Aberrant Behavior Detection in Time Series for Monitoring Business-Critical Metrics (DRAFT)

Evan Miller
IMVU, Inc.
emiller@imvu.com

Oct. 28, 2007

1 Abstract

Detecting failures swiftly is a key process for maintaining a high uptime for on-line applications. This paper describes a program that analyzes real-time business metrics and reports the appearance of unusual behavior, thus warning of potential application failure. We build upon Brutlag [1], which was designed for monitoring high-volume network traffic, and extend the mathematical model to work with comparatively low-volume metrics, such as number of incoming orders. We show some examples of the program's analysis of real data, and suggest additional improvements to the model and its implementation.

2 Introduction

Application failures that go unnoticed cause unnecessary revenue loss and customer frustration. While system and network level monitoring may catch a large number of problems, they do not address higher-level failures, such as the introduction of an application bug or an external service interface that changes without warning. One method for detecting these application failures is to run automated functional tests at a regular interval. These tests might emulate an end-user's client session, or interact with an external server; their common purpose is to verify that certain expected results occur.

Tests by themselves, however, are often insufficient. They might miss bugs that appear only a fraction of the time, or that are only manifest to a certain subset of all users. It is therefore useful to complement tests with a real-time analysis of business metrics in order to detect small but significant service degradations.

An analysis consists of two parts. First, *forecasting* consists of determining some set of expected values for a metric as a function of time (e.g., “between noon and 1 PM on Friday, we expect 20 orders”); these expected values might incorporate information about our uncertainty in them (“plus or minus 5”). Second, *anomaly analysis* consists of comparing the expectations to the incoming observations; an anomaly analysis will tell us whether a set of observations is *anomalous* or *non-anomalous*.

The simplest approach to forecasting is for a human to decide upon “reasonable” upper and lower thresholds for given metrics at given times of day; the anomaly analysis then consists of comparing incoming observations to this table of pre-defined thresholds, and alerting if any observations are out of bounds. However, manually creating these tables of expectations is time-consuming, and keeping them up-to-date as the business grows (or contracts) is burdensome compared to expectations that are generated dynamically.

Our work takes advantage of Holt-Winters forecasting as incorporated into RRDtool (an open-source time-series database) by Brutlag [1]; the algorithm uses exponential smoothing techniques to dynamically formulate expected values based on historical values, taking into account daily fluctuations as well as day-to-day trends. Readers interested in the details of Holt-Winters forecasting are referred to Brutlag [1]. The characteristic relevant to this paper is that Holt-Winters is modeled around seasonal variation; that is, predictions are formulated with respect to a daily, weekly, or yearly cycle.

Although Holt-Winters forecasting is well-studied and robust, we find Brutlag’s anomaly analysis to be insufficient when *the expected value is comparable to the expected deviation*. That is, if the “confidence band” he describes extends to zero, then Brutlag’s method will record as non-anomalous an infinite sequence of “0” observations. This characteristic makes the Brutlag method unusable for many low-volume applications where an observation of “0” for a small time interval should be acceptable, but an observation of “0” for many consecutive intervals should indicate an anomaly. Our work circumvents this deficiency, in part by defining a narrower problem.

3 Description of the Anomaly Analysis Model

3.1 Assumptions

We start with two assumptions:

1. Metrics count events that arrive independently
2. Under non-anomalous circumstances, the differences between predictions and observations at a given seasonal offset over several seasons can be described by a normal distribution about 0

The first assumption narrows the scope of our model, but it accurately describes many of our business-critical metrics. For example, each order that comes in is unrelated to other orders coming in; they represent independent buying decisions. We therefore restrict ourselves to analyzing *numbers* of actions, rather than *sizes* of actions (such as dollar amounts).

The second assumption is a common statistical assumption, used here for convenience. Note that we are assuming a normal distribution for the predictions associated with *each* seasonal offset. Their variances are unrelated. For example, under a daily cycle, the 5 PM predictions could be wildly inaccurate and the 6 PM predictions could be spot-on; we assume the former has a normal distribution with a large variance, and the latter a normal distribution with a small variance, both with averages of zero.

3.2 Interpretation of prediction and prediction deviation

Brutlag's implementation of Holt-Winters produces two important values for each point on the time horizon: a *prediction*, produced by an additive Holt-Winters model, and a *predicted deviation*, which is the exponentially smoothed difference between past predictions and past observations.

Given our assumptions, we interpret the prediction and predicted deviation to have special meaning. The prediction divided by the time interval to which it applies is interpreted, perhaps not surprisingly, as an average rate at which events are expected to arrive. The predicted deviation approximates a standard deviation of the accuracy of the prediction, weighted toward more recent observations. This follows from the similarity between the definition of the standard deviation and the prediction deviation. Define $x_{k,i}$ as the k^{th} observation of season i and $y_{k,i}$ as the k^{th} prediction of season i. The standard deviation of $x_k - y_k$ for some offset k examined over N seasons is:

$$\sigma_{k,N} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{k,i} - y_{k,i})^2 - \frac{1}{N}\left(\sum_{i=1}^{N}(x_{k,i} - y_{k,i})\right)^2} \tag{1}$$

Given assumption 2, that reduces to:

$$\sigma_{k,N} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{k,i} - y_{k,i})^2} = \sqrt{\langle (x_k - y_k)^2 \rangle} \tag{2}$$

And the prediction deviation d_k, measured at seasonal offset k, is described by a recursive formulation (Brutlag [1]):

$$\begin{aligned} d_{k,1} &= |x_{k,1} - y_{k,1}| \\ d_{k,i+1} &= (1-\gamma)(|x_{k,i+1} - y_{k,i+1}|) + \gamma d_{k,i} \end{aligned}$$

for some smoothing parameter $\gamma \in (0,1)$. Over N seasons, $d_{k,N}$ is then:

$$d_{k,N} = \gamma^{N-1}|x_{k,1} - y_{k,1}| + \sum_{i=2}^{N}\gamma^{N-i}(1-\gamma)|x_{k,i} - y_{k,i}| \tag{3}$$

which is actually a weighted average over $|x - y|$, because:

$$\gamma^{N-1} + \sum_{i=2}^{N}\gamma^{N-i}(1-\gamma) = \gamma^{N-1} + (1-\gamma)\sum_{i=0}^{N-2}\gamma^{i}$$
$$= \gamma^{N-1} + \sum_{i=0}^{N-2}\gamma^{i} - \sum_{i=1}^{N-1}\gamma^{i} = \gamma^{N-1} + \gamma^{0} - \gamma^{N-1} = 1$$

And so $d_{k,N}$ is a weighted average of $|x_k - y_k|$, while σ is equal to the RMS of $|x_k - y_k|$. Before moving on, we thus note that:

$$d_{k,N} \approx \sigma_{k,N} \tag{4}$$

3.3 Analysis of Brutlag's method

We can now, with greater yet imperfect precision, characterize circumstances under which Brutlag's method will completely fail, using the assumptions of 3.1.

Brutlag's method calls for choosing a "scaling factor" between 2 and 3; this scaling factor is multiplied by the prediction deviation, and the result is both added to and subtracted from the prediction in order to produce a "confidence band." If some specified number of measurements fall outside of this confidence band, the readings are said to be anomalous. We will describe when an infinite number of "0" readings will fail to be analyzed as anomalous.

Per assumption 1 of section 3.1, we are analyzing a Poisson process. Our observations therefore will follow a Poisson distribution, with variance equal to the mean, also known as the Poisson parameter. Since we interpret predictions as the expected average, a prediction y gives us a Poisson parameter $\lambda = y$.

Because the variance σ^2 of a Poisson distribution is equal to λ, we expect the prediction deviation $d \approx \sigma$ to be approximately equal to $\sqrt{y}$. Brutlag's confidence band will therefore encompass 0 and render the method nearly useless when $2\sqrt{y} \geq y \Rightarrow y \leq 4$, i.e., when the number of events predicted in an interval is about 4 or fewer.

3.4 The Model

The heart of the model is to treat incoming events as a Poisson process with an *uncertain* Poisson parameter, which we estimate with a normal distribution around $y_{k,i}$ (the prediction) with variance $d_{k,i}^2$ (the prediction deviation squared). We model the accuracy of the Poisson parameter with a normal distribution by assumption 2 of section 3.1. A more rigorous approach would use σ^2 instead of d^2 to represent the variance; however, storing σ^2 in RRDtool would require modifications in the RRDtool core, so we approximate σ^2 with d^2 per Equation (4).

More formally, given a prediction y with prediction deviation d, we calculate the probability p of observing some value x or lower than x as:

$$p_{y,d^2}(x) = \int_{-\infty}^{\infty} f(x;\lambda)\varphi_{y,d^2}(\lambda)\ d\lambda \tag{5}$$

where $f(x;\lambda)$ is a cumulative Poisson probability with parameter λ:

$$f(x;\lambda) = \sum_{i=0}^{x} \frac{e^{-\lambda}\lambda^i}{i!} \tag{6}$$

and $\varphi_{y,d^2}(\lambda)$ is the probability density of a normal distribution with average y and variance d^2 at value λ:

$$\varphi_{y,d^2}(\lambda) = \frac{1}{d\sqrt{2\pi}} \exp\left(-\frac{(y-\lambda)^2}{2d^2}\right) \tag{7}$$

But since the Poisson parameter must be positive, we cut off the distribution described by Equation (5) at $\lambda = 0$ and normalize thus:

$$p_{y,d^2}(x) = \frac{\int_0^{\infty} f(x;\lambda)\varphi_{y,d^2}(\lambda)\ d\lambda}{\int_0^{\infty} \varphi_{y,d^2}(\lambda)\ d\lambda} \tag{8}$$

The intuition for this model is as follows: because under non-anomalous conditions, the prediction often “misses” by some amount, we instead treat the prediction as a *range of possible predictions* described by the prediction deviation, and calculate the probability of seeing some observation for each prediction in the range. In other words, if an observation seems “off”, it could be that there is genuinely a problem; or it could be that the prediction is wrong. Our model takes both possibilities into account.

The original flaw we described in the Brutlag model is that a string of low-but-acceptable observations is treated no differently than a single low-but-acceptable observation. We now show how our model does not suffer from the same defect.

As observations come in, we perform the above analysis for T successively larger time intervals by consolidating small intervals. We will define a *health* value $h_{k,i}(T)$ as the lowest value of $p(x)$ that can be computed by consolidating up to T of the most recent observations. To consolidate intervals, we note that adding two normal distributions with averages μ_1 and μ_2 and variances σ_1^2 and σ_2^2 produces a normal distribution with average $\mu = \mu_1 + \mu_2$ and variance $\sigma^2 = \sigma_1^2 + \sigma_2^2$. Since our Poisson parameter is modeled as a normal distribution with average y and variance d^2, we consolidate a set of N predictions $y_1 \dots y_N$ and corresponding set of prediction variations $d_1 \dots d_N$ according to:

$$y = \sum_{i=1}^{N} y_i \qquad d^2 = \sum_{i=1}^{N} d_i^2 \tag{9}$$

We use these properties to formalize the health $h_{k,i}(T)$:

$$h_{k,i}(T) = \min_{t \in (1,\dots,T)} \left(p_{\sum_{j=1}^{t}(y_{k-j,i}),\sum_{j=1}^{t}(d_{k-j,i}^2)}(\sum_{j=1}^{T} x_{k-j,i}) \right) \tag{10}$$

In this way we can alert on a series of observations that, taken individually, would not be cause for alarm. As an example, consider two time intervals with $d = 0$, $x = 0$, $y = \lambda$. We can easily show that examining both intervals produces a health value $h_k(2)$ that is lower than examining just one interval, $h_k(1)$:

$$\begin{aligned}
h_k(1) &= p_{y,d^2}(x) = p_{\lambda,0}(0) = f(0;\lambda) = e^{-\lambda} \\
h_k(2) &= \min\left(p_{y,d^2}(x), p_{2y,(2d)^2}(2x)\right) = \min\left(p_{\lambda,0}(0), p_{2\lambda,0}(0)\right) \\
&= \min\left(f(0;\lambda), f(0;2\lambda)\right) = \min(e^{-\lambda}, e^{-2\lambda}) = e^{-2\lambda} < h_k(1)
\end{aligned}$$

If we set some alarm threshold A such that $h_k(1) > A > h_k(2)$, our model would characterize the one-interval case as non-anomalous and the two-interval case as anomalous.

4 Algorithmic analysis

Equation (6) can be calculated in time linear in x, where x is the size of the observation in a time interval; Equation (7) is constant time, and the integrals in Equation (8) can be computed with constant-time Riemann sums. The running time of calculating one value of $p(x)$ is therefore $O(x)$. If we run the calculations over T successively larger time intervals to find $h(T)$ by Equation (10), and if x is the average number of events in one interval, the average running time of computing one value of $p(x)$ is $O(Tx)$. Since calculating $h(T)$ requires computing T values of $p(x)$, the overall running time is $O(T \cdot Tx) = O(T^2x)$.

We might improve the running time by storing a table of values for Equation (6), rounding λ as appropriate. This optimization will reduce the cost of computing $p(x)$ to be $O(1)$, thus reduce the running time of computing $h(T)$ to $O(T)$. However, such a table requires $O(T^2x^2)$ memory.

5 Implementation

Thanks to the work of Brutlag [1], the Holt-Winters algorithm has previously been incorporated into the open-source application RRDtool. Brutlag implemented both the Holt-Winters forecasting as well as his own anomaly analysis into RRDtool. We have made slight modifications to the forecasting implementation within RRDtool, and have written stand-alone scripts to perform the anomaly analysis.

5.1 Modification to RRDtool

The Brutlag implementation calls for a periodic "smoothing" of predicted values. This smoothing is unrelated to seasonal exponential smoothing, but is instead a running average; some time after the first season, each prediction is replaced by an average of several surrounding predictions. Since "the seasonal effect is a smooth function over the period, not a discontinuous series of points,"[1] smoothing is designed to prevent an anomalous first observation from ruining predictions in that seasonal offset (note that the first observation is "special," as one can easily see from Equation (3); for small γ

it will influence predictions for much longer than subsequent observations). However, the size of this running average window was hard-coded to 5%. Since we use weekly seasons, this meant that each prediction averaged the surrounding 8 hours of observations, producing rather poor predictions. We have made this window a configurable parameter, and in practice we disable the running-average smoother altogether.

5.2 Stand-alone script

Our script analyzes several counters in a Holt-Winters-enabled Round-Robin Archive once per minute according to Equation (8), successively looking further and further into the past with the aid of Equations (9) and (9). We approximate the integrals using Riemann sums with a fixed number of points, setting the integral boundaries to be a few standard deviations on each side of y. The script retrieves prediction values from the HWPREDICT archive, and prediction deviations from the DEVPREDICT archive. Calculated health values are recorded in another Round-Robin Archive for retrieval by Nagios, and open-source alerting program. Nagios is configured to trigger an alarm if the health value ever drops below some pre-defined level. In practice, we find that threshold probabilities on the order of 10^{-5} minimize the number of false positives without creating too many false negatives.

6 Sample Data

Below we show an example of the model detecting a problem that the Brutlag method would have missed.

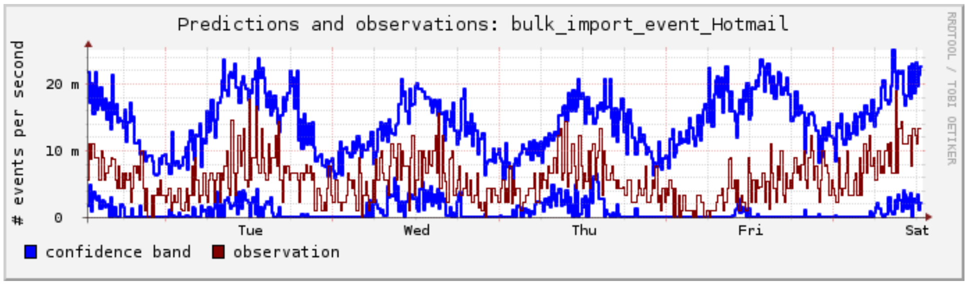

Figure 1: A set of readings on 17-19 October 2007, with confidence bands using the method of Brutlag [1].

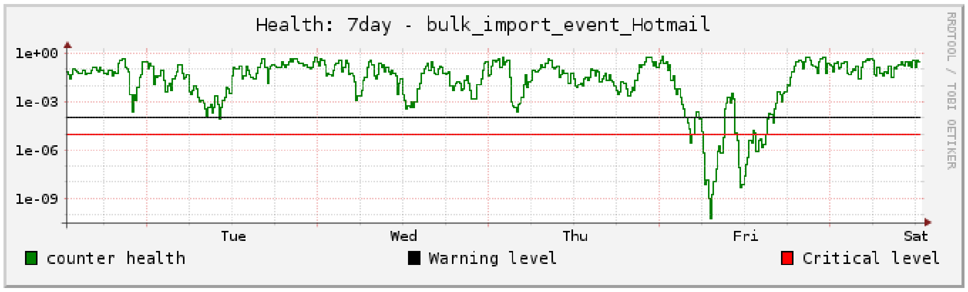

Figure 2: The health value calculated by our model, with pre-defined “warning” and “critical” levels; when the health dips below “critical”, an alert is triggered.

Figure 1 shows three days of observations for the number of IMVU users who invited their Hotmail contacts to open an account at IMVU. The line in maroon is a per-second invitation rate, with units $m = .001$. The blue lines represent the “confidence bands” described in Brutlag [1]. Because the maroon line does not cross the blue lines, an alert is not triggered in the time period shown.

Figure 2 shows how our model analyzed the data in Figure 1. At approximately 5 AM Friday, it first detects a problem, which persists most of the day. In fact, an external service provider had changed an interface early Friday morning, affecting some but not all of our users. The problem was not corrected until Friday afternoon.

7 Future Work

Probably the most needed improvement to our model is to resolve the problem with initial conditions. Because the first term in Equation (3) is not multiplied by γ like the rest of the terms, early predictions may vary widely based on the first set of observations (i.e., $|x_1 - y_1|$). A better model would take the variability of the first observations into account, rather than treat these observations as equivalent to a large set of exponentially-smoothed data. A statistical model would be preferable to the ad-hoc smoother described in 5.1.

Although we have restricted our model to Poisson processes, a model similar to ours might be used to resolve the many-zeroes problem of Brutlag’s method when analyzing non-Poisson processes. A simple extension to Brutlag’s method would be to consolidate time intervals using Equation (9), then apply the same confidence-band techniques described in his paper; the practicality of this approach should be explored further.

Our scripts do not use the dynamic programming technique described in section 4; an implementation would be necessary to analyze a large number of metrics, or metrics with large values. This optimization will require analyzing the tradeoff between space requirements and the degraded quality of calculation due to rounding.

As mentioned in 3.2, a more accurate model would use the actual standard deviation for each prediction's accuracy, rather than rely on the approximation in Equation (4). In practice, this change will require additional modifications to RRDtool, and has not seemed necessary.

8 Conclusion

In this paper, we demonstrated mathematically and with sample data why a popular anomaly-detection algorithm is unusable for independent events that occur roughly 4 or fewer times in an observation interval. We developed a statistical model for Poisson processes that exhibit seasonal fluctuations, and introduced an algorithm to trigger alerts based on the model. In its implementation at IMVU, the algorithm has quickly alerted engineers to application failures that would have gone unnoticed for much longer.

References

[1] Jake D. Brutlag. Aberrant behavior detection in time series for network monitoring. In *LISA*, pages 139–146, 2000.