

差分プライベートな確率的勾配降下法に関する一検討

長谷川 聡^{1,a)} 三浦 堯之¹

概要：本稿では、差分プライベートな確率的勾配降下法に関する検討を行う。深層学習を差分プライベート化するために従来から用いられている DP-SGD では、勾配の L2 感度を抑制するために、勾配の L2 距離が規定値以上のものをカットする「ノルムクリッピング」というアプローチを用いている。それに対し本稿では、ノルムクリッピングとは異なるアプローチを用いる。具体的には、深層学習の損失関数にリプシッツ連続の制約を設けることで、ノルムクリッピングをせずに勾配の L2 感度を抑える方法を検討する。この新たな方式の位置づけについて考察を行った結果を報告する。

1. はじめに

近年のコンピュータの計算能力や、AI 技術の向上に伴い、データ利活用が活発に行われている。特に個人に紐づく情報（パーソナルデータ）の利活用が盛んに進んでいる。パーソナルデータを安易に利用してしまうとプライバシーを侵害するリスクが生じることから、差分プライバシー [8] をはじめとしたプライバシーを保護した形でのデータ利活用技術が急速に発達しつつある。本稿では、AI 技術として近年最も注目を浴びている深層学習に着目し、差分プライバシーを適用したデータ利活用技術に関して取り扱う。

1.1 差分プライベートな確率的勾配降下法

差分プライバシーとは、パーソナルデータを入力とし、統計値や機械学習モデルなどを出力するシステムに対し、その出力結果から元データの個人情報を守る技術である [8]。深層学習に対して差分プライバシーを適用する方法として、差分プライベート確率的勾配降下法 (DP-SGD) がある [5], [7]。DP-SGD による深層学習アルゴリズムの差分プライベート化は、深層学習のモデルに依存せず適用可能であることから、その汎用性の高さによりデファクトスタンダードとなっている。特に Abadi ら [5] による勾配のノルムが一定以上のものをカットする「ノルムクリッピング」に基づく方法が用いられている。

一方、深層学習では、バッチ単位の計算（並列計算）によって高速化が実現されているが、ノルムクリッピング

はこの並列計算で処理することができない^{*1}。Tensorflow Privacy [4], [6] では、`tf.vectorized_map` [3] のようなバッチ内のサンプルの数だけモデルを呼び出すことで、勾配のノルムの並行計算を可能にしている。しかしながら、[3] に言及されているように、この方法では高速化の代わりにメモリを多く犠牲にすることから、大規模なモデルに対する適用が課題となる。

1.2 貢献

本稿では、DP-SGD の課題を解決をするため、深層学習のモデルに制約を入れることで、ノルムクリッピングを不要とする差分プライベートな深層学習アルゴリズムを提案する。この方法は、損失関数に k -リプシッツ連続の制約を設けることで、勾配のノルムを k 以下に抑えノルムクリッピングを不要とする。

本稿の貢献はまとめると 2 つある。

- (1) 勾配のノルムクリッピングを不要とする差分プライベートな深層学習アルゴリズムを提案。
- (2) 他の DP-SGD 手法とのプライバシー・計算量・汎用性の観点での比較検討評価。

2 節では、準備として深層学習、差分プライバシー、DP-SGD を導入し、3 節で重要となるリプシッツ連続を導入する。3 節では、リプシッツ連続の性質を利用した DP-SGD のアルゴリズムについて検討する。4 節では、他手法との比較検討を行う。5 節ではまとめと今後の課題について述べる。

¹ NTT セキュアプラットフォーム研究所
NTT Secure Platform Laboratories, Musashino-shi, Tokyo
180-8585, Japan

^{a)} satoshi.hasegawa.ks@hco.ntt.co.jp

^{*1} 昨今の Tensorflow [2] や Pytorch [1] 等の深層学習ライブラリではバッチ内のサンプルごとの勾配にアクセスすることができないからである。バックエンドで用いている cuDNN 等の制限にもよると言われている [15]

2. 準備

本稿で取り扱う機械学習とは、 N 個の入出力の組 $\{(\mathbf{x}_i \in \mathbb{R}^d, \mathbf{y}_i \in \mathbb{R}^c)\}_{i=1}^N$ に対し、損失関数 $\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}_i(f(\mathbf{x}_i; \theta), \mathbf{y}_i)$ が最も小さくなるような関数 $f: \mathbb{R}^d \rightarrow \mathbb{R}^c$ のパラメータ θ を決定することである。

2.1 深層学習

関数 f として多層ニューラルネットワークを用いたものは深層学習と呼ばれている。 L 層の多層ニューラルネットワークを、パラメータ $\theta = \{W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}\}_{l=1}^L$ および非線形関数 $\{\phi^{(l)}: \mathbb{R}^{d_l} \rightarrow \mathbb{R}^{d_l}\}_{l=1}^L$ の組み合わせにより、

$$f(\mathbf{x}_i; \theta) = \phi^{(L)}(W^{(L)}(\phi^{(L-1)}(W^{(L-1)} \dots \phi^{(1)}(W^{(1)} \mathbf{x}_i))))).$$

と表す。 $h^{(l)}(\mathbf{x}) = W^{(l)}\mathbf{x}$ とした場合、 f は $\phi^{(l)}$ と $h^{(l)}$ の合成関数、すなわち

$$f = \phi^{(L)} \circ h^{(L)} \circ \dots \circ \phi^{(1)} \circ h^{(1)} \quad (1)$$

と表すことができる。

2.1.1 確率的勾配降下法

多層ニューラルネットワークの学習には、一般的に確率的勾配降下法が用いられる。確率的勾配降下法とは、各入出力 $\{\mathbf{x}_i, \mathbf{y}_i\}$ ごとの勾配を計算し、その勾配に基づいてパラメータ θ を逐次更新する方法である。

$$\theta_{t+1} \leftarrow \theta_t - \alpha \nabla \mathcal{L}_i(f(\mathbf{x}_i; \theta_t), \mathbf{y}_i) \quad (2)$$

を $t = 1, \dots, T$ 回繰り返すことで θ を決定する。ここで $\alpha \in \mathbb{R}_{>0}$ は学習率である。ミニバッチ勾配降下法と呼ばれる M 個の勾配の平均 $\frac{1}{M} \sum_{i=1}^M \nabla \mathcal{L}_i(f(\mathbf{x}_i; \theta_t), \mathbf{y}_i)$ によって、式 2 を更新する方法がある。本稿では特にミニバッチ勾配降下法と確率的勾配降下法を区別せずに議論する。

2.2 差分プライバシー

プライバシー保護基準の 1 つとして差分プライバシー [8], [9] がある。差分プライバシーは、レコードのプライバシーを保護するために、データベースの 1 レコードが違っていても、そのデータベースを入力した出力の結果に、大きな差が生じないことを保証する。レコードを 1 個人と対応付けることで、1 個人のプライバシーを保護すると解釈ができる。

定義 1. ランダムメカニズム M が (ϵ, δ) -差分プライバシーを満たすとは、任意の隣接したデータベース D_1 および D_2 に対し、

$$p[M(D_1) \in S] \leq e^\epsilon p[M(D_2) \in S] + \delta$$

が成り立つことである。ただし、ここで S とは、 M の出力空間の任意の部分空間とする。

差分プライバシーは、任意の 1 レコード異なるデータベー

ス D_1, D_2 の出力の確率を、 ϵ, δ という 2 つのパラメータで抑えており、 ϵ, δ が小さい値であればあるほどプライバシー保護強度が強くなる (一般的に出力の有用性は低くなる)。

2.2.1 ガウシアンメカニズム

(ϵ, δ) -差分プライバシーを満たす方法として、出力結果にガウス分布に従うノイズを付与する方法が提案されており、ガウシアンメカニズムと呼ばれている [9]。ガウシアンメカニズムの定義のため、まず L_2 -感度を定義する。

定義 2. L_2 -感度とは任意の隣接データベース D_1 および D_2 に対し、以下の式で表されるものをいう。

$$\Delta_2(g) = \max_{D_1, D_2} \|g(D_1) - g(D_2)\|_2.$$

定義 3. ガウシアンメカニズムとは、 $\delta \in (0, 1), \epsilon \in (0, 1)$ において、

$$M_{Gauss}(x, g, \epsilon, \delta) = g(x) + \mathcal{N}(0, \sigma^2)$$

となるプライバシー保護メカニズムである。ただし $\sigma^2 = \frac{2 \log(1.25/\delta) \Delta_2(g)^2}{\epsilon^2}$ である。

2.3 DP-SGD

深層学習アルゴリズムを差分プライバシーにする方法として、差分プライバシー確率的勾配降下法 (DP-SGD) がある [5], [7]。確率的勾配降下法を差分プライバシーにしたものであり、機械学習全般に適用可能であるが、特に Abadi らが深層学習に用いたことで、現在の差分プライバシーな深層学習におけるデファクトスタンダードとなっている。

DP-SGD は、確率的勾配降下法の勾配のノルムをクリッピングし、 L_2 -感度を一定以下に抑制する。そしてその勾配に対してガウシアンメカニズムを適用することで、差分プライバシーを満たすようにしている。以下の 5 つのステップを $t = 1, \dots, T$ 回繰り返すことで、 θ を算出する。

1. サブサンプリング N 個の入出力の組 $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ から、 S 個のサンプル $S_t = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^S$ をランダム抽出する。

2. 勾配計算 各 $(\mathbf{x}_i, \mathbf{y}_i) \in S_t$ に対して、 $g_t(\mathbf{x}_i, \mathbf{y}_i) \leftarrow \nabla_{\theta_t} \mathcal{L}_i(f(\mathbf{x}_i; \theta_t), \mathbf{y}_i)$ を算出する。

3. ノルムクリッピング $g_t(\mathbf{x}_i, \mathbf{y}_i) \leftarrow \frac{g_t(\mathbf{x}_i, \mathbf{y}_i)}{\max(1, \frac{\|g_t(\mathbf{x}_i, \mathbf{y}_i)\|_2}{C})}$

4. ノイズ加算 $\bar{g}_t \leftarrow \frac{1}{S} (\sum_i g_t(\mathbf{x}_i, \mathbf{y}_i) + \mathcal{N}(0, \sigma^2 C^2 I))$

5. 勾配更新 $\theta_{t+1} \leftarrow \theta_t - \alpha \bar{g}_t$

通常差分プライバシーを満たすためには、 L_2 -感度を評価する必要がある。勾配の L_2 -感度は評価が難しいことから、DP-SGD[5] では、勾配の L_2 ノルムをクリッピングすることで、 L_2 -感度を意図的に一定の値 C 以下に抑えるアプローチを取る。

ノルムクリッピングは複数データを一括で処理するバッチ単位の処理に不向きである。ノルムクリッピングに必要なサンプルごとの勾配のノルムの取得は、GPU といったハー

ドウェアアクセラレーションによる高速化方法が提供されておらず、計算に時間を要してしまうからである。この問題に対し TensorflowPrivacy[4] では、`tf.vectorized_map[3]` のようにモデルを並列に呼び出すアプローチを取っているが、メモリ使用量が増えてしまい、大規模なモデルでの活用に課題が生じる。

2.4 リブシツ連続

リブシツ連続とは関数の連続性に関する性質の1つである。3節の議論で多用する性質であることからここで定義を行う。

定義 4. 任意の $v, w \in \mathbb{R}^p$ に対して、関数 $j : \mathbb{R}^p \rightarrow \mathbb{R}^q$ が k -リブシツ連続であることは、

$$\frac{\|j(v) - j(w)\|_2}{\|v - w\|_2} \leq k \quad (3)$$

が成り立つことをいう。

また、 j が k -リブシツ連続な関数である場合、

$$\|\nabla_v j(v)\|_2 \leq k$$

は同値である。

定義 5.

$$\|j\|_{lip} := \min\{k \in \mathbb{R}_{\geq 0} | j \text{ is } k\text{-Lipschitz Continuous}\}$$

を関数 j のリブシツ定数と呼ぶ。

以降の議論のため、以下3つの命題を考える。

命題 1. $j : \mathbb{R}^p \rightarrow \mathbb{R}^q$ が k_j -リブシツ連続、 $a \in \mathbb{R}_{>0}$ とするとき、関数 $aj : \mathbb{R}^p \rightarrow \mathbb{R}^q$ は ak_j -リブシツ連続である。

証明. $x, y \in \mathbb{R}^p$ とする。このとき、

$$\begin{aligned} \frac{\|aj(x) - aj(y)\|_2}{\|x - y\|_2} &\leq a \frac{\|j(x) - j(y)\|_2}{\|x - y\|_2} \\ &\leq ak_j \end{aligned}$$

である。□

命題 2. $j : \mathbb{R}^p \rightarrow \mathbb{R}^q$ が k_j -リブシツ連続、 $u : \mathbb{R}^p \rightarrow \mathbb{R}^q$ が k_u -リブシツ連続とする。このとき、関数 $j+u : \mathbb{R}^p \rightarrow \mathbb{R}^q$ は $k_j + k_u$ -リブシツ連続である。

証明. $x, y \in \mathbb{R}^p$ とする。このとき、

$$\begin{aligned} &\frac{\|j(x) + u(x) - j(y) - u(y)\|_2}{\|x - y\|_2} \\ &\leq \frac{\|j(x) - j(y)\|_2 + \|u(x) - u(y)\|_2}{\|x - y\|_2} \\ &= \frac{\|j(x) - j(y)\|_2}{\|x - y\|_2} + \frac{\|u(x) - u(y)\|_2}{\|x - y\|_2} \\ &\leq k_j + k_u \end{aligned}$$

である。□

命題 3. $j : \mathbb{R}^p \rightarrow \mathbb{R}^q$ が k_j -リブシツ連続、 $u : \mathbb{R}^q \rightarrow \mathbb{R}^r$ が k_u -リブシツ連続とする。このとき、関数 $u \circ j : \mathbb{R}^p \rightarrow \mathbb{R}^r$ は $k_j \cdot k_u$ -リブシツ連続である。

証明. $x, y \in \mathbb{R}^p$ とする。 $j(x) = j(y)$ ならば、

$$\frac{\|u(j(x)) - u(j(y))\|_2}{\|x - y\|_2} = 0 \leq k_j \cdot k_u$$

である。

$j(x) \neq j(y)$ の場合も、

$$\begin{aligned} &\frac{\|u(j(x)) - u(j(y))\|_2}{\|x - y\|_2} \\ &= \frac{\|u(j(x)) - u(j(y))\|_2}{\|j(x) - j(y)\|_2} \cdot \frac{\|j(x) - j(y)\|_2}{\|x - y\|_2} \leq k_j \cdot k_u \end{aligned}$$

である。□

3. ノルムクリッピングを不要とする DP-SGD の検討

本稿ではスケーラビリティに影響を及ぼすノルムクリッピングを不要とすることで、大規模なモデルでも適用可能な DP-SGD を検討する。本節では、ノルムクリッピング不要な DP-SGD の要件を整理し、その具体的な構成方法について示す。

3.1 k -リブシツ連続な損失関数による DP-SGD

Bassily ら [7] らが DP-SGD の要件として、損失関数が k -リブシツ連続であれば、ノルムクリッピングが不要であることを暗に示している。式4より、損失関数が k -リブシツ連続であれば、勾配の $L2$ ノルムが k 以下であること、すなわち勾配の $L2$ -感度が k であることを保証する。よって損失関数 $\mathcal{L}$ が k -リブシツ連続であれば、ノルムクリッピングを必要とせず、ノイズ付与のみを行えばよいといえる。

損失関数 $\mathcal{L}$ が 1-リブシツ連続であることを仮定した場合、以下の手続きを $t = 1, \dots, T$ 回繰り返すことにより θ を算出する。

1. サブサンプリング N 個の入出力の組 $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ から、 S 個のサンプル $S_t = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^S$ をランダム抽出する。
2. 勾配計算 各 $(\mathbf{x}_i, \mathbf{y}_i) \in S_t$ に対して、 $g_t(\mathbf{x}_i, \mathbf{y}_i) \leftarrow \nabla_{\theta_t} \mathcal{L}_i(f(\mathbf{x}_i; \theta_t), \mathbf{y}_i)$ を算出する。
3. ノイズ加算 $\bar{g}_t \leftarrow \frac{1}{S} (\sum_i g_t(\mathbf{x}_i, \mathbf{y}_i) + \mathcal{N}(0, \sigma^2 I))$
4. 勾配更新 $\theta_{t+1} \leftarrow \theta_t - \alpha \bar{g}_t$

この方式を Lipschitz Continuous DP-SGD (以降、DP-SGD with LC) と呼ぶこととする。

以降では、議論の簡単化のため、損失関数 $\mathcal{L}$ が 1-リブシツ連続であることを保証するための要件について整理する。なお、 k -リブシツ連続な場合には容易に拡張可能である。

3.2 1-リブシツ連続な損失関数

損失関数 $\mathcal{L}$ は、訓練データ集合 $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ 及びニューラルネットワーク $f(\cdot; \theta)$ から成る。いま、訓練データを固定すると、ニューラルネットワーク f や損失関数 $\mathcal{L}$ はパラメータ θ の関数と見ることができる (以降の議論での混乱を避けるため、それぞれを $f^\theta = f(\theta; \mathbf{x})$, $\mathcal{L}^\theta$ と明示的に書くこととする)。加えて損失関数 $\mathcal{L}^\theta$ は、2つの関数 $\mathcal{L}_i^\theta, f^\theta$ の合成関数 $\mathcal{L}^\theta = \mathcal{L}_i^\theta \circ f^\theta$ と見ることができる。損失関数 $\mathcal{L}^\theta$ を θ で微分すると、合成関数の微分により、

$$\nabla_\theta \mathcal{L}^\theta(\theta) = \frac{1}{N} \sum_{i=1}^N \frac{\partial \mathcal{L}_i^\theta}{\partial f^\theta} \frac{\partial f^\theta}{\partial \theta}$$

となる。よって、 $\mathcal{L}^\theta$ が1-リブシツ連続になるためには、 $\|\frac{\partial \mathcal{L}_i^\theta}{\partial f^\theta}\|_{lip} \|\frac{\partial f^\theta}{\partial \theta}\|_{lip} \leq 1$ とすれば良い。 $\mathcal{L}_i^\theta, f^\theta$ それぞれが1-リブシツ連続を満たせば十分である。

3.2.1 ニューラルネットワーク f^θ の1-リブシツ連続性

式1の f^θ が1-リブシツ連続であるためには、命題3より

$$\|f^\theta\| = \|\phi^{(L)}\|_{lip} \cdot \|h^{(L)}\|_{lip} \cdots \|\phi^{(1)}\|_{lip} \cdot \|h^{(1)}\|_{lip} \leq 1$$

を満たせば良い。 L 層目を除く (L 層目については次節で述べる) 非線形関数 $\phi^{(i)}$ (活性化関数) が1-リブシツ連続であると仮定すると*2, $h^{(i)}(\mathbf{x})$ が $W^{(i)}$ に関して1-リブシツ連続であれば良い。

$h^{(i)}$ に関して $\mathbf{x}$ を固定した際の $W^{(i)}$ に関するリブシツ連続性について評価する。まず、 $h^{(i)}(\mathbf{x}) = W^{(i)} \mathbf{x} = \mathbf{X}' \text{vec}(W^{(i)})$ となる $\mathbf{X}' \in \mathbb{R}^{d_i \times d_{i-1}}$ を用意する。ただし、 $\text{vec} : \mathbb{R}^{d_i \times d_{i-1}} \rightarrow \mathbb{R}^{d_i d_{i-1}}$ である。小さい値 $\xi \in \mathbb{R}^{d_i d_{i-1}}$ を用意し、式3を評価すると、

$$\begin{aligned} & \frac{\|\mathbf{X}'(\text{vec}(W^{(i)}) + \xi) - \mathbf{X}'\text{vec}(W^{(i)})\|_2}{\|(\text{vec}(W^{(i)}) + \xi) - \text{vec}(W^{(i)})\|_2} \\ &= \frac{\|\mathbf{X}'(\text{vec}(W^{(i)}) + \xi) - \mathbf{X}'\text{vec}(W^{(i)})\|_2}{\|\xi\|_2} \\ &= \frac{\|\mathbf{X}'\xi\|_2}{\|\xi\|_2} \leq \|\mathbf{X}'\|_2 = \Sigma(\mathbf{X}') \end{aligned}$$

が成り立つ。ここで $\Sigma(\mathbf{X}')$ は行列の $L2$ ノルム (またはスペクトルノルムともいう)、すなわち最大特異値を表す。 $h^{(i)}$ が1-リブシツ連続な関数であるためには、最大特異値を1以下にすれば良いとわかる。

最大特異値を1以下にするためには、対象となる行列から最大特異値を除算すればよい。すなわち、

$$\mathbf{X}' \leftarrow \mathbf{X}' / \Sigma(\mathbf{X}')$$

とすればよい*3。行列の最大特異値は、Power Iteration アルゴリズム [12] を用いれば高速に計算が可能である。

*2 ReLU[14], Leaky Relu[11] といった頻繁に用いられる非線形関数は k -リブシツ連続を満たすことが知られている。これらは区分的線形な関数であり、勾配を計算すれば容易に示すことができる

*3 Spectral Normalization[13] と似た方法である。Spectral Normalization は重み行列のスペクトルノルムを抑えるが、提案方式は各層の入力のスペクトルノルムを抑える点が異なる

3.2.2 最終層の非線形関数に関して

$\phi^{(L)}$ は最終層の非線形関数であり、出力を返すために用いる層である。ReLU[14], Leaky ReLU[11] などの活性化関数とは異なるものが用いられる。分類タスクでは2値分類であれば最終層として sigmoid 関数 $\text{sigmoid}(x) = \frac{1}{1+\exp(-x)}$ 、多値分類であれば入力 $\mathbf{x} = (x_1, \dots, x_d)$ に対して softmax 関数 $y_i = \text{softmax}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^d \exp(x_j)}$ などが利用される ($\sum_i y_i = 1$ となる)。sigmoid 関数に関しては以下の命題によりリブシツ連続であることがわかる。

命題 4. sigmoid 関数は $1/4$ -リブシツ連続を満たす。

証明. sigmoid 関数 $s(x) = \text{sigmoid}(x)$ とする。 s の微分は $\frac{ds(x)}{dx} = (1 - s(x)) \cdot s(x)$ であり、2階微分は $\frac{d^2s}{dx^2} = s(x)(1 - s(x))(1 - 2s(x))$ である。 $\frac{d^2s}{dx^2} = 0$ となる x は $x = 0$ であり、また $x < 0$ ならば $\frac{d^2s}{dx^2} > 0$ であり、 $x > 0$ ならば $\frac{d^2s}{dx^2} < 0$ であることからことから、 $\frac{ds(x)}{dx}$ の最大値は $1/4$ であるとわかる。よって、任意の入力 x に対しても微分の値が $1/4$ で抑えられることから、sigmoid 関数は $1/4$ -リブシツ連続を満たす。□

また、softmax 関数に関しても1-リブシツ連続であることが示されている [10]。

3.2.3 損失関数のリブシツ連続

最終層 $\phi^{(L)}$ を除く多層ニューラルネットワーク f が1-リブシツ連続であり、最終層 $\phi^{(L)}$ として sigmoid 関数もしくは softmax 関数を用いたと仮定する。損失関数 $\mathcal{L}_i^\theta$ として例えば二乗平均誤差 $\mathcal{L}_i = \mathcal{L}_{MSE}(\theta) = \frac{1}{4} \|f^\theta(\theta; \mathbf{x}) - \mathbf{y}\|^2$ は、1-リブシツ連続を満たす。

定理 1. 入出力 $\{\mathbf{x} \in \mathbb{R}^d, \mathbf{y} \in \mathbb{R}^c\}$ が固定された、最終層 $\phi^{(L)}$ が sigmoid 関数もしくは softmax 関数である1-リブシツ連続な多層ニューラルネットワーク $f^\theta(\theta; \mathbf{x})$ による損失関数 $\mathcal{L}_{MSE}$ は、1-リブシツ連続な関数である。

証明. $\mathcal{L}_{MSE}$ を θ で微分した場合、合成関数の微分により

$$\nabla \mathcal{L}_{MSE}(\theta) = \frac{1}{2} (\mathbf{y} - f^\theta(\theta; \mathbf{x})) \nabla_\theta f^\theta(\theta; \mathbf{x})$$

となる。このノルムを算出すると、

$$\begin{aligned} \|\nabla \mathcal{L}_{MSE}(\theta)\|_2 &= \frac{1}{2} \|\mathbf{y} - f^\theta(\theta; \mathbf{x})\|_2 \cdot \|\nabla_\theta f^\theta(\theta; \mathbf{x})\|_2 \\ &\leq \frac{1}{2} (\|\mathbf{y}\|_2 + \|f^\theta(\theta; \mathbf{x})\|_2) \cdot \|\nabla_\theta f^\theta(\theta; \mathbf{x})\|_2 \\ &\leq \frac{1}{2} (1 + 1) \cdot 1 \\ &= 1 \end{aligned}$$

となる。よって、 $\mathcal{L}_{MSE}$ は1-リブシツ連続な関数である。□

一方、一般的に分類問題で用いられるクロスエントロピーは、勾配を一定以下に抑えることができないため、1-リブシツ連続にならないことに注意が必要である。

表 1 DP-SGD の比較

アルゴリズム	プライバシー	計算速度	メモリ効率	汎用性
SGD	×	○	○	○
Vanilla DP-SGD[5]	○	×	○	○
Multiple DP-SGD[4]	○	○	×	○
DP-SGD with LC	○	○	○	×

4. 議論

DP-SGD および DP-SGD with LC を比較した結果を表 1 に示す。ここで Vanilla DP-SGD は、ノルムクリッピングを愚直に直列に計算する方法 [5]、Multiple DP-SGD は、Tensorflow-Privacy でも採用されているモデルを複製し並列化を図る方法 [4]、DP-SGD with LC は提案方式を表す。

4.1 計算速度

Vanilla DP-SGD は、ノルムクリッピング時に並列計算ができないことから、計算時間を要してしまう。一方、Multiple DP-SGD は並列性を損なわずに計算可能であることから、高速に動作する。DP-SGD with LC は、各層の入力のノルム抑制のために特異値分解を用いると計算時間を要してしまうが、Power Iteration アルゴリズムを用いることで最大特異値を高速に計算できることから、Multiple DP-SGD とほぼ同等の計算時間が期待できる。

4.2 メモリ使用量

Vanilla DP-SGD や DP-SGD with LC はモデルの並列呼び出し等は不要であることから、メモリ使用量は SGD と比べさほど変化はない。しかしながら、Multiple DP-SGD はメモリ使用量が大きく、大規模データへの適用が難しい。

4.3 汎用性

Vanilla DP-SGD や Multiple DP-SGD では、モデルの構造に制約がない。しかしながら、DP-SGD with LC では、損失関数やニューラルネットワークの構造に制約が生じることから、特定のタスクでしか利用できない制限がある。

5. 終わりに

本稿では、差分プライバシーな確率的勾配降下法として、リプシッツ連続を満たす損失関数に基づく具体的なアルゴリズムを検討した。このアルゴリズムはノルムのクリッピングを必要としないことから、高速かつ通常の深層学習と変わらないメモリ使用量での動作が期待でき、特に大規模モデルへの適用可能性が見込める。

今後の課題として本提案方式の数値実験があげられる。各層の入力のノルムを抑制することで、どの程度有用性に影響を及ぼすかを検討する必要がある。

参考文献

- [1] Pytorch. <https://pytorch.org/>.
- [2] Tensorflow. <https://www.tensorflow.org/>.
- [3] tensorflow vectorized map. https://www.tensorflow.org/api_docs/python/tf/vectorized_map.
- [4] TensorflowPrivacy. <https://github.com/tensorflow/privacy>.
- [5] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.
- [6] Ashish Agarwal and Igor Ganiev. Auto-vectorizing tensorflow graphs: Jacobians, auto-batching and beyond, 2019.
- [7] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 464–473. IEEE, 2014.
- [8] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pp. 265–284. Springer, 2006.
- [9] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, Vol. 9, No. 3-4, pp. 211–407, 2014.
- [10] Bolin Gao and Laca Pavel. On the properties of the softmax function with application in game theory and reinforcement learning, 2018.
- [11] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, 2013.
- [12] R. V. Mises and H. Pollaczek-Geiringer. Praktische verfahren der gleichungsaufloesung. ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik, Vol. 9, No. 1, pp. 58–77, 1929.
- [13] Takeru Miyato, Toshiaki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.
- [14] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, 2010.
- [15] Pranav Subramani, Nicholas Vadivelu, and Gautam Kamath. Enabling fast differentially private sgd via just-in-time compilation and vectorization, 2020.