一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
VLD2018-85,CPSY2018-95,RECONF2018-59(2019-01)

複数オンライン逐次学習コアによる教師なし異常行動検出の検討

伊藤　　怜† 　塚田　峰登†† 　近藤　正章††† 　松谷　宏紀†

† 慶應義塾大学 理工学部　〒 223-8522 神奈川県横浜市港北区日吉 3-14-1
†† 慶應義塾大学大学院 理工学研究科　〒 223-8522 神奈川県横浜市港北区日吉 3-14-1
††† 東京大学大学院 情報理工学系研究科　〒 113-8656 東京都文京区本郷 7-3-1
E-mail: †{rei,tsukada,matutani}@arc.ics.keio.ac.jp, ††kondo@hal.ipc.i.u-tokyo.ac.jp

あらまし　実環境では、正常データの特徴は時々刻々と変化し、その変化は環境ごとに様々である。異常検出問題では、正常データの環境変化に即時的に対応して正常モデルを学習することが重要である。また、あらゆる環境下を想定し、教師データを用意することは現実的に困難である。そこで、本論文ではオンライン逐次学習アルゴリズム OS-ELM(Online Sequential Extreme Learning Machine) をエッジ装置を想定した FPGA 上に実現し、教師なし異常行動検出を行なう。また、この異常行動検出器をマルチインスタンス化し、1 つの正常パタンごとに 1 つのインスタンスを割り当てて学習を行なうことを提案する。さらに、逐次学習時に正常な行動パタン数が変化したとき、異常検出器のインスタンス数を変化させ、動的に初期学習のやり直しを行なう手法を提案する。評価では、コマンド履歴ベンチマークを用いてなりすましの検出について検討した。正常な操作パタンに対して、なりすましを含んだ操作パタンの loss 値は 3,300 倍高くなり、検出の精度の高さを示した。また、正常な操作パタン数を変化させ、初期学習時の異常検出器のインスタンス数が正常な操作パタン数の変化に追従し、最適な数のインスタンス数に収束することを示した。
キーワード　オンライン逐次学習、異常行動検出、FPGA

A Case for Unsupervised Abnormal Behavior Detection Using Multiple Online Sequential Learning Cores

Rei ITO†, Mineto TSUKADA††, Masaaki KONDO†††, and Hiroki MATSUTANI†

† Faculty of Science and Technology, Keio University 3-14-1, Hiyoshi,Yokohama, JAPAN 223-8522
†† Graduate School of Science and Technology, Keio University 3-14-1, Hiyoshi,Yokohama, JAPAN 223-8522
††† Graduate School of Information Science and Technology, The University of Tokyo 3-14-1, Hongo,Tokyo, JAPAN 113-8656
E-mail: †{rei,tsukada,matutani}@arc.ics.keio.ac.jp, ††kondo@hal.ipc.i.u-tokyo.ac.jp

1. はじめに

近年、オンライン逐次学習による教師なし異常検出の重要性が高まりつつある。実際の現場において、正常データの特徴は環境や時間とともに変化する。異常検出問題では、正常データの特徴変化に即時的に対応して、正常モデルを学習することが求められる。オンライン逐次学習では、入力データを逐次的に学習し、パラメータを更新するため、データの特徴変化に追随でき、異常検出に有効であると言える。また、正常が時々刻々と変化する中で、正確な教師データをあらゆる状況下で用意することは難しい。そうした中、教師なしの異常検出は、教師データを必要とせず、正常モデルを学習できる。

異常検出には、外れ値検出、変化点検出、異常行動検出の 3 手法がある [1]。外れ値検出は、正常モデルから著しく外れたデータを検出し、変化点検出では、データの確率分布が変化した点を検出する。異常行動検出では、非定常の時系列データを扱うことができ、異常な行動パタンの出現を検出する。通常、異常行動検出において、正常な行動パタンは複数存在する。

本論文では、オンライン逐次学習アルゴリズム OS-ELM(Online Sequential Extreme Learning Machine) [2] を FPGA に実装し、教師なしの異常行動検出を行なう。OS-ELM は 3 層ニューラルネットワークを前提とするため、表現能力が低く、複数の正常な行動パタンの学習が難しい。そこで、それぞれの行動パタンに特化した学習を行なうために、OS-ELM の

マルチインスタンス化を提案する。また、逐次学習時に正常な行動パタン数が変化したときに、OS-ELM のインスタンス数を変化させ、動的に初期学習をやり直す手法を提案する。

本論文の構成は次に示す通りである。はじめに 2. 章で異常行動検出の既存研究、及び OS-ELM のアルゴリズムを示し、3. 章で OS-ELM による異常行動検出の手法、OS-ELM のマルチインスタンス化、初期学習の動的やり直しについて提案する。4. 章では実装の概要を示し、5. 章で異常検知の精度、初期学習のやり直しの追従速度、FPGA のリソース使用量について評価する。最後に 6. 章で本論文をまとめ、今後の課題について述べる。

2. 関 連 研 究

2.1 隠れマルコフモデルを用いた異常行動検出

隠れマルコフモデルは状態遷移を表現する確率モデルであり、非定常の時系列データの行動パタンをモデリングすることができる。この隠れマルコフモデルのパラメータを最適化するアルゴリズムに Baum-Welch アルゴリズム [3] がある。Baum-Welch は EM(Expectation-Maximization) アルゴリズムとして知られている。EM アルゴリズムは反復法の一種であり、期待値ステップと最大化ステップを繰り返すことで、確率モデルのパラメータを最尤推定する。そのため、EM アルゴリズムではパラメータが最適解に収束するための繰り返し処理が必要である。また、Baum-Welch アルゴリズムは、山登り法の一種であり、得られた解が最大値である保証はできない。隠れマルコフモデルによる行動モデリングは 1 つの行動パタンに対して行なうものであるが、AccessTracer [4] のように、混合隠れマルコフモデルを用いて複数の行動パタンをオンラインで学習する手法も提案されている。

2.2 ELM

OS-ELM の前に、その前提となるアルゴリズム ELM(Extreme Learning Machine) [5] について述べる。ELM は、バッチ学習アルゴリズムの一つであり、入力層、隠れ層、出力層の 3 層から構成される単層ニューラルネットワークの構造をとる。バッチサイズ k の n 次元の入力 $\boldsymbol{x} \in \boldsymbol{R}^{\boldsymbol{k\times n}}$ に対する m 次元の出力 $\boldsymbol{y} \in \boldsymbol{R}^{\boldsymbol{k\times m}}$ は $\boldsymbol{y} = G(\boldsymbol{x}\cdot\boldsymbol{\alpha}+\boldsymbol{b})\boldsymbol{\beta}$ として得られる。隠れ層のノード数を $\tilde{N}$ とすると、$\boldsymbol{\alpha} \in \boldsymbol{R}^{\boldsymbol{n\times\tilde{N}}}$ は入力層と隠れ層を結合する重みであり、$\boldsymbol{\beta} \in \boldsymbol{R}^{\boldsymbol{\tilde{N}\times m}}$ は隠れ層と出力層を結合する重みである。これらは任意の分布の乱数で初期化される。また、$\boldsymbol{b} \in \boldsymbol{R}^{\boldsymbol{\tilde{N}}}$ と G はそれぞれ隠れ層のバイアスと活性化関数を表している。

ここで、推論結果 $\boldsymbol{y}$ と教師データ $\boldsymbol{t} \in \boldsymbol{R}^{\boldsymbol{k\times m}}$ の誤差が 0 であると仮定すると次の等式が成り立つ。

$$G(\boldsymbol{x}\cdot\boldsymbol{\alpha}+\boldsymbol{b})\boldsymbol{\beta} = \boldsymbol{t} \tag{1}$$

また、隠れ層行列 $\boldsymbol{H} \equiv G(\boldsymbol{x}\cdot\boldsymbol{\alpha}+\boldsymbol{b}) \in \boldsymbol{R}^{\boldsymbol{k\times\tilde{N}}}$ を定義すると、重み $\boldsymbol{\beta}$ の最適解 $\hat{\boldsymbol{\beta}}$ は以下の式で求められる。

$$\hat{\boldsymbol{\beta}} = \boldsymbol{H}^{\dagger}\boldsymbol{t} \tag{2}$$

$\boldsymbol{H}^{\dagger}$ は $\boldsymbol{H}$ の擬似逆行列であり、SVD や QR 分解等の手法を用いて計算することができる。

ELM では重みの最適化は $\boldsymbol{\beta}$ のみ行なうため、計算コストの削減になる。また、$\hat{\boldsymbol{\beta}}$ は大域最適解であり、通常のニューラルネットワークで用いられる勾配計算での局所最適解への収束という問題を回避している。しかし、ELM はバッチ学習アルゴリズムであり、逐次学習ができないという欠点が存在する。新しいデータに対して学習を行なうためには、その都度過去の全データを含めて再学習を行なう必要がある。

2.3 OS-ELM

OS-ELM [2] は ELM を逐次学習に対応させた学習アルゴリズムであり、任意のバッチサイズで学習を行なうことができる。バッチサイズ k_i の i 番目の訓練データ $(\boldsymbol{x}_i, \boldsymbol{t}_i) \in \boldsymbol{R}^{\boldsymbol{k_i\times n}} \times \boldsymbol{R}^{\boldsymbol{k_i\times m}}$ が得られたとき、以下の損失を最小にする重み $\boldsymbol{\beta}_i$ を求める必要がある。

$$\left\| \begin{bmatrix} \boldsymbol{H_0} \\ \vdots \\ \boldsymbol{H_i} \end{bmatrix} \boldsymbol{\beta_i} - \begin{bmatrix} \boldsymbol{t_0} \\ \vdots \\ \boldsymbol{t_i} \end{bmatrix} \right\| \tag{3}$$

このとき隠れ層行列は、$\boldsymbol{H_i} = G(\boldsymbol{x_i}\cdot\boldsymbol{\alpha}+\boldsymbol{b})$ と表される。ここで、$\boldsymbol{K_i} \equiv \begin{bmatrix} \boldsymbol{H_0} \\ \vdots \\ \boldsymbol{H_i} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{H_0} \\ \vdots \\ \boldsymbol{H_i} \end{bmatrix}$ $(i \geqq 0)$ とすると、式 (3) は次式のように変形することができる。

$$\begin{aligned} \boldsymbol{\beta_i} &= \boldsymbol{\beta_{i-1}} + \boldsymbol{K_i^{-1}}\boldsymbol{H_i^T}(\boldsymbol{t_i} - \boldsymbol{H_i}\boldsymbol{\beta_{i-1}}) \\ \boldsymbol{K_i} &= \boldsymbol{K_{i-1}} + \boldsymbol{H_i^T}\boldsymbol{H_i} \end{aligned} \tag{4}$$

さらに、$\boldsymbol{P_i} \equiv \boldsymbol{K_i^{-1}}$ とすると、式 (4) は最終的に以下の式に変形される。

$$\begin{aligned} \boldsymbol{P_i} &= \boldsymbol{P_{i-1}} - \boldsymbol{P_{i-1}}\boldsymbol{H_i^T}(\boldsymbol{I} + \boldsymbol{H_i}\boldsymbol{P_{i-1}}\boldsymbol{H_i^T})^{-1}\boldsymbol{H_i}\boldsymbol{P_{i-1}} \\ \boldsymbol{\beta_i} &= \boldsymbol{\beta_{i-1}} + \boldsymbol{P_i}\boldsymbol{H_i^T}(\boldsymbol{t_i} - \boldsymbol{H_i}\boldsymbol{\beta_{i-1}}) \end{aligned} \tag{5}$$

ELM で逐次学習を行なうためには、その都度過去の全データを含めて再学習を行なう必要があった。OS-ELM では式 (5) が示すように、重み $\boldsymbol{\beta}$ とその中間結果である P は 1 つ前の学習によって得られる値から計算することができ、新しく生成された訓練データに対してのみ学習を行なえばよいことがわかる。

式 (5) において、計算のボトルネックとなっているのは、逆行列計算 $(\boldsymbol{I} + \boldsymbol{H_i}\boldsymbol{P_{i-1}}\boldsymbol{H_i^T})^{-1}$ の部分である。ここの行列のサイズは $k_i \times k_i$ であるため、バッチサイズを 1 に固定することによって、逆行列計算を単純な逆数の計算に置き換えることができ、計算量の削減になる。また、このことにより逐次学習時の主要な演算は行列積のみになり、ハードウェア実装における並列化を容易にしている。[6] では OS-ELM を FPGA に実装することで、高速化と省面積化を達成し、エッジデバイスとして異常検出問題に有効であることを示している。

3. 提　　案

本論文では、OS-ELM による教師なしの異常行動検出を行なうことを考える。3.1 節ではその手法を説明し、正常な行動パ

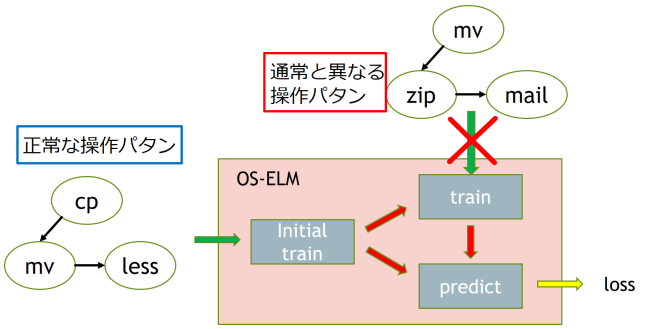

図 1　OS-ELM によるなりすましの検出 (正常な操作パタンが 1 つ)

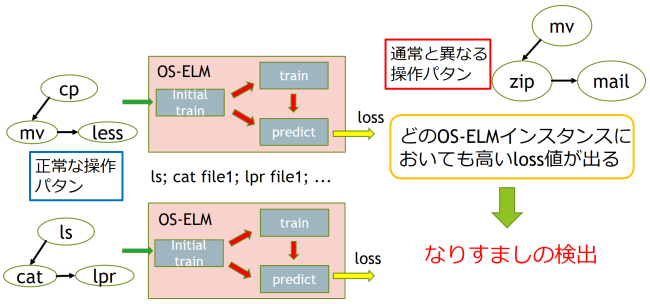

図 2　OS-ELM のマルチインスタンス化

タンが複数存在するときの単一インスタンスの限界について述べる。3.2 節ではその問題点を解決する方法として、OS-ELM のマルチインスタンス化について提案する。3.3 節では逐次学習時に正常な行動パタン数が変化したときに、動的に初期学習をやり直す手法を提案する。

3.1　OS-ELM による異常行動検出

異常行動検出では、非定常の時系列データを扱う。そこで、現在の状態から次の時刻の状態への遷移を記録する状態遷移表を用意する。つまり、この状態遷移表の要素数は状態数 × 状態数ということになる。OS-ELM による学習は状態遷移表を入力データとして行なう。

例として、UNIX コマンド列からのなりすましの検出について考える。ここでは、コマンドの操作パタンを状態遷移表として学習する。図 1 に正常な操作パタンが 1 つのときの OS-ELM への適用を示す。まず、正常な操作パタンを訓練データとして初期学習させる。その後、逐次学習では、loss 値が閾値を超えるデータについては異常な操作パタンと判断され、学習を行なわず、正常な操作パタンに対してのみ逐次学習を行なう。推論では loss 値を計算する。

しかし、異常行動検出では通常、正常な行動パタンは複数存在する。コマンド列の例では、ユーザや作業内容によって正常な操作パタンが異なるため、それぞれの正常な操作パタンを正常と判断できる推論器でなければ、異常検出を行なうことはできない。図 1 で正常な操作パタンは 1 つであるが、2 つ存在するときには、OS-ELM は 2 つの操作パタンに対して 1 つの正常モデルを学習する。このとき、OS-ELM は表現能力が低いため、どっちつかずの学習となり、正常な操作パタンについてもある程度 loss 値が上がり、検出の精度が低下する。この点に関して 5.1 節で評価を行なう。さらに、正常のパタン数が多いときや、正常同士の特性が大きく異なる場合では、検出の精度は大きく低下し、異常検出としての機能は期待できない。そこで、OS-ELM のマルチインスタンス化について提案する。

3.2　OS-ELM のマルチインスタンス化

複数の正常な操作パタンを学習するとき、単一の OS-ELM インスタンスでは限界がある。これは、OS-ELM をマルチインスタンス化し、正常な操作パタンごとに OS-ELM を割り当て、学習を行なうことにより解決できる。図 2 では正常な操作パタンが 2 つ存在するため、OS-ELM インスタンスを 2 つ用意し、操作パタンごとに学習を行なっている。一般にインスタンス数が N のとき、複数インスタンスによる学習の方法を以下に示す。

（1）　入力データ x に対して全インスタンスで推論を行なう

（2）　N 個の内、最小の loss 値を返したインスタンスに x を学習させる

異常検出では、この最小の loss 値が閾値を超えるとき異常であると判断される。このことは、どの OS-ELM インスタンスにおいても高い loss 値が出ることを表しており、どの正常な操作パタンにも当てはまらなかったことを意味している。このように OS-ELM のマルチインスタンス化によって、正常な操作パタンが複数存在するときも、高い精度でなりすましを検出することができる。

操作パタンごとに学習を行なうためには、操作パタンをクラスタに分ける必要がある。本論文では、クラスタリングの方法として K-means 法を利用した。K-means 法では最適なクラスタ数が予めわかっている必要がある。クラスタ数が既知でない場合は、DBSCAN [7] や SUBCLU [8] 等を採用する。K-means クラスタリングは初期学習時に利用することになるため、逐次学習時にもクラスタ数は一定で、正常な操作パタン数の変化に追従することができない。つまり、動的に初期学習をやり直す必要がある。

3.3　複数異常検知インスタンスの再構成

逐次学習時に正常な操作パタン数が変化したとき、OS-ELM のインスタンス数を変化させ、動的に初期学習をやり直しを行なう。図 3 は初期学習のやり直しの流れを表している。図 3 のように、初期学習時に現れなかった正常な操作パタンが逐次学習時に現れた場合、クラスタ数を 1 つ増やし、K-means クラスタリングを行なう。クラスタリングの結果から、クラスタごとに OS-ELM インスタンスを用意し、初期学習をやり直すという流れである。

ここで、逐次学習時の正常な操作パタン数の変化による初期学習やり直しの手法を以下に示す。

（1）　入力データを buffer に格納していく

（2）　buffer が full になったら、buffer のデータを K-means でクラスタリングする

（3）　定量評価による最適クラスタ数を決定する

（4）　最適クラスタ数が変化した場合、初期学習をやり直す

(1) における buffer size は K-means クラスタリングの全データ数となり、本論文では buffer size = 280 としている。(2) の

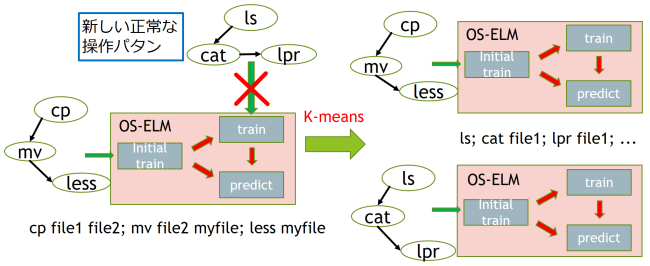

図 3　初期学習やり直しの流れ

クラスタリングは、現在のクラスタ数、現在のクラスタ数 ±1 の 3 通りについて行なう。(3) では (2) の 3 通りのクラスタリング結果の定量評価を行なう。評価の指標としては、**PseudoF** [9] を用いる。PseudoF は以下の式で表される。

$$PseudoF = \frac{(T-P)/(k-1)}{P/(n-k)} \tag{6}$$

ここで、T は全データの距離二乗和 (全データの平均と各データの距離の二乗和) であり、P はクラスタ内距離二乗和 (全クラスタについて、クラスタの重心とクラスタの各データの距離二乗和を求め、その和をとったもの) である。また、k はクラスタ数、n は全データ数を表している。PseudoF はクラスタ内の凝集性、クラスタ間の離散性を表現するため、クラスタリングの評価の指標として用いられる。この値が大きいほどクラスタリングとして良い結果であると言える。(4) では (3) の PseudoF による評価で最適なクラスタ数が変化していた場合、初期学習のやり直しを行なう。最適なクラスタ数の変化は正常な操作パタン数の変化を意味するためである。

4. FPGA 実装の概要

本論文では、実装対象として Xilinx 社の Xilinx Zynq UltraScale+ MPSoC ZCU102 ボードを想定した。FPGA デバイスは Zynq UltraScale XCZU9EG であり、4 コアの ARM Cortex-A53 プロセッサコアを内蔵している。また、開発環境としては C/C++による高位合成ツールである Vivado HLS 2016.4 を用いている。

図 4 に Zynq SoC を想定した実装の概要図を示す。CPU では、入力データ $\boldsymbol{x} \in \boldsymbol{R}^n$ に対して K-means クラスタリングを実行する。その後、クラスタごとに OS-ELM による初期学習を行ない、重み $\boldsymbol{\alpha}, \boldsymbol{\beta}$、及び中間結果 $\boldsymbol{P}$ を計算する。他には重みの reset、save、load を行なう。FPGA では、ソフトウェアで計算したクラスタごとの重み、中間結果を読み込み、クラスタごとに逐次学習または推論を行なう。逐次学習では重み $\boldsymbol{\beta}$ と中間結果 $\boldsymbol{P}$ の値を更新し、推論では損失値 $loss = L(G(\boldsymbol{x} \cdot \boldsymbol{\alpha})\boldsymbol{\beta}, \boldsymbol{x})$ を計算する。損失関数 $L(\boldsymbol{x}, \boldsymbol{y})$ は二乗平均誤差を用いた。マルチインスタンスの OS-ELM での loss 値は、各 OS-ELM での推論における loss 値の最小値である。

5. 評　　価

本章では、OS-ELM による異常行動検出の精度、正常な行動

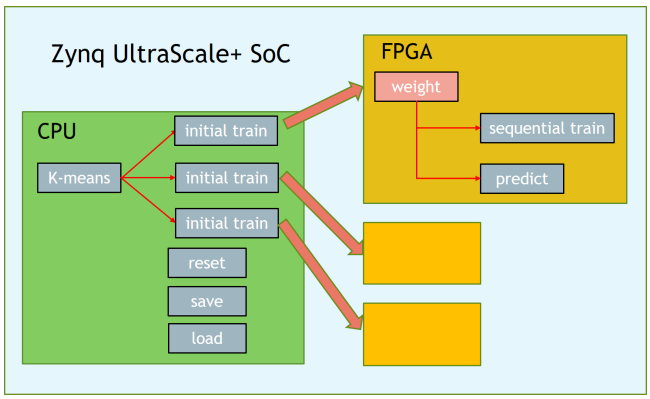

図 4　Zynq SoC を想定した実装の概要図

パタン数が変化するときの初期学習やり直しの追従速度、及び FPGA リソースの使用率について評価する。全評価に共通して、以下のスペックのサーバを評価マシンとして使用している。

- CPU: Intel Core i5-7500 4 コア (最大 3.4GHz)
- DRAM: 8GB
- Storage: SSD 480GB(RAID1)
- OS: Ubuntu 17.10(64bit)

本評価ではコマンド履歴のベンチマークを用いて、なりすましを検出することを考える。全評価で Schonlau データセット [10] を利用した。Schonlau データセットにはユーザ 50 人によるコマンド履歴が含まれており、ユーザ 1 人あたり 15,000 コマンドの履歴から構成されている。全ユーザに対して最初の 5,000 コマンドは全て正規ユーザによるものであり、残りの 10,000 コマンドは侵入者によるコマンドがランダムに挿入されている。入力データとなる状態遷移表の要素数はコマンドの種類数 × コマンドの種類数となり、コマンド数が足りないため、コマンドの遷移確率からコマンド履歴を水増しして評価に用いた。OS-ELM の入力層と出力層のノード数は 1024、隠れ層のノード数は 256 としている。初期学習に必要なデータ数は 280 であり、逐次学習時の buffer size も 280 とした。

5.1 OS-ELM による異常行動検出の精度

本節では、(1) まず正常な操作パタン数が 1 つのときの単一 OS-ELM インスタンスによるなりすまし検出の精度の評価を行なう。(2) その後、単一インスタンスの限界を示すために、正常な操作パタン数が複数存在するとき、単一インスタンスによる検出精度とマルチインスタンスによる検出精度を比較する。

(1) では 50 人のユーザの中からユーザ 9 を選び、ユーザ 9 の正常な操作パタン (最初の 5,000 コマンド) を学習させた。図 5 になりすまし検出の精度の評価結果を示す。図 5 では、ユーザ 9 の正規のコマンド履歴となりすましによるコマンド列を含んだコマンド履歴の loss 値の比較を行なっている。また、ユーザ 9 は 156 種類のコマンド履歴から構成されており、156×156(24,336) の状態遷移表が必要となる。図 5 の横軸は 1 つの状態遷移表の作成に要したコマンド遷移の回数を表している。縦軸の loss 値は対数目盛となっている。

いずれも正規のコマンド履歴となりすましを含むコマンド履歴の loss 値の差は非常に大きく、検出の精度は高いと言え

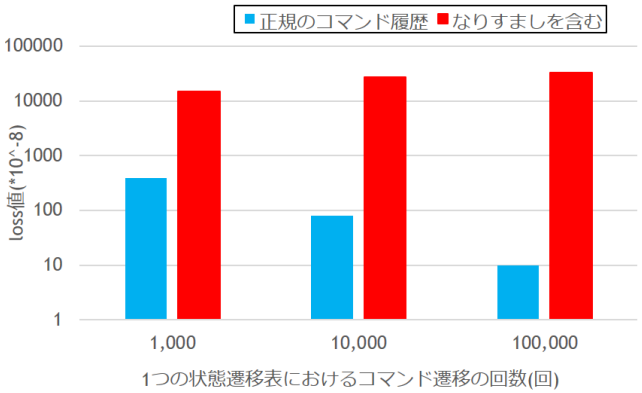

図 5　正規のコマンド履歴となりすましを含むコマンド履歴の loss 値の比較

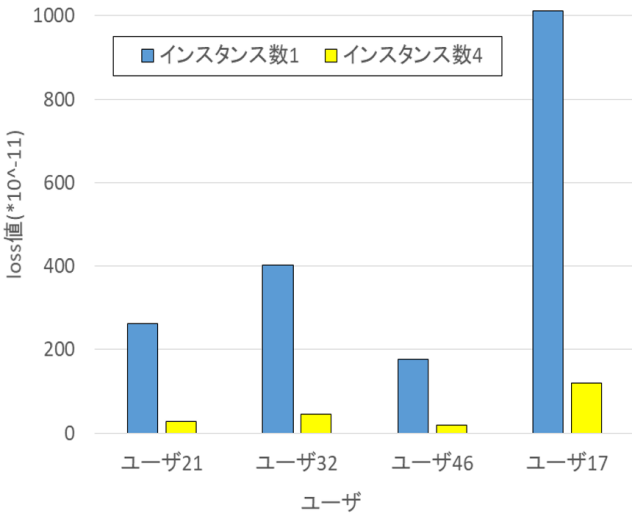

図 6　単一インスタンスとマルチインスタンスの loss 値の比較

る。また、コマンド遷移の回数ごとに loss 値の差を比較したとき、1,000 回のときは約 40 倍、10,000 回のときは約 350 倍、100,000 回のときは約 3,300 倍となっており、コマンド遷移の回数が多いほど検出の精度が高くなっていることがわかる。これは、遷移回数が少ないと状態遷移表がスパースとなり、正常と異常の差異が出にくくなるためである。

(2) ではユーザ 21, 32, 46, 17 を選択した。ここでは 4 人のユーザの正常な操作パタンに対して、単一インスタンスで学習を行なったときと 4 つのインスタンスを用意し、操作パタンごとに学習を行なったときの loss 値の差を比較した。図 6 にユーザごとに loss 値を比較したものを示す。

図 6 から、インスタンス数 1 の loss 値とインスタンス数 4 の loss 値の差は 8.4 倍から 9.0 倍となっており、検出の精度に大きな差があることがわかる。これは、OS-ELM が表現能力の低さから、単一のインスタンスに複数の正常パタンを覚えさせると検出の精度が低下することを表している。

5.2　正常な操作パタン数が変化するときの追従速度

本節では、逐次学習時に正常な操作パタン数が変化するときの初期学習の動的やり直しの追従速度について評価する。ここ

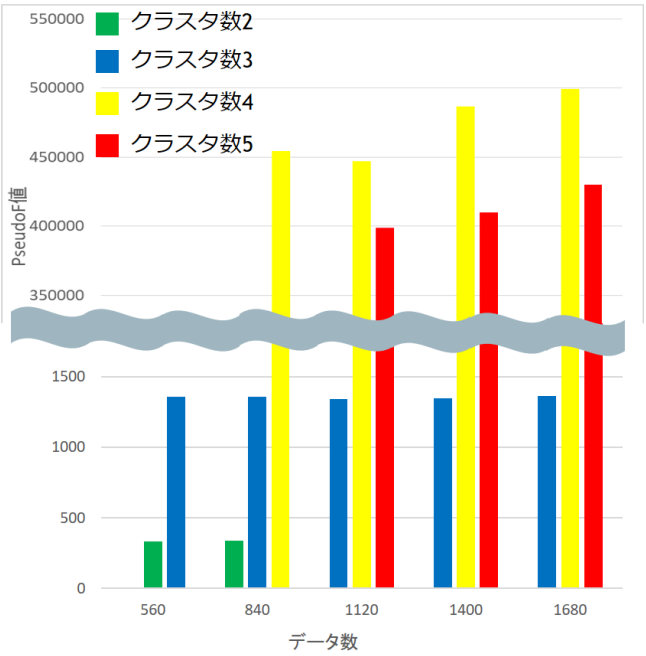

図 7　K-means クラスタリングにおける PseudoF 値の比較

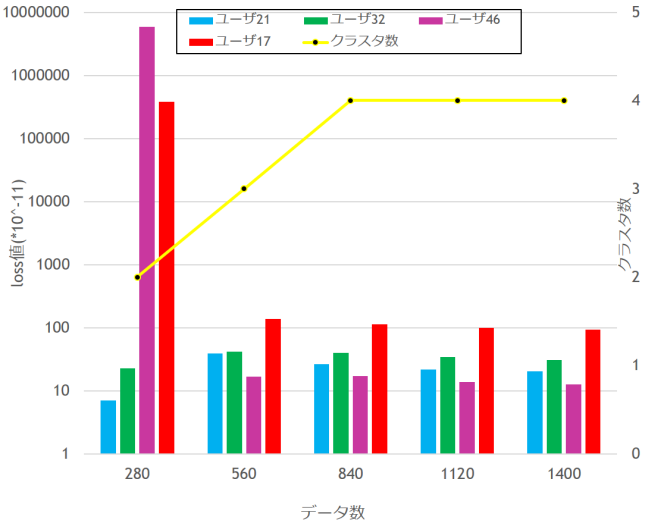

図 8　クラスタ数の変化に対するユーザごとの loss 値の変化

では 5.1 節と同様、ユーザ 21, 32, 46, 17 を選択した。まず、ユーザ 21 とユーザ 32 の正常な操作パタンに対して、クラスタ数 2 で K-means クラスタリングを実行し、クラスタごとに OS-ELM インスタンスを割り当て、初期学習を行なった。逐次学習では、ユーザ 21, 32, 46, 17 の正常な操作パタンを順に入力データとした。これは、正常な操作パタン数が 2 ⇒ 4 になったことを意味している。

図 7 では、逐次学習時に buffer が full になる度に行われる 3 通りの K-means クラスタリングにおける PseudoF 値の比較をしている。逐次学習フェーズで最初のクラスタリングは、現在のクラスタ数 2、2 ± 1 の 3 通りについて行なっている。ここで、クラスタ数が 1 のときは PseudoF の式が適用できないため、PseudoF = 0 としている。このとき、クラスタ数が 3 のと

きに PseudoF が最大となるため、クラスタ数、及び OS-ELM のインスタンス数を 3 として初期学習のやり直しを行なう。2 回目のクラスタリングは、現在のクラスタ数 3、3±1 の 3 通りについて行なう。このときはクラスタ数が 4 のときに PseudoF は最大となる。ゆえにクラスタ数、OS-ELM のインスタンス数を 4 として初期学習のやり直しが行われる。同様に、3 回目のクラスタリングは、現在のクラスタ数 4、4±1 の 3 通りについて行なう。ここでは PseudoF を最大にするクラスタ数は変化しないため、初期学習のやり直しは発生しない。4 回目以降のクラスタリングでも、クラスタ数が 4 のときに PseudoF は最大値をとり続ける。すなわち、最適なクラスタ数つまり正常な操作パタン数は 4 であると判断されたことがわかる。

図 8 はクラスタ数の変化に対する 4 人のユーザ 21, 32, 46, 17 の loss 値の変化を表している。クラスタ数 2 のときはユーザ 21, 32 に対して初期学習を行なったため、ユーザ 46, 17 の loss 値は高くなっている。また、逐次学習時の正常な操作パタン数の変化に対応し、クラスタ数が最適な数に近づくにつれて、全ユーザの loss 値が下がっていることがわかる。しかし、正常な操作パタン数は 4 に変化したにも関わらず、クラスタ数 3 のとき既に全ユーザの loss 値が低く出ている。これは状態遷移表の要素数が非常に大きく、遷移表の要素の多くは 0 あるいは 0 に近い値となること、またユーザごとの操作パタンの差異が大きくないことによるものである。状態遷移表の要素数は全ユーザのコマンド種類数の 2 乗と非常に大きくなるが、各ユーザの操作パタンを学習するには必要でない部分が多く存在するため、状態遷移表の圧縮について検討する必要がある。

5.3 FPGA 実装のリソース

最後に OS-ELM の FPGA 実装のリソース使用状況について示す。入力層と出力層のノード数は 1024、隠れ層のノード数は 256 である。

表 1　1 つの OS-ELM インスタンスのリソース使用状況

	BRAM	DSP	FF	LUT
合計	703	35	1196	2091
利用可能	1824	2520	548160	274080
使用率 (%)	38.54	1.39	0.22	0.76

表 2　8 つの OS-ELM インスタンスのリソース使用状況

	BRAM	DSP	FF	LUT
合計	5624	280	9568	16728
利用可能	1824	2520	548160	274080
使用率 (%)	308.33	11.11	1.75	6.10

表 1 には単一の OS-ELM インスタンス、表 2 には 8 つの OS-ELM インスタンスのリソース使用状況を示した。DSP, FF, LUT についてはまだリソースに余裕があると言えるが、BRAM についてはインスタンスが 8 つのとき、使用率が 100% を大きく超えてしまっており、今回の実装では BRAM の使用率の制約を満たすインスタンス数は 2 までとなる。オンボード SRAM や DRAM の活用は今後の課題である。

6. まとめと今後の課題

OS-ELM は 3 層ニューラルネットワークを前提とするため、表現能力が低く、複数の正常パタンの学習が難しい。しかし、異常行動検出において、正常な行動パタンは複数存在することが普通である。そこで、行動パタンごとに学習を行なうために OS-ELM のマルチインスタンス化について提案した。また、逐次学習時に正常な行動パタン数の変化したとき、動的に初期学習をやり直す手法を提案し、評価では初期学習時の OS-ELM のインスタンス数が正常な行動パタン数の変化に追従することを示した。

今後の課題としては、評価で述べたように、状態遷移表の圧縮が挙げられる。異常行動検出において、OS-ELM への入力は現在の状態から次の時刻の状態への遷移を表す状態遷移表となるため、状態遷移表の要素数は状態数 × 状態数となり、膨大な要素数となってしまう場合がある。しかし、状態遷移を表現するのに不要な部分が多く存在しており、実際には遥かに小さい状態遷移表で十分な場合が多い。圧縮の方法としては、過去の状態遷移のデータから水増しする方法や状態遷移の候補を絞り、有り得ない状態遷移を除外するベクトル化などが挙げられるが、アプリケーション依存な問題である。

文　献

[1] 山西健司，竹内純一，丸山祐子，“統計的異常検出 3 手法，” 情報処理，vol.46，no.1，pp.34–40，2005.

[2] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, “Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks,” IEEE Transactions on Neural Networks, vol.17, no.6, pp.1411–1423, Nov. 2006.

[3] 村上仁一，“Baum-Welch アルゴリズムの動作と応用例，” 電子情報通信学会 基礎・境界ソサイエティ Fundamentals Review，vol.4，no.1，pp.48–56，2010.

[4] 山西健司，データマイニングによる異常検知，共立出版，2010.

[5] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks,” Proceedings of the International Joint Conference on Neural Networks (IJCNN’04), pp.985–990, July 2004.

[6] M. Tsukada, M. Kondo, and H. Matsutani, “OS-ELM-FPGA: An FPGA-Based Online Sequential Unsupervised Anomaly Detector,” Proceedings of the International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms (HeteroPar’18), Aug. 2018.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD’96), pp.226–231, 1996.

[8] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, “Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications,” Proceedings of the International Conference on Management of Data (SIGMOD’98), pp.94–105, 1998.

[9] T. Calinski and J. Harabasz, “A dendrite method for cluster analysis,” Communications in Statistics, vol.3, pp.1–27, 1974.

[10] “Masquerading User Data”. `http://www.schonlau.net/intrusion.html`.