Anomaly Detection in streaming non-stationarity temporal data

Talagala, Hyndman, Smith-Miles, Kandanaarachchi, Munoz
December 12, 2018

as presented by Giorgos Kadilierakis MSc student at Computer Science, University of Crete.
Synopsis
Abstract / One slide

A framework that provides early detection of anomalous series within a large collection of non-stationary streaming time series data.

The described framework does:

- forecasts a boundary for the systems typical behavior using extreme value theory.
- uses a sliding window to test for anomalous series within a newly arrived collection of series.
Motivation & Problem statement

An anomaly is an observation that is very unlikely given a recent distribution of a system.

A wide range of applications:

- extreme weather condition detection
- gas and oil leakages
- power cable faults

Automatic detection of these critical events is vital since:

- manual monitoring is ineffective
- operational life is spend in a typical state.
Different types of anomalies

Figure 1: Anomalies are represented by red color while black color is corresponding to the typical behavior.
Novelty

Authors state 3 main contributions regarding anomaly detection in streaming non-stationary time series:

1. Early detection, presence of noisy signals and multimodal distributions.
2. Effectively dealing with non-stationarity (aka concept-drift).
3. Wide applicability, by using data from several application domains.
Abstract 2/2

Applicability & usefulness are shown by using both *synthetic* and *real-world datasets*.

The framework is implemented as an open-source R project in a github repo *oddstream*.
Multivariate time series plot of a fiber optic cable sensors

Figure 2: 640 time series each with 1459 time points. Yellow corresponds to low values while black to high values.
Background
Temporal Data
Temporal data is simply data that represents a state in time and are collected to analyze weather patterns and other environmental variables.

Non stationary data - Concept drift
A stationary process is one whose statistical properties do not change over time. In real life time-series are most often non-stationary. That is that the mean and the variance are not constant.
Streaming Data
Anomaly detection in temporal data:

1. *batch processing*, where entire data is available prior to the analysis
2. *streaming-data*, where data evolve over time.

Challenges in streaming data

1. large *volume* and high *velocity*
2. very noisy signals
3. non-stationarity

All of the above make the process of distinguish between new "typical" behaviors and "anomalous" events very difficult.
Extreme Value Theory (EVT) - Intro

Algorithms proposed are based on EVT, a branch of probability theory that relates to the statistical behavior of extreme order statistics in a given sample.

It seeks to assess, from a given ordered sample of a given random variable, the probability of events that are more extreme than any previously observed.
Anomaly Detection and the EVT
Anomalies are defined in terms of either (i) distance or (ii) density.

1. In the case of distance one expects to see relatively large separations between typical data and the anomalies. K-NN approaches, observations with a large $K$ are defined as anomalies: spacing theorem

2. Defining based on density means that an anomaly is an observation that has a very low chance of happening.

Main focus, when employing $EVT$, is on defining a threshold between typical and anomaly data points.
Anomaly Detection based on EVT, density-based

The EV distributions can be parameterized by the size of the sample from which the extrema is taken, $m$.

Based on [1] the following method is used for selecting a threshold for identifying anomalous points when $m \geq 1$.

A $\Psi$ transform method defines the ”most extreme” of a set of $m$ samples $X = \{x_1, x_2, ..., x_m\}$, distributed based on pdf, as $\arg\min_{x \in X}[f(x)]$. 
Extreme Value Distributions

**Figure 3:** Each different $m$ describes where the maximum of $m$ samples will lie.
Methodology

Pipeline process
The proposed framework consists of 2 distinct phases:

1. building a model of the typical behavior of a given system, \textit{off-line}

2. testing newly arrived data against the typical model, \textit{on-line}

Previous Limitations
Overcoming limitations of previous work:

- an alarm is triggered only in the presence of anomalous event(s)
- handling gradually arising anomalies that occur of highly dependent time-series.
Some applications do not exhibit large gaps between typical observations and anomalies.

**Figure 4:** (a),(c) correspond to time-series obtained via independent sensors while (b),(d) from sensors not independent to one another.
Algorithm 1 - Offline phase

- **input:** $D_{\text{norm}}$, a collection of $n$ time-series generated from $n$ sensors.
- **output:** $t$, anomalous threshold.

Steps 1/3

1. extract $m$ features from each of the time series in $D_{\text{norm}}$. Output is a NxM feature matrix $M$.
2. normalize feature values (m columns). Output is a new matrix (NxM) called $M^*$. 
3. apply PCA to the $M^*$ matrix. Output is a ranking list of the $m$ features principal components.
Methodology cont.

Feature representation of the time-series.

**Figure 5:** 640 time-series in Axis 'Cable'. Each plot correspond to a feature type(14 different) extracted from all 640 time-series.
Algorithm 1 steps 2/3

4. define a 2d space by selecting the first 2 principal components(or features) from step’s 3 ranking list. Output is a ’2D PC space’.

5. estimate the probability density of the ’2D PC space’ using a kernel density estimation. Output is the estimated probability density function(or distribution function) $f_2$.

6. draw a large number $N$ of extremes from $f_2$ and form an empirical distribution of the densities in the $\Psi$ transform space:
Algorithm 1 steps 3/3

\[
\Psi[\hat{f}_2(x)] = \begin{cases} 
(-2\ln(\hat{f}_2(x)) - 2\ln(2\pi))^{1/2}, & \hat{f}_2(x) < (2\pi)^{-1} \\
0, & \hat{f}_2(x) \geq (2\pi)^{-1}.
\end{cases}
\]

- 7. fit a *Gumbel* distribution, based parameters are obtained via MLE.
- 8. using the previous fitted distribution, find the location of an anomalous threshold \(t\) in \(f_2\) which translates in the 2D PC space in a contour \(t^*\).
Algorithm 2 - Online phase

A sliding window $w$ is used to handle the streaming data context in a current time point $t$.

- **input:** $W[t - w, t]$ the current sliding window with $n$ time-series and $t^*$ as threshold.
- **output:** a vector of indexes of the anomalous series.

**Steps**

1. extract $m$ features from each of the $n$-time series of the selected time window. Output is $NxM$ matrix $M_{test}$. 

Algorithm 2 steps cont.

2. project this new feature matrix $M_{test}$, on the same 2D PC space of the typical data that was built using the time series in $D_{norm}$.

3. find the probability density values of $y_1, y_2, .., y_n \in M_{test}$ with respect to $f_2$ from step 5.

4. find any $y_j$ where $f_2(y_j) < t^*$ and mark the time-series as anomalous in window $W[t - \omega, t]$.

5. repeat 1-4 steps for each new time window that is generated by the current time point $t$. 
Non-stationarity in practice

May occur in multiple forms as "time-varying" or "structural-break". Four classes:

1. **sudden**, a sudden switch from one distribution to another
2. **gradual**, moving to new distribution by switching back and forth to the old-new
3. **re-occurring**, a previously seen distribution reoccurs.
4. **incremental**, many slowly-changing intermediate distributions between the old and new.
Methodology - Algorithm 3

Detecting non-stationarity
Finding ”new” typical behaviors of the decision model. Consists of 2 approaches:

1. *blind approach*, model updates at regular time intervals
2. *informed approach*, model updates only in presence of a *concept-drift*.

Algorithm 3 - Updating model’s typical behavior

- **input:** $w$, length of the moving window & $D_{t_0}$ collection of $n$ time-series of length $w$ representing typical behavior.
- **output:** a vector of indexes of the anomalous series in each window
Algorithm 3 steps 1/2

1. estimate \( f_{t0} \), the pdf of the 2D-PC space defined by \( D_{t0} \).
2. extract \( m \) features from \( W[t \ w, t] \). Output is a \( n \times m \) feature matrix \( M_{test} \).
3. project \( M_{test} \) onto the 2D-PC space of \( D_{t0} \). \( Y_{tt} \) represents the new projected data points.
4. identify points on the 2D-space that correspond to the typical series in the window, using the anomalous threshold. \( Y_{ttnorm} \) represents the typical series of the window.
5. let \( p \) be the proportion of anomalies detected in the window. If \( p < p^* \), where \( p^* = 0.5 \) find \( f_t \) of \( Y_{ttnorm} \) else find \( f_t \) of \( Y_{tt} \).
Algorithm 3 steps 2/2

6. use a simple distance measure as the squared discrepancy, test the null hypothesis $H_0 : f_{t_0} = f_{t_t}$.

7. if $H_0$ is rejected: and $p > p^*$ then $D_{t_0}$ is set to $W[t - w, t]_{norm}$ else $D_{t_0}$ is set to $W[t - w, t]$.

8. repeat steps 1-7 for every new time window.
Experiment & Evaluation
Effectiveness of the proposed algorithms is evaluated upon 4 synthetic data that have been generated including:

1. multimodal typical classes
2. different patterns of non-stationarity
3. noisy signals

*Set of examples is relatively limited, results should only be viewed as illustrations of the algorithms.*
Anomaly detection in streaming data, no non-stationarity

*Demonstrating the application of Algorithms 1 & 2.*

Containing a *bi-modal* typical class through out the entire period. Time series contain on purpose *noisy* signals. Sliding window length is set at 150s and an anomaly is first introduced at 400s.

**Evaluation metric**

Avg accuracy: *ratio of the number of correctly classified series to the total number of series of each moving window.* False positive and false negative are also been reported.
Results in streaming data, no non-stationarity

Figure 6: Anomalous event is placed at 400s, algorithms 1&2 detect the anomaly right from the beginning: window reaches $W[151,300]$
Remarks
Anomaly stops at $t=1000$, system generates an alarm until $t=1500$ due to the length($=150$) of the moving window that is used.

System issues an alarm until it reaches a window that is completely free of the anomalous event resulting in increase of the false positive rate.

Evaluation results

- Avg accuracy: 0.989
- False positive rate: 0.0075
- False negative rate: 0.003
Anomaly detection with non-stationarity

Performance of Algorithm 3, along with 1 & 2, is now investigated using 4 synthetic datasets. Those datasets exhibit the four different kinds of non-stationarity:

1. sudden
2. gradual
3. reoccurring
4. incremental

In all 4 cases non-stationary behavior starts at $t = 300$. 
Experiment & Evaluation Part 2

Results in streaming non-stationary data: sudden

**Figure 7:** Anomalous event is placed from 450s to 475s between 150th and 170th series.
Results in streaming non-stationary data: gradual

**Figure 8:** Anomalous event is placed from 850s to 875s between 150th and 170th series.
Results in streaming non-stationary data: reoccurring

Figure 9: Anomalous event is placed from 825s to 875s between 150th and 170th series.
Remarks 1/2
In the first 3 cases decision model declares almost all points as anomalies. Meaning that: $p > p^* (p^* = 0.5)$ in $W[201, 300]$, where $p$: the proportion of outliers in the selected window $W$.

The model mistakes non-stationary data as anomalies and classifies them as such but as soon as the new pattern is established system is adapted fully to the new distribution.

Evaluation results 1/2
Initial low accuracy (because of the new distribution), system achieves high avg accuracy and low false positive and false negative rates.
Results in streaming non-stationary data: incremental

**Figure 10:** Anomalous event is placed from 825s to 875s between 150th and 170th series.
Remarks 2/2

In contrast in the 4th case none of the sliding windows declare more than half of the series (majority rule) to be outliers.

As a result model’s updating process is done using only the typical series detected for each window, based on step 5(a) of Alg. 3.

Evaluation results 2/2

Entire period’s results on incremental distribution:

- Avg accuracy is **0.95**
- Low false positive at **0.047**, and zero false negative on average.
Results in streaming non-stationary data, p-value:

Figure 11: Detection of concept drift. P value for the hypothesis test: $f = f_{tt}$. In these examples the significance level is set to 0.05 and is marked by the horizontal line in each plot.
Application
Application on real systems

All 3 proposed algorithms are applied to datasets obtained using fiber optic sensor cables attached to a system.

Since there is no truth for comparison, graphical representations are used to evaluate the performances of the proposed algorithms on these datasets.
Results in real datasets

**Figure 12:** Left panel: (black: high values, yellow: low values, black shapes are corresponding to anomalous events). Right panel: (black: outliers, gray: typical behavior)
Results in real datasets, p-value

**Figure 13:** Detection of concept drift. P value for the hypothesis test: \( f = f_{tt} \). In these examples the significance level is set to 0.05 and is marked by the horizontal line in each plot.
Conclusion
Conclusion 1/2
This paper proposes a methodology for the detection of anomalous series within a large collection of streaming time series using \textit{EVT}.

Copes with non-stationarity by using sliding window comparisons of feature densities, allowing the decision model to adjust to the changing environment automatically as changes are detected.

Analysis consists of both synthetic data and data obtained using fiber optic cables reveals that the proposed framework (Algorithms 1, 2 and 3) can work well in the presence of non-stationarity and noisy time series from multi-modal typical classes.
Conclusion 2/2

+ applying solutions for non-stationarity, proposing novel techniques.
+ deals with streaming-data by applying an informed-approach of anomaly detection.
+ open-source implementation, reproducibility.

- reporting evaluation numbers for real-life datasets
- does not specifically address high volume or high velocity as stated earlier.
- lack of comparison with similar frameworks.
References


Questions