差分プライバシと秘密計算の融合による秘匿性がデータ提供者の数に依存しない秘匿協調学習

岩花 一輝[1,a)]　矢内 直人[1]　藤原 融[1]

概要：多人数でデータを出し合いながら学習を行う協調学習において, 差分プライバシと秘密計算の 2 つを融合した秘匿協調学習が近年注目されている. しかし, 既存の方式ではデータ提供者の数に比例して秘匿性が保証されなくなる問題がある. 本稿ではデータ提供者の数に依存せず, 秘匿性と精度を両立できる融合方式 SPGC を提案する. SPGC では秘密分散で保護されたデータに対し, 秘匿回路の中で差分プライバシのノイズを生成しながら勾配を計算することで, 秘匿性と精度を両立する. また, 実証実験では, 学術用データセットに MNIST, 医療用データセットに Cancer および Diabetes を用いて SPGC の学習性能をそれぞれ評価した. とくに Cancer では, 学習時間は約 10 時間, 精度は 92.2%であった. これは局所差分プライバシに基づく手法と比較して精度が約 5.6%上回っている.

キーワード：秘匿協調学習, 協調学習, 秘密計算, 差分プライバシ

Privacy-Preserving Collaborative Learning Based on Integration of Secure Computation and Differential Privacy

Kazuki Iwahana[1,a)]　Naoto Yanai[1]　Toru Fujiwara[1]

Abstract: For collaborative learning which allows plural data owners to share data for the training, privacy-preserving learning protocols based on secure computation and differential privacy have been proposed in recent years. However, the confidentiality of training data in the existing protocols are downgraded in proportion to the number of data owners. In this paper, we propose a new integration protocol whose confidentiality is independent of the number of data owners. Loosely speaking, our protocol is able to guarantee both the confidentiality and accuracy by distributing data via secret sharing and generating noise within garbled circuits. We also conduct experiments to evaluate the accuracy and the training time with the MNIST dataset as an academic benchmark, and the Cancer and Diabetes datasets as medical diagnosis benchmarks.

Keywords: Privacy-Preserving Collaborative Learning, Collaborative Learning, Secure Computation, Differential Privacy

1. はじめに

1.1 背景

協調学習 (collaborative learning あるいは federated learning) は複数の提供者が保有する学習データで 1 つのモデルを構築する機械学習の一種である. 大まかには, まずそれぞれの提供者は自身が所有するデータを用いて, 他のデータ提供者と連携して 1 つのモデルを学習・構築する. これにより多くのデータを集めたモデルが構築できる. 学習データの情報を他者に漏らさない秘匿性は, 協調学習におけるプライバシ保護として重要な問題である. もしゲノムデータなど個人に紐づくデータが提供される場合, 他のデータ提供者とも連携して学習を行うことから, 各データ提供者は学習データの秘匿性を保つ必要がある. 実際に学習時においてモデルパラメータから学習データを復元する攻撃 [12], [15] など, 秘匿性への脅威が指摘されている.

[1] 大阪大学. Osaka University
[a)] k-iwahana@ist.osaka-u.ac.jp

協調学習においてデータの秘匿性を保つ一般的な手法では, 差分プライバシ [7] を用いている. 具体的には, 各提供者が手元で生成したノイズをモデルパラメータに加えることで, 秘匿性を保つ. しかし, この手法では提供者の数に依存してノイズ量が大きくなるため, モデルの精度が劣化してしまう懸念がある. その一方で, 生成するノイズを小さくしすぎると秘匿性が保証されない可能性もある.

この状況における潜在的な対策は, 差分プライバシと秘密計算を融合させること [6], [19] である. 秘密計算とは, 各データ提供者からの入力に基づいて, 出力以外の情報を漏らすことなく, 事前に定められた関数評価を行う暗号プロトコルである. 秘密計算と差分プライバシの融合においては, 各データ提供者の入力するデータを秘密計算で秘匿することで, 少量のノイズのみを用いて差分プライバシを満たす. 直観的には, 秘密計算を通じてノイズ量を調整できるため, 秘匿性と精度を両立できる.

しかし, 既存の協調学習における融合方式 [26], [28] ではデータ提供者の数が増加するにつれて秘匿性が低下するという欠点がある. 上述したとおり協調学習は多くのデータを集めることから, 多くの提供者を想定すべきである. すなわち融合方式におけるデータの秘匿性はデータ提供者の数から独立していることが望ましい.

1.2 貢献

本稿では差分プライバシと秘密計算の融合において, 各データ提供者の学習データをデータ提供者の数に依存することなく保護可能な初の協調学習方式 *SPGC (Secure and Private Gradient Computation)* を提案する. また, 実証実験を通じて, その学習性能を評価する.

本稿の 1 つ目の貢献は, 既存方式 [26], [28] とは着想が異なる構成を示したことである. 直観的には, 既存方式ではデータ提供者が手元でノイズを生成する局所差分プライバシ (LDP) [5] を採用し秘密計算でそのノイズを削減しているが, このような構成ではどのような秘密計算を用いても人数に依存して秘匿性が低下する. これに対し, SPGC では**秘密計算を通じてノイズを生成**する. 先行研究として Chase ら [4] も秘密計算を通じてノイズ生成する方式を提案しているが, Chase らは構成に関する一般的観点を議論していない. また, 彼らの方式では正負の異なる勾配が同じ値として扱われてしまい, 学習が正しく行えない場合があることも本稿で発見した. 本稿では SPGC の構成にあたり, 上述したような一般的な観点からの議論と Chase らの方式を修正している. 詳細は 4 節に記載するが, SPGC ではデータ提供者がノイズ生成に携わらないことで, データ提供者に依存しない秘匿性を実現できるようになる.

また, 2 つ目の貢献として, 学術的データセットと医療用データセットを用いた実証実験により, 現実的かつ実用的な利用状況での性能を評価した. その際, 秘密計算の通信量を正確に評価するため, ネットワーク通信機能も実装した. 結果として, 局所差分プライバシ (LDP) [5] に基づく単純な手法と比較して精度が改善できた. また, 学習時間を評価した際, 秘密計算の通信量が差分プライバシのノイズ量で変化することを確認した. これは Dwork らの結果 [6] とも一致するが, 本稿では協調学習において同様の結果を実験的に得た点が新しい. また, これにより, 既存方式 [26], [28] における学習時間の変化も（本稿では紙面上省略するが）説明がつくようになった. 詳細は 5 節に記載する.

2. 準備

本節では, 協調学習の概念, および, 秘密計算と差分プライバシの概念について述べる.

2.1 深層学習と協調学習

深層学習には学習データから最適な重みパラメータを持つモデルを作成する学習処理と, そのモデルを利用してタスクを解決する予測処理がある. 本稿では主に学習処理について説明する. まず, 時刻 t のモデルの重みを w_t とする. 一般に学習データ全体の損失関数 L は各学習データ $\boldsymbol{X} = \{x_1, x_2, ..., x_N\}$ に関する損失関数の値の平均であり, $L(w_t, \boldsymbol{X}) = \frac{1}{N}\sum_i L(w_t, x_i)$ と書ける. この損失関数 L を最小にするため, 確率的勾配降下法 (SGD) がよく用いられる. SGD では勾配 $g = \frac{1}{N}\sum_{x_i \in X} \nabla_{w_t} L(w_t, x_i)$ を計算し, 一般に, 学習における重みの更新はバッチと呼ばれるデータ群の単位で行う. 本稿では上述した重み更新の最適化アルゴリズムとして `AdamOptimizer()` を利用する.

上述した通常の深層学習はデータが中央管理されたサーバに集められることに対し, 協調学習は各データ提供者が中央管理されたサーバと連動して学習データやモデルのパラメータを出し合うことで, それぞれが自分の環境内でモデルを学習およびモデルを更新する. なお, 本稿では, モデルパラメータ, すなわち勾配を出し合うことを考える.

2.2 差分プライバシ

差分プライバシ [7] はプライバシを理論的に保証する概念である. 以下に Dwork ら [7] の定義を示す.

定義 1. 定義域 D と値域 R からなるメカニズム (関数) $M : D \to R$ を考える. M が (ϵ, δ)-差分プライバシを満たすとは, 任意の隣接した入力 $d, d' \in D$ および出力の部分集合 $S \subseteq R$ において, $Pr(M(d) \in S) \leq exp(\epsilon)Pr(M(d') \in S) + \delta$ を満たす.

具体的にはデータに対してノイズを加えることで, データごとの出力結果の違いを抑える. 一般的なメカニズムとしては, 任意の関数 f, 平均 0, 標準偏差 σ の正規分布を用い $M(D) = f(D) + N(0, \sigma^2)$, で計算されるガウシアンメカニズムがある. $\delta \leq \frac{4}{5}\exp -(\sigma\epsilon)^2/2$ と $\epsilon < 1$ のとき, ガウシアンメカニズムは (ϵ,δ)-差分プライバシを満たす.

2.3 秘密計算

本稿では秘密計算を実現するため, 秘匿回路と秘密分散を用いる. 以下に, それぞれ紹介する.

まず, 秘匿回路は 2 者間で入力を秘匿しながら任意の関数を評価できる秘密計算を実現するための手法の 1 つであり, 関数をブール 回路で表現し暗号化することで実現される. 直観的には, 暗号化された回路に対して互いに暗号化した入力をすることによって回路の出力結果のみ得られる. 公開利用可能ライブラリとして, TinyGarble [25] がある.

一方, 秘密分散はデータを元のデータの情報を漏らさないような方法で複数のシェアに分散する暗号技術である. 元のデータはしきい値としてあらかじめ設定されたシェアが集まったときのみ, 復元できる. 一般には秘密分散を用いた算術演算は高速に処理できる.

3. 秘匿協調学習

本節では秘匿協調学習の問題設定とその技術的困難性について述べる.

3.1 問題設定

秘匿協調学習とはモデルの学習時と利用時において, k 人の提供者によって提供された学習データを保護しながら行う協調学習である. まず, 提供者の集合 $P = \{p_1, p_2, ..., p_k\}$ に属する各提供者がそれぞれ個別にモデルを（以下, ローカルモデルと呼称）, 中央サーバの集合 $H = \{H_1, H_2, \cdots, H_n\}$ に属する全てのサーバが共通のモデル Y を所有するものとする. また, 提供者 $p_j \in P$ は全体のデータ集合 $\boldsymbol{X} = \{\boldsymbol{X}_{p_1}, \boldsymbol{X}_{p_2}, ..., \boldsymbol{X}_{p_k}\}$ のうち, データ群 $\boldsymbol{X}_{p_j}$ を所有する. このとき, p_j は勾配 $g(\boldsymbol{X}_{p_j})$ を計算し, $g(\boldsymbol{X}_{p_j})$ を各サーバ $H_j \in H$ に送信するものとする.

秘匿協調学習では共通のモデル $Y = f(g(\boldsymbol{X}_{p_1}), g(\boldsymbol{X}_{p_2}), ..., g(\boldsymbol{X}_{p_k}))$ の更新処理として, 秘匿化メカニズム M を用いることで学習データに関する情報を漏らさないようにモデル $M(Y)$ を構成する. ここで, ある信頼できる提供者 $p_j \in P$ のデータ $\boldsymbol{X}_{p_j} \in \boldsymbol{X}$ を秘匿することを考える. このとき, 秘匿協調学習の要件は以下のように定義される [28].

計算時の秘匿性: 敵対者は学習時に, 準正直な $n-1$ 個のサーバと $k-1$ 人の提供者と結託可能である. このとき, プロトコルを通じて, $M(Y)$ 以外の $\boldsymbol{X}_{p_j}$ に関するどんな情報も敵対者に対して漏れない.

出力時の秘匿性: 敵対者は推論時に, モデルの利用者と結託可能である. このとき, プロトコルを通じて, 入力 z に対するモデルの出力 $M(Y(z))$ を除いて, $\boldsymbol{X}_{p_j}$ に関するどんな情報も敵対者に対して漏れない.

これらの要件が必要な理由を以下に述べる. まず, 計算時の秘匿性が保証されないと, 1 節で述べたように学習時に勾配からデータを復元する手法 [12], [15] によって学習データが復元されるからである. また, 出力時の秘匿性が保証されないと, モデルの予測結果から学習データに関する情報が漏れるからである [23].

3.2 技術的困難性

秘密計算と差分プライバシを融合した既存方式 [26], [28] では, データ提供者の数が増加するにつれて学習データの秘匿性は保証されなくなる. 大まかには, 既存方式では各データ提供者 p_j がそれぞれ勾配 $g(\boldsymbol{X}_{p_j})$ に対して差分プライバシのノイズを加え, 中央サーバがグローバルモデルの更新処理 f として秘密計算によりノイズ付きパラメータの平均を計算する. このとき, ノイズは平均 0 の正規分布に従うため, データ提供者の数が増えるほどノイズがない勾配 $g(\boldsymbol{X}_{p_j})$ の割合が増える. さらに, f の中で行われる平均計算によりノイズが相対的に削減されることで, 差分プライバシを満たさなくなる.

この問題の解決は容易ではない. 一般に, 協調学習のような多人数設定で秘匿性を満たすには, 各データ提供者 p_j が手元で勾配 $g(\boldsymbol{X}_{p_j})$ にノイズを付与する局所差分プライバシ (LDP) [5] が考えられる. しかし, LDP では全体のノイズ量がデータ提供者数に依存して増加するため, 精度が劣化する. このため, 既存方式 [26], [28] では LDP のノイズを, 更新処理 f の中で秘密計算を通じて削減していた. これは, LDP を用いる時点で**どのように秘密計算を用いても**秘匿性か精度どちらかが失われることを意味する.

4. 提案方式 SPGC

本節では秘匿協調学習 SPGC (Secure and Private Gradient Computation) を提案する. まず前述した問題に対する設計方針を述べ, その具体的構成を示す. 次に, SPGC の正当性と Chase ら [4] の方式の誤りを述べる.

4.1 設計方針

SPGC では, 既存方式 [26], [28] とは対照的に, モデルの更新処理 f の中, すなわち秘密計算を通じて差分プライバシのノイズを勾配に対し生成する. この方法ではノイズに関する処理が各データ提供者 p_j の処理から独立するため, データ提供者の数に依存することなく秘匿性が保証できる.

具体的には, 各データ提供者 p_j は手元で計算した勾配 $g(\boldsymbol{X}_{p_j})$ を秘密分散を用いて秘匿し, 各サーバに送る. 次に, サーバは更新処理 f を 2 段階に分けて行う. まず秘匿化された各勾配 $g(\boldsymbol{X}_{p_j})$ を復号することなく集約する. この集約処理は秘密分散により高速に計算できる. 次に, 勾配の復号とノイズ生成を同時に行う. この処理は複雑となり秘密分散では難しいため, 任意の回路を計算できる秘匿回路を用いる. 秘匿回路は 2 者間で高速に動作するため SPGC ではサーバ数を $n = 2$ 台とする. この秘匿回路の出力をモデルの更新に用いる全体の勾配とする. これにより, SPGC

は人数に依存せず秘匿性の保証と精度の維持が可能となる.

このような構成として Chase ら [4] の方式もあるが, 彼らは上述した観点のような議論は行っていない. また, 4.3 節に示すが, Chase らの方式は勾配の値が秘密分散の法に合同となったときに勾配の正負が区別できず, 学習が機能しない. これに対し, SPGC では勾配が法と合同にならないよう設計することで, 常に学習が正しく機能する.

4.2 構成

まず, SPGC の初期設定を以下に述べる. (1) 各データ提供者 p_j は自身がもつデータ群 $\boldsymbol{X}_{p_j}$ をランダムに選び, 任意の $t \in [1, \ell]$ においてバッチ化した部分データ群 $\boldsymbol{B}_{t,p_j} \subseteq \boldsymbol{X}_{p_j}$ を生成する; (2) 全体のバッチサイズ m は $m = \sum_{j\in[1,k]} |\boldsymbol{B}_{t,p_j}|$ で計算され, 各 p_j のバッチサイズ $|\boldsymbol{B}_{t,p_j}|$ の合計を指す; (3) 各 p_j は共通のモデルを予め保持している. すなわち, 初期の重み w_0 と構造が共通である.

SPGC では, モデルの更新を各バッチ毎に行う. このため, SPGC のアルゴリズムとして 1 ステップの勾配計算を, 各データ提供者 p_j の処理とサーバ側 H_1, H_2 の処理に分けて述べる. なお, SPGC の秘密分散では剰余演算として, 任意の $x, C \in \mathbb{N}$ に対し $x \bmod C = ((x + C) \bmod 2C) - C$ で定義される smod 演算 [4] を用いる. 秘密分散で smod 演算を計算する際は, 整数 $x \in [-C, C)$ において, x のシェアを $\langle x\rangle_1 = (x + \langle x\rangle_2) \bmod C$ で生成する. ここで, $\langle x\rangle_2$ は $[-C, C)$ の範囲に一様分布する. このとき, シェア $\langle x\rangle_1, \langle x\rangle_2$ は秘密情報 x に関する情報を漏らさない. 秘密情報 x を復元には, $x = \langle x\rangle_1 - \langle x\rangle_2 \bmod C$ を計算する.

4.2.1 データ提供者側の処理

各データ提供者 p_j の処理をアルゴリズム 1 に示す. まず, 4 行目までに x_i に対する勾配 $\overline{g}(x_i)$ を計算し, クリッピングパラメータ C を用いて勾配 $\overline{g}(x_i)$ の範囲を $-C$ 以上 C 以下に制限する. 次に, バッチ $\boldsymbol{B}_{t,p_j}$ に対する合計の勾配 g を計算する. g は浮動小数点であることから, 秘密分散のために 6 行目で固定小数点の勾配 G に変換する. このとき, 勾配の値が法 2^N と合同にならないよう変換している. その後, -2^N から 2^N に一様分布する乱数 r を生成し, 加法的秘密分散のシェア $\langle G\rangle_1$ と $\langle G\rangle_2$ を作成する. 出力されたシェア $\langle G\rangle_1$ を H_1 に, $\langle G\rangle_2$ を H_2 にそれぞれ送信する.

4.2.2 サーバ側の処理

サーバ H_1, H_2 の処理をアルゴリズム 2 に示す. この処理は H_1, H_2 間の対話型処理となっており, 各行の左側は該当するサーバを表す. まず, サーバ H_1, H_2 は 1-4 行目にて, 各 $p_j \in P$ から送られてきたシェア $\langle G^{p_j}\rangle_1$ と $\langle G^{p_j}\rangle_2$ の集約 $\langle G_{H_1}\rangle_1, \langle G_{H_2}\rangle_2$ と, 乱数シードの生成を行う. 5 行目では, H_1 と H_2 は秘匿回路を利用して以下の (1) から (3) の処理まで同時に行うことで, ノイズ付き勾配 G^{DP} を計算する. (1) 集約された勾配を復元することで各提供者の勾配の合計を計算する; (2) 差分プライバシを満たすノイズを $N(0, (\frac{2^N-1}{m})^2)$ の正規分布から生成する; (3) ノイズを勾配に付与する. 最後に, 固定小数点数の勾配 G^{DP} を浮動小数点数の勾配 g^{DP} に変換し, 出力する. 勾配 g^{DP} から `AdamOptimizer()` を用いて, 次の重み w_{t+1} を計算する.

アルゴリズム 1 データ提供者 p_j の処理

入力: 学習ステップ t で利用するバッチ $\boldsymbol{B}_{t,p_j}$, 重み w_t, データサンプル $x_i \in \boldsymbol{B}_{t,p_j}$, 秘密分散の法 $N > log_2(mC+1)$ に対して 2^N, クリッピングパラメータ $C > 0$, 損失関数 $L(\cdot,\cdot)$, 全体のバッチサイズ m.
出力: 勾配に関する加法的秘密分散のシェア $\langle G\rangle_1, \langle G\rangle_2$.
1: **for** $x_i \in \boldsymbol{B}_{t,p_j}$ **do**
2: $\quad \overline{g}(x_i) = \nabla_{w_t} L(w_t, x_i)$
3: $\quad \tilde{g}(x_i) = min\left(1, \frac{C}{||\overline{g}(x_i)||_2}\right)\overline{g}(x_i)$
4: **end for**
5: $g = \sum_{x_i \in \boldsymbol{B}_{t,p_j}} \tilde{g}(x_i)$
6: $G = \frac{2^N-1}{mC} \cdot g$
7: $r \leftarrow [-2^N, 2^N)$
8: $\langle G\rangle_1 = G + r \bmod 2^N$
9: $\langle G\rangle_2 = r \bmod 2^N$

アルゴリズム 2 サーバ H_1, H_2 の処理

入力: 提供者集合 P, 提供者 $p_j \in P$ のアルゴリズム 1 の出力 $\langle G^{p_j}\rangle_1$, $\langle G^{p_j}\rangle_2$, 全体のバッチサイズ m, クリッピングパラメータ C, 標準偏差 σ, 秘密分散の法のサイズ $N > log_2(mC+1)$
出力: $g^{DP} = \sum_{p_j\in P} g^{p_j} + N(0, C^2\sigma^2)$
1: H_1 : $\langle G_{H_1}\rangle_1 = \sum_{p_j\in P}\langle G^{p_j}\rangle_1 \bmod 2^N$
2: H_1 : 乱数シード s_1 を生成
3: H_2 : $\langle G_{H_2}\rangle_2 = \sum_{p_j\in P}\langle G^{p_j}\rangle_2 \bmod 2^N$
4: H_2 : 乱数シード s_2 を生成
5: $G^{DP} = (\langle G_{H_1}\rangle_1 - \langle G_{H_2}\rangle_2) \bmod 2^N + N_{s_1\oplus s_2}(0, (\frac{2^N-1}{m}\sigma)^2)$
6: $g^{DP} = \frac{mC}{2^N-1} G^{DP}$

4.3 正当性

SPGC が勾配を正しく計算できることを簡単に示す. まず, アルゴリズム 1 の 8, 9 行目の式から $G_{H_1} = \sum_{p_j\in P}\left(G^{p_j} + r^{p_j} \bmod 2^N\right) \bmod 2^N$, $G_{H_2} = \sum_{p_j\in P} r^{p_j} \bmod 2^N$ と書ける. さらに, アルゴリズム 1 の 6 行目における $\frac{2^N-1}{mC}$ を通じて, $|\sum_{p_j\in P} G^{p_j}| < 2^N$ となる. このとき, 文献 [4] の補題 4 から, 秘匿回路中の処理は, $(\langle G_{H_1}\rangle_1 - \langle G_{H_2}\rangle_2) \bmod 2^N = \sum_{p_j\in P} G^{p_j} \bmod 2^N = \sum_{p_j\in P} G^{p_j}$ と書ける. すなわち, $\sum_{p_j\in P} G^{p_j}$ が各提供者による固定小数点数の勾配の合計に一致する.

なお, Chase ら [4] の方式では, アルゴリズム 1 の 4 行目のような固定小数点化は行っておらず, 勾配 g に対して法 mC を用いた $g \bmod mC$ なる演算を行っている. このとき, 勾配 g が mC と一致する際に, 勾配の値が一意に定まらない. このため, Chase らの方式は学習が機能しない.

4.4 安全性解析

3 節で述べた 2 つの要件を満たすことを簡潔に示す.

計算時の秘匿性: まず, 差分プライバシのノイズは秘匿

回路中で各サーバにとって未知な乱数シード $s_{i\in\{1,2\}}$ から生成されるため, ノイズの正確な値は秘匿されている. また, 秘密分散と秘匿回路を組み合わることで, 秘匿回路の計算結果 G^{DP} を得るまで復号が一度も行われていない. このとき, Kushilevitz ら [14] の結合定理より, G^{DP} 以外は攻撃者に対して秘匿されている. すなわち, 攻撃者は一部のデータ提供者あるいはどちらかのサーバと結託したとしても, 秘匿回路の出力である G^{DP} を除いて信頼できるデータ提供者 p_j の勾配 g^{p_j} の情報は得られない.

出力時の秘匿性: 出力時のプライバシの保証には, 差分プライバシを満たすことを示せばよい [29]. そのため, SPGC が差分プライバシを満たすことを示す. まず, アルゴリズム 1 およびアルゴリズム 2 に従うと, 学習の 1 ステップにおいて SPGC は (ϵ, δ)-差分プライバシを満たす. この学習を複数回繰り返したとき, Kairouz ら [13] の差分プライバシの結合定理より学習全体でも差分プライバシを満たす.

5. 実験

本節では, SPGC の学習時間と精度に関する実験評価について述べる. とくに, ノイズ量と精度の相関関係, および, ノイズ量と学習時間の相関について評価する.

5.1 実装

SPGC の実装に用いた計算機環境は OS が Ubunt 18.04, メモリが 83GB RAM, CPU に Intel Xeon(R) CPU E5-2630 v3 2.40GHz である. また, GPU は利用しない.

SPGC の主な機能は Python および TensorFlow ライブラリ[*1] により実装されている. また, ニューラルネットワークには, TensorFlow バックエンドで動作する Keras を利用した. また, 学習においては Goodfellow [10] の勾配計算を用いることで, 処理の並列化を行った. とくに, TensorFlow ライブラリの `vectrized_map`[*2] により実装されている.

また, 秘密計算は計算そのものに加えて, プロトコルの中で発生する通信量が実行時間に大きく影響する. 正確な通信時間を計測するため, SPGC の通信部分を Python の gRPC ライブラリ[*3]で実装した. 一方, 秘密分散は Python の Numpy ライブラリ, サーバ間で用いる秘匿回路は C++ のライブラリ TinyGarble [25] で実装した. また, 差分プライバシのノイズについては, シード値に対応するテーブルとして正規分布をあらかじめ秘匿回路に埋め込んだ.

5.2 実験設定

5.2.1 目的

差分プライバシ用のノイズを変化させることで, その精度への影響を測定する. このとき, 学習時間も併せて測定する. なお, 秘匿性については, 各モデル更新時において勾配に付与される差分プライバシのノイズ量として計算される. これは per-step privacy [1] とよばれる. 具体的には, (ϵ, δ)-差分プライバシにおいて ϵ と δ を具体化することで, モデルの更新時にそのパラメータに従った正規分布からノイズを生成することで実現される. 直観的には, 共通の ϵ, δ を用いることで同等の秘匿性が達成されるため, プロトコル間の公平な評価が可能となる.

実験のデータセットには, MNIST, Cancer[*4], Diabetes[*5] を用いる. MNIST は機械学習の研究でよく用いられる学術用ベンチマークデータセットである. 一方, Cancer と Diabetes は, 医療診断用ベンチマークである. MNIST と比べて特徴量は少ないが, 医療やネットワークといった現実の応用技術においては MNIST ですら複雑すぎることもある. すなわち, Cancer と Diabetes の利用は, 現実的かつ実用的な応用を想定した評価といえる.

5.2.2 ベースライン

SPGC の性能評価において, 以下の設定をベースラインとして用いる. 各設定と比較することで, 差分プライバシ, 秘密計算, あるいは両方の導入による影響を評価できる.

Non-privacy: プライバシ保護機能を持たない学習である. SPGC とは状況が異なり, サーバ 1 台を仮定し各提供者から送られてきたそのままの勾配を集約する.

局所差分プライバシ (LDP): LDP [5] を満たす状態で学習し, また秘密計算は用いない. Non-privacy 同様にサーバ 1 台とし, 各提供者によるノイズ付き勾配を集約する.

秘密計算のみ (Only-MPC): SPGC において秘匿回路内でノイズを生成せずに, 勾配の復号のみを行う.

5.2.3 パラメータ設定

アルゴリズム 1, アルゴリズム 2 において, 提供者の数は 3 人とし, クリッピングパラメータは $C = 1$, 法のサイズは $N = 16$ とする. また, 差分プライバシ用のノイズを生成する標準偏差は ϵ, δ から $\sigma = \frac{\sqrt{2log(\frac{1.25}{\delta})}}{\epsilon}$ で計算される [8]. なお, ノイズ用パラメータは $\epsilon = 0.5$, $\epsilon = 2.0$, $\epsilon = 8.0$, $\delta = 10^{-3}$ とする. これは Chase ら [4] の文献と共通である. これら以外のパラメータについては, 次節に記載する.

5.2.4 データセットとアーキテクチャ

MNIST: 縦 28 ピクセル, 横 28 ピクセルからなる 0 から 9 までの数字の白黒手書き画像 70,000 枚からなるデータセットである. 訓練用の画像を 60000 枚, テスト用の画像 10000 枚とし, タスクとしては手書き画像を入力とし, その数字を推定する. また, 訓練用画像 60000 枚を 3 人の提供者で 20000 枚ずつ所有しているものとする. さらに各提

[*1] TensorFlow: `https://www.tensorflow.org/`
[*2] vectorized_map: `https://www.tensorflow.org/api_docs/python/tf/vectorized_map`
[*3] gRPC: `https://grpc.io/docs/tutorials/basic/python/`
[*4] Cancer: `https://www.kaggle.com/uciml/breast-cancer-wisconsin-data`
[*5] Diabetes: `https://www.kaggle.com/uciml/pima-indians-diabetes-database`

供者は 20000 枚の画像において, 1000 枚ずつバッチ化し, これを各提供者で合計する. すなわちアルゴリズム 18 行目にあるバッチサイズは $m = 3000$ とする.

なお, アーキテクチャは以下の三層のものを用いる: (1) 各フィルタ 5×5, ウィンドウごとに 16 個の出力からなる畳み込み層であり, $2-2$ のストライド, パディング, および活性化関数として ReLU を持つ; (2) 各フィルタ $1 \times 5 \times 5$, ウィンドウごとに 32 個の出力からなる畳み込み層であり, $1-2-2$ のストライド, 活性化関数として ReLU を持ち, また, 第一軸での重み共有はないものとする; (3) 10 個の出力を持つ全結合層である.

Cancer: 569 個のデータサンプルを持ち, それぞれのデータは 30 の実数値化された特徴量を持つ. タスクとしては患者の診断結果を入力とし, その患者が癌か否か予測する. 本稿では全体の 569 個のデータを, 訓練用データ 390 個とテストデータ 179 個に分ける. これは提供者数 3 人の間で訓練用のデータを等分散するためである. すなわち, 各提供者は 390 個の学習データのうち, 130 個ずつ所有する. また, 各提供者におけるバッチサイズを 10 とし, アルゴリズム 1 の 8 行目における全体のサイズでは $m = 30$ とする.

データの前処理と使用するアーキテクチャは, バッチ正規化を用いない点をのぞいて, TAPAS [20] に準拠する. なお, バッチ正規化を用いない理由は, 本稿では計算の並列化を行うためである. 具体的には, 各サンプルにおいて 90 次元の二値化ベクトルを生成し, 90 個の入力から全結合層を経て出力層として 2 個のニューロンへとつながる.

Diabetes: 768 個のデータサンプルを持ち, それぞれのデータは 8 次元の特徴量を持っている. タスクとして患者の診断結果を入力とし, その患者が糖尿病かどうかを予測する. 本稿では全体の 768 個のデータを, 訓練用データ 600 個とテストデータ 168 個に分ける. これは提供者数 3 人の間で訓練用のデータを等分散するためである. すなわち, 各提供者は 600 個の学習データのうち, 200 個ずつ所有する. また, 各提供者におけるバッチサイズは Cancer と同様 10 とし, 全体では $m = 30$ とする.

データの前処理と使用するアーキテクチャは, バッチ正規化を用いない点をのぞいて, XONN [20] に準拠する. 具体的には, (1) 8 個の入力と 20 個の出力からなる全結合層で, 活性化関数として sign 関数を持つ, (2) 20 個の入力と 20 個の出力からなる全結合層で, 活性化関数として sign 関数を持つ, (3) 2 個の出力を持つ全結合層である.

5.3 実験結果

5.3.1 精度

MNIST について, 図 1a に示すとおり, Non-privacy の精度は 97.4%であり, Only-MPC は 97.1%であった. これに対し, とくに $\epsilon = 0.5$ では, SPGC の精度は 88.6%であり, LDP と比べて精度を 0.4%改善できている. 一方, SPGC と LDP の比較において, $\epsilon = 8.0$, $\epsilon = 2.0$ ではエポック 18 から同程度の精度となった. すなわち, ノイズ量が小さくなるにつれ, LDP と SPGC の精度の差は小さくなっている.

Cancer について, 図 1b に示すように, Non-privacy と Only-MPC が 98.3% の精度だった. また, SPGC は, $\epsilon = 0.5$ では精度 60.3%, $\epsilon = 2.0$ では精度 63.1%だった. 対照に, $\epsilon = 8.0$ において 92.1% と高い精度だった. また, SPGC と LDP の比較では, ノイズが $\epsilon = 0.5$ のように大きいとき, 精度の差はほとんど見られなかった. 一方, ノイズが小さいとき, SPGC は 92.1%, LDP は 86.6%だった. これにより, LDP より SPGC は精度が改善できている.

Diabetes について, 図 1c に示すように, Non-privacy と Only-MPC で精度は 67.2%であり, エポックごとの精度の変化は全く同じであった. SPGC の精度は $\epsilon = 0.5$ で精度は 35.1% であり, LDP よりも精度が 2.8%ほど下がっている. これに対し $\epsilon = 8.0$ では, SPGC の精度が 64.2% であり, LDP より 2.3%ほど上回っている.

5.3.2 学習時間

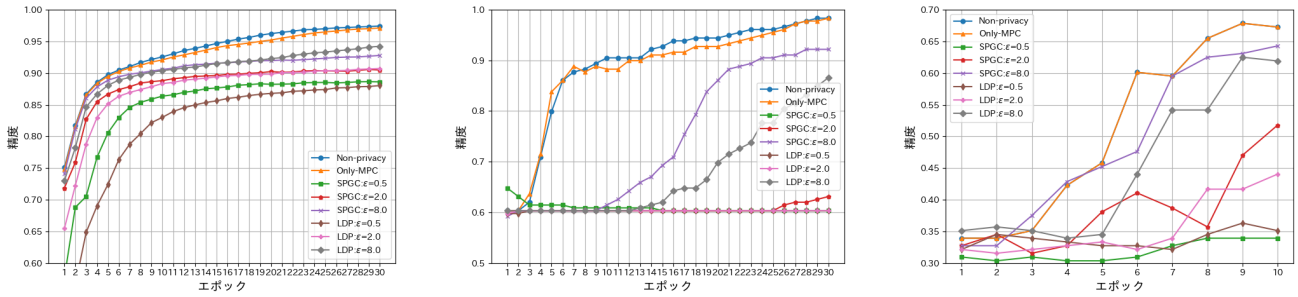
各データセットにおける, SPGC の各パラメータの学習時間と通信量を表 1 に示す. 括弧内に記載している値は各パラメータにおける LDP の値を示しており, 最下段には Only-MPC と括弧内に Non-privacy の値をそれぞれ示している. ”計算時間”はアルゴリズム 1 の 1-5 行目にかかった時間であり, ”通信時間”はそれ以外の処理に要した時間, すなわち SPGC と Only-MPC では提供者とサーバの通信時間およびサーバ H_1, H_2 の通信時間の合計である. また, LDP および Non-privacy では提供者とサーバの通信時間のみを示している. なお, ”通信量”は秘匿回路利用時のサーバ H_1, H_2 間のみの通信量を示している. 紙面上割愛するが, 損失関数の値から学習が収束していることは確認した. また, パラメータ間の学習時間の差異を明確にするため, $\epsilon = 0.01$, $\epsilon = 0.1$ でも実験を行なった.

MNIST では, SPGC が収束するまで約 115 時間であり, Only-MPC と比較して約 43 時間増加した. また $\epsilon = 8.0$ と $\epsilon = 0.01$ で, 学習時間の差が約 3 時間ほど見られた. Cancer では, SPGC で収束するまで約 10 時間であり, Only-MPC と比較して, 約 4 時間増加した. また $\epsilon = 8.0$ と $\epsilon = 0.5$ で, 学習時間の差が約 0.5 時間ほど見られた. Diabetes では, SPGC で収束するまで約 1.2 時間であり, Only-MPC と比較して約 0.4 時間増加した. また $\epsilon = 8.0$ と $\epsilon = 0.5$ で, 学習時間の差が約 0.05 時間ほど見られた. さらに, LDP と比較したとき, SPGC の通信時間は MNIST, Cancer, Diabetes それぞれで約 72 倍, 98 倍, 21 倍であった.

5.4 考察

5.4.1 精度

5.3 節で述べた結果から以下の 2 点を考察する. まず, Only-MPC と Non-privacy を比較して, Cancer や Diabetes



(a) MNIST　　(b) Cancer　　(c) Diabetes

図 1: 各データセットごとの精度

表 1: 各データセットごとの学習時間 (h) とその内訳 (計算時間 (h) と通信時間 (h)), および通信量 (GB)

	MNIST				Cancer				Diabetes			
	計算時間	通信時間	学習時間	通信量	計算時間	通信時間	学習時間	通信量	計算時間	通信時間	学習時間	通信量
$\epsilon = 0.01$	1.69(1.53)	115.77(0.04)	117.45(1.57)	5172.0(-)	0.11(0.11)	10.93(0.01)	11.03(0.12)	736.9(-)	0.07(0.07)	1.19(0.00)	1.26(0.07)	81.6(-)
$\epsilon = 0.1$	1.66(1.53)	114.47(0.04)	116.13(1.57)	5164.6(-)	0.11(0.10)	10.50(0.01)	10.61(0.11)	707.8(-)	0.07(0.06)	1.20(0.00)	1.27(0.06)	78.4(-)
$\epsilon = 0.5$	1.67(1.55)	114.05(0.04)	115.7(1.59)	5094.6(-)	0.11(0.10)	10.65(0.01)	10.76(0.11)	711.7(-)	0.07(0.06)	1.16(0.00)	1.23(0.06)	78.9(-)
$\epsilon = 2.0$	1.67(1.55)	113.45(0.04)	115.1(1.59)	5094.6(-)	0.11(0.09)	10.52(0.00)	10.63(0.10)	699.8(-)	0.07(0.06)	1.15(0.00)	1.22(0.06)	77.5(-)
$\epsilon = 8.0$	1.67(1.55)	112.78(0.04)	114.5(1.59)	5018.8(-)	0.11(0.09)	10.53(0.00)	10.63(0.10)	698.8(-)	0.07(0.06)	1.14(0.00)	1.21(0.06)	77.5(-)
Only-MPC(Non-privacy)	1.66(1.68)	70.16(0.05)	71.82(1.73)	289.2(-)	0.11(0.10)	6.40(0.01)	6.51(0.11)	39.5(-)	0.07(0.06)	0.73(0.00)	0.80(0.06)	4.4(-)

が同じ精度に対し, MNIST では精度劣化が見られた. これは MNIST の固定小数点化によるビット切り捨ての影響が, Cancer や Diabetes よりも大きいと考えられる. この理由として, MNIST のデータ数と特徴量が豊富であり, 固定小数点数化により影響されるデータが多かったと言える. 一方, Cancer や Diabetes は固定小数点化の影響を受けるデータが少なく, 精度劣化が抑えられた.

次に, SPGC と LDP を比較して, $\epsilon = 8.0, 2.0$ の場合で, MNIST や Diabetes の LDP が SPGC よりも精度が良い. この原因として, 提供者が 3 人の状況において, ノイズの影響が LDP では少なかったことである. これに加え, LDP は固定小数点数変換を行わないため, ビット切り捨ての影響が少ないことから精度劣化が抑えられた.

5.4.2 学習時間

各ノイズパラメータ間の学習時間の変動について考察する. 表 1 から各データセットにおいて, ϵ が小さくなる, すなわちノイズ量が大きくなるにつれ学習時間が増加している. これは, 表 1 の"通信量"から, 秘匿回路中の通信量が増加したことに起因する. とくに Only-MPC と比べて SPGC の通信量が大幅に増加していた.

この原因として, 差分プライバシのノイズを生成する機構, つまり秘匿回路内で $N(0, (\frac{2^N-1}{m}\sigma)^2)$ のノイズを生成する部分が回路規模を増加させたと考えられる. これは正規分布のパラメータに依存して通信量が増加する, すなわち, ノイズにより秘匿回路の通信量が増加することを意味している. 既存研究 [6] では秘密分散における通信量がノイズ量に応じて増加することを指摘しており, 秘匿回路という違いはあるが, 上述した結果は既存の結果と一致する.

6. 関連研究

秘密計算と差分プライバシの融合研究: Dwork ら [6] は秘密分散を通じたノイズ付き総和計算において, 秘密分散における通信量および回路規模を示した. 本稿の結果は Dwork らの結果と一致するが, 秘匿回路を含む協調学習という点で異なる. 後続研究の多くは中央値 [3], [19], [24] や総和 [11], [21] など基本的な計算だったが, 近年において機械学習への応用 [4], [26], [28] が示された.

本稿に最も近い研究は Chase ら [4] である. 本稿では, Chase らの方式は学習が機能しないことを新たに発見し, また, その問題を解決した. 一方, 本稿の主な背景は, HybridAlpha [28] と Truex らの構成 [26] が人数が増えたときにノイズが相対的に減ってしまうことにあった. 加えて, これらの研究では, 差分プライバシ用ノイズの有無による計算時間への影響も言及されていない.

プライバシ保護機械学習: プライバシ保護機械学習は差分プライバシ型 [1], [18], [22], [30] と秘密計算型 [2], [16], [17], [27] に大別される. 前者は学習時間への影響は少ない一方, 計算過程の保護と精度の両立が容易ではない. 一方, 秘密計算型では精度を維持したまま計算過程を保護できるが, モデルパラメータから学習データが漏れる可能性 [9], [23] がある. これらの利点を統合するべく, 前述した融合プロトコル [4], [26], [28] が提案された.

7. まとめ

本稿では差分プライバシと秘密計算を融合した秘匿協調学習アルゴリズム SPGC を提案した. とくに, 秘密計算の中でノイズを生成することで秘匿性がデータ提供者数に依

存しない構成を示した. 関連して, 類似した研究 [4] における勾配が正しく復号されない問題点を, SPGC では修正している. 次に, SPGC の学習性能を評価する実験において, 医療用データセット Cancer の学習が約 10 時間, 精度 92.7%で評価できた. 学習時間の内訳を調査したところ, その 98%以上が秘密計算の通信時間であった. 関連して, 学習時間がノイズ量に比例して増加していることも確認した. 今後の課題はより大きく複雑な機構で実験すること, また, WAN 環境で実験を行うことである.

謝辞: 本研究の一部は, 内閣府が進める戦略的イノベーション創造プログラム（SIP）「重要インフラ等におけるサイバーセキュリティの確保」（管理法人：NEDO）によって実施されました. また, 本研究に有益なコメントをいただきました株式会社レピダム様に感謝いたします.

参考文献

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proc. of CCS*, pages 308–318. ACM, 2016.

[2] N. Agrawal, A. Shahin Shamsabadi, M. J. Kusner, and A. Gascón. Quotient: Two-party secure neural network training and prediction. In *Proc. of CCS*, pages 1231–1247. ACM, 2019.

[3] J. Bohler and F. Kerschbaum. Secure sublinear time differentially private median computation. In *Proc. of NDSS*. Internet Society, 2020.

[4] M. Chase, R. Gilad-Bachrach, K. Laine, K. Lauter, and P. Rindal. Private collaborative neural network learning. Cryptology ePrint Archive, Report 2017/762, 2017. `https://eprint.iacr.org/2017/762`.

[5] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *Proc. of FOCS*, pages 429–438. IEEE, 2013.

[6] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Proc. of EUROCRYPT*, LNCS, pages 486–503. Springer, 2006.

[7] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. of TCC*, volume 3876 of *LNCS*, pages 265–284. Springer, 2006.

[8] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[9] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *Proc. of Usenix Security*, pages 17–32. Usenix Association, 2014.

[10] I. Goodfellow. Efficient per-example gradient computations, 2015.

[11] S. Goryczka and L. Xiong. A comprehensive comparison of multiparty secure additions with differential privacy. *IEEE Transactions on Dependable and Secure Computing*, 14(5):463–477, 2017.

[12] Z. He, T. Zhang, and R. B. Lee. Model inversion attacks against collaborative inference. In *Proc. of ACSAC*, pages 148–162. ACM, 2019.

[13] P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. *IEEE Transactions on Information Theory*, 63(6):4037–4049, 2017.

[14] E. Kushilevitz, Y. Lindell, and T. Rabin. Information-theoretically secure protocols and security under composition. *SIAM Journal on Computing*, 39:2090–2112, 2010.

[15] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *Proc. of IEEE S&P*, pages 691–706. IEEE, 2019.

[16] P. Mohassel and P. Rindal. Aby3: A mixed protocol framework for machine learning. In *Proc. of CCS*, pages 35–52. ACM, 2018.

[17] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *Proc. of IEEE S&P*, pages 19–38. IEEE, 2017.

[18] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson. Scalable private learning with PATE. In *Proc. of ICLR*, 2018.

[19] M. Pettai and P. Laud. Combining differential privacy and secure multiparty computation. Cryptology ePrint Archive, Report 2015/598, 2015. `https://eprint.iacr.org/2015/598`.

[20] A. Sanyal, M. Kusner, A. Gascon, and V. Kanade. Tapas: Tricks to accelerate (encrypted) prediction as a service. In *Proc. of ICML*, pages 4497–4506, 2018.

[21] E. Shi, T. H. Chan, E. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *Proc. of NDSS*, pages 1–17. Citeseer, 2011.

[22] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Proc. of CCS*, pages 1310–1321. ACM, 2015.

[23] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *Proc. of IEEE S&P*, pages 3–18. IEEE, 2017.

[24] A. Smith, A. Thakurta, and J. Upadhyay. Is interaction necessary for distributed private learning? In *Proc. of IEEE S&P*, pages 58–77. IEEE, 2017.

[25] E. M. Songhori, S. U. Hussain, A.-R. Sadeghi, T. Schneider, and F. Koushanfar. Tinygarble: Highly compressed and scalable sequential garbled circuits. In *Proc. of IEEE S&P*, pages 411–428. IEEE, 2015.

[26] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou. A hybrid approach to privacy-preserving federated learning. In *Proc. of AISec*, pages 1–11. ACM, 2019.

[27] S. Wagh, D. Gupta, and N. Chandran. Securenn: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 2019(3):26–49, 2019.

[28] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *Proc. of AISec*, pages 13–23. ACM, 2019.

[29] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *Proc. of CSF*, pages 268–282. IEEE, 2018.

[30] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex. Differentially private model publishing for deep learning. In *Proc. of IEEE S&P*, pages 332–349. IEEE, 2019.