Unsupervised Anomaly Detection in AIOps Lifecycle: A survey of Techniques

Metarya Ruparel(msr9732) **Safwan Mahmood(sm9453)** **Sriram Ramesh(sr5824)**

1 Abstract

Anomaly detection is an important part of maintaining the health of large scale cloud services. Large scale services need techniques that provide accuracy for diagnosis and Management actions. In this paper, we discuss methods for near real-time anomaly detection like Auto-regression(AR), Moving Average(MA), ARIMA, SARIMA along with traditional machine learning techniques like Isolation Forest (ISF), LSTM and HTM. We evaluate performance metrics on the aforementioned models and report our observations on real world CPU metrics datasets.

2 Introduction

Data centers and Software, today operate and compute at scale of peta-bytes and more. Solving complex computational problems, high availability and scalability of systems are the top priorities, while serving the consumers. Instrumenting and monitoring hidden layers of data centers like rack life cycle management, software management is complex in nature. Any failures or malfunctions in the systems running at such large scale can lead to very expensive losses. Anomaly detector is designed to predict or detect the problems ahead of time thereby enabling advanced diagnoses to determine remedies. Anomaly is described as a deviation from the expected behavior of a system. Anomaly detection should be done continuously, as long as a system is running.

3 Why Threshold-Based Alerting is Not Efficient?

Detection of server or application related problems, also known as 'Anomaly Detection' is more nuanced than simply raising a flag when a particular metric is above a certain predefined threshold because:

1. The value of metrics is dependent on both internal (such as the actual state of the hardware/servers) and external factors (number of transactions, seasonality, etc.)
2. Metric values also vary over time-periods: for example: cpu may be high on certain days of the week than others, and during certain times of the year because of increase in usage during those periods.
3. Often, multiple metrics move together and a real problem may exist only if two or three (or more) given metrics show anomalous behavior instead of just one.

Furthermore, thresholds actually represent one of the main reasons that monitoring is hard. Setting a threshold on a metric requires an Engineer to make a decision about the right value to configure. The difficulty is that there is no right value. A static threshold is just that, never changes over time. Systems are neither similar nor static. Every system is different from one another, and even individual systems changes both over the short term and long term. For these reasons, threshold-based alerting generates several false positives. As a result, Site Reliability Engineers (SRE) often spend a non-trivial amount of time trying to figure out which alerts imply real problems with the system. Our project aims to reduce that time by building an anomaly detection system which generates very few false positives, and presents data in a format that is easy to interpret, so that effort can instead be focused on fixing issues.

4 Dataset

We evaluated our approaches on the Operational Data, Processed and Labeled dataset which is useful for anomaly detection in operations of distributed software systems. The dataset is an industrial case study, on a large, real-world dataset of 7.5 million data points for 231 features. Dataset: Link
The dataset contains labeled feature vectors for each measurement (one minute apart). Some of the features we worked with are:

1. Process CPU : (Process(java) CPU)
2. System CPU : (MXBean(java.lang:type=OperatingSystem).SystemCpuLoad)

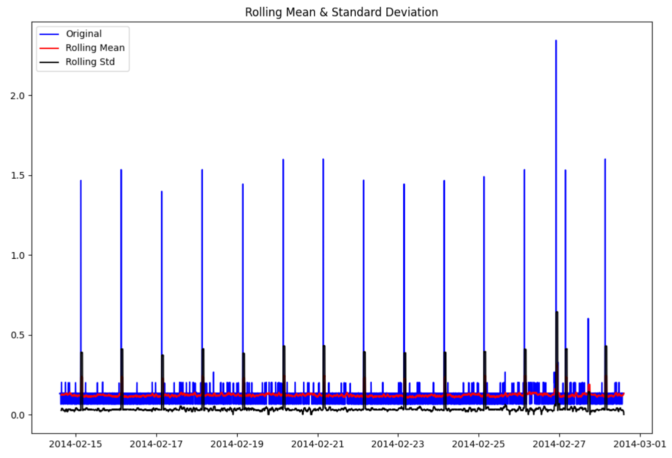


Figure 1: Data Visualization

5 Models

5.1 Auto-regression

The AR(p) model is defined as:

An auto-regressive model can thus be viewed as the output of an all-pole infinite impulse response filter whose input is white noise.[1]

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

Auto-regression uses observations from previous time steps as input to a regression equation to predict the value at the next time step. It is a very simple idea that can result in accurate forecasts on a range of time series problems.

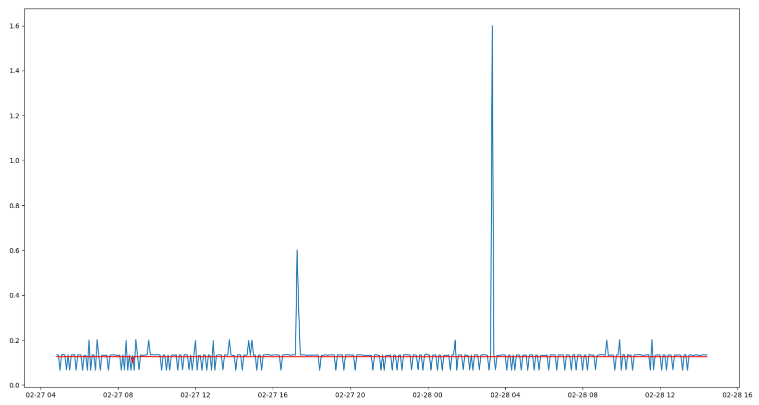


Figure 2: Comparison of Test data(blue) and Auto Regression Predictions(red)

5.2 ARIMA

An autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model.[1]

ARIMA is one of the most widely used forecasting methods for univariate time series data forecasting. It explicitly caters to a suite of standard structures in time series data, and as such provides a simple yet powerful method for making skillful time series forecasts.

Given time series data Xt where t is an integer index and the Xt are real numbers, an ARIMA(p',q) model is given by:

$$X_t - \alpha_1 \cdot X_{t-1} - \cdots - \alpha_{p'} \cdot X_{t-p'} = \varepsilon_t + \theta_1 \cdot \varepsilon_{t-1} + \cdots + \theta_q \cdot \varepsilon_{t-q}$$

Each of these components are explicitly specified in the model as a parameter. A standard notation is used of ARIMA(p,d,q) where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used.

The parameters of the ARIMA(p,d,q) model are defined as follows:

1. p: The number of lag observations included in the model, also called the lag order.
2. d: The number of times that the raw observations are differenced, also called the degree of differencing.
3. q: The size of the moving average window, also called the order of moving average.

The order p and q was determined using the sample auto-correlation function (ACF), partial auto-correlation function (PACF).

We used DickyFuller method to determine if our time-series was stationary or not. The analysis output indicated that we had stationary time-series. Hence we chose d = 0.

We split the training dataset into train and test sets, used the train set to fit the model, and generate a prediction for each element on the test set. A rolling forecast is required given the dependence on observations in prior time steps for differencing and the AR model. We performed this rolling forecast by re-creating the ARIMA model after each new observation is received. We manually keep track of all observations in a list called history that is seeded with the training data and to which new observations are appended each iteration.

Results: We got metrics as Test RMSE: 0.084, Mean Absolute Error: 0.026 Anomalies: 2 out of 3 anomalies in the analysed region were predicted.

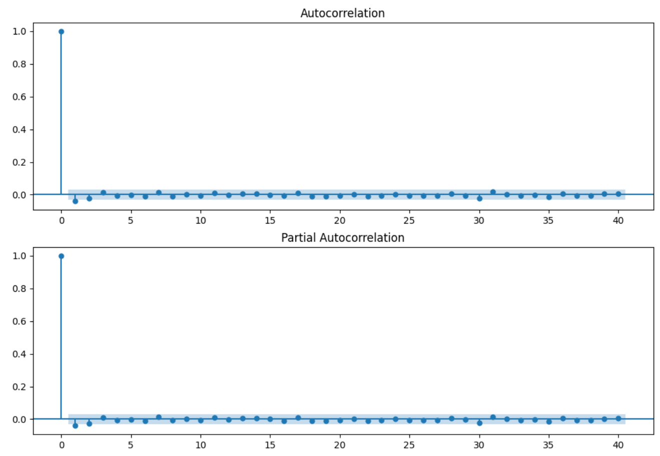


Figure 3: ACF and PACF plots

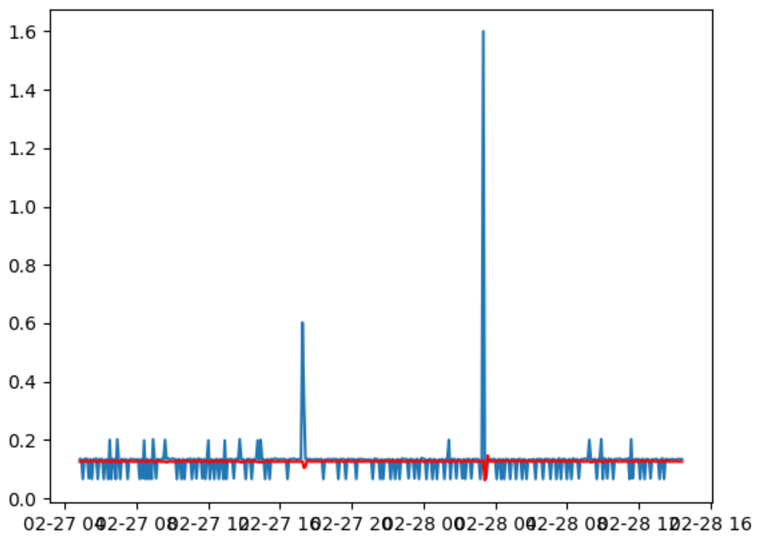


Figure 4: Comparison of Test data(blue) and ARIMA Predictions(red)

5.3 SARIMA

Although ARIMA method can handle data with a trend, it does not support time series with a seasonal component. It supports only an autoregressive and moving average elements. The integrated element refers to differencing allowing the method to support time series data with a trend.

A problem with ARIMA is that it does not support seasonal data. That is a time series with a repeating cycle. ARIMA expects data that is either not seasonal or has the seasonal component removed, e.g. seasonally adjusted via methods such as seasonal differencing.

An extension to ARIMA that supports the direct modeling of the seasonal component of the series is called SARIMA. Seasonal Autoregressive Integrated Moving Average, SARIMA or Seasonal ARIMA, is an extension of ARIMA that explicitly supports univariate time series data with a seasonal component. It adds three new hyperparameters to specify the autoregression (AR), differencing

(I) and moving average (MA) for the seasonal component of the series, as well as an additional parameter for the period of the seasonality.

The model SARIMA(p,d,q)(P,D,Q) [1] has:

Trend Elements
These are the same as the ARIMA model.

Seasonal Elements

1. P: Seasonal auto-regressive order.
2. D: Seasonal difference order.
3. Q: Seasonal moving average order.
4. m: The number of time steps for a single seasonal period.

There are four seasonal elements that are not part of ARIMA. We trained the data in similar manner as ARIMA as described before, with added seasonality elements.

Results: We got metrics as Test RMSE: 0.084, Mean Absolute Error: 0.022 Anomalies: 2 out of 3 anomalies in the analysed region were predicted.

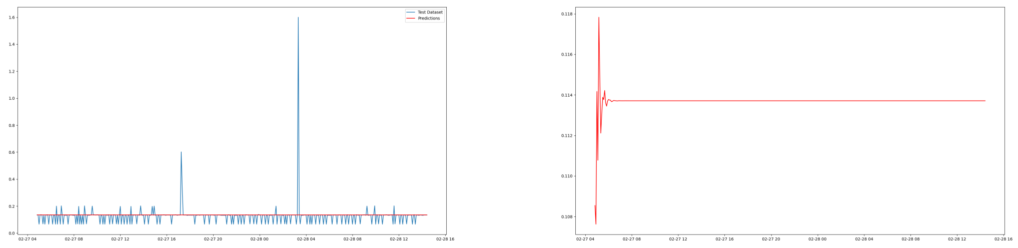


Figure 5: Comparison of Test data(blue) and SARIMA predictions(red)

5.4 ISF

Most existing model-based approaches to anomaly detection construct a profile of normal instances, then identify instances that do not conform to the normal profile as anomalies. ISF explicitly isolates anomalies instead of profiles normal points. The key observation here is that anomalies are ‘few and different’ and therefore they are more susceptible to isolation. In a data-induced random tree, partitioning of instances are repeated recursively until all instances are isolated. This random partitioning produces noticeable shorter paths for anomalies since (a) the fewer instances of anomalies result in a smaller number of partitions – shorter paths in a tree structure, and (b) instances with distinguishable attribute-values are more likely to be separated in early partitioning. Hence, when a forest of random trees collectively produce shorter path lengths for some particular points, then they are highly likely to be anomalies. It works by isolating the outliers by randomly selecting a feature from the given set of features and then randomly selecting a split value between the maximum and minimum values of the selected feature. This random partitioning of features will produce smaller paths in trees for the anomalous data values and distinguish them from the normal set of the data.

Isolation Forest can directly give us the Anomaly Scores for the data-points. We can use decision-function to retrieve these scores. The higher the scores, higher the deviation. A score of above 0.2 is considered aberrant enough to be called anomaly.

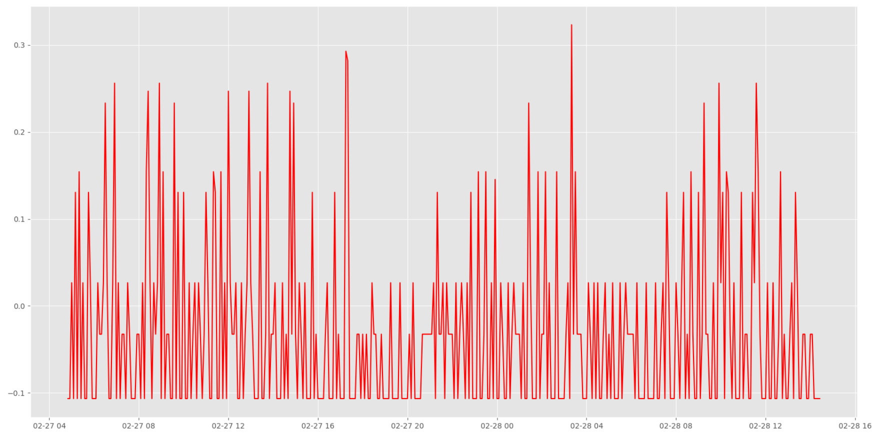


Figure 6: ISF Predictions

5.5 LSTM

The Recurrent Neural Network (RNN) is a generalization of feed-forward neural networks to sequences. Given a sequence of inputs (x1 , ..., xT), a standard RNN computes a sequence of outputs (y1, ..., yT).The RNN can easily map sequences whenever the alignment is known between the inputs the outputs ahead of time. Regular neural networks have been proven to be "universal function approximators". A feed- forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of Rn, under mild assumptions on the activation function. However, ANNs (Artificial Neural Networks) cannot deal with sequential/temporal data. The reason for this is that they have no memory. When analyzing sequential data, it becomes important to look at information in context. The value in each cell must be examined in context of all the values before it. Regular neural networks do not have persistent memory, but Recurrent Neural Networks do. For analyzing large chunks of sequential data, we need models which have long term memory. This led us to use LSTM (Long Short Term Memory) networks, which are a special kind of RNNs, capable of learning long-term dependencies. LSTMs have chain like structure similar to basic RNN but the repeating module has a different structure. Rather than having a single neural network layer, there are four, interacting in a very special way. Each layer of the LSTM network consists of gates which allow the network to forget certain information which is likely to be useless for future processing and retain other information which is likely to be used. A gate is essentially a mathematical function like the sigmoid function which takes a vector as input. The Vanilla RNN Cell is described below :
$S_t = \Phi(WS_{t1} + Ux_t + b)\Phi$ is the activation function.
$S_t \in R^n$ - Current State
$S_{t1} \in R^n$ - Prior State
$x^t \in R^m$ - Current Input
$W \in R^{n\times m}, U \in R^{m\times n}, b \in R^n$ - weights, bias
n - state size
m- input size

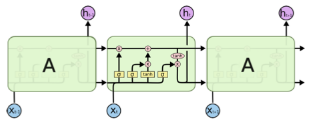


Figure 7: LSTM for Anomaly Detection

LSTMs is a supervised learning algorithm. Since, we did not have any labels for our dataset, we created them using previous values, i.e, label for time t would be t-1. We call this the look back. We tried different lookback values. For lookback = 1.

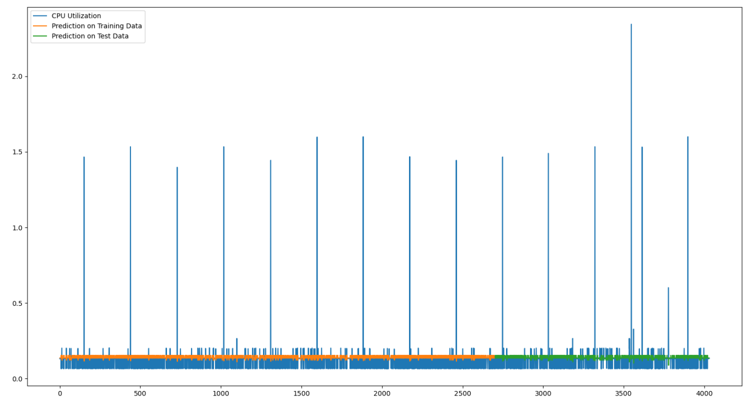


Figure 8: Test Comparison with Prediction having a Lookback of 1

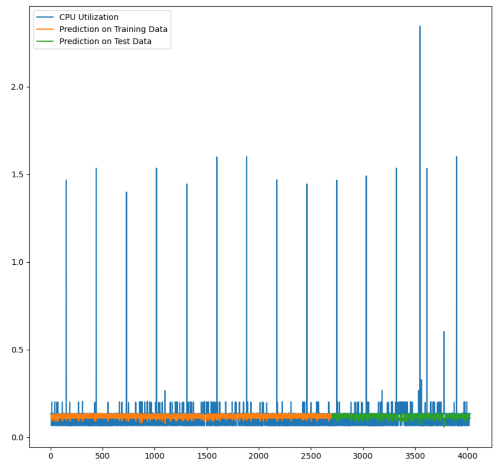
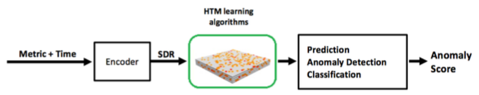


Figure 9: Test Comparison with Prediction having a Lookback of 3

Performance is better as the network has more context about it's past with a lookback of 3

5.6 HTM

Hierarchical Temporal Memory (HTM) is a biologically motivated machine intelligence technology that mimics the architecture and processes of the neocortex. The feature or metric value along with time are supplied to an encoder that converts them into a sparse distributed representation (SDR). SDRs enable several useful properties such as generalization across data streams, resistance to noise, and the attachment of semantic meaning to data points. And then next a sequence of SDRs is fed into the HTM algorithms. HTM algorithms simulate a small slice of the neocortex and are responsible for learning temporal sequences in the server/app metric data streams. Temporal sequences are nothing but patterns over time. HTM learning algorithms learn the temporal patterns present in the data continuously and new patterns are replaced by old patterns in the same way we remember recent events.



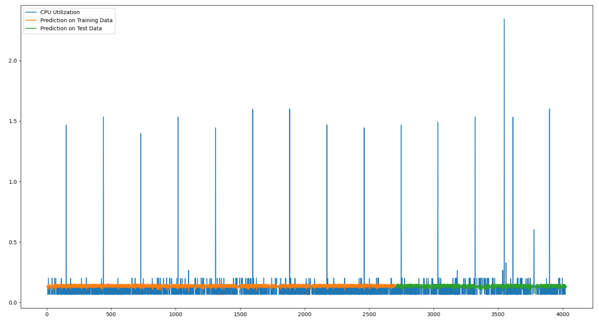


Figure 10: HTM

6 Related work

Anomaly detection in time-series is a heavily studied area of data science and machine learning. Many anomaly detection approaches exist, both supervised (e.g. support vector machines and decision trees) and unsupervised (e.g. clustering). Examples from industry include Netflix's robust principle component analysis (RPCA) method and Yahoo's EGADS.

The Numenta approach [2] was a benchmark on anomaly detection in streaming real-time applications but is generalised. AIOps [3] discussed the real-world challenges and research innovations in this space. It highlighted the complexity in Dev-ops and introduction of AI to ease IT operations along with the challenges in moving forward with AIOps.

Our use case is related to anomalies in OPS. Hence, more advanced time-series modeling and forecasting models are required to detect temporal anomalies in complex scenarios. ARIMA is a general purpose technique for modeling temporal data with seasonality. It is effective at detecting anomalies in data with regular daily or weekly patterns. Extensions of ARIMA enable the automatic determination of seasonality. Model-based approaches like LSTMs, HTMs are useful as they require explicit domain knowledge and leverage that fact in our use case.

7 Future Work

There is a lot of potential in AI Ops to evolve into a system that can grow, evolve, and disrupt everyday IT operations. But there are challenges involved in achieving the above goal. The data quality and quantity available today do not serve the needs of AIOps solutions. Although major cloud services today collect terabytes and even petabytes of telemetry data every day/month, there still lacks representative and high- quality data for building AIOps solutions.Due to limited availability of labelled data, AI Ops has to depend on unsupervised learning to train the models. Also, it is difficult to label what is abnormal in the data. These challenges should be overcome by engineering a system that ensures data/label quality monitoring and assurances, continuous model-quality validation, and actionability of insights.

8 Discussion

As per our observation, Isolation Forest seemed to work really work on the dataset. This is because ISF solely works by isolation the point rather than constructing a profile of normal instances. LSTMs with a look back of 3 also seemed to work really well. This can be backed by the fact that since LSTMs store the history, and we had provided it with a look back of 3, it was very well able to predict the normal values, and hence able to log any anomaly. While HTMs have given us fairly accurate results (as shown in the graph above), at present, we do not have the mathematical proof to say that HTMs perform objectively better or worse than LSTMs. All the Stats Models we tried also performed pretty well on our dataset, however, all of these models are very data centric, i.e, they use profiling of normal observations. As far as our understanding, any change to the normal points in the dataset would require considerable amount of data to perturb to the model, vs ISF only requires a maximum of 256 datapoints before acting up of any changes.

9 Conclusion

Initially, AI Ops needs cross disciplinary collaboration to succeed as new systems must be engineered to cater for new requirements and hyper parameters should be tuned on a case-by-case basis for attaining higher accuracy. Though AI ops promises lucrative benefits, it can be achieved only with close collaboration between the stakeholders.

References

[1] Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[2] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017. Online Real-Time Learning Strategies for Data Streams.

[3] Yingnong Dang, Qingwei Lin, and Peng Huang. Aiops: Real-world challenges and research innovations. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 4–5, 2019.