Vol.2015-SE-188 No.4
Vol.2015-EMB-37 No.4
2015/6/4

ベイジアンネットワークとクラスタリング手法を用いたシステム障害検知システムの有効性検証

爲岡 啓[1,a)]　植田 良一[2,b)]　松下 誠[1,c)]　井上 克郎[1,d)]

概要：Web サーバ群に起きうる障害を未然に発見することは，システムの可用性向上の観点などから重要である．これまで，Web システムから収集したメトリクス群に対して解析技術を用いて処理することによって，管理者の知識や経験に依存せず障害検知を行う手法が提案されている．しかし，そのような手法は多くのパラメータを適切に設定する必要があるため，障害検知を高い精度で行うのは困難であった．本研究では，ベイジアンネットワークとクラスタリングという，機械学習を用いた 2 つの解析技術を組み合わせて，メトリクス群を効率的に処理することにより，障害検知を行う手法を考案する．その手法を適用し，効率良く障害検知が行えていることを，評価実験によって示す．

1. まえがき

社会基盤としての Web システムは，不特定多数のクライアントを抱える重要な役割を果たしており，安定した長期稼働が要求される．しかし，今後システムの複雑化，クライアントの増加によって，システムに障害が起こる可能性は高まっていくことが予想される．たとえば，障害の原因として，Web サーバへの過負荷や，セキュリティの脆弱性を突いた攻撃などが挙げられる [1]．このような障害は時間，場面を問わず起こりうるものであるから，その障害をいかに早く発見し，対処するかということが，Web システムの長期安定稼働に必要だと考える [2]．

現状の Web システムにおいては，システム管理者がサーバマシンのメモリや CPU など少数の計測メトリクスの推移を確認して，一定時間内にシステムの不具合が発生するかどうかを判断する [3]．システム管理者の扱うデータ量は，システム規模の増大に伴い今後増加の一途を辿ると考えられる．不具合が発生するとき，個々のメトリクスが非常に複雑に連携している．したがって，膨大なデータログを人間が処理し，不具合が起こるかどうかという結論を導き出すことは困難であり，それは人間の処理限界を超えている [4]．

また，人間が限られた時間内に確認できる一部のデータに基づいた判断は推測の域を出ず，経験者の缶に頼るところが大きい．さらに，手動で行うことは時間と労力を要し，かつ常に可能であるとは限らない [5]．

近年，収集した大量のデータを，機械学習等の統計処理を施すことで，複雑に絡み合ったメトリクス間の関連をモデル化し，システムの異常判定，ひいては，異常の根本原因究明に活用する手法が提案されている．しかし，そのような手法の多くは，膨大な学習データを必要とするか，多くのパラメータを適切に設定する必要があり，高精度，高効率で障害検知を行うことは困難である．

そこで本研究では，ベイジアンネットワークとクラスタリングという，機械学習を用いた 2 つの解析技術を組み合わせて，膨大なデータ量を削減しつつ，メトリクス群を効率的に処理することにより，障害検知を行う手法を考案した．その手法を適用し，効率良く障害検知が行えていることを示すために検証実験を行った．その結果，より少量の学習データを利用して，全データを学習した場合と同等の効果が得られることを確認した．

以降 2 章では，本研究の背景と，機械学習を用いた既存のデータ解析技術，また，その問題点について説明する．3 章では，我々の提案手法について説明する．4 章では，我々の提案手法が有効であることを示す検証実験について説明する．5 章では，検証実験の結果についての考察を行う．6

[1] 大阪大学 大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, 1-5 Yamada-Oka, Suita, Osaka, 565-0871, Japan
[2] (株) 日立製作所 研究開発グループ 情報通信イノベーションセンタ
Hitachi, Ltd. Research & Development Group Center for Technology Innovation, Information and Telecommunications, 292 Yoshida-chou, Totsuka, Yokohama, Kanagawa, 244-0817, Japan
a) t-akira@ist.osaka-u.ac.jp
b) ryoichi.ueda.mb@hitachi.com
c) matusita@ist.osaka-u.ac.jp
d) inoue@ist.osaka-u.ac.jp

章では関連研究について述べ，7 章ではまとめと今後の課題について述べる．

2. 研究の背景

これまでに提案されている，機械学習による障害検知に用いられる技術として，ベイジアンネットワークとクラスタリングという，2 つのデータ解析技術を紹介する．

2.1 ベイジアンネットワーク

ベイジアンネットワーク (以下 BN) は，個々の事象の因果関係を条件付き確率で表すモデルで，観測対象の過去の状態を学習し，観測対象がある状態 S にある時の注目事象 E(例：応答時間が 3 秒以上となる) の発生確率 PE を算出する事ができる [6]．

ただし，学習データに含まれる過去の状態と同じとみなされる既知の状態での注目事象の発生確率は計算できるが，学習データに含まれない未知の状態下での注目事象の発生確率は正しく計算できない．注目事象の発生確率を正しく計算するためには学習データの量を増やすことが効果的である [3][7]．しかし，データ量が増えると学習処理にかかる時間が増大し，実用的な時間で完了しなくなるという課題がある．学習データに含まれる観測項目数を一定にした場合，計測回数に応じて学習時間が長くなる [8]．

2.2 クラスタリング

クラスタリング (以下 CL) は，観測対象の過去の状態をグルーピングし，いくつかのクラスタに分割する方法のひとつである [9]．教師データが存在しない場合は，個々のクラスタに対する意味付け，例えば正常時のクラスタかどうかという判定は，当該クラスタに含まれる観測値から人間が行う．

入力 (学習) データとして正常時のデータのみ与え，最近傍クラスタからの距離が閾値を越える観測値を異常と判定する診断方法が知られているが，正常範囲の一部のみが学習データに含まれる場合，正常であるにもかかわらず異常と判定してしまう可能性がある．このため，異常データを含まず，かつ正常範囲内でなるべく広い範囲を含む入力データが必要となる．この要件が十分に満たされないと，検知漏れや誤報を引き起こす可能性が高くなる [10]．

2.3 既存手法の問題点

BN，CL を含め，機械学習を活用する方法の多くは，期待する効果を得るために適切に学習データを選定しなければならない，という共通の課題がある．学習データは，効果があり，かつそのデータ量が少なければ少ないほど効率的である．現在は，学習パラメータの調整と合わせて，学習データの選定を人間が試行錯誤で行う事で，この課題に対処している [7][11][12][13][14]．

3. 提案手法

3.1 提案手法の発想

Web システムの運用中のほとんどの時間は安定した正常状態の範囲内にとどまり，ごく稀にこの正常状態からの逸脱が発生することが知られている．BN を用いた障害検知では，既知の状況下での注目事象の発生確率を計算する．したがって，このとき必要となる学習データとして必要なのは，この逸脱区間のものである．すなわち，Web システム運用中において，BN が組み入れるべき学習データ部分は殆どない．

この事実に着目し，我々は，より少量でありながら，多種の状態を含む学習データを選定する方法を考案する．方法を考案するにあたり，BN の学習データに同じ状態が繰り返し出現する回数を減らしつつ，異なる状態を多数含めることができれば，より少ないデータ量で学習した場合であっても，全データで学習した場合と同等の学習効果を得る事ができると考えた．

また，現在の学習データに含まれていない状態を検知するには，BN と同じ学習データを CL に入力データとして与え，現在の計測値と最近傍クラスタとの距離を計算することで実現できると考えた．

以上により，BN と CL を効果的に組み合わせることによって，膨大なデータの中から自動的に学習データを選定することができるのではないかと予想した．

3.2 提案手法の手順

提案する手法の手順は以下のとおりである (図 1)．

手順 1
CL に正常時データとして初期データを与えてモデルを生成する．

手順 2
Web システムから学習用データを取得する．

手順 3
学習用データをいくつかの区間として分割した上で，全学習データをモデルに入力し，CL の診断結果をもとに，閾値を超える適切な連続区間を選定する．

手順 4
選定した連続区間の学習データを利用して BN モデルを生成し，障害に対してその発生確率を算出する．

手順 3 における選定によって削減されるデータは，正常時の状態であり，正常時には同じ状態が繰り返し出現していると考えられるため，BN モデルに及ぼす影響は少ないと考える．よって，手順 4 における出力結果は，全学習データを利用した場合との強い相関が見られると考える．

Vol.2015-SE-188 No.4
Vol.2015-EMB-37 No.4
2015/6/4

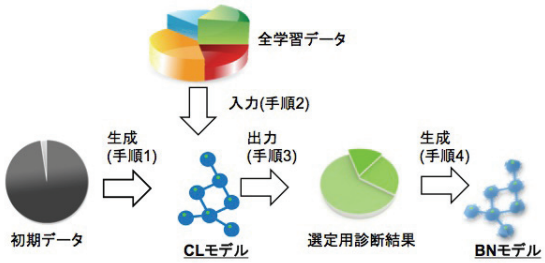

図 1　提案手法の手順

3.3　検証方法

我々の提案手法が期待した結果を導けるか確認するために，検証実験を行う．その手順は以下のとおりである (図 2)．

手順 1

Web システムから取得した学習用データを，いくつかの区間に分割する．

手順 2

それらの区間を基にして，連続区間のパターンを作成する．このパターンの中には，我々が提案手法において，CL で選定した学習データも含まれる．例えば，3 つの区間に対しては区間 1，2，3 それぞれのみと，区間 1-2，区間 2-3，区間 1-3(全学習データ) の 6 パターンがある．我々の提案手法において，CL が選定した区間が区間 1-2 であったとき，それはこの 6 パターンに含まれる．

手順 3

全てのパターンについて，学習データとして BN モデルを生成し，診断データに対して障害発生確率を算出する．

手順 4

それらの障害発生確率の精度について優劣を決定し，学習データの順位付けを行う．優劣の決定方法については，次節にて説明する．

一般的に，学習データの量は多ければ多いほど，注目事象の発生確率は正しく計算されるため，より多くの区間数を学習データとして組み入れたパターンが高順位となるはずである．我々が提案手法を用いて選定した学習データのみを用いて学習した場合の結果が高順位であるならば，より少ないデータ量であっても全データで学習した場合と同等の学習効果を得られる，と考えられる．

3.4　問題の設定

3.4.1　障害の定義

大規模システムにおける障害のひとつとして，その性能

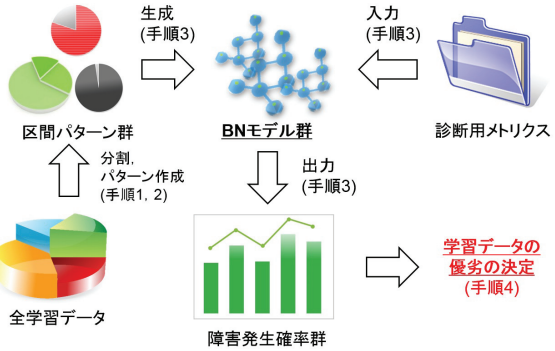

図 2　検証手順

劣化が挙げられることが多く，クライアントのリクエストに対する Web サーバの応答時間は，性能劣化と強い因果関係を持つ．ゆえに我々は，BN が出力する障害発生確率の時間変化と，その時間変化する区間における平均応答時間との相関係数を計測することによって，学習データについての優劣の比較を行うこととした．

3.4.2　相関係数

平均応答時間と確率の時間変化との相関係数 COR($0 \leq COR \leq 1$) は以下の式で与えられる．

$$COR = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

この時，相関係数 COR の値が大きければ大きいほど，その学習データは優れていると判断する．

3.4.3　CL モデルの閾値設定

予備実験としていくつかの初期データを用い実験を行った結果，CL モデルの算出距離が 500 を超えるとき，顕著に学習データとの差異を検出している，すなわち，異常を検出していることがわかった．そこで，本研究では，CL モデルの算出距離 500 を閾値とし，この閾値を超えた区間のみを，BN の学習データとして用いることとする．

4.　検証実験

ある一定の区間における，正常，異常を含めたあらゆるパターンの学習データについて，CL を利用して選定した学習データが，他のパターンに対して優れているということを検証する．そのための実験環境，方法について説明する．

4.1　実験環境

実装環境は，クライアント，ロードバランサ，Web サーバ，データベースの 4 つのコンポーネントで構成する．Web サーバは，一般的な Web システムには複数存在することを鑑み 2 台，その他のコンポーネントは，それぞれ 1

台の仮想計算機に対して実装する．以下に各システムの役割について説明する．

4.1.1 クライアント

Web サーバにリクエストを送信し，その結果を受信する．クライアント用のアプリケーションとして，Apache JMeter[16] を用いる．Apache JMeter は，クライアント-サーバシステムのパフォーマンス測定および負荷テストを行う Java アプリケーションである．Web ページを指定して，アクセスクライアント数，間隔，ループ回数等を設定することによって，クライアントからサーバへのアクセスを擬似的につくりだすことができる．

4.1.2 ロードバランサ

クライアントから送られたリクエストを Web サーバへと中継する．複数の Web サーバの状態をモニタリングして，リクエストによる負荷を分散させることができる．ロードバランサの実装ツールとしては，Apache の mod_proxy_balancer[17] を用いる．mod_proxy_balancer は，Apache が提供する負荷分散に利用するモジュールであり，クライアントからのアクセスを複数のサーバに分散することができる．本研究では，2 台の Web サーバに対して，均等に負荷分散を行っている．

4.1.3 Web サーバ

ロードバランサから受け取ったクライアントからの要求を，データベースを用いて処理し，その結果を返す．サーブレットコンテナとしては Apache Tomcat[18]，サーバアプリケーションとしては iBatis JPetStore[19] を利用する．Apache Tomcat とは，Java Servlet や JavaServer Pages を実行するためのサーブレットコンテナであり，これを利用することによって，サーバ上で HTML などの Web ページを動的に生成する Java サーブレットを動作させることができる．また，JPetStore とは，Java サーブレットを利用してサーバ上で動作するアプリケーションである．データベースの情報を基に架空のペットストアを Web ページとして表示することができる．

4.1.4 データベース

Web サーバから受け取った要求を基にデータの検索，抽出を行い，その結果を Web サーバへと返す．データベースシステムとしては，MySQL を利用する．MySQL とは，Oracle[20] が開発するリレーショナルデータベースを管理，運用するためのアプリケーションである．本研究では JPetStore のデータを登録し，Web サーバからアクセスすることによって，データベースサーバを実装する．

各アプリケーション，ツールをシステムに反映した図 3 を以下に示す．

本実験では 4 台の仮想計算機を利用した．そのうち 3 台については，CPU 1 コア，メモリ 1GB，HDD 20GB，1Gbps ネットワークとし，Web サーバ，ロードバランサとした．データベースよう計算機については，CPU 2 コア，

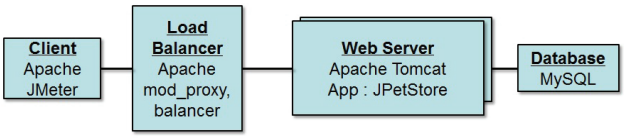

図 3 Web システム全体図

メモリ 4GB，HDD 110GB，1Gbps ネットワークとした．OS は全計算機において，Linux(CentOS 6.4) を採用した．

各仮想計算機には CPU/メモリ/ネットワーク/Disk の利用状況データを 10 秒間隔で収集するための監視エージェント (collectd[21]) が組み込まれており，観測データを取得することができる．クライアントにおいては，あらかじめ与えたシナリオ (例：データベース上の商品を検索→ 1 商品をカートに入れる→チェックアウト) と負荷パターンに応じて，ユーザの挙動を模擬するリクエストを発生させる．

収集データの一覧を表 1 に示す．本実験では，ロードバランサで表 1 における 1 から 3 までと 5 の 4 項目，2 台の Web サーバそれぞれで 1 から 4 までの合計 8 項目，データベースサーバで 1(2 コア分) と，2, 3, 4, の合計 5 項目，計 17 項目について，システム監視の際よく利用されるメトリクスを収集した．

表 1 監視対象メトリクス

	リソース名	監視対象
1	CPU(データベースのみ 2 つ)	利用率 (%)
2	メモリ	利用量 (bytes)
3	ネットワーク	送受信量 (bytes/sec)
4	Disk(ロードバランサ以外)	I/O オペレーション数 (ops/sec)
5	Web Access (ロードバランサのみ)	リクエスト数, 最大，平均応答時間

4.2 実験方法

Web サーバ上で稼働する JPetStore へ，クライアントからの擬似的なバックグラウンドトラフィックを与えながら，システムの異常状態を模擬する負荷を，Web サーバに与える．Apache JMeter を用いて，クライアントからサーバへ継続的にリクエストを発生させた状態で，システムの異常状態を模擬するために stress コマンド [22] にて負荷を与える．

学習データ，診断データいずれも，以下の図 4，表 2 のようなパターンの負荷を与えたものを収集する．また，学習区間 1-7 を，5 分毎の区切りとして，図 4 のように定める．

1 回の実験は，35 分単位で行う．

図 4 のような負荷に対して収集した学習用，診断用のデータに対して，実験用に作成したツールを適用する．学習データを入力すると，BN，CL それぞれのモデルを生成し，そのモデルに診断用データを入力することによって，

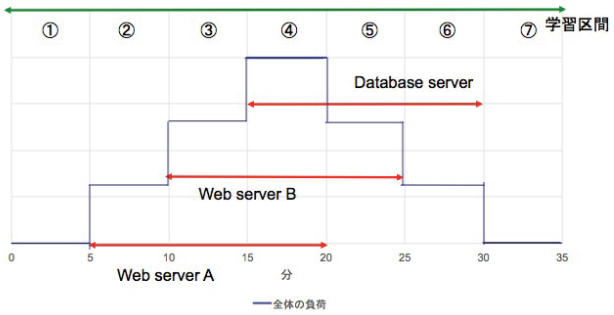

図 4 負荷パターン

表 2 実験プロセス

時間	要件
0：00-5：00	負荷注入なし
5：00-10：00	Web サーバ A に負荷注入
10：00-15：00	Web サーバ A，B に負荷注入
15：00-20：00	Web サーバ A，B，DB に負荷注入
20：00-25：00	Web サーバ B，DB に負荷注入
25：00-30：00	DB に負荷注入
30：00-35：00	負荷注入なし

障害の発生する確率や，最近傍クラスタからの距離を算出することができる．ツールによるモデル生成プロセスを図 5 に示す．

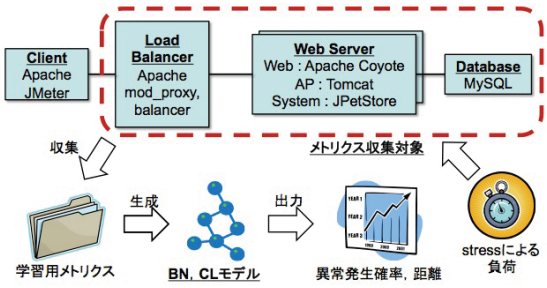

図 5 モデル生成プロセス

モデル生成プロセス (図 5) について説明する．監査対象システムに対して，クライアントからのバックグラウンドトラフィックを継続した状態で，stress による負荷を与える．それによって収集した学習用メトリクスを基に，BN，CL のモデルを生成する．生成した CL モデルに学習用データを入力することによって，データの選定を行うことができる．

また，学習データを基に生成した BN，CL モデルに，診断用メトリクスを入力することによって，診断区間における異常発生確率，距離を算出することが出来る．

ツールの実装に際して，BN 用の統計解析環境 R 用ライブラリ bnlearn[8]，CL 用の R 標準ライブラリの k-means[23] を用いた．

BN については，予備実験の結果，最大応答時間が 3 秒を超えた場合に，システムに障害が起こっていると判断することとした．従って，観測データに対し，最大応答時間が 3 秒を超える確率を逐次診断し出力する．

また，学習用データには 7 つの区間があるため，それらの連続区間のパターン総数は 28 である．これら全てのパターンに対して BN での障害発生確率を算出し，平均応答時間との相関係数を取る．

CL については，初期データとして，クライアントからのリクエストのみがトラフィックとして存在する区間 1 だけを学習区間として組み入れることで，正常時と異常時との最近傍クラスタとの距離の差を顕著にする．

4.3 実験結果

まず，図 4 のように負荷をかけた時の，診断用データにおけるリクエスト数，平均，最大応答時間の時間変化を図 6 に示す．グラフの○点が 10 秒間のリクエストの最大応答時間 (左軸：sec)，△点が同 10 秒間の平均応答時間 (左軸：sec)，＋点が同 10 秒間の処理リクエスト数 (右軸：request/sec)，横軸は経過時間 (秒) を表す．グラフタイトルの MA=5 はグラフ上の点が隣接 5 データの移動平均値である事を表す．システム全体に最大負荷のかかる区間 4 を中心として，区間 2-6 に最大応答時間の大きな増加が見られる．

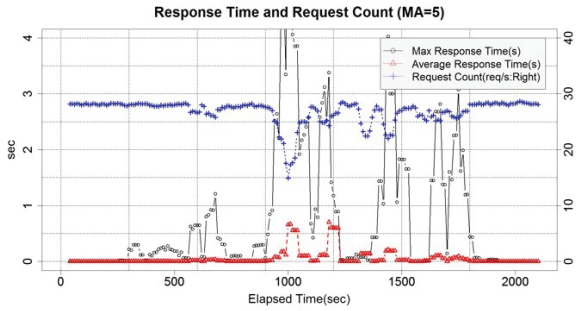

図 6 リクエスト，応答時間の時間変化

以下，提案手法について，手順に沿って実験結果の例を示す．

提案手法における手順 1-3 について，学習区間を区間 1 のみとした時の BN，CL における学習データの診断結果を図 7 に示す．グラフの○点が BN による最大応答時間が 3 秒を超える確率 P(左軸)，△点が CL による最近傍クラスタからの距離 D(右軸)，横軸は経過時間 (sec) を表す．正

常時のみを学習データとして組み入れているため，BN モデルは確率として 0 を継続出力している．図 7 において，区間 3-6 は CL モデルの閾値 500 を超えているため，この区間を CL が選定した区間とする．

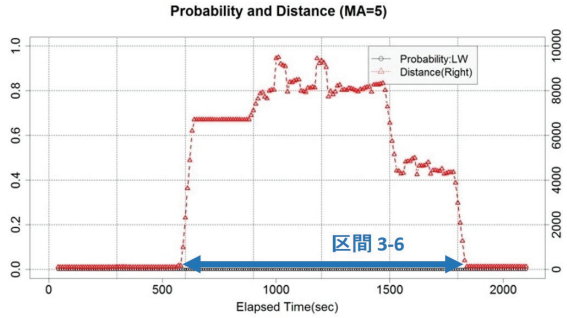

図 7 BN，CL による診断結果 (区間 1-1)

次に，提案手法における手順 4 について，区間 3-6 を学習データとして利用して BN モデルを生成し，診断データを入力した結果を図 8 に示す．各軸，値の見方については図 7 と同様である．

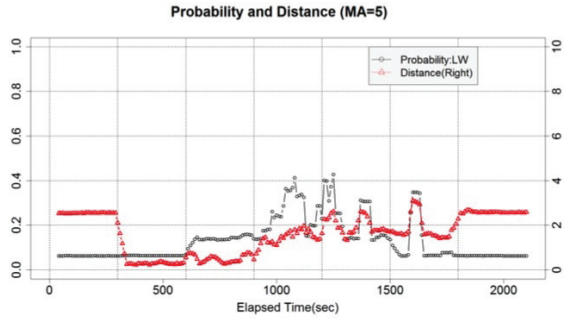

図 8 BN，CL による診断結果 (区間 3-6)

異常時を学習データとして組み入れたため，BN モデルの出力確率は図 6 における応答時間の変化に対応して変化している．BN モデルの出力確率と平均応答時間との相関係数を計測したところ，COR=0.88 となった．

5. 考察

5.1 相関係数に基づく優劣付け

前節の手順と同じく，連続区間の 28 パターン全てに対してそれぞれの学習モデルに基づいて，BN の算出確率と平均応答時間との相関係数を計測した．それらを値の高い順に並べた表 3 を示す．

一般的には，学習データの量は多ければ多いほど，注目事象の発生確率は正しく計算されるため，区間数 4 の学習データの順位は，より区間数の多い学習データの数が 6 つであることから，7 位以下となるはずである．しかし，提案手法を用いた学習データは 6 位となったため，少ないデータ量で，学習データとしてより高い効果を発揮すると言える．

表 3 相関係数による評価結果

順位	区間	相関係数	区間数
1	2-7	0.907	6
2	1-7	0.893	7
3	1-6	0.892	6
4	2-6	0.89	5
5	3-7	0.888	5
6	**3-6**	**0.88**	**4**
7	4-6	0.848	3
8	5-5	0.826	1
9	4-5	0.805	2
10	3-5	0.794	3
…	…	…	…
27	6-7	0.138	2
28	1-1	0	1
平均	-	0.684	3

5.2 全学習データとの診断比較

区間 3-6 を用いた時の BN の出力確率と，区間 1-7 全てを学習データとした時の BN の出力確率の相関係数を取り，同等の効果をあげているかを検証する．

区間 1-7 全てを学習データとした，BN，CL の診断結果を図 9 に示す．各軸，値の見方については図 7，図 8 と同様である．

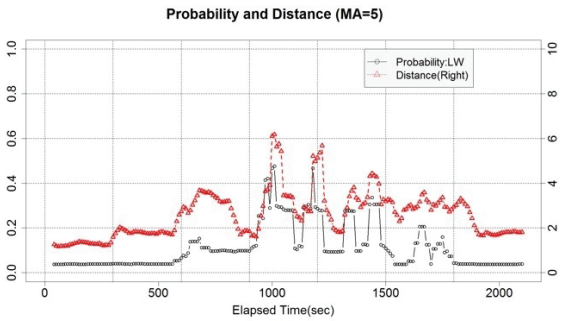

図 9 BN，CL による診断結果 (区間 1-7)

BN によるそれぞれの算出確率の時間変化について，相関係数を算出したところ，COR=0.993 と，高い相関値を見せた．区間数が減っても同等の結果が得られたということで，提案手法に対して検証することができた．

6. 関連研究

Edward Stehle らは [11] にて Convex-hull(凸包 [24]) を活用したソフトウェア障害診断，原因推定技術を提案した．

この方法では，事前に単独で発生する障害時のデータを学習させる必要があるため，未知の障害への対応は困難であると考えられる．また，同氏らは [12] にて，Convex-hull を活用したソフトウェア障害検知技術を発表した．学部の Web サイトへの 9 週間分のアクセスログから 3 週間分をランダム抽出し，1 週間分を学習用に，別の 1 週間分をモデル精度の確認用に，残りの 1 週間分をバックグラウンド負荷生成用に活用，あらかじめ障害を埋め込んだ http サーバに対して，リクエストを送信する実験で他のエンクロージャ形成法 (Chi-Square/Mahalanobis/ Hyper-rectangle) と比較し，誤検知率が高いものの，提案手法が唯一全 8 種の障害を検知できる事を確認した．埋め込んだ障害はバグコード (無限ループ，無限再帰，メモリリーク)，大量ログ出力による Disk 資源枯渇，脆弱性への攻撃 (Spambot であるトロイの木馬，DoS 攻撃) である．この方法は学習データの削減，選定を行っていない．

Ira Cohen らは [13] にて，我々の提案手法と同類の Tree-Augmented Bayesian Network[32] を活用して，様々なメトリクス観測結果とシステムの状態を関連付ける技術を発表した．Java PetStore に人工的なパターン (正弦波形) のバックグラウンド負荷 (1000req/min) を与えつつ，ユーザの挙動を模擬するリクエストを処理させる実験にて評価を行った結果，注目する事象に関係の深いメトリクスを選択できること，管理者の経験に依らず，各メトリクスの適切な閾値を算出できること，注目事象の発生を予測できること，等を確認した．この方法では学習データの削減，選定は行っていない．

Steve Zhang らは [29] にて，[13] の手法を拡張して，直近の区画の観測データから生成した新モデルが，当該区画の診断において，現行モデル集合内のどのモデルよりも良い診断結果を出した場合，この新モデルをモデル集合に追加する診断方法を提案した．この手法では複数のモデルの集合を管理するが，我々の手法では学習データの集合を管理する点で異なっている．

Satoshi Iwata らは [25] にて，性能劣化の根本原因を推定する方法を発表した．処理時間でクラスタリングを行い，原因の判明している既知のインシデントと同じクラスタに属する新規インシデントは同じ原因で発生していると推定する．ネットオークションアプリケーション RUBiS と，Web ページ間遷移を状態遷移確率表で指定してユーザの挙動を模擬する RUBiS client emulator を活用して人工的なパターンでリクエストを生成して実験を行った．27 種のリクエストに対する，各サーバでの処理時間の平均，中央，最大，最小をメトリクスとして活用して評価を行った．

Thanh H. D. Nguyen らは [14] にて，システムの性能劣化原因の自動特定技術を発表した．この技術はまず，事前の性能テストで性能計測データと性能劣化原因のペアを複数収集し，これを教師データとする．次に，発生中の事象が教師データのどれと類似しているかを機械学習アルゴリズムにて同定することで事象の原因を推定する．学習データ量削減のために Control Charts を利用する．Weka で利用可能な 17 種の機械学習アルゴリズムにて，それぞれの 6 種の性能劣化原因の究明率を比較した結果，1 原因あたり 4 つの学習データを準備することで，74-80%という高い精度で原因が究明できる事を確認した．これらの研究は，原因究明を目的としている点で我々と異なる．

7. まとめと今後の課題

計算機システムの CPU，メモリ，Disk，ネットワーク等の利用状況の観測値を基に，機械学習を活用してシステムの状態，特に性能異常の検知に寄与する，学習データの選定手法を考案し，検証実験によってその有効性を示した．オンライン販売システムを使い，提案手法で選定した学習データによる診断結果と，全データによる診断結果との相関を計測した結果，より少量のデータで全データ学習と同等の学習効果が得られることを確認した．

今後の課題として，まず，実験パターンを増やすことが挙げられる．本実験では学習用データ，診断用データいずれも，システムに対して意図的にトラフィックを発生させ，負荷を与えることによって異常をつくりだしたが，実際の障害に対して，我々の提案手法が適用できるかどうか，検証する必要があると考える．また，CL での正常，異常を判定する際の閾値の決定方法を考えること，また，時間変化に対する自動化手法が必要である．現状として，CL の閾値の決定や選定後のモデル生成，診断は人間が行なっているため，それらを状況や時間変化に対応して自動設定，自動診断が行えるような手法が必要である．

謝辞 図 1, 図 2 中で利用した画像の一部は, Vector Fresh を著作者とする図 [26] を元に著者が修正を加えたものである．本研究は，日本学術振興会科研費基盤研究 (S)(課題番号 25220003) の助成を受けたものである．

参考文献

[1] 谷 誠之, “年々難しくなる「障害検知」のコツ”, http://www.itmedia.co.jp/im/articles/1005/19/news106.html.
[2] 加藤 清志, “サービスの安定稼働を阻むサイレント障害”, http://thinkit.co.jp/article/1089/1.
[3] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, J.S. Chase, “Correlating instrumentation data to system state: A building block for automated diagnosis and control”, USENIX Association OSDI’04: 6th Symposium on Operating Systems Design and Implementation, 2004.
[4] A. Fox and D. Patterson, “Self-repairing computers”, Scientific American, 2003.
[5] G. A. Alvarez, E. Borowsky, et al. “An automated resource provisioning tool for large-scale storage systems”, ACM Transactions on Computer Systems (TOCS), pp483-518, 2001.

[6] J. Pearl, “Bayesian Networks, a Model of Self-Activated Memory for Evidential Reasoning”, Proceedings, Cognitive Science Society pp329-334, 1985.
[7] S. Zhang, I. Cohen, M. Goldszmidt, J. Symons, A. Fox, “Ensembles of Models for Automated Diagnosis of System Performance Problems”, The International Conference on Dependable Systems and Networks, Yokohama, Japan, 2005.
[8] bnlearn - an R package for Bayesian network learning and inference, http://www.bnlearn.com/.
[9] Y. Okada, T. Sahara, S. Ohgiya, T. Nagashima, “Detection of Cluster Boundary in Microarray Data by Reference to MIPS Functional Catalogue Database”, The 16th Int. Conference on Genome Informatics, Japanese Society for Bioinformatics, Proc. of The 16th Int. Conference on Genome Informatics, Tokyo, Japan, 2005.
[10] 鈴木 英明, 内山 宏樹, 湯田 晋也, “データマイニングによる異常検知技術”, 日本オペレーションズ・リサーチ学会, pp.506-511, 2012.
[11] E. Stehle, K. Lynch, M. Shevertalov, C. Rorres, and S. Mancoridis, “Diagnosis of Software Failures Using Computational Geometry”, 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, USA, Nov., 2011.
[12] E. Stehle, K. Lynch, M. Shevertalov, C. Rorres, and S. Mancoridis, “On the use of Computational Geometry to Detect Software Faults at Runtime”, 7th International Conference on Autonomic Computing, ICAC, Washington, DC, USA, 2010.
[13] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, J.S. Chase, “Correlating instrumentation data to system state: A building block for automated diagnosis and control”, USENIX Association OSDI’04: 6th Symposium on Operating Systems Design and Implementation, 2004.
[14] T.H.D. Nguyen, M. Nagappan, A.E. Hassan, M. Nasser, P. Flora, “An Industrial Case Study of Automatically Identifying Performance Regression-Causes”, MSR Hyderabad, India, 2014.
[15] R. Taylor, E. Rdcs. “Interpretation of the Correlation Coefficient: A Basic Review”, JDMS1, pp35-39, 1990.
[16] Apache JMeter, https://jmeter.apache.org/.
[17] Apache mod_proxy_balancer, http://httpd.apache.org/docs/2.2/ja/mod/mod_proxy_balancer.html/.
[18] Apache Tomcat, http://tomcat.apache.org/.
[19] iBatis JPetStore, http://sourceforge.net/projects/ibatisjpetstore/.
[20] Oracle, http://www.oracle.com/.
[21] collectd, http://collectd.org/.
[22] stress project page, http://people.seas.harvard.edu/-apw/stress/.
[23] J. B. MacQueen, “Some Methods for classification and Analysis of Multivariate Observations”, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability,University of California Press, pp281-297, 1967.
[24] C. Bernard, “An optimal convex hull algorithm in any fixed dimension”, 1993.
[25] S. Iwata, K. Kono, “Clustering Performance Anomalies Based on Similarity in Processing Time Changes”, IPSJ Transactions on Advanced Computing Systems, Vol.5 No.1 1-12, 2012.
[26] GATAG, http://free-illustrations-ls01.gatag.net/images/lgi01a201311181100.jpg.
[27] S. Thrun, C. C. Faloutsos, A. W. Moore, P. Spirtes, G. F. Cooper, “Learning Bayesian Network Model Structure from Data” 2003.
[28] D. Arthur, B. Manthey, H. Roglin, “k-means has polynomial smoothed complexity”, 2009.
[29] N. Friedman, M. Linial, I. Nachman, D. Pe’er, “Using Bayesian Networks to Analyze Expression Data”, Journal of Computational Biology 7, pp601-620, 2000.
[30] T. F. Abdelzaher, K. G. Shin, et al, “Performance guarantees for Web server endsystems: A controltheoretical approach”, IEEE Transactions on Parallel and DistributedSystems, 13(1), pp80-96, 2002.
[31] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth “From Data Mining to Knowledge Discovery in Databases”, 2008.
[32] N. Friedman, D. Geiger, M. Goldszmidt, “Bayesian Network Classifiers”, Machine Learning Volume 29, Issue 2-3, pp131-163, 1997.