EPJ Web of Conferences **245**, 07061 (2020) https://doi.org/10.1051/epjconf/202024507061
CHEP 2019

Machine Learning-based Anomaly Detection of Ganglia Monitoring Data in HEP Data Center

Juan Chen[1,2,*], *Lu* Wang[1,**], and *Qingbao* Hu[1,***]

[1]Institute of High Energy Physics, Chinese Academy of Sciences, 100049, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China

Abstract. This paper introduces a generic and scalable anomaly detection framework. Anomaly detection can improve operation and maintenance efficiency and assure experiments can be carried out effectively. The framework facilitates common tasks such as data sample building, retagging and visualization, deviation measurement and performance measurement for machine learning-based anomaly detection methods. The samples we used are sourced from Ganglia monitoring data. There are several anomaly detection methods to handle spatial and temporal anomalies within the framework. Finally, we show the rudimental application of the framework on Lustre distributed file systems in daily operation and maintenance.

1 Introduction

At present, the Institute of High Energy Physics (IHEP) local cluster consists of 20,000 CPU slots, hundreds of data servers, 20 PB disk storage and 10 PB tape storage. After data taking from the Jiangmen Underground Neutrino Observatory (JUNO) and the Large High Altitude Air Shower Observatory (LHAASO) [1] experiment, the data volume processed at this center will approach 10 PB per year. Prompt anomaly detection can improve operation and maintenance efficiency and assure high energy physics experiments can be carried out effectively. We develop a generic anomaly detection framework based on machine learning.

Anomalies are data points which are either different from the majority of others or different from the expectation of a reliable prediction model in a time series. For Ganglia monitoring metric data, we classify anomalies into spatial anomalies and temporal anomalies. Spatial anomalies are points of high-dimensional data without time dimension. For temporal anomalies, they may not be spatial anomalies, but they are quite different from the current sequence data by analyzing temporal characteristics.

We have broken down some machine learning-based methods into four broad categories. First, based on a classification [2] , anomalies can be detected by trained models with the help of supervised learning algorithms such as xgboost, random forest, etc. Second, based on cluster analysis [3, 4], samples are clustered by density analysis and cutting such as k-means, Isolation Forest [5], etc. The third and fourth categories detect anomalies by analyzing the difference between the real value and the predicted value after predicting. The prediction

*e-mail: chenj@ihep.ac.cn
**e-mail: wanglu@ihep.ac.cn
***e-mail: huqb@ihep.ac.cn

methods of the third category are based on statistics [6] such as Autoregressive Integrated Moving Average model (ARIMA) [7]. The prediction methods of the fourth category are based on deep learning [8] such as Hierarchical Temporal Memory (HTM) [9, 10], etc.

We develop a framework because a particular anomaly detection algorithm is usually applicable to only a special use-case. Unlike methods mentioned above, the anomaly detection framework we developed can detect anomalies by combining the relationship of multiple indicators in addition to using single metric. The framework we developed in Python is suitable to be expanded with statistical machine learning algorithms and deep learning algorithms. It provides some functions such as data sample building, retagging and visualization, deviation measurement and performance measurement for machine learning-based anomaly detection methods.

2 Architecture

As shown in the Figure 1, after collecting data from the Ganglia monitoring system and preprocessing, we detect spatial anomalies and temporal anomalies separately. In addition, we ignore irrelevant anomalies caused by particular known circumstances such as system upgrading by setting time intervals and nodenames in anomaly filter modules.

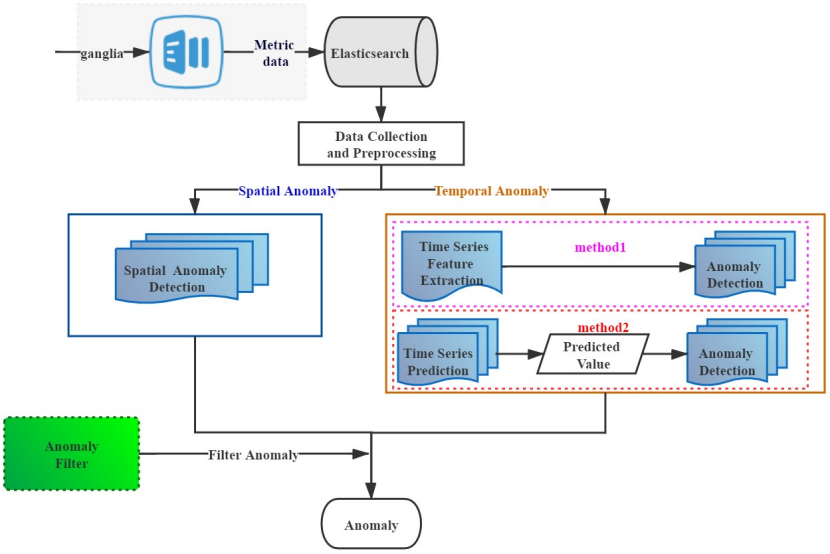

Figure 1. Anomaly detection process

We develop the framework based on Django [11], which is based on a MTV (model-template-view) architecture. There are over twenty general metrics and about one hundred special metrics of the monitoring data, which is collected by the Ganglia monitoring system at IHEP. The timestamped monitoring data are stored in ElasticSearch [12]. We develop a CRUD (create retrieve, update, and delete) interface based on Python ElasticSearch interface in the data interface layer. Configuration such as a list of metrics and model information is stored in MySQL and we deal with MySQL interaction based on Django ORM (Object Relational Mapping). The template layer is a collection of HTML pages which correspond to common functions of anomaly detection tasks. The visualization presents the results of prediction and detection more clearly in line diagrams and scatterplots.

3 Algorithms

We have two classes of anomaly detection methods for temporal anomalies. The first one is detecting after extracting time series features. The other one is detecting anomalies by judging the deviation of predicted data and true data.

3.1 Time-series features extraction

We maintain data of W time steps as time series x. There are W data points per metric in the time series. We extract statistical features and fitting features for every time series.

Statistical features consist of some general statistical features, skewness, kurtosis, volatility indicator and statistical features about repeating data. General statistical features consist of maximum value, minimum value, mean, variance, standard deviation, median, dot product, sum, range, locations and relative locations of maximum and minimum. The volatility indicator measures the volatility of data by computing mean, mean of absolute value, sum of first difference and counting the number of values in x that are lower or higher than the mean of x. Statistical features about repeating data consist of the percentage of recurring values and some general statistical features after removing duplicate values.

We take the last k (k = 6, 12, 18, 24, 30, 36) data points of time series x as time series Y (collections of time series) respectively. Fitting features are these values which are the difference between the last element of time series x and the smoothed values of each time series of Y after Moving Average Algorithm [13], Weighted Moving Average Algorithm[14], Exponential Moving Average Algorithm [15] and Double Exponential Moving Average Algorithm [16]. There are over 30 statistical features after time-series features extraction. The number of fitting features depend on W.

3.2 Prediction algorithms

Statistical approaches consist of Moving Average (MA), Exponential Moving Average (EWMA) and linear regression (LR). We predict the current data by fitting and smoothing historical data based on these algorithms mentioned above.

Long short-term memory (LSTM) [17] is well suited to predict based on time series data. The output and the input of the next sequence can be calculated together to obtain the output of the next sequence. We select some of the historical metric data to predict the current metric data. The number of metrics of the current data is m and the number of metrics of the selected historical data is n. For each sample, the input is shaped as (length of window, n) and the output is shaped as (1, m). The main configuration parameters include the number of LSTM layers, units (the number of hidden neurons), epochs, the number of batch and fraction of data to reserve for validation.

To evaluate the performance, we provide error indicators of different models when fitting the time-series with different metric data. In the framework, the error indicators consist of Mean Error (ME), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Percentage Error (MPE) and Mean Absolute Percentage Error (MAPE).

3.3 Detection algorithms

The algorithm shown in 3.3.1 and 3.3.2 are used after prediction. The Isolation Forest shown in 3.3.3 is used after time-series extraction or for spatial anomalies.

3.3.1 N-sigma

The predicted value at time t is y_t, the true value is x_t. Because some metrics are more volatile, we capture the relative error as Equation 1.

$$e_t = y_t - x_t \tag{1}$$

We can give a threshold directly, but we have to change the threshold manually for different situations. We used the relative errors of metrics (bytes_in, cpu_idle and mem_free) of two metadata servers to draw violin plots respectively and the data set is approximately modeled by a normal distribution. We assume that the relative error of time windows follows a normal distribution and set a threshold by confidence probability.

3.3.2 Q-function

We compute the anomaly score by a Gaussian tail probability. The mean value is μ. We define the anomaly score (s_t) as follows. The closer anomaly score of a sample is to 1, the more likely the anomaly is. The anomaly score ranges from 0.5 to 1.

$$s_t = 1 - Q(t) \tag{2}$$

$$Q(x) = \begin{cases} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt & x < \mu, \\ 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt & x >= \mu, \end{cases} \tag{3}$$

3.3.3 Isolation Forest

For Isolation Forest [18], we subsample randomly first and then build many binary trees based on cutting. In the process of building, we split randomly on features until each data tuple forms a leaf node or the height reaches the limit. For anomalies, they are more easily divided into leaf nodes, so their average path length is shorter than others. c(n) is the average pathlength of trees. The number of samples used for building trees is n. x is an instance and h(x) is the number of edges between the root node and the terminating node plus an adjustment c(T.size). T.size is the number of instances of the terminating node where x is located. We compute anomaly score S(x,n) based on the average path length E(h(x)):

$$s(x, n) = 2^{\frac{E(h(x))}{c(n)}} \tag{4}$$

$$c(n) = \begin{cases} 2H(n-1) - \frac{2(n-1)}{n} & n > 2, \\ 1 & n = 2, \\ 0 & otherwise. \end{cases} \tag{5}$$

$$H(n) = ln(n) + 0.5772156649 \tag{6}$$

For temporal anomaly detection, the features are time-series features. For spatial anomaly detection, the features are metrics of samples. The main parameters include the number of trees (n_estimators), the number of subsamples (max_samples), anomaly ratio (contamination) and whether to extract time-series features.

4 Experience in Lustre file system

4.1 Data

The real metadata of Lustre file system [19] in the IHEP data center is used for the experiments. The metrics used are shown in Table 1. For models that require a lot of historical data to train, we use 8 metadata servers from August 1st, 2019 to September 30th, 2019. The test dataset is one of the metadata servers from October 1st, 2019 to October 25th, 2019. There are almost 140,000 training samples and almost 6000 test samples. The data in the database does not have an anomaly tag, so we cannot compute precision and recall rate accurately at present.

Table 1. Metrics used in the experience

Categories	Metrics	Description
Network	bytes_in	The number of bytes received per second
	bytes_out	The number of bytes emitted per second
	pkts_in	The number of packets received per second
	pkts_out	The number of packets emitted per second
CPU	cpu_idle	Percentage of time the CPU is idle
	load_one	The load average from the last minute
	load_five	The load average from the last five minutes
	load_fifteen	The load average from the last fifteen
Memory	mem_free	Available memory capacity
	mem_buffers	Buffer capacity
	mem_cached	Cache capacity
	swap_free	Available swap capacity
	proc_run	Total number of running processes
	proc_total	Total number of processes

4.2 Prediction Experiments

We compare these algorithms (parameter configuration is shown in Table 2) with RMSE and MAPE. As shown in Figure 2 for network and CPU, the deviation is lower for LSTM. For memory, EWMA performs better. But LSTM can predict all metrics based on one model.

Table 2. Prediction algorithm parameter configuration

Algorithms	Parameter configuration
LR	windows=12*3
MA	windows=12*3
EWMA	windows=12*3,alpha=0.7
LSTM	windows=12*3, n=14, m=14, n_layers=2, units=128, epochs=300, input_shape=(windows,n), output_shape=m, learn_rate=0.001, optimizer=Adam, loss=mae

4.3 Anomaly Detection Experiments

We use three anomaly detection methods to detect anomalies. For temporal anomaly method 1, we use Isolation Forest (n_estimators=100, max_samples=256, contamination=0.0001)

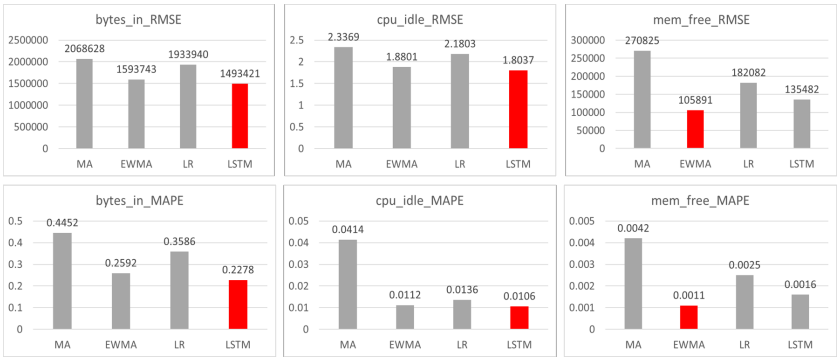

Figure 2. Prediction results. The left side of the figure shows the RMSE and MAPE of bytes_in_value. The middle diagram shows the RMSE and MAPE of cpu_idle_value. The right side of the figure shows the RMSE and MAPE of mem_free_value.

after extracting time-series features. For temporal anomaly method 2, the prediction algorithm is LSTM (parameters configuration as listed in Table 2) and the Q-function is the detection algorithm (st=0.9999). For spatial anomaly method, we use Isolation Forest (n_estimators=100, max_samples=256, contamination=0.0001). Detection results are shown in Figures 3-5.

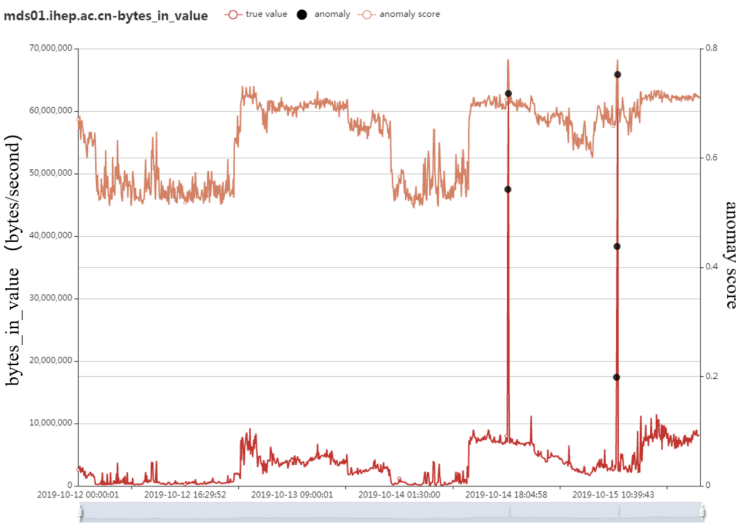

Figure 3. Result of temporal anomaly detection method 1. The red line shows true bytes_in_value of mds01.ihep.ac.cn. The orange line shows the anomaly score. The black points show the anomalies.

5 Conclusion

In this paper, we introduced a generic anomaly detection framework which provides the generic functionality required for anomaly detection tasks such as data sample building, retagging and visualization, deviation measurement and performance measurement. It was

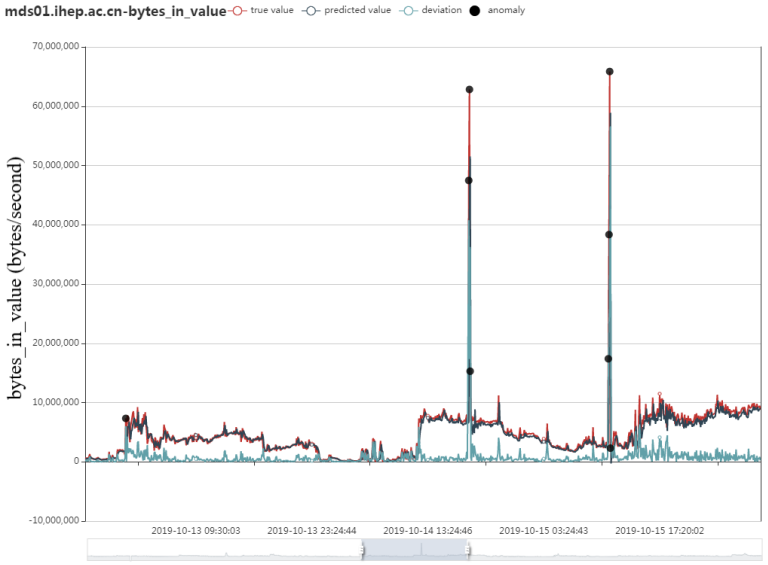

Figure 4. Result of temporal anomaly detection method 2. The red line shows true bytes_in_value of mds01.ihep.ac.cn. The dark blue line shows the predicted value of bytes_in_value by LSTM. The light blue shows the deviation between true value and predicted value. The black points show the anomalies.

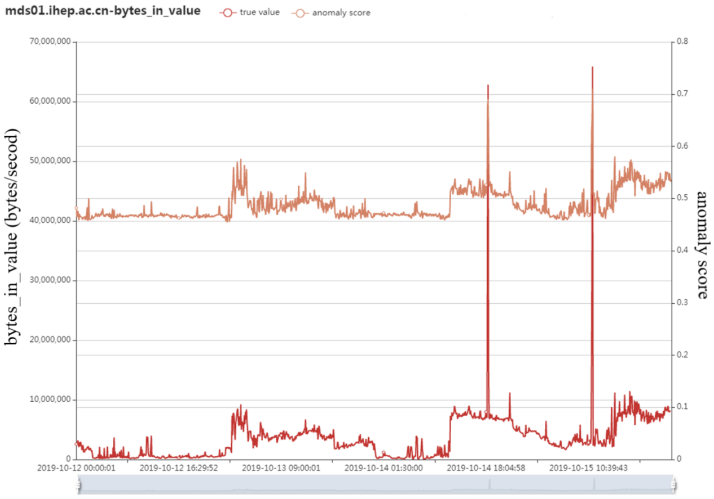

Figure 5. Result of spatial anomaly detection method. The red line shows true bytes_in_value of mds01.ihep.ac.cn. The orange line shows the anomaly score. There is no spatial anomaly detected.

initially applied to the metadata servers of Lustre file system, but it is not yet in production. Furthermore, the framework provides extracted time-series, different prediction models, and anomaly detection algorithms. In the future, we will associate these anomalies with anomalies in the actual environment such as disk failure, traffic abnormality, etc.

This paper was supported by National Natural Science Foundation of China (Project code: 11805226).

References

[1] G. D. Sciascio, LHAASO Collaboration, The lhaaso experiment: from gamma-ray astronomy to cosmic rays[J]. Nuclear and particle physics proceedings, **279**, 166-173 (2016)

[2] I. Steinwart, D. Hush, C. Scovel, A classification framework for anomaly detection[J]. Journal of Machine Learning Research, **6**, 211-32 (2005)

[3] D. Pokrajac, A. Lazarevic, L. J. Latecki, Incremental local outlier detection for data streams[C]//2007 IEEE symposium on computational intelligence and data mining, 504-515 (2007)

[4] S. Budalakoti, A. N. Srivastava, R. Akella, E. Turkov, Anomaly detection in large sets of high-dimensional symbol sequences[J]. (2006)

[5] F. T. Liu, K. M. Ting, Z. H. Zhou, Isolation-based anomaly detection[J]. ACM Transactions on Knowledge Discovery from Data (TKDD),**6**, 1-39 (2012)

[6] M. Markou, S. Singh, Novelty detection: a reviewpart 1: statistical approaches[J]. Signal processing, **83**, 2481-2497 (2003)

[7] G. P. ZHANG, Time series forecasting using a hybrid arima and neural network model[J]. Neurocomputing, **50**, 159-175 (2003)

[8] M. Markou, S. Singh, Novelty detection: a reviewpart 2: neural network based approaches[J]. Signal processing, **83**, 2499-2521 (2003)

[9] Y. CUI, S. AHMAD, J. HAWKINS, Continuous online sequence learning with an unsupervised neural network model[J]. Neural computation, **28**, 2474-2504 (2016)

[10] S. AHMAD, S. PURDY, Real-time anomaly detection for streaming analytics[J]. arXiv preprint arXiv:1607.02480 (2016)

[11] A. Holovaty, J. Kaplan-Moss, *The definitive guide to Django: Web development done right* (Apress, Berkeley, 2009)

[12] R. Kuc, M. Rogozinski, *Elasticsearch server* (Packt Publishing Ltd, Birmingham,2013)

[13] E. Booth, J. Mount, J. H. Viers, Hydrologic variability of the Cosumnes River floodplain[J]. San Francisco Estuary and Watershed Science, **4**(2), (2006)

[14] S. HANSUN, A New Approach of Browns Double Exponential Smoothing Method in Time Series Analysis[J]. Balkan Journal of Electrical and Computer Engineering ,**4**(2), 75-7 (2016)

[15] J. DING, N. MEADE, Forecasting accuracy of stochastic volatility, garch and ewma models under dierent volatility scenarios[J]. Applied Financial Economics, **20**(10), 771-783 (2010)

[16] J. Huang, C. Li, J. Yu, Resource prediction based on double exponential smoothing in cloud computing[C]//In 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), IEEE, 2056-2060 (2012)

[17] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory[J]. Neural Computation, **9**, 1735 (1997)

[18] F. T. LIU, K. M TING, Z. H. ZHOU, Isolation-based anomaly detection[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), **6**(1), 1-39 (2012)

[19] S. LUSTRE, Building a le system for 1000 node clusters[C]//Proc. Of the 2003 Ottawa Linux Symp. Ottawa: RedHat, **407**, (2003)