論　文

Web サービスシステムの応答性能劣化診断のための学習データ自動選定方法

植田 良一[†a)]　角井 健太郎[†]　爲岡 啓[††]　松下 誠[††]　井上 克郎[††]

Automated Training Data Selection for Performance Degradation Diagnosis of Web Service Systems Using Machine Learning Combination

Ryoichi UEDA[†a)], Kentaro KAKUI[†], Akira TAMEOKA[††], Makoto MATSUSHITA[††] and Katsuro INOUE[††]

あらまし　Web サービスシステムの CPU／メモリ／ネットワーク等を対象とした観測値を元に，システムの状態，特に性能異常の検知に寄与する技術を提案する．計算機システムにおける異常検知の方法は，熟練者が見出した相関に基づくルールベース方式から，過去の運用時データに機械学習等の統計処理を施すことでモデルを生成し，このモデルに基づいて異常の診断を行う統計処理方式へと変化しつつある．機械学習を応用する方法では，診断結果の精度を上げるために事前に学習させるデータ量を十分に多くする必要があるが，機械学習処理は計算量が多いため，やみくもに過去の全データを学習に使うことは現実的ではない．そこで我々は，選別労力をかけずに選んだ初期学習データから開始し，観測データの中から自動的に学習データに追加すべきデータを選別し，これが出現するたびに学習データへ自動的に組み入れ，モデルを逐次更新する方法を提案する．学習データの選別を行わずに全データ追加して学習させた場合と，我々が考案する方法で選定した部分データのみ追加して学習させた場合とで，同じ観測データを診断し，結果を比較する評価実験を行った．その結果，我々の考案した方法で追加すべきデータを選別して学習させた場合，10 種中 7 種の実験で，全データで学習した場合と同等の診断結果が得られる事を確認した．

キーワード　システム性能，性能異常，機械学習，異常検知，学習データ自動選定

1. まえがき

携帯電話をはじめとする携帯端末の普及を背景に，SNS やゲーム等様々なオンラインサービスが提供され，そのサービスを実現する Web サービスシステムの大規模化，複雑化が進展している．これに伴い，システムの安定稼働に向けた障害検知／原因究明を行うために，従来のように CPU 利用率等の個々のメトリクスの監視だけでは不十分となってきた[9]．熟練者が実際の運用を通じて獲得した知識を，障害判断条件とその対応方法の組で表現した if-then ルールによる監視手法は，障害の検知を行うためにある程度の効果はあるものの，ルール記述が困難，人間が認識できない隠れたルールは記述できない，等の課題がある[8]．

そこで，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，システムの正常／異常の判定，異常の根本原因究明に活用する手法が提案されている[5][6][7][8][9][10][11]．

2. 課題の説明

2.1 既知の方法

これまでに提案された，機械学習による異常診断方法の代表的な 2 つを以下で紹介する．

(a) ベイジアンネットワークによる異常診断[8][9]

ベイジアンネットワーク（以下 BN）は，個々の事

† （株）日立製作所 横浜研究所，横浜
Yokohama Laboratory, Hitachi Ltd., 292 Yoshida-chou, Totsuka, Yokohama, Kanagawa, 244-0817, Japan
†† 大阪大学大学院 情報科学研究科，吹田
Graduate School of Information Science and Technology, Osaka University, 1-5 Yamada-Oka, Suita, Osaka, 565-0871, Japan
a) E-mail: ryoichi.ueda.mb@hitachi.com

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 P_E を算出する事ができる．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．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，計測回数に比例して学習時間が長くなる[2].

(b) クラスタリングによる異常診断[5]

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．教師データが存在しない場合，個々のクラスタに対する意味付け（例：正常時のクラスタ等）は当該クラスタに含まれる観測値から人間が判断する．

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[5]，正常範囲の一部しか学習データに含まれない場合，正常であるにもかかわらず異常と判定してしまう．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[5].

2.2 既知の方法の課題

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．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．

2.3 提案手法の発想

我々は，これまでの経験から，Web サービス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と考えている．このような状況では，システムの状態に大きな変化がない期間の膨大な観測データの全てを学習データに利用することは効率的ではない．

そこで，我々は学習データの選定を自動化する方法を模索した．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，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．刻々と収集される観測データの異常診断を行うと同時に，現在の状態が学習データに含まれているかどうかの判定を行い，現在の学習データに含まれていないと判定した場合に，その期間のデータを学習データに追加して再学習する，という一連の動作を繰り返せば，より少量でありながら多種の状態を含む学習データを作り上げることができるとの仮説を立案した．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 に学習させ，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．よって，BN と CL を上記のように組み合わせることで，学習データを自動選定する異常検知システムが実現できると考えた．

3. 学習データ自動選定方法

我々が提案する，学習データの自動選定手法の手順を図 1 に示す．本手法では一定の時間分の連続した観測データの集合をデータ区画と呼び，学習データとしての追加要否を判定する最小単位としている．

はじめに，(1)あらかじめ手動で選定した初期学習データを CL で学習，モデルを生成し，診断を開始する．次に(2)監視対象システムの現在の状態を収集し，CL にて最近傍クラスタからの距離を計算し，記録する．次に(3)当該データ区画の最後まで到達したら，(4)直近のデータ区画の CL での診断結果の平均が閾値

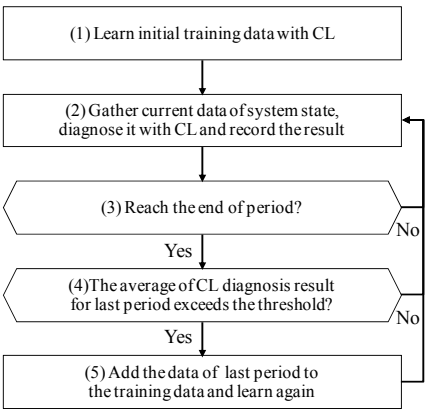

図 1 学習データ自動選定手順
Figure 1 Our automated training data selection process

を越えていないか検査する．閾値を越える事は，監視対象システムが，当該データ区画の間，現在の学習データに存在しない未知の状態にあった事を意味する．ゆえに，閾値を越えていた場合，(5)当該データ区画を学習データに追加し，新たな学習データで再度学習を行う．この再学習プロセスにより，新たな状態を含んだ学習データによるモデルが生成される．

上記(2)から(5)を繰り返すことで，徐々に，CL が未知と判定する頻度が減少し，学習データの集合が収束に向かい，全データと比較して少量の学習データの集合を作り上げる事ができる．

4. 提案手法の有効性検証

我々の目的は，全データを学習した場合と同等の診断結果が得られるような，より少ないデータの集合を自動的に選定する事にある．提案手法の有効性を検証するために，全データを学習データに追加した場合と，提案手法によって自動選定した学習データのみ追加した場合の，それぞれの BN による診断結果（平均応答時間が 1 秒を越える確率の時系列データ）のユークリッド距離と相関係数を比較することとした．全データを学習した場合の診断結果の移動平均を W=(w_1,w_2,w_3, ...)，提案手法を適用した場合の診断結果の移動平均を P=(p_1,p_2,p_3,...)とし，ユークリッド距離 E，相関係数 C をそれぞれ以下の計算式で算出した．$\bar{w}$は$w_1, w_2, \ldots, w_n$の平均値を表す．$\bar{p}$も同様．

$$E = \sqrt{\sum_{i=1}^{n} (w_i - p_i)^2}$$

$$C = \frac{\sum_{i=1}^{n} (w_i - \bar{w})(p_i - \bar{p})}{\sqrt{\sum_{i=1}^{n} (w_i - \bar{w})^2} \sqrt{\sum_{i=1}^{n} (p_i - \bar{p})^2}}$$

これにより，学習データ自動選定結果の善し悪しを判断する．

4.1 評価方法

JPetStore[3]（オンラインペット販売サイトアプリケーション）に，実アクセスパターンに基づくバックグラウンドトラフィックを与えながら，同時に，システムの異常状態を模擬する負荷を，システムを構成するサーバの一部に与える．監視対象システムを構成する全てのサーバで監視エージェントが稼働しており，一定間隔で観測データを監視サーバに集約される．実際の負荷パターンとして，WorldCup98[1]で提供される Web サーバへのアクセスデータの一部を活用した．システム全体の処理能力の差を考慮して秒間リクエスト数を 500 分の 1 程度に縮約し，リクエストの増減パターンのみ維持した．また，システムの異常状態を模擬するために stress コマンドにて負荷を与えた．

はじめに，あらかじめ収集されたデータの中から任意の初期学習データを抽出し，CL にて学習を行い，モデルを生成する．次に，現在のシステムの状態を CL にて診断し，記録しておく．

観測時刻がデータ区画の末尾に到達した際に，直近のデータ区画の，CL での診断結果の平均が閾値を越える場合，未学習であると判断し，当該データ区間のデータを学習データに追加し，再学習を行う．以降の診断は再学習で生成したモデルにて行う．この処理を一定期間繰り返し実行後，生成された学習データが，提案手法にて自動選定された学習データである．

最後に，自動選定された学習データの善し悪しを判定するために，全データを学習した場合と，提案手法にて自動選定した学習データのみ学習した場合の，それぞれの BN による診断結果のユークリッド距離と相関係数を比較する．

4.2 実験システム構成

実験環境（図 2）は，(1)監視対象システム，(2)トラフィック発生部(A)，(3)システム異常検知部(E)，からなる．監視対象システムはロードバランサ(B)，Web-AP サーバ(C_1)(C_2)，DB サーバ(D)からなり，JPetStore アプリケーションが稼働している．各サーバには，CPU/Memory/Network/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)が組み込まれており，システム異常検知部(E)に観測データを送付する．

トラフィック発生部(A)は，あらかじめ与えたシナリオ（例：商品検索→ 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．今回の実験では，LB で#1 から#3 と#5 までの 7 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，DB サーバで#1（2 コア分）と，#2, #3, #4, #6 の合計 7 項目，計 22 項目のシステム監視で一般的なメトリクスを収集した．

監視対象システムには，指定時刻になると，当該システムの性能を劣化させる stress コマンドが実行され

るように設定した．

システム異常検知部は，各サーバ内の監視エージェントが収集する観測データを蓄積し，常時異常検知を行いつつ，必要に応じて再学習を行う．

今回の実験では仮想サーバ環境上に，CPU:1core, Memory:1GB, HDD:20GB, Network:1Gbps のリソースを持つ 2 つの Web/AP サーバ，DB サーバ，LB の合計 4 サーバを使った．DB サーバだけは 2 コアを利用した．各仮想サーバ同士がリソースを取りあう事がないように異なる物理サーバ上に仮想サーバを作成した．

OS は全サーバで Linux(CentOS 6.5)を採用した．また，BN の実装として，統計解析環境 R[4]用ライブラリ bnlearn[2]を，CL の実装として R 標準ライブラリの k-means を活用した．CL のクラスタ数は 10 を指定した．システムの状態数に相当するクラスタ数は過去の運用の経験，および，事前のデータ分析により 10～20 程度が妥当であると判断したため，モデルが最も単純となる 10 を採用した．

表 1 監視対象メトリクス
Table 1　Metrics to be monitored

#	Resource	Monitoring Items
1	CPU(DB has 2 cores, others have 1 core)	User usage for each core (%)
2	Memory	Used (bytes)
3	Network (LB has 2)	Used (bps, sent+received)
4	Disk (except LB)	IOPS (ops/s)
5	Web Access (only LB)	(a)Request count, (b)average and (c)max response time (s)
6	DB Access (only DB)	(a)Processed data (bytes), (b)Written data (bytes)

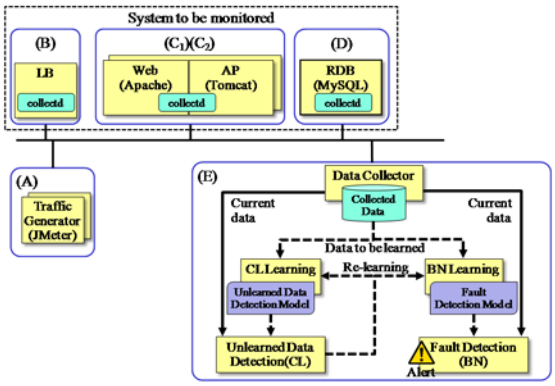

図 2 実験システムの構成
Figure 2　System architecture for our experiment

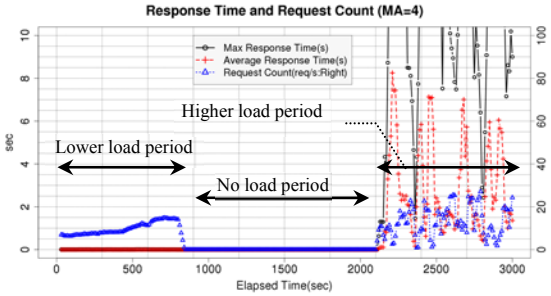

図 3 初期学習データのリクエスト量
Figure 3　Amount of requests for initial training data

4.3 実験方法

以下の(1)から(4)に示す一連の実験プロセスを 1 セットとして，合計 10 セットの実験を行った．10 セット中 5 セットは，初期学習期間を変化させ，診断期間を共通とした．残り 5 セットは逆に初期学習期間を共通とし，診断期間を変化させた．

(1) 初期学習データの生成

WorldCup98 のアクセスパターンの縮約比率を変化させて，(a)監視対象システムが平均応答時間 1 秒以下を維持できる程度のリクエストを処理している状態，(b)リクエストがほとんどない状態，(c)平均応答時間が 1 秒を越える状態，の 3 つの状態が繰り返し発生する 5 時間分のデータを用意した．その中から図 3 に示すように，(a)(b)(c)の 3 状態が現れる 50 分間のデータを異なる時間帯から選択し，初期学習データとして活用した．この 5 時間の間にはシステム障害を模擬する負荷は与えなかった．

(2) サーバ異常の模擬生成

WorldCup98 のアクセスパターン（図 4）を再現したトラフィックをバックグラウンドトラフィックとして与えている状態で，検知すべき障害を模擬する異常を各サーバ内に作り出した．今回の実験では，CPU, Memory, HDD の各リソースに対して stress コマンドにてユーザリクエスト処理を阻害する負荷をかけた．

(3) 提案手法による追加学習データの自動選定

図 4 バックグラウンドトラフィックパターンの例
Figure 4　An example of background traffic pattern

上記(2)を実行中に 10 秒毎に収集される，40 分間の観測データに対し，BN にて平均応答時間が 1 秒を越える確率を逐次診断する．この診断と同時に CL での最近傍クラスタからの距離計算も行い，5 分毎に直近 5 分間の距離の平均が閾値 6 を越えるかどうかを判定，越えた場合は学習データに当該 5 分間分のデータを追加し，再度学習を行う．本実験では，各クラスタの重心ベクトルと 1 回分の観測データのベクトルを用いて，正規化後ベクトル間のユークリッド距離の最小値を，当該観測の最近傍クラスタからの距離として利用した．

以降の観測データに対しては，再学習で生成されたモデルで診断を行う．50 分間の初期学習データに加え，最大で 8 区画 40 分間のデータが学習データに追加される．今回の実験では，後半 8 区画 40 分中の，連続しない 3 区画でシステム異常を模擬する 5 分間の負荷を，2 台の Web サーバと DB サーバのうちいずれか 1 台，または 2 台，または 3 台全てに与えた．診断期間を共通とした 5 セットの実験では，第 2 区画で Web サーバ Web_A に，第 4 区画ではもう一方の Web サーバ Web_B に，第 6 区画では 2 つの Web サーバに加えて，DB サーバにも負荷を与えた（図 5）．診断期間を変化させた 5 セットの実験では，連続しない 3 区画で上記負荷を順不同で与えた．

(4) システム異常検知結果の比較

CL にて 40 分間の診断を終えた後，生成された学習データの善し悪しを判定するために，(a)全データを BN にて学習し，後半 40 分を診断した結果（以下 Whole）と，(b)上記(3)にて自動選定した学習データを BN にて学習し，(a)と同一データを診断した結果の類似性を判定した．類似性判定には，ユークリッド距離と相関係数を採用した．また，学習データの追加の効果を見るために，学習データへの追加を行うたびに，新たな学習データによる診断結果と Whole との類似性を評価した．

異常検知結果の例を図 5 に示す．上段グラフの○点が BN による平均応答時間が 1 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（秒）を表す．下段グラフの○点が 10 秒間のリクエストの最大応答時間（左軸：秒），△点が同 10 秒間の平均応答時間（左軸：秒），＋点が同 10 秒間の処理リクエスト数（右軸：リクエスト／秒），横軸は経過時間（秒）を表す．各グラフタイトルの MA=4 はグラフ上の点が隣接 4 データの移動平均値である事を表す．

4.4 実験結果

表 2 に 1 セット分の実験結果を示す．1 セットの実験は初期学習データのみから生成したモデルで診断した結果(Base)で始まり，1 データ区画を学習データに追加して全データを診断した結果(L1)，さらに 1 データ区画を追加して全データを診断した結果(L2)，と学習データを追加するたびに番号を増やして区別できるようにした．表中の各行を Heat と呼ぶ．

表中 Heat ID は当該実験の Heat 名を，Training Data は当該 Heat の学習データを，Avg Dist.は当該 Heat で初期データに組み入れたデータ区画の最近傍クラスタからの距離の平均を，Euclid.と Correlation は全データで学習したモデルでの BN の診断結果と，当該 Heat のモデルでの BN の診断結果間のユークリッド距離と相関係数を，それぞれ表す．Euclid.が 0 に近付き，Correlation が 1 に近付くほど，Whole との類似性が高い．例えば，表 2 では，L1, L2, ...と学習データを増やすごとにユークリッド距離が 0 に近付き，かつ，相関係数が 1 に近付いている．これは，学習データの追加により，全データを学習した場合と同等の診断結果を出せる学習データの集合に徐々に近づいている事を意味する．

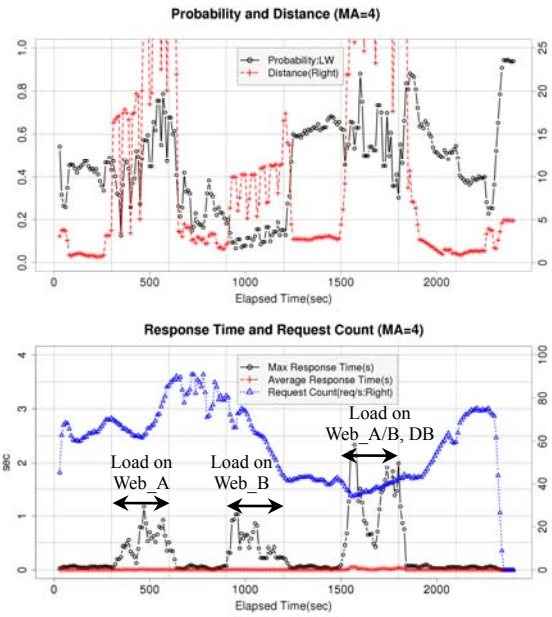

図 5　BN および CL での診断結果の例
Figure 5　A diagnosis result by BN and CL

表 2　実験結果：1 回分
Table 2　Experiment result for 1 turn

Heat ID	Training Data	Avg Dist.	Euclid.	Correlation
Base	Base	-	5.36	-0.16
L1	Base+0-5	20.49	3.21	0.34
L2	L1+5-10	21.97	3.03	0.38
L3	L2+15-20	6.02	0.91	0.66
L4	L3+25-30	8.79	0.16	0.98

Base の診断結果を可視化したグラフを図 6 に示す．第 1 区画（冒頭の 5 分）で CL の計算結果（△点，右軸）の平均が，あらかじめ我々が設定した閾値 6 を越えている（平均 20.49）ため，当該区画を学習データに組み入れる．組み入れた後の Heat である L1 の診断結果を図 7 に示す．L1 では第 2 区画（5〜10 分）で，CL の計算結果が 6 を越えている（平均 21.97）ため，当該区画を学習データに組み入れる．同様に L2, L3 と処理を進めた結果である L4 の診断結果（閾値越えなし）を図 8 に示す．

この実験では第 1,2,4,6 の 4 区画を初期学習データに追加した L4 が最終形で，Heat が L1 から L4 まで進む間に，ユークリッド距離および相関係数が共に改善された．最終的に L4 で，Whole とのユークリッド距離が 0.16，相関係数が 0.98 となり，全 8 区画中 4 区画の観測データで，全区画を学習した場合（図 9）と同等の診断結果が得られた．

この実験では，さらに，提案手法による学習データ選定の妥当性を検証するために，8 区画中，L4 で選択しなかった 4 区画を学習した場合(R4)や，提案手法による選定とは無関係に，L4 と同じ 4 区画分を，単純に 8 区画中の前半(F4)，中盤(C4)，後半(S4)の 4 区画，および，先頭と末尾の 2 区画ずつを選択した場合(P4)の診断結果との比較を行った（表 3）．

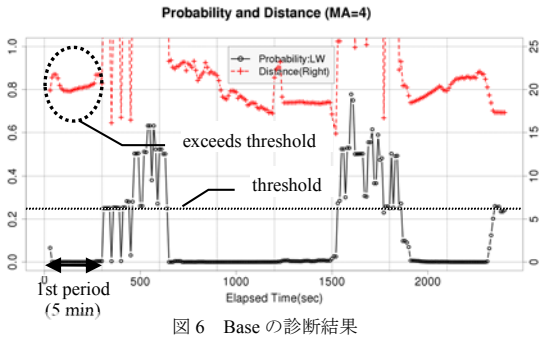

図 6　Base の診断結果
Figure 6　Diagnosis result of Base

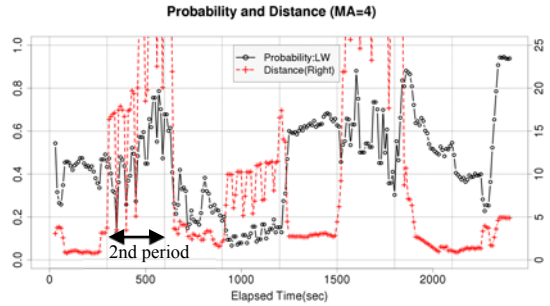

図 7　L1 の診断結果：第 2 区画で CL(△点)が閾値を越える
Figure 7　Diagnosis result of L1, CL value exceeds the threshold at 2nd period

表 3 他の学習結果との比較
Table 3　Comparison with other training data case

Heat ID	Training Data	Avg Dist.	Euclid.	Correlation
L4	L3+25-30	8.79	**0.16**	**0.98**
R4	Reverse of L4	-	0.38	0.96
F4	Base+0-20	-	0.42	0.78
S4	Base+20-40	-	1.21	0.85
C4	Base+10-30	-	1.50	0.60
P4	Base+0-10,30-40	-	1.49	0.43

その結果，いずれも L4 の結果には及ばず，L4 が Whole に最も近い診断結果を得られる事が分かった．

同様の実験（L1 から最大 L4 まで）を 10 回行った結果を表 4 と表 5 に示す．Exp. ID は実験 ID，Heat ID は当該実験で提案手法を適用した場合の最終 Heat ID または比較対象の Heat ID，Euclid.は当該 Heat と全データを学習させた結果とのユークリッド距離，Correlation はその相関係数を示す．各実験の 1 行目は最終 Heat の結果を，2 行目に，前項で説明したような他の学習データを選定した場合で最良の結果を出した Heat の結果を比較対象として記載した．Heat ID が R で始まる Heat は，提案手法で選択しなかった区画の中から，R に続く数字分の区画をランダムに選択して学習データに追加した場合の結果を意味する．

また，Eval.には提案手法が我々の期待通りの選定結果を出せた場合に O，そうでない場合に X を記載した．ただし，ユークリッド距離および相関係数の一方でのみ優位性が現れた，実験 Exp_Dd に関しては，相関係数のわずかな差よりもユークリッド距離の顕著な差を重要視して O とした．

実験の結果，提案手法は 10 種中 7 種の実験で期待する結果を出す事ができた．また，初期学習データを

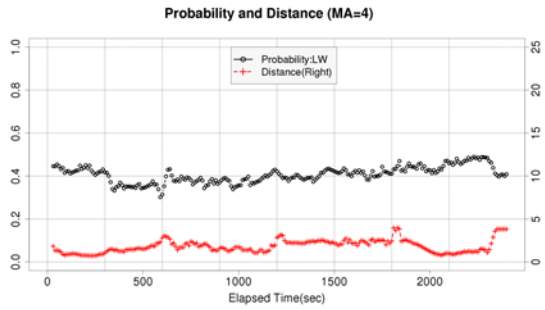

図 8　L4 の診断結果：CL の閾値越えなし
Figure 8　Diagnosis result of L4, CL value does not exceed the threshold, then compare with Whole

変更した場合と，診断データを変更した場合の実験結果に顕著な違いは見られなかった．

以上から，提案手法は大量の観測データの中から学習データとして利用すべき部分データを自動的に選択可能である事，また，BN の学習時間は学習データ量に比例するため[2]，ひいては，異常診断を行う BN の学習時間の短縮に貢献できる事が分かった．

4.5 結果の考察

今回の実験では再学習の発生頻度が上がりすぎないようにするために 5 分の固定長で各区画を区切り，この区切りに合わせて監視対象システムへ負荷をかけている．しかし，結果を詳細に分析したところ，観測データが表す負荷はこの 5 分区画の境界を越えて次の区画に影響を及ぼしている事が分かった．これにより，提案手法で選定した区画の次の区画を学習すると選定区画の一部を学習したのと同じ効果が得られる可能性があると考えられる．

この状況を勘案すると，負荷の影響の始まりと終わりを正しく認識して学習データに組み入れ，かつ，再学習の発生頻度を一定以下に制御することが，今後の改良点としてあげられる．

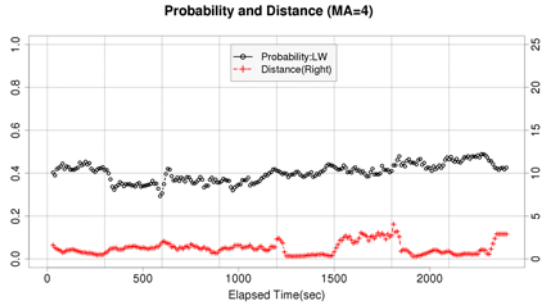

図 9　Whole の診断結果
Figure 9　Diagnosis result of Whole

実際に，実験 Exp_Sc では我々の提案手法が選定しなかった区画だけを学習した Heat R4 が，提案手法で選定した区画を学習した Heat L4 に準じる結果を出したが，R4 の学習に使った各区画から直前の区画（L4 に含まれている）の影響が残っている先頭の 1 分間（各区画 5 分の先頭 1 分）のデータを取り除いて実験を行った結果，相関係数が 0.96 から-0.03 へ，ユークリッド距離が 0.38 から 2.62 へと変化し，類似性が極めて低くなるという現象を確認した．

表 4 初期学習データを変更した場合の実験結果
Table 4　Result of experiments with different initial training data

Eval.	Exp. ID	Heat ID	Euclid.	Correlation
X	Exp_Sa	L4	0.66	0.36
		R4	0.60	0.88
O	Exp_Sb	L2	2.57	0.95
		R2	3.19	0.95
O	Exp_Sc	L4	0.16	0.98
		R4	0.38	0.96
X	Exp_Sd	L4	1.27	-0.07
		R4	0.73	0.28
O	Exp_Se	L4	1.26	0.37
		R4	1.31	0.30

表 5 診断データを変更した場合の実験結果
Table 5　Result of experiments with different diagnosis data

Eval.	Exp. ID	Heat ID	Euclid.	Correlation
O	Exp_Da	L2	1.27	0.90
		R2	1.45	0.90
X	Exp_Db	L2	1.34	0.90
		R2	0.97	0.95
O	Exp_Dc	L3	0.39	0.97
		R3	3.50	0.88
O	Exp_Dd	L3	0.57	0.80
		R3	2.42	0.85
O	Exp_De	L3	0.97	0.95
		R3	3.37	0.72

5. 関連研究

Ira Cohen らは[8]にて，我々の提案手法と同類の TAN(Tree-Augmented Bayesian Network)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．Steve Zhang らは[9]にて，本手法を拡張して，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を結合するのに比べ，我々の手法では学習データを結合する点が異なる．学習データの結合処理がデータファイルの単純な結合で実現できるのに比べ，モデルの結合は時間のかかる複雑な処理となる．

Satoshi Iwata らは[10]にて，性能劣化の根本原因を推定する方法を発表した．システムを構成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．事前に人手による分析を終えた学習データ（教師データ）を必要とする事が我々の手法と異なる点である．

Thanh H. D. Nguyen らは[11]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．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%の精度で原因究明できる事を確認した．少量ではあるが，事前に人手による分析を終えた学習データ（教師データ）を必要とする事が我々の手法と異なる点である．

6. むすび

Web サービスシステムの CPU／メモリ／ネットワーク等の利用状況の観測値を元に，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，学習データの自動選定手法を考案した．

オンライン販売システムを使い，あらかじめ用意した初期学習データに加えて，提案手法で自動選定した学習データを追加した場合の診断結果と，全データによる診断結果の類似性を判定する評価実験を行った結果，10 種中 7 種で，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，(1)CL での既学習／未学習を判定する際の閾値の決定方法，および，(2)未学習データを学習データに組み入れる際のデータ区画長の決定方法があると考える．本実験では，CL の判定閾値として，既学習データに対しては十分に大きいと人間が判定した 6 を，また，データ区画の分割方法として固定長 5 分を，それぞれ採用したが，後者に関してはその問題点を確認した．将来は，前者に関しては自動化を，後者に関しては CL の出力結果が急激に大きくなるエッジを捉えてから，急激に小さくなるまでの，可変長のデータ区画を採用し，かつ，再学習の頻度が上がりすぎないように制御するのが適切と考える．

文　献

[1] WorldCup98 Site Access Data, http://ita.ee.lbl.gov/html/contrib/WorldCup.html

[2] bnlearn - an R package for Bayesian network learning and inference http://www.bnlearn.com/

[3] JPetStore, http://sourceforge.jp/projects/sfnet_ibatisjpetstore/downloads/jpetstore4/4.0.5/iBATIS_JPetStore-4.0.5.zip/

[4] The R Project for Statistical Computing, http://www.r-project.org/index.html

[5] 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, Sep., 2012.

[6] 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.

[7] 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 2010, Washington, DC, USA, Jun., 2010.

[8] 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.

[9] 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, Jun. 2005.

[10] 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, Jan. 2012.

[11] T.H.D. Nguyen, M. Nagappan, A.E. Hassan, M. Nasser, P. Flora, "An Industrial Case Study of Automatically Identifying Performance Regression-Causes", MSR'14, Hyderabad, India, May., 2014.

縦 26.4mm
横 20mm

植田　良一（非会員）

1994 大阪大学大学院卒．2015（株）日立製作所横浜研究所．現在，クラウド分野の研究に従事．

縦 26.4mm
横 20mm

角井 健太郎（）

縦 26.4mm
横 20mm

爲岡　啓（正員）

縦 26.4mm
横 20mm

松下　誠（正員）

縦 26.4mm
横 20mm

井上　克郎（正員）

（平成 XX 年 XX 月 XX 日受付，XX 年 XX 月 XX 日再受付）

Abstract

We propose an automated training data selection method for response time degradation diagnosis of web service systems using low-level metrics such as CPU and network usage. Our method uses clustering to decide if data in latest period should be included in the training data as unobserved state of the system or not, and if necessary the data would be included and learn again with the new training data. As this process goes on, smaller set of training data than whole data is extracted. We experimentally evaluated our method and made sure its effectiveness in 7 out of 10 patterns.

key words

Web Service System, Response Time Degradation, Machine Learning, Failure Detection, Automated Training Data Selection