Vol.2016-CSEC-73 No.21
Vol.2016-IOT-33 No.21
2016/5/27

秘密分散ベース秘密計算を用いたニューラルネットワークのコスト評価

田中 哲士[1,a)]　山田 真徳[1]　菊池 亮[1]

概要：機械学習が大規模化するにつれて，機械学習のクラウドへの委託が行われるようになってきている．しかしクラウドからデータが漏洩した場合，ユーザのプライバシー情報のみならず，企業の資産とも言える機械学習パラメータも漏洩してしまう危険性がある．本論文ではこの問題を解決するために，ユーザが持つプライバシー情報と企業が持つ機械学習パラメータを秘匿しながら，クラウドに機械学習を委託する方法を提案する．提案方法は秘密分散ベース秘密計算を基に，ニューラルネットワークを用いた学習・予言を可能としている．さらに，提案方式の有効性を測るため，ニューラルネットワークで用いる活性関数としてランプ関数を用いた場合の計算コストを見積り，既存の実装結果と併せることで，機械学習にどの程度の時間がかかるかを推定する．

Cost Evaluation of Neural Network using Secret-sharing-based Secure Computation

Satoshi Tanaka[1,a)]　Masanori Yamada[1]　Ryo Kikuchi[1]

1. 研究背景

ユーザのパーソナルデータを活用して，ユーザの趣味嗜好に合わせたサービスを提供するパーソナライズドサービスは，従来の全ユーザに画一的に提供されるサービスと比較して，ユーザへの訴求効果，ユーザに対する利益の面で優れている．その為，事業者には他の画一的サービスよりも強い商品を提供できるメリット，ユーザには自分にとって最適なサービスを受益できるメリットがあり，事業者・ユーザの双方にとって利益のあるサービスといえる．

パーソナライズドサービスを実現する手段の一つとして機械学習を用いた手法が挙げられる．機械学習でユーザの趣味嗜好を学習し，学習した結果をユーザに適したサービスの判別器として利用する事で実現が可能である．近年の機械学習は大規模化が進んでおり，事業者が自前で用意するサーバだけでは計算が追いつかないため，クラウド事業者に計算を委託し大規模な計算を実現している．

しかし，パーソナライズドサービスの実現における，クラウド事業者による機械学習の委託計算には (1) プライバシーの問題，(2) 学習成果保護の問題という 2 つの問題点が存在する．プライバシーの問題は，機械学習に用いるデータセットに関する問題である．パーソナライズドサービスに関する機械学習では，ユーザの氏名，住所，購買履歴といったパーソナルデータを用いて学習を行う．従って，データを生のまま利用するとクラウド事業者にユーザの重要な情報が漏れる恐れがある．学習成果保護の問題は，学習後の判別器のルールに関する問題である．機械学習には多量のサンプルデータの収集と学習に莫大な時間を要する．その為，機械学習の学習成果は企業にとって重大な秘密情報となりえる．故に，機械学習の学習成果がクラウド事業者に漏えいを防ぐ必要がある．

これらの問題を解決する手段として，秘密計算が考えられる．秘密計算とは，暗号を用いてデータを秘匿したまま分析を行う方法である．秘密計算を用いることで，ユーザのデータおよび企業の学習成果を保護したまま，クラウドに計算を委託することができる．

[1] NTT セキュアプラットフォーム研究所
[a)] tanaka.s@lab.ntt.co.jp

本稿では，秘密分散をベースとした秘密計算を用いて，入力データや学習パラメータを秘匿したまま，機械学習の一方法であるニューラルネットワークを計算する**秘密計算ニューラルネットワーク**を提案する．また，既存の実装結果を用い，提案した秘密計算ニューラルネットワークの計算時間の見積もりを行い，少なくとも XOR の学習であれば現実的な時間で計算可能であることを示す．

2. 準備

ここではまず Shamir の (k,n)-閾値秘密分散，秘密分散ベースの秘密計算，およびニューラルネットワークの基本を紹介する．

2.1 Shamir の (k,n)-閾値秘密分散

データ保持者 DH が自身の持つデータを外部のパーティ $P_1,\ldots,$ に安全に預けたい状況を考える．このとき，あるデータを複数のデータのかけら（シェアと呼ぶ）に分け，それぞれのシェアを別個のパーティに分散することで安全にデータを預ける方式を**秘密分散**と呼ぶ．以下に，Shamir の (k,n)-閾値秘密分散と呼ばれる秘密分散の分散，復元，および処理コストについて述べる．

分散: 今，DH がデータ a を n 人組みのパーティ $\{P_1,\ldots,P_n\}$ に安全に分散したいとする．Shamir の (k,n)-閾値秘密分散では，DH は $a\in K$ となるような体 K を選択し，以下を行い分散する．

(1) $k-1$ 個 $(1<k\leq n)$ のランダムな K の元 $r_1,\ldots,r_{k-1}$ を選択し，a を切片とする $k-1$ 次多項式 $p(x)=\sum_{i=1}^{k-1} r_i x^i + a$ を構成する．
(2) n 個の k の元 $x_1,\ldots,x_n$ を選択し，$p(x_1),\ldots,p(x_n)$ を求める．（但し，任意の $t\neq t'$ について，$x_t\neq x_{t'}$）
(3) $p(x_t)$ を P_t に送信する．

このとき，$p(x_t)$ は P_t における a のシェアであり，$[a]_t := p(x_t)$ と表記する．

復元: DH は P_t に格納している $[a]_t$ から以下を行うことで，元のデータ a を復元することができる．

(1) $\{P_1,\ldots,P_n\}$ からパーティを k 個選択し，$\{P_{t_1},\ldots,P_{t_k}\}$ とする．（但し，任意の $j\neq j'$ について，$t_j\neq t_{j'}$）
(2) 選択した P_{t_j} から $[a]_{t_j}$ を受け取る．
(3) $a=\sum_{j=1}^{k}\lambda_{t_j}[a]_{t_j}$ を計算し，復元する．

このとき，λ_{t_j} は Lagrange 補完法における Lagrange 係数である．

処理コスト: 秘密分散の処理に必要な計算，および通信コストについて述べる．1 回の分散処理は，k 個の r_j の選択と，n 回の $p(x)$ の計算が必要となる．（x_t の選択も必要だが，事前に選択してもよい為，計算コストに含めない）このとき，$p(x)$ は $k-1$ 次多項式である為，x^{k-1} までのべき乗計算と，$k-1$ 回の体乗算, $k-1$ 回の体加算が必要となる．ここで，x^{k-1} は，x^{k-2} と x の積として考えれば，$p(x)$ の計算コストは，$k-1$ 回の体加算，$2k-3$ 回の体乗算である．また，r_j の選択は一回の乱数計算で処理できる為，分散処理の計算コスト $\mathbf{COST}_{\mathrm{share}}(k,n)$ は，

$$\mathbf{COST}_{\mathrm{share}}(k,n) = (k-1)\mathbf{RAN}_K + n(k-1)\mathbf{ADD}_K + n(2k-3)\mathbf{MUL}_K, \quad (1)$$

となる．ここで，$\mathbf{RAN}_K$, $\mathbf{ADD}_K$, $\mathbf{MUL}_K$ はそれぞれ，体 K のランダムな元の生成，体加算，体乗算の計算時間である．一方で，DH は分散時に各 P_t に対して $[a]_t$ を送信するために，1 回の通信を行う．本稿では，各パーティに 1 回の通信を行うような通信コストを 1 ラウンドの通信と呼ぶ．従って，分散における通信コストは 1 ラウンドである．

復元処理では，a の算出が必要となる．λ_{t_i} を事前に計算すれば，a の算出は k 項の積和演算であるため，復元処理の計算コスト $COST_{\mathrm{reconst}}(k,n)$ は，

$$COST_{\mathrm{reconst}}(k,n) = (k-1)\mathbf{ADD}_K + k\mathbf{MUL}_K, \quad (2)$$

となる．また，DH は選択した $P_{t_1},\ldots,P_{t_k}$ とそれぞれ 1 回の通信を行う為，通信コストは 1 ラウンドである．

2.2 秘密分散ベースの秘密計算

秘密計算は，データを何らかの形式で秘匿化し，秘匿化したまま計算する手法である．これにより，データの中身の情報を漏らすことなく，所望の計算を実現できる．本稿では，秘密分散ベースの秘密計算を用いる．これは，データの秘匿化に秘密分散を用いる方式である．以降，本稿では特に記述が無い限り，秘密計算は秘密分散ベースの秘密計算を意味する．ここで，本稿では以下を仮定する．

- 分散時に選択する x_t を事前に DH が用意する．同じ P_t に対して，必ず x_t を用いて計算する．また，$x_1,\ldots,x_n$ は，全てのパーティ（DH, $\{P_1,\ldots,P_n\}$）が共有する．
- $2k-1\leq n$ である．（例えば，$n=5$ なら，$k\leq 3$）

2.2.1 本稿で用いる演算

本稿では，5 つの演算（**加算，定数倍，乗算，m 項の線形和，大小比較**）を基本的な演算として用いる．それぞれの演算について述べる．

加算 $[a]_t$, $[b]_t$ から，$[a+b]_t$ を生成する．秘密計算の加算はシェア同士の加算で実現できる．従って，$[a+b]_t := [a]_t + [b]_t$ である．

定数倍 $[a]_t$ とデータ $c\in K$ から，$[ca]_t$ を生成する．このとき，c は全てのパーティ P_t で共有されている値である．秘密計算の定数倍も単純に 2 つの値の積をとればよく，$[ca]_t := c\times[a]_t$ である．

乗算 $[a]_t$, $[b]_t$ から，$[ab]_t$ を生成する．秘密計算の乗算は単純には実現できない，これは，2 つのシェアの積を

$[\![d]\!]_t := [a]_t \times [b]_t$ としたとき，$[\![d]\!]_t$ は $2(k-1)$ 次多項式上の点となってしまい，本来の多項式から次数が上がってしまうからである．そこで，各パーティ P_t は以下を行い，$[ab]_t$ を取得する．

(1) P_t は Shamir の (k,n)-閾値秘密分散の分散を用いて，$[\![d]\!]_t$ のシェア $[[\![d]\!]_t]_{t'}$ を求め，$P_{t'}$ に送信する．
(2) P_t は受け取った $[[\![d]\!]_{t'}]_t$ から，Shamir の $(2k-1,n)$-閾値秘密分散の復元を用いて，$[ab]_t$ を取得する．

線形和 $[a_1]_t,\ldots,[a_m]_t$，$[b_1]_t,\ldots,[b_m]_t$ から $\left[\sum_{j=1}^m (a_j b_j)\right]_t$ を生成する．単純には $m-1$ 回の秘密計算加算，m 回の秘密計算乗算で実現できる．しかし，以下の手法で計算コスト，および通信コスト双方を減らすことができる．

(1) P_t は $[\![d_j]\!]_t := [a_j]_t \times [b_j]_t$ を全ての j について求め，$\sum_{j=1}^m [\![d_j]\!]_t$ を計算する．
(2) P_t は Shamir の (k,n)-閾値秘密分散の分散を用いて，$\sum_{j=1}^m [\![d_j]\!]_t$ を計算し，シェア $[\sum_{j=1}^m [\![d_j]\!]_t]_{t'}$ を求め,$P_{t'}$ に送信する．
(3) P_t は受け取った $[\sum_{j=1}^m [\![d_j]\!]_t]_{t'}$ から，Shamir の $(2k-1,n)$-閾値秘密分散の復元を用いて，$\left[\sum_{j=1}^m (a_j b_j)\right]_t$ を取得する．

定数との大小比較 $[a]_t$, c_{bit} から a と c の大きさを比較し，$a<c$ ならば 1 の，$a \geq c$ ならば 0 のシェアとなる $[a \underset{?}{<} c]_t$ を生成する．ここで，c_{bit} は c のビット列表現である．($[a \underset{?}{>} c]_t$ も同様に考える）秘密計算で数値の大小比較を直接行うのは困難である為，$[a]_t$ を数値のシェアからビット列 a_{bit} のシェアに変換し，論理回路を構成して比較演算を行う．この数値のシェアからビット列のシェアに変換する演算をビット分解と呼ぶ．ビット分解には Damgård らの方式 [1] や五十嵐らの方式 [2] が知られている．

2.2.2 処理コスト

それぞれの演算における処理コストについて述べる．

加算，定数倍 秘密計算の加算，定数倍は単純な体加算，体乗算で実現できる．従って，加算の計算コストは $1\mathbf{ADD}_K$，定数倍の計算コストは $1\mathbf{MUL}_K$ である．以降，**ADD** を秘密計算加算，**CON** を秘密計算定数倍の計算コストとする．即ち，

$$1\mathbf{ADD} := 1\mathbf{ADD}_K, \tag{3}$$

$$1\mathbf{CON} := 1\mathbf{MUL}_K, \tag{4}$$

となる．また，P_t のローカルで計算が完了する為，通信コストは発生しない．

乗算 秘密計算の乗算では，2 つのシェアの積だけでなく，(k,n)-閾値秘密分散の分散処理と $(2k-1,n)$-閾値秘密分散の復元処理が必要となる．但し，x_t は事前に DH が決定しているため，ランダム元の生成は行わない．従って，乗算の計算コスト $\mathbf{COST}_{\mathrm{mul}}$ は，

$$\begin{aligned}\mathbf{COST}_{\mathrm{mul}} = {} & 1\mathbf{CON} + 1\mathbf{COST}_{\mathrm{share}}(k,n) \\ & + 1\mathbf{COST}_{\mathrm{reconst}}(2k-1,n),\end{aligned} \tag{5}$$

となる．以降，$1\mathbf{MUL} := 1\mathbf{COST}_{\mathrm{mul}}$ を秘密計算乗算の計算コストとする．また，シェアのやり取りの為，1 ラウンドの通信が必要となる．

線形和 秘密計算線形和では乗算と同様に 1 回の分散と 1 回の復元が必要となる．また，m 項の線形和には $m-1$ 回の体加算と，m 回の体乗算が必要となる．従って，線形和の計算コスト $\mathbf{COST}_{\mathrm{linear}}$ は，

$$\mathbf{COST}_{\mathrm{linear}} = (m-1)\mathbf{ADD} + (m-1)\mathbf{CON} + 1\mathbf{MUL}, \tag{6}$$

となり，1 ラウンドの通信が必要となる．

定数の大小比較 本稿では，Damgård らの方式 [1] を用いる．Damgård らの方式では，1 回のビット分解に 114 ラウンドの通信と $(110\ell\log\ell + 118\ell)\mathbf{MUL}$ の計算コストが必要となる．ここで，$\ell := \lceil \log |K| \rceil$ である．また，Damgård らの論理回路による比較関数 **BIT-LT** [1] は 19 ラウンドの通信と，$22\ell\mathbf{MUL}$ の計算コストが必要となる．従って，秘密計算定数の大小比較は 133 ラウンドの通信と，

$$\mathbf{COST}_{\mathrm{comp}} = (110\ell\log\ell + 140\ell)\mathbf{MUL}, \tag{7}$$

の計算が必要である．

2.2.3 想定する攻撃者

本論文では，semi-honest かつ k 人未満の攻撃者を想定する．すなわち，攻撃者はプロトコルには従い，攻撃者の数は全体の半数未満であり，攻撃者のみでの復元ができないことを意味する．

2.3 ニューラルネットワーク

ニューラルネットワークは，人間の脳の活動をモデル化し，再現する機械学習の一種である．

2.3.1 モデルと構造

ニューラルネットワークの学習モデルを図 1 に示す．一般に，ニューラルネットワークは重みつきの二部グラフを連結した構造を持つ．この二部グラフの片側のノードの集まりを層と呼ぶ．通常，ニューラルネットワークは 3 段以上の層で構成されている．本稿では層の数を H とする．この内，データを入力する層を入力層 L_1，ニューラルネットワークによる計算（予言と呼ぶ）の結果が出力される層を出力層 L_H と呼び，それ以外の間に有る層 $L_2,\ldots,L_{H-1}$ を中間層と呼ぶ．各 L_l にはそれぞれ，M_l 個のノード $\{\mathrm{node}_1^{(l)},\ldots,\mathrm{node}_{M_l}^{(l)}\}$ が存在する．更に，入力層を除く全ての $\mathrm{node}_j^{(l)}$ は，閾値 $\theta_j^{(l)}$ を所持する．また，隣接する 2 つの層のノード $\mathrm{node}_i^{(l)}$, $\mathrm{node}_j^{(l+1)}$ の間には重み $w_{(i,j)}^{(l)}$ のエッジが存在する．

2.3.2 学習フェーズ

ニューラルネットワークの学習について述べる．学習は

Vol.2016-CSEC-73 No.21
Vol.2016-IOT-33 No.21
2016/5/27

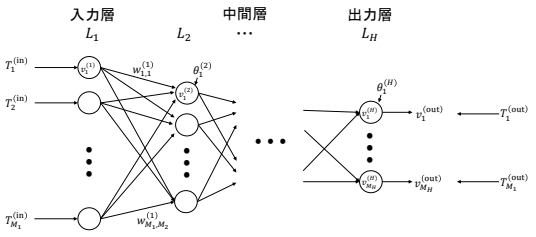

図 1 ニューラルネットワークの学習モデル

D 個の教師データ $T^{(1)}, \ldots, T^{(D)}$ を用いる．このとき，$1 \leq d \leq D$ について，$T^{(d)} := \{T^{(d,\mathrm{in})}, T^{(d,\mathrm{in})}\}$ であり，$T^{(d,\mathrm{in})} := \{T_1^{(d,\mathrm{in})}, \ldots, T_{M_1}^{(d,\mathrm{in})}\}$, $T^{(d,\mathrm{out})} := \{T_1^{(d,\mathrm{out})}, \ldots, T_{M_H}^{(d,\mathrm{out})}\}$ である．

まず，ニューラルネットワークは，$T^{(d,\mathrm{in})}$ を入力層に入れ，**前方伝播**を行い，出力値を計算する．その後，出力値と $T^{(d,\mathrm{out})}$ から定義した誤差が小さくなるように**誤差逆伝播**により，重み，閾値を修正する．

前方伝播は入力層の値を基に，繰り返し現在層の値を次層の値に伝播させることで出力値を求める処理である．今，$T^{(d,\mathrm{in})}$ を入力したときの，$\mathrm{node}_j^{(l)}$ が持つ値を $v_j^{(d,l)}$ とし，前方伝播により $\mathrm{node}_j^{(H)}$ から出力される出力値を $v_j^{(d,\mathrm{out})}$ とする．このとき，前方伝播は以下のステップで実行される．

(1) 全ての j について，$v_j^{(d,1)}$ に $T_j^{(d,\mathrm{in})}$ を代入する．
(2) $l = 2$ から $l = H$ まで，以下の 2 ステップを実行する．
(3) 全ての j について，$X_j^{(d,l)} := \sum_{i=1}^{M_{l-1}} w_{i,j}^{(l-1)} v_i^{(d,l-1)} + \theta_j^{(l)}$ を計算する．
(4) 全ての j について，$v_j^{(l)} \leftarrow f(X_j^{(d,l)})$ を計算する．
(5) 全ての j について，$v_j^{(d,\mathrm{out})} = g_j(v_1^{(d,H)}, \ldots, v_{M_H}^{(d,H)})$ を計算する．

ここで，$f(x)$ は中間層の活性関数と呼ばれ，正規化線形関数（ランプ関数）やシグモイド関数等が利用される．また，$g_j(x_1, \ldots, x_{M_H})$ は出力層の活性関数と呼ばれ，問題によって恒等写像，soft-max 関数等が利用される．通常，$f(x)$ と $g_j(x_1, \ldots, x_{M_H})$ は分けて設計される．

誤差逆伝播は，出力値と教師信号の誤差を前の層に伝播させ，誤差が最小になるように重み，閾値を修正する処理である．誤差逆伝播は以下のステップで実行される．

(1) 全ての d, j について，出力層の修正率 $\delta_j^{(d,H-1)} := \frac{\partial E}{\partial v_j^{(d,H)}} f'(X_j^{(d,H)})$ を求める．
(2) $l = H - 2$ から $l = 1$ まで，全ての d, j について $\delta_i^{(d,l)} := \sum_{j=1}^{L_M} (\delta_j^{(d,l+1)} w_{i,j}^{(l)}) f'(X_i^{(d,l)})$ を求める．
(3) 全ての i, j, l について，$w_{i,j}^{(l)} := w_{i,j}^{(l)} - \Delta w_{i,j}^{(l)} \sum_{d=1}^{D} v_i^{(d,l)} \delta_j^{(d,l)}$ を計算する．
(4) 全ての j, l について，$\theta_j^{(l)} := \theta_j^{(l)} - \Delta\theta_j^{(l)} \sum_{d=1}^{D} \delta_j^{(d,l)}$ を計算する．

ここで E は出力値と教師信号からなる誤差関数である．E は選択する $g(x)$ によって決まるが，$g(x)$ が恒等写像，soft-max 関数の場合，通常，E の $v_j^{(H)}$ による偏微分は，$\frac{\partial E}{\partial v_j^{(d,H)}} := v_j^{(d,\mathrm{out})} - t_j^{(\mathrm{out})}$ となる．また，$X_j^{(l)}$ は前方伝播で計算したものを用いる．このとき，$\Delta w_{i,j}^{(l)}$, $\Delta\theta_j^{(l)}$ はそれぞれ，$w_{i,j}^{(l)}$, $\theta_j^{(l)}$ に対する学習率である．

本稿は，ミニバッチ確率的勾配降下法を用いる．これは，1 回の学習で教師データを D 個全て利用するのではなく，教師データの内，D' 個（$D' \leq D$）をランダムに選択し，選択した $T^{\sigma_1}, \ldots, T^{\sigma_{D'}}$（$\sigma_d \in \{1, \ldots, D\}$）に対して前方伝播，誤差逆伝播を行い学習する方式である．このとき，D' をバッチサイズと呼ぶ．

2.3.3 予言フェーズ

予言フェーズは，学習後のニューラルネットワークを活用するフェーズである．学習したニューラルネットワークに対してデータ入力を行い，出力層から結果を受け取る．計算としては，学習における前方伝播と同等なため，前方伝播を用いてそのまま実現できる．

3. 秘密計算ニューラルネットワーク

本節では，秘密計算を適用したニューラルネットワークの手法について述べる．以降の秘密計算のパーティを以下のように定義する．

C ：事業者，機械学習を委託する．
U ：ユーザ，自身のデータを提供し，C の持つ学習結果に基づく予言を得る．
P_t ：t 番目（$1 \leq t \leq n$）の分散パーティ，秘密計算を行う．

提案する秘密計算ニューラルネットワークの学習プロトコルについて，プロトコル 1 に示す．それぞれのステップの詳細については，次節以降に示す．

プロトコル 1 秘密計算ニューラルネットワークの学習

Require: パーティ: $C, \{P_1, \ldots, P_n\}$
データ： $C \leftarrow w_{i,j}^{(l)}, \theta_j^{(l)}, T^{(d)} := \{T^{(d,\mathrm{in})}, T^{(d,\mathrm{out})}\}$, $\Delta w_{i,j}^{(l)}, \Delta\theta_j^{(l)}, D'$, 学習回数 TIME_{ite}.
Ensure: $P_t \leftarrow$ 学習した重みのシェア $[w_{i,j}^{(l)}]_t$, 学習した閾値のシェア $[\theta_j^{(l)}]_t$.
1: C は全パーティ $P_1, \ldots, P_n$ に対して**初期設定**を行う．
2: **for** $c = 1$ to TIME_{ite} **do**
3: P_1 はランダムに D' 個の $\sigma_d \in \{1, \ldots, D\}$ を選択し，全ての P_t に $\sigma_1, \ldots, \sigma_{D'}$ を送信する．
4: 全 P_t は $1 \leq d \leq D'$ について，プロトコル 2 **前方伝播**を行い，$[T^{(\sigma_d,\mathrm{in})}]_t$ から $[v_1^{(\sigma_d,\mathrm{out})}]_t, \ldots, [v_{M_H}^{(\sigma_d,\mathrm{out})}]_t$ を取得する．
5: 全 P_t はプロトコル 3 **後方伝播**を行い，全ての $[v_j^{(\sigma_d,\mathrm{out})}]_t$, $[T^{(\sigma_d,\mathrm{out})}]_t$ から $[w_{i,j}^{(l)}]_t$, $[\theta_j^{(l)}]_t$ を更新する．
6: **end for**

3.1 初期設定

ニューラルネットワークの初期設定では，C が持つ学

習に必要なパラメータ（$w_{i,j}^{(l)}$, $\theta_j^{(l)}$, $T^{(1)},\ldots,T^{(D)}$, $\Delta w_{i,j}^{(l)}$, $\Delta\theta_j^{(l)}$, バッチサイズ D', 学習回数 TIME_{ite}）を各 P_t に送信する．このとき，C は $w_{i,j}^{(l)}$, $\theta_j^{(l)}$, $T^{(d)}$ を秘密分散し，それぞれのシェアを送る．一方，$\Delta w_{i,j}^{(l)}$, $\Delta\theta_j^{(l)}$, D, TIME_{ite} については元のデータのまま送信する．

3.2 前方伝播

秘密計算ニューラルネットワークにおける前方伝播をプロトコル 2 に示す．尚，本稿では，ニューラルネットワークの活性関数 $f(x)$ として正規化線形関数，

$$\mathrm{relu}(x) := \begin{cases} x & (x>0), \\ 0 & (x\le 0), \end{cases} \tag{8}$$

を採用する．これは，知られている活性関数の中でも秘密計算で効率的に実装可能で，かつ学習の効率が良いことが知られているためである．ここで，$x>0$ の判定を，$(x \underset{?}{>} 0)$ とすれば，$\mathrm{relu}(x) = x(x \underset{?}{>} 0)$ である．従って，秘密計算 $\mathrm{relu}(x)$ は以下のステップで実現可能である．

(1) 全 P_t は**秘密計算により**，$[x \underset{?}{>} 0]_t$ を求める．
(2) 全 P_t は**秘密計算により**，$[\mathrm{relu}(x)]_t := [x(x \underset{?}{>} 0)]_t$ を求める．

また，$g_j(x_1,\ldots,x_{M_H}) := x_j$ とする．従って，$[g_j(x_1,\ldots,x_{M_H})]_t$ の計算は単純に，$[x_j]_t$ を代入すればよい．

3.3 誤差逆伝播

秘密計算ニューラルネットワークにおける後方伝播をプロトコル 3 に示す．ニューラルネットワークの後方伝播では，誤差関数 E に基づき，変化率 $\delta_j^{(l)}$ の計算，重み $w_{i,j}^{(l)}$, 及び閾値 $\theta_i^{(l)}$ の更新を行う．本稿では，E に最小二乗誤差 $E = \frac{1}{2}\sum_{d=1}^{D'}\sum_{j=1}^{M_H}(v_j^{(\sigma_d,\mathrm{out})} - T_j^{(\sigma_d,\mathrm{out})})^2$ を用いる．また，節 3.2 で示したように本稿では，$f(x) = \mathrm{relu}(x)$, $g(x) = x$ である．従って，$f'(x) = (x \underset{?}{>} 0)$ であり，秘密計算では 1 回の大小比較で実現でき，

$$[\delta_i^{(\sigma_d,H-1)}]_t = [(X_j^{(\sigma_d,H)} \underset{?}{>} 0)(v_j^{(\sigma_d,\mathrm{out})} - T_j^{(\sigma_d,\mathrm{out})})]_t, \tag{9}$$

$$[\delta_i^{(\sigma_d,l)}]_t = [(X_j^{(\sigma_d,l)} \underset{?}{>} 0)\sum_{j=1}^{M_{l+1}}\delta_j^{(\sigma_d,l+1)}w_{i,j}^{(l)}]_t, \tag{10}$$

となる（但し，$1\le l\le H-2$）．

プロトコル 2 前方伝播

Require: パーティ: 分散パーティ $\{P_1,\ldots,P_n\}$
データ: $P_t \leftarrow [w_{i,j}^{(l)}]_t$, $[\theta_j^{(l)}]_t$, $[T^{(\sigma_d,\mathrm{in})}]_t := \{[T_1^{(\sigma_d,\mathrm{in})}]_t,\ldots,[T_{M_1}^{(\sigma_t,\mathrm{in})}]_t\}$, σ_t.
Ensure: $P_t \leftarrow [v_j^{(\sigma_d,\mathrm{out})}]_t$, $[v_j^{(\sigma_d,l)}]_t$, $[X_j^{(\sigma_d,l)}]_t$
1: 全 P_t は全ての i について，$[v_1^{(1)}]_t \leftarrow T^{(\mathrm{in})}]_t$ を行う．
2: **for** $l=2$ to H **do**
3: 全 P_t は全ての j について，**秘密計算により** $[X_j^{(\sigma_d,l)}]_t := [\sum_{i=1}^{M_{l-1}} w_{i,j}^{(l-1)}v_i^{(\sigma_d,l-1)} + \theta_j^{(l)}]_t$ を求める．
4: 全 P_t は全ての j について，**秘密計算により** $[v_j^{(\sigma_d,l)}]_t \leftarrow [f(X_j^{(\sigma_d,l)})]_t$ を求める．
5: **end for**
6: 全 P_t は全ての j について，**秘密計算により** $[v_j^{(\sigma_d,l)}]_t \leftarrow [g_j(v_1^{(\sigma_d,H)},\ldots,v_{M_H}^{(\sigma_d,H)})]_t$ を求める．

プロトコル 3 誤差逆伝播

Require: パーティ: $\{P_1,\ldots,P_n\}$,
データ: $P_t \leftarrow [w_{i,j}^{(l)}]_t$, $[\theta_j^{(l)}]_t$, $[v_j^{(\mathrm{out})}]_t$, $[T_j^{(\sigma_d,\mathrm{out})}]_t$, $[v_j^{(\sigma_d,l)}]_t$, $[X_j^{(\sigma_d,l)}]_t$, $\Delta w_{i,j}^{(l-1)}$, $\Delta\theta_{i,j}^{(l)}$.
Ensure: $P_t \leftarrow$ 更新した重みのシェア $[w_{i,j}^{(l)}]_t$,
更新した閾値のシェア $[\theta_j^{(l)}]_t$.
1: 全 P_t は，全ての d,j について，**秘密計算により** $[\delta_j^{(\sigma_d,H)}]_t \leftarrow [f'(X_j^{(\sigma_d,H)})(v_j^{(\sigma_d,\mathrm{out})} - T_j^{(\sigma_d,\mathrm{out})})]_t$ を求める．
2: **for** $l=H-1$ to 2 **do**
3: 全 P_t は，全ての d,j について，**秘密計算により** $[TEMP_i^{(\sigma_d)}]_t \leftarrow [\sum_{j=1}^{M_{l+1}}\delta_j^{(\sigma_d,l+1)}w_{i,j}^{(l+1)}]_t$ を求める．
4: 全 P_t は，全ての d,j について，**秘密計算により** $[\delta_i^{(\sigma_d,l)}]_t \leftarrow [f'(X_i^{(\sigma_d,l)})\,TEMP_i^{(\sigma_d)}]_t$ を求める．
5: **end for**
6: **for** $l=H$ to 2 **do**
7: 全 P_t は，全ての i,j について，**秘密計算により** $[w_{i,j}^{(l-1)}]_t \leftarrow [w_{i,j}^{(l-1)} - \Delta w_{i,j}^{(l-1)}\sum_{d=1}^{D'}\delta_j^{(\sigma_d,l)}v_i^{(\sigma_d,l-1)}]_t$ を求める．
8: 全 P_t は，全ての i,j について，**秘密計算により** $[\theta_j^{(l)}]_t \leftarrow [\theta_j^{(l)} - \Delta\theta_j^{(l)}\sum_{d=1}^{D'}\delta_j^{(\sigma_d,l)}]_t$ を求める．
9: **end for**

3.4 予言フェーズ

秘密計算ニューラルネットワークにおける予言フェーズをプロトコル 4 に示す．予言フェーズの計算は，学習フェーズにおける前方伝播を利用することで実現可能である．予言フェーズが学習フェーズと異なるのは，事業者 C がデータの入力を行うのではなく，ユーザ U が行う点である．同様に，出力結果も U が受け取り，復元する．

プロトコル 4 予言フェーズ

Require: パーティ: U, $\{P_1,\ldots,P_n\}$
データ: $U \leftarrow$ データ $d_1,\ldots,d_{M_1}$, $P_t \leftarrow w_{i,j}^{(l)}$, $\theta_j^{(l)}$.
Ensure: $U \leftarrow$ 予言結果 $v_1^{(\mathrm{out})},\ldots,v_{M_H}^{(\mathrm{out})}$.
1: U は，$1\le i\le M_1$ について，d_i を Shamir の (k,n)-閾値秘密分散で分散し，全 P_t に対応するシェア $[d_i]_t$ を送信.
2: 各 P_t は, プロトコル 2 により**前方伝播**を行い，$[d_1]_t,\ldots,[d_{M_1}]_t$ から $[v_1^{(\mathrm{out})}]_t,\ldots,[o_{M_H}^{(\mathrm{out})}]_t$ を求め，U に送信.
3: U は，全ての $[v_j^{(\mathrm{out})}]_t$ から Shamir の (k,n)-閾値秘密分散の復元を行い，元の $v_1^{(\mathrm{out})},\ldots,v_{M_H}^{(\mathrm{out})}$ を得る．

4. 秘密計算ニューラルネットワークの計算コスト

本節では，Shamir の (k,n)-閾値秘密分散をベースとした秘密計算ニューラルネットワークの計算コストについて

示す．はじめに，個々のコスト評価を簡単にするために，ニューラルネットワーク内の全体のノード数 $N_{\rm node}$，及びエッジの数 $N_{\rm edge}$, 教師データのサイズ S_T について，以下に定義する．

$$\begin{aligned} N_{\rm node} &:= \sum_{l=1}^{H} M_l, \\ N_{\rm edge} &:= \sum_{l=1}^{H-1} M_l M_{l+1}, \\ S_T &:= M_1 + M_H. \end{aligned}$$

このとき，$f(x) := \mathrm{relu}(x)$, $g(x_1, \ldots, x_{M_H}) := x$ とすると，秘密計算ニューラルネットワークの学習全体の計算コスト $\mathbf{COST}_{\rm NN}$ と通信ラウンド数 $\mathbf{ROUND}_{\rm NN}$ を以下の式に示す．

$$\begin{aligned} &\mathbf{COST}_{\rm NN} \\ &= \mathbf{COST}_{\rm init} + \mathbf{TIME}_{\rm ite}\mathbf{COST}_{\rm learn} \\ &= \Big\{\mathbf{TIME}_{\rm ite} D'(3N_{\rm edge} + 2M_H - M_1M_2) \\ &\quad + n(k-1)(N_{\rm node} + N_{\rm edge} + DS_T - M_1)\Big\}\mathbf{ADD} \\ &\quad + \Big\{\mathbf{TIME}_{\rm ite}\big(N_{\rm node} - M_1 \\ &\quad + D'(3N_{\rm edge} - 2N_{\rm node} + 2M_1 + M_H - M_1M_2)\big) \\ &\quad + n(2k-3)(N_{\rm node} + N_{\rm edge} + DS_T - M_1)\Big\}\mathbf{CON} \\ &\quad + \mathbf{TIME}_{\rm ite}\Big\{N_{\rm edge} + D'\big(4N_{\rm node} - 4M_1 - M_H \\ &\quad + 2(N_{\rm node} - M_1)(110\ell\log\ell + 140\ell)\big)\Big\}\mathbf{MUL} \\ &\quad + \mathbf{TIME}_{\rm ite} D'\mathbf{RAN}_D. \end{aligned} \tag{11}$$

$$\begin{aligned} &\mathbf{ROUND}_{NN} \\ &= \mathbf{ROUND}_{\rm init} + \mathbf{TIME}_{\rm ite}\mathbf{ROUND}_{\rm learn} \\ &= \mathbf{TIME}_{\rm ite} N_{edget} + \{D'(270N_{node} - 270M_1 - M_H + 1)\} \\ &\quad + 2N_{\rm edge} + 2N_{\rm node} + DS_T - M_1 + 2. \end{aligned} \tag{12}$$

但し，$\mathbf{TIME}_{\rm ite}$ は学習の繰り返し回数，$\mathbf{RAN}_D$ は乱数 $\sigma_d \in \{1, \ldots, D\}$ の選択である．また，$\mathbf{COST}_{\rm init}$ はニューラルネットワークの初期化，$\mathbf{COST}_{\rm learn}$ は 1 回の学習にかかる計算コストであり，$\mathbf{ROUND}_{\rm init}$, $\mathbf{ROUND}_{\rm learn}$ はそれぞれに対する通信コスト，$\mathbf{TIME}_{\rm ite}$ は学習の繰り返し回数である．それぞれの計算コスト，通信コストについては後述する．

4.1 初期設定

初期設定で, $P_1, \ldots, P_n$ に送信するデータは, 重み $w_{i,j}^{(l)}$, 閾値 $\theta_j^{(l)}$, 教師データ $T^{(d)}$, 学習率 $\Delta w_{i,j}^{(l)}$, $\Delta\theta_j^{(l)}$, バッチサイズ D', 学習回数 $\mathbf{TIME}_{\rm ite}$ の 7 種である．このとき，$w_{i,j}^{(l)}$, $\theta_j^{(l)}$, $T^{(d)}$ は秘密分散により分散し，シェアの形式で送信する．その為，これらは 1 つに付き $n(k-1)\mathbf{ADD} + n(2k-3)\mathbf{CON}$ の計算が必要となる．また，$w_{i,j}^{(l)}$, $\Delta w_{i,j}^{(l)}$ は各エッジに用意するため，それぞれ N_{edge} 個，$\theta_j^{(l)}$, $\Delta\theta_j^{(l)}$ は L_1 のノードを除く全てのノードに用意するため，それぞれ $N_{node} - M_1$ 個，$T^{(d)}$ は 1 つの教師データに付き，S_T 個のデータが必要となる．従って，初期設定における計算コスト $\mathbf{COST}_{\rm init}$, 通信コスト $\mathbf{ROUND}_{\rm init}$ は以下の式で表現される．

$$\begin{aligned} &\mathbf{COST}_{\rm init} \\ &= n(k-1)(N_{\rm edge} + N_{\rm node} + DS_T - M_1)\mathbf{ADD} \\ &\quad + n(2k-3)(N_{\rm edge} + N_{\rm node} + DS_T - M_1)\mathbf{CON} \end{aligned} \tag{13}$$

$$\mathbf{ROUND}_{\rm init} = 2N_{edge} + 2N_{node} + DS_T - 2M_1 + 2. \tag{14}$$

4.2 学習

1 回の学習では，D' 個の乱数 σ_d 選択，D' 回の前方伝播，及び 1 回の誤差逆伝播が必要となる．従って，後述する式 16, 及び式 22 より，1 回の学習で必要な計算コスト $\mathbf{COST}_{\rm learn}$ は，

$$\begin{aligned} &\mathbf{COST}_{\rm learn} \\ &= D'\mathbf{RAN}_D + D'\mathbf{COST}_{\rm forward} + \mathbf{COST}_{\rm back} \end{aligned} \tag{15}$$

となる．同様に，通信コスト $\mathrm{ROUND}_{\rm learn}$ は，乱数 σ_d の共有に，P_1 から各 P_t に対して D 回の通信が必要なため，式 17, 及び式 23 より，以下の式となる．

$$\mathbf{ROUND}_{\rm learn} = D' + D'\mathbf{ROUND}_{forward} + \mathbf{ROUND}_{back}.$$

4.3 前方伝播

前方伝播は，以下の 3 つの計算で構成される．

- $w_{i,j}^{(l-1)}$ と $v_i^{(\sigma_d,l-1)}$, 及び $\theta_j^{(l)}$ の線形和: $[X_j^{(\sigma_d,l)}]_t := [\sum_{i=1}^{M_{l-1}} w_{i,j}^{(l-1)} v_i^{(\sigma_d,l-1)} + \theta_j^{(l)}]_t$.
- 中間層の活性関数: $[f(X_j^{(\sigma_d,l-1)})]_t$.
- 出力層の活性関数: $[g_j(v_1^{(\sigma_d,H)}, \ldots, v_{M_H}^{(\sigma_d,H)})]_t$.

それぞれの計算について考える．但し，$[g_j(v_1^{(\sigma_d,H)}, \ldots, v_{M_H}^{(\sigma_d,H)})]_t := [v_j^{(\sigma_d,H)}]_t$ であるため，出力層の活性関数では計算が発生しない．

$X_j^{(\sigma_d,l)}$ は M_{l-1} 項の線形和と 1 回の加算で構成される．式 6 より，$X_j^{(\sigma_d,l)}$ の計算コストは，

$$M_{l-1}\mathbf{ADD} + (M_{l-1} - 1)\mathbf{CON} + 1\mathbf{MUL},$$

であり，通信コストは 1 ラウンドである．

$[f(X_j^{(\sigma_d,l-1)})]_t := [\mathrm{relu}(f(X_j^{(\sigma_d,l-1)})]_t$ は 1 回の大小比較と 1 回の乗算で構成される．式 7 より，$[f(X_j^{(\sigma_d,l-1)})]_t$ の計算コストは，

$$(110\ell\log\ell + 118\ell + 1)\mathbf{MUL},$$

であり，通信コストは 134 ラウンドとなる．

前方伝播の計算は，入力層以外の全ノードで，$X_j^{(\sigma_d,l)}$, $[f(X_j^{(\sigma_d,l-1)})]_t$ を計算し，出力層の全ノードで，

$[g_j(v_1^{(\sigma_d,H)},\ldots,v_{M_H}^{(\sigma_d,H)})]_t$ を求める．従って，$N_{\rm node}-M_1$ 回の計算を行う．ゆえに，前方伝播における計算コスト $\mathbf{COST}_{\rm forward}$ は，

$$\begin{aligned}&\mathbf{COST}_{\rm forward}\\ =\ & N_{\rm edge}\mathbf{ADD}+(N_{\rm edge}-N_{\rm node}+M_1)\mathbf{CON}\\ &+(N_{\rm node}-M_1)(110\ell\log\ell+118\ell+2)\mathbf{MUL},\end{aligned}\quad(16)$$

であり，通信コスト $\mathbf{ROUND}_{\rm forward}$ は，

$$\mathbf{ROUND}_{\rm forward}=135(N_{\rm node}-M_1),\quad(17)$$

となる．

4.4 誤差逆伝播

誤差逆伝播は以下の 4 つの計算によって構成される．

- $l=H$ における，
  $[\delta_j^{(\sigma_d,H)}]_t:=[f'(X_j^{(\sigma_d,H)})v_j^{(\sigma_d,\rm out)}-T_j^{(\sigma_d,\rm out)}]_t$ の計算．
- $1<l<H$ における，
  $[\delta_j^{(\sigma_d,l)}]_t:=[f'(X_j^{(\sigma_d,l)})\sum_{j=1}^{M_{l+1}}w_{i,j}^{(l)}\delta_j^{(\sigma_d,l+1)}]_t$ の計算．
- $w_{i,j}^{(l)}$ の更新:
  $[w_{i,j}^{(l)}]_t\leftarrow[w_{i,j}^{(l)}-\Delta w_{i,j}^{(l)}\sum_{d=1}^{D'}v_i^{(\sigma_d,l)}\delta_j^{(\sigma_d,l+1)}]_t$.
- $\theta_j^{(l)}$ の更新: $[\theta_j^{(l)}]_t\leftarrow[\theta_j^{(l)}-\Delta\theta_j^{(l)}\sum_{d=1}^{D'}\delta_j^{(\sigma_d,l)}]_t$.

それぞれの計算について考える．

$[\delta_j^{(\sigma_d,H)}]_t$ は，1 回の加算と $[f'(X_j^{(\sigma_d,H)})]_t$ の計算，及び 1 回の乗算で計算できる．$[X_j^{(\sigma_d,H)}]_t$ の算出は前方伝播で行っているため，計算は発生しない．また，この計算は，出力層のノードのみで行うため，1 つの教師データに対して，M_H 回行う．実際には，誤差逆伝播において D' 個の教師データを用いるため，計算コストは

$$D'M_H\mathbf{ADD}+D'M_H(110\ell\log\ell+118\ell+1)\mathbf{MUL}.\quad(18)$$

である．また，$134D'M_H$ ラウンドの通信コストが必要となる．

一方，$1<l<H$ における，$[\delta_i^{(\sigma_d,l)}]_t$ は，次層の変化率 $\delta_j^{(\sigma_d,l+1)}$ と $w_{i,j}^{(l)}$ による M_{l+1} 項の線形和と $[f'(X_j^{(\sigma_d,l)})]_t$ の計算，及び 1 回の乗算で計算できる．この計算は，全ての中間層のノードで行う．また，教師データは D' 個用いるため，計算コストは，

$$\begin{aligned}&D'(N_{\rm edge}-N_{\rm node}+S_T-M_1M_2)\mathbf{ADD}\\ &+D'(N_{\rm edge}-N_{\rm node}+S_T-M_1M_2)\mathbf{CON}\\ &+D'(N_{\rm node}-S_T)(110\ell\log\ell+118\ell+2)\mathbf{MUL}.\end{aligned}\quad(19)$$

である．また，$135D'(N_{\rm node}-S_T)$ ラウンドの通信コストが必要となる．

$w_{i,j}^{(l)}$ の更新は $v_i^{(\sigma_d,l+1)}$ と $[\delta_i^{(\sigma_d,l)}]_t$ による D' 項の線形和と，1 回の加算，及び 1 回の定数倍で構成される．この計算は，各エッジに対して 1 回行わため，計算コストは，

$$D'N_{\rm edge}\mathbf{ADD}+D'N_{\rm edge}\mathbf{CON}+N_{\rm edge}\mathbf{MUL},\quad(20)$$

となる．また，$N_{\rm edge}$ ラウンドの通信が必要となる．

$\theta_j^{(l)}$ の更新は $v_i^{(\sigma_d,l)}$ による D' 項の総和と，1 回の加算，及び 1 回の定数倍で構成される．この計算は，各エッジに対して 1 回行わため，計算コストは，

$$D'(N_{\rm node}-M_1)\mathbf{ADD}+(N_{\rm node}-M_1)\mathbf{CON},\quad(21)$$

となる．また，通信は発生しない．

以上より，誤差逆伝播における計算コスト $\mathbf{COST}_{\rm back}$ は，

$$\begin{aligned}&\mathbf{COST}_{\rm back}\\ =\ & D'(2N_{\rm edge}+2M_H-M_1M_2)\mathbf{ADD}\\ &+\{N_{\rm node}-M_1+\\ &D'(2N_{\rm edge}-N_{\rm node}+S_T-M_1M_2)\}\mathbf{CON}\\ &+\{D'(N_{\rm node}+M_1)(110\ell\log\ell+118\ell)\\ &+D'(2N_{\rm node}-2M_1-M_H)+N_{\rm edge}\}\mathbf{MUL},\end{aligned}\quad(22)$$

となる．また，通信コスト $\mathbf{ROUND}_{\rm back}$ は，

$$\begin{aligned}&\mathbf{ROUND}_{\rm back}\\ =\ & D'(135N_{\rm node}-135M_1-M_H)+N_{\rm edge},\end{aligned}\quad(23)$$

である．

4.5 予言フェーズ

予言フェーズは学習フェーズの前方伝播と同様の計算をすればよいので，計算コストは前方伝播と同様の計算コストとなる．

5. XOR の学習における計算時間

本節では，XOR を学習するニューラルネットワークについて，秘密計算を適用した場合の学習に関する秘密計算ニューラルネットワークにおける計算時間の評価を示す．

5.1 設定

使用する体として $F_{2^{31}-1}$ を想定する．即ち，$\ell=\lceil\log(2^{31}-1)\rceil=31$ である．また，秘密計算の環境として，$(2,3)$-閾値秘密分散法に基づく秘密計算を行う (P_1,P_2,P_3)．更に，計算環境として濱田らの MEVAL [3] を想定する．

ニューラルネットワークの構造，及び設定は以下のとおりとなる．

- 層の数: $L=3$（中間層は 1 層）.
- ノード数: 入力層 2, 中間層 2, 出力層 1.
- 教師データとして XOR の真理値表を用いる．（$D=4$）
- バッチサイズ: $D'=4$.

5.2 初期設定

ニューラルネットワークの設定から $N_{node}=5, N_{edge}=6$

である．また，$S_T = 3$ である（教師データは 2 個の入力と 1 個の出力を持つ）．従って，式 13, 及び式 14 より，

$$63\mathbf{ADD} + 69\mathbf{CON},$$

の計算と，32 ラウンドの通信が必要となる．

5.3 学習

学習フェーズの計算コスト，及び通信コストは以下の通りである．

学習の開始時に，4 個の教師データをランダムに選択する必要があるため，$4\mathbf{RAN}_4$ の計算コストと，4 ラウンドの通信が必要となる．

式 16 より，前方伝播の計算コストは以下の通りとなる．

$$6\mathbf{ADD} + 3\mathbf{CON} + (10,980 + 10,230 \log 31)\mathbf{MUL}.$$

また，405 ラウンドの通信が必要となる．

5.4 誤差逆伝播

式 22 より，誤差逆伝播の計算コストは以下の通りとなる．

$$40\mathbf{ADD} + 27\mathbf{CON} + (102,450 + 95,480 \log 31)\mathbf{MUL}.$$

また，1,622 ラウンドの通信が必要となる．

5.5 全体の計算時間

式 11 より，繰り返し回数を $\mathbf{TIME}_{\text{ite}}$ に対して，

$$\begin{aligned}&(64\mathbf{TIME}_{\text{ite}} + 63)\mathbf{ADD} + (39\mathbf{TIME}_{\text{ite}} + 69)\mathbf{CON}\\ &+(146,370 + 136,400 \log 31)\mathbf{TIME}_{\text{ite}}\text{MUL}\\ &+4\mathbf{TIME}_{\text{ite}}\mathbf{RAN}_4. \qquad (24)\end{aligned}$$

の計算コストと $3,246\mathbf{TIME}_{\text{ite}} + 32$ ラウンドの通信が必要となる．

表 1 に MEVAL と同等の計算環境における XOR の学習時間を示す．この評価では MEVAL におけるインターネット環境による加算と乗算の実行時間の内，1 ラウンドの計算回数が最も近い 100 回の加算結果と 108,290 回の乗算結果を基に評価している．また，定数倍の計算時間は加算の 2 倍であると仮定して計算している．

表 1 XOR の学習時間

繰り返し回数（回）	計算時間（秒）
100	176.974
1,000	1,769.728
10,000	17,697.268
100,000	176,972.669

また，Python により，XOR の学習を実装し繰り返し回数による誤差の減少度合いを測定した．結果を図 2 に示す．測定結果に依れば，2,000 回の繰り返しで誤差をほぼ 0 にすることが可能である．表 1 から，インターネット環境では約 1 時間で十分な学習が可能であるといえ，現実的な時間で学習可能である．

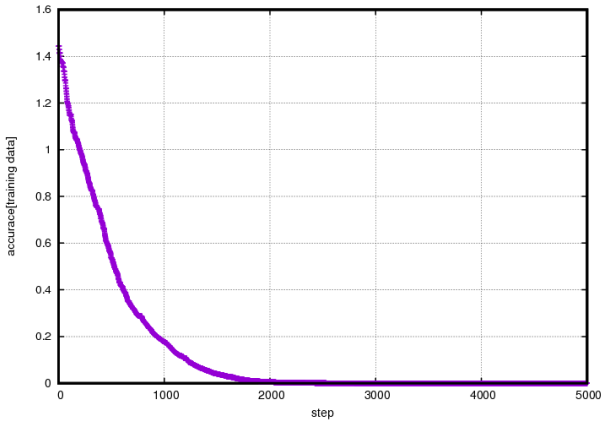

図 2 XOR の誤差の減少度合い（横軸：繰り返し回数，縦軸：誤差）

6. まとめ

本稿では，秘密計算を適用したニューラルネットワークを提案し，実装時の計算時間を評価した．その後，XOR の学習についての計算時間を示し，現実的な時間で学習が可能であることを示した．

6.1 今後の課題

本稿では，二乗誤差で計算を行ったが，多クラス分類問題を解くとなると交差エントロピー誤差と出力層の活性関数として softmax がよく用いられている．一方で交差エントロピー誤差や softmax のような指数や対数が入ってくる計算は秘密計算とは相性が良くない．よって今後の課題として計算コストと精度を両立できる誤差関数と活性関数の提案が重要となる．

また，XOR を学習する秘密計算ニューラルネットワークを実装し，学習の実行時間と本稿の実行時間評価と比較を行う．

参考文献

[1] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In S. Halevi and T. Rabin eds., TCC, Vol. 3876 of Lecture Notes in Computers Science, pp. 285–304. Springer, 2006.

[2] 五十嵐大，菊池亮，濱田浩気，千田浩司．少パーティ数の秘密分散ベース秘密計算のための $O(\ell)$ ビット通信ビット分解および $O(|p'|)$ ビット通信 Modulus 変換法．In CSS, pp. 785–792, 2013.

[3] 濱田浩気，五十嵐大，菊池亮，千田浩司，諸橋玄武，富士仁，高橋克己．実用的な速度で統計分析が可能な秘密計算システム MEVAL．In CSS, pp. 777–784, 2013.