Time Series Anomaly Detection with Discrete Wavelet Transforms and Maximum Likelihood Estimation

Markus Thill[1], Wolfgang Konen[1], and Thomas Bäck[2]

[1] Department of Computer Science, TH Köln – University of Applied Sciences, 51643 Gummersbach, Germany
{markus.thill,wolfgang.konen}@th-koeln.de
[2] Department of Computer Science, Leiden University, LIACS, 2333 CA Leiden, The Netherlands
t.h.w.baeck@liacs.leidenuniv.nl

Abstract. A new algorithm, based on the Discrete Wavelet Transform (DWT), for unsupervised anomaly detection in time series is introduced in this paper. The approach is based on using maximum likelihood estimation (MLE) on the DWT of time series. On a diverse set of 158 time series, the algorithm is compared with three other state-of-the-art anomaly detectors and it is shown to outperform the other approaches on the test set. Thanks to the linear time complexity of the DWT, our new algorithm is also computationally efficient.

Keywords: Time series, anomaly detection, wavelet transform, DWT, maximum likelihood estimation

1 Introduction

Anomaly detection in time series is a key technology in many areas. Industries have more and more devices (predictive maintenance for industry equipment, sensors in the internet of things, or server technologies in cloud services of the internet) which are collecting increasingly large streams of data. Research institutions (e.g. high energy physics or astronomy) are collecting vast amounts 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 closed form. A further complication is that anomalies can appear on quite different time scales: they can be spikes (short-time events) or broader structures (long-term irregularities). Most anomaly detection algorithms available today have their strength either in shorter or in longer time scales, but not in both.

Wavelets are a well-established technique in signal processing which allow to extract features in a self-similar fashion over a broad range of time scales

(frequencies). This makes them ideally suited to detect anomalies on different time scales where the time scale is a priori unknown.[3]

We present in this paper a new method for detecting anomalies based on wavelet processing and maximum likelihood estimation (MLE). Section 2 explains our method, Sec. 3 describes the experimental setup. Sec. 4 shows results and discusses them. Sec. 5 concludes.

1.1 Related Work

Despite the fact that wavelets are used for decades in signal processing and feature extraction, e. g. for classification of whole time series (machinery data) [12,13], there is only very little work with wavelets being used for anomaly detection, i. e. finding precise time intervals in time series containing anomalies: Kwon et al. [5] use wavelet transforms for the detection of network anomalies in the case of a possible attack by a malicious user. Kanarchos et al. [4] use wavelets in conjunction with neural networks and Hilbert transforms. Their algorithm was only tested on two time series which consisted of synthetic normal data and a synthetic anomaly.

In this work we test our algorithm on two large anomaly benchmarks, one being the well known Numenta Anomaly Benchmark (NAB, 58 time series, most of them real-world) [7] and the other being a subset of the Yahoo's S5 Webscope benchmark [6] (A3, 100 synthetic time series). We compare our algorithm with other state-of-the-art anomaly detectors: Numenta's NuPic, based on Hierarchical Temporal Memory (HTM) [2], our previous algorithm SORAD [10] which is specialized for short-time anomalies, and Twitter's ADVec algorithm [11].

2 DWT-MLEAD Algorithm

In this section we introduce our new unsupervised DWT-MLEAD algorithm which uses **D**iscrete **W**avelet **T**ransform and **M**aximimum **L**ikelihood **E**stimation for **A**nomaly **D**etection in time series.

2.1 Wavelet Transforms

Wavelet transforms [8] allow to represent a time series signal in terms of waves (the so called wavelets) with little local support. While (short-time) Fourier transforms always have a trade-off between accuracy in the frequency domain and accuracy in the time domain, wavelet transforms are used to retrieve accurate time-localized frequency information. The wavelet transform of a time series signal is composed with scaling and shifting functions. They take a mother wavelet and stretch and shrink it (scaling), dilate it along the time axis (shifting), and finally form the scalar product with the time series. Sampling wavelets

[3] We note in passing that the visual or auditory system of higher vertebrates contains information-processing structures similar to wavelets [3], thus underpinning the importance of wavelets for *natural computing*.

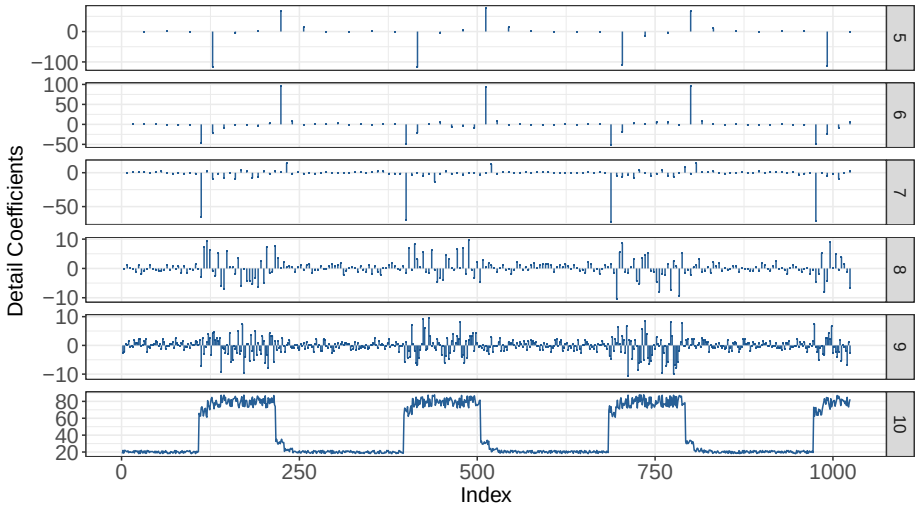

Fig. 1. Example of a decimating DWT using Haar Wavelets for a time series of the NAB data. The original time series is depicted on scale 10. On the scales 5–9 the detail coefficients of the DWT are shown. While we move towards lower scales, the number of coefficients is halved in each step, with 32 coefficients left on scale 5.

in a discrete manner leads to the so called discrete wavelet transform (DWT), which is commonly used in practice and has linear computational complexity. In its current form, DWT-MLEAD performs a decimating DWT using Haar wavelets on each time series. For this purpose, the R-package *wavetresh* [9] is used. Since the package requires the time series to have a length equal to a power of two, we currently artificially extend – where required – a time series of length n to a length $m = 2^{\lceil \log_2(n) \rceil}$, by mirror copying the last segment of the original time series into the extended area. However, we do not consider anomalies which are detected at instances $> n$. DWT-MLEAD utilizes both the detail coefficients $d_{k,\ell}$ and the approximation coefficients $c_{k,\ell}$, computed by the DWT (lines 6–7 in Algorithm 1), where ℓ addresses the level and $k \in 1, \ldots, m$ the time index. The lowest level $\ell = 0$ contains only one coefficient. The highest level $L = \log_2(m)$ has no approximation coefficients but only detail coefficients $d_{k,L}$ which represent the original time series. In Fig. 1 the DWT of a time series from NAB is illustrated.

2.2 Sliding Windows

In order to express temporal relationships, a simple and common approach in many machine learning tasks involving time series is to employ sliding windows of a certain size w (e.g. $w = 10$), which are used to generate fixed-sized input vectors for a model. By stacking the transposed input vectors, we obtain a matrix $\mathbf{X}$ with w columns which can be used to train a model. In the DWT-MLEAD

algorithm (Algorithm 1, lines 9–10), a window of size w_ℓ is slid over the detail and approximation coefficients $d_{k,\ell}$ and $c_{k,\ell}$ at each DWT level $\ell \in \{\ell', \ldots, L\}$ in order to generate the matrices $\mathbf{D}^{(\ell)}$ and $\mathbf{C}^{(\ell)}$. Subsequently, for each matrix a multivariate Gaussian distribution is estimated, as described in the following.

2.3 Gaussian Distributions & Maximum Likelihood Estimation

In order to learn the usual patterns in the time series, DWT-MLEAD estimates multivariate Gaussian distributions for the data generated by the sliding window approach. A Gaussian distribution is fully parametrized by a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$. Assuming that an observed data sample was drawn from a specified distribution (a Gaussian), the maximum likelihood estimation (MLE) finds the parameters of this distribution such that these parameters maximize the likelihood of observing the given sample. The function MLE in Algorithm 2 does just this for a given matrix $\mathbf{X}$, where $\mathbf{X} \in \mathbb{R}^{n \times w}$, with $n = m - w + 1$ being the number of input vectors generated by sliding the window over the time series, $\boldsymbol{\mu} \in \mathbb{R}^w$ is a w-dimensional vector, which indicates the center of the distribution, and $\boldsymbol{\Sigma} \in \mathbb{R}^{w \times w}$ describes the covariances between individual dimensions. In Algorithm 1, line 12, DWT-MLEAD estimates the distribution parameters for each $\mathbf{D}^{(\ell)}$ and $\mathbf{C}^{(\ell)}$.

Subsequently, for every entry in $\mathbf{D}^{(\ell)}$ and $\mathbf{C}^{(\ell)}$ we compute the log-likelihood vector $\boldsymbol{p}$ using the previously determined parameters of the Gaussian distribution. This is done in function logProbDensity of Algorithm 2.

2.4 Quantile Boundaries

In order to separate unusual from usual window patterns in $\mathbf{D}^{(\ell)}$ and $\mathbf{C}^{(\ell)}$, one has to find a suitable boundary. The first method we use computes an empirical ϵ-quantile z_ϵ (e.g. the first percentile) for the log-likelihood vector $\boldsymbol{p}$. Another approach we use to estimate the threshold z_ϵ involves a Monte Carlo method, which samples from the estimated Gaussian distribution and determines the $(1-\epsilon)$-quantile for the Mahalanobis distances of the sample to the center of the distribution. After computing z_ϵ in Algorithm 1, line 14, instances are flagged as "unusual" in a binary vector $\boldsymbol{a}$ if their log-likelihood p_i lies beneath z_ϵ (line 15).

2.5 Leaf Counters

For each instance in the original time series the DWT-MLEAD algorithm maintains a leaf counter h_i. If an instance $c_{k,\ell}$ or $d_{k,\ell}$ on a certain level ℓ of the DWT is flagged as unusual (has a flag $a_k = 1$) then an event e – marked as a black node in Fig. 2 – is passed down the DWT tree to all leaf nodes connected with the e node. Each leaf node has a counter h_i (blue rectangles in Fig. 2) which counts all such events (Algorithm 1, line 16). After all events are processed, all counters with a count $h_i < 2$ are deleted (line 17).

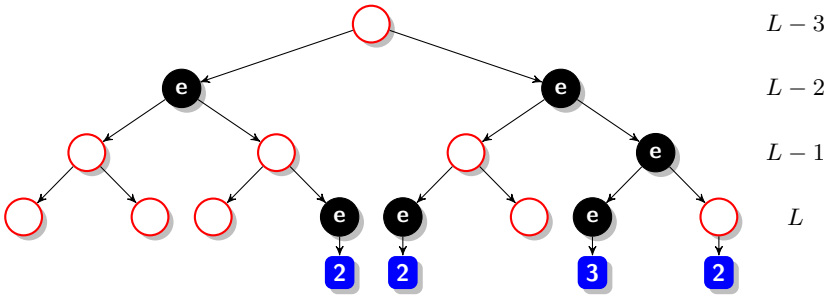

Fig. 2. Detecting anomalies with leaf counters. Along the vertical axis are the DWT levels ℓ, along the horizontal axis are the time indices k. The leftmost event e thus comes from either an unusual $c_{1,L-2}$ or $d_{1,L-2}$. Each event increases the leaf counters (blue rectangles) connected with the e node. Only counters with count ≥ 2 are shown.

2.6 Detecting the Anomalies

Once all the leaf counters are updated, DWT-MLEAD forms clusters C_j of all leaf counters h_i having a neighbor not more than d_{max} apart (Algorithm 1, line 18). Specifically, a cluster C_j is here a set of counters, each counter carrying its leaf position in the original time series and its event count. For each cluster C_j a sum s_j over all event counts is computed. In Fig. 2 for example, all counters form *one* cluster with sum $s_j = 9$. If a sum s_j exceeds the predefined threshold B, then the center of cluster C_j is labeled as anomaly event (line 23). The center $\mu(C_j)$ of cluster C_j is the weighted center of mass of all leaf positions, where the weights are the event counts.

3 Experimental Setup

3.1 The Benchmarks

The Numenta Anomaly Benchmark (NAB) [7] is a publicly available dataset that consists of 58 time series with in total 365,558 data points – the shortest series containing 1,127, the longest containing 22,695 and the average series containing approx. 6,300 instances. The majority of the time series are real-world data coming from application areas such as server monitoring, network utilization, sensor readings from industry and social media statistics [7]; 11 time series were generated artificially, from which 5 are anomaly-free. In total, over all 58 time series, 115 anomalies were labelled, most of which were identified manually. It has to be emphasized that the 58 time series are very diverse. The second benchmark we will investigate is the A3 data from the Webscope S5 benchmark [6]. It consists of 100 synthetic time series, each of length 1500, with in total 850 short-term anomalies. In our setup, the ground truth anomaly labels are not provided to the anomaly detection algorithms, which have to learn to separate anomalies from normal behavior in an unsupervised fashion.

Algorithm 1 DWT-MLEAD, an anomaly detection algorithm using the Discrete Wavelet Transform.

Define: ℓ' as starting level in DWT for analyzing the time series
    ϵ for computation of quantiles (e.g., the 1st percentile)
    d_{max}: maximum distance for same-cluster points
    B: threshold for the counter sum in a cluster that triggers an anomaly
function MLEANOMALY($y = (y_1, y_2, \ldots, y_m)$) ▷ m is a power of 2
    Compute DWT of y for levels $\ell \in \{\ell', \ldots, L\}$, with $L = \log_2(m)$
    Get detail coefficients $d_{k,\ell}$ and approximation coefficients $c_{k,\ell}$ of DWT
    Initialize a leaf counter $h_i = 0$ for each y_i, counting the events it receives
    Set window sizes for each level: $w_\ell = \max\{2, \ell - \ell' + 1\}$
    $\forall \ell \in \{\ell', \ldots, L\}$: Build $\mathbf{D}^{(\ell)}$, $\mathbf{C}^{(\ell)}$ by sliding window of size w_ℓ over $d_{k,\ell}$, $c_{k,\ell}$
    for all $\mathbf{X} \in \{\mathbf{D}^{(\ell)}, \mathbf{C}^{(\ell)} \mid \ell = \ell', \ldots, L-1\} \cup \mathbf{D}^{(L)}$ **do**
        $(\mu, \Sigma) = \text{MLE}(\mathbf{X})$ ▷ Defined in Algorithm 2
        $\boldsymbol{p} = \text{LOGPROBDENSITY}(\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ ▷ Defined in Algorithm 2
        Compute ϵ-quantile z_ϵ
        $\boldsymbol{a} = \text{PREDICT}(\boldsymbol{p}, z_\epsilon)$ ▷ Defined in Algorithm 2
        For all $\boldsymbol{a}_i = 1$: Trigger an event moving down the tree to any connected leaf
    When all events are processed: Delete all event counters with count $h_i < 2$
    Form clusters C_j of leaf counters having a neighbor not more than d_{max} apart
    $\mathbf{S} = \{\}$ ▷ Set of detected anomalies
    for all C_j **do**
        s_j = sum of counter values in C_j
        if $s_j > B$ **then**
            $\mathbf{S} = \mathbf{S} \cup \{\mu(C_j)\}$ ▷ Add center $\mu(C_j)$ of C_j to anomaly set
    return S

Algorithm 2 Helper functions for Algorithm 1.

function MLE($\mathbf{X}$)
    $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$ ▷ Vetor $\boldsymbol{x}_i \in \mathbb{R}^w$ is the ith row of matrix $\mathbf{X} \in \mathbb{R}^{n \times w}$
    $\boldsymbol{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T$
    return $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

function LOGPROBDENSITY($\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$)
    $\boldsymbol{p} \in \mathbb{R}^n$ ▷ n is the number of rows in $\mathbf{X}$
    for each row $\boldsymbol{x}_i$ of $\mathbf{X}$ **do**
        $\boldsymbol{p}_i = -\frac{1}{2}\log\det(2\pi\boldsymbol{\Sigma}) - \frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu})$
    return $\boldsymbol{p}$

function PREDICT($\boldsymbol{p}$, z_ϵ)
    $\boldsymbol{a}$: vector of same size as $\boldsymbol{p}$
    for all $\boldsymbol{a}_i$ **do**
        $a_i = \begin{cases} 1, & \textbf{if } p_i < z_\epsilon \\ 0, & \textbf{otherwise} \end{cases}$ ▷ Binary anomaly flag vector
    return $\boldsymbol{a}$

3.2 Algorithms and their Settings

In the following, we compare DWT-MLEAD with three online anomaly detection algorithms, namely SORAD, NuPic, and ADVec. Although we did not systematically tune the parameters of each algorithm, we empirically determined for each algorithm and each dataset the best parameters from an informal search.

DWT-MLEAD Overall, three main parameters in Algorithm 1 have to be set, which are fixed for the whole dataset: a threshold $\epsilon \in [0, 1]$ for the ϵ-quantiles, which is varied to adjust the tradeoff between precision and recall, a parameter B (threshold for counter sum), and a starting level ℓ'. From Sec. 2.4, we use the empirical quantiles for the NAB data and the Monte Carlo based quantiles for the A3 data. We empirically determined the setting $B = 3.5$, $\ell' = 5$ for the NAB data and $B = 1$, $\ell' = 7$ for the A3 data. The window size w_ℓ is set by Algorithm 1 in a level-dependent fashion. In its current form the DWT-MLEAD algorithm operates offline on each time series, the remaining algorithms investigated in this work are all online.

SORAD In this work we will also report results for a simple online regression anomaly detection (SORAD) algorithm which we recently developed [10]. The algorithm has several parameters which are set as follows for the experiments: We set the forgetting factor of the algorithm to $\lambda = 0.98$, the anomaly threshold ϵ will be varied over a larger range, and the window-size is set to $w = 10$ for the A3 data and to $w = 200$ for the NAB data.

NuPic Numenta's NuPic[4] [2] requires a large number of parameters which cannot be set easily. Although NuPic provides a swarming algorithm [1] that optimizes the parameters, we found that the results for the swarmed parameter search are not significantly different from those for a standard parameter setting, which was also used for the reported results in [7]. Hence, we use the standard parameter setting for all experiments. The only parameter which is adjusted by us is an anomaly threshold that can be varied in the interval [0,1] and – similar to ϵ in SORAD and DWT-MLEAD – trades off precision and recall.

ADVec Twitter's ADVec Algorithm [11], which is available as open-source R package AnomalyDetection from Github[5] is the last algorithm which we will review in this work. The algorithm requires three main parameters, which are as follows: The first parameter α describes the level of statistical significance with which to accept or reject anomalies. As in the other algorithms, this parameter can be interpreted as an anomaly threshold. During our experiments, this parameter will be varied over a large range of values. ADVec requires a second parameter, a period-length, which we fix to the value 40 – which has shown to give

[4] https://github.com/numenta/nupic
[5] http://github.com/twitter/AnomalyDetection

the best results on the investigated data – entirely throughout this work. Finally, we found that the setting of the parameter $\max_{anoms}$ is crucial for the performance of ADVec, especially on the NAB dataset. This parameter determines the maximum number of anomalies that the algorithm will detect as a percentage of the data. We choose $\max_{anoms} = 1\%$ for the A3 data and $\max_{anoms} = 0.1\%$ for the NAB data.

3.3 Algorithmic Performance Measures

Similar to typical classification tasks, for time series anomaly detection problems an algorithm has to classify each time series sample as either anomalous (unusual) or as normal (usual). Commonly, correctly identified anomalies and normal instances are considered as true-positives (TP) and true negatives (TN), respectively. Misclassifications are accordingly referred to as false-positives (FP) and false-negatives (FN). In these cases normal/usual instances are falsely flagged as anomalous (FP) or the algorithm fails to detect real anomalies (FN). Due to the typically large number of TN for anomaly detection tasks, we renounce reporting this score. Based on the three remaining measures additional metrics can be derived, which are useful for comparing the performance of algorithms and will be used in later sections.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1}$$

Ideally, one attempts to maximize precision and recall (with a max. value of one). However, since precision and recall are conflicting objectives in practice, the F_1 score – which takes both precision and recall into account – can be used to assess an algorithm's performance. The F_1 score is defined as:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2}$$

Since temporal anomalies can span over larger intervals, we use so-called anomaly windows for the scoring process. For the NAB data the already specified anomaly windows are used and for Yahoo's Webscope S5 data we place windows of size 10 around the labeled anomalies. While each detection outside of an anomaly window will be counted as a FP, multiple detections inside a window are only counted as *one* TP. Conversely, no detection within an anomaly window will be counted as *one* FN as well.

4 Results & Discussion

Table 1 summarizes the results for the four algorithms on the A3 and NAB data. On the A3 data with short-term anomalies, DWT-MLEAD and SORAD both clearly outperform the other algorithms NuPic and ADVec, achieving both, a high precision and recall. NuPic and ADVec produce a large amount of FP and at the same time miss most of the true short-term anomalies. For the NAB data

Table 1. Results for various algorithms on the A3 and NAB dataset. Shown are the sums of TP, FP, FN over all time series and the metrics precision, recall and F_1, cf. Eqs. (1)–(2), derived from these sums. All algorithms have their threshold chosen such that F_1 is maximized (in brackets: F_1 for threshold such that FP $\approx$ FN).

Dataset	Algorithm	Threshold	TP	FP	FN	Precision	Recall	F_1 Score
A3	DWT-MLEAD	0.015	806	8	44	**0.99**	**0.95**	**0.97** (0.95)
	NuPic	0.4	172	267	678	0.39	0.2	0.27 (0.26)
	SORAD	10^{-4}	810	22	40	0.97	**0.95**	0.96 (**0.96**)
	ADVec	20	190	216	660	0.47	0.22	0.3 (0.26)
NAB	DWT-MLEAD	0.02	69	65	46	**0.51**	0.6	**0.55** (**0.55**)
	NuPic	0.55	76	113	39	0.4	**0.66**	0.5 (0.47)
	SORAD	10^{-9}	57	313	58	0.15	0.5	0.24 (0.21)
	ADVec	100	66	164	49	0.29	0.57	0.38 (0.34)

Table 2. Computation times of the algorithms on datasets A3 and NAB. Shown is the average and standard deviation from 20 runs each. The runs were performed on a PC with an i7-3520M CPU and 8 GB of RAM.

	Computation Time (s)			
Dataset	DWT-MLEAD	SORAD	NuPic	ADVec
A3	13.6 ± 0.3	34.6 ± 0.1	810.9 ± 1.3	2.6 ± 0.2
NAB	12.2 ± 0.2	111.6 ± 0.2	1636.4 ± 2.7	5.8 ± 0.5

we observe rather different results: while DWT-MLEAD still outperforms the remaining algorithms according to the overall F_1 score, SORAD now performs the worst according to all metrics. In particular, the precision is rather low for SORAD, due to the large number of FP. NuPic delivers similar results as DWT-MLEAD, with a slight advantage for DWT-MLEAD.

Two example time series from the NAB data with the detections of the individual algorithms are shown in Fig. 3. In the first example it can be clearly seen that SORAD produces many FP at the recurring spikes in the time series. This is due to the fact that SORAD has no long-term memory so that such recurring spikes appear to be anomalous. Only DWT-MLEAD and ADVec detect both anomalies in both examples, although ADVec produces a few more false-positives.

All algorithms examined in this work have a threshold which can be varied in a certain range and which trades off FP and FN (as well as precision and recall) to a certain extent. In Fig. 4 the precision is plotted against the recall for different thresholds. For the A3 data the recorded points of DWT-MLEAD and SORAD clearly dominate those of NuPic and ADVec. For the NAB data the results are more diverse: while SORAD shows the worst performance of all algorithms, DWT-MLEAD and NuPic show the best performance, with NuPic

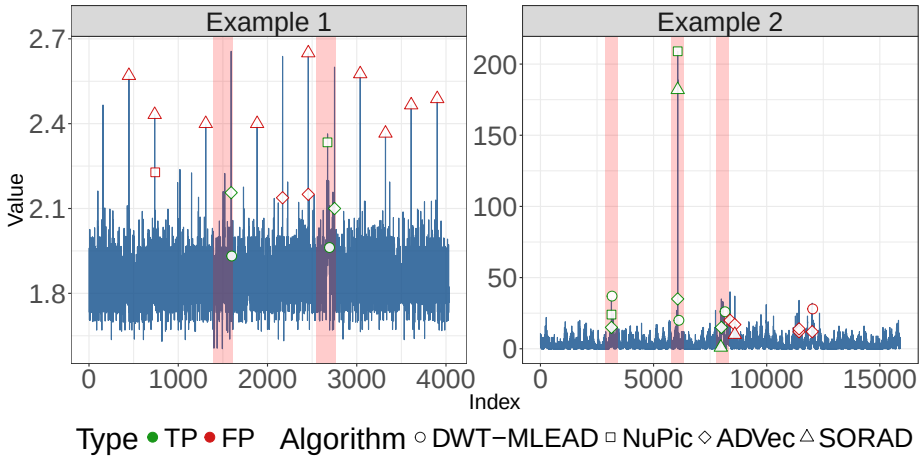

Fig. 3. Example time series taken from the NAB data with the anomalies detected by the algorithms DWT-MLEAD, NuPic, ADVec, and SORAD. The red vertical bars in the plot indicate the true anomaly windows. True-positives are indicated by green colors while False-positives are colored red.

having a slightly higher precision in larger recall ranges (recall > 0.6) and DWT-MLEAD in the lower recall ranges (recall < 0.6).

In Table 2, the computation times for the four algorithms on the A3 and NAB data are shown (mean and standard deviation from 20 runs). Overall, ADVec shows the best results regarding the computation time. On the A3 (NAB) data DWT-MLEAD is faster by a factor of 2.5 (9) than SORAD and 60 (134) than NuPic. However, we assume that an online implementation of DWT-MLEAD might require some additional computation time.

4.1 Discussion

The wavelet transform allows to capture features of the time series on different frequency levels. This is beneficial for detecting both long- and short-term anomalies. It is thus not unexpected that DWT-MLEAD is the only algorithm in our comparison which performs equally well on both benchmarks A3 and NAB. The event pooling mechanism shown in Fig. 2 with a minimum event count of 2 in each leaf counter is effective in shielding against noise which may produce an unusual event in just one frequency level. As expected, SORAD operates only well on short-term anomalies, since it analyzes only a short-term window in the original time series, which is too short for anomalies with a longer range.

The algorithm DWT-MLEAD in its current form has some limitations:

– We did so far only explore Haar wavelets and only modeled a multivariate Gaussian distribution to the data. It may be that other wavelets or other distributions would lead to better results.

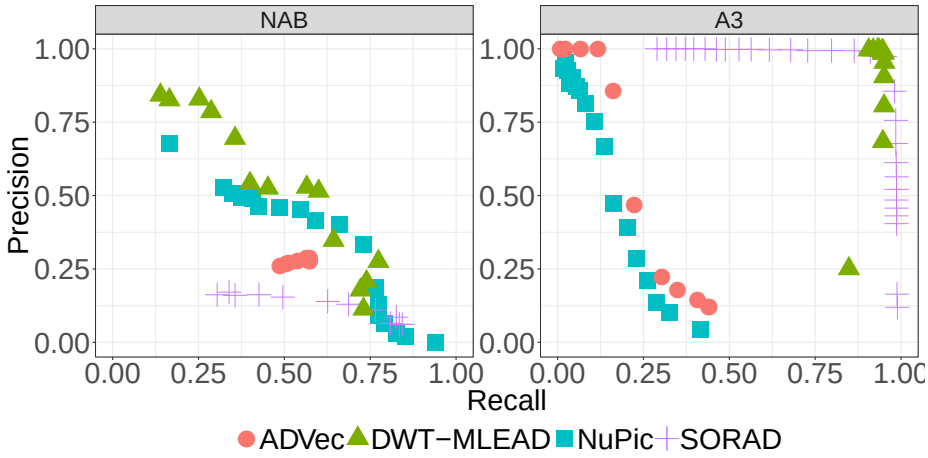

Fig. 4. Multiobjective plot for the NAB and A3 dataset. Precision and recall are computed based on the results of *all* time series of the corresponding data set.

- It is offline, i. e. the anomaly detection is undertaken when the whole time series is available. (It is still unsupervised since no information about prior anomalies is given to the algorithm.) There is however no obstacle to turn it into an semi-online algorithm on longer time series, where the whole algorithm would be repeated after short time intervals (e. g. 100 or 200) on the last 2^m (e. g. 1024 or 2048) time steps of the time series.
- We assume a certain degree of stationarity for the algorithm to work. Trends and change-points cannot be handled well in the offline form. Again, a semi-online version could offer more flexibility in the sense that trends and change-points can be learned by looking at the history of all 2^m-windows.
- If a time series has long-term periodic structures, not all anomalies might be detected correctly. This can happen if the frequency of the long-term periodic structure is lower than the lowest wavelet level ℓ' considered in Algorithm 1. In such cases it might help to extend the algorithm by a periodicity detector and subtract such a periodicity prior to analysing the time series with DWT-MLEAD.

5 Conclusion & Future Work

We have shown that the discrete wavelet transform (DWT) is beneficial for detecting anomalies in time series on various time scales. Specifically, our new algorithm DWT-MLEAD shows consistently good results on two larger benchmarks, one containing short-term anomalies (A3) and the other containing long-term anomalies (NAB). We tested this algorithm against three other state-of-the-art anomaly detectors and found DWT in first place on both benchmarks. It is remarkable that a single algorithmic principle works well over such a diverse

set of time series. Due to the efficient implementation available for DWT, our algorithm is computationally efficient (fast) as well.

5.1 Future Work

DWT-MLEAD works better than the other algorithms tested in this study, but it is not perfect yet, especially not on the NAB benchmark. Future work in this area should focus on improving the first layout of this algorithm, as outlined in Sec. 4.1 (Discussion): other wavelets than Haar wavelets, other than multivariate Gaussian distributions, a semi-online version of the algorithm, automated algorithm parameter tuning, and a periodicity detector.

References

1. Ahmad, S.: Running swarms (May 2017), `http://nupic.docs.numenta.org/0.6.0/guide-swarming.html`
2. George, D., Hawkins, J.: Towards a mathematical theory of cortical micro-circuits. PLoS Comput Biol 5(10), e1000532 (2009)
3. Jones, J., Palmer, L.: An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. J. Neurophysiol. 2, 1233–1258 (1987)
4. Kanarachos, S., Mathew, J., Chroneos, A., Fitzpatrick, M.: Anomaly detection in time series data using a combination of wavelets, neural networks and Hilbert transform. In: International Conference on Information, Intelligence, Systems and Applications (IISA). pp. 1–6 (2015)
5. Kwon, D., Ko, K., Vannucci, M., Reddy, A.N., Kim, S.: Wavelet methods for the detection of anomalies and their application to network traffic analysis. Quality and Reliability Engineering International 22(8), 953–969 (2006)
6. Laptev, N., Amizadeh, S.: Yahoo anomaly detection dataset S5 (2015), `http://webscope.sandbox.yahoo.com/catalog.php?datatype=s\&did=70`
7. Lavin, A., S. Ahmad, S.: Evaluating real-time anomaly detection algorithms – the Numenta anomaly benchmark. In: IEEE Conference on Machine Learning and Applications (ICMLA2015) (2015), `http://arxiv.org/pdf/1510.03336`
8. Meyer, Y., Salinger, D.: Wavelets and Operators:, Cambridge Studies in Advanced Mathematics, vol. 1. Cambridge University Press (1995)
9. Nason, G.: wavethresh: Wavelets Statistics and Transforms (2016), `https://CRAN.R-project.org/package=wavethresh`, R package version 4.6.8
10. Thill, M., Konen, W., Bäck, T.: Online anomaly detection on the Webscope S5 dataset: A comparative study. In: IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS 2017). p. 1. Springer (2017)
11. Vallis, O., Hochenbaum, J., Kejariwal, A.: A novel technique for long-term anomaly detection in the cloud. In: 6th USENIX Workshop on Hot Topics in Cloud Computing, Philadelphia, PA (2014)
12. Yen, G.Y., Lin, K.C.: Wavelet packet feature extraction for vibration monitoring. In: IEEE International Conference on Control Applications. vol. 2, pp. 1573–1578 (1999)
13. Zhengjia, H., Jiyuan, Z., Yibin, H., Qingfeng, M.: Wavelet transform and multiresolution signal decomposition for machinery monitoring and diagnosis. In: IEEE International Conference on Industrial Technology (ICIT96). pp. 724–727 (1996)