2017 Evolving and Adaptive Intelligent Systems (EAIS)
Ljubljana, Slovenia, May 31 - June 2, 2017

Online Anomaly Detection on the Webscope S5 Dataset: A Comparative Study

Markus Thill and Wolfgang Konen
Department of Computer Science,
Cologne University of Applied Sciences,
51643 Gummersbach, Germany
{markus.thill, wolfgang.konen}@th-koeln.de

Thomas Bäck
Department of Computer Science,
Leiden University, LIACS,
2333 CA Leiden, The Netherlands
T.H.W.Baeck@liacs.leidenuniv.nl

***Abstract*—An unresolved challenge for all kind of temporal data is the reliable anomaly detection, especially when adaptability is required in the case of non-stationary time series or when the nature of future anomalies is unknown or only vaguely defined. Most of the current anomaly detection algorithms follow the general idea to classify an anomaly as a significant deviation from the prediction. In this paper we present a comparative study where several online anomaly detection algorithms are compared on the large Yahoo Webscope S5 anomaly benchmark. We show that a relatively *S*imple *O*nline *R*egression *A*nomaly *D*etector (SORAD) is quite successful compared to other anomaly detectors. We discuss the importance of several adaptive and online elements of the algorithm and their influence on the overall anomaly detection accuracy.**

I. Introduction

Anomaly detection in time series plays a more and more important role in many areas. Research institutions (e.g. high energy physics or astronomy) are collecting vast amounts of data. Industries have more and more devices (predictive maintenance in industry, sensors in the internet of things, or server technologies in cloud services of the internet) which are collecting increasing streams of data. To cope with this data, it is of importance to have automated procedures which separate the large amount of normal data from the anomalies, i.e. to have fast and reliable anomaly detection.

An anomaly is however difficult to define. In its most general form it is the absence of normality, but „normality“ depends largely on the context and cannot be expressed in *one* standard formula.

This is the reason why anomaly detection algorithms are very difficult to benchmark. An anomaly detection algorithm which performs very well on a certain benchmark dataset might perform surprisingly bad on another benchmark dataset. As an example, we will consider in this study Yahoo's well-known Webscope S5 dataset [1] with labeled anomalies of various kind.

The purpose of this paper is twofold: (1) We show that benchmarking anomaly detection algorithms is difficult with currently well-established benchmark datasets. As an example we found that the well-known algorithm NuPic (based on Hierarchical Temporal Memory - HTM), which shows remarkable successes on other benchmarks (e.g. NAB [2]), performs not so well on Yahoo's S5 dataset. From this point we will argue that there is a need for benchmark datasets better capturing the variety of anomalies in time series data. (2) Numerous applications call for fast yet reliable anomaly detection algorithms. We present here, as a preliminary study, a relatively simple one, the **S**imple **O**nline **R**egression **A**nomaly **D**etector (SORAD), which nevertheless shows good performance on the Yahoo Webscope S5 dataset. It is preliminary because SORAD needs to be refined, extended and tested on more diverse anomaly data benchmarks.

Our long-term goal is to extend SORAD by a variety of other simple online-adaptive algorithms which then form an ensemble of anomaly detection algorithms. Reehuis et al. [3], [4] discuss an interesting approach for novelty and interestingness in the context of optimization. This approach can be used for anomalies in time series as well: It is based on the dispersion of predictions within an ensemble and has – as an ensemble approach – the chance of increased robustness on a variety of benchmarks.

In this paper, we address the following research questions:

1) How does a simple online regression algorithm perform on the Webscope S5 dataset?
2) How do other algorithms perform on the same benchmark? In particular, we compare our new algorithm's performance to the following:
    - Numentas Hierarchical Temporal Memory [5]
    - Twitters ADVec algorithm [6]
3) How important is the online capability?

A. Related Work

Various surveys of anomaly detection techniques can be found in [7]–[10]. Many algorithms have been proposed over the years, but there is no clear gold-standard or *one* anomaly detection algorithm suitable for all kind of time series anomalies.

We compare in this work with two recent developments in online anomaly detection where the frameworks are available as open-source: (a) Hierarchical Temporal Memory (HTM) [2], [5], which is available as software NuPic from Numenta [1], and (b) Twitter's ADVec Algorithm [6], which is

[1] http://www.numenta.com

978-1-5090-6444-1/17/$31.00 ©2017 IEEE

available as open-source R package AnomalyDetection from Github [2].

The Yahoo S5 anomaly detection benchmark has been investigated in [11]–[13]. While [11] uses it for detecting whether a whole time series is anomalous, [12] presents an anomaly detection approach with echo state networks to which we will later compare. The algorithm introduced in [13] gives only results for FP (false positives) and is thus not well comparable.

II. Methods

We present in the following sections A–B the methods necessary for the offline variant of SORAD, which we call Offiline-RAD. Sections C–D cover the methods necessary to extend this offline variant to the online algorithm SORAD. Having both variants available allows us to precisely measure the effect of online adaptivity.

A. *Feature Generation using Sliding Windows*

In order to model temporal relationships in machine learning, a common approach is to employ a so called sliding window of a fixed length ℓ, which creates feature (input) vectors of length ℓ. When applied to a time series or sequence of length N in the form $(y_0, y_1, \dots, y_{N-1})$, the sliding window creates for each instance y_k a feature vector $\vec{x}_{k+1}$ consisting of a bias term and the ℓ previous instances, i. e. $\vec{x}_k = (1, y_k, y_{k-1}, \dots, y_{k-\ell+1})^T$. During the transient phase $(k < \ell)$ we pad values with negative indices with y_0. Matrix $\mathbf{X}$ is composed of the transposed inputs $\vec{x}_k$, one vector per row. The number of rows is the size K of the training set. Likewise, matrix $\mathbf{X}_{test}$ has $N - K$ rows, starting with vector $\vec{x}_K$.

B. *Offline Regression Anomaly Detection (Offline-RAD)*

The offline regression algorithm divides each time series into a training phase $t \leq K$ and a test or detection phase $t > K$. The general procedure is described in Algorithm 1. This algorithm requires a matrix inversion for each pass through the training data to build the model vector $\vec{\theta}$. When the parameters $\vec{\theta}$ are estimated, the prediction for new examples is computed with $\tilde{y}_k = \vec{\theta}^T \vec{x}_k$. The tuning of the regularization parameter ρ is done as follows: The $K = 500$ training data are divided further: For various values of ρ the training phase is done on the first 400 training data and the mean prediction error is measured on the remaining 100 validation data. Choosing the value ρ with the smallest prediction error, the final model is trained on all $K = 500$ training data.

C. *Online Estimation of a Distribution's Mean and Variance*

The naive approach for online estimation of a distribution's mean and variance from the sum of squares suffers from numeric instability. The Welford algorithm [14] proposes a numerically stable variant. It is used in this paper to estimate the parameters of the normal distribution.

Algorithm 1 Offline-RAD: Offline anomaly detection algorithm. **Input**: Time series (y_k), anomaly threshold ϵ, training set size $K = 500$. **Output**: Anomaly flags $\vec{a}_{flags}$.

```
Initialize:
Anomaly flags $\vec{a}_{flags} \leftarrow 0$                    ▷ Binary Vector
Tune regularization parameter $\rho$ (Sec. II-B).

Training phase:
Create training data $\mathbf{X}$ and $\vec{y}$ from the $K$ first time steps.
for $i = 1 \dots 2$ do          ▷ Two passes through training data
    $\vec{\theta} \leftarrow (\mathbf{X}^T\mathbf{X} + \rho\mathbf{I}_{\ell+1})^{-1}\mathbf{X}^T\vec{y}$
    $\vec{\delta} \leftarrow \vec{y} - \mathbf{X}\vec{\theta}$          ▷ Compute train prediction error
    Compute mean $\mu_\delta$ and std. deviation $s_\delta$ for $\vec{\delta}$
    Calculate the $\epsilon$-quantile $z_\epsilon$ of $\mathcal{N}(0, s_\delta^2)$ (see Fig. 1)
    $E \leftarrow [\mu_\delta - z_\epsilon,\ \mu_\delta + z_\epsilon]$
    for all $\{k \,|\, \vec{\delta}_k \notin E\}$ do
        Remove $k$-th row from $\mathbf{X}$ and $\vec{y}$
    end for
end for

Detection phase:
Create test data $\mathbf{X}_{test}$ and $\vec{y}_{test}$
$\vec{\delta}_{test} \leftarrow \vec{y}_{test} - \mathbf{X}_{test}\vec{\theta}$
for all $\{k \,|\, \vec{\delta}_{test,k} \notin E\}$ do
    $a_{flags,k} \leftarrow 1$                    ▷ Flag $y_{test,k}$ as anomalous
end for
```

As a new element, we introduce in this paper a forgetting factor for mean and variance as well. This is done by weighting the elements, e. g. for the sample variance we use

$$s_n^2 = \sum_{i=1}^{n} \frac{w_i (x_i - \mu_n)^2}{\sum_{i=1}^{n} w_i}, \tag{1}$$

where x_i is the ith instance in the sample and μ_n is the current sample mean. The weights

$$w_i = \lambda^{n-i} \tag{2}$$

decay exponentially so that historic elements contribute less to the sample variance s_n^2. A similar formula can be obtained for the sample mean μ_n. Both formulas can be combined with the Welford algorithm to get an online formulation of Eqs. (1) and (2). This is presented in Algorithm 3.

D. *SORAD*

The Offline-RAD algorithm has two main disadvantages: (a) It needs a training period (500 time steps in our application) before it can perform predictions, and (b) it does not learn any further in the detection phase. Both aspects call for an online version of this algorithm.

One of the most popular online models of the past few decades is the recursive least-squares (RLS) algorithm [15] from adaptive filter theory. The standard RLS formulation often includes a forgetting factor $\lambda \in (0, 1]$ that allows to deal with non-stationary systems to some extent [16], [17].

[2] http://github.com/twitter/AnomalyDetection

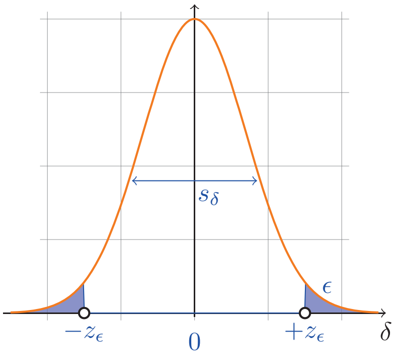

Figure 1. The ϵ-quantiles $\pm z_\epsilon$ of $\mathcal{N}(0, s_\delta^2)$ which define the border between anomalous and normal data in Offline-RAD and SORAD.

RLS is used in Algorithm 2 to estimate the model vector $\vec{\theta}$ recursively [17]

$$\mathbf{P} \leftarrow \frac{1}{\lambda}\mathbf{P} - \frac{1}{\lambda}\frac{\mathbf{P}\vec{x}_k\vec{x}_k^T\mathbf{P}}{1+\vec{x}_k^T\mathbf{P}\vec{x}_k} \tag{3}$$

$$\vec{\theta} \leftarrow \vec{\theta} + \delta_{k+1}\mathbf{P}\vec{x}_k \tag{4}$$

with prediction error δ_{k+1} and forgetting factor λ. Each input vector $\vec{x}_k$ is used only once. Matrix $\mathbf{P}$ is a recursive estimate of the covariane matrix $\mathbf{X}^T\mathbf{X}$. Having an online estimation of $\mathbf{P}$ makes it possible to introduce a forgetting factor λ which causes the algorithm to slowly fade out the long-ago parts of history and adapt the model to non-stationary elements in the time series. This is not possible in the offline version. Of course, a careful balancing between stability and plasticity of the model is necessary.

Another advantage of the online algorithm is that it does not need a regularization parameter ρ nor the complicated tuning of ρ for each time series anew. – Finally, the Welford algorithm (Sec. II-C) provides an online estimation for mean and variance of the error δ_{k+1} (Algorithm 2, step 20).

Algorithm SORAD has a short transient phase for $k < \ell$. In this period, where the input vector $\vec{x}_k$ needs to be partially padded (see Sec. II-A), the model vector $\vec{\theta}$ is left in its initial state. But the changes $\Delta\vec{\theta}$ according to Eq. (4) are accumulated separately and added to $\vec{\theta}$ at $k = \ell$. Likewise, the standard deviation s_δ is kept at ∞, leading to an inhibition of anomaly detection. The changes Δs_δ are accumulated and applied to s_δ at $k = \ell$. After the transient phase, $\vec{\theta}$ and s_δ are updated normally (Algorithm 2 and 3).

III. Experimental Setup

A. Yahoo's S5 Webscope Dataset

The Webscope S5 dataset is a relatively large anomaly detection benchmark which is publicly available [1]. It consists of 367 time series, each of length 1500, in four different classes A1/A2/A3/A4 with class counts 67/100/100/100. While class A1 has real data from computational services, classes A2, A3, and A4 contain synthetic anomaly data with increasing complexity. Example plots are shown in Figs. 2–5.

Algorithm 2 Pseudo code of SORAD. **Input:** Time series (y_k), anomaly threshold $\epsilon \in (0, 1)$, forgetting factor $\lambda \in (0, 1]$. Additionally, there is a short transient phase (see Sec. II-D).
Output: Anomaly flags $\vec{a}_{flags}$.

```
 1: Initialize and Transient Phase
 2: $\vec{\theta} \leftarrow \left(\theta_{Bias} \quad 2^{-1} \quad 2^{-2} \quad \cdots \quad 2^{-\ell}\right)^T$, with $\theta_{Bias} = 0$
 3: $(\mu_\delta, s_\delta^2) \leftarrow (0, \infty)$
 4: $\mathbf{P} \leftarrow 500\mathbf{I}_{\ell+1}$, with the identity matrix $\mathbf{I}_{\ell+1}$
 5: Anomaly flags $\vec{a}_{flags} \leftarrow 0$                          ▷ Binary Vector
 6:
 7: Set instance counter $k \leftarrow 0$
 8: while instance $y_{k+1}$ available do
 9:     $\vec{x}_{k+1} \leftarrow (1, y_k, y_{k-1}, \dots, y_{k-\ell+1})^T$
10:     $\tilde{y}_{k+1} \leftarrow \vec{\theta}^T\vec{x}_{k+1}$                      ▷ Predict next step
11:     Observe $y_{k+1}$
12:     $\delta_{k+1} \leftarrow y_{k+1} - \tilde{y}_{k+1}$             ▷ Compute prediction error
13:     Calculate the $\epsilon$-quantile $z_\epsilon$ of $\mathcal{N}(0, s_\delta^2)$ (see Fig. 1)
14:     $E \leftarrow [\mu_\delta - z_\epsilon,\ \mu_\delta + z_\epsilon]$
15:     if $\delta_{k+1} \notin E$ then
16:         $a_{flags,k+1} \leftarrow 1$             ▷ Flag $y_{k+1}$ as anomalous
17:         $k \leftarrow k + \ell - 1$                ▷ Skip next $\ell$ instances
18:     else
19:         Update $\mathbf{P}$ and $\vec{\theta}$ according to Eq. (3) and (4)
20:         Update $\mu_\delta, s_\delta^2$ with UPDATEESTIMATION($k, \delta_{k+1}$)
21:     end if
22:     $k \leftarrow k + 1$
23: end while
```

Algorithm 3 Online estimation of sample mean and sample variance for the prediction errors.

```
Initialize:
$\mu_\delta = 0,\ s_\delta^2 = 0,\ M_\delta = 0,\ \Sigma = 0$

function UPDATEESTIMATION($k, \delta_{k+1}$)
    $\Delta \leftarrow \delta_{k+1} - \mu_\delta$
    $\mu_\delta \leftarrow \mu_\delta + \frac{\Delta}{\lambda k+1}$
    $M_\delta \leftarrow \lambda M_\delta + \Delta \cdot (\delta_{k+1} - \mu_\delta)$
    $\Sigma \leftarrow \lambda\Sigma + 1$
    $s_\delta^2 \leftarrow \frac{M_\delta}{\Sigma}$                          ▷ Without bias-correction
end function
```

The ground truth anomaly information is available for all time series. We choose this benchmark dataset for our comparative study, because it is a relatively large benchmark which has, to the best of our knowledge, not yet been used with the algorithms NuPic and ADVec.

B. Algorithm Setup

All algorithms (except Offline-RAD) perform online on the time series.

1) Offline-RAD: We use a window size of $\ell = 10$. The first $K = 500$ instances of each time series are used for training, the remaining 1000 instances for testing (detection).

2) Setup of Numenta's NuPic (HTM): To verify the correctness of our NuPic installation, we applied it to the Numenta

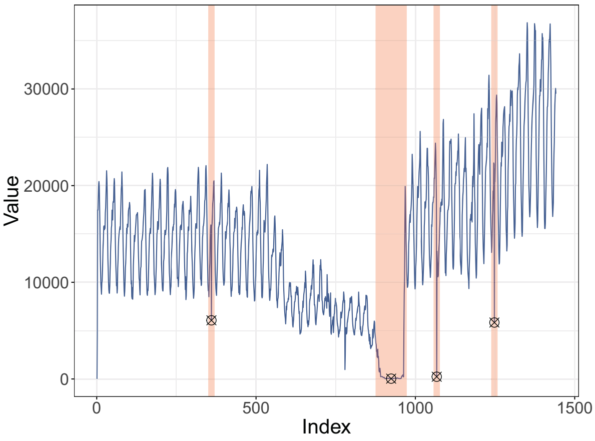

Figure 2. Example plot for the A1 data. Black crosses mark the true anomalies and the red vertical bars indicate their anomaly windows. The time series of the A1 data were taken from real world applications and anomalies were manually labeled.

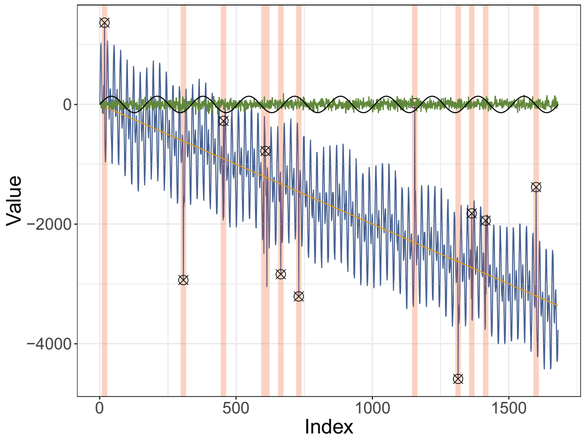

Figure 4. Example plot for the A3 data. The time series of the A3 data (blue) were generated synthetically. The time series include a trend, three seasonal (periodic) components and noise (green). The anomalies were added at random instances.

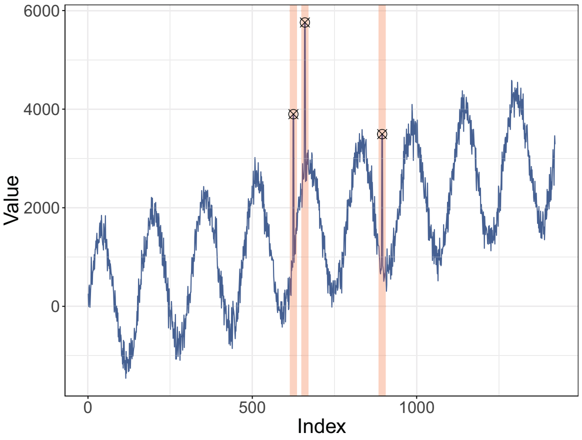

Figure 3. Example plot for the A2 data. The time series of the A2 data were generated synthetically. The time series include a trend, a seasonal (periodic) component and noise. The anomalies were added at random instances.

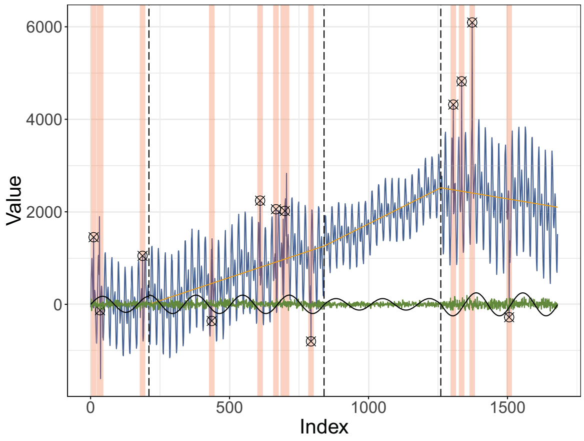

Figure 5. Example plot for the A4 data (blue). Similar to the A3 data. Additionally, change point anomalies are introduced (indicated by the vertical dashed lines) in which the characteristics (such as slope or frequency) of the individual components are changed.

Anomaly Benchmark (NAB) [2] using the standard parameter settings and confirmed that we can exactly reproduce the results[3] published in [2]. In a first round of experiments we used the same parameter setting for the S5 datasets. In a second round we used NuPic's so called swarming algorithm [18], a tool to aid automatic parameter search for a given dataset. The results were very similar, so we list in the following only the results for the standard parameter settings.

3) Setup of Twitter's ADVec Algorithm: The ADVec algorithm has two parameters. The first parameter α describes the level of statistical significance with which to accept or reject anomalies. Similar to an anomaly threshold, this parameter trades off false-positives and false-negatives. ADVec requires a second parameter, a period-length, which we fix to the value 40 in all experiments[4].

C. Algorithm Performance Measures

Similar to a typical binary classification tasks, each instance in a time series can be labeled as either anomalous or non-anomalous. For time series taken from real world problems the anomalies are often labeled manually. This might lead to some inaccuracies in the labeling process, since experts (e.g., based on knowledge in the domain) might interpret certain instances differently. For artificial time series the labeling process can normally be automatized and is mostly very accurate.

In our setup, the real anomaly labels are not passed to the algorithms at any time (i.e., the task is treated as an unsupervised learning task like in a real-world application setting). Hence, each anomaly detection algorithm has to learn

[3] The result files can be obtained from: https://github.com/numenta/NAB/tree/master/results/numenta.

[4] We tuned this parameter over a wide range for all four datasets.

the normal structure of a time series and has to be able to identify anomalies among the observed time series values.

A misclassification occurs if the anomaly detection algorithm fails to either flag a real anomaly as anomalous or if it erroneously flags normal instances as anomalous. The first case is a false-negative (FN) or type II error and the second case is a false-positive (FP) or type I error. Similarly, correctly identified anomalies are commonly referred to as true-positives (TP) and correctly identified normal instances as true-negatives (TN).

In our experiments, the scoring of an individual algorithm is done as follows: Each instance in the time series is evaluated and tagged as TP, TN, FP or FN. This is done with the help of so called anomaly windows (see Sec. III-D). Derived performance measures are

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$

Since there is a tradeoff between both measures, depending on the alarm threshold of the algorithm, one often considers the so-called F1 score which is the harmonic mean between precision and recall

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{7}$$

The F1 score is relatively independent to the threshold.

Another algorithm performance measure is its computation time.

D. Anomaly Windows

Anomaly windows span for the duration of the true anomalies plus 5 time steps before and thereafter. The scoring process for anomaly detection is depicted in Fig 6. Detections that fall into the anomaly windows are considered as TP. However, only the first TP in each window is counted. The remaining detections in the same window are ignored. If an algorithm fails to flag any instance inside an anomaly window as anomalous, this will be considered as one FN. All detections outside the window are considered as FP, even if the anomaly detection algorithm generates a burst of FPs in a short interval.

IV. Results

Table I summarizes the results of all algorithms running on the four datasets A1–A4. To have a fair comparison to Offline-RAD, we include for all algorithms only time steps $t > 500$ (after the Offline-RAD training phase) into the anomaly detection phase. Apparently, SORAD has a better performance than Offline-RAD and both are on most datasets significantly better than NuPic and ADVec in nearly all performance measures.

The great advantage of online algorithms is of course that they are much faster up-and-running. We show in Table II the results when including all time steps $t > \ell$ beyond the transient phase in the anomaly detection. Basically, the results are very similar. This means that the ‚anytime-ready‘ feature of online-algorithms does not severely influences accuracy.

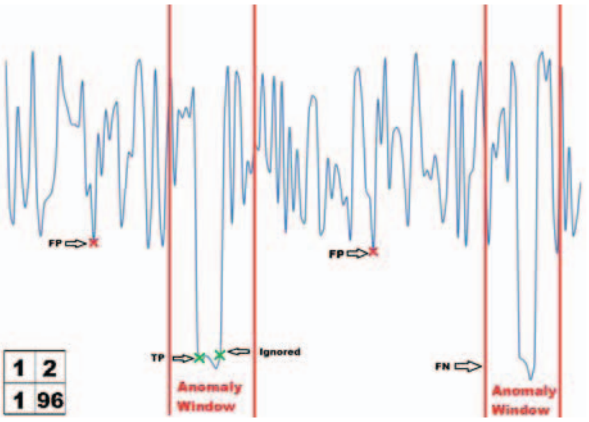

Figure 6. Example of a scoring process for an anomaly detection algorithm, using anomaly windows.

Fig. 7 shows an example time series from the A4 dataset with the anomlies detected by the various algorithms included. It is seen that NuPic and ADVec produce a number of false-positives (FP).

Anomaly detection algorithms always have a threshold parameter which allows the user to control the trade-off between FP and FN. Fig. 10 shows that the performance measure F1 is very stable over several threshold decades.[5] The multiobjective plot (where the measures FP and FN are plotted against each other) in Fig. 8 varies these thresholds and shows the multiobjective front for each algorithm. It is seen that both RAD algorithms dominate the others irrespective of the chosen threshold. SORAD is significantly better than Offline-RAD.[6] This might be due to the fact that the online algorithm continues to learn, which is advantageous for non-stationary time series.

We test in Fig. 8 the one-pass and the two-pass variant of Offline-RAD, showing that there is no significant difference between both. This means that the poorer performance of Offline-RAD (as compared to SORAD) is *not* due to the occasional presence of anomalies in the training phase.

All of these results were obtained with a forgetting factor $\lambda = 1$, that is, with no forgetting. If a time series is non-stationary, then anomaly detection might benefit from a certain degree of forgetting the past values. We show in Fig. 11 the effect of varying the forgetting factor in the range [0.8,1.0]. Additionally, Table II shows the overall results for two SORAD variants with forgetting: While forgetting in the RLS part (SORAD-F) does not change much, a forgetting mechanism additionally for the online estimation of the error distribution (μ_δ, s_δ) is of great benefit (SORAD-FMS), especially for the datasets A1 and A4.

Table III shows the computation times for all algorithms.

[5] Fig. 10 shows the stability of F1 for SORAD (without forgetting), but it holds the same way for the SORAD-variants with forgetting.

[6] Additionaly, the variation of the threshold produces for SORAD a more diverse population on the multiobjective front with no 'holes' as in Offline-RAD.

Table I

Results for various algorithms on the Yahoo S5 datasets A1–A4. Shown are TP, FP, FN for each dataset (sum over all time series, $t > 500$) and the quantities precision, recall and F1 for each dataset (over all time series, $t > 500$). All algorithms have their threshold chosen such that FP $\approx$ FN. (Only for the F1 score in brackets the threshold is chosen such that F1 is maximized.)

		Dataset			
Algorithm	Measure	A1	A2	A3	A4
Offline RAD	TP, FP, FN	73, 101, 54	197, 3, 3	603, 18, 37	312, 382, 395
	Precision, Recall	0.42, 0.57	0.98, 0.98	0.97, 0.94	0.45, 0.44
	F1 score	0.49 (0.53)	0.98 (0.99)	0.96 (0.96)	0.45 (0.48)
Offline RAD 2-Pass	TP, FP, FN	73, 107, 54	197, 3, 3	615, 36, 25	306, 394, 401
	Precision, Recall	0.41, 0.57	0.98, 0.98	0.94, 0.96	0.44, 0.43
	F1 score	0.48 (0.5)	0.98 (0.99)	0.95 (0.96)	0.43 (0.49)
SORAD	TP, FP, FN	85, 43, 42	197, 0, 3	627, 16, 13	460, 272, 247
	Precision, Recall	0.66, 0.67	1, 0.98	0.98, 0.98	0.63, 0.65
	F1 score	**0.67** (0.67)	**0.99** (0.99)	**0.98** (0.98)	**0.64** (0.66)
NuPic	TP, FP, FN	67, 57, 60	91, 102, 109	151, 465, 489	109, 677, 598
	Precision, Recall	0.54, 0.53	0.47, 0.46	0.25, 0.24	0.14, 0.15
	F1 score	0.53 (0.55)	0.46 (0.48)	0.24 (0.26)	0.15 (0.19)
ADVec	TP, FP, FN	60, 62, 67	114, 65, 86	165, 458, 475	112, 578, 595
	Precision, Recall	0.49, 0.47	0.64, 0.57	0.26, 0.26	0.16, 0.16
	F1 score	0.48 (0.48)	0.6 (0.64)	0.26 (0.29)	0.16 (0.17)

Table II

Same as Table I, but now the detection phase is larger (all time steps after the transient phase) and only the online algorithms (SORAD in different variants, NuPic, ADVec) are compared. Similar accuracy as in Table I, but the online algorithms are faster up-and-running. For SORAD-F and SORAD-FMS the forgetting factor is fixed to $\lambda = 0.98$ for all experiments.

		Dataset			
Algorithm	Measure	A1	A2	A3	A4
SORAD	TP, FP, FN	94, 62, 58	197, 0, 3	877, 24, 28	648, 301, 331
	Precision, Recall	0.6, 0.62	1, 0.98	0.97, 0.97	0.68, 0.66
	F1 score	0.61 (0.62)	**0.99** (0.99)	**0.97** (0.97)	0.67 (0.67)
SORAD-F	TP, FP, FN	95, 59, 57	197, 1, 3	888, 51, 17	672, 331, 307
	Precision, Recall	0.62, 0.62	0.99, 0.98	0.95, 0.98	0.67, 0.69
	F1 score	0.62 (0.63)	**0.99** (0.99)	0.96 (0.96)	0.68 (0.68)
SORAD-FMS	TP, FP, FN	98, 52, 54	197, 1, 3	855, 24, 50	796, 289, 183
	Precision, Recall	0.65, 0.64	0.99, 0.98	0.97, 0.94	0.73, 0.81
	F1 score	**0.65** (0.66)	**0.99** (0.99)	0.96 (0.96)	**0.77** (0.83)
NuPic	TP, FP, FN	69, 110, 83	91, 102, 109	177, 816, 728	129, 1034, 850
	Precision, Recall	0.39, 0.45	0.47, 0.46	0.18, 0.2	0.11, 0.13
	F1 score	0.42 (0.5)	0.46 (0.48)	0.19 (0.2)	0.12 (0.15)
ADVec	TP, FP, FN	81, 171, 71	114, 93, 86	241, 668, 664	147, 844, 832
	Precision, Recall	0.32, 0.53	0.55, 0.57	0.27, 0.27	0.15, 0.15
	F1 score	0.4 (0.4)	0.56 (0.59)	0.27 (0.3)	0.15 (0.16)

V. Discussion

1) Transient Phase: As described in Sec. II-D, algorithm SORAD needs a short transient phase. During this phase the model vector changes are accumulated and no anomaly detection is allowed. It has proven to be adversial to start directly with the recursive procedure while the sliding window is not yet filled completely with true data. Our experiments have shown that in such a case the estimation of the vector $\vec{\theta}$ tends to be unstable and the anomaly error rate increases.

2) Forgetting Factor: Online algorithms open the possibility to add a certain degree of forgetting. Our results have shown that the usual forgetting element in the RLS part (SORAD-F) does not play a significant role. But the new element to add forgetting to the estimation of the error distribution (SORAD-FMS) has proven beneficial in the datasets A1 and A4. The increase in F1 score (13%) is most prominent for dataset A4 which is the dataset with the largest non-stationary elements. Note that all datasets in the Yahoo S5 benchmark are relatively short (1500 time steps), thus putting a boundary on the degree of non-stationarity one can observe. For longer time series the effect of having or not having a forgetting factor can be much larger.

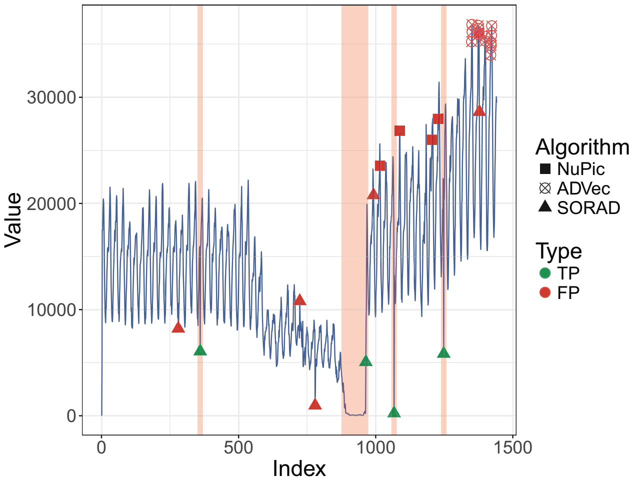

Figure 7. Example time series taken from data set A1 with the anomalies detected by the various algorithms SORAD, HTM (NuPic) and ADVec. The red vertical bars in the plot indicate the true anomaly windows.

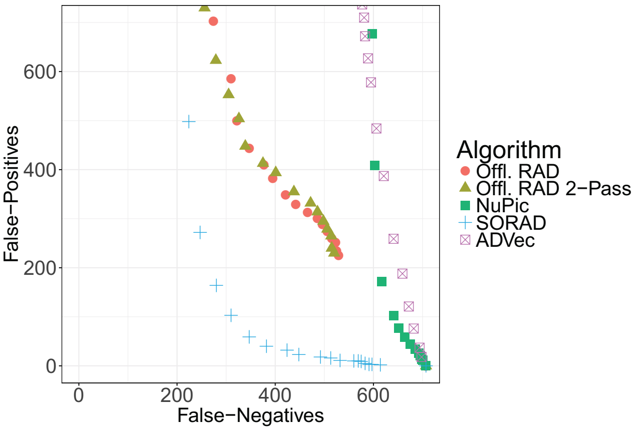

Figure 8. Multiobjective plot for different algorithms and thresholds for $t > 500$. Here, the results for the A4 data is shown. The FNs and FPs are the sum over the 100 time series of the A4 data.

Table III
COMPUTATION TIMES OF THE ALGORITHMS ON DATASETS A1–A4. SHOWN IS THE AVERAGE AND STANDARD DEVIATION FROM 20 RUNS EACH. THE RUNS WERE PERFORMED ON A PC WITH AN I7-3520M CPU AND 8GB OF RAM.

	Computation Time (s)			
Algorithm	A1	A2	A3	A4
Offline RAD	7.7 ± 0.1	11.6 ± 0.1	12.9 ± 0.1	12.7 ± 0.1
SORAD-FMS	21.1 ± 0.1	31.8 ± 0.1	35.8 ± 0.1	36.3 ± 0.1
NuPic	368 ± 5	693 ± 2	813 ± 3	828 ± 4
ADVec	3.3 ± 0.1	4.8 ± 0.2	5.6 ± 0.4	6.0 ± 0.7

3) Detection Accuracy: The most striking result of this comparative study is that our relatively simple regression anomaly detection works better on all datasets A1–A4 than NuPic or ADVec. It has to be said that NuPic has a large number of parameters and we cannot exclude with certainty the possibility that there might be another parameter set leading to better results for NuPic. But at least we can say that it must be hard to find, since even the parameter optimization procedure built into NuPic (swarming algorithm) did not reveal such a parameter set.

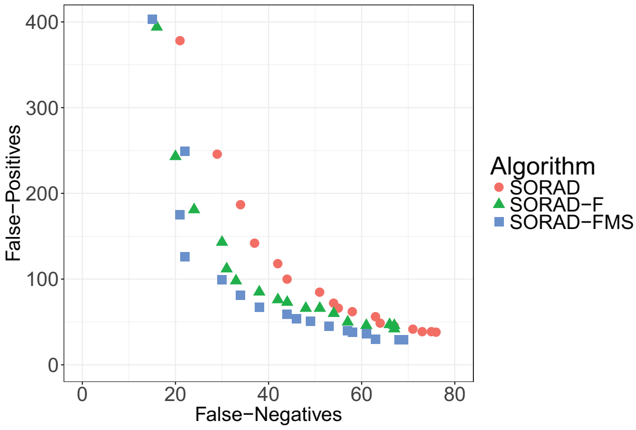

Figure 9. Multiobjective plot for different SORAD variants and thresholds for all t after the transient phase. Here, the results for the A1 data is shown. The FNs and FPs are the sum over the 100 time series of the A1 data.

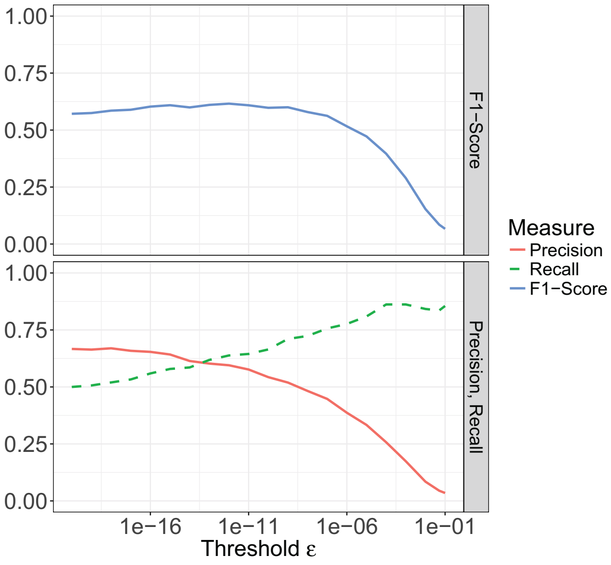

Figure 10. The performance of SORAD (without forgetting) over a wide range of thresholds for the A1-data. The F1 score is virtually constant for a wide range $\epsilon \in [1e-17, 1e-9]$.

4) Limitations of SORAD: Initially, we intended SORAD to act as a baseline algorithm (to show how far simple algorithms get on a certain benchmark and how much further more advanced algorithms would lead). It was a surprise for us that SORAD actually performed better on A1–A4. We are led to the conclusion that anomaly detection benchmarks with a larger variety of anomalies are important for proper benchmarking.

We do *not* claim that SORAD is for all time series the

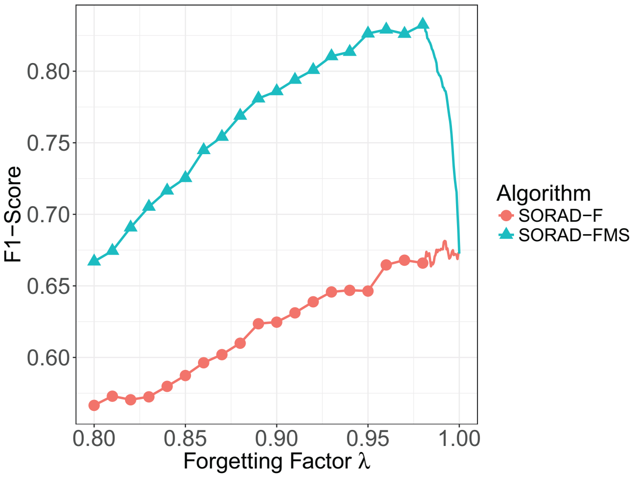

Figure 11. Comparing results on A4 for various forgetting factors of SORAD. The first curve SORAD-F shows the results for SORAD with forgetting in RLS; the forgetting factor $\lambda = 0.992$ results in the highest F1 score of $F1 = 0.68$ in this case. For the second curve SORAD-FMS, the forgetting is applied to the estimation of μ_δ and s_δ of the error signal distribution as well. The best F1 score on the A4-data with a value of $F1 = 0.83$ is reached for a forgetting factor of $\lambda = 0.9805$.

better anomaly detector. It has not enough memory for more complicated long-term interactions. On the Numenta anomaly detection benchmark NAB [2], where NuPic achieves very good results (F1=0.51), SORAD does not perform too well in its current form (F1=0.28). Further research effort is needed here.

5) Other algorithms: Suh et al. [12] propose an echo state network approach to anomaly detection and test it on Webscope S5, but only for the A1 dataset. Besides methodological issues (they devote 44 complete time series to training, 11 to validation, omit the 45th time series and perform their evaluation only on the remaining 11 time series, without cross validation), their results for precision / recall / F1 = 0.54 / 0.51 / 0.52 are inferior to the results of SORAD.

VI. Conclusion

In concluding this paper, we refer to our research questions by providing the following summarizing answers:

(1) A simple online regression anomaly detector (SORAD) performs surprisingly well on the Webscope S5 anomaly benchmark dataset. (2) It outperforms other anomaly detections algorithms (NuPic, ADVec) on these datasets. This is at least true if those algorithms are used with their standard parameter settings. A search for better parameter settings for NuPic by using its own parameter optimization routine did not reveal significantly different results.

(3) Our third research question on the importance of the online capability is answered as follows: We compared algorithm SORAD with its offline sibling (Offline-RAD) and could thus assess in this comparative study the differences between both: The online variant is clearly superior to the offline variant (around 40% increase in F1 score on datasets A1 and A4). We showed that it is important to make *all* elements of the algorithm adaptive, including the parameters of the error distribution. This underpins the importance of building *online and adaptive* anomaly detection algorithms to cope successfully with today's large data streams.

A. Future Work

It is necessary to build up more diverse anomaly detection benchmarks and to collect comprehensive results of algorithms working on them. As said before, our algorithm SORAD is not yet good on all anomaly benchmarks. For example, it does not work too well on the Numenta Anomaly Benchmark (NAB). In the future we want to improve SORAD or to combine it with further anomaly detectors leading to better performance through the ensemble approach.

References

[1] N. Laptev and S. Amizadeh, "Yahoo anomaly detection dataset S5," 2015. [Online]. Available: http://webscope.sandbox.yahoo.com/catalog.php?datatype=s\&did=70

[2] A. Lavin and S. S. Ahmad, "Evaluating real-time anomaly detection algorithms – the Numenta anomaly benchmark," in *14th International Conference on Machine Learning and Applications (IEEE ICMLA15)*, 2015. [Online]. Available: http://arxiv.org/pdf/1510.03336

[3] E. Reehuis, M. Olhofer, M. Emmerich, B. Sendhoff, and T. Bäck, "Novelty and interestingness measures for design-space exploration," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 1541–1548.

[4] E. Reehuis, M. Olhofer, B. Sendhoff, and T. Bäck, "Learning-guided exploration in airfoil optimization," in *Int. Conf. on Intelligent Data Engineering and Automated Learning*. Springer, 2013, pp. 505–512.

[5] D. George and J. Hawkins, "Towards a mathematical theory of cortical micro-circuits," *PLoS Comput Biol*, vol. 5, no. 10, p. e1000532, 2009.

[6] O. Vallis, J. Hochenbaum, and A. Kejariwal, "A novel technique for long-term anomaly detection in the cloud," in *6th USENIX Workshop on Hot Topics in Cloud Computing, Philadelphia, PA*, 2014.

[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[8] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[9] M. Agyemang, K. Barker, and R. Alhajj, "A comprehensive survey of numeric and symbolic outlier mining techniques," *Intelligent Data Analysis*, vol. 10, no. 6, pp. 521–538, 2006.

[10] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Comput. Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[11] R. J. Hyndman, E. Wang, and N. Laptev, "Large-scale unusual time series detection," in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, 2015, pp. 1616–1619.

[12] S. Suh, D. H. Chae, H.-G. Kang, and S. Choi, "Echo-state conditional variational autoencoder for anomaly detection," in *Neural Networks (IJCNN), 2016 Int. Joint Conf. on*. IEEE, 2016, pp. 1015–1022.

[13] O. Ibidunmoye, T. Metsch, and E. Elmroth, "Real-time detection of performance anomalies for cloud services," in *Quality of Service (IWQoS), 2016 IEEE/ACM 24th Int. Symposium on*. IEEE, 2016, pp. 1–2.

[14] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[15] S. Haykin, *Adaptive filtering theory*. Pearson, 5th edition, 2013.

[16] S. van Vaerenbergh, I. Santamaría, and M. Lázaro-Gredilla, "Estimation of the forgetting factor in kernel recursive least squares," in *2012 IEEE International Workshop on Machine Learning for Signal Processing*. IEEE, 2012, pp. 1–6.

[17] G. Bontempi and S. B. Taieb, "Statistical foundations of machine learning, chapter: Recursive least squares," March 2016. [Online]. Available: https://www.otexts.org/1582

[18] S. Ahmad. (2015, December) Running swarms. [Online]. Available: https://github.com/numenta/nupic/wiki/Running-Swarms