Human-in-the-Loop Selection of Optimal Time Series Anomaly Detection Methods

Cynthia Freeman and Ian Beaver
Verint Intelligent Self-Service
12809 E. Mirabeau Pkwy
Spokane Valley, WA 99216
E-mail: <first>.<last>@verint.com

Abstract

The existence of an anomaly detection method that works optimally for all domains is a myth. Thus, there exists a continuously increasing number of anomaly detection methods for a wide variety of domains. But a strength can also be a weakness; given this massive library of methods, how can one select the best method for their application? An extensive evaluation of every anomaly detection method is simply not feasible. In this work-in-progress, we present an efficient human-in-the-loop technique to intelligently choose the optimal anomaly detection methods based on the characteristics the time series displays such as seasonality, trend, concept drift, and missing time steps. Once the optimal anomaly detection methods are selected via these characteristics, humans can optionally annotate the predicted outliers which are then used to tune and improve the selected methods for their data. Applying our methodologies can save users time and effort by surfacing the most promising anomaly detection methods instead of experimenting extensively with an expanding library of anomaly detection methods, especially in an online setting.

Introduction

An anomaly in a time series is a pattern that does not conform to past patterns of behavior in the series. Time series anomaly detection is a difficult problem for a multitude of reasons: **(1)** What is anomalous may differ based on application. There is no one-size-fits-all method (Vallis, Hochenbaum, and Kejariwal 2014; Laptev, Amizadeh, and Flint 2015). **(2)** Anomaly detection often must be done online in real-world streaming applications. **(3)** Given the application-specific nature of anomaly detection, it is unlikely that anomaly detection systems will have access to large numbers of tagged datasets. **(4)** Non-anomalous data occurs in significantly larger quantities than anomalous data. **(5)** It is important to detect as many anomalies as accurately and efficiently as possible, but minimizing false positives is also desirable to avoid alarm fatigue. This demands the selected anomaly detection method be optimal to the application for success. **(6)** There is a massive wealth of anomaly detection methods to choose from. Because of these difficulties inherent in time series anomaly detection, we present an efficient, human-in-the-loop technique for the classification of time series and choice of anomaly detection method based on the characteristics the time series possesses.

Approach

In Algorithm 1, we propose an approach based on the characteristics a given time series (ts) possesses. This is essential as some anomaly detection methods perform better on certain characteristics than others. For example, if the time series data in a user's application exhibits concept drift, the user may want to consider a forecasting RNN and not Twitter AnomalyDetection (Freeman et al. 2019).

First, **missing time steps** may make it difficult to apply anomaly detection methods without some form of interpolation. However, other methods can handle missing time steps innately. The system determines the minimal time step difference in the input time series to find missing time steps. The user can then decide if the missing time steps should be filled ($fill$) using some form of interpolation (e.g. linear, etc., called $fill_{option}$) or if the system should limit the selection of anomaly detection methods to those that can innately deal with missing time steps. Next, the system determines if **concept drift** is present in the time series where the definition of normal behavior changes over time (Saurav et al. 2018). Concept drifts can be difficult to detect especially if one does not know beforehand how many concept drifts there are. In (Adams and MacKay 2007), this number does not need to be known. An implementation of this paper is available in (Kulick 2016) using t-distributions for each new concept, referred to as a *run*. The posterior probability ($P(r_t|x_{1:t})$) of the current run r_t's length at each time step (x_i for $i = 1...t$) can be used to determine the presence of concept drifts. In our system, the user selects a threshold for the posterior probability for what is considered a run ($thresh_{post}$) and also how long a run must be before it is actually a concept drift (len_{run}). The system then determines if a time series contains **seasonality**, the presence of variations that occur at specific regular intervals. The system makes use of the `FindFrequency` function in the R *forecast* library (Hyndman, Khandakar, and others 2007) which first removes linear trend from the time series if present and determines the spectral density function from the best fitting autoregressive model. By determining the input f that produces the maximum output spectral density value, FindFrequency returns $\frac{1}{f}$ as the periodicity. If no seasonality is present, 1 is returned. Finally, the system determines if **trend** is present in the time series. Our system detects two types of trend: stochastic (removed via differencing the time series) and deterministic (removed via detrending or removing the line of best fit from the time series). Stochastic trend is identified using the Augmented Dickey-Fuller (ADF) test (Cheung and Lai 1995), and deterministic trends are detected using the Cox-Stuart test (Linden 2000).

In our previous experiments (Freeman et al. 2019), for seasonality and trend, decomposition-based anomaly detection methods such as analyzing the residuals of STL (seasonal decomposition of time series by Loess) (Cleveland et al. 1990), SARIMA (seasonal auto-regressive integrated moving average), and Prophet (Sean J. Taylor 2017) perform the best. For concept drift, more complex methods are necessary such as RNNs (Saurav et al. 2018), Hierarchical Temporal Memory Networks (Hawkins, Ahmad, and Dubinsky 2010), and Prophet. For missing time steps, the number of directly applicable anomaly detection methods is drastically reduced. Although one can interpolate, this does introduce a degree of error. If no interpolation is desired, SARIMA, STL, and Generalized Linear Models are options.

Narrowing the choices down to a smaller class of promising anomaly detection methods (optimalMethods) saves time as there is an ever expanding library of anomaly detection methods. The definition of what is an anomaly is highly subjective, so human input is essential in the decision-making process. Although we automate as much of the process as we can (determining the presence of characteristics, narrowing down the search space of anomaly detection methods), it is not advisable to completely remove the human element. For every selected anomaly detection method, its predicted anomalies (outliers) are given to the user to annotate (Is the predicted anomaly truly an anomaly?), and based on their decision, the parameters for that method can be tuned to reduce the error. Parameter tuning is dependent on the anomaly detection method. For example, if a method produces an anomaly score $\in [0, 100]$ with an anomaly threshold of 75, the system will raise the threshold to reduce false positives. Using this feedback, the system learns to minimize false positives for the user's data.

What about AutoML? There are tasks that are less prone to automation if no labeled data are initially available (exactly our situation). In addition, extreme class imbalance (common in anomaly detection) was the main difficulty for the 2018 AutoML Challenge Series (Hutter, Kotthoff, and Vanschoren 2019), and there was a 15 to 35% performance gap between competitors that solely used AutoML vs those that included human intervention. Yahoo's EGADS (Laptev, Amizadeh, and Flint 2015) is the existing human-in-the-loop system with a similar purpose to ours. However, there are some key differences. EGADS gives users two options: the user can choose (1) how to model the normal behavior of the time series such that a significant deviation from this model is considered an outlier or (2) which decomposition-based method to use with thresholding on the noise component. EGADS then gives users the predicted anomalies to annotate and trains a binary classifier to predict if an anomaly is relevant to the user. The classifier is given the time series and its characteristics such as periodicity and kurtosis as features. However, we focus on the characteristics present in the time series to first discard suboptimal anomaly detection methods. By filtering suboptimal methods, we save users time as they do not need to select from an ever expanding library of anomaly detection methods; they can directly begin working with more promising methods. Our method also avoids potential error introduced by the filtering classifier.

Algorithm 1: Select Optimal Detection Method

```
input     : $ts$
output    : optimal detection method and parameters
parameter : $thresh_{post}$, $len_{run}$, $fill$, $fill_{option}$
seasonality, trend, miss, conceptDrift ← false;
selectedMethod ← none;
if HasMiss($ts$, $fill$, $fill_{option}$) then
    if $fill$ then ts ← FillTS($ts$,$fill_{option}$) ;
    else miss ← true;
if HasConceptDrift($ts$, $thresh_{post}$, $len_{run}$) then
    conceptDrift ← true;
if FindFrequency($ts$) $> 1$ then
    seasonality ← true;
if CoxStuart($ts$) $< .05$ or
  AugmentedDickeyFuller($ts$) $>= .05$ then
    trend ← true;
optimalMethods ←FindOptimal(seasonality, trend,
  conceptDrift, miss);
while selectedMethod is none do
    for (method, methodParams) in optimalMethods
      do
        outliers ←FindOutliers($ts$, method,
          methodParams);
        if User accepts outliers then
            selectedMethod ← method;
            break;
        else
            tags ← User annotates outliers;
            methodParams ← TuneParams(method,
              tags);
return selectedMethod, methodParams
```

Conclusion

In summary, anomaly detection is a hard problem for many reasons, with one of them being method selection in an ever expanding library, especially for non-experts. Our system tackles this problem by determining the characteristics present in the given data and narrowing the choice down to a smaller class of promising anomaly detection methods. We then incorporate user feedback on predicted outliers from the methods in this smaller class to optimize these methods to the user's data. We demonstrate our procedure in this paper and are working towards a rigorous evaluation on a large variety of time series and methods.

References

Adams, R. P., and MacKay, D. J. 2007. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*.

Cheung, Y.-W., and Lai, K. S. 1995. Lag order and critical values of the augmented dickey–fuller test. *Journal of Business & Economic Statistics* 13(3):277–280.

Cleveland, R. B.; Cleveland, W. S.; McRae, J. E.; and Terpenning, I. 1990. Stl: A seasonal-trend decomposition. *Journal of Official Statistics* 6(1):3–73.

Freeman, C.; Merriman, J.; Beaver, I.; and Mueen, A. 2019. Experimental comparison of online anomaly detection algorithms. In *The 32nd International Flairs Conference*.

Hawkins, J.; Ahmad, S.; and Dubinsky, D. 2010. Hierarchical temporal memory including htm cortical learning algorithms. *Technical report, Numenta, Inc, Palto Alto*.

Hutter, F.; Kotthoff, L.; and Vanschoren, J. 2019. Automated machine learning-methods, systems, challenges.

Hyndman, R. J.; Khandakar, Y.; et al. 2007. *Automatic time series for forecasting: the forecast package for R*. Number 6/07. Monash University, Department of Econometrics and Business Statistics . . . .

Kulick, J. 2016. Bayesian changepoint detection. https://github.com/hildensia/bayesian_changepoint_detection.

Laptev, N.; Amizadeh, S.; and Flint, I. 2015. Generic and scalable framework for automated time-series anomaly detection. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1939–1947. ACM.

Linden, M. 2000. Testing growth convergence with time series data—a non-parametric approach. *International Review of Applied Economics* 14(3):361–370.

Saurav, S.; Malhotra, P.; TV, V.; Gugulothu, N.; Vig, L.; Agarwal, P.; and Shroff, G. 2018. Online anomaly detection with concept drift adaptation using recurrent neural networks. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, 78–87. ACM.

Sean J. Taylor, B. L. 2017. Forecasting at scale. *PeerJ preprints 5:e3190v2*.

Vallis, O.; Hochenbaum, J.; and Kejariwal, A. 2014. A novel technique for long-term anomaly detection in the cloud. In *6th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 14)*.