Operational Anomaly Detection

by Val Delane, version 2010-09-22

“One man’s noise is another man’s signal.” – Edward Ng

“Prediction is very difficult, especially if it's about the future.” – Nils Bohr

“Our first obligation is to keep the foo counters turning.” – Dictionary of the TMRC Language

Todo

This is a draft. Still to do:

- Flesh out last section Declaring an Anomaly
- After the introduction this is a terse iteration and brief explanation of methods, many of which are hard to understand without examples. Too dense as is. Add examples.
- Perhaps expand on the issue of data collection/quality
- Add references/citations

Introduction

A typical operations monitoring system uses simple thresholds to trigger most alerts, for example:

- Device/service has not responded for N minutes or N periods
- Performance metric is above or below absolute boundary
- Performance metric is within absolute boundary but “too close” (N units or N% away)

Simple thresholds are great because they are easy to understand, easy to implement, and catch most of the serious operational concerns. However, even when behavior is nominal according to the thresholds, the monitoring data are a treasure trove that, properly analyzed, can reveal anomalous, cycling, and trending behavior well worthy of human attention. For example, we might like to know about the following operating conditions:

- Performance/load/temperature/etc. of service/device differs significantly from itself in the past (includes typical threshold monitoring)
- Performance of device differs significantly from the usual performance of the same type of device sharing the same job and load (intra-cluster)
- Performance of device differs significantly from the overall usual performance of the same type of device
- Performance of application running on a cluster differs significantly from itself in the past
- Performance of application running on a cluster differs significantly from usual performance of application across all clusters

- Recent performance cycles do not match previous cycles
- Performance is trending linearly or the trend itself is changing (accelerating or decelerating)
- User access pattern on a device or cluster differs significantly from the past

“Performance” includes all health-oriented observations, including the error/failure/recovery profile. In the above list replace “performance” with “configuration” for another set of differences we might like to know about. Performance and configuration variables can be generalized as attributes. For the purpose of identifying anomalies and trends, performance and configuration fundamentally differ only by how often we expect the attribute values to change. For example, we expect the amount of server memory used to vary over time, but we do not expect the amount of physical RAM to change very often. We want to compare attributes as well as objects carrying one or more vectors of attributes. All of these are comparisons either of an object to itself across time, or of an object to a like object at the same time. These comparisons can generate alerts and reports of interest to different parties:

- Sudden change: operational personnel, perhaps escalated to systems engineering
- Outliers comparing objects to like objects: systems engineering, perhaps escalated to management
- Seasonality (cyclic patterns) and trends (gradual directed change): systems engineering and management

When such conditions are properly identified and communicated we are spurred to take advantage of these opportunities, among others: quickly mitigate events, solve the right problems, provision appropriate capacity, identify and reclaim unused capacity, up-sell additional capacity.

In order to detect operational anomalies, trends, and seasonality we can summarize the challenges as: exposing information in a univariate time series, measuring distance between vectors and between objects, and declaring an anomaly when differences are significant enough to require human attention.

Exposing Information in a Time Series

We can apply univariate (single variable) time series analysis to reveal information in the monitoring data. A univariate time series is a sequence of observations (measurements) of a single attribute of a single object, with the observations performed at a consistent frequency. It can also be described as the vector of an interval variable (range) across the time domain.

Using transformations (filters/functions) with different characteristics we can suppress some information in order to highlight (expose/detect) other information. We would like to expose the following information: acute changes, trends, and seasonality.

In the immediate sense an acute change is a sudden departure from recent behavior, marked by the most recent observation(s) being significantly different from the immediately previous

observation(s). In retrospect of a longer time series, acute changes are indicated by local outliers —that is, observations that are significantly different than observations nearby in time.

A trend is a gradual change. A basic trend is a straight line on a slope. The trend itself can also trend, in other words accelerate or decelerate. Such second-order trending is of special interest and can constitute an anomaly in that we intuitively expect and plan only for first-order trends.

Seasonality is the parlance of time series analysts for cyclical (repeated) variation. In early work they needed a way to describe and discount the usual variation in sales from season to season. For example, automobile sales are usually higher in Spring than Autumn. The term holds even for short cyclical variation, such as traffic at noon usually being greater than traffic at midnight.

To expose acute changes, suppress the overall trend and seasonality. To expose trends, suppress seasonality and high-frequency noise. To expose seasonality, suppress the overall trend and high frequency noise. There exist several prediction formulas designed to generate a grand unifying model simultaneously accounting for noise, trends, seasonality—but they require very careful tuning of several mysterious parameters and are prone to instability or over-fitting. The more parameters in a model, the more likely it only follows the data and does not represent an underlying relationship. We are likely better off splitting up the study and applying just the appropriate filters based on what information we wish to expose for that part of the study.

Caveats and Assumptions for Time Series Analysis

Classical statistical methods of time series analysis assume that the series has a normal distribution of noise with possible components of trend and/or seasonality. This is a simplification which sometimes leads to artifacts (spurious results). A violation of the assumption of normal noise can be mitigated with a robust statistical estimator (function that measures distance from model to data) that is less sensitive to outliers than a classical estimator.

Statistical methods also assume that the time series contains values observed at a consistent frequency and for which every value is a valid observation. Inconsistent sampling frequency and nonsensical observation values contribute to artifacts. Because of such collection issues operational monitoring data are, strictly speaking, event series—not time series. However, analyzing an event series is much harder than analyzing a time series, so we approximate the latter the best we can by striving to improve data quality: the collection mechanism should be reliable, consult a state table to decide if the system is in a mode such that a given attribute is meaningful at the time of observation, and finally, discard ridiculous values. The data retrieval mechanism should be able to interpolate missing values when feeding time series analyses.

If observation sampling is of insufficient resolution to capture at least the rough shape of any high-amplitude seasonality there will be artifacts.

We assume there will be some artifacts in time series analyses results so we will subsequently apply additional techniques to minimize the incidence of false positive and false negative errors in anomaly detection.

Exposing Acute Changes in a Time Series

To expose acute changes in a time series, use a high pass filter. Here is a crude one:

- Take a poor man's first order derivative by differencing: from raw series *foo* generate a new series *bar* that consists of the difference between adjacent observations in *foo*.
- Values of *bar* beyond a certain level of standard deviation from mean may be significant and therefore an acute anomaly.

This filter takes advantage of the most basic principle of auto-correlation in a time series: in the absence of any other model, the previous observation is the best prediction of the next observation. In fact, when there is more than one factor in the behavior this works better than most models—which says something about how difficult it is to make a good model.

Here is a better high pass filter:

- Run a low period smoothing function on series *foo* to generate lightly smoothed series *bar*. A good and popular choice is the exponential moving average (EMA) where more recent observations are given more weight. A better choice is a kernel (local) smoother, which has two major advantages: it works well even when the observation period is irregular, and it preserves the phase of the original series—unlike any moving average, which contributes a phase lag proportional to the degree of smoothing.
- Subtract *bar* from *foo* to obtain residual (irregular component) series *baz*.
- Values of *baz* beyond a certain level of standard deviation from mean may be significant.

Neither of these methods compensate for an accelerating trend that would generate more outliers toward the end of the sequence. This is intentional; we want to know about such behavior. Because we want to highlight outlying observations we use classical standard deviation as an outlier amplifier.

Exposing Trends in a Time Series

To expose the trend, suppress seasonality and high-frequency noise:

- Choose a time frame that is at least five times longer than the longest prominent seasonality in the data. A shorter time frame may appear to contain a trend that is actually just one side of seasonality.
- Perform a multiple linear regression of first and second-order polynomials on raw series *foo* to find the trend curve *bar*.
- Plot *bar* against *foo*. A significant trend should be easily observed and sufficient for planning purposes.
- If there is a second-order polynomial component the trend is accelerating or decelerating, which may be a significant anomaly.

In this case we prefer that outliers not skew the results so we use a robust estimator, median absolute deviation, for regression fitting.

Exposing Seasonality in a Time Series

To expose seasonality, suppress the overall trend and high frequency noise:

- Subtract from raw series *foo* the trend found in the previous section to generate series *bar*.
- Run a kernel smoother on *bar* to generate lightly smoothed series *baz*.
- Plot *baz*. Significant seasonality of a period less than, say, 1/10 of the series length should be easily observed and sufficient for planning purposes.

It is possible to automatically identify seasonality using signal processing techniques. If a time series signal is detrended, cyclo-stationary (consistently repeats itself, i.e., completely predictable) and is just the sum of one or more cycles sufficiently distinct in frequency, we can use a Fourier transform to rotate the time domain into a frequency domain and decompose the signal into a list of frequencies, amplitudes, and phases. Unfortunately, signals amenable to Fourier analysis are rare in the real world. While the operational data of interest are generated by deterministic processes, there are too many factors for the results to be very predictable. A generalization of Fourier processing is wavelet processing, which can extract frequency information from a non-stationary signal. Wavelet theory is a fascinating diversion, but is unnecessary to achieve our goal of robust anomaly detection. It turns out that we do not have to explicitly identify the seasonality in order to discount it or detect a change in it. Therefore we do not need to pursue independent component analysis or any other blind source separation methods unless we are after a spectral density signature.

Measuring Distance Between Vectors and Objects

We compare current performance of like objects (devices/services/applications) with the intent of exposing those objects behaving differently. It is easy to compare vectors that are sequential observations of the same attribute over the same time frame:

- From all input vectors generate median vector *foo* (*foo* element 0 is the median of element 0 of all input vectors, element 1 is the median of element 1 of all input vectors, etc.)
- For each input vector subtract *foo* then sum the absolute values of all remainders. The sum (one for each vector) represents the distance from the median.
- Distances beyond a certain level of standard deviation may be significant and therefore indicate anomalous behavior.

How do we compare like vectors of an object to itself across time? Recall the basic principle of auto-correlation in a time series: in the absence of any other model, the previous observation is the best prediction of the next observation. This applies to a chunked series of observations as well: we can generally expect the next time frame to be similar to the previous time frame. If we pick the appropriate frame and it's not roughly the same then there is a trend or a more complex behavioral change. If we compare the essential shape of entire frames and find matches we can discount the redundancy (repeating patterns) leaving a residual of the non-repeating behavior—anomalous by definition. The redundancy itself may also be of some interest for planning purposes. This autocorrelation—comparing a signal to a time-lagged version of itself—amounts to measuring the distance between like vectors but with a serious wrinkle: we do not know the

period of the repeating part of the signal, so we do not know the appropriate lag to apply before comparing vectors, nor the appropriate frame width. After discounting the overall trend, there are several ways to attack this problem:

The classic autocorrelator is the parametric Box and Jenkins autoregressive moving average model. Unfortunately, the parameters are very difficult to estimate and the model works best on a time series with uncomplicated behavior. It is not recommended for this application.

The brute-force approach is to iteratively compare frames of increasing width against every phase (lag) of the vector to find the best matches (judged as those with the smallest difference between the frames). The description of this concept helps us to understand what we are looking for, but is computationally unfeasible with vectors of a useful length.

We could apply parallel-hybrid evolutionary computation methods (steady state, variable length real-value genome, roulette-wheel and genetic variance selection for breeding and culling, two-parent propagation with modified uniform crossover, local hill-climbing via parthenogenesis, self-adaptive mutation rate, etc.)—but models generated by evolutionary computation often overfit to the input data and therefore are not reliable for prediction. EC is best for finding specific solutions, not models.

Dynamic Time Warping (DTW) introduced by Bellman et al. is a popular, visually intuitive method of pairwise comparison based on a recurrence plot. The vectors are represented on two axes and the intersection is flagged if the vector elements have the same value. Congruent autocorrelation appears as a diagonal run (slope of 1). Stretched autocorrelation (pattern match but at different speed) appears as a diagonal run of positive slope other than 1. DTW reveals correlations even when the repeated patterns have different gaps between them in the two vectors. We discount portions of the vector corresponding to long diagonal runs of slope 1. Portions not corresponding to a diagonal run are anomalous. Although the basic DTW function is nonparametric, for this application we would set a threshold of intersection (diagonal run) length above which we consider a significant match and therefore discountable from the vector. We can also use the fraction of points forming diagonal lines or the mean diagonal line length as measures of determinism or inverse divergence, respectively. DTW is an excellent exploration tool because the human eye can immediately spot correlation patterns in the recurrence plot. However, it is probably not the best tool for automated anomaly detection.

The Compression-based Dissimilarity Measure (CDM) by Keogh et al. is an exciting and elegant nonparametric method of anomaly detection based on Kolmogorov Complexity (a measure of randomness) and more specifically the Minimum Description Length (MDL) principle by Rissanen. When applied to a time series CDM uses a divide and conquer algorithm iteratively comparing the compressibility of a subvector to the compressibility of each half of the subvector. The result is a direct identification of the location and degree of greatest novelty (anomaly) in the global time series vector without having to explicitly find appropriate phases or frame widths for comparison. CDM is much newer than DTW but gaining in popularity and is relatively easy to implement. CDM can also be used to compare multivariate (multi-dimensional) objects and it implicitly collapses redundant dimensions with no additional effort. Nominal and ordinal variables are handled natively, as are interval variables if there are not too many different

values. Real number values should be summarized as described below. The only required CDM "parameter" is the type of compression algorithm, which is easy to select; it is simply the algorithm among those on hand which produces the shortest output from the global vector. Finally, although it is trivial to comparing one-dimensional vectors of like objects in the same time frame, we might as well use CDM for that comparison as well, because we already need it for comparing objects across time.

In order for DTW or CDM to cope effectively with a real number vector we must first summarize the vector. In one sense it is filtering out the high frequency noise, but more importantly it is finding a "close enough" approximation of values such that we can minimize the number of values used while retaining the essential shape of the signal. There are many methods for this, including Fourier or wavelet transforms (only effective if there are obvious periodicities in the signal), Singular Value Decomposition (SVD), Chebyshev polynomials (CHEB), Indexable Piecewise Linear Approximation (IPLA), Adaptive Piecewise Constant Approximation (APCA), and Piecewise Aggregate Approximation (PAA). A recent and excellent branch of PAA is the Symbolic Aggregate approXimation (SAX) by Keogh et al. In its simplest application, SAX encodes a time series vector into a compact string. SAX can also be used to compare two vectors but for our purposes it is best used as a preprocessor for CDM.

Also, once real number vectors have been encoded into strings we can measure the Levenshtein distance between them (the least number of edit operations—insertion, deletion, substitution of a character) that would transform one string into the other. This is a very popular metric in bioinformatics and might be a useful estimator to supplement CDM.

Declaring an Anomaly

Assumption: on average, we're taking good care of the systems, so the median of their signatures is a healthy signature.

We intuitively define an outlier as something that isn't part of the normal group. In order to assess if something is in the group, we measure the similarity then apply a threshold and declare that items beyond that threshold of similarity are outliers. The threshold is subjective—but similarity (1/distance) can be scored and ranked objectively.

Use clustering (many methods to choose from)

Tag objects with possible anomalies. Consider using a decaying score because the most important anomalies are recent.

Rank anomalous objects in presentation so that most important ones receive first attention.

We can send alerts, reports, and/or automatically trigger external processes.

If computing resources are limited, consider prioritizing the acute change analysis phase over the others, starting with the more recent and shortest time frame. Then give it a fast track through the anomaly sanity check.

Thanks

Reviewers: Josh B,