![w_j]\!] \leftarrow [\![ST]\!]/[\![(n-1)\times\lambda)]\!]$
end for

4. 実験

4.1 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

4.2 実験設定

行った実験は下記の 3 つである. すべての実験において更新回数は 10 回とした.

(1) 20 属性 ×1000〜1000 万件のダミーデータをデータセットとして用いた.

(2) 4, 20, 100 属性 ×10 万件のダミーデータをデータセットとして用いた.

(3) 14 属性 ×505 件の実データ（scikit-learn[1] に含まれるボストン市の住宅価格データ）をデータセットとしてを用いた. ハイパーパラメータは $\alpha = 1, \lambda = 0.5$ とした.

4.3 処理時間

処理時間は全て 5 回測定した結果の平均値である. default はパラメータ更新に Algorithm 3 を用い, 最適化有はパラメータ更新に Algorithm 4 を用い, 最適化有+シフト有は Algorithm 1, 4 を用いた結果である. 平文は sckit-learn[1] を用い, 秘密計算せずに ElasticNet を実行した結果である.

図 1 に実験（1）の処理時間を示す. データ数が 10 万件以下のときは, 数秒で処理が終わるため, 平文比べても劣らない時間であるがそれ以上になると, 平文より大幅に時間がかかってしまう.

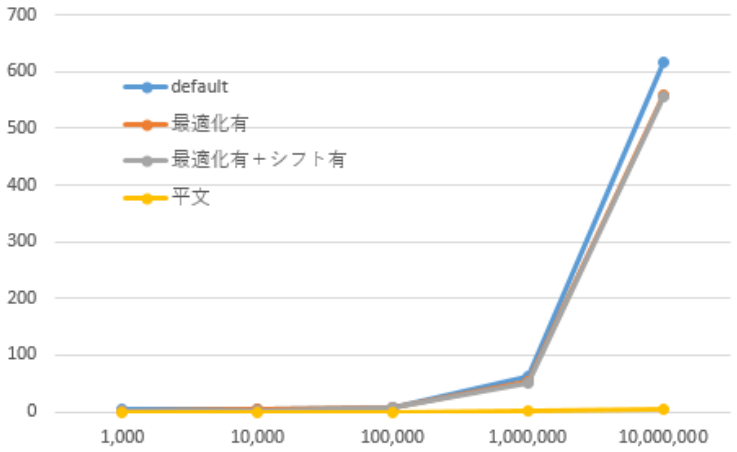

図 1 Secure ElasticNet 処理時間

図 2 に実験（2）の処理時間を示す. ElasticNet はパラメータの推定をする際にパラメータを 1 つづつ更新する必要がある. そのため秘密計算ではパラメータ更新 1 回にかかる処理コストが大きいため, 属性数が増えると処理時間が増えてしまう. 提案手法を用いることで, 属性数に対する処理時間の増加を軽減できる事がわかる.

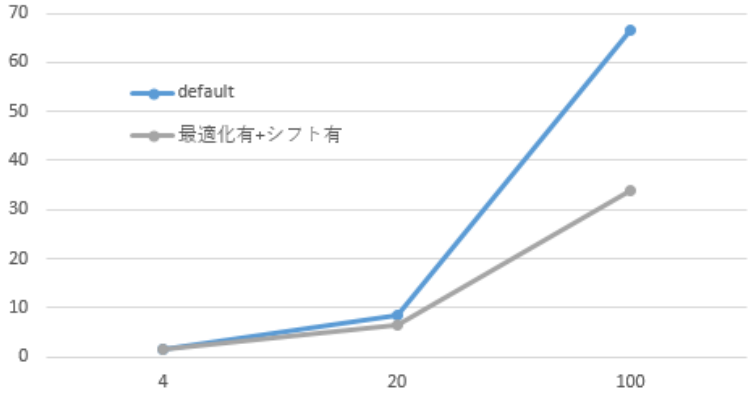

図 2 属性数に対する処理時間

4.4 計算精度

Secure ElasticNet と Secure Lasso 回帰では, 扱う値の精度を 30[bit] となるよう調節し計算を行っているため, 推定されるパラメータの精度も 30[bit] である. 実験（3）のパラメータ推定結果を表 2 に示す.

表 2 パラメータ推定結果

パラメータ	w_0	w_1	w_2	w_3	w_4
平文	22.5328	0	-0.34358	0.08096	-0.26979
提案手法	22.5328	0	-0.34320	0.08027	-0.26974
パラメータ	w_5	w_6	w_7	w_8	w_9
平文	0.40329	0.24034	2.36241	0	0
提案手法	0.40332	0.24089	2.36286	0	0
パラメータ	w_{10}	w_{11}	w_{12}	w_{13}	w_{14}
平文	0	-0.31045	-1.26679	0.46109	-2.33522
提案手法	0	-0.31078	-1.26703	0.46122	-2.33495

表 2 より少数第 3 位まで平文とほぼ同じ値を出力していることがわかる．また, 目的変数に影響の少ない説明変数を平文と同様に削減できている．

5. おわりに

本稿では, Soft-Thresholding 関数の軽量化と秘密計算のための計算式の最適化を行い, 大きいデータセットでも平文と同様の精度で ElasticNet および Lasso 回帰を秘密計算で実現可能なことを示した．

参考文献

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

[2] Borja Balle Mariana Raykova Jack Doerner Samee Zahur Adrià Gascón, Phillipp Schoppmann and David Evans. Privacy-preserving distributed linear regression on high-dimensional data, 2017.

[3] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. Fast and secure linear regression and biometric authentication with security update. *IACR Cryptology ePrint Archive*, Vol. 2015, p. 692, 2015.

[4] 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.

[5] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *TCC*, pp. 342–362, 2005.

[6] Rob Hall, Stephen E. Fienberg, and Yuval Nardi. Secure multiple linear regression based on homomorphic encryption, 2011.

[7] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing schemes realizing general access structure. In *Proc. of the IEEE Global Telecommunication Conf., Globecom 87*, pp. 99–102, 1987. Journal version: Multiple assignment scheme for sharing secret. J. of Cryptology, 6(1):15-20, 1993.

[8] Adi Shamir. How to share a secret. *Communications of the ACM*, Vol. 22, No. 11, pp. 612–613, 1979.

[9] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, Vol. 58, pp. 267–288, 1996.

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

[11] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, Vol. 67, pp. 301–320, 2005.

[12] 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.

[13] 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.

[14] 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.