부하테스트를 활용한 클라우드 운영 환경의 이상탐지 알고리즘 성능 분석

김진희*, 이찬재*, 윤호영*
*오케스트로(주)
jh.kim@okestro.com, cjlee@okestro.com, hy.yun@okestro.com

Anomaly Detection Algorithm Performance Analysis of Cloud Operating Environment using Stress Test

Jin Hui Kim*, Chan Jae Lee *, Ho Young Yun *
jh.kim@okestro.com*, cjlee@okestro.com, hy.yun@okestro.com
*Okestro Ltd.

요 약

안정적인 서버 운영을 위해 이상 패턴 및 개체를 식별하는 이상탐지 연구가 활발하게 연구되어 오고 있다. 이상탐지의 대표적인 예로 서버의 사용량 증가를 꼽을 수 있지만, 실제 이상 데이터 수집 및 현상의 재현이 어렵다는 점은 해당 연구의 어려움으로 존재한다. 본 연구는 다양한 시나리오 기반의 부하테스트를 설계하고, 클라우드 환경에서 이상 데이터를 생성 및 수집하였다. 해당 데이터는 이상탐지에 대표적으로 사용되는 알고리즘의 성능을 비교 분석에 활용하였으며, 실험을 통해 각 알고리즘의 신뢰 수준을 확인하였다. 이는 다양한 서버 운영 환경에 적합한 알고리즘을 채택하는데 활용 가능하며, 결과적으로 안정적이고 효율적인 서버 운영에 기여할 수 있을 것으로 사료된다.

1. 서론

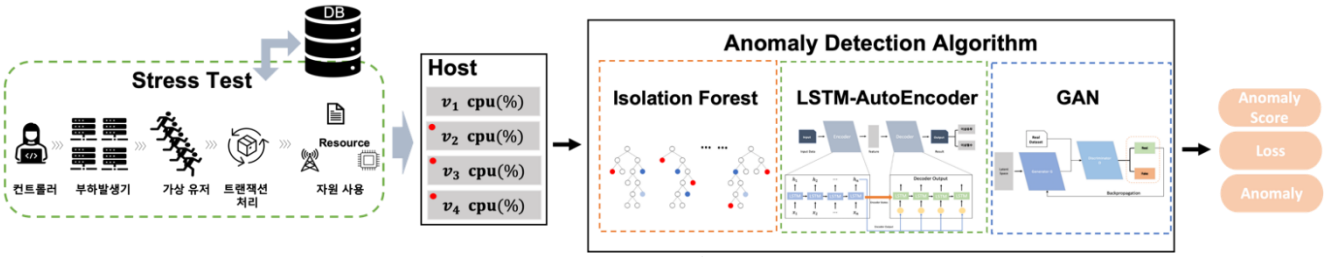
최근 빅데이터, 인공지능 기술 등이 확산됨에 따라 데이터 센터 운영의 안정성이 중요시되고 있다. 안정적인 데이터 센터 운영을 위한 방법으로 서버의 장애를 사전에 예측하여, 기민한 장애 대응을 목적으로 하는 이상탐지(Anomaly Detection) 기법이 많은 관심을 받고 있다. 이상탐지는 과거의 데이터를 기반으로 데이터 패턴 및 개체를 인식하여 이상 유무를 관찰한다. 이러한 이상탐지 기법은 서버 운영 환경의 분야, 범위, 규모에 따라 데이터 패턴이 다르며, 이상데이터가 발생한 명확한 원인 규명이 어렵고 실제 데이터를 확보하는 것이 현실적으로 어려운 한계점이 존재한다.

이에 본 연구팀은 이상탐지를 위해 클라우드 환경으로 운영되고 있는 자체 물리 서버의 가상머신을 대상으로 CPU 사용량 데이터를 수집하였다. 또한 다양한 서버 운영 환경을 반영한 시나리오를 설계하여 부하테스트를 통해 다양한 데이터 셋을 추가적으로 제공한다. 알고리즘의 성능 분석은 이상탐지 기법에 대표적으로 사용되는 알고리즘 중 비지도 학습 기반의 알고리즘을 대상으로 하였다.

2. 관련 연구

최근 이상탐지 기법은 데이터의 유형에 따라 지도 학습, 비지도 학습 방법으로 나눌 수 있다. 지도 학습의 경우 데이터 셋에 라벨에 대한 정보가 존재할 때 사용하는 학습 방법으로 목표변수의 이상 확률을 추정하는 기법이다. 본 연구에서 고려한 비지도 학습은 데이터 셋의 라벨이 따로 주어지지 않으며, 거리 기반의 추정 척도를 계산하여 이상 개체를 인지하는 기법으로 머신러닝과 딥 러닝을 예로 들 수 있다[1].

이상탐지에 사용되는 대표적인 비지도 학습 기법으로 Regression Decision Tree 기반의 Isolation forest[2] 기법이 존재하는데, 이는 정상 데이터일 때 비정상 데이터보다 더 많은 재귀 이진분할이 필요하다는데 착안하여 트리의 깊이가 짧을 수록 이상 데이터라고 판별한다. 또 다른 방법으로 Outlier Detection Datasets (ODDS)에서 제공하는 벤치마크 데이터 셋을 Encoder, Decoder LSTM 아키텍처를 적용하여 구현한 LSTM-AutoEncoder[3] 기법이 있다. 해당 기법은 정상 데이터를 학습 시킨 뒤, 비정상 데이터를 입력할 경우 학습한 정상 데이터와 차이점을 계산하여 이상탐지한다. 딥러닝을 이용한 이상탐지 기법에는 GAN 알고리즘이



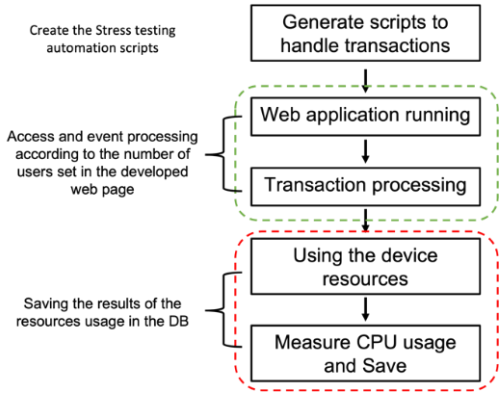
(그림 1) 부하테스트 및 이상징후 판별 알고리즘 흐름도.

있으며, 최근 MNIST, CIFAR-10 등의 이미지 데이터 셋을 기반으로 연구되고 있다[4]. GAN 은 정상 데이터에 대한 분포를 학습한 후 정상/비정상 데이터를 입력하여 출력된 점수를 통해 이상 여부를 판단한다. 본 연구에서는 Isolation Forest, LSTM-AutoEncoder, GAN[5]을 통해 알고리즘 성능을 비교하였다.

3. 부하테스트 및 이상징후 탐지

본 연구에서 대상으로 한 이상 데이터는 물리 서버를 통한 클라우드 환경에서의 네 개의 가상머신 $V(v_1, v_2, v_3, v_4)$을 대상으로 생성하였다. V는 동일한 환경구성으로 설정하였으며, 부하를 주지 않은 가상머신 v_1과 부하를 준 가상머신 v_2, v_3, v_4로 나누어 데이터 셋을 수집하였다. V 에서 수집된 사용량 데이터 집합 $R = \{r_{timestamp}, r_{value}\}$은 timestamp 와 CPU 값의 집합으로 이루어져 있다. 수집된 데이터 셋을 통해 알고리즘 별 이상탐지 정확도를 다양한 평가지표를 통해 성능 비교하였고 전체적인 과정은 그림 1 과 같다. 본 장에서는 부하를 통해 의미 있는 수치를 측정하기 위한 테스트 환경 설정 및 시나리오와 각각의 알고리즘에 대해 설명하였다.

3. 1 부하 테스트



(그림 2) 부하테스트 흐름도.

3.1.1. Jmeter

본 연구에서는 부하를 주기 위한 실험 도구로 Apache Jmeter 를 사용하였다. Jmeter 는 클라이언트-서버 구조로 된 소프트웨어의 성능 테스트 목적으로 만들어진 도구이다. 기본적인 동작으로는 그림 2 와 같이 컨트롤러에 의해 가상 유저를 생성 및 작동 시간을 설정하고, 웹 애플리케이션을 통해 트랜잭션을 수행하며, 트랜잭션을 처리함으로써 자원을 사용하여 CPU 사용량을 늘리게 된다. 다음으로 이를 실험하기 위한 웹 스크립트 작성과 시나리오에 대해 설명한다.

3.1.2 웹 스크립트 및 부하 시나리오

가상 유저가 웹 페이지를 접속하고 이후 발생할 사건들에 대해 자원이 사용되도록 추가적인 스크립트를 작성해야 한다. 본 연구에서는 하나의 웹 페이지를 Java, PHP, HTML5 등을 이용하여 구축하였고, 로그인 및 회원 정보를 수정하는 스크립트를 작성하였다. 이는 로그인 진행과 동시에 DB 에 저장된 정보를 조회하는 것과 회원 정보 수정을 통해 업데이트를 실행함으로써 자원을 사용하기 위한 설정이다.

본 연구에서 설정한 시나리오는 직장인의 생활 패턴을 기준으로 구성하였으며, v_2 의 경우 평일 출근 및 퇴근 시간의 음악, 지도 등의 사용환경을 고려하여 시나리오를 설계하였다. v_3의 경우 평일 점심시간 및 퇴근 시간과 주말을 제외한 평일 업무 시간의 사용률을 나타내기 위한 시나리오를 설계하였다. v_4 의 경우 주말의 쇼핑 및 여가 생활을 기준으로 사용률을 측정하기 위한 시나리오를 설계하였다.

3. 2 이상징후 판별 알고리즘

3.2.1 Isolation forest

3 절에서 언급한 데이터 집합 R은 정상, 비정상 데이터로 이루어지고, 비정상 데이터에서도 각기 다른 value 값을 가지며, 다양한 특징을 보유한다고 판단하여 해당 알고리즘을 채택하였다. R 에서 훈련 및 테스트 데이터 셋으로 구분하여 각각의 인스턴스에 대한 Isolation Tree 를 구성하여 이상 점수 결과를 얻는다.

3.2.2 LSTM-AutoEncoder

정상 데이터 대비 이상치가 적은 특징 및 시간적 특징을 고려하여 LSTM-AutoEncoder[6] 알고리즘을 사용하였다. Encoder 및 Decoder 는 LSTM 레이어로 구성되어 있으며, 학습하는 과정에서는 정상데이터만을 사용하여 학습한다. 최대가능도방법을 이용하여 이상점수를 구하고, Threshold 를 지정해 상회할 시 비정상이라 판단한다.

3.2.3 GAN

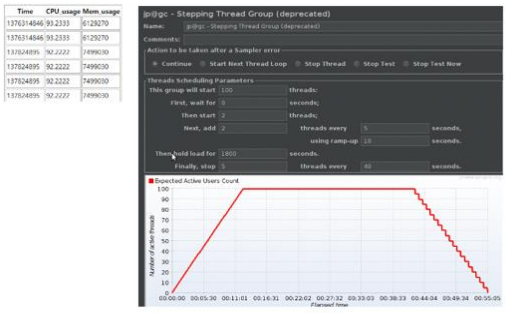
R을 네트워크 입력 값으로 사용하기 위해 시퀀스

리스트로 변환하는 전처리를 진행한다. 또한 벡터가 존재하는 공간인 임베딩 공간(Latent Space, 이하 LS)을 통해 시퀀스 리스트 데이터를 벡터로 받아 정규화를 진행하여 데이터를 생성한다. LS 에서 무작위로 추출한 벡터 값을 Stack LSTM 네트워크 구조로 이루어진 생성자에게 보내주어 가짜 데이터를 생성한다. 구분자의 경우 생성자를 통해 만들어진 가짜 데이터를 판별한 후 실제 데이터를 입력하여 진짜 데이터로 판별한다. 그런 다음 구분자 및 생성자는 역전파를 통해 업데이트를 진행한다. 가짜 데이터와 실제 데이터가 떨어진 정도를 통해 최소화하도록 이 과정을 반복하여 업데이트를 진행한다. 마지막으로 실 데이터를 LS 에 매핑하여 학습 과정을 진행한 다음 최종적인 Loss 값을 Anomaly Score 로 사용하도록 한다.

4. 성능실험

4.1 실험환경

4.1.1 부하테스트 및 데이터 셋



(그림 3) 부하테스트 설정 및 웹 애플리케이션.

3.1.2 장에서 제시한 시나리오를 바탕으로 그림 3 과 같이 CPU 사용량 데이터를 생성하였으며, 부하 시나리오 설정은 표 1 과 같다. 예를 들어, v_2의 경우 직장인 기준 평일 출퇴근시 걸리는 평균 시간 1 시간을 기준으로 잡은 것이며, 이에 대한 부하 발생을 위해 최대 100 명의 접속자를 생성하며 10 초마다 2 명씩 증가하였고, 100 명의 접속자가 모두 접속하면 1800 초(30 분)를 유지하였다. 그리고 40 초 간격으로 5 명의 유저 수를 제거하는 방식으로 부하를 주어 시나리오에 맞는 테스트를 진행한다. v_3의 경우 출근 시점부터 점심시간 전까지 부하를 주었고, 점심시간 이후부터 퇴근시간까지 부하를 주기 위해 v_2보다 유저 수 및 유지 시간 수를 늘리기 위해 표와 같이 설정하였다. v_4의 경우 주말만 부하테스트를 진행한 방법이며, 점심 및 저녁시간의 차이를 줄여 부하를 진행하였다. 이는 시간에 따른 부하 데이터 비율에 따른 학습 정도와 결과에 미치는 영향을 생각하여 설정한 값이다. 이처럼 사용량 데이터와 부하테스트를 통해 얻은 데이터 셋이 입력 데이터가 되며, 데이터 형태는 Timestamp(2021-09-30 00:00:00), Value(0.12, 0.56..) 쌍으로 구성되어 있다.

<표 1> 부하 시나리오 설정

	최대 유저 수 / 최대 유지 시간	유저 증가/ 제거 수	증가/제거 시간
v_1	부하를 주지 않은 실제 가상머신		
v_2	100 명/ 1800 초	2 명/5 명	10 초/40 초
v_3	200 명/10800 초	2 명/5 명	50 초/100 초
v_4	200 명/10800 초	2 명/5 명	80 초/100 초

4.1.2 Isolation forest

실험에 사용한 Isolation forest 는 125 개의 노드 수를 생성하도록 하였으며, outlier 의 경우 자동으로 색출하게 하고, 이상치 데이터의 경우 -1 값으로 나타내어 정상 데이터와 구분하였다.

4.1.3 LSTM-AutoEncoder

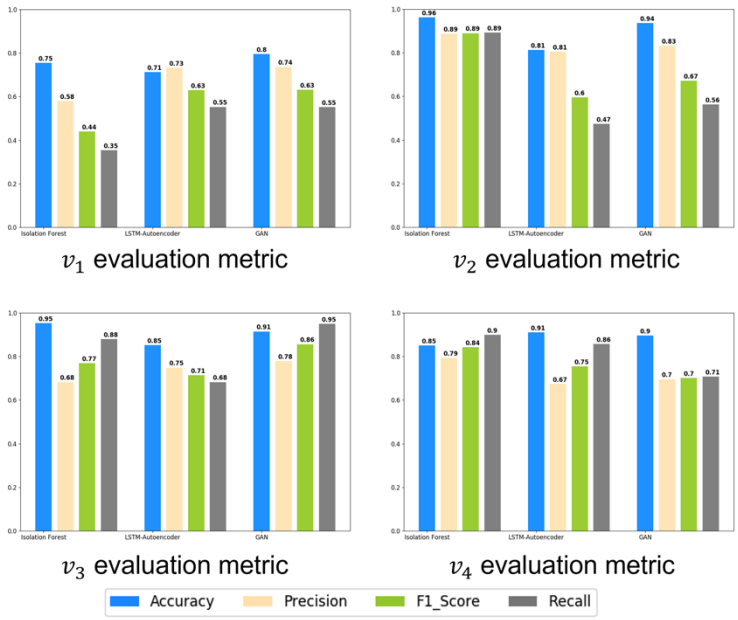
Encoder 및 Decoder 의 입력으로 n개의 연속한 벡터를 사용한다. 입력, 출력, 히든 차원은 1 로 설정하였으며, 10000 번의 반복을 통해 훈련하였다. 최적화는 Adam 을 사용하고, Threshold 의 경우 0.08 로 지정하였으며, 모델 훈련시 MSE 방법을 통해 각 시퀀스에 대한 오차를 모두 합하여 이를 총 손실로 사용하였다.

4.1.4 GAN

해당 알고리즘의 입력 데이터로 사용하기위한 전처리로 Window size 는 60, Sliding 은 1 로 설정하여 시퀀스 리스트로 변환하였다. 생성자의 경우 Hidden unit 은 32, 64, 128 로 Stack LSTM 구조를 사용하였고, 구분자의 경우 1 개의 LSTM 레이어로 구성하고, Hidden unit 은 100 으로 설정하였다. 또한 Threshold 는 1.0 으로 설정하여 이상징후 판별을 진행하였다.

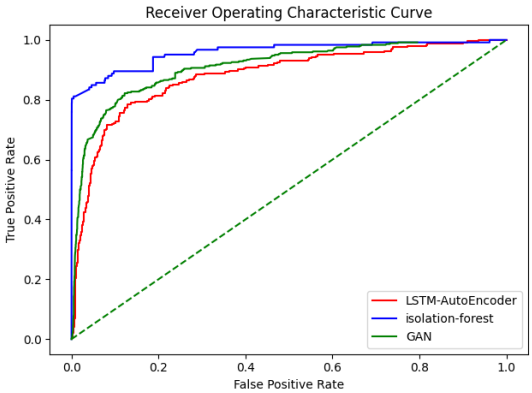
4.2 성능평가

각 알고리즘을 4 가지 성능평가 지표(Accuracy, Precision, F1-Score, Recall)를 기준으로 각 시나리오 별로 성능을 평가를 실시하였다. 그림 4 에서와 같이 v_1의 경우 GAN 의 Accuracy 가 0.8 로 가장 좋은 성능을 보였고, LSTM-AutoEncoder 와 Isolation forest 가 각각 0.71 과 0.75 의 성능을 보였다. 부하를 발생시킨 v_2, v_3, v_4에서는 모든 평가지표가 향상된 결과를 확인할 수 있었고, v_2, v_3의 경우 다른 가상머신보다 이상 데이터의 빈도 수가 일정하며, v_4의 경우 급격한 이상 데이터 발생에 있어서 v_2, v_3보다 정확도가 낮은 걸 확인할 수 있다. 또한 부하를 주지 않은 v_1의 경우 급격한 CPU 사용률이 탐지되지 않았으며, 0 에 가깝게 분포되어 있어 v_2, v_3, v_4 보다 이상탐지의 정확도가 낮은 걸로 판단된다. 일반적인 경우 Isolation forest 와 GAN 이 높은 성능을 보였으며, 급격한 이상 데이터가 드물게 나타나는 경우는 Isolation forest 보다 GAN 이 가장 성능이 높음을 알 수 있다.



(그림 4) 가상머신 별 알고리즘 평가지표.

그림 5는 도메인과 상관없이 좋은 성능을 나타내는 기법을 비교하고자 각 시나리오 v_1, v_2, v_3, v_4의 누적합을 이용하여 데이터 간의 정량적 평가를 위한 수신자 조작 특성(Receiver operating characteristic)을 이용하여 나타냈다. Isolation forest 가 가장 좋은 성능을 보였으며, GAN 과 LSTM-Autoencoder 는 비슷한 성능을 보였다. 빈번한 사용률 증가가 있는 경우, 특징점을 토대로 이상을 탐지하는 Isolation forest 의 성능이 가장 좋았으며, 시간의 특징과 함께 이상탐지를 진행하는 LSTM 구조의 기법은 드물게 사용량 증가가 있는 환경에 접목시킬 수 있는 것을 확인하였다.



(그림 5) 알고리즘 별 ROC.

5. 결론

본 연구에서는 자체 보유한 물리 서버내 가상머신의 CPU 사용량 데이터를 기반으로 여러 유형의 데이터 셋을 추가적으로 생성하여, 이상탐지 기법을 비교 분석하였다. 이는 이상 데이터 수집이 어려운 한계를 보완했을 뿐만 아니라 안정적인 서버 운영 환경에 기여할 수 있을 것이라고 사료된다. 향후 CPU 사용량 뿐만 아니라 Memory, Disk 등 다양한 자원의 상관관계를 고려하여 이상탐지 기법의 성능을 향상시키고자 한다.

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2021-0-00256, 클라우드 자원의 지능적 관리를 위한 이종 가상화(VM+Container) 통합 운용 기술 개발)

참고문헌

[1] Munir, Mohsin, et al. "DeepAnT: A deep learning approach for unsupervised anomaly detection in time series.", IEEE Access, Vol. 7, 2018

[2] F. T. Liu, K. M. Ting, and Z. H. Zhou, “Isolation forest”, in Proceedings - IEEE International Conference on Data Mining, ICDM, pp. 413– 422, 2008.

[3] Amarbayasgalan, Tsatsral, Bilguun Jargalsaikhan, and Keun Ho Ryu. "Unsupervised novelty detection using deep autoencoders with density based clustering." _Applied Sciences, vol. 8, 9, pp. 1468, 2018.

[4] Li, Dan, et al. "Anomaly detection with generative adversarial networks for multivariate time series." arXiv preprint arXiv:1809.04758.2018.

[5] Geiger, Alexander, et al. "TadGAN: Time series anomaly detection using generative adversarial networks." _2020 IEEE International Conference on Big Data (Big Data). IEEE, 2020.

[6] Malhotra, Pankaj, et al. "LSTM-based encoder-decoder for multi-sensor anomaly detection." arXiv preprint arXiv:1607.00148.2016.