Vol.2020-DPS-182 No.22
Vol.2020-CSEC-88 No.22
2020/3/12

分散機械学習における勾配の異常検知を用いたブロックチェーンシステムの提案とその実現性の評価

長友 誠[1]　油田 健太郎[2]　岡崎 直宣[2]　朴 美娘[1]

概要：近年，IoT デバイスやモバイルデバイスの普及によって生成されるデータ量が増加している．それらを深層学習などの機械学習で学習する際に 1 つのサーバで行うと計算量が大きくなるため，サーバ下にあるエッジノードが勾配を求めサーバがそれを統合する分散機械学習が研究されている．しかし，エッジノードが異常な勾配をサーバへ送信すると学習モデルの識別精度が低下する可能性がある．そこでパブリック型ブロックチェーンを用いた分散機械学習が研究されており，既存研究では勾配の異常を検知するために計算ノードが類似度計算によって異常な勾配を排除していた．しかし，Adversarial Examples を学習した勾配など，類似度が高くても学習モデルの識別精度が下がる勾配が存在し，また，攻撃者が多ければ異常な勾配を排除することも難しい．そこで本論文では，プライベート型ブロックチェーンを用い，ブロックチェーン管理者であるサーバと計算ノードで異常検知を行うことで勾配の異常を検知し，サーバが異常な勾配を送信するノードを排除することで学習モデルの精度低下を防ぐシステムを提案する．また，提案システム内の異常検知を評価することでシステムの実現可能性を示す．

キーワード：分散機械学習，ブロックチェーン，勾配，異常検知

Proposal and Evaluation of Blockchain System using Anomaly Detection Method of Gradients on Decentralized Machine Learning

Makoto Nagatomo[1]　Kentaro Aburada[2]　Naonobu Okazaki[2]　Mirang Park[1]

Abstract: Recently, amount of data generated by IoT devices and mobile devices has been increasing. When training the data by deep learning, the distributed machine learning is proposed. Here, the server integrates gradients computed by edge nodes. However, it is possible for the accuracy of learning model to decreas when edge nodes send anomaly gradients. Hence, the public blockchain system is proposed. In this system, the computation node excludes anomaly gradients by calculating similarity between the gradient and the rest of others. However, there exist gradients that the accuracy of learning model decreases and the similarity is high such as gradients trained adversarial example. Therefore, in this paper, we propose a private blockchain system that the server and the calculation node detect anomaly gradients, and prevents decreasing the accuracy by excluding the nodes which send anomaly gradients. In addition, we evaluate the anomaly detection methods in order to show the possibility of the proposed system.

Keywords: distributed machine learning, blockchain, gradient, anomaly detection

1. はじめに

近年，機械学習の発展が目覚ましく，2024 年までに管理職の仕事の約 69%が AI にとって代わると予測されている [1]．一方，IoT デバイスやモバイルデバイスの普及に

[1] 神奈川工科大学
Kanagawa Institute of Technology
[2] 宮崎大学
University of Miyazaki

よって生成されるデータの増大 [2] によって機械学習の需要も増加しており，そのようなビッグデータを学習することで複雑な識別を行うことができる教師あり学習の深層学習 [3] が注目されている．深層学習において，入力に対する識別を出力する学習モデルはパラメータを持ち，学習時は与えられた学習データからパラメータの勾配を求め，それを用いてパラメータを更新する．

そのようなモデルの学習システムとして，1 つのサーバが各エッジノードから学習データを集めそれらを学習することでモデルを生成することが多いが，サーバに学習計算が集中するため，サーバの計算能力が高い必要がある．

そこで，各エッジノードが勾配を計算し，サーバがそれらの平均勾配を用いてパラメータを更新する分散機械学習が研究されている [4], [5], [6]．分散機械学習は各ノードがパラメータを把握する必要があるため，“white-box 攻撃” が成立する．機械学習における white-box 攻撃 1 つとして，学習モデルに対する入力に微小なノイズを加えることでその識別を誤らせる Adversarial Examples(A.E.) 攻撃がある [7], [8]．また，分散機械学習においては，勾配を計算するノードとパラメータを更新するサーバが別であるため，攻撃ノードが他ノードと異なる勾配をサーバへ送信するビザンチン将軍問題が生じる [9]．その結果，学習モデルの識別精度が低下する．そこで，本研究では学習モデルの識別精度を低下させる勾配を “異常な勾配” と呼ぶこととする．

そこで，本研究の目的は学習モデルの識別精度を低下させる勾配を排除できる分散機械学習システムの提案とする．

関連研究として，ビザンチン将軍問題への対策を行う分散機械学習が提案されている [10]．この研究では，他の勾配との距離が離れている勾配を排除する．また，ブロックチェーン (BC: Blockchain)[11] を用いた分散機械学習システムが提案されている [12]．これは，中央集権のサーバを持たないパブリック型 BC による分散機械学習であり，BC ネットワークに参加するノードが送信する平均勾配とコサイン類似度が低い勾配を排除する．

しかし，上記 2 つの研究では，勾配間の違いのみを用いて異常な勾配を排除しているため，学習モデルにとって異常であるかを考慮しておらず，十分に異常な勾配を排除できていないと考えられる．また，BC ネットワークに攻撃者が多数存在すると異常な勾配を排除できない問題がある．

そこで本研究では，BC ネットワークのノードの出入りを管理するサーバが存在するプライベート型 BC による分散機械学習システムを提案する．BC のブロックを生成する全ての計算ノードが BC に記録された正常・異常な勾配と，最新の学習モデルに対する正常・異常な勾配を学習した異常検知モデルを作成し，サーバがそれらを用いて勾配の異常検知を行うシステムとなる．過去の勾配の正常・異常から勾配の異常を判別するため，攻撃ノード数が多くても異常な勾配を排除でき，学習モデルの識別精度低下を防ぐことができる．

また，提案システムの実現可能性を評価するため，異常な勾配として A.E. を学習した勾配を作成し，勾配の異常検知を行う．今回は初期段階の実験として，1 つの計算ノードが最新の学習モデルに対する正常・異常な勾配を学習した異常検知モデルを作成したと仮定した実験を行う．

以降，2 章で関連研究を述べ，3 章で提案システムについて述べる．4 章で異常な勾配の作成とその異常検知の実験を行い，最後に 5 章でまとめとする．

2. 関連研究

2.1 深層学習

深層学習 [3] は，教師あり学習の一つであり，入力データに対する特徴を自動的に学習することで複雑な識別が可能となる．入力データ $\mathbf{x}$ と t 回の更新を終えた最新のパラメータ $\mathbf{w}_t$ を持つ学習モデルを関数 $f(\mathbf{x}, \mathbf{w}_t)$ として表すことができ，各クラスの確率を出力する．その中で最大の確率を持つクラスが識別結果となる．学習の際は，$f(\mathbf{x}, \mathbf{w}_t)$ と真のクラス $\mathbf{y}$ の損失関数 $L(f(\mathbf{x}, \mathbf{w}_t), \mathbf{y})$ を定義し，学習データ D から損失関数の勾配を以下の式で算出する．

$$\Delta \mathbf{w}_t = \sum_{\mathbf{x} \in D} \frac{\partial L}{\partial \mathbf{w}_t} \tag{1}$$

パラメータを更新するときは，(1) 式で求めた勾配を用い以下の式で更新する．

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \Delta \mathbf{w}_t \tag{2}$$

ただし，η は学習速度を決める学習率である．

具体的な学習モデル $f(\mathbf{x}, \mathbf{w}_t)$ については，多層のニューラルネットワークを用いることが多く，DNN(Deep Neural Network) とも呼ばれる．

2.2 分散機械学習

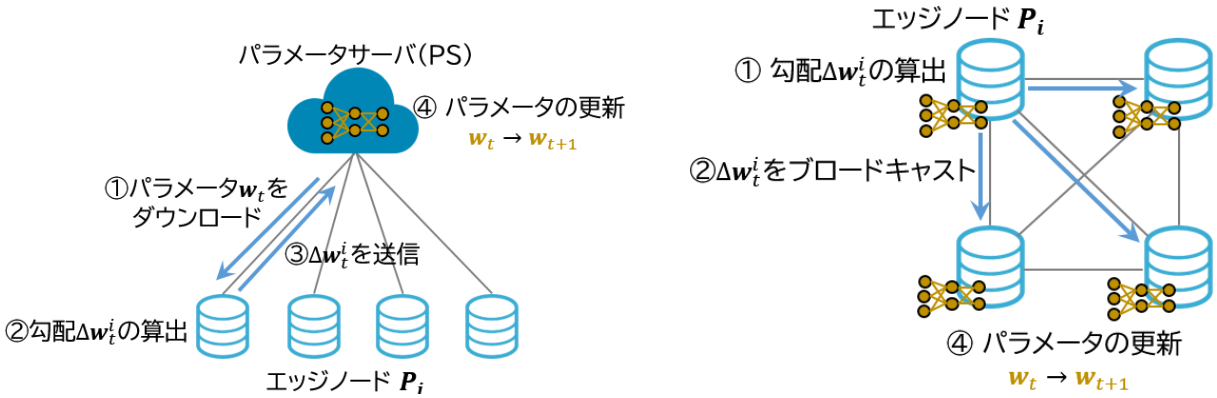
分散機械学習 (**図 1a**) では，それぞれのエッジノードが学習モデルのパラメータをパラメータサーバ (PS: Parameter Server) からダウンロード，勾配を計算し，それを PS へ送信する．PS ではそれらを統合することでパラメータを更新する [4], [5], [6].

具体的には，各エッジノード $P_i (i \in [1, N])$ が学習データ D_i を学習するとき，パラメータ $\mathbf{w}_t$ を更新するための勾配を，以下の式で求める．

$$\Delta \mathbf{w}_t^i = \sum_{\mathbf{x} \in D_i} \frac{\partial L}{\partial \mathbf{w}_t} \tag{3}$$

これらを受け取った PS は，$\Delta \mathbf{w}_t^i$ の平均を用いて以下の式のようにパラメータを更新する．

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{1}{N} \sum_{i=1}^{N} \Delta \mathbf{w}_t^i \tag{4}$$



(a) 中央集権型の分散機械学習システム　(b) 非中央集権型の分散機械学習システム

図 1: 分散機械学習システムの種類

ただし，η は学習の速度を決める係数である．

分散機械学習は，各エッジノードがパラメータ $\mathbf{w}_t$ を既知としているため，“white-box 攻撃”が成立する．その一つとして，Adversarial Examples(A.E.) 攻撃がある [7]．A.E. 攻撃は学習モデル $f(\mathbf{x}, \mathbf{w}_t)$ に対する入力 $\mathbf{x}$ に微小なノイズを加えることで識別を誤らせる手法である．その中で，A.E. を高速に作成する方法として FGSM(Fast Gradient Sign Method) が提案されている [8]．FGSM では，損失関数 $L(f(\mathbf{x}, \mathbf{w}_t), \mathbf{y})$ を用いて以下の式のように $\mathbf{x}$ にノイズを加える．

$$\mathbf{x} \leftarrow \mathbf{x} + \epsilon \ \mathrm{sign}\left(\frac{\partial L}{\partial \mathbf{x}}\right) \tag{5}$$

ただし，ϵ はノイズの量を表す．(5) 式を繰り返すことで特定のクラス $\mathbf{y}$ へ誤分類させる A.E. が作成できる．

また，分散機械学習においては，勾配を算出するノードとパラメータを更新する PS が別であるため，攻撃ノードが他のノードと異なる勾配を PS へ送信する，ビザンチン将軍問題 [9] が生じる．その結果，学習モデルの識別精度が低下する．

2.3 ビザンチン将軍問題に対する分散機械学習

攻撃ノードが他の勾配と距離が離れた勾配を PS へ送信した際に，それを排除する分散機械学習 Krum[10] が提案されている．Krum では，最初に PS が各エッジノード $P_i (i \in [1, N])$ から受け取った勾配 $\Delta \mathbf{w}_t^i$ に対して以下の式でスコア $s(i)$ を求める．

$$s(i) = \sum_{i \rightarrow j} ||\Delta \mathbf{w}_t^i - \Delta \mathbf{w}_t^j||^2 \tag{6}$$

ただし，$i \rightarrow j$ は $\mathbf{w}_t^i$ から距離が近い $n - F - 2$ 個のノードを表す．F は想定攻撃ノード数を表す．パラメータ更新時は以下の式のように最小の $s(i)$ を持つノードの勾配を用いる．

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\left(\Delta \mathbf{w}_t^{\ \arg\min_i s(i)}\right) \tag{7}$$

しかし，勾配間の距離のみで勾配の選択を行っているため，学習モデルにとって異常かどうか考慮されていない．また，攻撃ノード数が多ければ攻撃ノードの勾配が選択される確率が上がり，学習モデルの識別精度低下が起こる．

2.4 ブロックチェーンを用いた分散機械学習

2.4.1 ブロックチェーン

ブロックチェーン (BC: Blockchain)[11] とは，BC ネットワークに参加しているノード全体で合意形成を行うことでデータの保存を行う分散型台帳技術である．データを保持するノード全体が同じデータを共有しており，ネットワークに参加しているノードが誰でもデータを見れる，透明性がある．また，追加のデータを保存する際は，データをネットワーク上にブロードキャストし，複数の計算ノードと呼ばれるノードがブロック (ブロードキャストされたデータを集めたもの) を追加するためにある条件を満たす計算を行い，計算が最も早かった計算ノードのブロックが BC に追加される．ブロックををチェーンのように前ブロックと関連付けてつなげることによって改ざん困難性を確保している．ビットコインでは，前ブロックのハッシュ値とランダムな値 (Nonce 値) をブロックに追加し，そのブロックのハッシュ値の先頭 16 桁が 0 になることができればブロックとして認められる．

BC の特長として，1 つのサーバに全てのデータを集約する必要が無く，管理者が必要ない分散型のデータベースである点が挙げられる．これはパブリック型 BC と呼ばれる．また，管理者が存在するプライベート型 BC も存在する．

2.4.2 パブリック型ブロックチェーンを用いた分散機械学習

非中央集権型の分散機械学習 (**図 1b**) として，パブリック BC を用いた分散機械学習システムである，Learn-

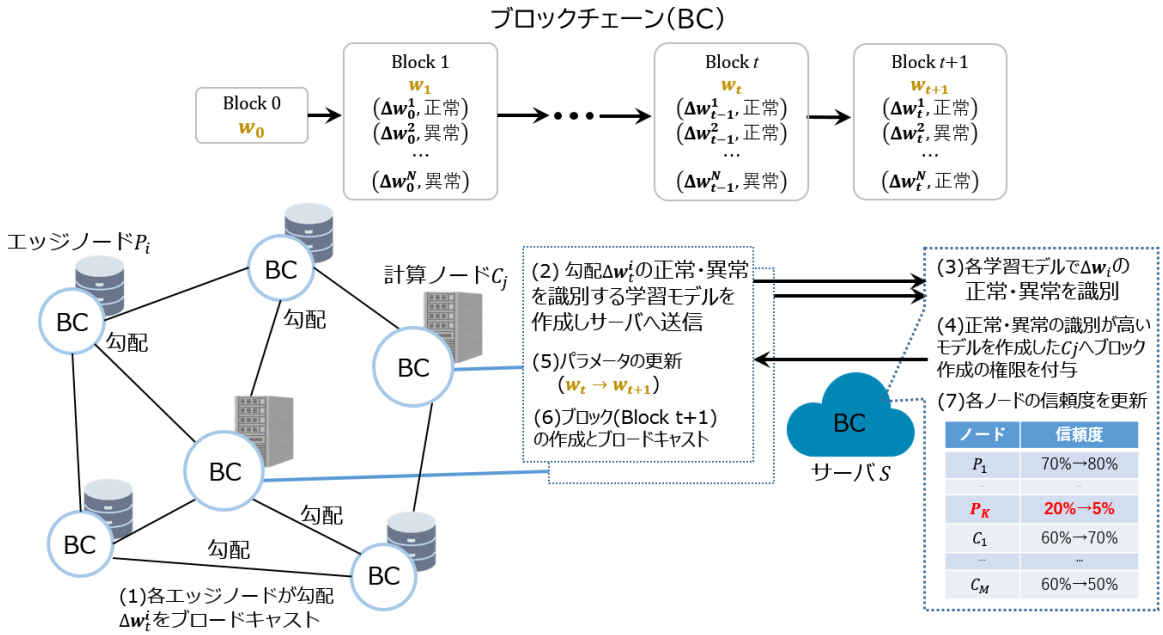


図 2: 提案システムにおけるモデルの学習手順

ingChain[12] が提案されている．LearningChain では，BC ネットワーク上のブロックに，更新されたパラメータが随時記録される．各エッジノード $P_i(i \in [1, N])$ は BC に保存されている最新のパラメータ $\mathbf{w}_t$ を用いて勾配 $\Delta \mathbf{w}_t^i$ を算出し，BC ネットワーク上へブロードキャストする．そして，ブロックの追加を行う計算ノードがそれらの平均を以下の式で計算する．

$$average = \frac{1}{N} \sum_{i=1}^{N} \Delta \mathbf{w}_t^i \tag{8}$$

その後に，$average$ とのコサイン類似度を以下の式のように計算する．

$$similarity_i = \frac{average \cdot \Delta \mathbf{w}_t^i}{||average|| \cdot ||\Delta \mathbf{w}_t^i||} \tag{9}$$

$similarity_i$ は -1 から 1 の値をとり，1 に近ければ正の相関，-1 に近ければ負の相関を表す．加えて，高いコサイン類似度を持つ勾配を l 個数 (N より小さい個数) 選び，それらの平均を用い以下の式でパラメータ更新を行う．

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{1}{l} \sum_{i=1}^{l} \Delta \mathbf{w}_t^i \tag{10}$$

最後に，計算ノードは $\mathbf{w}_{t+1}$ と勾配 $\Delta \mathbf{w}_t^i$ が記録されたブロックを BC に追加する．

しかし，このシステムでは，コサイン類似度が高く学習モデルの識別精度を低下させる勾配を排除できない．また，Krum[10] と同様に攻撃ノード数が多ければ学習モデルの識別精度の低下を防げない問題がある．

3. プライベート型ブロックチェーンを用いた分散機械学習システム

BC ネットワークのノードの出入りを管理するサーバが存在するプライベート型 BC による分散機械学習システムを提案する．各計算ノードが BC から過去の勾配の正常・異常と最新の学習モデルに対する正常・異常な勾配を学習した異常検知モデルを作成し，サーバがそれらを統合した異常検知結果を求めることで異常な勾配を排除するシステムとなる．過去の正常・異常な勾配を学習した異常検知モデルを用いることで，攻撃ノード数が多くても異常な勾配を排除でき，学習モデルの識別精度低下を防ぐことができる．また，学習モデルの識別精度低下をさらに防ぐため，サーバが各ノードの信頼度を算出し，信頼度の低いノードを BC ネットワークから排除する．

3.1 学習手順

本システムにおけるモデルの学習手順は以下の通りである (図 **2**).

(1) 勾配計算フェーズ：
各エッジノード $P_i(i \in [1, N])$ は学習データと BC に保存されている最新のパラメータ $\mathbf{w}_t$ を用いて勾配 $\Delta \mathbf{w}_t^i$ を算出し，BC ネットワークへブロードキャストする．

(2) 異常検知モデル作成フェーズ：
ブロック生成に参加したい計算ノード $C_j(j \in [1, K])$ は，BC に記録された正常・異常な勾配と，最新の学習モデルに対する正常・異常な勾配を学習した異常検知モデル M_j を作成し，サーバ S へ送信する．

(3) 異常検知フェーズ：
サーバ S は各 M_j を用いて各勾配 $\mathbf{w}_t^i$ の正常・異常を

判別する．その判別結果を R_j とする．

(4) ブロック作成権限付与フェーズ：
各勾配 $\mathbf{w}_t^i$ について，全ての C_j が正常・異常を判別する割合が多い方をその異常検知結果として求める．全ての勾配の異常検知結果を R とし，R と最も近い判別を行った C_k へ R と共にブロック作成権限を与える．

(5) パラメータ更新フェーズ:
C_k は R より，正常な勾配のみを用いてパラメータを $\mathbf{w}_t$ から $\mathbf{w}_{t+1}$ へ更新する．

(6) ブロック作成フェーズ
C_k は $\mathbf{w}_{t+1}$，各勾配とその正常・異常判別結果 ($\mathbf{w}_t^i$, 正常・異常) を記録した Block $t+1$ を作成し，BC ネットワークへブロードキャストする．

(7) 信頼度更新フェーズ:
S は各 R_j と R に基づいて各 P_i, C_j の信頼度 $conf_{P_i}$, $conf_{C_j}$ を更新する．

手順 (7) において，$conf_{P_i}$ については，R で Δw_t^i が正常と判断されていれば上昇し，異常と判断されていれば減少する．$conf_{C_j}$ は R と R_j 間で同じ正常・異常の判別数が半分以上であれば上昇，それより小さければ減少する．ここで，信頼度が低いノードは BC ネットワークから除外する．

Algorithm1 に上記手順の疑似コードを示す．計算ノードの異常検知モデルを集め，各勾配の正常・異常を判定することで高い判別率の異常検知が可能となる．また，最も判別率の高い異常検知モデルを作成した計算ノードがブロックを作成できるインセンティブ構造となる．

3.2 異常な勾配と異常検知モデルの作成

提案システムを実現するためには，まず，異常な勾配を作成し，3.1 節の手順 (2) における各計算ノードの異常検知モデルが作成できるか確認する必要がある．

異常な勾配として他の勾配にノイズを加えた勾配や正常でない学習で算出された勾配が考えられる．前者は勾配間の距離を用いる Krum[10] やコサイン類似度を用いる LearningChain[12] によって容易に検知ができる．しかし，後者の中ではそれらの方法で検知が難しい勾配があると考えられる．例えば，A.E. を学習した勾配が考えられ，理由として人間にとって異常な学習でも，学習モデルにとっては正常な学習であるためである．

異常検知モデルとして，勾配の特徴を人間で判別することは難しいため，生成モデルを用いた異常検知が有効であると考えられる．

4. 実験

提案システムの実現可能性を評価するため，A.E. による異常な勾配の作成と異常検知モデルによる正常・異常な勾配を判別し，異常な勾配の排除を確認する実験を行う．今

Algorithm 1 プライベート型のブロックチェーンを用いた異常な勾配を排除する分散機械学習システム

Network Initialization: エッジノード $P_1, ..., P_N$ と計算ノード $C_1, ..., C_K$，サーバ S は peer-to-peer ネットワークを組む．C_j は勾配の正常・異常 (1 または 0) を出力する学習モデル M_j を作成できる．

Parameter Initialization: S は学習モデルのパラメータ $\mathbf{w_0}$ を初期化し，ネットワークへブロードキャストする．

$t \leftarrow 0$
while do
  (1) 勾配計算フェーズ
  P_i : Block t から $\mathbf{w_t}$ を取得する．
  P_i : 勾配 $\Delta\mathbf{w}_t^i$ を算出する．
  P_i :$\Delta\mathbf{w}_t^i$ をブロードキャストする．
  (2) 異常検知モデル作成フェーズ
  C_j :M_j を作成する．
  C_j :M_j を S へ送信する．
  (3) 異常検知フェーズ
  for $j \in [1, K]$ **do**
    S :$R_j \leftarrow ()$
    for $i \in [1, N]$ **do**
      S :$R_j \leftarrow add(R_j, M_j(\Delta\mathbf{w}_t^i))$
    end for
  end for
  (4) ブロック作成権限付与フェーズ
  $R \leftarrow ()$
  for $j \in [1, K]$ **do**
    if $|\{i|R_j[i] = 1, i \in [1, N]\}| \geq \frac{N}{2}$ **then**
      $R \leftarrow add(R, 1)$
    else
      $R \leftarrow add(R, 0)$
    end if
  end for
  $k \leftarrow \arg\max_j(|\{i|R[i] = R_j[i], i \in [1, N]\}|)$
  S :C_k に R を送信する．
  (5) パラメータ更新フェーズ
  C_k : $R' = \{i|R[i] = 0\}$
  C_k : $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\frac{1}{|R'|}\sum_{i \in R'} \Delta\mathbf{w}_t^i$ (η は学習率)
  (6) ブロック作成フェーズ
  C_k :$\mathbf{w}_{t+1}$, ($\mathbf{w}_t^i$, 正常 or 異常) 記録した Block $t+1$ を作成する．
  C_k :Block $t+1$ をブロードキャストする．
  (7) 信頼度更新フェーズ
  S :R を用いて $conf_{P_i}$ を更新する．
  S :(R_j, R) を用いて $conf_{C_j}$ を更新する．
  $t \leftarrow t + 1$
end while
return

回の実験では，初期段階として 1 つの計算ノードが最新の学習モデルに対する正常・異常な勾配のみを学習した異常検知モデルを作成したと仮定し，そのモデルを用いて異常検知を行う．

実装環境として，OS は ubuntu，CPU が Intel Xeon W-2135，GPU として NVIDIA GP106 を搭載したデスクトップパソコンを用いる．開発言語として，Python を用い，学習モデルとしてライブラリ keras で提供されている InceptionV3[13] を用いる．InceptionV3 は ImageNet[14] から取得した画像を学習しており，299×299 ピクセルのカラー画像を 1,000 クラスへ分類する CNN(Convolutional Neural Network) である．本実験では InceptionV3 の初めの層である畳み込み層のパラメータ (864 個) の勾配のみについて異常検知を行う．

4.1 Adversarial Examples による異常な勾配の作成

A.E. を用いて異常な勾配を作成する．A.E. を学習した勾配の作成にあたり，A.E. を作成するため，原画像として ImageNet から収集したブドウに属するの画像 2,000 枚を用意した．また，A.E. の作成については FGSM[8] を用い，90%以上の確率でクラス hen(めんどり) と識別されるよう作成した．

上記の A.E. を用い，正常な勾配と A.E. を学習した勾配を以下の手順で作成する．

(1) 正常な勾配として，原画像を学習した勾配を算出する．
(2) A.E. を学習した勾配として，(原画像，原画像の A.E.) を学習した勾配を算出する．ただし，A.E. のクラスは hen として学習する．
(3) 正常な勾配と A.E. を学習した勾配のコサイン類似度を確認する．

上記の手順で勾配を作成した結果，正常な勾配と A.E. を学習した勾配との平均コサイン類似度が 0.97 となり，高い正の相関があることが分かった．

また，正常な勾配と A.E. を学習した勾配を用いてパラメータ更新を行ったときの，識別結果の変化率を**表 1** に示す．A.E. を学習した勾配を用いてパラメータの更新を行うと正常な勾配を用いたパラメータ更新と比べ識別精度が低下していることが確認できる．

以上より，異常な勾配として，原画像を学習した正常な勾配とのコサイン類似度が高く，原画像の識別精度を低下させる勾配を作成できた．

4.2 正常・異常な勾配の判別

4.1 節で作成した正常な勾配と異常な勾配を判別するため，異常検知モデルとして生成モデル Autoenconder[15] を用いる．Autoencoder は NN(Neural Network) の一種である．入力層と出力層のユニット数を同じにし，中間層を小さいユニット数とすることで，中間層で自動的に次元削減を行い入力の特徴を学習する．学習データに類似したデータを入力することで類似した出力が得られる．

表 1: 予測結果の変化率

	正しいクラス	異なるクラス
正常な勾配	100%	0%
A.E. を学習した勾配	89.2%	10.8%

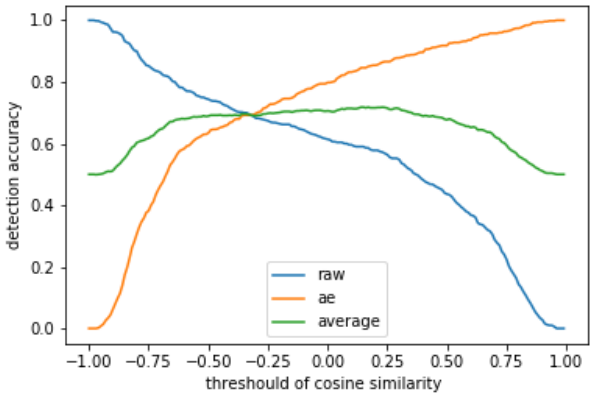


図 3: Autoencoder による勾配の判別率

ここで，4.1 節で作成した正常な勾配と異常な勾配はコサイン類似度が高いため，Autoencoder に正常な勾配を入力するとコサイン類似度が 1.0 に近い勾配が出力され，A.E. を学習した勾配を入力するとコサイン類似度が −1.0 に近い勾配が出力されるよう学習し，モデルを生成する．勾配の正常・異常の判別には，Autoencoder へ入力する勾配と出力される勾配のコサイン類似度が閾値以上であれば正常，閾値より小さければ異常であると判別する．その閾値として，正常な勾配と異常な勾配の判別率の平均が最大となる値を用いる．

その閾値を求めるため，入力層と出力層のユニット数を 864，中間層を 1 つだけを持つ Autoencoder を用意した．中間層のユニット数は 128，活性化関数として sigmoid 関数を用いる．この Autoencoder を異常な勾配を判別するモデルとして生成するため，4.1 節で作成した正常な勾配と A.E. を学習した異常な勾配を各 1,500 個学習させた．残りの各 500 個の勾配はテストデータとし，生成した Autoencoder の判別率算出に利用する．

図 3 に，コサイン類似度の閾値を −1 から 1 と変化させたときの，正常な勾配の判別率 (raw)，A.E を学習した異常な勾配の判別率 (ae)，それらの平均判別率 (average) を示す．結果として，コサイン類似度の閾値が 0.14 のとき，最大の平均判別率が 71.8%となった．

4.3 異常な勾配の排除の確認

分散機械学習における異常な勾配の排除が可能か確認するため，攻撃ノード数を変化させたときの，正常な勾

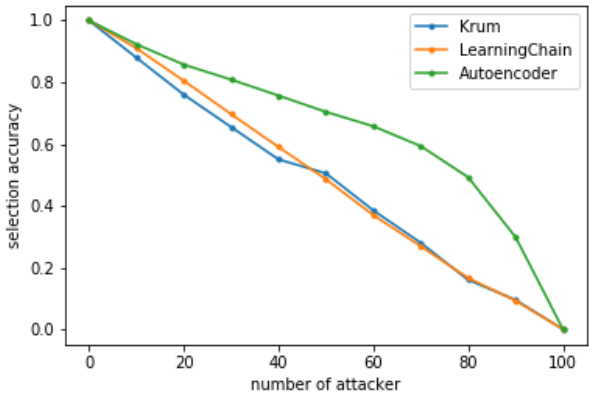


図 4: 攻撃ノード数ごとの正常な勾配の選択率

配の選択率を算出する．選択方法は Krum[10] と LearningChaing[12]，4.2 節で作成した Autoencoder を用い比較する．モデルの学習に参加しているノードの内，正常なノードは正常な勾配，攻撃ノードは A.E. を学習した異常な勾配を送信するノードと仮定する．具体的には，総ノード数を 100 と固定し，攻撃ノード数を 0〜100 の 10 区切りで変化させたときの勾配の正常・異常の選択率を以下の手順で求める．

(1) 4.2 節のテストデータから，正常なノード数の正常な勾配と攻撃ノード数の A.E. を学習した異常な勾配をランダムに選択する．
(2) Krum, LearningChain, Autoencoder による勾配の正常・異常の選択率を算出する．
(3) 手順 (1), (2) を 200 回繰り返し，平均選択率を算出する．

手順 (2) において，Krum の選択率については，式 (6),(7) より選択された勾配が正常であれば 1，異常であれば 0 とする．LearningChain については，各勾配をそれらの平均とのコサイン類似度が高い順に並べ，正常なノード数を選択し，その中で正常な勾配数の割合を選択率とする．Autoencoder による検知については，入力・出力のコサイン類似度が 4.2 節で定めた閾値 0.14 以上である勾配の中から正常ノード数の勾配を選択し，その中で正常な勾配の割合を選択率とする．

図 4 に Krum, LeraningChain，Autoencoder の正常・異常な勾配の選択率を示す．結果として，Krum, LearningChain 共に攻撃ノード数に比例して選択率が低下しているため，正常な勾配と A.E. で作成した異常な勾配をほぼ判別できていないことが分かる．Autoencoder については，全ての攻撃ノード数において，Krum, LearningChain より選択率が高くなった．よって，分散機械学習において A.E. を学習した勾配を排除するために Autoencoder による検知が有効であり，提案システムの実現可能性が示された．

5. おわりに

本論文では，分散機械学習においてエッジノードから送信される異常な勾配による学習モデルの識別精度低下を防ぐため，プライベート型ブロックチェーンを用いたシステムの提案を行った．また，提案システムの実現可能性を示すため，Adversarial Examples を学習した異常な勾配を作成し，生成モデル Autoencoder によって勾配の異常検知を行った．その結果，71.8%の判別率となり，関連研究より Autoencoder によって A.E. を学習した異常な勾配を排除できることが分かった．

今後の課題として，エッジノードの信頼度算出方法の検討や，提案ブロックチェーンシステムの実装と評価が挙げられる．

参考文献

[1] Gartner, Gartner Predicts 69% of Routine Work Currently Done by Managers will Be Fully Automated by 2024, `https://www.gartner.com/en/newsroom/press-releases/2020-01-23-gartner-predicts-69--of-routine-work-currently-done-b`(参照 2020-2-17).
[2] IDC,「2018 年 国内 IoT 市場 データエコシステム / Data as a Service に関わるプレイヤー分析」，2018.
[3] I. Goodfellow, Y. Bengio, A. Bourville, “Deep Learning,” MIT Press, 2016.
[4] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. A. Ranzato, A. Senior, P. Tucher, K. Yang, An. Y. Ng, “Large Scale Distributed Deep Networks,” Proc. of the 25th International Conference on Neural Information Processing Systems (NIPS 2012), vol. 1, pp. 1223-1231, 2012.
[5] T. Akiba, K. Fukuda, S. Suzuki, “ChainerMN: Scalable Distributed Deep Learning Framework,” Proc. of the 31th Internationl conference on Neural Information Processing Systems (NIPS 2017), 2017.
[6] P. Goyal, P. Dollar, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, K, He, “Accurete, Large Minibatch SGD: Training ImageNet in 1 Hour,” arXiv preprint arXiv:1706.02677, 2018.
[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, “Intriguing properties of neural networks,” arXiv preprint arXiv:1312.6199, 2013.
[8] I. J. Goodfellow, J. Shlens, C. Szegedy, “Explaining and Harnessing Adversarial Examples,” Proc. of International Conference on Learning Representations (ICLR 2015), 2015.
[9] L. Lamport, R. Shostak, M. Pease, “The byzantine generals problem,” ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 4, no. 3, pp. 382–401, 1982.
[10] I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett, “Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent,” Part of: Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 119-129, 2017.
[11] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[12] X. Chen, J. Ji, C. Luo, W. Liao, P. Li, "When Machine Learning Meets Blockchain - A Decentralized, Privacy-preserving and Secure Design," Proc. of 2018 IEEE International Conference on Big Data, pp. 1178-1187, 2018.
[13] InceptionV3, `https://keras.io/ja/applications/#inceptionv3`(参照 2020-2-17).
[14] ImageNet, `http://www.image-net.org/`(参照 2020-2-17).
[15] G. E. Hinton, R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," Science, vol.313, no. 5786, pp. 504-507, 2006.