修　士　論　文

題　　目　　CPSの異常検出のためのログ分析手法の比較

主任指導教員　　水野　修　教授

京都工芸繊維大学大学院 工芸科学研究科

情報工学専攻

学生番号　　16622036

氏　　名　　原田　禎之

平成30年2月9日提出

様式 2 号

学位論文の要旨（和文）

平成 30 年 2 月 9 日

京都工芸繊維大学大学院
工芸科学研究科長　殿

工芸科学研究科　　情報工学専攻
平成 28 年入学
学生番号　　16622036
氏　　名　　原田　禎之　㊞

（主任指導教員　　水野　修　㊞　）

本学学位規則第 4 条に基づき、下記のとおり学位論文内容の要旨を提出いたします。

1.　論文題目

CPS の異常検出のためのログ分析手法の比較

2.　論文内容の要旨（400 字程度）

昨今の情報化社会において，ソフトウェアがサイバー空間以外にも影響を及ぼすようなシステムである CPS（サイバーフィジカルシステム）が普及しつつある．CPS の例として挙げられるスマートグリッドや自動運転車などの例で顕著なように，CPS の異常な動作は重大な事故を引き起こす可能性があり，早期の異常検出が課題となっている．

そこで，本研究では CPS の異常をシステムログを教師なし機械学習手法により分析することにより検出を行う手法について提案及び比較実験を行った．提案した手法は以前の研究で我々が提案した手法である “LOF（Local Outlier Factor）を用いた手法”，今回新しく提案した “PCA（主成分分析）による次元削減技術を用いた手法”，“Isolation Forest（iforest）を用いた手法”，“RNN（リカレントニューラルネットワーク）を用いた手法” の 4 種類である．既存の手法として採用したものは “synoptic とよばれるシステムモデル推定技術を利用した手法” である．

実験の結果，PCA を用いた手法，RNN を用いた手法，LOF を用いた手法は既存手法であるシステムモデル推定技術を用いた手法と同等以上の性能があると結論付けた．

Comparison of log analysis methods for Anomaly Detection of CPS

2018 16622036 *HARADA Yoshiyuki*

Abstract

In today's information-oriented society, CPS, which is the system in which software influences other than cyberspace, is becoming widespread. As seen in smart grids, automatic driving vehicles, which is an example of CPS, abnormal operations of CPS may cause serious accidents. So, the technique for detecting anomaly in advance is required.

In this research, we aimed to detect abnormal behavior by analyzing a system log as information source for detection and using outlier detection method. For this purpose, we proposed 4 methods as Anomaly Detection method, using Local Outlier Factor (LOF), Isolation Forest (iforest), PCA and using Recurrent Neural Network (RNN) . And we also adopted a existing method, using the system model estimation algorithm.

We applied these five methods to actual CPS, and evaluated the accuracy of anomaly detection methods and analyzed what kinds of anomaly can be detected.

As a result of the experiment, it was concluded that the method using PCA, the method using RNN and the method using LOF have performance comparable to or better than the existing method, which using the system model estimation algorithm.

目　次

1. 緒言

昨今の情報化社会において，ソフトウェアがハードウェアを制御することによりサイバー空間以外にも影響を及ぼすことを可能とするシステムである CPS（サイバーフィジカルシステム）が普及しつつある．CPS の例として挙げられるスマートグリッドや自動運転車，IoT 機器などで顕著なようにこれらのシステムの異常動作は重大な事故を引き起こす可能性があり，早期の異常検出が課題となっている [1].

我々の研究ではこのような CPS の異常検出について，システムログを分析することにより検出を行う手法を取り扱う．既存の研究では特定のセンサーの値などについて異常検出を行う研究はなされているものの，複数のセンサーなどの連続値データとテキストラベルの離散データを用いて検出を行う手法については提案が少なく，また存在する既存手法もシステムモデルに何らかの仮定が必要なものがほとんどである．これらの手法は CPS のようなシステムモデルが複雑でシステムモデルの仮定が難しいシステムへの適用が困難となっている．

そこで我々は以前に統計的な外れ値がシステムなどの異常を示唆しているという仮説に基づき，教師なし機械学習手法の一種である統計的外れ値検出手法を応用した異常検出手法を提案した [2]．外れ値は統計的に他の値から外れた値という概念であり，異常とはシステムなどの通常とは異なる動作のことである．これらはそのまま対応する概念というわけではないが，我々は統計的な外れ値がシステムの異常を示唆しているという仮定に基づき，外れ値検出手法によりシステムの異常を検出しようとした．提案手法の概要は，まず分析対象のシステムログを実数からなる特徴成分列に展開する．その後，展開されたシステムログに対して外れ値検出手法 LOF（Local Outlier Factor）[3] を適用し，計算された外れ値度合い（外れ値スコア）を元に該当のシステムログが異常かどうかの判定を行うというものである．

本研究ではまず，以前の提案手法である **LOF を用いた手法**のバリエーションとして，外れ値検出アルゴリズムに LOF の代わりに PCA による次元削減による外れ値検出アルゴリズムや Isolation forest [4] と呼ばれる外れ値検出アルゴリズムを用いた手法の 2 手法を評価対象に加えた（以下 **PCA を用いた手法**，**Isolation Forest を用いた手法**と呼称）．更に RNN（リカレントニューラルネットワーク）の一種である LSTM を用いた異常検出手法 [5] も評価対象として採用した（以下 **RNN を用いた手**

法と呼称）．これらの評価対象となる 4 手法を評価する上で，既存の手法であるモデル推定アルゴリズム [6] を用いた手法（以下**モデル推定を用いた手法**と呼称）についてもベースラインとして実験の評価対象に採用し，最終的に合計 5 手法についての評価実験を行った．評価実験は対象として採用した実際の CPS である**あくあたん**に各手法を適用し，それらの結果を分析，評価するというものである．

本論文の 2 章以降の構成は，まず 2 章で研究背景や関連研究，提案手法の概要を説明し，3 章では評価実験に用いた環境や実装，評価データについて説明する．最後に 4 章にて結果の説明及び研究設問への回答を行い，6 章をもって納言とする．

本論文で設定した研究設問は以下の 3 つである．

- **RQ1:** 外れ値検出手法の適用でシステムログから異常を検出できるか
- **RQ2:** 比較した検出手法それぞれの検出精度及び計算コストはどの程度か
- **RQ3:** CPS への適用に適した異常検出手法はどのような手法か

2. 関連研究・背景

2.1 関連研究

異常検出手法はシステムの異常の検出のみならず，様々な対象について多様なアプローチ [7] [8] で行われている．

CPS はハイブリッドシステムと呼ばれる離散的な動作と連続的な動作が混在するシステムの一種である．ハイブリッドシステムに対する異常検出分野においては関連研究が多数存在する [9–13]．一方，これらの研究ではシステムモデルが存在することを前提としており，システムモデルが複雑でその全体を把握することが困難である CPS に対する適用が可能であるかどうかの疑問が残る．本研究で比較対象として用いたモデル推定による異常検出もシステムモデルの存在を前提としている．

先述したとおり，我々が以前に提案した LOF を用いた手法 [2] はシステムモデルの存在を前提とできないシステムのためのものである．この手法はシステムモデルに関係なく，システムログを高次元の特徴空間に展開して統計的な分析を行う手法である．このような高次元空間における外れ値検出にも既存手法が多数存在し [14]，本研究で提案した LOF を用いた手法のバリエーションについても既存の高次元空間での外れ値検出手法を適用したものである.

2.2 実験対象

2.2.1 対象システム

本研究では実験対象のシステムとしてソフトウェア工学研究室で複数の水槽に生息する水生生物の飼育管理を行っている**あくあたん** [15] を用いた．

あくあたんは頭脳となる中央サーバと情報収集用のセンサーや水温制御や給餌ロボットの動作のためなどのアクチュエータからなる CPS である．この CPS の主要部分は水槽，給餌ロボット，空調管理の 3 ユニットで構成されており，それらのユニットは twitter アカウントを操作用のフロントエンドとして持ち，自然言語による命令を受けて処理を行う場合もある．各ユニットの機能の詳細は以下のとおりである．

水槽

あくあたんが制御する水槽は主として3つ存在し，それぞれの水槽にはセンサーとして水温計，水位計が取り付けられており，アクチュエータによる水温や水位の制御に用いられてる．アクチュエータとしては冷却用ファン，照明が取り付けられており水槽の環境に対する制御を行っている．

給餌ロボット

給餌ロボットは主として水生生物に対する給餌を行うロボットである．このロボットは小さな魚などへの給餌とイモリなどへの給餌を担当しており，大小の餌を別々に投入することが可能である．給餌は毎日正午に行われるが，定期的な給餌以外にもtwitterフロントエンドからの命令により給餌を行うこともある．

また，給餌ロボットの一部は水槽間を巡回する仕組みや撮影用のカメラを備えており，定期巡回時やtwitterフロントエンドから命令があった時に撮影を行いアップロードすることが可能である．

空調管理

あくあたんの制御する空調は水槽のある本学8-320の空調である．制御方法は室内のエアコン制御パネルのメインスイッチにアクチュエータが取り付けられており，オンオフによる制御を行うものである．

その他（twitterフロントエンドやウェブフロントエンド等）

先述したとおり，あくあたんはtwitterアカウント(注1)をフロントエンドとして操作を受け付けるようになっており，更に他のアカウントの発言を分析して学習することによる簡易な雑談対話モジュールも組み込まれている．他のアカウントの発言を分析する際には感情推定アルゴリズムも用いており，発言から推定される感情がマイナス方向やプラス方向へ大きく振れていた場合には特定の返信を行うなどの機能も存在する．感情推定アルゴリズムの動作履歴もシステムログに反映されている．

余談ではあるがtwitterアカウントでは“しりとり”や日々の挨拶に付き合うなど，頻繁に会話をするアカウントに対して内部で評価を行っている．この評価の程度によって先述のロボットへの命令権が付与される仕様となっている．更に，昨今では本学のソフトウェア工学研究室の学生居室の冷蔵庫の開閉管理なども当CPSの一部で行っており，それらの動作記録もシステムログに反映されている．冷蔵庫の開閉管理については閉め忘れなどの通知をtwitterフロントエンド経由で行うなど，実用

(注1)：https://twitter.com/sel_aquarium

的なものとなっている．また，twitter フロントエンド以外にもウェブページ(注 2) を持っており，そこでは水槽の状態のリアルタイム配信を見ることができる．

2.2.2 実験データ

あくあたんの出力するログは図 2.1 のような 5 列からなる CSV 形式で得ることができる．各列の内容はログエントリの ID（ID），タイムスタンプ（timestamp），ログエントリのコマンドの種類（label），数値データ（value），テキストデータ（note）である．ログの種類はあくあたんのシステムを制御するコマンドと対応しており，例えば図 2.1 の例における ID48 のエントリに存在するコマンド “pressure” は大気圧センサから値を読み出すコマンドであり，実数データである 1001.9 は読み取られた大気圧情報である．テキストデータには数値化出来ない，もしくはしないほうが意味上都合がいい情報が格納されており，例えば図 2.1 の ID54 のエントリでの “light2_status” コマンドの “off” は水槽の照明 2 を消灯したという意味である．

このシステムログには抽出した時点で 2014 年 5 月から 2016 年 7 月までの 1,027,547 エントリが存在し，本実験ではこのうち約 1,000,000 件を分析対象とした．また，コマンドの種類は合計で 130 種類存在し，そのうち 46 種類が水槽管理の目的からは逸脱したものであったため，後述するフィルタ処理の対象とした．

2.2.3 ログに見られる異常の例

本実験で用いるシステムログから見つかる異常の例として，図 2.2 のような例が挙げられる．この例では ID112–118 のシステムログのうち，ID115 のログエントリの “humidity” コマンドの数値データが 0 となっている．“humidity” コマンドは水槽の設置されている室内の湿度を計測するコマンドである．本来湿度センサーが 0 を返すことは考えづらく，センサーの故障などの原因によりセンサーからの値読み出しが失敗していると考えられる．このようにシステムログのシーケンスの中に異常が存在するものを主に本研究では検出対象とする．

(注 2)：http://se.is.kit.ac.jp/aquarium/

```
48,"2014-05-04 16:40","pressure",1001.9,NULL
49,"2014-05-04 16:40","water2",26.6,NULL
50,"2014-05-04 16:40","water1",24,NULL
51,"2014-05-04 16:40","water3",25.9,NULL
52,"2014-05-04 16:40","air",25,NULL
53,"2014-05-04 16:40","humidity",30,NULL
54,"2014-05-04 16:46:03","light2_status",NULL,"off"
55,"2014-05-04 16:46:03","light2_ontime",NULL,"08:02"
56,"2014-05-04 16:46:04","light2_offtime",NULL,"16:46"
```

図 2.1: 対象システムにより出力されたシステムログの一例

```
112,"2014-05-04 17:30","water1",24.1,NULL
113,"2014-05-04 17:30","water3",25.9,NULL
114,"2014-05-04 17:30","air",25,NULL
115,"2014-05-04 17:30","humidity",0,NULL
116,"2014-05-04 17:40","cputemp",47.1,NULL
117,"2014-05-04 17:40","pressure",1006.9,NULL
118,"2014-05-04 17:40","water2",26.7,NULL
```

図 2.2: 対象システムのログにみられる異常の例

2.3 提案手法

2.3.1 全手法に共通する処理

(1) システムログデータのクレンジング

2.2.1 章で先述したように，本実験で用いる CPS のシステムログには CPS 本体の分析の際に不必要と考えられる要素が多く存在する．その為，後述する前処理を適用する前に不要と思われる種類のログをコマンド名から判別して除去するフィルタをかけた．

(2) システムログデータのチャンク分割

また，分析対象としたシステムログの範囲は広すぎるため，各手法を適用するのに適切な範囲で切り分けてチャンクと呼ばれる単位にして扱った．チャンクの範囲は原則 100,000 エントリとしたが，モデル推定を用いた手法のみメモリ量の制約があったため 30,000 エントリで 1 チャンクとした．以後の実験は特に断りのない限りこのチャンクを一度の入力として行った．

(3) 異常スコアの計算

チャンクごとに前処理を行った後，各アルゴリズムによって異常度を示す**異常スコア**を計算した．

2.3.2 前処理手法

(1) 統計的外れ値検出のための数値ベクトル化

統計的外れ値検出手法を用いる前のベクトル化に際しては過去の我々の研究 [2] と同様の手法で特徴ベクトル化とウィンドウ化を行った．この手法は図 2.3 及び以下に説明するとおりである．

各ログエントリの数値化

まず最初に，入力されたシステムログの中の各ログエントリを timestamp，label，value，note[0]–note[3] の 7 要素に分割して数値化を行う．timestamp は入力システム

ログの最初のログの記録された時刻からの相対時刻を秒単位で表現したものであり，label はログエントリの label の種類を表現する整数，value はログエントリの value の値である．note[0]–note[3] はログエントリの note を md5 ハッシュ化し，その結果の下位 2bit を取り出してその結果が 0 であれば note[0] のみが 1，note[1]–note[3] が 0 となり，1 であれば note[1] のみが 1 と，2 であれば note[2] のみが 1，3 であれば note[3] のみが 1 となる one-hot ベクトルとする．

各ログエントリの one-hot 表現化

次に，数値化した各ログエントリを one-hot 表現のような高次特徴ベクトルに変換する．（図 2.3 の One-hot representation of log entries 参照）

特徴ベクトルの正規化

作成した特徴ベクトルを各要素の平均が 0，分散が 1 となるように正規化する．

ウィンドウ化

正規化した特徴ベクトルを所定のウィンドウサイズ分の個数ずつ連結して最終的な特徴ベクトルとする．（図 2.3 の Windowed vector 参照）本研究では実験対象の CPS のシステムログにおけるログエントリがもつ，前後のログエントリとの関係を認識するために必要なウィンドウサイズは概ね 11 程度で十分と考えたため，ウィンドウサイズは 11 で固定とした．

(2) モデル推定手法のための前処理

本研究で synoptic を用いたモデル推定を行うにあたって，入力であるシステムログを処理単位に分ける必要があった．そのため，システムログに記録されたそれぞれのエントリのタイムスタンプを元に，エントリ間の時間が一定以上開いているところでシステムログを分割し，それらを処理単位としてそれぞれの処理単位に固有の ID を付与したものを入力とした．本研究ではエントリ間の時間については 300 秒以上空いているものに意味的な区切りがあると仮定し，300 秒以上の空白がある場所で分割を行った．

また，synoptic を用いた手法ではタイムスタンプ及びコマンド名のみを入力として用いた．

モデル推定手法のための前処理前後の例を図 2.4 及び図 2.5 に示す．図 2.5 の例で

は ID が 926997，927007 のログエントリの直後で分割が行われており，分割により生成される 3 つの分割後の範囲のログエントリの集合の固有 ID は 130636，130637，130638 である．

以後，この分割を行ったあとの同じ固有 ID を持つログエントリの集合をトレースと呼称する場合がある．

(3) RNN 利用手法のための前処理

RNN 利用手法のための前処理はシンプルで，各ログエントリの label，note をそれぞれの種類を表す整数ラベルに置き換え，value と合わせて入力となる 3 シーケンスを作成している．この際，note の中に時刻情報が存在し，value が空であるログエントリに対して，note の示している時刻の 0 時 0 分からの秒数を計算して value に代入する操作を行っている．これは，特に照明を制御するコマンドに含まれる点灯予約時刻や消灯予約時刻について，本来連続値として記録されるべき値であると考えたためである．RNN を用いた手法では入力が連続値である場合を分けることによる精度向上が見込まれるため，特別にこのような前処理を行っている．

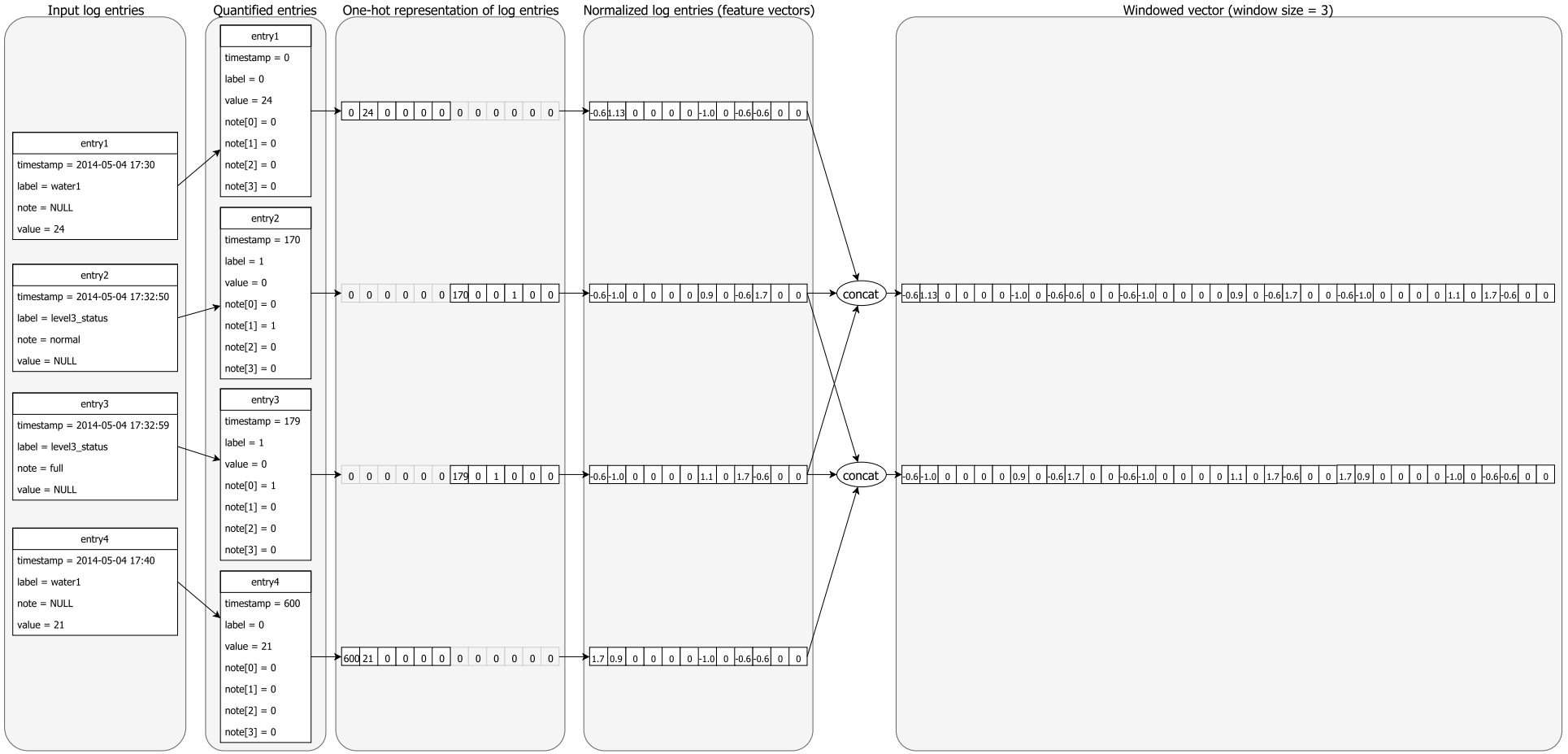


図 2.3: 統計的外れ値検出で用いた数値ベクトル化手法

2.3.3 提案手法

(1) 統計的外れ値検出

LOF を用いた手法

LOF [3] は特徴ベクトルの集合から各特徴ベクトルの外れ値スコアを算出するアルゴリズムの一つである．外れ値スコアの計算に際しては，特徴空間の中の特徴ベクトルは，外れ値であればあるほど特徴空間内で密度が疎な場所に存在するという仮定に基づいている．

PCA を用いた手法

PCA（principal component analysis）とは複数の特徴量からなる特徴ベクトルに対し，特徴量の数の削減を行う手法である．削減に際しては特徴量を削減した後の特徴ベクトルの集合の分散を最大化する，すなわち情報量を最大限に維持したまま特徴量の数を減らすことにより，より情報量の少ない特徴量を削減するという手法である．

本研究では PCA を特徴量を削減する目的ではなく，削減した際に失われた情報量に着目することで外れ値の検出を行うために用いた．

Isolation Forest を用いた手法

Isolation Forest（iforest）[16] とは二分木のアイディアを用いた外れ値検出アルゴリズムで，特徴空間のランダムな次元で特徴ベクトル同士を分離しようと試み，分離できるまでに掛る手数が小さいほど外れ値らしいと判別するというアルゴリズムである．

(2) モデル推定，及び RNN による外れ値検出

モデル推定を用いた手法

システムの状態遷移を推定する手法には様々なアルゴリズムが存在するが，実装が公開されており離散ラベルデータへの適用が容易であることから本研究では実装の一つである synoptic [6] を利用した．

本研究では synoptic を適用して外れ値スコアを計算するために以下の手順を実装した．

```
926991,"2016-05-21 16:15:16","droid_swing_h",3,NULL
926992,"2016-05-21 16:15:16","droid_swing_v",21,NULL
926993,"2016-05-21 16:15:16","droid_status",NULL,"Operating"
926994,"2016-05-21 16:15:19","droid_movediff",NULL,"-9,1"
926995,"2016-05-21 16:16:08","droid_status",NULL,"Waiting"
926996,"2016-05-21 16:16:08","droid_tank_pos",2,NULL
926997,"2016-05-21 16:16:08","droid_lift_pos",6,NULL
926999,"2016-05-21 16:20","level_3",7,NULL
927000,"2016-05-21 16:20","level_2",3,NULL
927001,"2016-05-21 16:20","water3",26.5,NULL
927002,"2016-05-21 16:20","water2",27.6,NULL
927003,"2016-05-21 16:20","pressure",1005.6,NULL
927004,"2016-05-21 16:20","level_1",11,NULL
927005,"2016-05-21 16:20","water1",25.9,NULL
927006,"2016-05-21 16:20","humidity",27.7,NULL
927007,"2016-05-21 16:20","air",27.8,NULL
927008,"2016-05-21 16:21:01","fan1_status",NULL,"on"
927009,"2016-05-21 16:21:01","fan1_hightemp",NULL,"26"
927010,"2016-05-21 16:21:01","fan1_lowtemp",NULL,"25"
```

図 2.4: モデル推定手法のための前処理例：前処理前

```
926991, 0.0, droid_swing_h, 130636
926992, 0.0, droid_swing_v, 130636
926993, 0.0, droid_status, 130636
926994, 3.0, droid_movediff, 130636
926995, 52.0, droid_status, 130636
926996, 52.0, droid_tank_pos, 130636
926997, 52.0, droid_lift_pos, 130636
926999, 0.0, level_3, 130637
927000, 0.0, level_2, 130637
927001, 0.0, water3, 130637
927002, 0.0, water2, 130637
927003, 0.0, pressure, 130637
927004, 0.0, level_1, 130637
927005, 0.0, water1, 130637
927006, 0.0, humidity, 130637
927007, 0.0, air, 130637
927008, 0.0, fan1_status, 130638
927009, 0.0, fan1_hightemp, 130638
927010, 0.0, fan1_lowtemp, 130638
```

図 2.5: モデル推定手法のための前処理例：前処理後

1. システムログを一定規則によりトレースごとに分割する
2. 分割したトレースを用いてシステムの状態遷移，及びその確率を推定する
3. 推定した状態遷移確率から全てのログエントリの出現確率を計算し，確率が低いものを異常であると判定する

分割の後 synoptic を用いて状態遷移を推定した結果は遷移確率付き有向グラフで得られる．この推定結果を用い，Algorithm1 に擬似コードで示したアルゴリズムにより各ログエントリの出現確率を計算した．

RNN を用いた手法

RNN による外れ値検出手法の原案は産業技術総合研究所 山形賴之らによるものである．

本研究では原案のネットワーク構造を改良し，損失関数（Loss function）やネットワークの出力部を最適化した手法を提案した（図 2.6）．

このネットワークは LSTM [5] を用いてシステムログの中の 1 番目〜t 番目までのログエントリを入力として受け付ける．その際の予測対象は $t+1$ 番目のログエントリであり，これを予測した際の誤差が小さくなるように重みを訓練する．

ネットワークから異常検出のためのスコアを得るために，一度ログを学習させた後に再度 1 番目〜t 番目までのログエントリを入力して $t+1$ 番目のログエントリを予測した際の予測誤差を外れ値スコアとして用いる．

必要なパラメータについて，入力段の線形結合層の出力以降の中間層の数はすべて 300 とした．

2.3.4 パラメータの調整

評価対象手法のうち，パラメータの必要ないモデル推定を用いた手法とパラメータの調整が時間的に困難であった RNN を用いた手法以外について，パラメータの調整を行った．LOF，Isolation Forest，PCA について，主要なパラメータはどの手法も 1 つであり，それぞれ k，$samples$，$n\_components$ と呼称する．本研究ではこれらの値について，$k = 130, samples = 16384, n\_components = 512$ とした．

パラメータの決定のために，これらの値を変更しながら評価用データセットを用いて 3.2.2 章で後述する方法で ROC 曲線の AUC を計算し，この値を最大化するパラ

メータを採用した．パラメータの候補として用いた値は，k は (10, 30, 50, 60, 80, 100, 130, 140, 150, 160, 190, 200, 270, 510, 610, 660, 810, 860, 980, 1100, 1130, 1430, 1480, 1490, 1950, 2900)，$samples$ は (16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536)，$n\_components$ は (32, 64, 128, 256, 512) である．

Algorithm 1 synoptic の出力を用いたエントリ出現確率の計算

$n \Leftarrow$ 状態遷移図の状態の数

$E \Leftarrow n$ 要素の配列，i 要素目は状態 i になるために最後に受理するログエントリ

$N \Leftarrow n$ 行 n 列の配列，i 行 j 列目は状態 i のとき $E[j]$ を受理して状態 j になる確率

$P \Leftarrow n$ 要素，すべての値が $1/n$ の一様分布の配列

$prob \Leftarrow$ ログエントリの数と同じサイズの配列，結果が格納される

$counter \Leftarrow 1$

for $entry$ in log **do**

  $P_{new} \Leftarrow n$ 要素，すべての値が 0 の配列

  for $i = 1$ to n **do**

    if $entry == E[i]$ **then**

      $P_{new}[i] \Leftarrow 0$

      for $j = 1$ to n **do**

        $P_{new}[i] \Leftarrow P_{new}[i] + P[j] \times N[j][i]$

      end for

    end if

  end for

  $prob[counter] \Leftarrow \mathrm{sum}(P_{new})$

  $P_{new} \Leftarrow P_{new}/\mathrm{sum}(P_{new})$

  $P \Leftarrow P_{new}$

  $counter \Leftarrow counter + 1$

end for

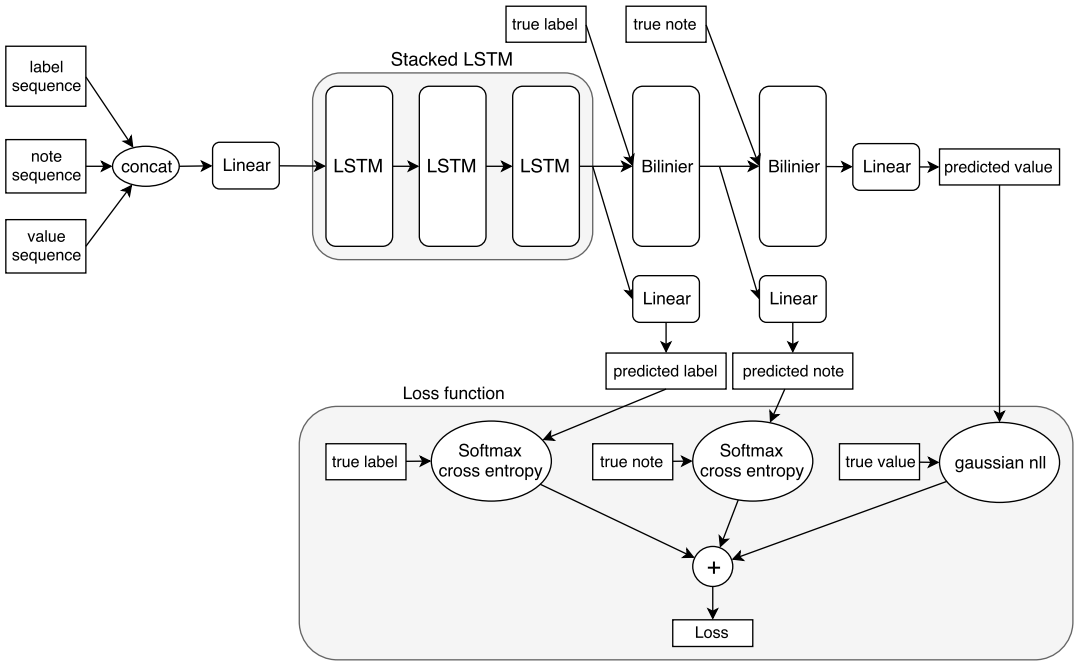


図 2.6: RNN 手法で用いたニューラルネットワーク構造

3. 研究手法

3.1 実験環境及び実装

3.1.1 実験環境

本研究では実験環境として Intel®Xeon®CPU E5-2630 v4 @2.20GHz 2 基，RAM64GB，NVIDIA GeForce®GTX 1080 ti 1 基，OS に Ubuntu 16.04 を搭載した計算機を用いた．

3.1.2 実装

実験のための実装は Python™3.5 [17] を開発言語として用い，前処理にはデータ分析ライブラリ pandas [18] を用いた．その他のライブラリとして，LOF を用いた手法では Java で記述されたデータマイニングライブラリ ELKI [19] を，モデル推定手法では同じく Java で記述された実装 synoptic [6] を利用した．また，Isolation Forest，PCA の実装として scikit learn [20] を，RNN 手法の実装のためには Chainer [21] を利用した．

3.2 評価

3.2.1 評価用データセット

一般的に教師なし機械学習におけるモデルの性能評価には一部のデータに人手により正解ラベルをつけて評価用データとすることで定量的な評価を行う．

本実験でも一部の期間のシステムログに CPS の開発者と協議のもとで異常か正常かの正解ラベルをつけることにより評価用データセットとした．本実験で用いた正解ラベル付きシステムログは ID が 248299〜268299 の範囲のものである．

3.2.2 評価指標の計算

異常の検出精度を評価するためには一般的に分類器の評価で用いられる Precision（適合率），Recall（再現率），F1 値を利用する．これらは表 3.1 に示した混同行列を用いて式 3.1–式 3.3 に示したとおり定義される．本実験においては Precision は検出

したものの中で実際に異常だった率，Recall は実際に異常であるもののうち検出された率，F1 値は Precision と Recall の調和平均である．

また，以上の指標の他に ROC（Reciever Operating Charactristic）曲線，及び ROC 曲線から求められる AUC（Area Under the Curve），False Positive rate（FPr）も利用する．

ROC 曲線及び AUC は表 3.2 に示したデータの場合，図 3.1 に示したとおりの概形となる．この例では簡単のため間引きを行っているが，ROC 曲線は分類機の出力したすべてのスコアを閾値の候補として，それぞれの閾値により得られる True Positive rate（TPr）と False Positive rate（FPr）をそれぞれ y 座標，x 座標としてプロットしたものである．AUC は ROC 曲線下の面積であり，0 から 1 の値を取る．ROC 曲線はどの閾値でも TPr が FPr より高ければ高いほど上方向に広がり，その広がりが大きければ大きいほどよい分類器であると評価できる．ランダム分類器の場合はどの閾値をとっても TPr 及び FPr は 0.5 付近となり，ROC 曲線は概ね直線と，AUC は約 0.5 となる．

ただし，Precision，Recall，F1 値，FPr を計算するための閾値は出力されたスコア全てを閾値として F1 値を計算し，その値が最大となるものを採用した．

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.3}$$

表 3.1: 混同行列

		判定結果	
		True	False
真のクラス	True	TP	FN
	False	FP	TN

表 3.2: ROC の例で用いるサンプルデータ

スコア	真のクラス
87.0	True
86.0	False
85.0	True
84.0	True
81.0	False
71.0	True
68.0	True
62.0	True
55.0	False
50.0	True
41.0	True
38.0	False
21.0	True
18.0	False
17.0	True
11.0	False
7.0	False
3.0	True
2.0	False
0.0	False

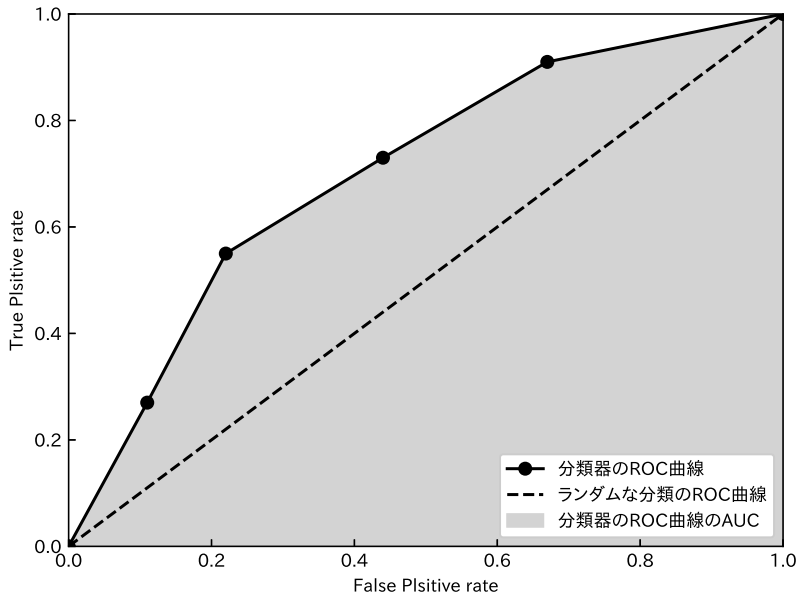


図 3.1: ROC 曲線と AUC の例

4. 結果

4.1 定量評価

評価用データセットを用いて提案手法の各種精度指標及び計算コストを得た結果は表 4.1 のとおりと，ROC 曲線は図 4.1 のとおりとなった．

ただし，計算時間は RNN を用いた手法以外はシステムログ全体に適用した実測時間であり，RNN 手法についてはシステムログの一部，全体の 1/10 に対して適用したものを 10 倍したものである．

また，定量評価の際に各手法が出力した異常スコアの分布をヒストグラムによって表し，閾値を用いて異常と判定された範囲を可視化したものを図 4.2 に示す．

4.2 定性評価

また，定量評価のみではどのようなログエントリが異常として検出されたかが不明なため，定量評価に用いたものと同じログの区間である，ID が 200000 から 300000 までの範囲であるログエントリからなるシステムログに各手法を適用した結果について，異常スコアが上位から 500 個であったログエントリを抽出してそれらのログエントリがどのような理由により異常として検出されたのかを分析した．この分析の結果は表 4.2 に示したとおりである．

表 4.1: 各手法の適用実験結果

手法	Precision	Recall	FPr	F1 値	AUC	時間 (h)	RAM(GB)	VRAM(GB)
PCA を用いた手法	0.821	0.878	1.833e-03	**0.848**	**0.999**	1.62	28.5	-
RNN を用いた手法	**0.974**	0.469	**5.054e-05**	0.633	0.977	775	-	**0.595**
モデル推定を用いた手法	0.0787	0.784	2.365e-02	0.143	0.96	7.88	60.7	-
LOF を用いた手法	0.207	**0.922**	7.761e-02	0.337	0.944	7.30	**27.9**	-
Isolation Forest を用いた手法	0.157	0.581	7.436e-02	0.248	0.915	**1.34**	28.0	-

4.2.1 異常を正しく検出できた例

表 4.2 の分類のうち，実際に異常と思われるものが検出されていたものは “構成端末の障害”，“管理者による手動操作”，“電力使用率の警報を受信”，“センサー値異常”，“droid_status の異常” の 5 分類であった．これらの分類の一例を以下に説明する．

構成端末の障害

分類 “構成端末の障害” は，システムを構成する一部の端末がネットワークの不調や端末自身の不調によりメインサーバから疎通できなくなったことによる異常である．また，疎通のできない状態から復帰した状態についても検出が行われていた場合はこの分類とした．この異常が発生したと認識されたシステムログの一例を図 4.3 に示す．

管理者による手動操作

分類 “管理者による手動操作” は，不調などによりシステムの動作に管理者からの手動操作を割り込ませる必要が生じ，自動的な制御が一部または全部失われた状態を異常として検出したものである．先述した “構成端末の障害” の一部を引き起こした場合にも自動的にこの状態となる．この状態についても復帰したことを検出している場合にも同じ分類に含めている．一例は図 4.4 に示したとおりである．

電力使用率の警報を受信

分類 “電力使用率の警報を受信” は，本学の使用電力が契約電力に対して余裕のない状態になったため，システムの一部を省電力状態に移行した状態を異常として検出したものである．この省電力状態への移行は本学の電力警報メール配信サービスとの連携によって実現されている．この状態を示すシステムログの一例は図 4.5 である．

センサー値異常

分類 “センサー値の異常” は，システムを構成するセンサーからの値読み出しが失敗したなどの理由で異常なセンサー値が記録されているものを異常として検出したものであり，原因としてはセンサーの故障やバグ，ネットワークの障害が挙げられる．この状態は一例として図 4.6 に示したとおりのシステムログから検出された異常である．

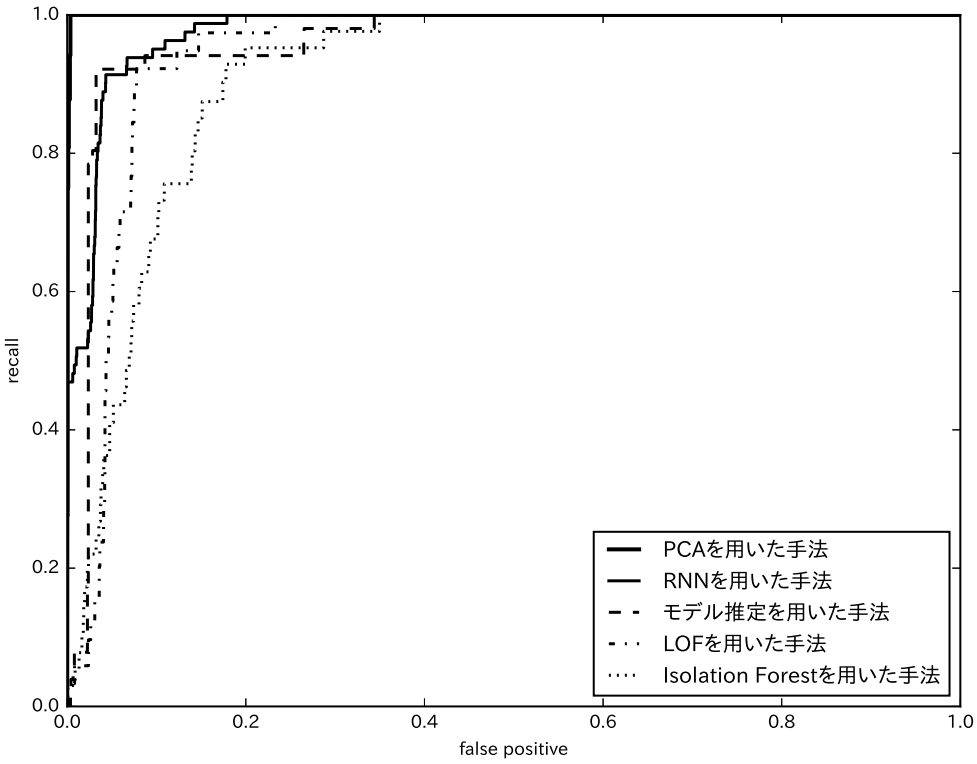
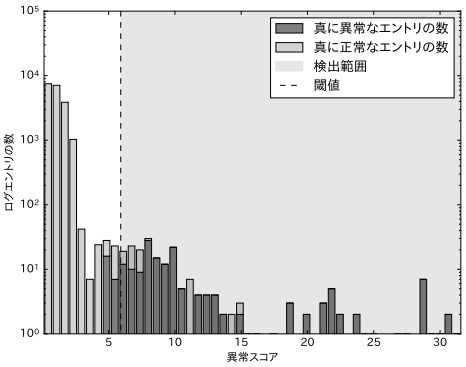
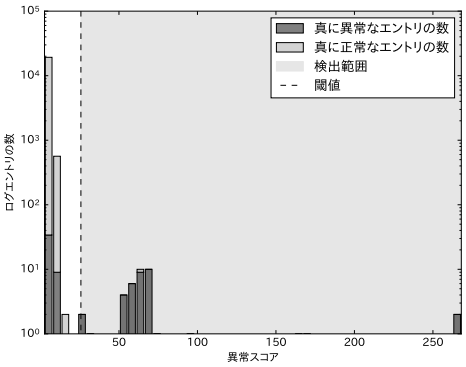


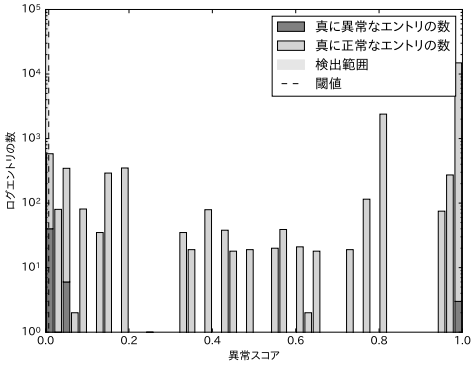
図 4.1: 各手法の ROC 曲線



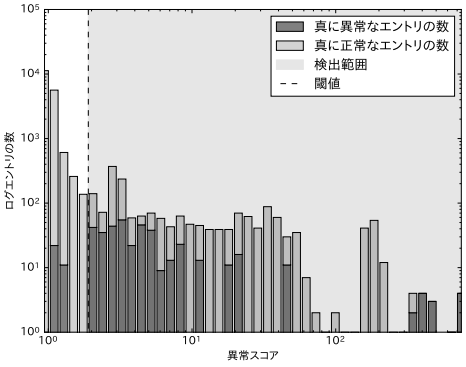
(a) PCA を用いた手法



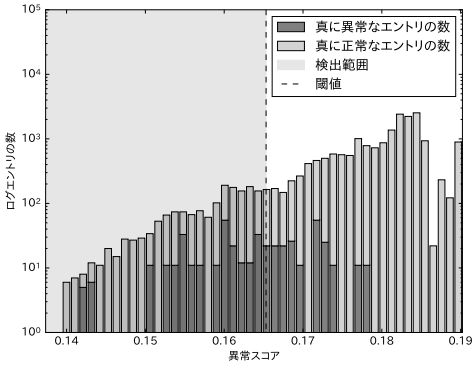
(b) RNN を用いた手法



(c) モデル推定を用いた手法



(d) LOF を用いた手法



(e) Isolation Forest を用いた手法

図 4.2: 各手法により得られた異常スコアの分布

表 4.2: 検出結果の内訳

分類		PCA	RNN	モデル推定	LOF	iforest
異常	構成端末の障害	100	8	5	126	7
	管理者による手動操作	91	0	17	200	33
	電力使用率の警報を受信	0	0	2	1	14
	センサー値異常	1	40	2	0	6
	droid_status の異常	0	0	1	0	0
正常	照明の制御	4	51	125	0	219
	エアコン制御	190	7	19	119	28
	定期自動給餌	64	85	164	0	157
	ファン制御	11	0	0	30	0
	ロボットの手動操作	39	9	10	0	19
	ロボットの定期動作	0	300	80	0	13
	センサーの定期動作	0	0	41	24	0
	カメラの定期動作	0	0	1	0	0
	例外動作と定期動作の混合	0	0	33	0	4
合計		500	500	500	500	500

droid_status の異常

分類 “droid_status の異常” はシステムのうちロボット部分の状態を表現するコマンドであり，表現される状態は “Operating” と “Waiting” が存在する．“Operating” 及び “Waiting” はそれぞれシステムのうち物理的なロボット部分を動作状態に移行する，及び動作状態を終了することを示している．このシステムでは “Operating” と “Waiting” の間には通常数エントリの動作命令が挟まり，その間は別の動作命令を受け付けない排他制御が行われている．一方で図 4.7 に示したシステムログでは “Waiting” のみが見られ，“Operating” が見られない．また，“Waiting” の直前のシステムログはロボットの動作命令ではないコマンドが実行されており，排他制御の失敗もしくは意図しない動作を示していると考えられる．

4.2.2 異常でないものを誤検出した例

照明の制御

水槽の照明を制御する際に実際の日照時間に合わせた制御を行うため，日の入りと日の出の時刻を反映した制御を行っている．そのため，毎日制御を行う時刻がずれており，そのずれが異常として検出された．

エアコン制御，ファン制御

対象システムは気温を制御することにより水温調整を行う機能も持っている．このシステムは気温の高低を検出するとエアコンを動作させる．その為，気温を読み取ったときに極稀に動作コマンドが発行される形となっており，その稀さが異常と検出される原因となった．

定期自動給餌

水槽の生物に対して給餌を行うため，毎日正午にロボットを動かして水槽を巡回，餌の投入を行う．この動作に用いられるコマンドの出現頻度はシステムログ全体から見た場合非常に稀であるため，その稀さが異常として検出される原因となっている．

ロボットの手動操作

この手動操作は twitter フロントエンドからの命令によるものである．異常として分類することも可能であるが，システムの動作としては正常なものであるため誤検出に分類した．

```
261273,"2015-01-04 08:40","water3",20.1,NULL
261274,"2015-01-04 08:40","air",15.5,NULL
261275,"2015-01-04 08:43:01","target_192.168.68.90_status",NULL,"Lost"
261276,"2015-01-04 08:45:11","target_192.168.68.90_status",NULL,"Found"
261277,"2015-01-04 08:50","pressure",1010.2,NULL
```

図 4.3: 分類 “構成端末の障害” の一例

```
222163,"2014-11-27 12:42:25","target_192.168.68.93_status",NULL,"Lost"
222164,"2014-11-27 12:43:25","droid_mode",NULL,"Manual"
222165,"2014-11-27 12:45:11","target_192.168.68.21_status",NULL,"Found"
222166,"2014-11-27 12:45:12","target_192.168.68.90_status",NULL,"Found"
222167,"2014-11-27 12:45:13","target_192.168.68.92_status",NULL,"Found"
222168,"2014-11-27 12:45:15","target_192.168.68.93_status",NULL,"Found"
222169,"2014-11-27 12:46:20","droid_mode",NULL,"Automatic"
222170,"2014-11-27 12:50","pressure",1014.7,NULL
```

図 4.4: 分類 “管理者による手動操作” の一例

```
271900,"2015-01-14 12:50","air",22.2,NULL
271901,"2015-01-14 12:54:24","target_B4:18:D1:4F:41:A3_status",NULL,"Found"
271902,"2015-01-14 13:00:00","power_warn",1,NULL
271903,"2015-01-14 13:00:04","light1_status",NULL,"off"
271904,"2015-01-14 13:00:04","light1_ontime",NULL,"07:03"
```

図 4.5: 分類 “電力使用率の警報を受信” の一例

```
222368,"2014-11-27 18:30","level_3",0,NULL
222370,"2014-11-27 18:50","level_3",0,NULL
222372,"2014-11-27 19:00","level_3",0,NULL
222374,"2014-11-27 19:10","level_3",0,NULL
```

図 4.6: 分類 “センサー値異常” の一例

```
222921,"2014-11-28 10:00","water2",24.8,NULL
222922,"2014-11-28 10:00","level_3",8,NULL
222923,"2014-11-28 10:00","water1",19.9,NULL
222924,"2014-11-28 10:00","water3",22.9,NULL
222925,"2014-11-28 10:00","air",20.9,NULL
222926,"2014-11-28 10:00:31","droid_status",NULL,"Waiting"
```

図 4.7: 分類 “droid_status の異常” の一例

ロボットの定期動作，センサーの定期動作，カメラの定期動作

実験対象のシステムはロボットやセンサー，カメラに対して定期的に動作命令を行っている．この動作命令の一部が異常として誤検出された．

例外動作と定期動作の混合

先述した照明や手動操作などの例外的な動作の直後のシステムログを異常だと誤検出したものである．

4.3 RQ への回答

前章までに示した実験結果を踏まえて 1 章にて設定した RQ1–RQ3 についての考察を以下の通り行う．

4.3.1 RQ1: 外れ値検出手法の適用でシステムログから異常を検出できるか

(1) 概要

評価対象として採用した 5 つの外れ値検出手法や異常検出手法について，直感的にはシステムログというコンテキストの上での外れ値はシステム上の何らかの異常を示している場合が多いと考えられるが，実際に外れ値検出手法を異常検出に適用することができるのかという点について実験結果を踏まえて考察する．

(2) 結果

表 4.2 に示した結果より，評価対象とした 5 手法全てにおいて実際に異常であるものを検出することが可能であった．特に PCA を用いた手法，RNN を用いた手法や LOF を用いた手法ではシステムの障害などの異常を十分に検出できており，これらの外れ値検出手法により検出される外れ値はシステムログ中の異常を示唆していると考えられる．

一方，手法によっては意味的には正常であるシステムログも異常として検出してしまうなど，議論の余地の残る結果となった．手法ごとの優劣については RQ2, RQ3 への回答で言及する．

4.3.2 RQ2: 比較した検出手法それぞれの検出精度及び計算コストはどの程度か

(1) 概要

各外れ値検出手法について，実際の応用を考えた場合にはなるべく高い検出精度を低い計算コストで実現する必要がある．この観点から各手法における検出精度及び計算コストを比較及び考察する．

(2) 結果

検出精度

まず，表 4.1 及び図 4.1 に示した定量評価結果より定量的な精度評価を行う．表 4.1 に示した通り，F1 値及び AUC について評価対象とした 5 手法の中では PCA を用いた手法が最も良い評価となった．一方で Precision 及び FPr は RNN を用いた手法が，Recall は LOF を用いた手法が最良という結果となった．一方精度が最低となったものは Isolation Forest を用いた手法であった．また，図 4.1 より，PCA を用いた手法，RNN を用いた手法，モデル推定を用いた手法の順で ROC 曲線の概形からも高性能であると考えられる．特に PCA を用いた手法は非常に性能が高い手法であると結論付けられる．

次に表 4.2 に示した詳細分析結果より詳細な精度評価を行う．それぞれの手法で検出されたログエントリについて，実際に異常であると考えられるものの割合は表 4.3 に示したとおりとなった．この結果を鑑みた場合，精度は LOF を用いた手法が最も高く，次点で PCA を用いた手法が高精度であると考えることもできる．

計算コスト

計算コストについては表 4.1 の結果を参考に考察を行う．計算に要したタイムコストは比較対象の中では Isolation Forest を用いた手法が最も優れており，次点で PCA を用いた手法が優れているという結果となった．これら 2 手法の間には大きな差は無かった．一方，最もタイムコストを必要とした手法は RNN を用いた手法で，最も計算時間のかからなかった Isolation Forest を用いた手法の 500 倍以上を必要とする結果となった．RNN を用いた手法以外の中ではモデル推定を用いた手法が最も悪い評価となったが，こちらは Isolation Forest を用いた手法の 6 倍程度で済んでいる．

表 4.3: 分析結果の中で実際に異常と考えられるものの割合

手法	割合（%）
PCA	38.4
RNN	9.6
モデル推定	5.4
LOF	**65.2**
iforest	12.0

また，必要としたメモリ量はLOFを用いた手法が最も優れており，27.9GBを必要とする結果であった．またPCA，Isolation Forestを用いた手法についても必要であったメモリ量はほぼ変わらず，30GB以内に収まっている．一方，最も多くなった手法はモデル推定を用いた手法で，その他の手法の3割程度の入力からの検出であるにも関わらず60.7GBを必要とした．ビデオメモリの使用量ではRNNを用いた手法が0.595GBの消費であるなど値の上では優れているが，ビデオメモリと通常のメモリを同列に比較するのは不適切と考えるためあくまで参考値とする．

4.3.3 RQ3: CPSへの適用に適した異常検出手法はどのような手法か

(1) 概要

RQ3では前章までに論じた結果をもとに，比較した手法の中でどのような手法がCPSのシステムログを用いた異常検出に適切かを議論する．

(2) 結果

前章までの結果より，分類器としての精度はF1値，AUC，タイムコストの点で優れているPCAを用いた手法が適用に適していると結論付ける．F1値の高さからも読み取れるがPrecisionとRecallのバランスは随一である．他の手法と比較した際のPCAを用いた手法の精度の高さは図4.2aからも直感的に理解することができる．

一方，RNNを用いた手法ではタイムコストの点に課題が残るがPrecision及びFPrを向上させることが可能であった．

また，異常分析においては検出漏れを極力少なくしたいケースがあると考えられる．その際，PCAを用いた手法やRNNを用いた手法のRecallの低さが問題となることが考えられる．Recallの低さを問題視する場合はFPrを犠牲にして閾値を下げることでも緩和はできるが，評価対象手法の中ではLOFを用いた手法のRecallが最高となっており，Recallの観点からはLOFを用いた手法が優れていると考えることもできる．

上記の3手法については既存手法であるモデル推定を用いた手法よりもF1値の点で全て優れており，特にPCAを用いた手法，RNNを用いた手法ではAUCの点でも勝っているなど，我々の提案手法の有用性を示せた結果となった．

一方，Isolation Forest を用いた手法については定量評価，定性分析の結果ともに性能が低いという結果となった．

5. 妥当性の検証

5.1 構成概念妥当性

構成概念妥当性では実験における評価手法の妥当性について議論する．

精度評価指標

本実験では検出手法の精度の評価指標として Precision，Recall，False Positive rate，F1 値，AUC を用いた．これらの値は機械学習における性能評価指標として一般的に利用されており，本実験における手法の評価指標として妥当であると考えられる．

計算コスト評価指標

本実験では検出手法の計算コストの評価指標として計算時間，メモリ使用量，ビデオメモリ使用量を用いた．これらについて，計算時間は一般的に計算コストを評価する際に用いられている指標であり，計算環境が同一であるという条件を満たしている本実験で用いることは妥当であると考えられる．メモリ使用量，及びビデオメモリ使用量については各々を混同せずに評価するに限っては十分に妥当であると考えられる．

5.2 外的妥当性

外的妥当性では実験結果の一般性についての妥当性について議論する．

本実験では適用対象となる CPS として あくあたん のみを用いている．あくあたん はその性質上十分に複雑かつシステムモデルが十分に定義されておらず，また頻繁に仕様が変更されており，一般的な CPS と比べても異常の検出の難易度は高いと考えられる．そのため，本実験で検出されたものと同質の異常であれば一般的な CPS からも検出できる可能性は高い．ただし，一般的な CPS に適用した結果が必ずしも本実験と一致するとは限らない点は妥当性への脅威であり，十分に留意が必要である．

5.3 内的妥当性

内的妥当性では本実験の方法の妥当性について議論する．

評価データ作成手法

本研究では教師なしの分類器を扱っている関係上，完全な正解データというものが存在しない．そのため，定量評価のためには対象となるシステムの作成者や共同研究者と共に正解データを作成することで評価データを定義した．そのため，評価データにはデータ作成者の主観による異常判定が行われており，内的妥当性への脅威となりえると考えられる．

また，対象システムのシステムログが膨大であるなどの都合上評価データを作成するのが困難であるため，評価データの数が限られてしまう点についても妥当性への脅威となりえる．実際に本実験においても定性評価結果と定量評価結果が食い違う結果となっており，この点に関しては評価データの数を増すことにより一層信頼性の高い実験結果が得られると考えられる．

実験の実装の正確さについて

本実験で利用したプログラムは重要な部分についてはテスト駆動で開発を行うなどの不具合低減措置は講じたが，不具合が残っている可能性は否めない．そのため内在するバグが内的妥当性への脅威となりえる．

6. 結言

本研究ではCPSの異常を教師無しで検出する手法を提案し，検出アルゴリズムとして採用したアルゴリズムそれぞれによる検出精度や検出可能な異常について分析を行った．評価実験の結果では，実験対象システムでの結果が一般化可能であるかについては議論の余地が残るものの，提案した4手法，既存の手法である1手法ともにCPSの異常を検出することが可能であった．

各手法の比較では，手法の性能はPCAを用いた手法が最も高評価であるという結論となった．一方，一部の精度指標ではRNNを用いた手法及びLOFを用いた手法も優れている．また，定性評価を行った結果についてもLOFを用いた手法が直感的な異常を検出している数が多いなど，優秀な性能であった．一方でIsolation Forestを用いた手法は精度に関しては他の手法に及ばない結果となった．

また，既存手法であるモデル推定を用いた手法も十分に異常を検出することはできているが，我々の提案した上記3手法，特にPCAを利用した手法とRNNを利用した手法のほうが高性能な手法と比較すると精度，必要とする計算コスト共に劣っているという結果となった．

以上より，我々の提案した手法，特にPCAを用いた手法，RNNを用いた手法にはCPSの異常検出に十分な性能があると結論付ける．

謝辞

本研究を行うにあたり，研究課題の設定や研究に対する姿勢や本報告書の作成，データの提供に至るまで，全ての面で丁寧なご指導を頂きました，本学情報工学・人間科学系 水野修 教授に厚く御礼申し上げます．また，研究方針についての議論や論文執筆手法について丁寧なご指導を頂くのみならずRNN手法のベースラインを設計し，実装の指針についても指導して頂きました国立研究開発法人産業技術総合研究所 情報技術研究部門 山形賴之 主任研究員をはじめとする産業技術総合研究所の皆様にも厚く御礼申し上げます．最後に，本報告書執筆にあたり貴重な助言を多数頂きましたソフトウェア工学研究室の植村佳治君，北村紗也加さん，黒田翔太君，小林勇揮君，近藤将成君，田中健太郎君，中川要さん，中村勝一君，西浦生成君，廣瀬早都希さん，洪浚通君，渡辺大輝君をはじめとする本学情報工学専攻，情報工学課程の皆様，ポスター発表やスライド発表のデザインについて助言を頂きました応用生物課程 國領正真君，本研究の実験対象となるのみならず研究活動の進捗や進路についての助言を頂きましたソフトウェア工学研究室のあくあたん，及び学生生活を通じて著者の支えとなった家族や友人に深く感謝致します．

参考文献

[1] D. Yadron and D. Tynan, “Tesla driver dies in first fatal crash while using autopilot mode,” *The Guardian*, July 1, 2016.

[2] Y. Harada, Y. Yamagata, O. Mizuno, and E.-H. Choi, “A log-based anomaly detection of CPS using a statistical method,” Proceedings of the 8th IEEE International Workshop on Empirical Software Engineering in Practice (IWESEP2017), pp.1–6, IEEE CPS, March 2017. Tokyo, Japan.

[3] M.M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander, “LOF: Identifying Density-Based Local Outliers,” Proceedings of SIGMOD, pp.1–12, 2000.

[4] F.T. Liu, K.M. Ting, and Z. Zhou, “Isolation forest,” Proceedings of ICDM, pp.413–422, IEEE, 2008.

[5] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, “Long Short Term Memory Networks for Anomaly Detection in Time Series,” 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), Bruges, Belgium, 22-24 April 2015., pp.89–94, 2015.

[6] S. Schneider, I. Beschastnikh, S. Chernyak, M.D. Ernst, and Y. Brun, “Synoptic: Summarizing system logs with refinement,” Proceedings of the Workshop on Managing Systems via Log Analysis and Machine Learning Techniques (SLAML), pp.1–10, Vancouver, Canada, Oct. 2010.

[7] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly Detection : A Survey,” ACM computing surveys, pp.1–72, 2009.

[8] C. Aggarwal, Outlier Analysis, Springer Publishing Company, 2015.

[9] V. Verma, G. Gordon, R. Simmons, and S. Thrun, “Real-time fault diagnosis,” IEEE Robotics and Automation Magazine, vol.11, no.2, pp.56–66, 2004.

[10] S. Narasimhan and G. Biswas, “Model-Based Diagnosis of Hybrid Systems,” IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol.37, no.3, pp.348–361, 2007.

[11] M.W. Hofbaur and B.C. Williams, “Mode estimation of probabilistic hybrid systems,” Lecture Notes in Computer Science, vol.2289, pp.253–266, 2002.

[12] F. Zhao, X. Koutsoukos, H. Haussecker, J. Reich, and P. Cheung, “Monitoring and Fault Diagnosis of Hybrid Systems,” IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol.35, no.6, pp.1225–1240, 2005.

[13] M.W. Hofbaur and B.C. Williams, “Hybrid estimation of complex systems.,” IEEE transactions on systems, man, and cybernetics, Part B: Cybernetics, vol.34, no.5, pp.2178–2191, 2004.

[14] C. Aggarwal, “High-Dimensional Outlier Detection: The Subspace Method,” Outlier Analysis, pp.135–167, Springer, 2013.

[15] O. Mizuno, “Auto-feeding and moving aquarium management droid — everybody’s Raspberry Pi contest, the grand prix work,” Nikkei Linux (in Japanese), vol.17, no.4, pp.77–83, April 2015.

[16] F.T. Liu, K.M. Ting, and Z.H. Zhou, “Isolation forest,” 2008 Eighth IEEE International Conference on Data Mining, pp.413–422, Dec. 2008.

[17] Python Software Foundation, Python.org, （オンライン）, 入手先〈https://www.python.org/〉（参照 2018-1-30）.

[18] The pandas project, pandas: Python Data Analysis Library, （オンライン）, 入手先〈https://pandas.pydata.org/〉（参照 2018-1-30）.

[19] The ELKI Team, ELKI: Environment for Developing KDD-Applications Supported by Index-Structures, （オンライン）, 入手先〈https://elki-project.github.io/〉（参照 2018-1-30）.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol.12, pp.2825–2830, 2011.

[21] Preferred Networks, inc. and Preferred Infrastructure, inc., Chainer: A flexible framework for neural networks, （オンライン）, 入手先〈https://chainer.org/〉（参照

2018-1-30).