Real-Time and Scalable Anomaly Detection using Parallel Regression

Derek Farren

WalmartLabs
derek@walmartlabs.com

Abstract

In this paper we propose Sherlock, a scalable real-time Anomaly Detector that has been tested in large production systems. This Anomaly Detector was developed in order to accurately detect anomalies in massive data streams that can not be monitored with traditional Anomaly Detection solutions because of to the big processing needs required by such streams. We show Sherlock's efficacy and speed by comparing it against other popular Anomaly Detection solutions.

I. Introduction

Anomaly detection is a well studied problem. However, most research focuses on either a very accurate, but not scalable model, or an inaccurate but scalable model. Sherlock aims to provide an accurate anomaly detector that is scalable. Our proposed algorithm is based on the state of the art Anomaly Detection algorithms, and it can run in parallel processors which makes it scalable.

The data we use is taken from the time series produced by monitoring the system. In order to simplify explanations this paper assumes that the time series are aggregated by minute, that the detector checks for anomalies every minute and that the model is retrained every 24 hours.

II. Related Work

The survey in [6] offers a good and comprehensive discussion about the state of the art in Anomaly Detectors. [7] has what we believe are the best Anomaly Detection methodologies for time series data, and [1] builds on to of it by proposing the use of Lasso in order to find causal relations between time series. [4] offers a scalable Anomaly Detector for time series, however its use is very specific. Our research is aimed to provide a scalable version of [1].

III. Sherlock

The time series extracted from a live system usually show correlations between each other. Our algorithm uses these correlations to model the system's behavior and check whether this behavior suddenly changes. If it does, we call this change an anomaly. This approach has been successfully explored by [1] and [7], so it is no surprise it worked well in this work.

The data size of the feature set for this supervised model is usually large. Since this model was developed to be used in retail, daily, monthly and yearly seasonality is important. A reasonable dataset size used in order to capture these seasonalities may have a few years worth of minute aggregated data. If we chose to find up to year long patterns (like holidays spikes for example), and we had 1,000 time series, our feature dataset would be composed of 525,600,000 features and 525,600 instances. This is a large, but most importantly, high dimensional dataset on which a supervised model can be hard to accurately train under time constrains.

In order to efficiently train the model we use Shotgun [3] and parallelize the training over a cluster of machines. Shotgun is a perfect algorithm for this because it parallelize the training by optimizing the objective function using several sets of random features at the

same time.

I. The model

Combining the ideas from previous Anomaly Detectors [7, 1, 6], time series modeling [11, 10, 9, 8] and Lasso scalability [3], we propose an auto regressive distributed lag model with L_1 regularization, that is fully scalable.

Let us define the following:

n = number of time series
$s_j^{(i)}$ = value of time series i at timestamp j
w = the window size used in the auto regression
h = the number of historic timestamps used to train the model

Our model is:

$$s^{(i)} = \sum_{j=0}^{wn} \beta_j s_{j \bmod w}^{(\lfloor \frac{j}{w} \rfloor)} + e_t \quad (1)$$

where $s^{(i)}$ is the next future value in time series i and e_t is white noise. The model is fit using least mean squares:

$$J(\beta) = \sum_{t=0}^{h} (\sum_{j=0}^{wn} \beta_j s_{j \bmod w}^{(\lfloor \frac{j}{w} \rfloor)} - s^{(i)})^2 + \lambda \|\beta\| \quad (2)$$

where J is the error function and λ is the L_1 regularization parameter.. The parameters of the model are:

$$\beta_j = \arg\min_{\beta_j} J(\beta) \quad (3)$$

This model offers a sparse regression that selects the timeseries that have predictive value and identifies the temporal correlation between these time series. The bias and size of the resulting model can be set by modifying λ. This is crucial since the smaller the model is, the less coefficients it has, and thus, the less data it needs to make predictions. Since retrieving data over a network could take several seconds, and low latency when catching anomalies is a priority in this algorithm, the size of the data to be retrieved should be as small as posible without greatly affecting the model's accuracy.

Training this model with a dataset like the one described in section III, can take days. However, we used Shotgun [3], an algorithm that parallelizes training for Lasso regression [5].

II. Catching anomalies

The model described in section I makes real time predictions of all time series in the system at the same time that it receives new data. In order to catch anomalies, we compared the prediction the model does with its real value coming from the data stream. The comparison is scored with a t-test and scores over threshold A are labeled as anomalies.

Algorithm 1 Sherlock

```
1:  procedure CHECK(threshold)
2:      for i ← 0, n do                          ▷ run parallel
3:          x^p ← predict(i)
4:          x^r ← real(i)
5:          if Ttest(x^p, x^r) > threshold then
6:              raiseAlarm()
7:          end if
8:      end for
9:  end procedure

10: procedure TRAIN(processors, features)
11:     while not converged do
12:         for p ← 0, processors do
13:             iterate(features)                ▷ run parallel
14:         end for
15:     end while
16: end procedure

17: procedure ITERATE(features)
18:     for i ← 0, features do
19:         j ← random(wn)
20:         δβ_j ← β_j
21:         if β_j < ▽J(β)_j then
22:             δβ_j ← ▽J(β)_j
23:         end if
24:         β_j ← β_j + δβ_j
25:     end for
26: end procedure
```

The pseudo algorithm of Sherlock is described in Algorithm 1. The *check*() procedure is called every minute and the *train*() procedure is called every 24hrs. The *iterate*() procedure runs in parallel in different processors and it is basically an adaptation of Shotgun [3].

IV. Results

We compared Sherlock performance with Dynamic Principal Components model (DPCA) [7] and with a Gaussian model [4]. We ran the models on a year worth of server logs data, aggregated as 100,000 time series. The data was labeled by humans as anomaly or not anomaly. We used an auto regression window size w of one week. The output of each model by minute (anomaly or not anomaly class) was compared against the real label of the data and we calculated the F1 score to assign the accuracy of each model.

Table 1: *accuracy*

Algorithm	F1 score
Sherlock	0.90
DPCA	0.92
Gaussian	0.33

Table 1 shows that even thought Sherlock's accuracy is below DPCA's accuracy, the gap is not much.

Table 2: *latency*

Algorithm	latency (millsecs)
Sherlock	12
DPCA	2,432
Gaussian	40

Table 2 shows that Sherlock is faster than the very accurate DPCA. It's eve faster than the Gaussian model, which is explained by the fact that Sherlock doesn't need to load all the data in the autoregression window, just the datapoints that are highly correlated.

V. Future work

Basically, Sherlock infers the correlation graph of the time series data using a L_1 regularized regression. One disadvantage of this is that we assume that correlations have a temporal causal relationship, which may not be the case. As a future improvement, we will test modeling the relations between the time series by using graph inferring techniques, like [2]. This may lead us to predict anomalies by modeling the cascades produced by an anomaly.

I. Subsection One

References

[1] Huida Qiu and Yan Liu. *Granger Causality for Time-Series Anomaly Detection*. IEEE International Conference on Data Mining, 2012.

[2] M. Gomez-Rodriguez, J. Leskovec and A. Krause. *Inferring Networks of Diffusion and Influence*. ACM Transactions on Knowledge Discovery from Data (TKDD), 2012.

[3] Joseph K. Bradley, Aapo Kyrola, Danny Bickson and Carlos Guestrin. *Parallel Coordinate Descent for L1-Regularized Loss Minimization*. Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, USA, 2011.

[4] Derek Farren. *Predicting retail website anomalies using Twitter data*. Stanford, 2011.

[5] Robert Tibshirani. *Regression Shrinkage and Selection via the Lasso*. Journal of the Royal Statistical Society. Series B (Methodogical), Volume 58, Issue 1, 1996, 267-288.

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. *Anomaly Detection: A Survey*. ACM Computing Surveys, 2009.

[7] L.H. Chiang, Richard D. Braatz, E.L. Russell. *Fault Detection and Diagnosis in Industrial Systems*. Springer Science & Business Media, 2001.

[8] Lutkepohl, H. *Vector Autoregressions*. unpublished manuscript, Institut fÃijr Statistik und Okonometrie, Humboldt-UniversitÃd't zu Berlin, 1999.

[9] Mills, T.C. *The Econometric Modeling of Financial Time Series, Second Edition*. Cambridge University Press, Cambridge, 1999.

[10] Runkle, D. E. *Vector Autoregressions and Reality*. Journal of Business and Economic Statistics, 5 (4), 437-442, 1987

[11] Sims, C.A. *Macroeconomics and Reality*. Econometrica, 48, 1-48, 1980.