LSTM-RNN を用いたフロー予測による インターネット・バックボーンを対象とした異常検知手法

和久井 拓[1]　近藤 賢郎[1]　寺岡 文男[2]

概要：インターネット・バックボーンには多種多様なトラフィックが混在する．その結果観測されるトラフィックは平常時でも時間軸に対して変動が大きいジャギーな特性を有する．一方で，インターネット・バックボーンでは，機器故障や攻撃活動などに起因する多種多様な異常事象が生じうる．安定したインターネット運用の実現のためには，ジャギーな特性を持つトラフィックを前提として多種多様な異常事象を検知する汎用的な技術が必要となる．本稿では，インターネット・トラフィックを対象として多様な異常事象に対する汎用的な異常検知手法を提案する．トラフィック情報にはトラフィック流量に対してスケーラブルに観測可能なフロー情報を用いる．また多種多様な異常事象を観測可能なメトリックとしてトラフィックのスループットを用いる．異常事象の発見のために参照されるモデルは，インターネット・トラフィックの周期性に着目して LSTM-RNN (Long Short-Term Memory RNN) によって作成する．ジャギーな特性を持つトラフィックは終点アドレス，始点アドレスの双方の視点で分類される．終点アドレスは BGP (Border Gateway Protocol) で構成される経路表に含まれる AS PATH リストを集約したクラスに分類される．結果的に，生成されるモデルの数は終点アドレス数に対してスケールする．本稿では，BGP で構成される経路表やフロー情報のパーサ，LSTM-RNN に基づくトラフィック・モデルの学習器を実装した．LSTM-RNN に基づく学習器は cuDNN, Chainer といったライブラリをもとに GPU 環境で動作するものを実装した．学習に利用するデータセットには WIDE バックボーン (AS2500) で観測されるフロー情報と BGP の経路表を用いた．生成されたトラフィック・モデルとデータセットに含まれる実測値の比較の結果，曜日単位でのモデル学習を通じ妥当なトラフィック・モデルを生成した．また，Event Traffic の検知を確認した．

Anomaly Detection Method for Internet Backbone Traffic by Flow Prediction with LSTM-RNN

Taku Wakui[1]　Takao Kondo[1]　Fumio Teraoka[2]

1. はじめに

近年, NN (Neural Network) ベースのネットワークの異常状態の検知に関する研究が増加している．特に, DDos (Distributed Denial of service) 攻撃の検知や Botnet (Bot Network) の検知など, 特定の攻撃手法や脆弱性に着目した研究が多く存在する．インターネット・バックボーンのように複数のインターネットで構成される規模の大きいネットワークは様々なユーザやサービスのトラフィックが含まれる．そのためインターネット・バックボーンでは多種多少な異常事象が発生しうる．ネットワークの状態異常は, 故障によるものと, 故障を伴わないものが存在する．前者はネットワークを構成する機器等が正常に動作しないことが原因の異常である．後者は攻撃や不正な操作により異常な挙動を起こす Abuse や, アクセスの集中や大規模な通信障害が影響する Event traffic などが該当する．しかし安定したインターネット運用の実現のためには, 機器故障や Event Traffic を含めた様々な異常事象を検知する汎用的な技術が必要である．

図 1, 2 は, 本稿で解析対象とした NetFlow のにおける

[1] 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University
[2] 慶應義塾大学理工学部
Faculty of Science and Enginnering, Keio University

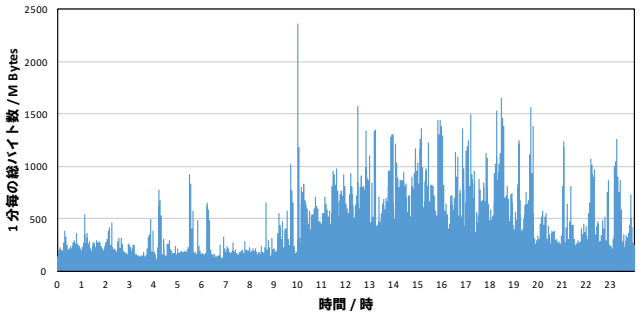


図 1　トラフィックの 1 日単位の推移.

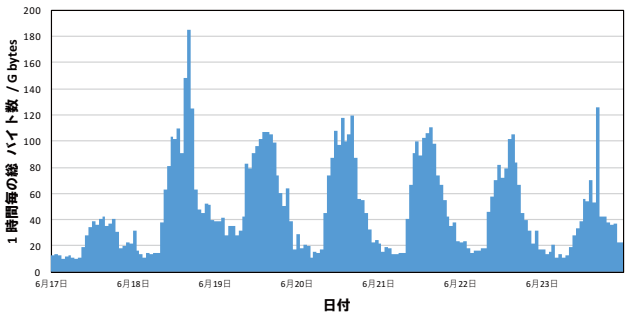


図 2　トラフィックの週単位の推移.

単位時間当たりの総バイト数の変化を表したグラフである. 図 1 は 6 月 17 日のデータを, 図 2 は 6 月 17 日から 23 日までのデータをそれぞれ表している. インターネットにおける時系列情報は 図 1 のようにジャギーで変動が不規則である. これは規模が大きいネットワークほど不規則な変化をする傾向にある. そこでクラス分けや特定の情報のみを抽出 (フィルタリング) するなど, データに前処理を施すのが一般的である. また, 1 日の一部分のみを見るとインターネット・トラフィックの動きは非常に不規則に見えるが, 図 2 のように 1 日を周期とした変動をとる側面も持ち合わせている. これは 1 日の中でネットワークの利用が集中する時間帯とそうでない時間帯が存在することが原因である. ネットワークの情報を学習する際に, NN の構成や, 学習させる時系列データの単位などの決定にこれらの特性を考慮する必要がある.

これらを踏まえ本稿では, 学習と予測に用いるジャギーなフロー情報に対し, RIB (Routing Information Base) が持つ AS パスによるクラス分けを施した. また, 異常事象の発見のために参照されるモデルは LSTM-RNN (Long Short-Term Memory RNN) によって作成した. RNN (Recurrent Neural Network) の拡張である LSTM-RNN は, より長期的な周期や特徴を持つ時系列データの学習に適している. LSTM を用いることで, フローデータの 1 日や 1 週間といった単位での周期や特徴をより正確に学習することを目的としている. ネットワークの構成と, データの周期性を意識したモデル構築を通じフロー情報を正確に予測することで, ネットワークの異常状態を検知することを目的としている.

2. 関連研究

ネットワークの異常検知に関する研究は近年増えている. 特に不正な通信や侵入, 攻撃など (Abuse) などの兆候を検知する IDS (Intrusion Detection System) はセキュリティの観点から特に増えている研究テーマである. 本節では IDS を含む, 様々な異常検知に関する研究について, その手法や特徴を紹介した上で, 本稿における要求事項を提示する.

2.1 Signature-based

異常検知の手法は大きく分けて, Signature-based と, Anomaly-based の 2 つに区分される. 前者は既知の異常データ (不正な通信のログなど) をもとに, 不正や攻撃を目的とした通信を IP などの特徴を用いて定義し, その特徴に合致する通信をブロックする手法である. この手法は処理時間が Anomaly-based に比べ速く, リアルタイムの検知に適している. 一方, 全く新しい特徴を持つ不正な通信や攻撃を防ぐことが難しいという欠点を持つ [1] [2]. 本稿では Machine Learning ベースの手法で新しい特徴を持つ不正な通信や攻撃にも対応する異常検知手法を提案する.

2.2 Anomaly-based

Anoaly-based は新しい特徴を持つ不正な通信や攻撃を防ぐことを目的とした手法である. 統計学的手法 [3] や Machine Learning をベースとした手法で近年では, SVM (Support Vector Machine), KNN (K-Nearest Neighborhood), SOM (Self Organizing Map)[4] など, Machine Learing をベースとした手法が増えている.

また, 特に IDS の研究は, 検知の対象とする不正な通信や攻撃の手法を限定したものが多い. Botnet による攻撃や不正な操作の検知を目的とした研究 [5] [6] や, SDN (Software Defined Network) における不正通信の検知を目的とした研究 [4] がその例である. 特に Botnet は, Bot を通じて Botnet を操作することも可能であり, 不正リモートユーザである Botmaster の特徴を定義することが難しい. そのため Signature-based の手法よりも, NN を用いた手法が適している.

Machine Learning をベースとした多くの研究は, Abuse があったと判明している情報と, Abuse がなかったことが判明している情報のデータセットを用いる. これは教師データとして通常と異常の情報をラベル付けし, それを正しく分類できるように学習させる 2 値分類問題である. これらの研究は性能検証にもラベリングされたデータを用い, 混同行列で評価するのが一般的である [4] [5] [6]. しかし機器故障や Event Traffic についてはラベリングされたデータは収集が困難である. そこで本稿では, 通常時のフロー

表 1　要求事項と関連研究の対応表.

手法	Signature-based		Machine-Learning				
	[1]	[2]	[3]	[4]	[5]	[6]	本稿
スケーラビリティ	×	○	○	×	×	×	○
多種多様な異常状態に対する汎用性	×	×	×	×	×	×	○
トラフィックの周期への着目	×	×	×	○	×	×	○
ラベリングされたデータの要否	必要	必要	不要	必要	必要	必要	不要

情報を学習させ, その後のフロー情報を予測し, 実際のフロー情報との挙動の違いから異常検知する. この手法はラベリングされたデータを要さない.

2.3　本稿の要求事項

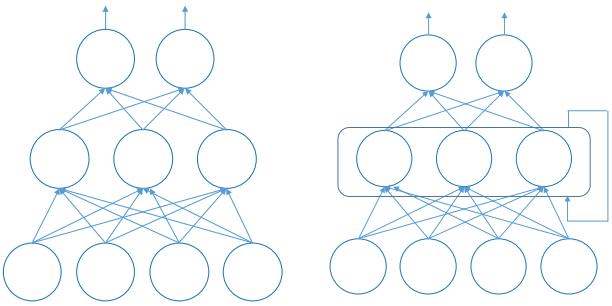
本稿では, 通常時のフロー情報を学習させ, 先のフロー情報を予測し, 実際のフロー情報との挙動の違いから異常検知する. そのためラベリングされたデータセットは要さない. また検知対象とする異常状態を限定しない汎用的手法である. 異常検知に参照されるモデルには, トラフィックの長期的な周期性に着目し, LATM-RNN を使用する. また BGP (Border Gateway Protocol) で構成される経路表に含まれる AS PATH リストを集約したクラスに分類する. 生成されるモデルの数は宛先アドレス数に対して集約される. 以上の要求事項と本章で紹介した関連研究の対応関係を表 1 に示す.

3.　関連技術

3.1　RNN (Recurrent Neural Network)

RNN は内部に有向閉路を持つニューラルネットの総称である. この構造により RNN は, 過去に入力した情報を一時的に記憶させることが可能になった. これにより, 単純な NN (Neural Network) では扱えなかった, 時系列データに代表される系列データ (個々の要素が順序を持ち, その順序に意味を持つデータ) を取り扱うことができる.

図 3 に単純な NN (順伝播型ネットワーク) と, RNN の概要図を示す. NN は複数の入力を受け, 1 つの出力を計算するユニットで構成され, 層状に並べたユニットが隣接層間で結合した構造を持つ. 単純な NN では, 入力 x に対する望ましい出力を d とした時, この x と d のペアの集合, $(x_1, d_1), (x_2, d_2), ...(x_n, d_n)$ を学習のデータに用いる. ただし n は有限な自然数とする. このとき x を学習データ, d を教師データと呼ぶ. x_1 を NN に入力すると, NN 内で定義されたパラメータをもとにした線形計算を経て出力 y_1 が得られる. y_1 が, 理想的な出力である d_1 に近くなるように, NN 内のパラメータを調整し, それを繰り返すことで, 出力されるデータ y が 理想的な出力に近い値になるよう試みることを学習という. また学習によりパラメータが調整され, 出力結果が明らかでない入力に対しても適用できるようになった状態をモデルと呼ぶ. 一方, RNN で扱う入力データは系列データである. RNN への入力に対し, NN 同様出力が得られるが, ここで得られる出力は, RNN 内部にある帰還路によって, 出力を計算する際, RNN が過去に受け取ったすべての入力 (入力の履歴) が関与した出力である. $x_1(t), x_1(t+1)$ をある系列データにおける連続する要素であるとする. RNN 帰還路はネットワーク内部において, 出力を自分自身に戻す役割を持つので, 入力 $x_1(t)$ に対する出力 $y_1(t)$ は帰還路を通り, 次の入力 $x_1(t+1)$ と共に RNN に入力され, 次の出力 $y_1(t+1)$ を得る. 学習では $y_1(t+1)$ が, 理想的な出力に近付くようにパラメータの調整を入出力の度に繰り返し, 系列データのモデルを作る.



[1] 順伝播型ネットワーク.　　[2] RNN.

図 3　NN の構造概略図.

なお, ネットワークの出力を理想的な出力に近づけるには, 2 つの値の近さを測り, 微積計算によりパラメータを調整する必要がある. この 2 つの値の近さを測る尺度を誤差関数と呼ぶ. この誤差関数は, 解決する問題によって様々である.

3.2　LSTM (Long Short-Term Memory)

RNN は系列データの文脈を捉えて推定するが, このとき現時刻からどれだけ遠い過去の入力を出力に反映させられるかが重要である. 理論上, 過去の全入力履歴が考慮されるはずであるが, 実際に RNN で出力に反映できるのは高々過去の 10 件程度である [7]. この限界は, NN の勾配消失問題と同じ原因で生じている. 勾配消失問題は, 層数の多いネットワークにおいて, パラメータ更新に必要な勾配というものを計算する際に生じる. 層を遡るにつれて勾配の値が爆発的に大きくなるか, あるいは 0 に消滅する性質である. RNN は帰還路を含む構造により, 元々の層数は少なくても, 勾配計算時には深い層を扱う NN と同様, 勾配が爆発するか消滅するかの 2 択になり, RNN では長い系列を扱えない. そのため短期的な記憶は実現できても, より長期にわたる記憶を実現することが難しい.

この問題に対し, 長期にわたる記憶を実現する方法の 1 つとして LSTM が提案されている. LSTM は, 中間層の各ユニットを, 図 4 に示すメモリユニットと呼ぶ要素で置き換えた構造を持つ. メモリユニットは内部にメモリセル

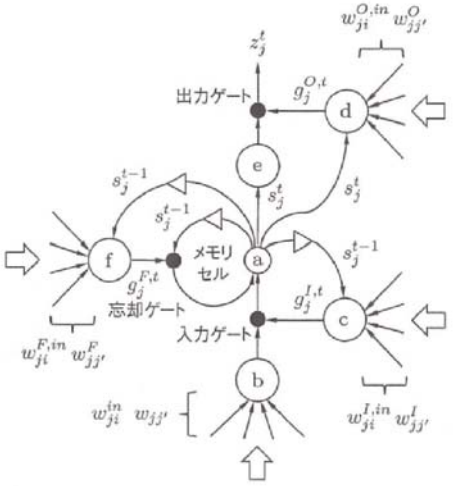


図 4　LSTM のメモリユニット (文献 [7] から引用).

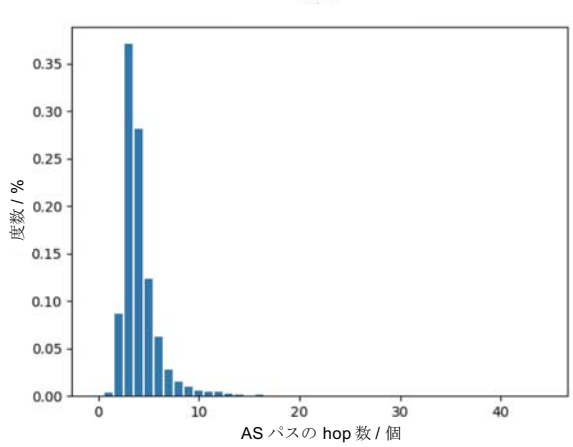


図 5　統計日数と AS パターンの数の関係.

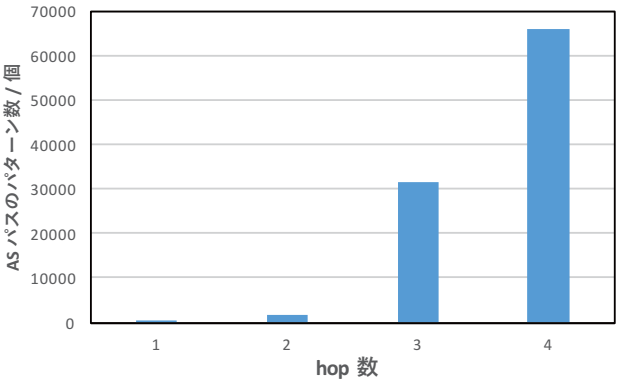


図 6　hop 数と AS パスのパターン数の関係.

と呼ばれる帰還路と複数のユニット, またその内部ユニット間の入出力値を調整するゲートで構成されている. この手法を用い, より長期にわたる学習を可能にした RNN を LSTM-RNN , もしくは単純に LSTM と呼ぶ. LSTM-RNN をはじめとする長期にわたる学習が可能な RNN は, 長期にわたる周期性や特徴を持つ時系列データの学習に適している.

4. 提案

4.1 AS パスによる Aggregation

本稿では過去のフロー情報を学習させ, 以降のフロー情報を予測する. しかし, 一般にネットワーク単位のフロー情報は変動が不規則で, そのままの状態では学習, 予測が困難である. そのため多くの研究では, フロー情報に含まれる IP アドレスなどの情報をもとに, 学習の対象とするデータを制限したり, データを切り分け学習させたりすることで, より正確な学習, 及び予測を試みている. 本稿では, 経路サーバにて計算された BGP の経路表である RIB を用い BGP の構成を意識してクラスを定義した. RIB は BGP の経路情報であり, フルルートの Destination の IP アドレスと, 観測地点からその IP までの AS パスが記録されている. BGP の経路は, 始点と終点によって, 経由する AS も, 経由する AS の数も異なる. 図 5 は, 6 月 17 日の RIB の AS パスを解析し, 経由した AS の数をヒストグラムにした結果である. 最小値は 0 (経由 AS が無い) , 最大値は 44 であった. 経由する全 AS の組み合わせは膨大な数であるため, 観測地に近い方から, 経由する AS の経路が途中まで同じものを同じクラスに分類させた. 図 5 に示した解析の結果, 最頻値は 3, 中間値は 4 であった.

次に, ある日の RIB について, 観測点に近い方から 4 hop 目までを切り出したときの AS パスのパターン数 (重複なしで数えた際の数) を調べた. 結果, 図 6 に示す通り, 3 hop では 31,258 個, 4 hop では 66,029 個の AS パスのパターンが存在した. さらに, 6 月のある 1 週間の RIB について解析し, AS パスのパターン数と日数の関係を調べた結果が図 7 である. AS パスのパターン数は累積数である. 4 hop では 1 週間で約 6.7 % 増加しているのに対し, 3 hop の場合では約 2.9 % の増加にとどまっている. IP アドレスを AS パスのパターンで Aggregate しクラスを定義する際, クラス作成に使用するデータの日数の増加に対するクラス数の増加が小さい方が, クラス数が安定する. 本稿では, 観測地から 3 hop まで経路が同じである IP を同じクラスに分類した. 例えば, AS パスが 4713 2914 38040 23969 であるものと, 4713 2914 4651 23969 であるものは異なるクラスに属するが, 2947 7670 18144 と, 2947 7670 18144 4826 は同じクラスに属する. これにより, IPv4 のフルルートで 2^{32} (4,294,967,296) 件存在する Destination の IP アドレスが, 3 万余りのクラスに Aggregate された.

本稿において, 各日, およびある期間の RIB に記録されている AS パスについて, 観測点側から 3 hop までを切り出し, 重複無く記録した. これをクラスリスト呼び, クラス別の時系列データの作成とクラス数の管理に使用した.

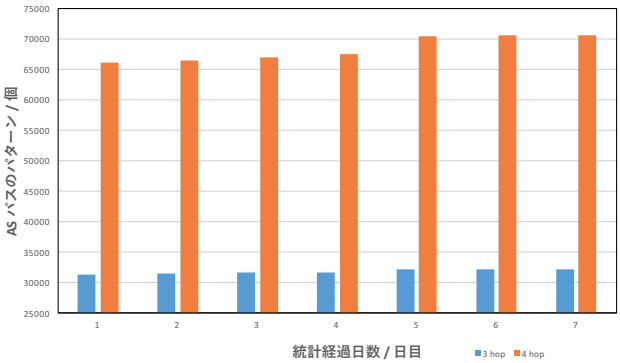


図 7　AS hop 数のヒストグラム.

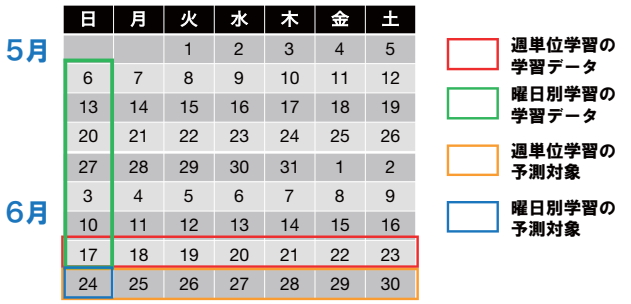


図 8　各学習方法の学習データと予測対象.

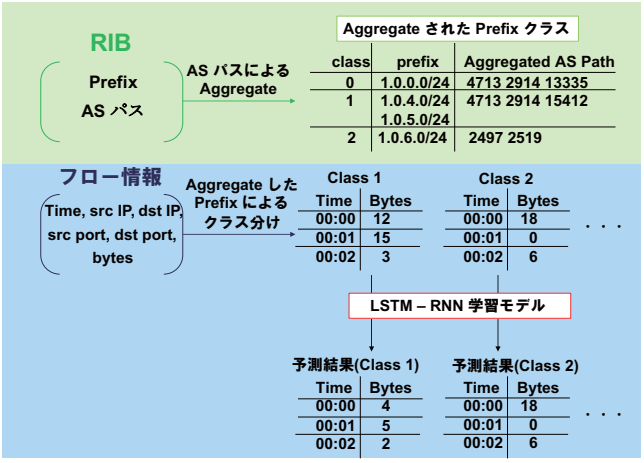


図 9　提案の全体像.

4.2　学習方法

本稿で用いた情報を観測している WIDE BB ネットワークは学術ネットワークであり, NOC (Network Operation Center) には大学も含まれている. そのため, 1 週間を 1 単位としたフローの特徴, 曜日毎の特徴があると考え, 週単位の学習, 曜日別の学習, という 2 つのアプローチをとった. NN (Neural Network) は学習させるデータにより予測する値が異なるため, 2 つのアプローチの結果を比較し, 本稿で使用した BB ネットワークのフロー予測に適した学習のアプローチを評価する.

4.2.1　週単位の学習

週単位の学習では, 予測対象となるフロー情報の日付の前の週のフロー情報を 1 週間分学習させてモデルを作成する. モデル作成の際に使用するクラスリストは, 前の週の 1 週間分の RIB に記録されている AS パスを用い作成した. 本稿では図 8 に示すように, 2018 年 6 月 17 日から 6 月 23 日までの 1 週間の情報でモデルを作り, 6 月 24 日 から 6 月 30 日までのフロー情報を予測し実データと比較した. 週単位の学習では, 直近のまとまったフロー情報をもとに作成したモデルを用いた予測の性能を評価する.

4.2.2　曜日別の学習

曜日別の学習では, 日曜日から土曜日までの 7 つの曜日別で情報を分けて学習させる. 週単位の学習と比較するため, 学習させる情報は 7 日分とする. 学習に使用するフロー情報は, 同じ曜日のフロー情報を日付の古い情報から順に連続させたものを用い, クラスリストは該当曜日の中で最も新しい日付の RIB ファイルに記録されている AS パスを用い作成した. 本稿では, 5 月の第 2 週から, 6 月の第 4 週の情報でモデルを作り, 6 月の第 5 週の各曜日のフロー情報を予測した. 図 8 に日曜日のモデルの例を示す. モデル作成には 5 月 6, 13, 20, 27 日, 6 月 3, 10, 17 日の情報を用い, 6 月 24 日のフロー情報を予測し実測データと比較した. 曜日別の学習では, 学習データに直近ではない情報を含めるが, 各曜日の特徴を捉えることを目的としたモデルを用い予測の性能を評価する.

4.3　提案の全体像

以上を踏まえ, 図 9 に提案の全体像を示す. まず RIB のもつ情報からクラスリストを作成する. さらにフロー情報の dst IP と RIB の Prefix をマッピングさせ, クラスリストをもとにクラス別時系列データを作成する. このデータを LSTM-RNN で学習させ予測する.

5.　実装

5.1　実装環境

表 2 に示すように, 実装環境として Ubuntu Server 18.04.1 がインストールされたサーバを用い, LSTM の実装とデータ処理を含む全ての実装は Python 3.7.0 を用いた. また, 学習, 予測を含む LSTM の実装には Chainer 5.1.0 を用い, フローデータの変換に nfdump version 1.6.17 と bgpdump version 1.4.99.13 を用いた. さらに, LSTM の学習, 予測の計算の GPU (Graphics Processiog Unit) による高速化を図った. GPU プラットフォームには CUDA 9.0 を使用した.

5.2　データの前処理

5.2.1　NetFlow の処理

フロー情報である NetFlow はバイナリファイルとして記録されている. そこで, nfdump を用いてテキストファイルに変換した. また, NetFlow は 1 時間分の情報を 1 つの

表 2　実装環境.

項目	内容
OS	Ubuntu Server 18.04.1 LTS
実装言語	Python 3.7.0
アプリケーション	nfdump version 1.6.17
アプリケーション	bgpdump version 1.4.99.13
ライブラリ	Chainer 5.1.0
GPU プラットフォーム	CUDA 9.0

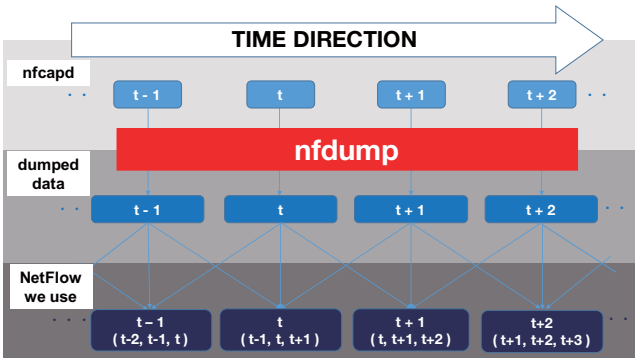


図 10　NetFlow の処理.

ファイルにまとめているが, 前後の時間の情報, 例えば, ある日付の 0 時の記録において, 本来 1 つ手前の時間のファイルに含まれているべき 23 時のフロー情報や, 本来 1 つ後の時間のファイルに含まれているべき 1 時のフロー情報が記録されていた. そこで, より正確な入力データを生成するため, 図 10 のように, 前後 2 つのファイルと合わせたファイルを生成し, CSV 形式に変換し記録したファイルを当該時間のフロー情報として扱った.

5.2.2　RIB の処理

RIB の情報は MRT フォーマットで記録された BGP 経路情報である. bgpdump を用いてテキストファイルに変換した. また, RIB から, AS パスの情報を抜き出して記録した AS パスファイルと, PREFIX の情報を抜き出して記録した Prefix ファイルを生成した. 図 11 に変換された RIB ファイルの一部を, 図 12 に RIB から生成した AS パスファイルの一部を, 図 13 に Prefix ファイルの一部を示す. 日付を共有する AS パスファイルと Prefix ファイルは, 行数をインデックスとして共有させている. したがって Prefix ファイルのある行に記録されている PREFIX への AS パスは, 同じ日付がついた AS パスファイルの同じ行数に記録されている.

5.3　学習, 予測までの実装

5.3.1　実装の全体像

フローデータの前処理以降の処理を図 14 に示す. 図中の赤い箇所が実装したプログラムである. 本節では, クラスリストの定義と生成, クラス別データの生成, LSTM による学習, 予測の実装についてそれぞれ説明する.

```
TIME: 05/19/18 15:00:00
TYPE: TABLE_DUMP_V2/IPV4_UNICAST
PREFIX: 1.0.0.0/24
SEQUENCE: 0
FROM: 203.178.136.13 AS2500
ORIGINATED: 05/09/18 20:36:01
ORIGIN: IGP
ASPATH: 4713 2914 13335 13335
NEXT_HOP: 203.178.136.14
LOCAL_PREF: 100
AGGREGATOR: AS13335 103.22.201.1
COMMUNITY: 2500:5971
```

図 11　RIB の一例 (冒頭のみ).

```
4713 2914 13335 13335
4713 2914 13335 13335
4713 2914 15412 4826 38803 56203
4713 2914 15412 4826 38803 56203
4713 2914 15412 4826 38803 56203
4713 2914 15412 4826 38803 56203
4713 2914 15412 4826 38803 56203
4713 2914 15412 4826 38803 56203
4713 2914 15412 4826 38803 56203
4713 2914 15412 4826 38803 56203
```

図 12　AS パスのみ抽出した結果の例 (文頭から 10 行を抜き出し).

```
1.0.0.0/24
1.0.0.0/24
1.0.4.0/22
1.0.4.0/22
1.0.4.0/24
1.0.4.0/24
1.0.5.0/24
1.0.5.0/24
1.0.6.0/24
1.0.6.0/24
```

図 13　Prefix のみ抽出した結果の例 (文頭から 10 行を抜き出し).

5.3.2　AS パスによるクラス定義

クラス定義とクラスの抽出には, 各日の RIB から AS パスの情報のみを抽出しテキストファイルに出力させたものを用いる. クラスリストの作成には学習データ中の適当な日, もしくは期間の RIB から抽出した AS パスの情報を用いる. まず, AS パスの 3 hop 目までを 1 行目から最後までを重複無く記録する. クラスリスト作成に使用する RIB が複数日の場合, これを繰り返す. 期間中の AS パスを重複なく全て記録したものがクラスリストである. クラスリ

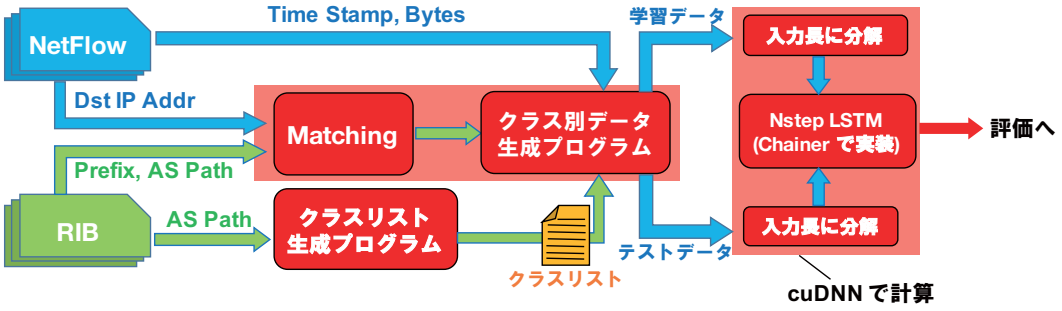


図 14　実装の全体像.

```
1:  ['2914', '2828', '26764']
2:  ['2914', '174', '27589']
3:  ['2914', '8708', '48332']
4:  ['2914', '174', '21670']
5:  ['2914', '174', '201553']
6:  ['2914', '6453', '23474']
7:  ['2914', '3303', '13283']
8:  ['4713', '131924']
9:  ['2914', '3320', '50458']
10: ['2914', '61832', '28257']
```

図 15　クラスリストの例 (文頭から 10 行を抜き出し).

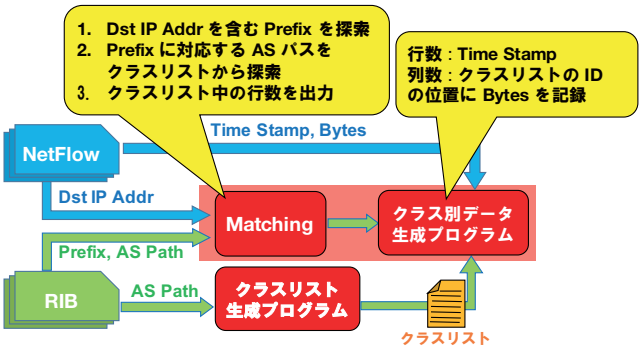


図 16　入力データ生成の概要.

ストにおいて, AS パスがクラスの定義であり, 行数がクラス番号 (インデックス) の機能を持つ. 図 15 に 6 月 3 日から, 6 月 9 日までの 1 週間分 RIB で作ったクラスリストの一部を示す.

5.3.3　クラス別時系列データの作成

本項では, 5.3.2 項で定義したクラス毎にデータを分け, RNN へ入力する時系列データに変換する操作について説明する.

まず, 本項で作るクラス別時系列データの形を説明する. クラス別時系列データは配列構造で, 行は時間を, 列はクラスを管理し, 配列の値はフローの大きさを記録している. 本稿で扱った NetFlow ではバイト数 (Bytes) を記録している. 行の値が大きくなるにつれ, 後の時間の情報を記録している. そのため, ある列において 0 行から順に読むことで, そのクラスの時系列データを得ることができる. 時間軸を n 分で区切っている場合, 配列には n 分間の バイト数の合計が記録される.

このような形のテキストデータを生成するプログラムが, クラス別データ生成プログラムである. 図 16 はクラス別データ生成プログラムの概要図である. このプログラムでは, NetFlow と 同じ日付の RIB の情報に加え, 前項で説明したクラスリストを用いる.

まず フロー情報が記録されたファイルの各行から Destination IP adress (Dst IP addr) を取り出し, RIB ファイルからその Dst IP Addr に該当する (含む) Prefix を探す. 該当する Prefix が見つかった場合, その Prefix に対応する AS パスを出力する. 該当 Prefix が見つからない場合はそのフロー情報を破棄する. なお, 破棄した情報の多くは Private Address であった. RIB に含まれる Prefix の情報はフルルートのため件数が膨大であるので, Prefix の値を主値とした平衡二分木 (Red-Black-Tree[8]) を作り, NetFlow ファイルの dst IP を検索することで計算時間を短縮させた. 次に, 出力した AS パスの情報をクラスリストから探し, 見つけた行数をクラスリストのインデックスとして出力する. このクラスリストのインデックスは, 配列データの列数を示している. 続いて NetFlow に記録された時間 (time stamp) を用いて出力する行数を決定する. 時間軸を n 分で区切っているとし, データの記録時間が x 時 y 分である場合, 出力行数 $= 60x/n + \lfloor y/n \rfloor$ と計算できる. 最後に, 決定した出力行数, 列数に, NetFlow に記録されている bytes の値から計算した値を加算する.

5.4　LSTM - RNN の実装

本節では, フロー情報の学習し, 予測値を計算する LSTM-RNN の実装について説明する. LSTM の実装はニューラルネットワークでの計算や学習のためのオープンソースソフトウェアライブラリである Chainer を用いた.

5.4.1　LSTM 実装の概要

本稿では, Chainer でサポートされているクラス NstepLSTM を用いた. NstepLSTM は, cuDNN [9] を用い最適化されているため, 従来の LSTM よりも高速な学

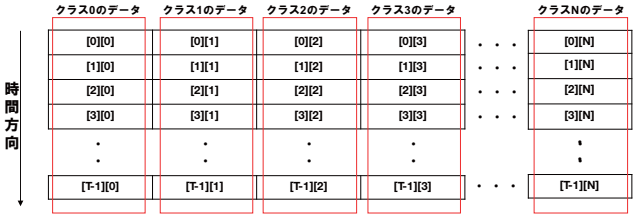


図 17　クラス別時系列データの概略図.

習が可能な LSTM の拡張クラスである [10].

5.4.2　学習時のデータ形式

5.3.3 項で作成した配列データを LSTM-RNN に入力する際の処理について説明する. 図 17 に, 本項で作成する, クラス別時系列データが記録された配列データの概略図を示す. 配列の列数はクラス数 (使用するクラスリスト内に含まれる AS パスの種類) であり, 配列の行数は学習させる日数や, データを区切る時間に応じて定まる. 例えば本稿では, 5 分間のバイト数や IP size の合計を記録した情報を作成するため, 1 クラス 1 時間あたりのデータ数は 12 であり 1 日分のデータ数は 288 となる. また, これを 7 日分作成し学習に使用しているので, 学習に用いる時系列データの全長 T は $T = 2,016$ である. 図 18 に, この学習データを RNN へ入力するのに適した形式にする手順を示す. ここで, 入力時の時系列データの長さを *sequence_size*, 入力に適したデータの形を Sequence data と定義する. 学習, 及び予測においては, *sequence_size* 分の連続したデータを入力とし, 出力は直後のデータの予測値である. *seaquence_size* 分の情報から直後の値を予測計算するため, *seaquence_size* は, 学習させ再現させたいデータの周期と同じ長さか, それ以上である必要がある. Sequence data を作成する際には, 全長 T の学習データを, 始点を 1 つずつずらしながら $L = sequence\_size + 1$ の長さで分割する. この分割されたデータを Sequence と呼ぶ. 長さ L 内の情報のうち, 末尾の値以外の *seqence_size* 分の情報は学習データ, 末尾の値は教師データである. 図 19 のように, *seaquence_size* 分の入力に対する出力と, 末尾の教師データとを誤差関数で比較し, 比較結果に応じてパラメータが調整され学習が進む. このようにして作成した Sequence data を, 上の Sequence から順に入力することで 全長 T の学習データ を学習させる.

予測の際は, $L = sequence\_size$ で Sequence data を作り, それぞれの入力に対する出力を予測値として別の配列に記録する.

5.4.3　学習に用いるフロー情報の単位

本稿で学習させた週単位の学習と曜日別の学習は, いずれも 7 日分の情報を用いモデルを作成した. これは WIDE BB が学術ネットワークであり, NOC には大学も含まれているためである. 大学は 1 週間を周期として授業や活動などを繰り返す. そのため, フロー情報を 1 週間分学習させ

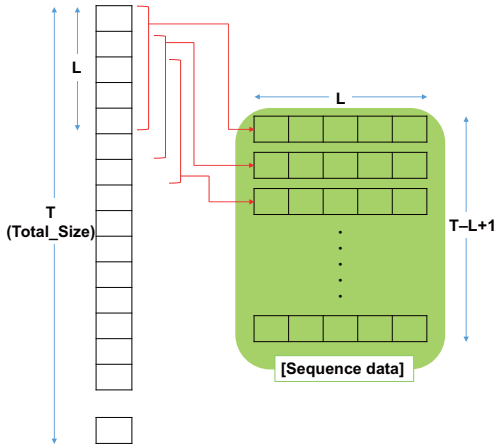


図 18　Sequence data の作成.

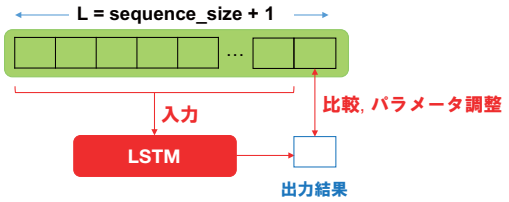


図 19　Sequence の入力と学習.

ることで, 1 周期分の変動を学習させる目的で 7 日分の情報を用い学習させモデルを作成した.

また Nstep-LSTM の *sequence_size* は 1 日分の情報にあたる 288 とした. *seaqence_size* は学習時に一度に入力するデータ長であり, *sequence_size* 分の入力値に対し, 直後の値を予測するため *sequence_size* を 288 にすることで, 丸 1 日の情報をもとに次の値を予測することになる. フローの時系列データは, ネットワークの利用が多い昼間や晩は値が大きくなりやすく, 一方深夜や早朝は利用が少なく値が小さくなりやすい. そのため時間軸を横軸にとったグラフでは 1 日を周期とした山なりになる傾向がある.

6.　評価方法

6.1　評価に使用したデータセット

6.1.1　WIDE BB (Back Bone) ネットワーク

本稿では, WIDE BB (AS2500) で観測されるフロー情報 (Netflow) と, 経路サーバにて計算された BGP の経路表である RIB を使用した. 図 20 に WIDE BB ネットワークのトポロジを示す. WIDE BB は学術ネットワークとして, stab に慶應義塾大学内の “矢上 NOC” や “藤沢 NOC”, 東京大学と接続する “根津 NOC” を構え, また, 大手町には西日本地区とのハブの役割を担う WIDE 関東地区のコア拠点である “大手町 NOC” を構える BB ネットワークである. また, 国内はもとより San Francisco, Los Angeles, Bangkok など海外にも拠点 (NOC, Network Operation Center) を持つ広大なレイヤ 2 およびレイヤ 3

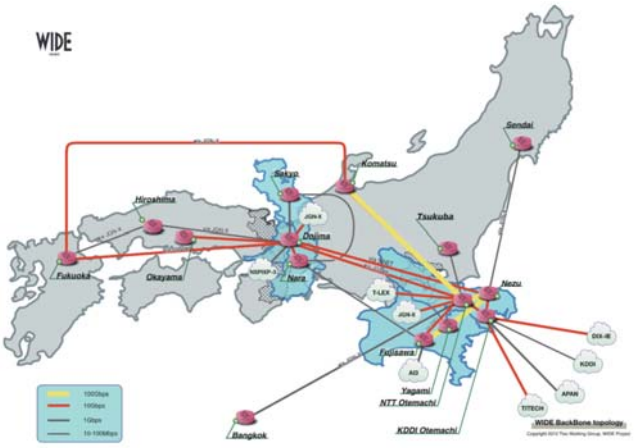


図 20　WIDE BB のトポロジ [11].

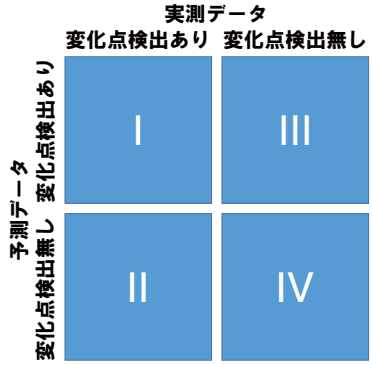


図 21　変化点検出を通した分類の区分.

ネットワークである. WIDE バックボーンネットワークは各接続組織の対外接続ネットワークとして活用されるだけではなく, インターネットの新技術を開発している研究者, 開発者らの新技術の運用実験の場としても頻繁に活用されている. 本稿では藤沢 NOC で観測された NetFlow と RIB を用いた.

6.1.2　各データの概要

本稿で使用した, フロー情報である NetFlow と, RIB について説明する. NetFlow は 1 時間毎の情報が 1 つのファイルとして記録されている. また RIB は各日について, 正午 12 時に記録された情報をその日の情報として使用した.

6.2　評価指標

評価には, 学習モデルによって予測された予測データと, 予測した日付の実測データを用いた. 予測データと実測データは共にクラス別時系列データの形式を持つ. まず両データにおいてクラス毎に変化点を検出し, 変化を検出した時刻と変化が検出されなかった時刻を記録する. 変化点検出の閾値は, 各日について変化量の平均値 a, 変化量の標準偏差 σ を計算し, 変化量が $a + \sigma$ を越える時刻を変化点とした. 続いて, 予測データと実測データで変化点が検出された時刻と検出されなかった時刻を比べる. 図 21 に示すように, 予測データでも実測データでも共通して変化点が検出された時刻を [Cat. I], 予測データで変化点が検出されなかったが実測データでは検出された時刻を [Cat. II], 予測データでは変化点が検出されたが実測データでは検出されなかった時刻を [Cat. III], 予測データでも実測データでも変化点が検出されなかった時刻を [Cat. IV] に分類した. 本評価方法において各カテゴリは以下の意味を持つ.

- Cat.I: 予測結果と実測が一致 (異常無し).
- Cat.II: 予測出来なかった変化点が観測された (異常).
- Cat.III: 予測上存在した変化点が観測されない (異常).
- Cat.IV: 予測結果と実測が一致 (異常無し).

以上 4 つのカテゴリに属する時刻の件数を用いて, モデルの妥当性と異常検知性能を評価する以下の 3 つの指標を設ける.

表 3　評価指標と意味.

指標	値が大きい意味	値が小さい意味
X 値	予測性能が妥当でない	予測性能が妥当
Y 値	無いはずのトラフィックがあった (予測出来なかった変化点が多い)	予測が正しい
Z 値	あるはずのトラフィックが無かった (存在しない変化点を多く予測)	予測が正しい

$$X = \frac{Cat.II + Cat.III}{Cat.I + Cat.II + Cat.III + Cat.IV} \quad (1)$$

$$Y = \frac{Cat.III}{Cat.I + Cat.III} \quad (2)$$

$$Z = \frac{Cat.II}{Cat.I + Cat.II} \quad (3)$$

以上の 3 つの指標をクラスごとに算出し, 学習方法毎の学習モデルの妥当性と異常検知性能を評価する. これらの指標とその意味を表 3 に示す. これらの指標をクラスごとに算出し, 学習方法ごとの学習モデルの妥当性を評価する. またネットワークに影響を及ぼしうるイベントが発生した日のデータを用い, 異常検知性能を試験した.

6.3　評価内容

6.2 節で定義した 3 つの指標を用いて, 学習方法別のモデルの妥当性と, 異常検知性能を評価する. 本節ではこれらを評価するのに用いるデータと, そのデータを用いる意義を説明する.

6.3.1　学習方法の評価

学習方法の性能評価は, 各学習方法で 2018 年 6 月 24 日から 30 日までのフローを予測する.

まず, 週単位学習には 2018 年 6 月 17 日から 23 日までの NetFlow を用いた. 各日のクラス別データの生成は, 17 日から 23 日までの RIB から生成したクラスリストを用いた.

表 4　異常候補日と詳細.

日付	イベント	詳細
10/17	Youtube 接続障害	a.m.10:15 ～ p.m.0:45 (JST)
11/22	三田祭	@慶大三田キャンパス
12/6	SoftBank 通信障害	p.m.01:39 ～ p.m.06:04 (JST)

次に, 曜日別学習の評価には 2018 年 5 月 6 日 から 6 月 30 日までの Net Flow を用いた. 7 週分のデータを曜日毎に学習させ, 当該曜日を予測させた. 各日のクラス別データの生成は, 17 日から 23 日までの各 RIB から生成した各曜日のクラスリストを用いた.

それぞれの学習方法で作成した同じ日付の予測データと実測データで導かれる評価指標で予測性能を評価した.

6.3.2　異常検出性能の評価

異常検知性能の評価では, ネットワークに影響を及ぼしうるイベントが発生した日 (異常候補日) について予測する. 予測データと実測データで導かれる評価指標で異常検知性能を評価した. 本稿では異常候補日として, 7 月 7 日, 10 月 17 日, 11 月 22 日, 12 月 6 日を予測した. 各日のネットワークに影響を及ぼしうるイベントは表 4 の通りである. 三田祭は慶應義塾大学三田キャンパスで行わる. 本稿で使用した NetFlow の観測点 (藤沢 NOC) がある慶應義塾大学湘南藤沢キャンパスの授業は, 開催中休講となる. そのため普段の木曜日と比べ, 湘南藤沢キャンパス内での研究室やゼミなどの活動が少ないと同時に, スマートフォンなどの端末でキャンパス内の Wi-Fi を使用する学生は少ないと考えられる.

7.　結果

7.1　学習方法毎の結果

7.1.1　週単位学習によるモデルの指標値算出結果

図 22 から図 24 は, 週単位学習によるモデルの X 値, Y 値, Z 値をクラスごとに算出し箱ひげ図で表した結果である.

X 値は第 3 四分位数 に当たる箱の上辺はいずれも 0.4 未満, 第 1 四分位数に当たる箱の下辺はいずれも 0.4 未満であった. 最大値はばらつきがあったが, 最小値は箱の下辺と等しかった. Y 値, Z 値は, 指標が取りうる最高値 (= 1.0) と箱の上辺が接した. これは多くのクラスで予測データと実測データとの変化点が食い違ったことを示す. X 値が小さいモデルは予測データと実測データの変化が検出された点, 変化が検出されなかった点の不一致が少ないことを意味するが, Y 値, Z 値の大きさを踏まえると, 予測データにおいても実測データにおいても変化点を検出しなかった点が多かったため, X 値が小さい値になったと考えられる.

以上 3 つの指標の結果から, 直前の 1 週間のデータを学習させる週単位学習モデルは予測性能が十分ではなく, 異常状態の検知には適さないモデルであると考えられる.

7.1.2　曜日別学習によるモデルの指標値算出結果

図 25 から図 27 は, 曜日別学習によるモデルの X 値, Y 値, Z 値をクラスごとに算出し箱ひげ図で表した結果である.

X 値はいずれの曜日においても, 図 22 に示した週単位学習によるモデルの結果より小さい値を記録した. 箱は 0.02 以下に沈み, 最大値も 0.2 程度にとどまっている. Y 値もいずれの曜日においても, 図 23 に示した週単位学習によるモデルの結果より小さい値を記録した. 週単位学習では 1.0 に接していた箱は, 曜日別学習では 0.0 に接し, 中央値はいずれの曜日でも 0.5 以下であった. 曜日別学習による学習では週単位学習に比べ, 多くのクラスで, 予測データで検出された変化点が実測データにおいても変化点として検出されていることが分かる. Z 値もいずれの曜日においても, 図 24 に示した週単位学習によるモデルの結果より小さい値を記録した. 箱はいずれの曜日でも 0.0 から 0.5 にかけ発生し, 月曜日, 金曜日以外の曜日では中央値は 0.05 を下回った. 曜日別学習による学習では週単位学習に比べ, 多くのクラスで, 実測データで検出された変化点が予測データにおいても変化点として検出されていることが分かる. また曜日毎に比較すると, 日曜日が他の曜日と比べ, Y 値における箱の上辺と, Z 値における中央値の値が大きかった. これについては 7.2.1 で後述する. 3 つの指標の結果から, 曜日別学習は週単位学習に比べ, 多くのクラスにおいて正確な変化の予測が行われていることが分かる. これは本稿で使用した WIDE BB (BackBone) ネットワークが学術ネットワークであることが起因していると考えられる. 曜日別学習でモデルを構築した 藤沢 NOC は慶應義塾大学湘南藤沢キャンパス内の NOC である. 大学は曜日で授業やミーティングなどのイベントがあるため, ネットワークの利用が多い時間やその頻度が定まりやすい. 特に, 大学を訪れる学生, 職員が極端に少ない日曜, 祝日は平日と全く異なる特徴を持つと考えられる. 週単位学習ではそれらの 1 日ごとに特徴が異なるフローを一括して学習させたことで, 特徴が押し並べて学習されモデルの妥当性に差が生じたものと考えられる.

7.2　異常検知試験の結果

7.1 の結果をもとに, 本項での異常検知性能の評価は曜日別学習による指標値を用いる. 図 28 から図 30 は, 表 4 の日付を対象とした, 週単位学習によるモデルの X 値, Y 値, Z 値をクラスごとに算出し箱ひげ図で表した結果である. “Y1”, “Y2” は Youtube の接続障害, “三田” は三田祭, “SB1”, “SB2” は SoftBank の通信障害が発生した日の結果を示している. Youtube と SoftBank はそれぞれ 2 種類存在する. それぞれの 1 は通常の曜日別学習通り過去 7 週

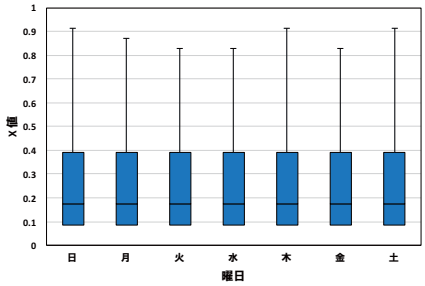


図 22　週単位学習の X 値.

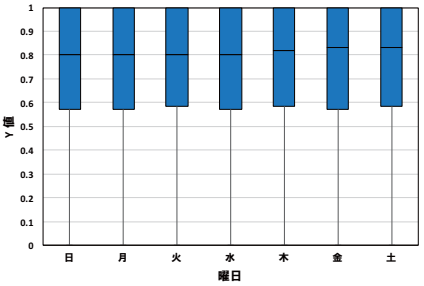


図 23　週単位学習の Y 値.

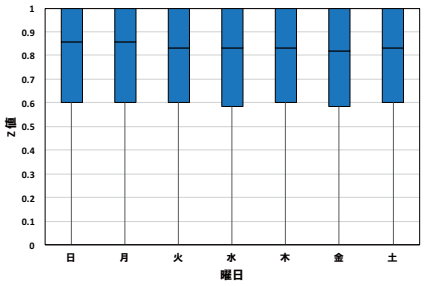


図 24　週単位学習の Z 値.

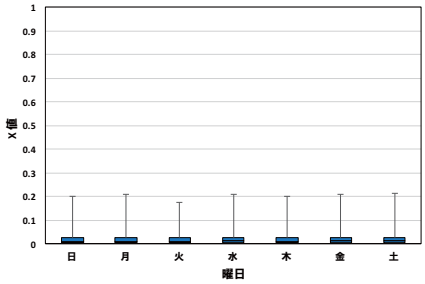


図 25　曜日別学習の X 値.

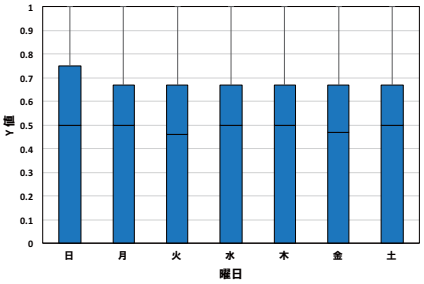


図 26　曜日別学習の Y 値.

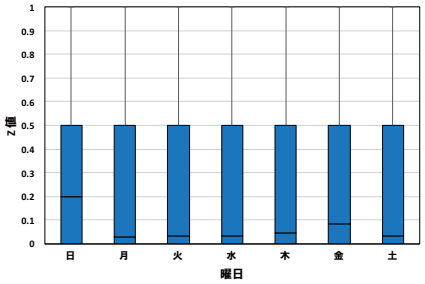


図 27　曜日別学習の Z 値.

のデータを用い学習させている．一方, 2 は 7 週のデータのうち異常データを含んでいる可能性がある日のデータを取り除き, 残ったデータを用い学習させている．例として, SB1 では 12 月 6 日の曜日別学習に, 三田祭が開催された 11 月 22 日のデータが含まれている．Y2, SB2 ではこれらのデータを取り除き, より正確なモデルの作成を試みた．

図 29 に示した Y 値は, 図 26 に示した 6 月最終週の曜日別学習の Y 値の結果と比べ, 第 3 四分位数を示す箱の上辺が僅かに大きい値を記録した．図 30 に示した Y 値は, 図 27 に示した 6 月最終週の曜日別学習の Z 値の結果と比べ, 箱の位置は変わらないものの, 中央値が 0.1 前後を記録した．また Y1 と Y2, SB1 と SB2 について, それぞれ学習データを吟味した 2 の方が Z 値の中央値が大きくなった．これは学習データから異常を含む可能性があるデータを取り除いたことで正常なデータが正確に学習され, 当該の日について異常検知を示す指標値が大きくなったと考えられる．

7.2.1　追加検証

ここで, 図 26, 図 27 に示した 6 月最終週の曜日別学習の Z 値の日曜日の結果に注目する．日曜日は他の曜日と比べ, Y 値における箱の上辺と, Z 値における中央値の値が大きかった．この日曜日の結果は図 29, 図 30 に示した結果に似通っている．これは 6 月の最終週の月曜日以降のデータを正常, 異常候補日のデータを異常と仮定すると, この日曜日 (6 月 24 日) で異常が検出されたことを意味する．そこで本項では 6 月 24 日のデータについて異常が見られるか検証する．

図 31 は 6 月 24 日と, その過去 4 週分の日曜日について, 各時間で観測されたトラフィックの総バイト数の変化をまとめたグラフである．このグラフから, 6 月 24 日の 0 頃時と, 9 時から 11 時頃において, 他の日付の結果よりも大きな値を記録していることが分かる．この挙動の原因は不明だが, 6 月 24 日において, 他の日付のデータには見られないこれらの異常データが含まれていることが確認できた．本稿の指標は, この特異な挙動の影響を受けたと考えられる．

8.　おわりに

本稿では, 大規模な BB ネットワークで観測されたフロー情報について, RIB がもつ AS パスの情報による Aggregate で定義したクラスごとに独立して学習させた．まず, この手法による学習モデルについて, 週単位学習と曜日別学習の 2 つの学習アプローチで予測性能を比較した．結果, 曜日別学習は作成したモデルは予測データと実測データの変化点が週単位学習よりも多くのクラスで一致し, より正確な予測が実現できていた．このことから, AS パスによる Aggregate により捉えられたデータの特徴が曜日別学習によって正確に学習されたことが示された．さらに, 曜日別学習で作成したモデルで異常検知性能を評価した．その結果, 異常候補日の Y 値と Z 値において, 6 月の最終週の結果と僅かに違いが見られた．また, 6 月 24 日 (日曜日) について, 各指標の結果が異常候補日と似通っていた．当該日について, 過去 4 週分の日曜日と挙動を比較したところ, 特異な挙動が確認された．

本稿の手法が, 汎用的なネットワーク異常検知を実現できるかを評価するには, 本稿で用いたデータとは異なる日

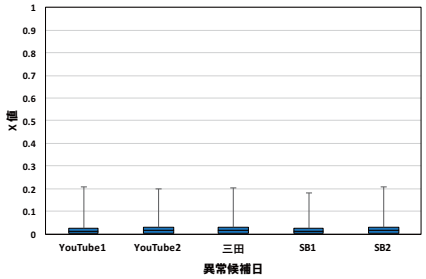


図 28　異常候補日の X 値.

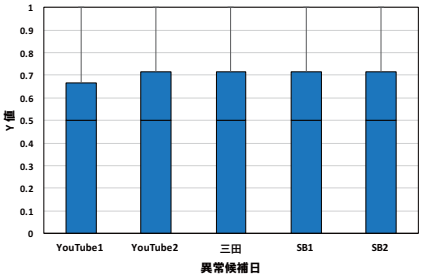


図 29　異常候補日の Y 値.

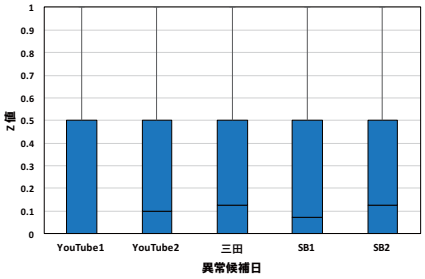


図 30　異常候補日の Z 値.

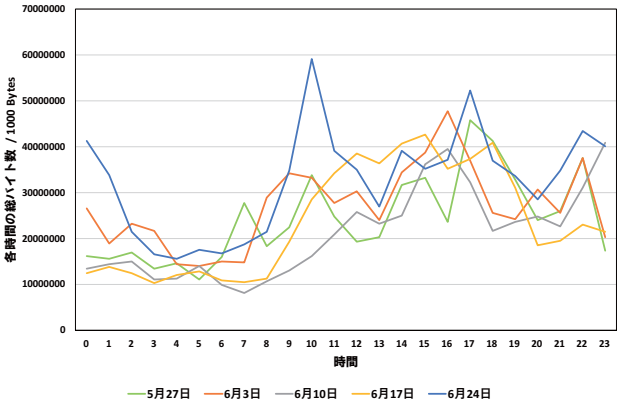


図 31　日曜日のトラフィックの様子.

付の情報を用いた学習モデルによる性能調査, 異常候補日を増やしての異常性能評価が必要である.

参考文献

[1] Roshan Kumar and Deepak Sharma. HyINT: Signature-Anomaly Intrusion Detection System. In *Proceedings of 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, July 2018.

[2] Jonghoon Kwon, Jehyun Leea, Heejo Lee, and Adrian Perrig. PsyBoG: A scalable botnet detection method for large-scale DNS traffic. *Computer Networks (2016)*, 97:48–73, March 2016.

[3] Olumuyiwa Ibidunmoye, Ali-Reza Rezaie, and Erik Elmroth. Adaptive Anomaly Detection in Performance Metric Streams. *IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT*, 15(1):217–231, March 2018.

[4] Tuan A Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks. In *Proceedings of 2018 IEEE International Conference on Network Softwarization (NetSoft 2018)*, pages 202–206, June 2018.

[5] Shao-Chien Chen, Yi-Ruei Chen, and Wen-Guey Tzeng. Effective Botnet Detection Through Neural Networks on Convolutional features. In *Proceedings of 2018 IEEE International Conference On Trust, Security And Privacy In Computing And Communication*, pages 372–378, August 2018.

[6] Elaheh Biglar Beigi, Hossein Hadian Jazi, Natalia Stakhanova, and Ali A. Ghorbani. Towards Effective Feature Selection in Machine Learning-Based Botnet Detection approaches. In *Proceedings of 2014 IEEE Conference on Communication nad Network Security*, pages 247–255, December 2014.

[7] 岡谷　貴之. 深層学習機械学習プロフェッショナルシリーズ. 集英社, April 2015.

[8] Red-Black-Tree. `https://github.com/stanislavkozlovski/Red-Black-Tree/blob/master/rb_tree.py`.

[9] NAVIDIA cuDNN. `https://developer.nvidia.com/cudnn`.

[10] Orizuru: NstepLSTM. `https://orizuru.io/blog/machine-learning/nsteplstm/`.

[11] Network Diagrams of WIDE Backbone. `http://www.wide.ad.jp/About/report/pdf2015/usb/wide-memo-two-report-2015-00.pdf`.