分散システムの性能異常に対する機械学習の解釈性に基づく原因診断手法

鶴田 博文[†1,a)]　坪内 佑樹[†1,†2,b),c)]

概要：Web サービスを構成する分散システムは，利用者からの多様な要求に応えるために，システム構成が複雑化している．また，システムへの変更頻度が高くなっており，システム構成の変化が速くなっている．これらの要因により，システムに性能異常が起きた際に，システム管理者が原因の診断に要する時間が増大するため，迅速な原因診断手法が必要である．先行手法として，システムの性能を示す時系列データであるメトリックに機械学習モデルを適用する手法がある．しかし，モデルとして学習に長い時間を要する深層学習が用いられているため，迅速に診断を行うには事前にモデルを学習する必要がある．モデルへの入力となるメトリックの系列数は固定であるため，システム構成が変更されて系列数が増減する場合，新たなモデルを学習しなければならない．これにより，システム構成の変更に迅速に追従した原因診断が難しい．解決方法として，高速に学習できる軽量な機械学習モデルを用いて，異常検知後に学習を行う方法が挙げられる．しかし，軽量な機械学習モデルは一般に深層学習よりも表現力が低いため，それに伴い診断精度が低くなる可能性がある．一方，機械学習モデルの予測の解釈性に関する研究が現在盛んに行われており，これらが原因診断にも有用であることが示されている．本論文では，異常検知後に軽量な機械学習モデルを学習し，解釈手法として注目されているシャープレイ値を用いて原因診断を行う手法を提案する．提案手法は，異常検知後の学習により，システム構成が頻繁に変更される場合でも常に現状の構成を反映した診断ができる．また，シャープレイ値が診断精度を高められるか検討する．実験から，提案手法は原因のメトリックの系列を 44.8%の精度で上位 1 位，82.3%の精度で上位 3 位以内に特定することを示した．

A Method for Diagnosing the Causes of Performance Issues in Distributed Systems Based on the Interpretability of Machine Learning

Hirofumi Tsuruta[†1,a)]　Yuuki Tsubouchi[†1,†2,b),c)]

Abstract: To respond to various demands from users, the configuration of distributed systems becomes more complex. In addition, the system configuration is changing faster due to more frequent changes in the system. Since these factors increase the time required to identify the cause of performance issues, a rapid method for diagnosing the cause is necessary. A previous method is to apply machine learning to metrics, which are time-series data that indicate the system performance. However, since deep learning, which takes a long time to train, is used as the model, it is necessary to train the model before an anomaly occurs. Since the number of metric series used as input to the model is fixed, if the number of series changes due to a change in the system configuration, a new model must be trained. As a result, it is difficult to diagnose the cause of issues by quickly following changes in the system configuration. One solution is to use a lightweight machine learning model that can be trained quickly, and train after the anomaly is detected. However, lightweight models have less expressive power than deep learning model, which may result in lower diagnostic accuracy. On the other hand, in the field of the interpretability of machine learning, methods for calculating the contribution of features to the model prediction have been studied, and these have been shown to be useful in diagnosing causes. In this paper, we propose a method for diagnosing the cause of issues using a lightweight machine learning model and Shapley value, which has attracted attention as an interpretation method for machine learning. The proposed method can always reflect the current configuration even when the system configuration is frequently changed. We also investigate whether the diagnosis accuracy can be improved by using the Shapley value. The experimental results show that the proposed method identifies the causal metric series to the top 1 with 44.8% accuracy and to the top 3 with 82.3% accuracy.

1. はじめに

e コマース，SNS（Social Networking Service），動画配信や IoT（Internet of Things）などの IT の商用システムを提供する際に，クラウドの分散コンピューティング環境を使用して，開発・運用されることが一般的である．クラウド上のシステムは，複数の異なるサーバが相互にネットワーク通信するような分散システムとして構成される．システムの利用者が増大すると，サーバへの負荷を分散する必要があるため，サーバ数を増大させるスケールアウト戦略がとられる．また，システムの機能に変更を加えたときの全体への影響を低減するために，機能ごとの小さなサブシステムとしてシステムを分割して再構成するマイクロサービスアーキテクチャ [1] が普及している．マイクロサービスアーキテクチャの採用により，より小さな単位でシステムの変更が可能となるため，システムの変更が容易になっている．Puppet 社の報告 [2] によると，先進企業では，必要であれば 1 時間より短い時間でシステムを変更するようになっているため，システムの変更頻度が高くなっている．

クラウド上に展開された分散システムの構成要素数と構成要素の種類が増大しているため，システム管理者はシステムの状態を把握することが難しくなっている．そのため，システム管理者は，各構成要素の CPU やメモリなどのリソース消費量，秒間リクエスト数，および，応答速度などのシステムの状態を知るための手がかりを定量的に示す指標（以降，メトリックとする）を時系列データとして常時収集している [3]．しかし，構成要素数は増大し続けているため，それに比例して，収集されるメトリックの系列数が増大している [4]．そのため，システムの性能に異常が発生したときに，システム管理者が収集されたメトリックの系列を網羅的に目視することに要する時間が増大している．また，システムへの変更頻度が高くなっていることから，システムの構成や負荷特性の変化が速くなっている．そのため，システム管理者の認知負荷が増大している．これにより，システム管理者が，異常発生時にその原因の特定に要する時間が長くなり，サービスの停止時間の増大やそれに伴う機会損失などに繋がる．したがって，メトリックの系列数の増大とシステムの構成と負荷特性の変化に対応するために，システム管理者がシステムの異常の原因を迅速に診断するための手法が必要となる．

先行手法として，機械学習モデルを用いたアプローチ [5,6] がある．これらの手法は，メトリックの系列間の複雑な依存関係を機械学習モデルを用いて学習し，学習後のモデルの予測値と実測値との誤差の総和を異常度とする．このモデルから得られる異常度に対して，モデルへの入力の特徴量である各メトリックの貢献度を算出することで，異常の原因診断を行う．しかし，既存手法では，モデルとして学習に長い時間を要する深層学習が用いられているため，異常を検知後に迅速に原因診断を行うためには事前にモデルを学習しておく必要がある．モデルの学習の際に入力となる特徴量の数は固定であるため，システム構成が変更されてメトリックの系列数が増減した場合は，再度，新たなモデルを学習しなければならない．これにより，システムの変更に迅速に追従した原因診断を行うことが難しい．

この課題を解決する方法として，深層学習に比べて高速に学習が可能である軽量な機械学習モデルを用いて，異常の検知後にモデルの学習を行う方法が挙げられる．この場合，システムの変更に追従してモデルを学習し直す必要がなく，異常発生時のシステムの構成を反映した原因診断ができる．しかし，一般に軽量な機械学習モデルは深層学習を用いたモデルよりも複雑なデータの傾向を捉える能力が低いため，それに伴い原因診断の精度が低くなる可能性がある．一方，機械学習の解釈性の研究分野において，モデルの予測値に対する入力の特徴量の貢献度を算出する手法が，ここ数年で多く提案されている [7]．協力ゲーム理論のシャープレイ値 [8] を用いた機械学習の解釈手法は現在注目されている手法の一つであり，異常の原因診断に有用であることを示すいくつかの結果が報告されている [9–11]．このシャープレイ値の計算は，一般に特徴量の数が増えるにつれて計算量が膨大になる [12]．しかし，既存研究では即時性が求められる異常の原因診断において，シャープレイ値の計算が実用的な時間内に計算可能であるかの議論は行われていない．

本論文では，異常の検知を起点に軽量な機械学習モデルを学習し，シャープレイ値を用いてモデルから得られる異常度に対する各メトリックの貢献度を算出することで，異常の原因診断を行う手法を提案する．提案手法では，モデルの学習を事前に行わずに異常検知後に行うため，システムの構成が頻繁に変更される場合でも常に現状の構成を反映した診断ができる．また，機械学習の解釈性の分野で有用性が示されているシャープレイ値を用いることで，診断の精度を高められるか検討する．シャープレイ値の計算を高速化するために，シャープレイ値を近似する手法である SHAP [13] を用いる．マイクロサービスのテストベッド環境にて異常を再現する実験の結果，提案手法は，異常の根本原因のメトリックの系列を 44.8%の精度で上位 1 位に，82.3%の精度で上位 3 位以内に特定できることを確認した．また，原因診断にかかる時間を評価した結果から，シャープレイ値を用いた原因診断が実用的な時間内で診断結果を返すことができるシステムの構成要素数を議論した．

†1 さくらインターネット株式会社 さくらインターネット研究所
SAKURA internet Research Center, SAKURA internet Inc., Akasaka, Chuo-ku, Fukuoka 810-0042 Japan
†2 京都大学情報学研究科, Graduate School of Infomatics, Kyoto University, Kyoto 606–8501, Japan
a) hi-tsuruta@sakura.ad.jp
b) y-tsubouchi@sakura.ad.jp
c) y-tsubouchi@net.ist.i.kyoto-u.ac.jp

本稿の構成を述べる．2 章では，関連研究について述べる．3 章では，提案する異常の原因診断手法を説明する．4 章では，提案手法の評価の結果と考察を述べる．最後に，5 章では，本論文をまとめ，今後の展望を述べる．

2. 関連研究

2.1 分散システムにおける原因診断手法

分散システムにおいて利用者へ悪影響のある性能異常を検知した場合，システム管理者は，システムの状況の観察と診断により異常の原因を特定し，その原因を除去して異常から復旧させる．システム管理者は，システムの状況を観察するために，各種メトリックをデータベース等に収集し，主要なメトリックを可視化するためのダッシュボードを作成する [14]．しかし，メトリックの系列数が増大する場合，システム管理者が異常発生時にダッシュボード上のメトリックを網羅的に目視して原因を特定することは難しいため，異常の原因を自動的かつ迅速に診断できる手法が必要である．

先行手法として，システムから収集したメトリックの系列間の因果グラフを自動で推定するアプローチ [15–17] がある．推定した因果グラフ上で，異常を観測したメトリックを起点に有向エッジをたどることで，原因となるメトリックを探索する．このアプローチは，原因となるメトリックの特定に加えて，異常を観測したメトリックから原因となるメトリックまでの伝搬経路も特定することを目的としている．一方，その結果，原因診断を行うためには異常を観測したメトリックから原因となるメトリックまでの因果の向きを全て正しく推定しなければならない．先行手法で因果グラフを推定するために採用されている PC アルゴリズム [18] は，メトリックの系列間の条件付き独立性により構築した無向グラフからルールベースで因果の向きを決定しており，向きが決定できずに無向エッジのまま残る場合がある．この場合，原因となるメトリックを探索することができなくなる．したがって，異常の伝搬経路を特定することを目的とせず，原因となるメトリックの特定のみに目的を絞ることで，さらに原因診断の精度を高める余地がある．

メトリックの特定のみを目的とする手法として，機械学習モデルを用いたアプローチ [5, 6] がある．これらの手法では，まず，正常時におけるメトリックの系列間の複雑な依存関係を機械学習モデルで学習する．次に，異常発生時に観測されたメトリックを学習済みのモデルに入力し，各メトリックごとの予測値を得る．最後に，メトリックごとの予測値と実測値との誤差の総和を異常度とし，その異常度に対する各メトリックの貢献度を算出する．貢献度として，Wu らの手法 [5] では，メトリックごとの誤差を異常への貢献度として採用し，MTAD-GAT [6] では，メトリックごとの誤差をベースに独自に貢献度を定義している．いずれの手法においても，得られた各メトリックの貢献度を降順に整列することで，異常の原因となるメトリックを特定できる．しかし，いずれの手法も深層学習をベースとしたモデルを採用しているため，モデルの学習に長い時間を要する．学習時間は，モデルのパラメータや計算環境，データセットなど様々な要因に影響を受けるが，一般に数時間オーダーである [19]．そのため，異常発生後に迅速に原因診断を行うためには事前にモデルを学習しておく必要がある．モデルの学習の際に入力となる特徴量の数は固定であるため，システムの構成要素に変更があり，診断に用いるメトリックの系列数が増減した場合は，数時間をかけて新たなモデルを学習しなければならない．そのため，新たなモデルの学習が完了するまでは，システムの変更を反映した原因診断を行うことができない．

2.2 機械学習の解釈性

機械学習モデルは一般に計算過程が複雑であるため，モデルの予測の根拠を人間が理解できないことが問題視されている．そのため，機械学習モデルの解釈性についての研究が盛んに行われている．特に，特定の入力データに対する学習後のモデルの予測を解釈することは局所的な解釈と呼ばれ，これまでに多くの手法が提案されている [7]．代表的な局所的な解釈手法として，協力ゲーム理論のシャープレイ値 [8] を用いた解釈手法 [12, 13] がある．シャープレイ値とは，複数のプレイヤーが協力してゲームに取り組んだ結果として得られた利得を各プレイヤーの貢献度に応じて公正に分配するための手段の一つである．シャープレイ値では，あらゆるプレイヤーの組み合わせで得られる利得を元に，他のプレイヤーとの協力により発生する利得の影響を排除した特定のプレイヤーの貢献度を算出する．協力ゲームにおけるプレーヤーをモデルに入力する特徴量，利得をモデルが出力する予測値に置き換えることで，モデルの予測に対する各特徴量の貢献度を算出できる．

機械学習モデルにより異常検知を行う場合，正常と異常のラベルが付与されたデータを十分に集めることが困難であることや，これまで観測されたことがない未知の異常に対応することなどの理由から，正常時のデータのみを用いた教師なし学習が行われることが多い．モデルは正常時のデータの特徴を学習し，学習後に入力された特定のデータの正常時からの逸脱に基づき異常度を出力する．このようなモデルとシャープレイ値を用いることで，異常度に対する各特徴量の貢献度を求めることができるため，異常の原因となる特徴量を特定することに活用できる．これまでに，シャープレイ値を用いた解釈手法は，異常の原因診断に有用であることを示すいくつかの結果が報告されている [9–11]．一方，シャープレイ値は，あらゆる特徴量の組み合わせにおけるモデルの予測値を元に計算が行われるため，特徴量の数が増えるにつれて組み合わせの数が膨大に

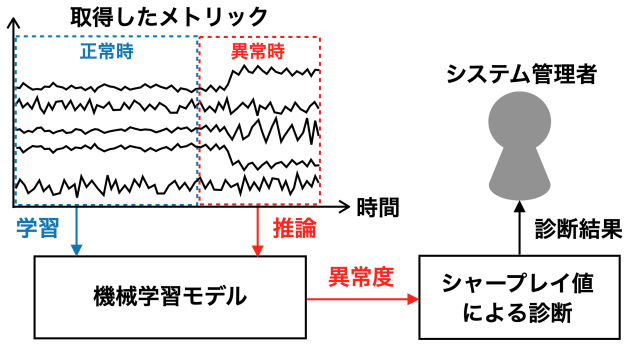


図 1: 提案手法の概要図

なり，計算時間が増大する [12]．しかしながら，先行研究では，シャープレイ値の計算を含む原因診断に要する時間の評価が行われておらず，即時性が求められる異常の原因診断において，シャープレイ値を用いた原因診断が実用的な時間内に計算可能であるかの議論が必要である．

3. 提案する原因診断手法

異常の原因となるメトリックの診断精度を高めるためには，異常の伝搬経路を特定することを目的とせず，原因となるメトリックの特定のみに目的を絞った機械学習モデルを用いたアプローチが有効である．一方，既存の機械学習モデルを用いたアプローチでは，システムの構成が頻繁に変更される場合，その変更に迅速に追従した原因診断を行うことが難しい．本論文では，異常の検知後に高速に学習可能である軽量な機械学習モデルと，機械学習の解釈手法として注目されているシャープレイ値を用いて，高い診断精度とシステム構成の変更に対する追従性を両立した上で，高速性を持つ原因診断手法を提案する．

3.1 アーキテクチャの概要

図 1 は，提案手法の概要図である．機械学習モデルは，現在異常検知に広く利用されているが，提案手法では，機械学習モデルを用いて異常検知を行わず，異常が検知された後の原因診断を行うことに焦点を当てている．異常検知のためには，サービスの信頼性に関する目標値である SLO（Service Level Objective）やメトリックの系列ごとに設定した閾値などを用いる．また，機械学習モデルは事前に学習を行ったモデルを用いて予測を行うことが一般的であるが，提案手法では，システム構成が頻繁に変更される場合に対応するために，予測を行う直前に学習を行う．以下，提案手法の実行の流れを順を追って説明する．

まず，異常が検知された場合，システムの各構成要素から得られるメトリックを検知の時間から一定期間遡って取得する．取得した複数のメトリックの系列には，異常時のメトリックの系列と，その直近の正常時のメトリックの系列が含まれている．そのため，図 1 のようにメトリックの系列を時間の軸に対して正常時と異常時に分割する．実際には，取得したメトリックの期間のうち，異常時のメトリックの時間範囲（以降，これを診断期間と呼ぶ）は不明である．現在の提案手法では，システム管理者が任意の診断期間を設定する必要がある．分割したデータのうち，正常時のメトリックのみを学習データとして，機械学習モデルの教師なし学習を行う．

次に，異常時のメトリックを学習済みのモデルに入力し，モデルからの出力として異常度を得る．この異常度に対する各メトリックの貢献度をシャープレイ値を用いて算出し，算出した貢献度を降順に整列することで，異常の原因となるメトリックの候補をランク付けする．最後に，ランク付けしたメトリックを診断結果として，システム管理者に提示する．

3.2 機械学習モデルの学習

システム構成の変更に対する追従性の要件を満たすために，提案手法では機械学習モデルの学習を事前に行わずに異常検知後に行うアーキテクチャを採用する．これにより，システム構成の変更のたびにモデルを学習し直す必要がなく，システムの構成を頻繁に変更する場合でも常に異常時の構成を反映した診断が可能である．

モデルの学習を異常検知後に行う場合，異常検知からシステム管理者に診断結果を提示するまでの時間にモデルの学習の時間が含まれることになる．そのため，異常検知後に迅速に診断結果を提示するためにはモデルの学習を高速に行う必要がある．例えば，モデルとして深層学習ベースのモデルを用いる場合，一般に数時間オーダーの学習時間を要するため，異常検知後に学習することは実用上問題となる．また，モデルに求められる他の要件として，モデルは学習後に与えられる特定のデータに対して異常度を返すモデル，すなわち，異常検知の用途に利用可能なモデルである必要がある．

以上の要件を満たすモデルとして，主成分分析（PCA）ベースのモデルが挙げられる．PCA は，高次元のデータをなるべく情報が失われないように，より低次元のデータに圧縮する次元圧縮手法の一つである．PCA は，正常時のデータのみを用いて，低次元への圧縮方法を学習する．学習後，与えられる特定のデータに対して，一度次元を圧縮した後に再度元の次元にデータを復元した際の誤差（以降，再構成誤差と呼ぶ）を異常度として，異常検知を行うことができる [20, 21]．PCA の学習は与えるデータのサイズに依存するものの，深層学習をはじめとした他の機械学習モデルに比べて高速に学習が可能である．

3.3 シャープレイ値による診断

前節で述べた機械学習モデルは，正常時のメトリックを用いて学習を行い，異常時のメトリックを学習済みのモデルに入力することで異常度を得る．提案手法では，この異

表 1: テストベッド環境のハードウェアとソフトウェアの構成

項目	仕様	
Worker ノード (サービス用途)	CPU	Intel Xeon CPU 2.20GHz
	vCPU	2 core
	Memory	4 GiB
	Type	e2-medium
	OS	Container-Optimized OS
Worker ノード (管理用途)	CPU	Intel Xeon CPU 2.20GHz
	vCPU	2 core
	Memory	4 GiB
	Type	e2-medium
	OS	Container-Optimized OS
解析用サーバ	CPU	Intel Xeon CPU 2.40GHz
	vCPU	4 core
	Memory	24 GiB
	OS	Ubuntu 18.04
	Python	v3.8.2
Kubernetes	Version	1.20.8-gke.900
Prometheus	Version	2.20.0

常度に対する各メトリックの貢献度をシャープレイ値を用いて算出することで，異常の原因となるメトリックを特定するアーキテクチャを採用する．シャープレイ値は，これまでに異常の原因診断に有用であることを示すいくつかの結果が報告されているため [9–11]，シャープレイ値を用いることで，原因診断の精度を高めることが期待できる．

診断の高速性の要件を満たすためには，モデルの学習だけでなく，シャープレイ値の計算も高速に行う必要がある．一般に，シャープレイ値は，計算量が大きく，特に特徴量の数が増大するにつれて厳密な計算が不可能になる [12]．そのため，シャープレイ値を高速かつ正確に近似する手法が必要となる．

以上の要件を満たすシャープレイ値の近似手法として，SHAP [13] が挙げられる．SHAP には，いくつかの近似アルゴリズムが存在し，例えば Kernel SHAP では，重み付きの最小二乗線形回帰によりシャープレイ値を近似できる．いずれの近似アルゴリズムも，シャープレイ値の厳密な計算と同様に，特徴量の数が増大にするにつれて計算量が大きくなる．そのため，本論文では，SHAP を用いた原因診断が実用的な時間内に計算可能であるかを，テストベット環境を用いて評価し，4 章に結果を示す．

4. 評価

本章では，マイクロサービスのテストベッド環境にて異常を再現する実験により，提案手法の原因診断の精度および診断に要する時間（以降，診断時間とする）を評価する．

4.1 実験の環境

実験に用いたシステムの環境は以前の我々の研究 [22] を参考に構築した．

テストベッド 表 1 にテストベッド環境のハードウェアとソフトウェアの構成を示す．マイクロサービスのベンチマークアプリケーションである Sock Shop[*1] を Kubernetes クラスタの自動管理サービスである GKE（Google Kubernetes Engine）[*2] 上に構築した．Kubernetes クラスタは 5 台の Worker ノードから構成されており，4 台を Sock Shop を配置するためのサービス用途，1 台をメトリックの収集と負荷生成のための管理用途とした．また，原因診断のために用いる解析用サーバは，さくらのクラウド[*3] 上の仮想サーバを用いた．

表 2: 診断に用いるコンテナのメトリック

メトリック名	説明
container_cpu_usage_seconds_total	CPU 使用時間
container_memory_working_set_bytes	メモリ使用量
container_network_transmit_bytes_total	送信バイト数
container_network_receive_bytes_total	受信バイト数
container_fs_writes_total	ファイルシステムへの書き込み数
container_fs_reads_total	ファイルシステムへの読み込み数

メトリックの収集 コンテナのリソース使用量などのメトリックを取得するために，cAdvisor[*4] を用いた．メトリックの収集には，システムの構成要素から各種メトリックスを収集・保存するためのツールである Prometheus[*5] を用いた．メトリックは，15 秒間隔で収集した．

負荷生成 Sock Shop に対して利用者からのアクセスなどによる負荷を擬似的に生成するために，負荷試験のツールである Locust[*6] を用いた．

4.2 実験の設定

異常の再現 マイクロサービスの性能異常を擬似的に生成するために，関連研究 [5] の実験で採用されている CPU 過負荷およびメモリリークを特定のコンテナ上で再現した．異常の再現には，Kubernetes 管理化のコンテナに対して故意に異常を再現可能である LitmusChaos[*7] を使用した．LitmusChaos は様々な異常をサポートしており，CPU 過負荷に相当する異常は pod-cpu-hog[*8]，メモリリークに相当する異常は pod-memory-hog[*9] である．

診断に用いるメトリックの取得 異常を再現後，5 分経過したのちに直近 30 分間のコンテナのメトリックを取得した．取得したコンテナのメトリックは，メトリックの種類が多い，かつネットワークの送信バイト数と送信パケッ

*1 Sock Shop: `https://microservices-demo.github.io/`
*2 Google Kubernetes Engine: `https://cloud.google.com/kubernetes-engine`
*3 さくらのクラウド: `https://cloud.sakura.ad.jp/`
*4 cAdvisor: `https://github.com/google/cadvisor/`
*5 Prometheus: `https://prometheus.io/`
*6 Locust: `https://locust.io/`
*7 LitmusChaos: `https://litmuschaos.io/`
*8 pod-cpu-hog: `https://hub.litmuschaos.io/generic/pod-cpu-hog`
*9 pod-memory-hog: `https://hub.litmuschaos.io/generic/pod-memory-hog`

ト数のように単位は異なるが，時間に対して概ね同様の変動傾向を示す冗長なメトリックが存在する．そのため，取得した全てのメトリックを診断に用いる場合，診断にかかる時間の増大や精度の低下が起きる．そこで本実験では，診断に用いるメトリックとして，表 2 に挙げた 6 個のメトリックを各コンテナから取得した．メトリックを取得したコンテナの数は 11 であるため，合計 66 個のメトリックを診断に用いた．メトリックの系列間の単位の違いを統一するために，各系列の平均が 0，分散が 1 となるように標準化した．

提案手法 機械学習モデルとして，異常検知に広く用いられており，高速に学習が可能な PCA を用いた．また，シャープレイ値の計算には，あらゆる機械学習モデルに適用可能な Kernel SHAP を用いた．Kernel SHAP の計算を高速化するために，マルチコアで並列計算が可能な ShapPack[*10] を用いた．

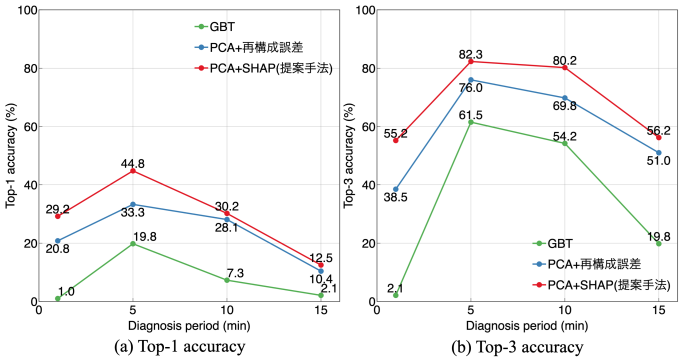
ベースラインとする手法 ベースラインとする手法として，まず，提案手法が原因診断に PCA を採用することの有用性を評価するために Gaussian Based Thresholding (GBT) を用いた手法 [23] を用いる．GBT を用いた手法は，メトリックの系列ごとの正常時の平均値と異常時の平均値の差分を異常への貢献度として原因診断を行う．次に，提案手法が原因診断にシャープレイ値を採用することの有用性を評価するために，PCA の再構成誤差を用いた手法 [24] を用いる．PCA の再構成誤差を用いた手法は，PCA が出力するメトリックごとの予測値と実測値との誤差である再構成誤差を異常への貢献度として原因診断を行う．

評価指標 提案手法の原因診断の精度を評価するために Top-k 精度を用いた．Top-k 精度は，原因となるメトリックの系列が上位 k 番目までにある確率である．k の値には，関連研究 [5, 10, 11] の実験で共通して採用されている k=1 および k=3 を用いた．

著者らが管理する GitHub リポジトリ[*11] 内に実験に利用したソースコードおよびデータセットを公開している．データセットを生成するためのシステムを同様に GitHub リポジトリ[*12] に公開している．

4.3 診断精度の評価

提案手法の原因診断の精度を評価するために，Sock Shop を構成するコンテナのうち 8 個のコンテナに CPU 過負荷およびメモリリークの異常を再現する実験をそれぞれ 6 回ずつ行った計 96 個のデータに対して，Top-1 精度および Top-3 精度を評価した．3.1 節で述べたように，診断のためには取得した 30 分のメトリックから診断期間を設定する必要がある．本実験においては，診断期間を 1，5，10，15 分で変化させて精度の評価を行った．例えば，診断期間が 1 分の場合，取得した 30 分のメトリックのうち，時間が古い方から 29 分間のメトリックを学習データとしてモデルに入力し，残りの 1 分間のメトリックをモデルによる異常度の計算および SHAP による貢献度の計算に用いる．



(a) Top-1 accuracy　(b) Top-3 accuracy

図 2: 診断期間に対する Top-k 精度の変化

図 2 は，診断期間を変化させた際の Top-1 精度および Top-3 精度の評価結果である．Top-1 精度および Top-3 精度は，提案手法である PCA と SHAP を用いた手法が最も精度が高く，次に PCA と再構成誤差を用いた手法，最後に GBT となった．GBT では，メトリックごとに平均値の変化の大きさを見る手法であるため，メトリックの系列間の相関を取り扱わない．一方，PCA を用いることで，メトリックの系列間の相関を考慮した診断が可能となる．実験環境におけるコンテナ同士は，相互のネットワーク通信や，ホストとなるノードの共有のため，メトリックの系列間に相関を持つ．そのため，メトリックの系列間の相関を考慮できる PCA を用いた手法は GBT に比べて診断の精度が高いと考えられる．また，PCA と再構成誤差を用いた手法では，異常の原因となるメトリックの系列が変動した場合，正常時に相関が高かったメトリックの再構成誤差も大きくなり，異常への貢献度が高く見積もられることがある．SHAP を用いた手法では，2.2 節で述べたシャープレイ値の性質により，正常時の相関の高さによる影響を排除して，個々のメトリックの貢献度を計算できる．これにより，提案手法である SHAP を用いた手法が最も高い診断精度となったと考えられる．

同一の手法内での診断精度は，診断期間が 5 分の場合が最も精度が高い結果となり，提案手法では，Top-1 精度が 44.8 %，Top-3 精度が 82.3 %であった．診断精度は，診断期間が 5 分から短くなる場合や，長くなる場合に低下することがわかった．これは，今回の実験では，異常を再現してから 5 分後にメトリックを取得しているため，診断期間を 5 分とした場合が，各系列を最もよく時間の軸に対して正常時と異常時に分割できるためである．診断期間が 5 分よりも短くなった場合，機械学習モデルに与える学習データに異常時のデータが含まれるため，モデルの学習が

[*10] ShapPack: https://github.com/tsurubee/shappack
[*11] https://github.com/ai4sre/shap-anomaly-diagnosis
[*12] https://github.com/ai4sre/meltria

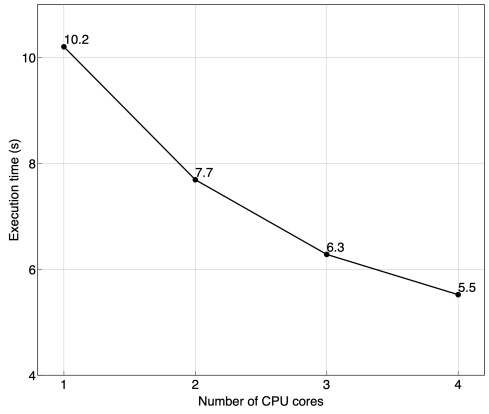


図 3: CPU コア数に対する診断時間の変化

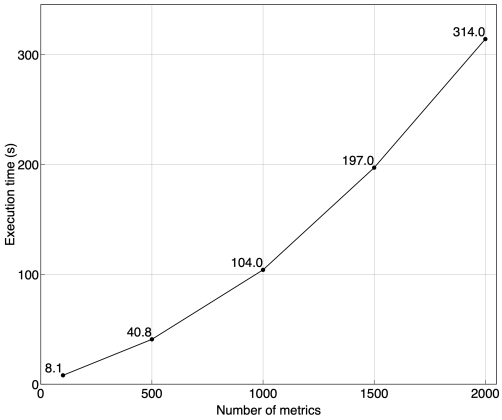


図 4: メトリックの系列数に対する診断時間の変化

表 3: CPU コア数を 4 に設定したときの診断時間の内訳

測定項目	実行時間/s
PCA の学習	0.002
SHAP による貢献度の計算	5.52
診断時間	5.52

うまく行えずに精度が低下する．一方，診断期間が 5 分よりも長くなった場合，モデルは正常時のデータのみで学習できるものの，異常への貢献度の計算に正常時のデータが含まれることにより精度が低下する．これらの結果から，診断期間の設定が診断の精度に大きく影響を与えることがわかった．実際のシステムにおける異常に対する適切な診断期間は，異常ごとに異なるため，異常時に取得したメトリックから適応的に診断期間を設定する必要がある．この点については今後の課題とする．

4.4 診断時間の評価

本節では，提案手法が異常検知後に動作を開始してから診断結果を返すまでの診断時間を評価する．実験では，診断期間を 5 分に固定して，CPU コア数およびメトリックの系列数に対する提案手法の診断時間を評価した．

図 3 は，計算に用いる CPU コア数を変化させたときの診断時間を示している．メトリックの系列数は，前節の診断精度の実験時と同じ 66 で固定した．図 3 より，CPU コア数が増加するにつれて提案手法の診断時間が短くなった．また，CPU コア数を 4 に固定した際の診断時間の内訳を表 3 に示す．診断時間には，主に PCA の学習時間と SHAP による貢献度の計算時間が含まれる．表 3 より，PCA の学習は 2 ms であり，診断時間に対しての影響が無視できるほど小さい．したがって，提案手法の診断時間は，SHAP による貢献度の計算に要する時間とほぼ同等であると言える．

システムの構成要素数の増大に伴い，診断に用いるメトリックの系列数が増大する場合に，提案手法の診断時間が実用に耐えうるかを確認するために，メトリックの系列数に対する提案手法の診断時間を評価した．メトリックの系列数を増加させたデータを作成するために，既存のコンテナのメトリックを複製し，コンテナ数を擬似的に増加させた．CPU コア数は 4 で固定した上で，メトリックの系列数を増加させたときの提案手法の診断時間を図 4 に示す．図 4 より，メトリックの系列数の増加に伴い，診断時間が増加した．その診断時間は，メトリックの系列数が 1500 のときに 197 s，系列数が 2000 のときに 314 s であった．

診断の結果は，システム管理者が異常からシステムを復旧させる作業に活用することを想定しているため，診断時間は短いことが望まれる．一方，システム管理者が異常検知の通知に気づき，手動で復旧の作業を行う時間を考慮すると，診断時間を秒単位で完了させる必要はない．また，Google Cloud[*13]がサービスの品質保証に関して利用者との間で合意した SLA (Service Level Agreement) [25] によると，5 分以内であればサービスの停止期間とみなしていない．これらを踏まえると，診断時間は 5 分以内が理想である．本実験では，各コンテナから 6 個のメトリックの系列を取得しているため，メトリックの系列数が 1500 の場合は 250 個のコンテナ，2000 の場合は 333 個のコンテナで構成されるシステムに相当する．図 4 より，333 個のコンテナで構成されるシステムの場合，診断時間は 5 分を超える．コンテナ数が 200 程度のシステムであれば，診断時間は 3 分以内であることがわかった．したがって，コンテナ数が約 300 より大きいシステムに対応した原因診断を行うためには，SHAP による貢献度の計算を高速化する必要がある．この点については今後の課題とする．

5. まとめと今後の展望

本論文では，異常の検知を起点に軽量な機械学習モデルの学習を行い，シャープレイ値を用いてモデルから得られる異常度に対する各メトリックの貢献度を算出することで，異常の原因診断を行う手法を提案する．提案手法では，

[*13] Google Cloud: `https://cloud.google.com/`

モデルの学習を事前に行わずに異常検知後に行うため，システムの構成を頻繁に変更する場合でも常に現状の構成を反映した診断ができる．マイクロサービスのテストベッド環境にて異常を再現する実験の結果，診断期間の設定が適切に行えた場合，提案手法は異常の根本原因のメトリックの系列を 44.8%の精度で上位 1 位に，82.3%の精度で上位 3 位以内に特定できることを確認した．また，診断時間の評価結果から，提案手法はコンテナ数が 200 程度のシステムであれば，3 分以内に診断結果を返すことができることを示した．

提案手法の診断精度は，診断期間の設定に大きく依存することが実験から明らかになったため，今後は，異常検知後に取得したメトリックから適応的に診断期間を設定する手法を検討する．具体的には，変化点検出を用いて異常の開始時間を推定することで診断期間を自動で設定する方法を検討をする．また，提案手法をより構成要素数が大きい規模のシステムに対応させるために，SHAP による貢献度の計算を高速化する手法を検討する．具体的には，構成要素間のネットワーク通信の関係を事前知識として活用することで SHAP による貢献度の計算を高速化できないか検討する．さらに，2.1 節で挙げた因果グラフを用いるアプローチと提案手法の診断精度の比較評価を行い，異常の伝搬経路を特定せずに原因となるメトリックの特定のみに目的を絞ることで，診断精度を高められるかを確認する．

参考文献

[1] S. Newman, “Building Microservices”, O’Reilly Media, Inc., 2021.

[2] Puppet Inc., State of DevOps Report 2021, 2021, https://puppet.com/resources/report/2021-state-of-devops-report/.

[3] G. Aceto, A. Botta, D. De, P. Walter and A. Pescapè, “Cloud Monitoring: A Survey”,Computer Networks, vol. 57, no. 9, pp.2093-2115, 2013.

[4] 坪内佑樹, 脇坂朝人, 濱田健, 松木雅幸, 小林隆浩, 阿部博, 松本亮介, “HeteroTSDB: 異種分散 KVS 間の自動階層化による高性能な時系列データベース”, 情報処理学会論文誌, vol. 62, no. 3, 2021, pp. 818-828.

[5] L. Wu, J. Bogatinovski, S. Nedelkoski, J. Tordsson and O. Kao, “Performance Diagnosis in Cloud Microservices using Deep Learning”, in *International Conference on Service-Oriented Computing*, May 2020, pp. 85-96.

[6] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong and Q. Zhang, “Multivariate Time-series Anomaly Detection via Graph Attention Network”, arXiv:2009.02040 2020.

[7] A. Adadi and M. Berrada, “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)”, *IEEE access*, vol. 6, pp. 52138-52160, Sep. 2018.

[8] L. S. Shapley, “A Value for N-person Games”, in *Contributions to the Theory of Games*, Princeton University Press, 1953, pp. 307-317.

[9] L. Antwarg, R. M. Miller, B. Shapira and L. Rokach, “Explaining Anomalies Detected by Autoencoders Using SHAP”, arXiv:1903.02407 2020.

[10] N. Takeishi and Y. Kawahara, “On Anomaly Interpretation via Shapley Values”, arXiv:2004.04464 2020.

[11] N. Takeishi, “Shapley Values of Reconstruction Errors of PCA for Explaining Anomaly Detection”, in *2019 International Conference on Data Mining Workshops (ICDMW)*, Nov. 2019, pp. 793-798.

[12] E. Strumbelj and I. Kononenko, “Explaining Prediction Models and Individual Predictions with Feature Contributions”, *Knowledge and information systems*, vol. 41, pp. 647665, Aug. 2014.

[13] S. M. Lundberg and S. I Lee, “A Unified Approach to Interpreting Model Predictions”, in *Proceedings of the 31st international conference on neural information processing systems*, Dec. 2017, pp. 4768-4777.

[14] B. Beyer, C. Jones, J. Petoff and N. R. Murphy, “Site Reliability Engineering: How Google Runs Production Systems”, O’Reilly Media, Inc., 2016.

[15] Y. Meng, S. Zhang, Y. Sun, R. Zhang, Z. Hu, Y. Zhang, C. Jia, Z. Wang and D. Pei, “Localizing Failure Root Causes in a Microservice through Causality Inference”, in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, June 2020, pp. 1-10.

[16] M. Ma, P. Wang, J. Xu, Y. Wang, P. Chen and Z. Zhang, “AutoMAP: Diagnose Your Microservice-based Web Applications Automatically”, in *Proceedings of The Web Conference 2020*, Apr. 2020, pp. 246-258.

[17] J. Qiu, Q. Du, K. Yin, S. L. Zhang and C. Qian, “A Causality Mining and Knowledge Graph Based Method of Root Cause Diagnosis for Performance Anomaly in Cloud Applications”, *Applied Sciences*, vol. 10, no. 6, pp. 2166, Mar. 2020.

[18] P. Spirtes and C. Glymour, “An Algorithm for Fast Recovery of Sparse Causal Graphs”, *Social Science Computer Review*, vol. 9, no. 1, pp. 62-72, 1991.

[19] C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Ré and M. Zaharia, “DAWNBench: An End-to-End Deep Learning Benchmark and Competition”, in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

[20] J. Camacho, A. Pérez-Villegas, P. García-Teodoro and G. Maciá-Fernández, “PCA-based Multivariate Statistical Network Monitoring for Anomaly Detection”, *Computers & Security*, vol. 59, pp. 118137, June 2016.

[21] Q. Ding and E. D. Kolaczyk, “A Compressed PCA Subspace Method for Anomaly Detection in High-Dimensional Data”, *IEEE Transactions on Information Theory*, vol. 59, pp. 74197433, Nov. 2013.

[22] 坪内佑樹, 鶴田博文, 古川雅大, “TSifter:マイクロサービスにおける性能異常の迅速な診断に向いた時系列データの次元削減手法”, インターネットと運用技術シンポジウム論文集, 2020, pp. 9-16.

[23] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low and M. C. Chan, “GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection”, in *2019 IEEE Conference on Communications and Network Security (CNS)*, June. 2019, pp. 91-99.

[24] B. Mnassri and M. Ouladsine, “Reconstruction-based Contribution Approaches for Improved Fault Diagnosis Using Principal Component Analysis”, *Journal of Process Control*, vol. 33, pp. 6076, June 2015.

[25] Google LLC, Compute Engine Service Level Agreement (SLA), 2015, https://cloud.google.com/compute/sla-20151016/.