Computer Security Symposium 2018
22 - 25 October 2018

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

三品　気吹[1]　五十嵐　大[1]　濱田　浩気[1]　菊池　亮[1]

概要：本稿では秘密分散ベースの秘密計算上で，効率的かつ高精度のロジスティック回帰分析を実装する．ロジスティック回帰分析は機械学習分野の基礎的かつ利用頻度の高い重要な分析手法であるが，シグモイド関数と呼ばれる非線形関数を必要とし，線形演算を得意とすることの多い秘密計算上では性能と精度の両立が容易でないことで知られる．本稿ではこのシグモイド関数の計算法に関して 2 種類の方式を提案・実装する．一つは，秘密一括写像を用いる方式である．秘密一括写像では処理コストとトレードオフの任意精度で，任意の関数すなわち写像を計算できる．このため一つ目の方式では平文上のロジスティック回帰と遜色ない予測精度となることが確認された．二つ目は，条件分岐と線形近似のハイブリッド型の近似である．こちらは精度を平文と比較して 2%の誤差に抑えながらも，既存方式よりも 12 倍程度高速であったことを報告する．

キーワード：秘密計算，秘密分散，ロジスティック回帰

Designs and Implementations of Efficient and Accurate Secret Logistic Regression

1. はじめに

秘密計算とは，データを秘匿したまま任意の関数を計算する暗号技術であり，この特徴を活かした，システム運用者にもデータ利活用者にもデータを漏らさないデータ利活用の形態が期待されている．例えば, 個人の病歴を秘匿しつつ健康指導を行ったり，IoT 機器のログを秘匿しつつ異常を検知することが可能となる．その中でも特に複数のデータホルダーからデータを収集し，上記 2 者に加えてデータホルダー同士も他者のデータが分からないような，統合データ分析は特に秘密計算特有の応用であり有望である．

近年の秘密計算の実用化研究の発展により，秘密計算によるデータ分析として，統計分析システムが既に幾つか実用化されている．それと同時に研究が盛んになり始めているのが，機械学習やデータマイニングと呼ばれる，統計分析をさらに発展させたより高度かつ複雑な機能領域である．機械学習・データマイニングはビッグデータ・IoT データの活用においても AI においても技術のコアであり，今後ますますニーズは加速すると考えられる．

本稿の大きな目的は，上記領域を秘密計算で実用的な性能とスケーラビリティを伴ってカバーすることで秘密計算の価値を世の中に広く提供することである．

そんな中，2017 年，セキュリティのトップ会議の一つである S&P において，機械学習の基本的なアルゴリズムである線形回帰，ロジスティック回帰，ニューラルネットワークを秘密計算上で実装した SecureML[9] が提案された．SecureML は 100 万件 784 属性のデータに対するロジスティック回帰を数分で処理できるなど，非常に現実的な性能を持つ画期的なソフトウェアである．しかし，ロジスティック回帰の分析精度面に関しては，アルゴリズム中に含まれるシグモイド関数を簡易な関数で代替しており，本稿の実験結果として後述するように出力の精度に課題がある．

1.1 本稿の成果

本稿では，秘密計算での計算が容易ではない非線形関数 (シグモイド関数) を含むロジスティック回帰分析を，秘密分散ベース秘密計算上で高精度かつ高速に実現した．提案の中心となるのはシグモイド関数の秘密計算上における高精度・高効率な近似法であり，下記の 2 つの近似手法を提案し，それらを用いたロジスティック回帰を実装評価した．

[1] NTT セキュアプラットフォーム研究所

(1) 秘密一括写像による方法
(2) 条件分岐と線形近似のハイブリッド近似法

[高精度] レコード数 1,000〜1,000 万件の学習データを用いて学習を行った実験では，平文との予測精度の差の平均は既存の最も高速かつ精度の良い結果である SecureML [9] で 10〜15%のところ，秘匿一括写像でほぼ 0，ハイブリッド近似で 2%程度と，既存手法と比べ大きく改善し平文と比較しても遜色ない高い精度を示した．

[高速] 処理時間は 1,000 万レコード，説明変数が 2 属性のデータで秘密一括写像で約 40 秒，ハイブリッド近似では約 7.5 秒と実用的な性能であった．また既存研究との比較では，上記 SecureML では 100 万レコード 784 属性で約 405 秒のところ，本稿では秘匿一括写像で 50.45 秒，ハイブリッド近似で 34.73 秒と，前者は約 8 倍，後者は約 12 倍と大きく上回る結果である．

1.2 秘密分散を用いた秘密計算

秘密計算には幾つかの方式が存在する．その中でも秘密分散を構成要素にするものは，データの処理単位が小さく，高速な処理が可能であることが知られている [1], [14].

秘密分散とは，秘密情報をシェアと呼ばれる幾つかの断片に変換する方法である．本論文では 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) 閾値法と呼ばれる秘密分散を扱う．秘密分散の具体的な構成方法は Shamir の秘密分散 [11] や複製秘密分散 [4], [6] などが知られている．

1.3 ロジスティック回帰分析

回帰分析とは，ある既知の変数 (説明変数)X を元にして別の変数 (目的変数)Y の値を導出する関係式 (モデル)$Y = f(X)$ を推定することである．例えば “年齢” と “血圧” という 2 つの属性があり，年齢を説明変数，血圧を目的変数とした場合に，年齢から血圧を計算する関係式を得ることが回帰分析となる．最も基本的なモデルは $Y = aX + b$ で表される，線形単回帰モデルである．モデルを決定する係数 a や切片 b を，まとめて “モデルパラメータ” と呼ぶ．モデルパラメータの推定には，教師あり学習という機械学習の手法を用いる．教師あり学習とは学習データ (説明変数と目的変数の組の列) を用意し，これらのデータを用いて最も適切な $Y = f(X)$ のモデルパラメータを推定する (学習する) 手法である．学習した $f(X)$ に新たな X を入力することで，未知の Y を予測できるようになることが回帰分析の最終目標である．上記の例では，年齢から血圧を予測できるようになることが最終目標となる．説明変数は 2 属性以上でも良く，説明変数が 1 属性のものを単回帰，2 属性以上のものを重回帰と呼ぶ．

ロジスティック回帰分析 [3] は，目的変数が 2 値，典型的には “事象 A が起こるかどうか” の場合の回帰分析である．線形回帰ではモデルが直線のため，出力は実数全体の範囲をとり，“事象 A の起こりやすさ” として解釈することは難しい (値が幾つだとどれくらい起こりやすいのか分からない)．これに対してロジスティック回帰分析は 0〜1 の範囲の “A が起こる確率” という直感的な値を推定できることが特徴である．例えばある人の健康診断結果から，その人がある疾患に罹患している確率を推定できる．このような “事象が起きる確率” を知りたいケースはごく一般的であるため，ロジスティック回帰分析は機械学習の中でも特によく使われる分析の一つである．

1.4 関連研究

青野ら [16] や呉ら [13] はシグモイド関数もしくはそれに類する関数を多項式で近似し，加法準同型暗号を用いてロジスティック回帰における学習を行った．暗号化した状態では加算のみしか計算できないため，クライアントは多項式計算専用のデータを用意する必要があり，また学習する毎にその時得られたモデルパラメータは秘密にすることができない．

Hane ら [5] は，シグモイド関数を 3 次多項式で近似し，完全準同型暗号を用いてミニバッチ確率的勾配降下法での学習をデータを暗号化したまま行った．加法準同型と異なり暗号化したまま繰り返し学習が可能だが，その分処理速度は低下しており，例えば 11,982 件の 196 属性データに対する学習で 132 分かかったと報告されている．

Mohassel と Zhang [9] は，秘密分散に garbled circuit [12] や紛失通信 [10] といった暗号を組み合わせ，シグモイド関数は簡易な関数に置き換えてミニバッチ確率的勾配降下法での学習を行う SecureML を実装した．一部処理の処理時間が明記されていないが，記載されている処理時間のみの総計による推定では 100 万レコード 784 属性で約 405 秒である．分析精度に関しては [9] の実験では二値分類に適用した場合に 97.9%の精度だったとされているが，本稿で実験したところシグモイド関数を用いた場合と平均 10%以上の乖離があり (4 節)，二値分類では乱数でも 50%の精度となることを考えると精度が高いとは言えないだろう．

2. 準備

ベクトルを $\vec{a} := (a_0, \ldots, a_{n-1})$ と書き．a を b で定義することを $a := b$ と書き，同じ要素数の 2 つのベクトル $\vec{a}, \vec{b}$ の内積を $\vec{a} \cdot \vec{b}$ と書く．また，長さ n のベクトル $\vec{a}$ に対し $(1, \vec{a})$ はベクトルの結合を意味し，$(1, \vec{a}) := (1, a_0, \ldots, a_{n-1})$ である．m 個の要素の集合 $\{a_i \mid 0 \leq i < m\}$ を単に $\{a_i\}_m$ と書く．$\mathbb{Z}_2$ を $\mathbb{Z}/2\mathbb{Z}$ とする．$[\cdot]$ は述語を表し，例えば $[a > b]$ は $a > b$ ならば 1，そうでなければ 0 である．

2.1 ロジスティック回帰分析

ロジスティック回帰分析のモデルは，$\vec{w} \in \mathbb{R}^{n+1}$ をパラメータとする，下記の式である．

$$f(\vec{x}) = \sigma(-\vec{w} \cdot (1, \vec{x}))) \tag{2.1}$$

ただし $\cdot$ は内積, $(1, \vec{x})$ は $(1, x_1, \ldots, x_n)$ である．σ はシグモイド関数 (またはロジスティック関数) と呼ばれる．シグモイド関数 $\sigma(x)$ は式 2.2 で表され，図 2.1 で示すように $\lim_{x \to \infty} \sigma(x) = 1$, $\lim_{x \to -\infty} \sigma(x) = 0$ となる性質を持っている単調増加連続関数である．

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{2.2}$$

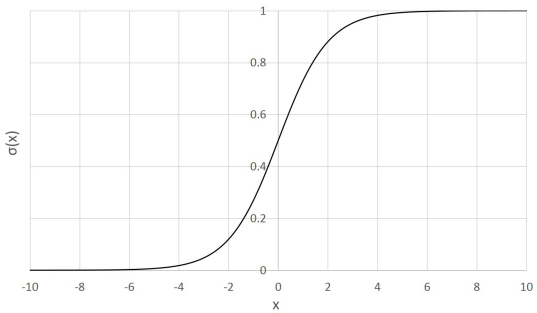

図 2.1 シグモイド関数

式 2.1 の内積部分は，線形回帰モデル $Y = aX + b$ の説明属性 X を複数属性とする下記の式に対応する．

$$f(\vec{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \tag{2.3}$$

ロジスティック回帰モデルは式 2.3 の左辺を “A である確率 p”/“A でない確率 $1 - p$” を表すオッズ比の対数とした以下の式に由来する．

$$\log \frac{p}{1-p} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \tag{2.4}$$

式 2.4 を p に関して解くと式 2.3 となり，問題であるシグモイド関数が出現する．

2.1.1 最急降下法

式 2.3 におけるモデルパラメータである $\vec{w}$ を学習するための手法として，関数の最小値を探索する学習法である最急降下法 (Steepest Descent Method) が知られている．

- データセット $\{\vec{x_i}, y_i\}_m$ (m はデータ数, $\vec{x_i}$ は説明変数，y_i は目的変数)
- 適切な初期モデルパラメータ $\vec{w}$
- 適切に設定した学習率 $\eta (0 < \eta < 1)$
- 適切に設定した学習回数 T

最急降下法ではモデルパラメータ $\vec{w}$ に適当な初期値を与えた後，自分で設定した学習回数だけ式 2.5 を計算して $\vec{w}$ の学習を行う．式 2.5 では，w_j と $x_{i,j}$ はそれぞれ $\vec{w}$ と $(1, \vec{x_i})$ の $j (0 \leqq j \leqq n)$ 番目の値を表し，$t (1 \leqq t \leqq T)$ 回更新を行ったモデルパラメータ w_j を $w_{j,t}$ と表している．

$$w_{j,t+1} = w_{j,t} - \eta \frac{1}{m} \sum_{i=0}^{m} (f(\vec{x_i})_t - y_i) x_{i,j} \tag{2.5}$$

$$f(\vec{x_i}) = \sigma(\vec{w} \cdot (1, \vec{x_i})) \tag{2.6}$$

2.1.2 ミニバッチ確率的勾配降下法

最急降下法は全データを用いて式 2.5 を計算するため，学習を繰り返す度により精度の高い，すなわち目的変数を説明変数が良く予測するパラメータに更新されていく一方，学習のたびに全データを用いて学習するため，処理時間が多くかかったり，大きいデータではメモリ上に乗らず実行が難しいなどといった問題がある．ミニバッチ確率的勾配降下法はこれらを解決するために，全データのうち一部のデータ（ミニバッチと呼ばれる）を入力とし，式 2.5 を用いてパラメータを更新していく方法である．全データを用いていないため一定の確率で精度の悪いパラメータに遷移することもあるが，学習を繰り返せば精度の高いパラメータへ収束していくことに加え，全データで学習する必要がないため，効率的な学習が可能である．

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秘密計算のライブラリである [15]．MEVAL では，ユーザが加算・乗算・ソート・等結合等の 100 以上の演算を組み合わせて自由にプログラムできる．

本稿ではこの特徴を用い，MEVAL 上のプログラムとしてロジスティック回帰を計算する．MEVAL は演算に応じて複数の秘密分散を用いている．特にロジスティック回帰分析で用いる秘密分散は以下の 2 つである．

- $[\![\cdot]\!]$: $\mathrm{GF}(2^{61} - 1)$ 上の Shamir 秘密分散のシェア．
- $\{\cdot\}$: $\mathbb{Z}_2$ 上の複製秘密分散のシェア．

これらのシェアの列をそれぞれ $[\![\vec{a}]\!], \{\vec{a}\}$ と書く．どちらのシェアにおいても，それぞれの環での定数倍・加算は通信無しで行うことができる．これを $a[\![b]\!] + c = [\![a\vec{b} + c]\!] = (ab_0 + c, ab_1 + c, \ldots, ab_{m-1} + c)$ などと書く．また，上記のようにベクトルの要素ごとに同じ演算を行う場合，ベクトルの単一要素への演算と同様の記法をとる．MEVAL で用意されている演算のうち，本論文では下記を用いる．処理単位はベクトルであることに注意されたい．

- and: 2 つのビット列のシェア $\{\vec{a}\}, \{\vec{b}\}$ を入力とし，要素毎の論理積 $\{\vec{a} \wedge \vec{b}\}$ を出力する．
- not: ビット列のシェア $\{\vec{a}\}$ を入力とし，要素毎の論理否定 $\{\neg\vec{a}\}$ を出力する．
- mul: 2 つのシェアの列 $[\![\vec{a}]\!], [\![\vec{b}]\!]$ を入力とし，要素毎の積 $\{\vec{ab}\}$ を出力する．
- hpsum: $\vec{a_i} := (a_{i,0}, \ldots, a_{i,m-1})$ としたとき，2 つのシェアベクトルの列

$(\llbracket \vec{a_0} \rrbracket, \ldots, \llbracket \vec{a_{n-1}} \rrbracket)$ と $(\llbracket \vec{b_0} \rrbracket, \ldots, \llbracket \vec{b_{n-1}} \rrbracket)$ を入力とし，各シェアベクトルの要素ごとの積和 $(\llbracket \sum_{i=0}^{n-1} a_{0,i} b_{0,i} \rrbracket, \ldots, \llbracket \sum_{i=0}^{n-1} a_{m-1,i} b_{m-1,i} \rrbracket)$ を出力する．

- less_than: シェアの列 $\llbracket \vec{a} \rrbracket := (\llbracket a_0 \rrbracket, \ldots, \llbracket a_{m-1} \rrbracket)$ と公開値 t を入力とし，$\{[\vec{a} < t]\}$ $(= (\{[a_0 < t]\}, \ldots, \{[a_{m-1} < t]\}))$ を出力する．
- greater_than: シェアの列 $\llbracket \vec{a} \rrbracket$ と公開値 t を入力とし，$\{[\vec{a} > t]\}$ を出力する．
- map:秘密一括写像. シェアの列 $\llbracket \vec{a} \rrbracket := (\llbracket a_0 \rrbracket, \ldots, \llbracket a_{m-1} \rrbracket)$ と定義域および値域 $(x_0, \ldots, x_{\ell-1})$, $(y_0, \ldots, y_{\ell-1})$ を入力とし，各入力値を写像させたシェア，すなわち $0 \leq i < m$ について $x_j \leq a_i < x_{j+1}$ 且つ $b_j = y_i$ であるような $(\llbracket b_0 \rrbracket, \ldots, \llbracket b_{m-1} \rrbracket)$ を出力する．
- toP: シェアの列 $\{\vec{a}\}$ を入力とし，環を変換した $\llbracket \vec{a} \rrbracket$ を出力する．

2.3 先行研究におけるシグモイド関数の近似法

2.3.1 最小二乗法による多項式近似

[5][13] では最小二乗法によって得た多項式でシグモイド関数を近似している．

例として，-18〜18 の範囲で最小二乗法を用いてシグモイド関数の 3 次多項式近似を行うと図 2.2 のようになる．

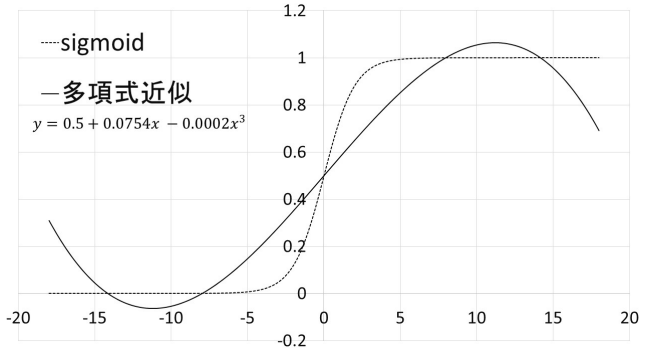

図 2.2 3 次関数による多項式近似

2.4 SecureML の方法

SecureML [9] では，シグモイド関数ではないが $\lim_{x \to \infty} f(x) = 1$, $\lim_{x \to -\infty} f(x) = 0$, 単調増加という性質を持つ，簡略化した関数を用いた．

$$f(x) = \begin{cases} 1 \text{ if } x > 1/2 \\ 1/2 + x \text{ if } -1/2 \leq x \leq 1/2 \\ 0 \text{ if } x < -1/2 \end{cases} \tag{2.7}$$

この関数は，1/2 が 2 進数で 0.1 であり位数 2 べきの整数環や論理回路で非常に高速な処理ができることを利用した，速度最優先の関数である．

3. 提案方式

注記:ここまではベクトルは主にテーブルの 1 行（1 レコード）を表したが，ここからは実際の処理に合わせてベクトルはテーブルの 1 列 (1 属性) を表す．

まずは提案するロジスティック回帰プロトコルの全体を以下の Algorithm1 に示す．全体では最急降下法を採用したプロトコルとなっており，プロトコル内の Sigmoid という演算で，本稿で提案する 2 種類のシグモイド関数の計算法を用いる．

本稿では処理コストの観点で，浮動小数点ではなく固定小数点数を用いる．秘密分散では主に整数が分散されるため，小数を含む値を整数にエンコードする必要があり，精度 $\alpha^{-\beta}$ のとき，秘密 x は $\lfloor x\alpha^{\beta} \rfloor$ として分散されている．例えば精度が 10^{-5} である場合，2.987654 は 298765 として分散される．固定小数点数では，乗算のたびに数値精度が増大しデータ型の上限を超えてしまうことがあるが，途中で意図的な桁落とし（数値精度の調整）を行うことでこれを回避できる．

Algorithm 1 秘密ロジスティック回帰

入力 1：$\llbracket \{\vec{x_j'}\}_{n+1} \rrbracket = (\llbracket \vec{x_0'} \rrbracket, \llbracket \vec{x_1'} \rrbracket, \cdots, \llbracket \vec{x_n'} \rrbracket)$

入力 2：$\llbracket \vec{y'} \rrbracket = (\llbracket y_1' \rrbracket, \llbracket y_2' \rrbracket, \cdots, \llbracket y_m' \rrbracket)$

パラメータ：学習率 $\eta (0 < \eta < 1)$，学習回数 T

出力：モデルパラメータ $\llbracket \vec{w} \rrbracket = (\llbracket w_0 \rrbracket, \llbracket w_1 \rrbracket, \cdots, \llbracket w_n \rrbracket)$

1: $\llbracket \vec{w} \rrbracket$ を適当な値で初期化
2: $a \leftarrow 1/m$
3: $t \leftarrow 0$
4: **while** $t < T$ **do**
5: $\quad \llbracket \vec{b} \rrbracket \leftarrow \mathsf{hpsum}(\llbracket \vec{w_t} \rrbracket, \llbracket \{\vec{x_j'}\}_{n+1} \rrbracket)$
6: $\quad \llbracket \vec{c} \rrbracket \leftarrow \mathsf{Sigmoid}(\llbracket \vec{b} \rrbracket)$
7: $\quad \llbracket \vec{d} \rrbracket \leftarrow \llbracket \vec{c} \rrbracket - \llbracket \vec{y'} \rrbracket$
8: $\quad j \leftarrow 0$
9: $\quad$ **while** $j < n + 1$ **do**
10: $\quad\quad \llbracket e \rrbracket \leftarrow \sum_{i=0}^{m} \llbracket d_i \rrbracket \llbracket x'_{j,i} \rrbracket$
11: $\quad\quad \llbracket w_{j,t+1} \rrbracket \leftarrow \llbracket w_{j,t} \rrbracket - \eta a \llbracket e \rrbracket$
12: $\quad\quad j \leftarrow j + 1$
13: $\quad$ **end while**
14: $\quad t \leftarrow t + 1$
15: **end while**

3.1 秘密一括写像によるシグモイド関数の近似

秘密一括写像はルックアップテーブルを計算する機能であり，定義域と値域を任意に定めることができる．例として入力を 10 倍する関数を一括写像で計算する場合を考える．ある定義域 $X = \{1, 3, 5\}$ と，その 10 倍の値の集合である値域 $Y = \{10, 30, 50\}$ を用意したとする．秘密一括写像では，定義域に属さない入力 x に対しては x 以下の最大の定義域に対応する値が出力される．そのため 4 を入力すると 30 が出力される．しかし定義域を $X = \{1, 2, 3, 4, 5\}$,

値域を $Y = \{10, 20, 30, 40, 50\}$ というふうにより細かく設定することで，4 を入力したときに 40 が出力されるようになり，高い精度での計算が可能になる．この性質を使うことで，固定小数点数でもデータ型の上限を超えることがなく且つ計算誤差も低い，適切な精度を設定することができる．

“一括写像” というのは，入力として比較的サイズの大きなベクトルに対して (要素ごとの) 写像を計算することを目的としているということである．一回の一括写像には，入力ベクトルのサイズに非依存だが，定義域や値域のサイズに比例するオーバーヘッドがかかる．そのため，10^4 程度の写像を用意し 10^{-4} 程度の精度とするのが現実的である．

シグモイド関数を近似するために用いた定義域は，$-15 \leqq x \leqq 15$ の範囲を定義域の要素数で分割することで決めている．$\sigma(-15) < 10^{-6}, 1 - \sigma(15) < 10^{-6}$ となるため，無視しても精度に殆ど影響を与えないためである．

3.2 条件分岐と線形近似のハイブリッド近似

秘密一括写像はシグモイド関数を誤差 10^{-4} などの十分高い精度で計算でき精度面で有利だが反面，処理コストは多項式近似よりも大きい．ここでは 2 つ目の方針として，SecureML の方式を拡張してシグモイド関数を近似する方式を提案する．同時に，実装に必要である，効率的な Mersenne 素体上右シフトプロトコルの提案も行う．

SecureML では式 2.7 を用いてシグモイド関数の代替とした．本稿ではこれを下記のように一般化し，x がある閾値 t_1 以上なら 1, t_0 と t_1 の間ではある $g(x)$, t_0 未満では 0, と大小比較による条件分岐を行う．

$$\text{近似シグモイド関数 } \sigma'(x) = \begin{cases} 1 \text{ if } x \geq t_1 \\ g(x) \text{ if } t_0 < x < t_1 \\ 0 \text{ if } x \leq t_0 \end{cases}$$

これは, IF c THEN a ELSE b の条件分岐が数式 $ca+(1-c)b$ で表現できることを用い，$c = [x < t_1]$, $d = [x > t_0]$ とおくと，$cdf(x) + (\neg c)$ と表せる．

$g(x)$ に関しては，性能と精度のバランスとして，一次関数がかなり良い．秘密分散ベースの秘密計算では加算および公開値との乗算の処理時間はほぼ無視してよいほど高速であるため，公開値である傾きを掛けて切片を加算するだけである一次関数の計算コストはほぼゼロである．傾きと切片に関しては，$x = 0$ 付近の近似精度が高いことが良い結果につながるという経験則に基づき，シグモイド関数の $x = 0$ における接線 $g(x) = 1/2 + (1/4)x$ を採用した．近似精度を確認するため, $t_0 = -2$, $t_1 = 2$, $g(x) = 1/2 + (1/4)x$ としたグラフを図 3.1 に示す．シグモイド関数の近似を目指していない secureML の関数は言うまでもなく，多項式近似の図 2.2 よりも十分に近似精度が高いことが，目視でも見てとれる．

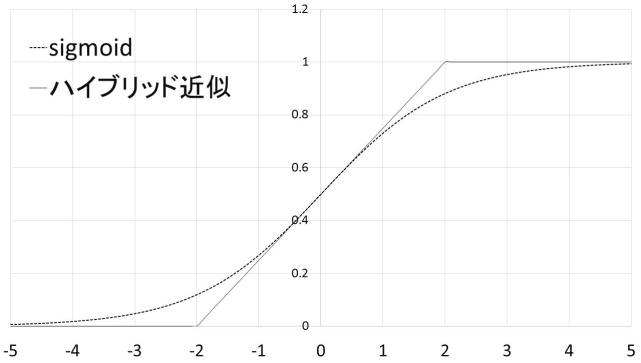

図 3.1 ハイブリッド近似

秘密計算上で図 3.1 のような近似を行うためのプロトコルを Algorithm 2 に示す．本プロトコルでは MEVAL に用意されている関数のうち 2.2 節に記載されているものを用いている．

Algorithm 2 条件分岐と線形近似によるシグモイド関数のハイブリッド近似
入力：$[\![\vec{x}]\!] = ([\![x_0]\!], [\![x_2]\!], \cdots, [\![x_{m-1}]\!])$
パラメータ：t_0, t_1, a, b
出力：$[\![\sigma'(\vec{x})]\!] := ([\![\sigma'(x_0)]\!], [\![\sigma'(x_2)]\!], \cdots, [\![\sigma'(x_{m-1})]\!])$

1: $\{\vec{c}\} \leftarrow \mathsf{less\_than}([\![\vec{x}]\!], t_0)$
2: $\{\vec{d}\} \leftarrow \mathsf{greater\_than}([\![\vec{x}]\!], t_1)$
3: $\{\vec{e}\} \leftarrow \mathsf{not}(\{\vec{c}\})$
4: $\{\vec{k}\} \leftarrow \mathsf{and}(\{\vec{c}\}, \{\vec{d}\})$
5: $[\![e]\!] \leftarrow \mathsf{toP}(\{\vec{e}\})$
6: $[\![k]\!] \leftarrow \mathsf{toP}(\{\vec{k}\})$
7: $[\![\sigma'(\vec{x})]\!] := \mathsf{mul}([\![k]\!], a[\![\vec{x}]\!] + b) + [\![\vec{e}]\!]$

3.2.1 Mersenne 素体上右シフトプロトコル

秘密一括写像を用いる方法では，写像の入力 x の数値精度が増大しても値域を定義域とは独立に設定できるため，数値精度を任意に調整できた．しかしハイブリッド近似は論理演算と線形変換から成るため，このような数値精度の調整ができず，データ型の上限を超えないよう，意図的に数値精度を落とす演算が必要となる．

本稿ではそのような数値精度を落とす演算として，右シフトプロトコルを提案する．右シフトは [2] で提案されているが位数 2 べきの整数環上であり，本稿では MEVAL で扱われる，Mersenne 素体上のプロトコルを提案する．Mersenne 素体は MEVAL 以外にも [8] 等でも採用されており，特に重要な構造である．

3.2.1.1 数学的解析

右シフトは整数除算であり, 加算・乗算から成る環演算ではない. 定理 3.1 で一般の整数除算を, そしてその系 3.1 で右シフトを, 加法的秘密分散の商 q を用いた環演算で表現する. 商 q は商転移により効率的に計算できる [7].

定理 3.1 ($\mathbb{Z}_p$ 上の公開値による商) 任意の分散数 $m \in$

$\mathbb{N}$, 素数 $p \in \mathbb{N}$, 除数 $d \in \mathbb{N}$, 被除数 $a \in R$, $a = \sum_{i<m} a_i \mod p$ を満たすシェア $a_0, \ldots, a_{m-1}$ があるとする.
ここで, 各 i に対して非負整数 a_{iQ}, a_{iR} はそれぞれ a_i の d による商と剰余, 非負整数 p_Q, p_R はそれぞれ p の d による商と剰余, 非負整数 q は $\sum_{i<m} a_i$ の p による商, さらに z_Q は $\sum_{i<m} a_{iR} + q(d - p_R)$ の d による商とおく.
このとき, a の d による商 a_Q は以下の式で表される.

$$a_Q = \sum_{i<m} a_{iQ} - (p_Q + 1)q + z_Q \tag{3.1}$$

証明 1 仮定 $a = \sum_{i<m} a_i \mod p$ より, a は $\sum_{i<m} a_i$ の p による剰余である. 仮定 "q は $\sum_{i<m} a_i$ の p による商" より, $\sum_{i<m} a_i = qp + a$ である.
仮定より $a_i = a_{iQ}d + a_{iR}$, $p = p_Q d + p_R$ だから,

$$\begin{aligned} a &= (\sum_{i<m} a_{iQ})d + \sum_{a_{iR}} -qp_Q d - qp_R \\ &= (\sum_{i<m} a_{iQ} - qp_Q)d + (\sum_{i<m} a_{iR} - qp_R) \end{aligned}$$

である. さらに $\sum_{i<m} a_{iR} - qp_R$ の d による商を y_Q とおけば,

$$a_Q = (\sum_{i<m} a_{iQ} - qp_Q) + y_Q$$

である. 一方 z_Q は $\sum_{i<m} a_{iR} + q(d - p_R)$ の d による商だから, $y_Q = z_Q - q$ である.
□

系 3.1 (Mersenne 素体上右シフト公式) 定理 3.1 で素数ビット長 $\ell \in \mathbb{N}$ を任意, シフトビット数 $b \in \mathbb{N}$ を $b \leq \ell$ として素数 $p = 2^\ell - 1$, 除数 $d = 2^b$ とする.
すると z'_Q は $\sum_{i<m} a_{iR} + q$ の 2^b による商であり, a_Q は以下の式で表される.

$$a_Q = \sum_{i<m} a_{iQ} - 2^{\ell-b} f + z'_Q \tag{3.2}$$

証明 2 詳細は略. $p = 2^\ell - 1$, $d = 2^b$ を代入せよ.
□

3.2.1.2 プロトコル本体

提案する右シフトプロトコルを 3 に記す. 系 3.1 の式 3.2 に従って計算する. この節では以下の記法を用いる. 提案するプロトコルは種々の秘密分散で適用できるため, $[\![\cdot]\!]$ と $\{\cdot\}$ はより一般的に定義される. また, 商転移定理の系 [7] を使っている.

- $[\![\cdot]\!]$: 入出力の形式となる秘密分散で分散された値. 加法, 公開値倍, 下記の加法的秘密分散との相互変換, 下記のビット列からのビット合成ができる必要がある. 例は Shamir の秘密分散, 複製秘密分散である.
- $\langle\cdot\rangle$: 加法的秘密分散 (復元が加法で行われる秘密分散) で分散された値. 例は単純な加法的秘密分散, 複製秘密分散である.
- $\{\cdot\}$: ビット列として分散された値. すなわち, ビットを秘密分散した値の列として数値をビット表現した値. 論理回路を処理できる必要がある. 例は $\mathbb{Z}_2$ 上の複製秘密分散の列である.

系 3.2 (Mersenne 素体上の商転移) 任意の分散数 $k \in \mathbb{N}$ と Mersenne 素数 p に対して, $u \in \mathbb{N}$ は $\log m \leq u$ を満たし, $a_0, \ldots, a_{m-1} \in \mathbb{Z}_p$ は $\sum_{i<m} a_i \mod 2^u = 0$ を満たすものとする. このとき q を $\sum_{i<m} a_i$ の p による商とすると, q は下記の式で表される.

$$q = -\sum_{i<m} a_i \mod 2^u$$

Algorithm 3 $\mathbb{Z}_p$ 上右シフトプロトコル (シフト量公開)
入力: 被シフト数 $[\![a]\!]$, シフト量 b
パラメータ: 加法的秘密分散の分散数 m, $\log m$ 以上の整数 u
出力: $[\![a \gg b]\!]$(ただし $\gg$ は右シフト演算子)

1: 公開値倍により $[\![a']\!] = [\![2^u a]\!]$ を計算する.
2: $[\![a']\!]$ を加法的秘密分散に変換し, $\langle a'\rangle$ を得る.
3: $\langle s\rangle$ を, $\langle s\rangle_i = \langle a'\rangle_i \gg b + u$ とする.
4: $\langle s\rangle$ を線形秘密分散に変換して $[\![s]\!]$ を得る.
5: 各 $\langle a'\rangle_i$ の下位 u ビットを分散して, $\langle a'\rangle_i \mod 2^u$ のビット表現 $\{a'_i \mod 2^u\}$ を得る.
6: 加算回路・符号反転回路で $-\sum_{i<m} \{a'_i \mod 2^u\}$ の下位 u ビットを計算する. この出力は商転移定理より q に等しいので, $\{q\}$ とおく.
7: 各 $\langle a'\rangle_i$ の下位 $b+u$ ビットを分散して, $\langle a'\rangle_i \mod 2^{b+u}$ のビット表現 $\{a'_{iR}\}$ を得る.
8: 加算回路により $\{z\} = \sum_{i<m} \{a'_{iR}\} + \{q\}$ を計算し, (一部, 上記 u ビットの加算回路の処理結果を使い回せる.) $b+u$ ビット目 (0 スタート) 以降のビット列を $\{z_Q\}$ とおく.
9: $\{q\}, \{z_Q\}$ をビット合成により線形秘密分散に変換し, $[\![q]\!], [\![z_Q]\!]$ を得る.
10: $[\![s]\!] - [\![2^{\ell-(b+u)}]\!] + [\![z_Q]\!]$ を出力する.

線形秘密分散と複製秘密分散を含む加法的秘密分散の相互変換, ビット合成, 複製秘密分散の各ビットの分散はいずれも [7] に記載がある.

3.2.1.3 効率

右シフトプロトコルの通信量とラウンド数を評価する. 係数は考慮し, 定数項はを無視する. 本稿の想定では u は passive で 1, active でも 2 と小さいため, u は定数と見なして評価する.
[通信量] パーティあたりの送信ビット数を単位とする. 線形秘密分散から加法的秘密分散への変換は 0, 加法的秘密分散から線形秘密分散への変換は ℓ, 下位 $b+u$ ビットの分散は b, 加算回路は b, 符号反転回路は 0, ビット合成は 1 回

あたり ℓ を 2 回であり, 合計は素数ビット長 ℓ，シフト量 b を使って $3\ell + 2b$ と表される.

[ラウンド数] 加算回路は b ラウンド, 他は定数ラウンドであり, 合計 b ラウンドである.

4. 実験

4.1 実装方法

シグモイド関数の計算には 2 種類の提案方式を用い，さらにそれぞれの方式についてバッチ法 (最急降下法) とミニバッチ法 (ミニバッチ確率的勾配降下法) の両方を MEVAL3[7] で実装した．比較用として実装した平文のバッチ法とミニバッチ法は，numpy のベクトル演算を用いて高速化した python のプログラムである．

4.2 測定環境

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 7.2.1511
CPU	Intel Core i7 6900k(3.20GHz 8 コア/16 スレッド)
メモリ	32GB
NW	サーバ間 IntelX550T2 10Gbps 2 ポート リング型接続

4.3 データセットの生成方法

以降全ての実験で，データセットは下記のロジックで生成した.

(1) $n+1$ 個の一様乱数 $w_0, w_1, \cdots, w_n$ を生成し，正解となるモデル $f(\vec{x}) = w_0 + w_1 x_1 + \cdots + w_n x_n$ を作る

(2) 説明変数となる一様乱数 $\vec{x} = (x_1, x_2, \cdots, x_n)$ を生成

(3) $\vec{x}$ を $f(\vec{x})$ に代入し，$f(\vec{x}) < 0$ ならば $y = 0$，それ以外は $y = 1$ として学習データを生成

4.4 実験 1:レコード数を変化させた場合の精度・性能

実験 1 では 1,000〜10,000,000 レコード, 説明変数が 2 属性のデータセットを生成した．1,000 件〜100,000 件のデータに対してはバッチ法，100,000 件〜10,000,000 のデータに対してはミニバッチ法を採用している (100,000 件では両方を実験した)．バッチ法は少ないデータ数でも安定したパラメータ推定ができるが大きいデータではメモリや処理時間を多く消費すること，ミニバッチ法は効率的な学習ができるが少ないデータ数ではパラメータ推定の精度が悪くなる場合もあることから，このようにデータ数に応じて適切な学習法を選択した．バッチ法における学習反復回数は 100 回，ミニバッチ法のバッチサイズはレコード数の 1/100 である．図 4.1 では，平文のバッチ法で得られた予測精度 [%] と，同じ学習データを用いて各提案方式および既存方式で得られた予測精度 [%] の差を計算し，その 10 回分を平均した値を誤差として示している．$*$ を付記した項目が提案手法である．図 4.2 の処理時間は，各手法で 5 回ずつ測定して得られた結果の平均値を示している.

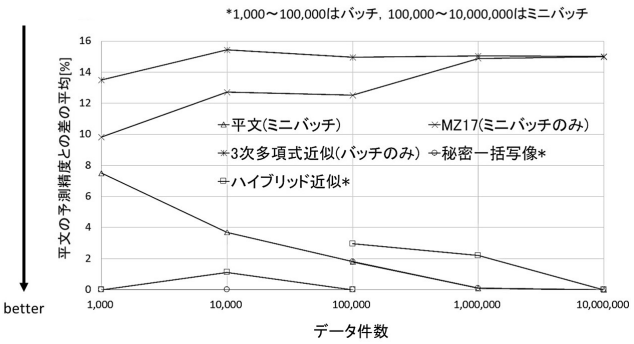

図 4.1 予測精度の誤差

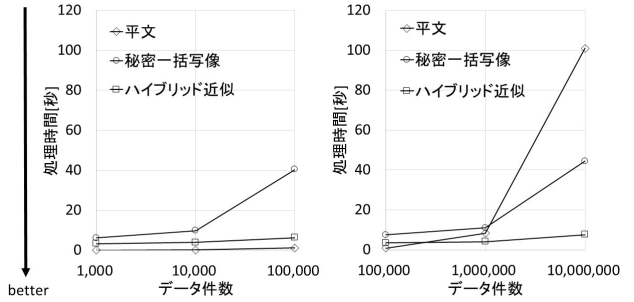

図 4.2 2 つの提案手法と平文の処理時間の比較

図 4.1 から，2 つの提案方式はバッチ法・ミニバッチ法の両方で既存方式よりも平文との誤差が少ない．特に秘密一括写像を用いた手法はグラフが平文と重なっており，誤差がほぼ無いことが分かる．ミニバッチ法で 10 万件を入力した場合は提案方式でも少し誤差が大きくなっているが，これは秘密計算の影響ではなく，ミニバッチ法では学習データが少ないときにバッチ法よりもパラメータ推定の精度が悪くなる場合があるためである．図 4.2 の結果からは，1,000 万件という大きな学習データを入力した場合でも，現実的な処理時間で学習できるということが分かる．特にハイブリッド近似を用いた手法では，わずか 7.5 秒で 1,000 万件の学習を終えることができ，平文の python のプログラムと比較しても非常に高速である.

4.5 実験 2: 既存結果との比較

実験 2 では，SecureML [9] と同様に，784 属性で 10 万および 100 万レコードのデータセットを用意し，処理時間の比較を行った．2 種類の提案方式での結果と，[9] の結果を表 2 に示す．既存手法の処理時間は，トータルの処理時間が明記されていないため，784 属性で 10 万件，100 万件

のデータを入力した際のオンラインの処理時間 (グラフ目視で約 10 秒と 100 秒) と，100 属性で 10 万件のデータを入力した際のオフラインの処理時間 (3.9 秒) に基づいて，処理時間が属性数とデータ数に比例すると仮定して計算した結果の合計時間を示した．レコード数が 10 万の場合は

表 2 [MZ17] との処理時間の比較 [秒]

レコード数	100,000	1,000,000
秘密一括写像	20.61	50.45
ハイブリッド近似	16.13	34.73
MZ17	41	405

差が小さいが，レコード数 100 万の場合は本稿の手法と実装の組み合わせが秘密一括写像では約 8 倍，ハイブリッド近似では約 12 倍と非常に高速である．

4.6 実験 3: 属性数の処理時間への影響

10 万レコードで説明変数の属性数が 1，10，100，1,000 のデータセットを用意し，2 種類の提案方式を用いてミニバッチ法で学習を行った際の処理時間を測定した．図 4.2 に示す結果は各 5 回ずつ測定した平均である．属性数を 10

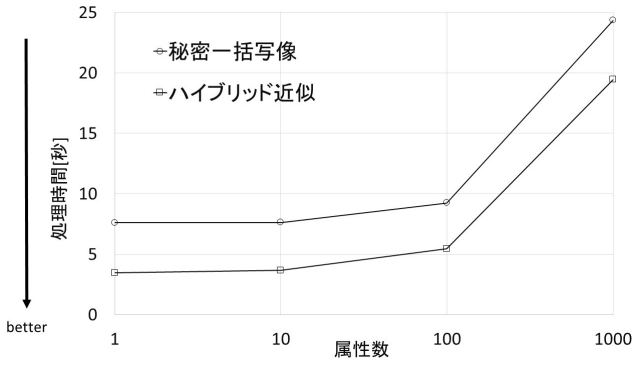

図 4.3 説明変数の属性数を変化させた場合の処理時間の推移

倍に増やしても処理時間の増分がたかだか 4 倍程度と，属性数が処理時間に与える影響が少ないことが分かる．これは，属性数分の積和を行う積和プロトコル (hpsum) の通信量が属性数に依存しないことに起因すると考えられる．

5. おわりに

本稿では秘密分散ベース秘密計算上で，効率と精度を兼ね備えたロジスティック回帰分析手法を 2 種類提案し，実装した．一つ目は秘密一括写像を用いる，精度が特に高い方式である．二つ目は条件分岐と線形近似のハイブリッド近似で性能がより高い方式であり，実装中で必要な基本演算である秘密右シフト演算を含め提案した．性能は，前者は既存方式の約 8 倍，後者は約 12 倍と大きく上回った．予測精度は，前者は平文と比較しても遜色ない精度で，後者はわずかに劣るが既存方式よりも高い精度を発揮した．

参考文献

[1] 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.

[2] Dan Bogdanov, Margus Niitsoo, Tomas Toft, and Jan Willemson. High-performance secure multi-party computation for data mining applications. *Int. J. Inf. Sec.*, Vol. 11, No. 6, pp. 403–418, 2012.

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

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

[5] Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. Efficient logistic regression on large encrypted data.

[6] 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.

[7] Ryo Kikuchi, Dai Ikarashi, Takahiro Matsuda, Koki Hamada, and Koji Chida. Efficient bit-decomposition and modulus-conversion protocols with an honest majority. In *Information Security and Privacy - 23rd Australasian Conference, ACISP 2018, Wollongong, NSW, Australia, July 11-13, 2018, Proceedings*, pp. 64–82, 2018.

[8] Yehuda Lindell and Ariel Nof. A framework for constructing fast MPC over arithmetic circuits with malicious adversaries and an honest-majority. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pp. 259–276, 2017.

[9] 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.

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

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

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

[13] 呉双, 照屋唯紀, 川本淳平, 佐久間淳, 菊池浩明. Privacy-preservation for stochastic gradient descent. 人工知能学会全国大会論文集 2013 年度人工知能学会全国大会 (第 27 回) 論文集, pp. 3L1OS06a3–3L1OS06a3. 一般社団法人 人工知能学会, 2013.

[14] 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.

[15] 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*, pp. 1–8, 2018.

[16] 青野良範, 林卓也, , レ チュウフォン, 王立華. 大規模かつプライバシー保護を可能とするロジスティック解析手法の提案. In *SCIS*, 2016.