自己組織化マップを用いたネットワークトラフィックのエントロピーに基づく異常検出

二瓶 凌輔[1,a)]　青木 茂樹[1,b)]　宮本 貴朗[1]

概要：近年のサイバー攻撃は検出されにくいように攻撃の傾向を変化させており，攻撃を持続的に検出するためには，攻撃の変化に逐次対応する必要がある．そこで本稿では，逐次学習を行える自己組織化マップを異常検出に適用し，サイバー攻撃が変化する場合にも持続的に攻撃を検出できる手法を提案する．まず，トラフィックデータからパケットのサイズなどの特徴量を抽出し，特徴量ごとにエントロピーを算出して特徴ベクトルを生成する．次に，生成した特徴ベクトルを自己組織化マップで学習する．学習の際，自己組織化マップの勝者ノードと近傍のノードで，重みを付けて学習回数を算出する．抽出した特徴ベクトルに対して学習した自己組織化マップを適用し，選択された勝者ノードの学習回数が閾値未満の場合に異常を検出，閾値以上の場合に逐次学習を行う．実験では，MWS データセットおよび大阪府立大学ネットワークのトラフィックデータを用いて，本手法の有効性を確認した．

キーワード：ネットワーク異常検出，逐次学習，自己組織化マップ

Anomaly Detection Based on Entropy of Network Traffic using Self-Organizing Maps

Ryosuke Nihei[1,a)]　Shigeki Aoki[1,b)]　Takao Miyamoto[1]

Abstract: In recent years, since cyber attacks have changed, it is difficult to detect them. It is necessary to respond to those changes. In this paper, we propose a method for continuously detecting changing attacks using self-organizing map that can be sequential learning. First, we extract feature quantities such as packet size from traffic data and calculate entropy for each feature quantity, and generate feature vectors. Next, the generated feature vector is learned using a self-organizing map. And we calculate weighted learning times at winner node and nearby nodes in the self-organinzing map. After that, we extract feature vectors from new traffic data. We recognize the extracted feature vectors using self-organizing map. When the number of the selected winner node is less than threshold, we detect anomaly. In the experiment, we confirmed the effectiveness of our method using MWS dataset and traffic data of Osaka Prefecture University network.

Keywords: Anomaly Detection of Network, Online Learning, Self-Organizing Maps

1. はじめに

近年，インターネットの普及に伴ってサイバー攻撃による被害が増加し，社会的な問題となっている．サイバー攻撃を検出する手法として，ネットワーク上の不正なトラフィックを検出する侵入検知システム (IDS:Intrusion Detection System) の研究が盛んに行われている．IDS はシグネチャ型とアノマリ型の 2 種類に大別できる．シグネチャ型 IDS はあらかじめ異常を定義したパターンファイルに基づいて異常の検出を行う方式である．この方式には，パターンファイルに定義した異常は検出できるが，定義されていない異常を検出できない欠点がある．アノマリ

[1] 大阪府立大学大学院人間社会システム科学研究科
Graduate School of Humanities and Sustainable System Sciences, Osaka Prefecture University
[a)] sza04214@edu.osakafu-u.ac.jp
[b)] aoki@kis.osakafu-u.ac.jp

型 IDS[1], [2], [3], [4] は正常な通信のみを含むデータを学習し，そこから外れた状態を異常として検出する方式である．そのためアノマリ型 IDS では，パターンファイルに定義された攻撃だけでなく，定義されていない未知の攻撃も検出できる．

これまでに提案されている IDS では，事前に定義した異常や学習した結果に基づく異常は検出できるが，サイバー攻撃は常に変化しており，攻撃を持続的に検出するためには攻撃の変化に逐次対応する必要があると考えられる．文献 [5], [6] では，逐次的な学習が可能な自己組織化マップ（Self-Organizing Maps，以後 SOM とする）を異常検出に適用した手法を提案している．文献 [6] の手法では，SOM は逐次学習しているものの，決定木を逐次更新していないために決定木の更新前に検出精度が低下する可能性がある．

本稿では，パケットのヘッダから抽出した特徴量のエントロピーを求めて特徴ベクトルとし，特徴ベクトルを SOM で逐次学習しながら異常検出する手法を提案する．実験では，MWS データセットと大阪府立大学ネットワークのトラフィックデータに本手法を適用して有効性を確認した．

2. 関連研究

本研究に関連する従来研究として，アノマリ型 IDS の代表的な手法である文献 [1], [2], [3], [4] と逐次学習を行いながら異常を検出する手法である文献 [5], [6] について述べる．

文献 [1] では，ネットワークのトラフィックは複数の正常状態で表されると考え，複数の正常状態を定義し，各状態との違いから異常を検出する手法を提案している．この手法では，異常を含まないデータから単位時間当たりの ICMP や TCP パケット数等を計測してクラスタリングする．メンバが少ないクラスタは削除し全てのクラスタにおいて閾値以上のメンバ数となるまでクラスタリングを繰り返す．クラスタリング結果を正常状態として定義し，新たに観測されたデータから同様の特徴を抽出し，正常クラスタとの距離を基に異常の判別をしている．

文献 [2] では，複数の特徴量の組み合わせによる異常検出手法を提案している．この手法では，異常をトラフィック量の異常，通信手順の異常，通信内容の異常の 3 種類に分け，単位時間あたりのトラフィック量を数値化した特徴量，フロー毎のフラグの出現回数を数値化した特徴量，フロー内のパケットのペイロードのパターンの傾向を数値化した特徴量を学習用データからそれぞれ抽出する．そして新たなデータからこれらの特徴量を抽出し，学習用データの値と閾値以上離れている特徴量が存在する場合に異常であると判断する．

文献 [3] では，パケットトレースから一般的な通信の特徴量 70 種類とマルウェア特有の特徴量 25 種類の計 95 種類の特徴量を抽出し，DBSCAN を用いてクラスタリングしている．異常検出時にはパケットトレースから 95 種類の特徴量を抽出し，クラスタ内のデータとのユークリッド距離を算出する．算出した距離の最小値が閾値未満の場合は既知のマルウェアとして既存のクラスタに分類され，閾値以上の場合は未知のマルウェアとしている．

文献 [4] では，パケットのヘッダ情報から得られたエントロピーにより異常を検出する手法を提案している．この手法ではパケットを到達した順番に並べ，一定のパケット数で分割する．そして，分割したパケットから IP アドレスやポート番号など毎にパケット数を計測する．次に，パケットの出現確率を求め，求めた出現確率からエントロピーを算出する．その後，EMMM 法 (Entropy based Multi dimensional Mahalanobis distance Method) によりエントロピーの変化を確認し，変化が大きい時間を攻撃などが含まれている状態として検出している．この手法では学習および異常検出にかかる時間の短縮が課題となっている．

従来研究では，事前に定義した異常や学習した結果に基づいた異常は検出できていた．しかし，サイバー攻撃の変化については考慮されていない．攻撃を持続的に検出するためには，攻撃の変化に逐次対応させる必要がある．

文献 [5], [6] では逐次学習を行える SOM を用いることで学習したモデルを逐次更新しながら異常を検出する手法を提案している．文献 [5] では，プログラムを動作させた際に呼び出される Windows システムサービスの呼び出し列と入出力パラメータに設定される文字列を用いてマルウェアを検出する手法を提案している．正常なデータと異常なデータから文字列を抽出し，抽出した文字列から生成した特徴ベクトルを SOM に入力してマップを作成する．マップ上のノードが持つ参照ベクトルと抽出した特徴ベクトルとのユークリッド距離を算出する．算出した距離が最小のノードを勝者ノードとして選択する．勝者ノードとその近傍のノードの参照ベクトルを，抽出した特徴ベクトルで更新することにより逐次学習する．そして，選択した勝者ノードが異常データを学習していた場合に異常を検出する．

文献 [6] では，パケットのヘッダから抽出した特徴量を決定木と SOM で学習し，それらを組み合わせることで攻撃の検出と逐次学習を行う手法を提案している．SOM では学習時に参照ベクトルの更新とマップ上の各ノードで勝者ノードに選択された回数を算出している．決定木では正常と異常のラベル付けを行う．異常検出時には，勝者ノードに選択された回数を用いて異常を検出し，抽出した特徴量と決定木でラベル付けした結果を用いて SOM で逐次学習する．また，SOM の異常検出結果により定期的に決定木を再構築する．この手法では逐次学習に対応した SOM と決定木を定期的に再構築することでサイバー攻撃の変化に対応した持続的な攻撃の検出を実現している．しかし，決定木の更新を逐次実施していないことから決定木が攻撃

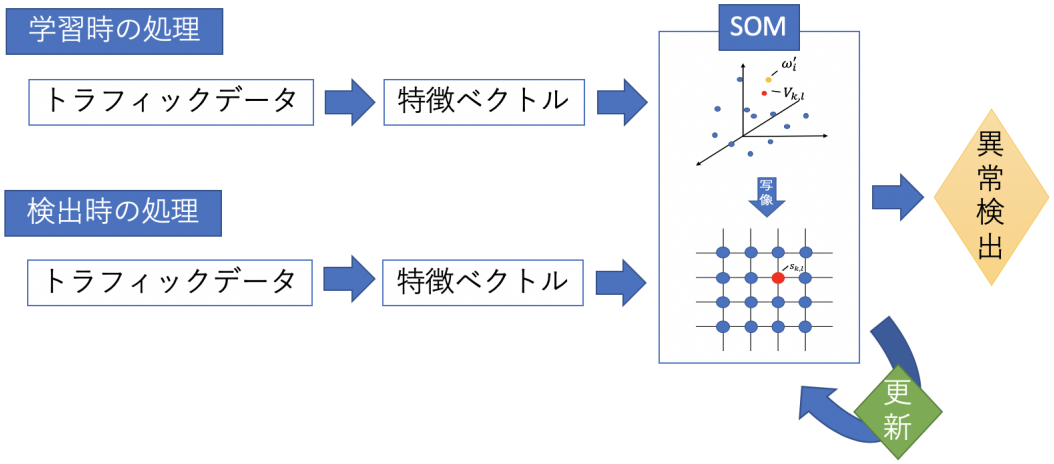

図 1　提案手法の概要
Fig. 1　Outline of Proposed Method.

の変化に対応できず精度が低下する可能性が考えられる．

3. 提案手法

本稿では，パケットのヘッダから抽出した特徴量のエントロピーを求めて特徴ベクトルを生成し，SOM により特徴ベクトルを逐次学習しながら異常を検出する手法を提案する．本手法では異常検出時に SOM で逐次学習を行うことにより，攻撃の変化に対応して精度を維持したまま未知の攻撃を検出できると考えられる．

本手法の異常検出手順について述べる．手法の流れを図 1 に示す．本手法は学習と検出の 2 つの処理に分かれている．学習時の処理では，学習用のトラフィックデータを一定のパケット数で分割し，各区間に含まれるパケットのバイト数などの 9 種類の特徴量を抽出する．抽出した特徴量からエントロピーを算出して特徴ベクトルを生成し，SOM で学習する．また，SOM 学習時に勝者ノードとして選択された回数に重みを付けて数えておく．

検出時の処理では，学習時とは異なるトラフィックデータから同様に特徴ベクトルを抽出し，学習済みの SOM に入力する．入力された特徴ベクトルと最も近い参照ベクトルを持つ SOM のノードを取得し，学習時に数えた重み付き学習回数が閾値未満となる場合に異常と判定する．また，閾値以上の場合に入力した特徴ベクトルを用いて SOM の参照ベクトルと重み付き学習回数を逐次更新する．

3.1 特徴ベクトルの抽出

ここではエントロピーを用いて特徴ベクトルを生成する手法について述べる．エントロピーを用いることで，例えば DoS 攻撃やスキャン攻撃など特定の IP アドレスやポート番号に対するパケットが増加する場合，送信先 IP アドレスやポート番号などの情報量が変化するために通常時のエントロピーとの差が顕著に生じると考えられる．そのため，本手法ではエントロピーを用いて特徴ベクトルを生成する．注目しているネットワークに対する異常を検知するために，ネットワークから収集したトラフィックデータを一定のパケット数 n で分割し，各区間で表 1 に示すパケットのバイト数などの 9 種類の特徴量を抽出する．次に，抽出した特徴量を用いてエントロピーを算出する．区間 i における j 番目の特徴量が H 種類出現した時，h 番目の特徴量の出現回数 $x_i^{j,h}$ から出現確率 $P_i^{j,h} = x_i^{j,h}/n$ を算出し，エントロピー $E_i^{j,h}$ を次式で算出する．

表 1　ヘッダから抽出した特徴量
Table 1　Packet Header Features.

送信元 IP アドレス
送信先 IP アドレス
送信元ポート番号
送信先ポート番号
パケットバイト数
プロトコル
TTL 値
フラグメント ID
フラグメント用フラグの状態

$$E_i^{j,h} = -P_i^{j,h} \log_2 P_i^{j,h} (1 \leq j \leq 9) \qquad (1)$$

その後，次式でエントロピーの総和 E_i^j を算出する．

$$E_i^j = \sum_h^H E_i^{j,h} \qquad (2)$$

収集した全てのトラフィックデータの区間からエントロピーを算出すると，抽出したエントロピーの値は，特徴量ごとに大きく異なると考えられる．そのため各特徴量の値の平均が 0，分散が 1 になるように次式で標準化する．

$$E_i'^j = \frac{E_i^j - \overline{E^j}}{\sigma_{E^j}} \qquad (3)$$

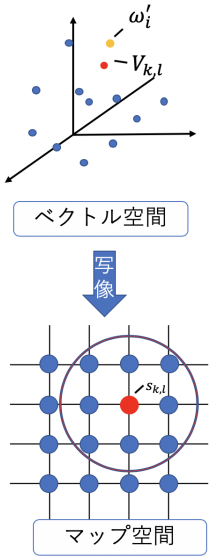

図 2　SOM の概要
Fig. 2　Overview of SOM.

ここで，$E_i^{'j}$ は標準化後のエントロピーを示し，$\overline{E^j}$ と σ_{E^j} は j 番目のエントロピーの平均と標準偏差を表す．そして，標準化したエントロピーで表すベクトル $\boldsymbol{\omega}_i = (E_i^{'1}, E_i^{'2}, \cdots, E_i^{'9})$ を特徴ベクトルとする．

3.2　SOM による学習

学習用のトラフィックデータから特徴ベクトル $\boldsymbol{\omega}_i'$ を抽出し，SOM で学習する．SOM は，コホネンによって提唱された教師なしニューラルネットワークである [7]．SOM の概要を図 2 に示す．SOM では入力された高次元データを性質の類似するデータ同士の距離が近くなるように，入力された高次元のベクトル空間を低次元のマップ空間上に写像する．マップ空間は人間が見やすいように 2 次元で用いられることが多く，本稿でも 2 次元の SOM を用いることとする．マップ空間上の各ノードは入力データと同次元の参照ベクトルを持ち，参照ベクトルと入力された特徴ベクトルとのユークリッド距離を用いることで類似度を算出する．

まず，マップ空間上にノード $s_{k,l}$（k,l は任意の整数）を並べる．SOM の各ノードは参照ベクトル $\boldsymbol{V}_{k,l}$ を持っている．SOM の学習では，抽出した特徴ベクトルと参照ベクトルとのユークリッド距離を算出する．算出したユークリッド距離の中で最小値をとる参照ベクトル $\boldsymbol{V}_{k,l}$ を持つノード $s_{k,l}$ を勝者ノードとする．勝者ノードとその周囲にあるノードの参照ベクトルを式 (4)〜(7) により更新する．

$$\boldsymbol{V}_{k,l} = \boldsymbol{V}_{k,l} + \theta_s * L * (\boldsymbol{\omega}_i' - \boldsymbol{V}_{k,l}) \tag{4}$$

$$L = L_0 * \exp(\frac{t}{\lambda}) \tag{5}$$

$$\theta_s = \exp(\frac{d^2}{2\sigma_s^2}) \tag{6}$$

$$\sigma_s = \sigma_{s0} * \exp(-\frac{t}{\lambda}) \tag{7}$$

ここで，$\boldsymbol{\omega}_i'$ は入力の特徴ベクトル，t は学習する特徴ベクトルの数，λ は学習した特徴ベクトルの総数，d は算出したユークリッド距離，L_0，σ_{s0} は学習 1 回あたりの参照ベクトルの更新率や更新する近傍領域の大きさを決定するパラメータである．そして，各ノードで勝者ノードに選択された回数を重みを付けて数える．ノード $s_{k,l}$ が勝者ノードとして選択された場合，学習回数 $\alpha_{k,l}$ に 1 加える．また，勝者ノードと近傍のノードでは以下の式に従って，学習回数 $\alpha_{k,l}$ に重みを付けて更新を行う．

$$\alpha_{k,l} = \alpha_{k,l} + \theta_s * L \tag{8}$$

以上の処理を全ての特徴ベクトルに対して行い，SOM で学習させる．

3.3　異常検出

新たに観測されたトラフィックデータを一定のパケット数 n で分割し，それぞれの区間で 9 次元の特徴ベクトル $\boldsymbol{\omega}_i''$ を抽出する．抽出した特徴ベクトルを用いて SOM で逐次学習と異常検出を行う．SOM では，学習した参照ベクトルの中から特徴ベクトル $\boldsymbol{\omega}_i''$ に最も近い参照ベクトル $\boldsymbol{V}_{k,l}$ を持つノードを選び，勝者ノード $s_{k,l}$ とする．勝者ノードに選択された場合，学習時に算出した重み付き学習回数 $\alpha_{k,l}$ を確認する．学習時に存在しないデータから特徴ベクトルを抽出した場合，選択される勝者ノードの重み付き学習回数は少なくなる．そのため閾値 θ 未満の場合に異常，閾値 θ 以上の場合に正常と判定する．異常検出の概要を図 3 に示す．また，正常と判定された特徴ベクトルを用いて学習時と同様に SOM の参照ベクトルと重み付き学習回数 $\alpha_{k,l}$ を逐次更新する．

4.　実験

本手法の有効性を確認するために MWS2018 データセットおよび大阪府立大学のネットワークで収集したトラフィックデータを用いて実験を行った．

4.1　MWS2018 データセットにおける実験

4.1.1　実験条件

MWS2018 データセット [8] 中の BOS データセットを用いて実験を行った．BOS データセットは，組織内ネットワークへの侵害活動を想定した動的活動観測のデータセットである．BOS データセットではマルウェアの進行度として 1 から 8 まで定義されている．進行度 1，2 のデータはマルウェアを実行したが C2 サーバとの通信は発生して

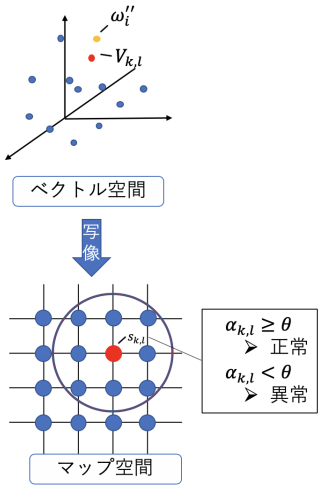

図 3　異常検出の概要
Fig. 3　Overview of Anomaly detection.

おらず，進行度 3，4，5 のデータは C2 サーバとの通信は発生したが C2 サーバとの通信は成立していない状態を示している．進行度 6，7，8 のデータは通信が発生し，C2 サーバとの通信も成立している状態である．本稿では，進行度 2 のトラフィックデータを異常が含まれていない正常なトラフィックデータ，進行度 7，8 のトラフィックデータを異常を含むトラフィックデータとして実験に用いた．実験では，C2 サーバとの通信が含まれている区間を異常，それ以外の区間を正常として実験を行った．各進行度における正常と異常の区間数を表 2 に示す．

SOM のマップサイズは 40 × 40 とし，SOM のパラメータ L_0，σ_{s0} はそれぞれ 0.0001，20，1 区間のパケット数 n は 100 パケット，SOM の学習回数は 15 とした．実験に用いた PC の CPU とメモリはそれぞれ，Intel Core i7-4790 と 16GB である．OS と開発言語は Ubuntsu16.04 と Python2.7.12 を使用した．

評価方法には ROC 曲線と F-measure，Precision，Recall，学習および異常検出に要した時間を用いた．ROC 曲線とは正常と異常を分類する際の閾値を変化させたときの TPR(True Positive Rate) と FPR(False Positive Rate) を計算し，プロットしたグラフである．ここで，TPR とは異常データを正しく異常とした割合を示し，FPR とは正常データを誤って異常とした割合を示す．すなわち ROC 曲線が左上側に近づくほど検出精度が高いことを示している．実験では 3.3 節の異常検出時の閾値を変化させ，ROC 曲線を描画した．F-measure は Precision と Recall の調和平均を用いて算出することができる．Precision は異常とした中で実際に異常データであった割合を示し，Recall は異常データを正しく異常とした割合を示す．

4.1.2　実験結果と考察

進行度 2 のトラフィックデータで学習し，進行度 8 のトラフィックデータでテストした際の ROC 曲線を図 4 に示す．閾値を 2.2 とした際，F-measure，Precision，Recall はそれぞれ 0.907，0.941，0.876 であった．進行度 2 のトラフィックデータには正常しか含まれていないため正常の特徴をうまく学習し，異常を検出することができていた．

正常と異常のマップ上での分布の違いを確認するための実験を行った．進行度 2，進行度 7，進行度 8 のトラフィックデータで 15 回目の学習をした際に，各ノードが勝者ノードとして選択された回数を調べた結果をそれぞれ図 5，図 6，図 7 に示す．図中 x 軸と y 軸は SOM のマップ，z 軸は勝者ノードとして選択された回数を表している．なお，SOM 学習時の初期状態については全て共通としている．全ての進行度で，青丸で囲ったノードが多く選択されていた．進行度 2 のトラフィックデータには正常の区間しか存在していないため，青丸で囲った部分のノードで正常の特徴を学習していることを確認できた．一方，進行度 7 や 8 のトラフィックデータでも青丸で囲ったノードが多い結果となったのは，これらのデータにも正常の区間が多く存在し，青丸で囲ったノードを勝者ノードとして選択していたためである．また赤四角で囲ったノードには，進行度 2 のトラフィックデータでは勝者ノードがほとんど存在せず，進行度 7 と進行度 8 のトラフィックデータには勝者ノードが存在する結果となった．進行度 7 では異常区間のうちの約 73%，進行度 8 では約 51%の区間が赤四角で囲ったノードを勝者ノードとしており，異常の特徴を学習していることを確認できた．以上の結果から正常と異常では異なるノードを勝者ノードとして学習していることを確認できた．

次に，SOM の逐次学習が提案手法に与える影響を確認す

表 2　進行度ごとの区間数
Table 2　Number of sections in each progress.

進行度 2	進行度 7	進行度 8
正常:7023/異常:0	正常:4203/異常:3283	正常:6505/異常:639

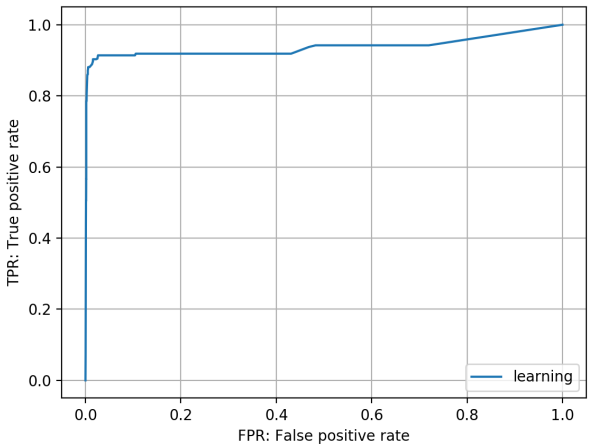

図 4　逐次学習時の ROC 曲線
Fig. 4　ROC curve when learning sequentially.

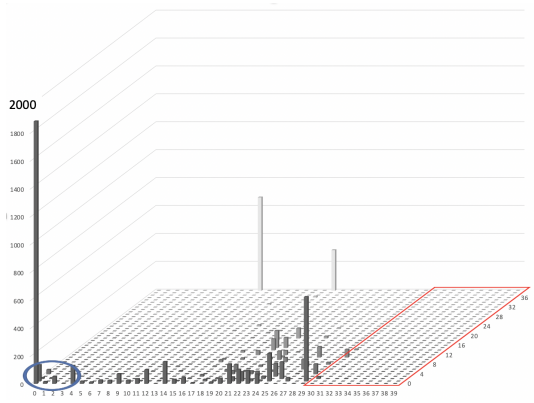

図 5　学習時の進行度 2 の勝者ノード

Fig. 5　Number of Winner node when progress 2 is learning.

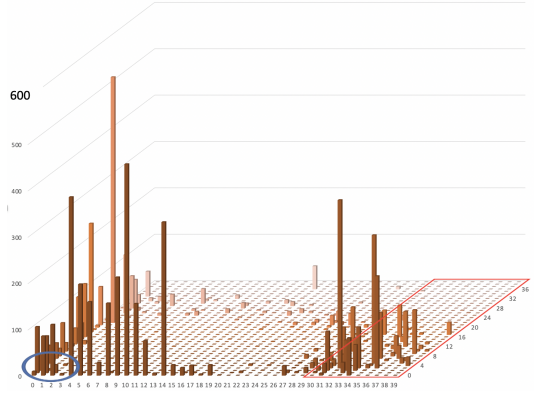

図 6　学習時の進行度 7 の勝者ノード

Fig. 6　Number of Winner node when progress 7 is learning.

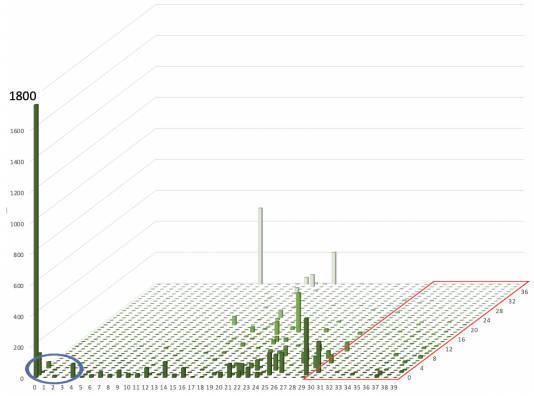

図 7　学習時の進行度 8 の勝者ノード

Fig. 7　Number of Winner node when progress 8 is learning.

る実験を行った．進行度 7 のトラフィックデータで学習し，進行度 2 のトラフィックデータでテストを行った後，進行度 8 のトラフィックデータでテストした際の F-measure，Precision，Recall を逐次学習を行った場合と行わなかった場合で調べた結果を表 3 に示す．閾値は 5.25 とした．逐次学習を行った場合，行わなかった場合と比較して F-measure と Recall の値が高くなり，Precision の値が同程度となった．進行度 7 のトラフィックデータで学習し，進行度 8 のトラフィックデータでテストを行った場合，閾値を最適

表 3　逐次学習に関する検出結果

Table 3　Result of Detection when learning sequentially.

	F-measure	Precision	Recall
逐次学習あり	0.769	0.849	0.704
逐次学習なし	0.505	0.886	0.353

化しても F-measure，Precision，Recall がそれぞれ 0.668，0.913，0.527 であった．進行度 7 のトラフィックデータを学習に用いた場合，学習時に異常が多く存在するため検出精度が低くなっていた．一方，進行度 2 のトラフィックデータで逐次学習し，進行度 8 のトラフィックデータでテストした場合，検出精度が向上していた．進行度 2 のトラフィックデータを逐次学習することで正常の種類数が増加し，ロバストな学習結果を得られたと考えられる．なお，進行度 2 のトラフィックデータで学習し，進行度 8 のトラフィックデータでテストした場合，逐次学習する方が F-measure，Precision，Recall の値が若干低下した．これは，逐次学習によって本来異常であるデータを正常と誤認識したためであった．この問題は，逐次学習の際の閾値の調整によって解決できると考えられる．

学習データとテストデータの進行度の組み合わせごとに閾値を最適化し，F-measure，Precision，Recall，学習および異常検出に要した時間を算出した．本手法における結果と文献 [4] における結果をそれぞれ表 4，表 5 に示す．進行度 2 のトラフィックデータで学習し，進行度 7 および進行度 8 のテストデータで実験を行った結果，どちらの手法においても全ての値が 0.8 を上回る結果となった．また，進行度 7 のトラフィックデータで学習を行い，進行度 8 のトラフィックデータでテストを行った場合と進行度 8 のトラフィックデータで学習を行い，進行度 7 のトラフィックデータでテストを行った場合ではどちらの手法においても進行度 2 を学習した場合と比較して全ての値が低い結果となった．

進行度の組み合わせごとに文献 [4] の結果と比較しながら考察する．まず，進行度 2 のトラフィックデータで学習し，進行度 7 および進行度 8 のトラフィックデータでテストを行った場合について考察する．閾値は 2.2 とした．いずれの進行度の組み合わせにおいても F-measure は，どちらの手法においても 0.9 程度の結果となった．いずれの進行度の組み合わせにおいても Precision の値については本手法の方が高い結果となった．学習データが全て正常な通信であったために正常な通信の特徴をうまく学習することができ，テストデータに現れた正常な通信を正しく識別することができた．一方，Recall の値については文献 [4] の手法の方が高い結果となった．進行度 2 と進行度 8 の組み合わせで実験を行った際に異常な通信を誤って正常な通信とみなした区間は文献 [4] の手法では存在せず，本手法では合計で 79 区間存在した．これらの区間の多くの勝者

表 4　検出結果
Table 4　Result of Detection.

学習データ	テストデータ	F-measure	Precision	Recall	Time(s)
進行度 2	進行度 7	0.968	0.999	0.940	23.89
進行度 2	進行度 8	0.907	0.941	0.876	24.38
進行度 7	進行度 8	0.668	0.913	0.527	25.30
進行度 8	進行度 7	0.846	0.766	0.944	23.95

表 5　[4] を用いた検出結果
Table 5　Result of Detection using Existing method [4].

学習データ	テストデータ	F-measure	Precision	Recall	Time(s)
進行度 2	進行度 7	0.985	0.975	0.995	69.51
進行度 2	進行度 8	0.899	0.817	1.0	84.20
進行度 7	進行度 8	0.237	0.158	0.469	83.31
進行度 8	進行度 7	0.692	0.539	0.967	64.28

ノードは学習回数が少ないノードであった．そのため，正常と識別する閾値を調整することなどで改善できると考えられる．

次に，進行度 7 のトラフィックデータで学習し，進行度 8 のトラフィックデータでテストを行った場合と進行度 8 のトラフィックデータで学習し，進行度 7 のトラフィックデータでテストを行った場合について考察する．進行度 7 のトラフィックデータで学習し，進行度 8 のトラフィックデータでテストを行った場合の閾値は 5.25，進行度 8 のトラフィックデータで学習し，進行度 7 のトラフィックデータでテストを行った場合の閾値は 61 とした．いずれの組み合わせにおいても，本手法の方が文献 [4] の手法と比べて全ての値が同程度もしくは高くなった．本手法において進行度 7 のトラフィックデータで学習し，進行度 8 のトラフィックデータでテストを行った場合，Precision の値は高いが，F-measure と Recall の値はやや低めとなった．学習に用いた進行度 7 のデータには異常が多く含まれているため，正常の特徴だけでなく異常の特徴も学習してしまい，テストデータの進行度 7 と同様の異常をうまく検出することができなかった．本手法において進行度 8 のトラフィックデータで学習し，進行度 7 のトラフィックデータでテストを行った場合，比較的高い結果となった．進行度 8 のトラフィックデータは異常区間が少なく，正常区間が多いため正常の特徴をうまくとらえて学習し，異常を検出することができた．以上の結果から，学習時のトラフィックデータに異常が含まれていないもしくは少数含まれている場合，正常の特徴をうまく学習でき高精度で異常を検出できた．今回は進行度の組み合わせごとに最適な閾値を与えたが，閾値を自動で決定する方法を今後検討したいと考えている．

学習と異常検出に要した時間について考察する．いずれの組み合わせにおいても文献 [4] の手法では最短でも 60 秒以上要しており，本手法の方が短い時間で学習と異常検出ができた．文献 [4] の手法ではマハラノビス距離を基に異常検出しているため，入力された特徴ベクトルの全てを分散の計算に用いていた．そのため，特徴ベクトルの数に比例して異常検出に要する時間も増加したと考えられる．一方で本手法では逐次学習を行う際，参照ベクトルの更新を行うためだけに入力された特徴ベクトルを用い，更新後は必要としないため異常検出に時間を要さなかった．

4.2　大阪府立大学のトラフィックデータにおける実験
4.2.1　実験条件
本手法の実運用中のネットワークでの有効性を確認する実験を行った．大阪府立大学のネットワークとインターネットの間にあるファイアウォールの外側で収集したトラフィックデータを用いて実験を行った．学習データに 2019 年 7 月 8 日 7 時 25 分 1 秒から 10 分間収集したトラフィックデータ，テストデータには 2019 年 7 月 8 日 7 時 35 分 1 秒から 10 分間収集したトラフィックデータを用いた．テストデータでは，7 時 41 分 46 秒に UTM(Unified Threat Management) で TCP flood 攻撃が観測されている．学習時のパラメータや特徴抽出時の 1 区間あたりのパケット数は MWS データセットにおける実験と同条件で行った．テストデータは 100 パケットで分割した際，約 111500 区間存在した．UTM において DoS 攻撃を観測した区間はテストデータの約 78972 区間目である．実運用中のトラフィックデータを学習およびテストに用いた場合，正解ラベルを付与することができず，F-measure などの評価指標を用いることができない．そこで特定のノードに着目し，100 区間ごとに勝者ノードに選択された回数を数え，その遷移に注目することでどのような特徴を持つかを確認した．
4.2.2　実験結果と考察
DoS 攻撃が行われた 7 時 41 分 46 秒より少し前の区間を含む 7 時 41 分 8 秒からの約 10 秒間にあたる 400〜500 番目付近から集中的に (0,0) のノードが勝者ノードに選択されていた．そのため，ここでは (0,0) のノードが勝者ノードに選択された回数の遷移を図 8 に示す．勝者ノードに選択された数が 200 番目と 400〜800 番目付近で多くなっていることがわかった．200 番目におけるパケットを確認したところ，再送処理のためのパケットが多く含まれていた．また 400〜800 番目付近の内，特に 400〜500 番目付近で選択された勝者ノードは (0,0) のノードのみであった．400〜500 番目にあたるパケットのフラグを確認したところ，SYN パケットが連続して見られておりこれは DoS 攻撃の影響であると考えられる．UTM における検出は 789 番目であるが，一定の閾値を超えた場合に行われる．そのため，実際に DoS 攻撃が行われたのは 78972 区間目よりも前の時刻であると考えられる．本手法で，UTM では検知できなかった異常に対する変化を検出できたことを確認

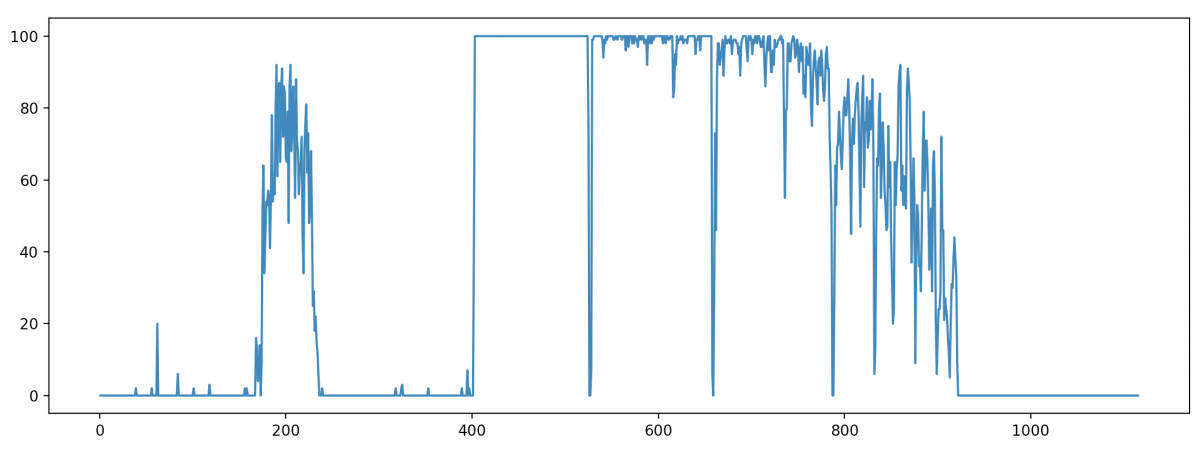

図 8　(0,0) のノードが勝者ノードに選択された回数の遷移
Fig. 8　Number of transitions when the node (0,0) is selected as the winner node.

できた．また，(0,0) のノードは学習時にも集中的に存在する区間があり，学習回数が多かったため，異常として検出することはできなかった．(0,0) のノードは連続して選択されていたため，ノードの遷移に注目することで異常検出が可能になるのではないかと考えられる．また，実運用中のトラフィックでは異常を全く含まないデータを準備するのは困難である．そのため，異常を含む場合における閾値の設定方法の検討などが今後の課題として挙げられる．

5. まとめ

本稿では，パケットのヘッダから抽出した特徴量ごとにエントロピーを算出し，生成した特徴ベクトルに対して SOM で学習を行い，異常検出に適用することで持続的に攻撃を検出することができる手法を提案した．実験では，MWS データセットおよび大阪府立大学のトラフィックデータの 2 種類のデータを用いて本手法の有効性を確認した．MWS データセットを用いた実験では，学習に用いるデータに異常が含まれていない場合と異常が少ない場合に異常検出が可能であることを確認できた．また，大阪府立大学のトラフィックデータを用いた実験では，UTM が観測した異常で SOM の勝者ノードが特定のノードに集中することを確認した．

今後の課題としては，SOM のパラメータの調整や実運用中のトラフィックを用いた実験の精査，閾値の設定方法の検討などが挙げられる．

参考文献

[1] 平松尚利, 和泉勇治, 角田裕 : 複数の通常状態を用いたネットワーク異常検出, 電子情報通信学会技術報告, CS2006-32, pp.61-66(2006).

[2] 佐藤陽平, 和泉勇治, 根元義章：複数の検出モジュールの組み合わせによるネットワーク異常検出の高精度化, 電子情報通信学会技術報告, NS2004-144, pp.45-48(2004).

[3] Mitsuhiro Hatada, and Tatsuya Mori:Finding New Varieties of Malware with the Classification of Network Behavior, IEICE TRANSACTIONS on Information and Systems, vol.E100.D, No.8, pp.1691-1702(2017).

[4] 小島俊輔, 中嶋卓雄, 末吉敏則：エントロピーベースのマハラノビス距離による高速な異常検知手法，情報処理学会論文誌, Vol.52, No.2, pp.656-668(2011).

[5] 柿本圭介, 田中英彦 : 自己組織化マップを用いた Windows システムサービスコールの分類によるマルウェア検出手法, 情報処理学会研究報告, vol.2008, No.45(2008-CSEC-041), pp.43-48(2008).

[6] 小久保博崇, 金岡晃, 満保雅浩, 岡本栄司：攻撃通信検知のための合成型機械学習手法の一検討, 情報処理学会論文誌, Vol.53, No.9, pp.2086-2093(2012).

[7] Kohonen, T.,The self-organizing map, Proceedings of the IEEE, Vol.78, No.9, pp.1464-1480(1990).

[8] 高田雄太, 寺田真敏, 松木隆宏, 笠間貴弘, 荒木粧子, 畑田充弘 : マルウェア対策のための研究用データセット MWS 2018 Datasets , 情報処理学会研究報告, Vol.2018-CSEC-82, No.38, pp.1-8(2018).