Chapter 4

SORAD: A Simple Online Regression Anomaly Detection Algorithm

4.1 Introduction

A challenging aspect of time series anomaly detection is working in environments with non-stationary behavior or unknown or vaguely defined nature of future anomalies. Most of the current anomaly detection algorithms follow the general idea to classify an anomaly as a significant deviation from the prediction. This chapter presents a comparative study where several online anomaly detection algorithms are compared on the 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 detection algorithms. We discuss the importance of several adaptive and online elements of the algorithm and their influence on the overall anomaly detection accuracy.

In many applications, data is collected continuously sequentially in the form of data streams or time series. A common characteristic of data streams is the inherent non-stationarity. In a vast amount of real-world problems, data is generated by non-stationary processes. Typically, certain properties of streams or time series (such as trends, noise, periodicity, and other parameters) change over time.

Machine learning tasks involving such non-stationary streaming data are considered challenging since many classical learning approaches are not applicable due to their offline character: offline learning algorithms are trained on a whole batch of data. They require a complete repetition of the training procedure if new examples are added to the data set. Due to memory and time constraints, such approaches are typically infeasible for problems with streaming data. In such cases, it is necessary to operate in an online setting on the data and process the data in an example-by-example manner and incrementally learn from every new example without re-training an entirely new model each time.

Especially in non-stationary environments, an anomaly is difficult to define. In its most general form, it is the absence of normality, but „normality“ depends mainly on the (current) 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 that performs very well on a particular benchmark dataset might perform surprisingly poorly on another benchmark. As an example, we will consider in this chapter Yahoo's well-known Webscope S5 dataset [87] with labeled anomalies of various kinds.

This chapter's purpose 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 [89]), 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.

In this chapter, 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 Numenta's NuPIC algorithm[160, 49] and Twitter's ADVec algorithm [173].
3. How important is the online capability?

4.1.1 Related Work

In this chapter, we compare with two developments in online anomaly detection where the frameworks are available as open-source: (a) Hierarchical Temporal Memory (HTM) [49, 89], which is available as software NuPIC from Numenta [1], and (b) Twitter's ADVec Algorithm [173], which is available as open-source R package AnomalyDetection from Github [2]. Both algorithms are described in more detail in chapter 3.

The Yahoo S5 anomaly detection benchmark has been investigated in [68, 154, 69, 115]. While [68] uses it for detecting whether a whole time series is anomalous, [154] presents an anomaly detection approach with echo state networks to which we will later compare. The algorithm introduced in [69] gives only results for FP (false positives) and is thus not well comparable. In [115], the CNN-based algorithm DeepAnT is introduced and evaluated on the Yahoo S5 benchmark. The performance of DeepAnT for A1–A4 is later compared to SORAD's.

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

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

4.2 Methods

In the following sections (4.2.1 – 4.2.2), we present the methods necessary for the offline variant of SORAD, which we call Offline-RAD. Sections 4.2.3 – 4.2.4 cover the methods necessary to extend this offline variant to the online algorithm SORAD. Having both variants available allows us to measure the effect of online adaptivity precisely.

4.2.1 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 $\mathbf{x}_{k+1}$ consisting of a bias term and the ℓ previous instances, i. e. $\mathbf{x}_k = (1, y_k, y_{k-1}, \dots, y_{k-\ell+1})^T$. During the transient phase ($k < \ell$) we pad values with negative indices with y_0. Matrix $\mathbf{X}$ is composed of the transposed inputs $\mathbf{x}_k$, 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 $\mathbf{x}_K$.

4.2.2 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 $\boldsymbol{\theta}$. When the parameters $\boldsymbol{\theta}$ are estimated, the prediction for new examples is computed with $\tilde{y}_k = \boldsymbol{\theta}^T \mathbf{x}_k$. 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. After choosing the value ρ with the smallest prediction error, the final model is trained on all $K = 500$ training data. We perform two passes through the data in order to make the training data approximately anomaly-free.

4.2.3 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 [176] proposes a numerically stable variant. A variant of it is used in this chapter to estimate the parameters of the normal distribution.

As a new element, we introduce a forgetting factor for mean and variance. The forgetting is realized by weighting the elements, e. g., for the sample variance we use

$$s_n^2 = \sum_{i=1}^{n} \frac{w_i (x_i - \mu_n)^2}{\sum_{i=1}^{n} w_i}, \tag{4.1}$$

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

```
Initialize:
Anomaly flags $\mathbf{a}_{flags} \leftarrow 0$                              ▷ Binary Vector
Tune regularization parameter $\rho$ (Sec. 4.2.2).

Training phase:
Create training data $\mathbf{X}$ and $\mathbf{y}$ from the $K$ first time steps.
for $i = 1 \dots 2$ do                                  ▷ Two passes through training data
    $\boldsymbol{\theta} \leftarrow (\mathbf{X}^T\mathbf{X} + \rho\mathbf{I}_{\ell+1})^{-1}\mathbf{X}^T\mathbf{y}$
    $\boldsymbol{\delta} \leftarrow \mathbf{y} - \mathbf{X}\boldsymbol{\theta}$                          ▷ Compute train prediction error
    Compute mean $\mu_\delta$ and standard deviation $s_\delta$ for $\boldsymbol{\delta}$
    Calculate the $\epsilon$-quantile $z_\epsilon$ of $\mathcal{N}(0, s_\delta^2)$ (see Fig. 4.1)
    $E \leftarrow [\mu_\delta - z_\epsilon,\ \mu_\delta + z_\epsilon]$
    for all $\{k \mid \boldsymbol{\delta}_k \notin E\}$ do
        Remove $k$-th row from $\mathbf{X}$ and $\mathbf{y}$

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

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

$$w_i = \lambda^{n-i} \tag{4.2}$$

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 modified Welford algorithm to get an online formulation of Eqs. (4.1) and (4.2). This is presented in Algorithm 2.

One can estimate the memory (the number of time steps after an observation is "forgotten" again) with the formula

$$n_{mem} \approx \frac{1+\lambda}{1-\lambda},$$

as shown in Appendix B.2.4. A thorough derivation of the online weighted estimation of mean and (co-) variance can be found in Appendix B.2.

4.2.4 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.

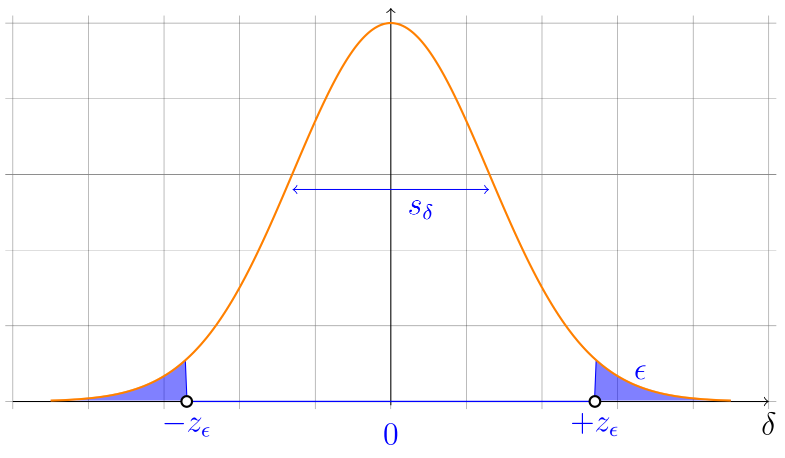

Figure 4.1: The ϵ-quantiles $\pm z_\epsilon$ of $\mathcal{N}(0, s_\delta^2)$ which define the border between anomalous and normal data in Offline-RAD and SORAD.

One of the most popular online models of the past few decades is the recursive least-squares (RLS) algorithm [59] 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 [172, 17]. RLS is used in Algorithm 3 to estimate the model vector $\boldsymbol{\theta}$ recursively [17]:

$$\mathbf{P} \leftarrow \frac{1}{\lambda}\mathbf{P} - \frac{1}{\lambda} \cdot \frac{\mathbf{P}\mathbf{x}_k\mathbf{x}_k^T\mathbf{P}}{1+\mathbf{x}_k^T\mathbf{P}\mathbf{x}_k} \tag{4.3}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta_{k+1}\mathbf{P}\mathbf{x}_k \tag{4.4}$$

with prediction error δ_{k+1} and forgetting factor λ. Each input vector $\mathbf{x}_k$ is used only once. A RLS variant for mini-batch updates (updating the parameters $\boldsymbol{\theta}$ with more than one example) is derived in Appendix B.1 and can also be used in practice.

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.

Finally, the modified weighted Welford algorithm (Sec. 4.2.3) provides an online estimation for mean and variance of the error δ_{k+1} (Algorithm 3, step 20).

SORAD has a short transient phase for $k < \ell$. In this period, where the input vector $\mathbf{x}_k$ needs to be partially padded (see Sec. 4.2.1), the model vector $\boldsymbol{\theta}$ is left in its initial state. But the changes $\Delta\boldsymbol{\theta}$ according to Eq. (4.4) are accumulated separately and added to $\boldsymbol{\theta}$ 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, $\boldsymbol{\theta}$ and s_δ are updated normally (Algorithm 3 and 2).

Algorithm 2 Online estimation of sample mean and sample variance for the prediction errors. A more detailed derivation of the update rules can be found in Appendix B.2.

```
Initialize:
$\mu_\delta = 0,\ s_\delta^2 = 0,\ M = 0,\ W = 0$

function updateEstimation($k, \delta_{k+1}$)
    $W \leftarrow \lambda W + 1$
    $\Delta \leftarrow \delta_{k+1} - \mu_\delta$
    $\mu_\delta \leftarrow \mu_\delta + \frac{\Delta}{W}$
    $M \leftarrow \lambda M + \Delta \cdot (\delta_{k+1} - \mu_\delta)$        ▷ Use new value of $\mu_\delta$
    $s_\delta^2 \leftarrow \frac{M}{W}$                                  ▷ Without bias-correction
```

4.3 Experimental Setup

4.3.1 Algorithm Setup

All algorithms (except Offline-RAD) operate online on the time series. We evaluate all algorithms on the Yahoo Webscope S5 benchmark (introduced in Section 2.3).

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).

Setup of Numenta's NuPIC (HTM) To verify the correctness of our NuPIC installation, we applied it to the Numenta Anomaly Benchmark (NAB) [89] using the standard

Algorithm 3 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. 4.2.4). **Output:** Anomaly flags $\mathbf{a}_{flags}$.

```
Initialize and Transient Phase
$\boldsymbol{\theta} \leftarrow (\theta_{Bias} \;\; 2^{-1} \;\; 2^{-2} \;\; \cdots \;\; 2^{-\ell})^T$, with $\theta_{Bias} = 0$
$(\mu_\delta, s_\delta^2) \leftarrow (0, \infty)$
$\mathbf{P} \leftarrow 500\mathbf{I}_{\ell+1}$, with the identity matrix $\mathbf{I}_{\ell+1}$
Anomaly flags $\mathbf{a}_{flags} \leftarrow 0$                          ▷ Binary Vector

Set instance counter $k \leftarrow 0$
while instance $y_{k+1}$ available do
    $\mathbf{x}_{k+1} \leftarrow (1, y_k, y_{k-1}, \ldots, y_{k-\ell+1})^T$
    $\tilde{y}_{k+1} \leftarrow \boldsymbol{\theta}^T \mathbf{x}_{k+1}$                          ▷ Predict next step
    Observe $y_{k+1}$
    $\delta_{k+1} \leftarrow y_{k+1} - \tilde{y}_{k+1}$                    ▷ Compute prediction error
    Calculate the $\epsilon$-quantile $z_\epsilon$ of $\mathcal{N}(0, s_\delta^2)$ (see Fig. 4.1)
    $E \leftarrow [\mu_\delta - z_\epsilon,\ \mu_\delta + z_\epsilon]$
    if $\delta_{k+1} \notin E$ then
        $a_{flags,k+1} \leftarrow 1$                          ▷ Flag $y_{k+1}$ as anomalous
        $k \leftarrow k + \ell - 1$                          ▷ Skip next $\ell$ instances
    else
        Update $\mathbf{P}$ and $\boldsymbol{\theta}$ according to Eq. (4.3) and (4.4)
        Update $\mu_\delta, s_\delta^2$ with updateEstimation$(k, \delta_{k+1})$
    $k \leftarrow k + 1$
end while
```

parameter settings and confirmed that we can exactly reproduce the results[3] published in [89]. 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 [3], 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.

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. The period-length is set to the value 40 in all experiments[4]. Finally, the parameter $\max_{anoms}$ is left at its default ($\max_{anoms} = 2\%$).

4.4 Results

Table 4.1 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 4.2 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 influence accuracy.

Fig. 4.2 shows an example time series from the A4 dataset with the anomalies 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 that allows the user to control the trade-off between FP and FN. Fig. 4.5 shows that the performance measure F_1 is very stable over several threshold decades.[5] The multiobjective plot (where the measures FP and FN are plotted against each other) in Fig. 4.3 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 environments.

[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.

[5]Fig. 4.5 shows the stability of F_1 for SORAD (without forgetting). However, 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.

We test in Fig. 4.3 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. 4.6 the effect of varying the forgetting factor in the range [0.8,1.0]. Additionally, Table 4.2 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 4.3 shows the computation times for all algorithms.

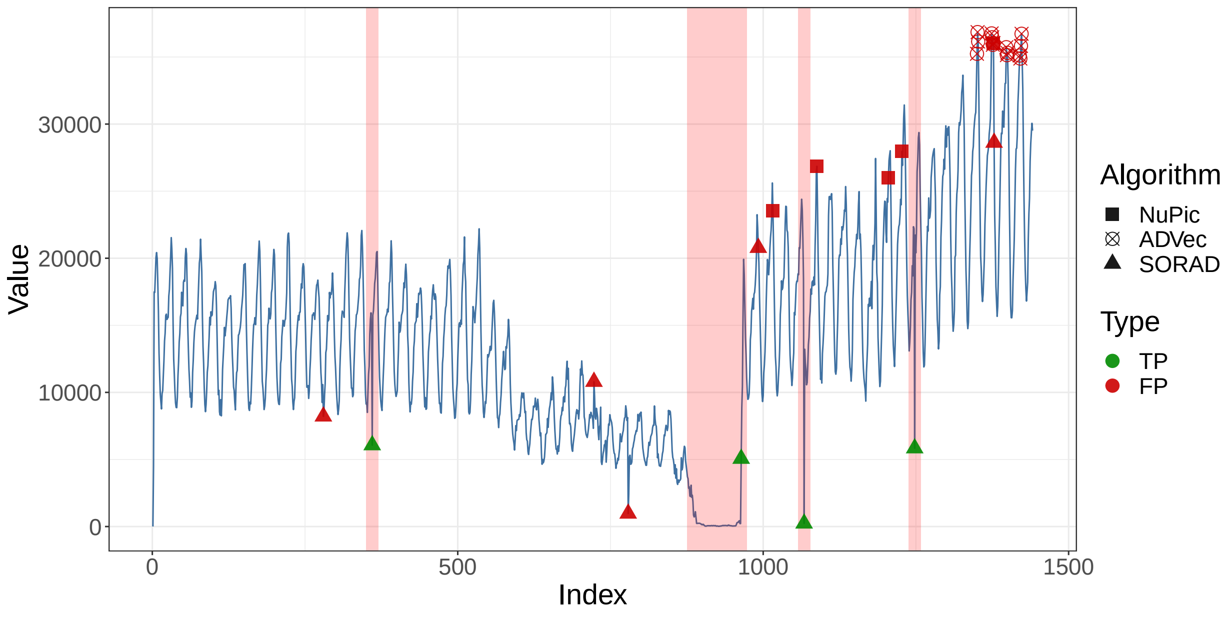

Figure 4.2: Example time series taken from data set A1 with the anomalies detected by the various algorithms SORAD, HTM (NuPIC), and ADVec. The red vertical bars in the plot indicate the true anomaly windows.

4.5 Discussion

4.5.1 Transient Phase

As described in Sec. 4.2.4, algorithm SORAD needs a short transient phase. During this phase, the model vector changes are accumulated, and no anomaly detection is allowed. It

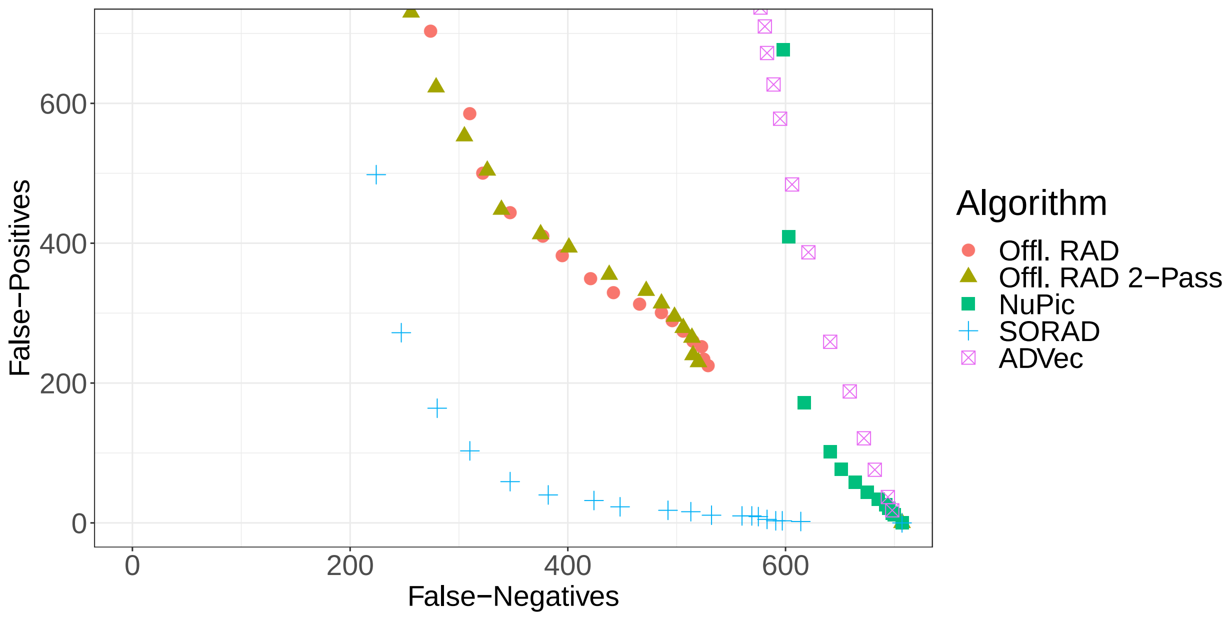

Figure 4.3: Multiobjective plot for different algorithms and thresholds for $t > 500$. Here, the results for the A4 data are shown. The FNs and FPs are the sums over the 100 time series of the A4 data.

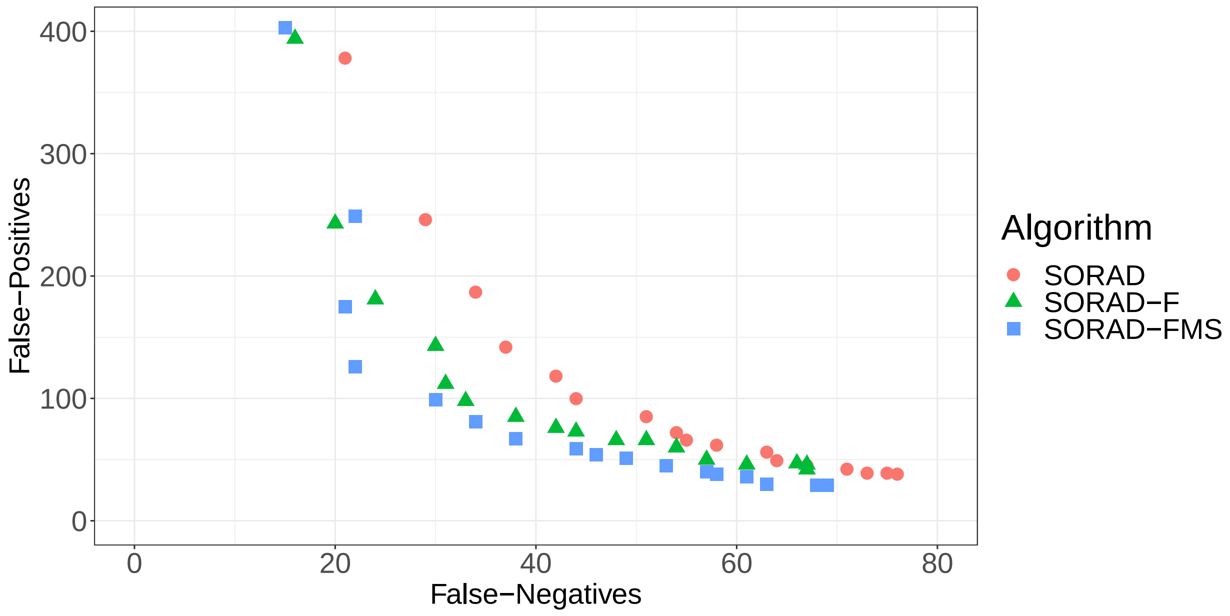

Figure 4.4: Multiobjective plot for different SORAD variants and thresholds for all t after the transient phase. Here, the results for the A1 data are shown. The FNs and FPs are the sums over the 100 time series of the A1 data.

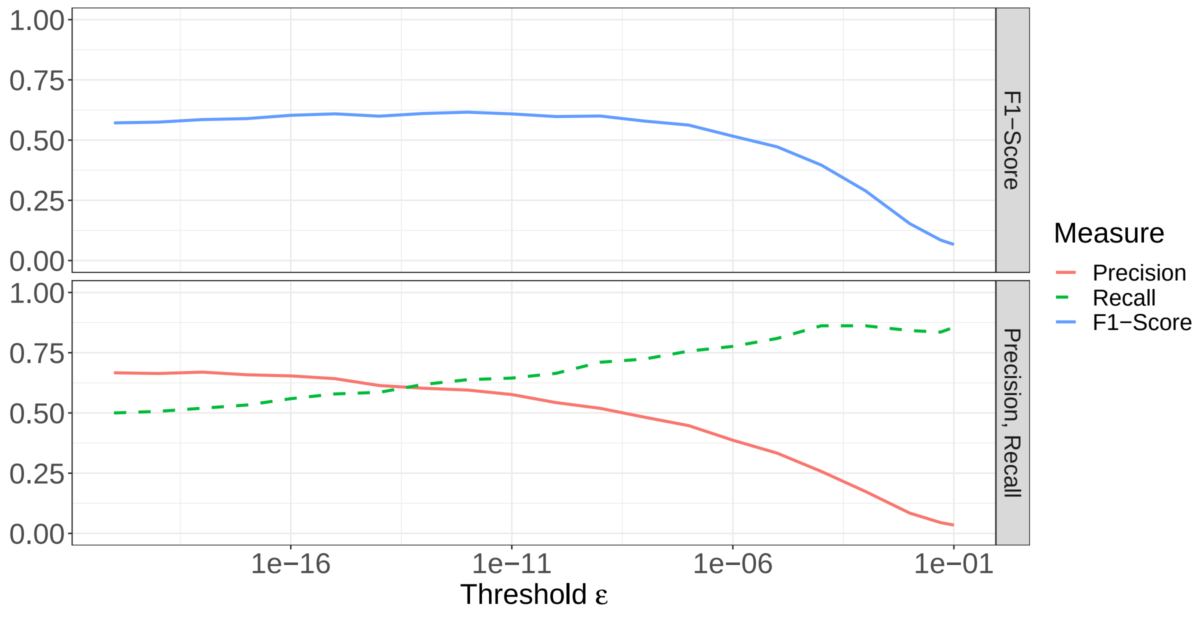

Figure 4.5: The performance of SORAD (without forgetting) over a wide range of thresholds for the A1-data. The F_1 score is virtually constant for a wide range $\epsilon \in [1e-17, 1e-9]$.

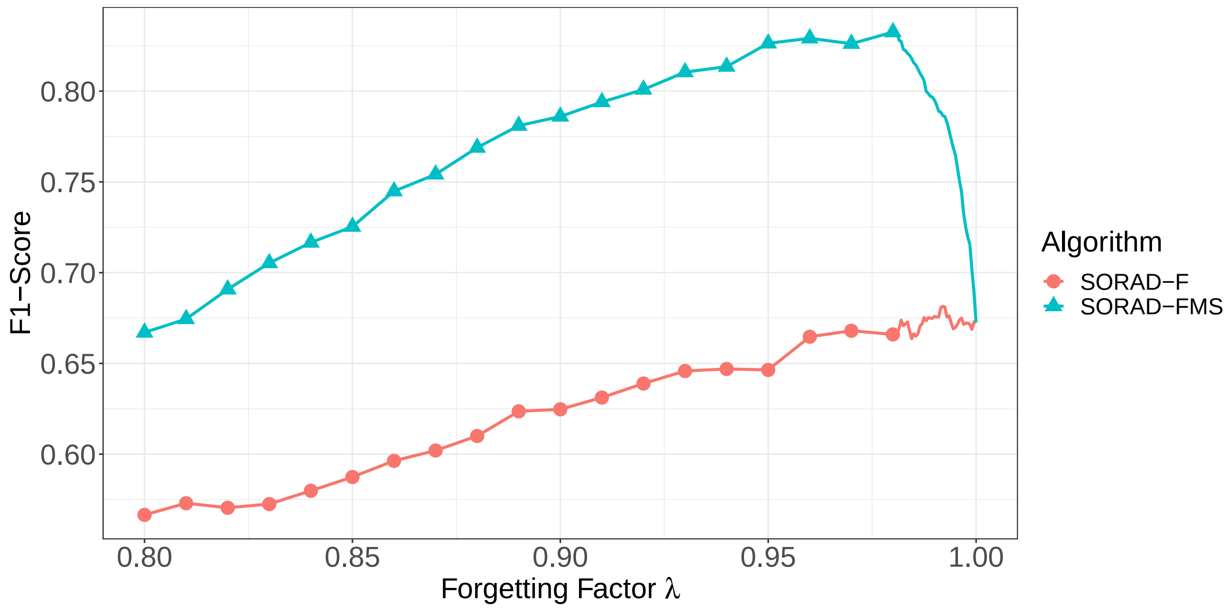

Figure 4.6: Comparing results on A4 for various forgetting factors of SORAD. The first curve SORAD-F shows the results for SORAD with forgetting in RLS; the forgetting factor $\lambda = 0.992$ results in the highest F_1 score of $F_1 = 0.68$ in this case. For the second curve SORAD-FMS, the forgetting is applied to the estimation of μ_δ and s_δ of the error signal distribution as well. The best F_1 score on the A4-data with a value of $F_1 = 0.83$ is reached for a forgetting factor of $\lambda = 0.9805$.

Table 4.1: 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 F_1 for each dataset (over all time series, $t > 500$). All algorithms have their threshold chosen such that FP $\approx$ FN. (Only for the F_1 score in brackets, the threshold is chosen such that F_1 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	$0.42, 0.57$	$0.98, 0.98$	$0.97, 0.94$	$0.45, 0.44$
	F_1 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	$0.41, 0.57$	$0.98, 0.98$	$0.94, 0.96$	$0.44, 0.43$
	F_1 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	$0.66, 0.67$	$1, 0.98$	$0.98, 0.98$	$0.63, 0.65$
	F_1 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	$0.54, 0.53$	$0.47, 0.46$	$0.25, 0.24$	$0.14, 0.15$
	F_1 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	$0.49, 0.47$	$0.64, 0.57$	$0.26, 0.26$	$0.16, 0.16$
	F_1 score	0.48 (0.48)	0.6 (0.64)	0.26 (0.29)	0.16 (0.17)

has proven to be adversarial to start directly with the recursive procedure while the sliding window is not entirely filled with true data. Our experiments have shown that in such a case, the estimation of the vector $\boldsymbol{\theta}$ tends to be unstable, and the anomaly error rate increases.

4.5.2 Forgetting Factor

Online algorithms open the possibility to add a certain degree of forgetting. We investigated different forms of forgetting in SORAD. SORAD-F only uses forgetting in RLS. For SORAD-FMS, the forgetting is applied to the estimation of μ_δ and s_δ of the error signal distribution as well. Our results indicate that the usual forgetting element in the RLS part (SORAD-F) does not play a significant role. However, the new element to add forgetting to the estimation of the error distribution (SORAD-FMS) has proven beneficial in datasets A1 and A4. The increase in F_1 score (13%) is most prominent for dataset A4, which is the dataset with the most significant non-stationary elements. Note that all datasets in the Yahoo S5 benchmark are relatively short (1500 time steps), thus putting a boundary on the

Table 4.2: Same as Table 4.1, 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 4.1, 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	$0.6, 0.62$	$1, 0.98$	$0.97, 0.97$	$0.68, 0.66$
	F_1 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	$0.62, 0.62$	$0.99, 0.98$	$0.95, 0.98$	$0.67, 0.69$
	F_1 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	$0.65, 0.64$	$0.99, 0.98$	$0.97, 0.94$	$0.73, 0.81$
	F_1 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	$0.39, 0.45$	$0.47, 0.46$	$0.18, 0.2$	$0.11, 0.13$
	F_1 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	$0.32, 0.53$	$0.55, 0.57$	$0.27, 0.27$	$0.15, 0.15$
	F_1 score	0.4 (0.4)	0.56 (0.59)	0.27 (0.3)	0.15 (0.16)

Table 4.3: Computation times of the algorithms on datasets A1–A4. Shown are 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

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.

4.5.3 Detection Accuracy

This comparative study's most striking result 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 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.

4.5.4 Other algorithms

Suh et al. [154] 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 / F_1 = 0.54 / 0.51 / 0.52 are inferior to the results of SORAD. In [115], the F_1 scores of an CNN-based algorithm (DeepAnT) on A1–A4 are reported: The obtained F_1-scores for A1, A2, A3 and A4 are 0.46, 0.94, 0.87, and 0.68, respectively, which are all slightly lower than the reported values for SORAD-FMS in Table 4.2.

4.5.5 Limitations of SORAD

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

We do *not* claim that SORAD is for all time series the better anomaly detector. It has not enough memory for more complicated long-term interactions. On the Numenta anomaly detection benchmark NAB [89], where NuPIC achieves good results (F_1=0.51), SORAD does not perform too well in its current form (F1=0.24). Further research effort is needed here.

4.6 Conclusion

In concluding this chapter, we refer to our research questions from 4.1 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 detection 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 using its 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 superior to the offline variant (around 40% increase in F_1 score on datasets A1 and A4). We showed that it is crucial 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.

4.6.1 Possible 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). Simple extensions of the algorithm, such as multi-step ahead prediction (predict several horizons instead of just one), additional non-linear feature transformations, replacing the RLS algorithm with an kernel-based RLS approach [41], and the usage of other online approaches for modeling the prediction errors (instead of a Gaussian distribution) could likely improve the performance of SORAD.

In the following chapter, we will introduce the DWT-MLEAD algorithm, which – contrary to SORAD – can analyze time series at different time scales and thus, performs better on the time series (with longer-range anomalies) of the Numenta Anomaly Benchmark.