# Machine learning and deep learning approaches for multivariate time series prediction and anomaly detection

Thill, M.

# Chapter 5

# An Anomaly Detection Algorithm based on Discrete Wavelet Transforms

## 5.1 Introduction

In the previous chapter, we introduced the SORAD algorithm which demonstrated its effectiveness on the Yahoo Webscope S5 benchmark, outperforming other state-of-the-art algorithms. However, the benchmark time series mostly contained only short-term anomalies. Initial experiments on the Numenta Anomaly Benchmark (NAB) revealed that SORAD has difficulties with time series with longer-term patterns and anomalies. A common problem in practice is that anomalies can appear on quite different time scales: they can be spikes (short-time events) or broader structures (mid- or long-term irregularities). Many anomaly detection algorithms available today have their strength either in shorter or in longer time scales, but not in both. However, it is from great practical relevance to have algorithms which work robustly on diverse time-series data. Thus, the underlying research question for this chapter is: Is it possible to propose an online-adaptable anomaly detection algorithm which works robustly on a very *diverse* set of benchmarks?

We believe that wavelet-based methods could be a suitable approach to answer this question. Wavelet transforms [111] are commonly used to decompose a time series signal into accurate time-localized frequency information. This makes them ideally suited to detect anomalies on different time scales where the time scale is a priori unknown.[1]

In the context of the above research question, a new algorithm for unsupervised anomaly detection in time series, called DWT-MLEAD, is introduced in this Chapter. The approach is based on the discrete wavelet transform (DWT) of time series and the identification of abnormal behavior on individual frequency scales using a Mahalanobis-distance-based method. Wavelets allow feature extraction on different frequency levels in different representations and allow a timely precise localization of anomalies in time series. Initially, we

[1]We note in passing that the visual or auditory system of higher vertebrates contains information-processing structures similar to wavelets [72], thus underpinning the importance of wavelets for *natural computing*.

experimented with an offline version of our algorithm and found that it performs well on a set of (mostly) stationary time series. On a diverse set of 158 time series (taken from NAB and the Yahoo Webscope S5 benchmark), the algorithm is compared with three other state-of-the-art anomaly detectors and it is shown to outperform the other approaches. Thanks to the linear time complexity of the DWT, the algorithm is also computationally efficient.

Later in this chapter, we extend DWT-MLEAD to operate fully online. Given streaming data or time series, the algorithm iteratively computes the (causal and decimating) discrete wavelet transform. For individual frequency scales of the current DWT, the algorithm now detects unusual patterns across frequency scales by computing the Mahalanobis distance in an online fashion. The online DWT-MLEAD algorithm is tested on all 425 time series from the Yahoo Webscope S5 benchmark and NAB. A comparison to the other state-of-the-art online anomaly detectors shows that our algorithm can mostly produce results similar to the best algorithm on each dataset. It produces the highest average F1-score with one standard parameter setting. That is, it works more stable on high- and low-frequency-anomalies than all other algorithms. We believe that the wavelet transform is an important ingredient to achieve this.

This chapter is based on our three earlier publications [162, 164, 165].

### 5.1.1 Related Work

Although wavelet transforms are widely used in many fields of signal and image processing and in many data mining tasks such as time series classification, clustering, prediction & forecasting and similarity search [27], their potential for time series anomaly detection is only partially exploited. From those techniques found in the literature, most are designed for high-frequency anomaly detection (e.g. in network traffic data), such as [81, 85] and [100]. The early work of [6, 5] describes anomaly detection based on non-decimating wavelet transforms. [74] developed an anomaly detection algorithm for time series, based on wavelets, neural networks and Hilbert transforms. The algorithm was tested on a relatively simple benchmark, including two synthetic time series. Shahabi et al. [148] developed an anomaly detection approach which is based on the visual inspection of the DWT of time series data. In this chapter, 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) [89] and the other being a subset of the Yahoo's S5 Webscope benchmark [87]. We compare our algorithm with the already introduced state-of-the-art anomaly detectors: Numenta's NuPIC, based on Hierarchical Temporal Memory (HTM) [49], our previous algorithm SORAD (Chapter 4 & [163]) which is specialized on short-term anomalies, and Twitter's ADVec algorithm [173].

In the following Section 5.2, we describe the unsupervised algorithm which uses **D**iscrete **W**avelet **T**ransform and **M**aximimum **L**ikelihood **E**stimation for **A**nomaly **D**etection in

time series. Note, however, that the name might be slightly misleading. Initially, we assumed that the data-generating distributions are Gaussian. Hence, we used maximum likelihood estimation (MLE) to estimate the parameters of a Gaussian distribution (sample mean and covariance) which are required to compute the Mahalanobis distance. However, we noticed later that the assumption of normality is not generally necessary since the Mahalanobis distance does not impose any conditions on the distribution of the data.
For each time series the DWT-MLEAD algorithm (i) computes a *decimating* DWT using Haar wavelets, (ii) estimates a mean and covariance matrix for the individual frequency scales of the DWT, (iii) flags unusual data points in each frequency scale based on a (Mahalanobis-distance-based) quantile estimate, and finally (iv) aggregates the unusual data points of all frequency scales and detects anomalies in the original time series. We perform a few initial experiments with the offline DWT-MLEAD algorithm and shortly discuss the results.

In Section 5.3, we describe the necessary modifications in order to make DWT-MLEAD fully online. In order not to violate causality-constraints, the resulting algorithm is partially quite different from its offline counterpart. We compare online DWT-MLEAD on all time series taken from the Yahoo S5 data and NAB. Section 5.2.5 concludes this chapter and gives an outlook on possible future work.

## 5.2 The Offline DWT-MLEAD Algorithm

### 5.2.1 Methods

#### 5.2.1.1 Wavelet Transforms

Wavelet transforms [111] 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. For sampled time series data, often the so called discrete wavelet transform (DWT) is applied, which has linear time complexity and can be implemented efficiently for time series streams using finite impulse response (FIR) filters. Usually a decimating DWT is performed, in which the filtered series are downsampled. The DWT decomposes the original time series into so called approximation and detail coefficients which are arranged in different levels. Due to the decimating (downsampling) property of the DWT one can represent both coefficient sets in two binary tree structures.

In its current form, DWT-MLEAD performs a decimating DWT using Haar wavelets (other wavelets are also applicable, but require some additional considerations) on each time

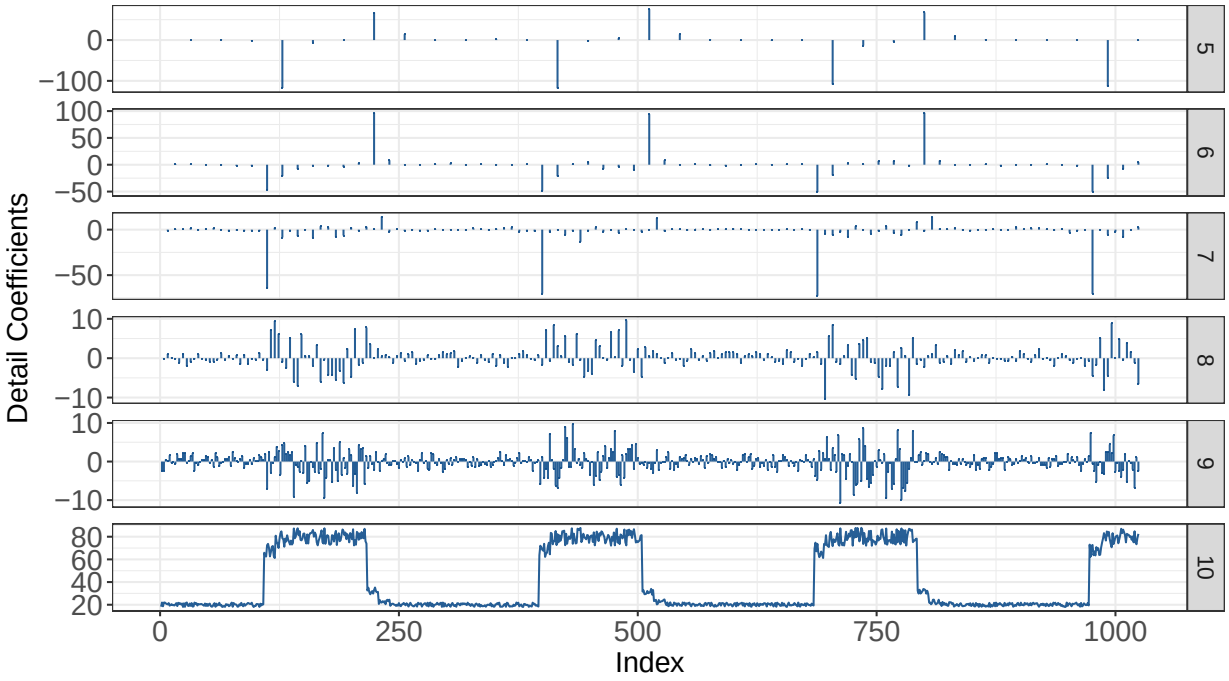


**Figure 5.1:** Example of a decimating DWT using Haar Wavelets for a time series of the NAB data. The original time series is depicted on scale 10. On the scales 5–9 the detail coefficients of the DWT are shown. While we move towards lower scales, the number of coefficients is decimated (i.e. halved in each step), with 32 coefficients left on scale 5.

series. For this purpose, the R-package *wavetresh* [117] 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 $N = 2^{\lceil \log_2(n) \rceil}$, 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 7–8 in Algorithm 4), where $\ell$ addresses the level and $k \in 1, \ldots, N$ the time index.

The lowest level $\ell = \log_2(N)$ contains only one coefficient. The highest level $\ell = 0$ has no approximation coefficients but only detail coefficients $d_{k,0}$ which represent the original time series. Since lower levels of the DWT usually do not contain patterns which are useful for anomaly detection, only the $L$ highest levels ($L$ is a parameter of the algorithm) are considered, where $\ell = L - 1$ describes the lowest considered level and $\ell = 0$ addresses the highest possible level (the original time series). In Fig. 5.1 the DWT of a time series from NAB is illustrated.

#### 5.2.1.2 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. Our algorithm employs an individual sliding window for the detail and approximation coefficients at each level of the DWT tree.

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 4, lines 10–11), a window of size $w_\ell$ 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 mean vector and a covariance matrix are estimated, as described in the following.

#### 5.2.1.3 Detecting Unusual Patterns on Individual Frequency Scales

In order to distinguish between normal and unusual patterns in the individual levels of the DWT, our algorithm estimates two parameters for each considered level: a mean vector $\bar{\mathbf{x}}$ and a covariance matrix $\bar{\mathbf{\Sigma}}$. This is done separately for the approximation and detail coefficients ($c_{n,\ell}$ and $d_{n,\ell}$). The function ESTIMATE in Algorithm 5 estimates $\bar{\mathbf{x}}$ and $\bar{\mathbf{\Sigma}}$ for a given matrix $\mathbf{X}$, where $\mathbf{X} \in \mathbb{R}^{n \times w_\ell}$, with $n = N - w + 1$ being the number of input vectors generated by sliding the window over the time series, $\bar{\mathbf{x}} \in \mathbb{R}^{w_\ell}$ is a $w$-dimensional vector, which indicates the center of the distribution, and $\bar{\mathbf{\Sigma}} \in \mathbb{R}^{w_\ell \times w_\ell}$ describes the covariances between individual dimensions. The dimensions of both quantities depend on the length of the sliding window $w_\ell$ used in each level of the DWT. In Algorithm 4, line 13, DWT-MLEAD estimates both parameters for each $\mathbf{D}^{(\ell)}$ and $\mathbf{C}^{(\ell)}$.

For a given estimate $(\bar{\mathbf{x}}, \bar{\mathbf{\Sigma}})$, one can compute the so-called squared Mahalanobis distance for a new observation $\mathbf{x}_i \in \mathbb{R}^{w_\ell}$:

$$m(\mathbf{x}_i) = (\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T} \bar{\mathbf{\Sigma}}^{-1} (\mathbf{x}_i - \bar{\mathbf{x}}) \tag{5.1}$$

The Mahalanobis distance is a measure which can be used as an indicator for unusual behavior. If the Mahalanobis distance is large, it is likely that the observed point $\mathbf{x}$ does not represent a normal instance.

Note that using the Mahalanobis distance in this sense as anomaly indicator does not require any assumption about the distribution of the data $\mathbf{x}_i$. It is often found in the literature that Mahalanobis distance requires the data to be normally distributed, but this is not necessary to make $m(\mathbf{x}_i)$ a useful norm. (If the data were Gaussian distributed, then $m(x_i)$ would follow a Chi-Squared distribution, as derived in Appendix B.3, but we do not need this fact in our algorithm.)

For every entry in $\mathbf{D}^{(\ell)}$ and $\mathbf{C}^{(\ell)}$ we compute and collect the squared Mahalanobis distances in a vector $\mathbf{m}$ using the previously determined parameters $\bar{\mathbf{x}}$ and $\bar{\mathbf{\Sigma}}$. This is done in function MAHALANOBIS of Algorithm 5.

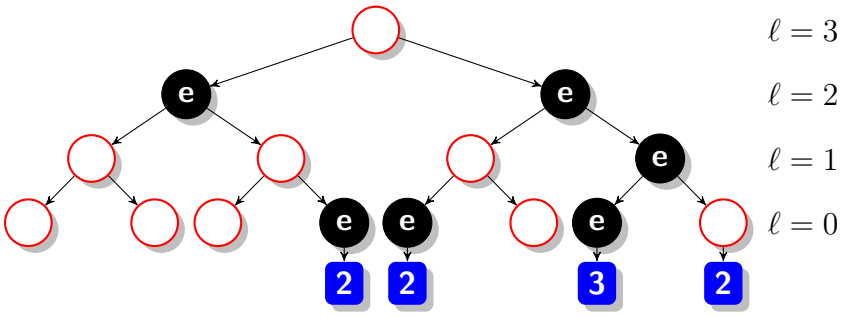


**Figure 5.2:** Detecting anomalies with leaf counters. Along the vertical axis are the DWT levels $\ell$, along the horizontal axis are the time indices $k$. The leftmost event $e$ thus comes from either an unusual $c_{1,L-2}$ or $d_{1,L-2}$. Each event increases the leaf counters (blue rectangles) connected with the $e$ node. Only counters with count $\geq 2$ are shown.

#### 5.2.1.4 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 threshold for the squared Mahalanobis distance. We use an empirical $\epsilon$-quantile $m_\epsilon$ (e.g. the $99^{\text{th}}$ percentile) to determine the threshold. After determining $m_\epsilon$ in Algorithm 4, line 15, instances are flagged as "unusual" in a binary vector $\mathbf{a}$ if their (squared) Mahalanobis distance $m_i$ lies above $m_\epsilon$ (line 16).

#### 5.2.1.5 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 $\ell$ of the DWT is flagged as unusual (has a flag $a_k = 1$) then an event $e$ – marked as a black node in Fig. 5.2 – 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. 5.2) which counts all such events (Algorithm 4, line 17). After all events are processed, all counters with a count $h_i < 2$ are deleted (line 18).

#### 5.2.1.6 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 4, line 19). 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. 5.2 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 24). 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.

**Algorithm 4** Offline DWT-MLEAD, an anomaly detection algorithm based on the Discrete Wavelet Transform (DWT).

Define:
    $L$: Number of levels considered in the DWT
    $\epsilon$ 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($y = (y_1, y_2, \ldots, y_N)$) ▷ $N$ is a power of 2
    Compute DWT of $y$ for levels $\ell \in \{0, \ldots, L-1\}$
    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: $w_\ell = \max\{2, \log_2(m) - \ell\}$
    $\forall \ell \in \{0, \ldots, L-1\}$: Build $\mathbf{D}^{(\ell)}$, $\mathbf{C}^{(\ell)}$ by sliding window of size $w_\ell$ over $d_{k,\ell}$, $c_{k,\ell}$
    **for all** $\mathbf{X} \in \{\mathbf{D}^{(\ell)}, \mathbf{C}^{(\ell)} \mid \ell = 1, \ldots, L-1\} \cup \mathbf{D}^{(0)}$ **do**
        $(\bar{\mathbf{x}}, \bar{\mathbf{\Sigma}}) =$ ESTIMATE($\mathbf{X}$) ▷ Defined in Algorithm 5
        $\mathbf{m} =$ MAHALANOBIS($\mathbf{X}, \bar{\mathbf{x}}, \bar{\mathbf{\Sigma}}$) ▷ Defined in Algorithm 5
        Determine threshold $m_\epsilon$ as empirical $\epsilon$-quantile or with $m_\epsilon = \chi^2_{1-\epsilon}(w_\ell)$
        $\mathbf{a} =$ PREDICT($\mathbf{m}, m_\epsilon$) ▷ Defined in Algorithm 5
        For all $\mathbf{a}_i = 1$: 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 5** Helper functions for Algorithm 4.

**function** ESTIMATE($\mathbf{X}$)
  $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ ▷ Vetor $\mathbf{x}_i \in \mathbb{R}^w$ is the $i$th row of matrix $\mathbf{X} \in \mathbb{R}^{n\times w}$
  $\bar{\mathbf{\Sigma}} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T}$
  **return** $(\bar{\mathbf{x}}, \bar{\mathbf{\Sigma}})$

**function** MAHALANOBIS($\mathbf{X}, \bar{\mathbf{x}}, \bar{\mathbf{\Sigma}}$)
  $\mathbf{m}$ : vector of size $\mathbb{R}^n$ ▷ $n$ is the number of rows in $\mathbf{X}$
  **for each** row $\mathbf{x}_i$ of $\mathbf{X}$ **do**
    $m_i = (\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T}\bar{\mathbf{\Sigma}}^{-1}(\mathbf{x}_i - \bar{\mathbf{x}})$
  **return m**

**function** PREDICT($\mathbf{m}$, $m_\epsilon$)
  $\mathbf{a}$: vector of same size as $\mathbf{m}$
  **for all** $\mathbf{a}_i$ **do**
    $\mathbf{a}_i = \begin{cases} 1, & \textbf{if } m_i > m_\epsilon \\ 0, & \textbf{otherwise} \end{cases}$ ▷ Binary anomaly flag vector
  **return a**

### 5.2.2 Algorithms and their Settings

In the following, we compare DWT-MLEAD with three online anomaly detection algorithms, namely SORAD, NuPIC, and ADVec. We compare the algorithms on the A3 data of Yahoo's Webscope S5 benchmark and on NAB (introduced in 2.3). 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 4 have to be set, which are fixed for the whole dataset: a threshold $\epsilon \in [0, 1]$ for the $\epsilon$-quantiles, which is varied to adjust the tradeoff between precision and recall, a parameter $B$ (threshold for counter sum), and the number of considered levels $L$. From Sec. 5.2.1.4, we use the empirical quantiles for the NAB data and the $\chi^2$-based quantiles for the A3 data. We empirically determined the setting $B = 3.5$, $L = \log_2(N) - 5$ for the NAB data and $B = 1$, $L = \log_2(N) - 7$ for the A3 data (longer time series also consider more DWT levels). The window size $w_\ell$ is set by Algorithm 4 in a level-dependent fashion. In its current form the DWT-MLEAD algorithm

**Table 5.1:** 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. (2.1)–(2.3), 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) |

operates offline on each time series, the remaining algorithms investigated in this section are all online.

**SORAD** The algorithm's parameters which are set as follows for the experiments: We set the forgetting factor of the algorithm to $\lambda = 0.98$, the anomaly threshold $\epsilon$ 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** NuPIC is described in Section 3.1.2. 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 $\epsilon$ in SORAD and DWT-MLEAD – trades off precision and recall.

**ADVec** Twitter's ADVec Algorithm (Section 3.1.1) is the last algorithm which we will review in this section. The algorithm requires three main parameters, which are as follows: The first parameter $\alpha$ is used as anomaly threshold. The second parameter, the period-length, is fixed to the value 40, which has shown to give the best results on the investigated data. Finally, we found that the setting of the parameter $\max_{anoms}$ is crucial for the performance of ADVec, especially on the NAB dataset. We choose $\max_{anoms} = 1\%$ for the A3 data and $\max_{anoms} = 0.1\%$ for the NAB data.

**Table 5.2:** 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 | $13.6 \pm 0.3$ | $34.6 \pm 0.1$ | $810.9 \pm 1.3$ | $2.6 \pm 0.2$ |
| NAB | $12.2 \pm 0.2$ | $111.6 \pm 0.2$ | $1636.4 \pm 2.7$ | $5.8 \pm 0.5$ |

### 5.2.3 Results for the Offline DWT-MLEAD Algorithm

Table 5.1 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 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. 5.3. 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 section 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. 5.4 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 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 5.2, 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.

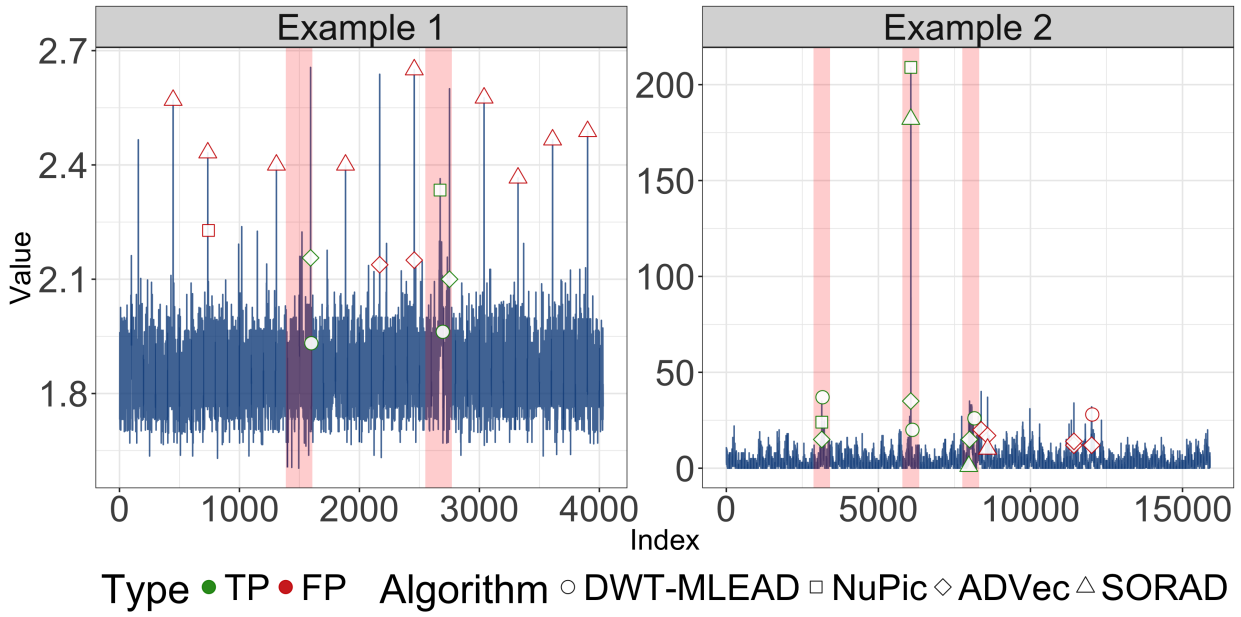


**Figure 5.3:** Example time series taken from the NAB data with the anomalies detected by the algorithms DWT-MLEAD, NuPIC, ADVec, and SORAD. The red vertical bars in the plot indicate the true anomaly windows. True-positives are indicated by green colors while False-positives are colored red.

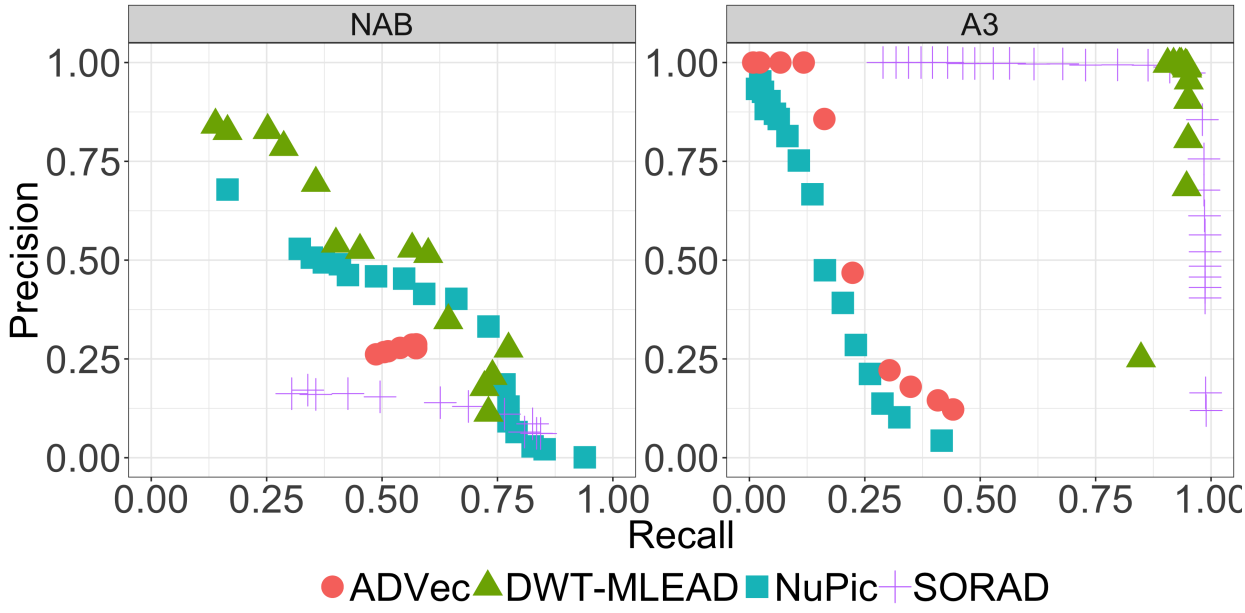


**Figure 5.4:** Multiobjective plot for the NAB and A3 dataset. Precision and recall are computed based on the results of *all* time series of the corresponding data set.

### 5.2.4 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. 5.2 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.

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

- 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.). This limitation will be lifted in Section 5.3.
- We assume a certain degree of stationarity for the algorithm to work. Trends and change-points cannot be handled well in the offline form. A (semi-)online version could offer more flexibility in the sense that trends and change-points can be learned.
- 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 $\ell$ considered in Algorithm 4. In such cases it might help to extend the algorithm by a periodicity detector and subtract such a periodicity prior to analyzing the time series with DWT-MLEAD.

### 5.2.5 Summary

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. However, as mentioned before, DWT-MLEAD is not online yet. In the following section, we add several modifications to the algorithm to operate fully online.

## 5.3 Online-Adaptable DWT-MLEAD Algorithm

In order to turn the offline DWT-MLEAD algorithm, presented in the previous section, into a fully online algorithm, several points have to be taken into consideration: In an online setup, a causal implementation of all algorithm's components is required. The DWT, the sliding

window approach, the estimation of mean and covariance, and the event detection on the individual layers of the DWT tree have to be modified to satisfy the causality requirement. Furthermore, not all components can be directly translated into an incremental (online) version. For example, it is not straight-forward to estimate a weighted sample mean and covariance matrix incrementally, and an online version has to be derived (see Appendix B.2). Also, stability issues in online settings have to be considered while ensuring that the model acquires new knowledge as fast as possible (stability-plasticity dilemma). In the following we present a solution for an online DWT-MLEAD algorithm and study it on a comprehensive benchmark.

### 5.3.1 Online and Causal DWT & Sliding Windows

The online DWT-MLEAD algorithm also utilizes both the detail coefficients $d_{k,\ell}$ and the approximation coefficients $c_{k,\ell}$. For the online implementation of the algorithm, a strictly causal computation scheme is adhered to: For example, two data points in the original time series have to be collected first before the next coefficient in level $\ell = 1$ can be computed. Similarly, $2^\ell$ data points from the original time series are necessary to compute the next coefficient in level $\ell$.

We experimented with the length of the sliding windows and found that for the online version of DWT-MLEAD, window sizes in the form of $w_\ell = \max\{1, \lfloor b^{o-\ell} \rfloor\}$ appear to be the best choice. $b, o \in \mathbb{R}$ are two additional hyperparameters of the algorithm. As soon as a new coefficient in level $\ell$ is available ($c_{n,\ell}$ or $d_{n,\ell}$), the corresponding window is slid one step further and the new window embedding is collected and passed to a model, which estimates the likelihood of observing such a vector. Unlikely vectors would indicate unusual behavior on the corresponding DWT level. The sliding windows at lower levels are moved with a slower rate than those on higher levels, since new coefficients are only generated after every $2^\ell$ time steps in the original time series. This is necessary, to ensure causality of the system.

Anomaly detection starts after an initial transient phase, when the sliding windows can be completely filled.

### 5.3.2 Online Estimation of the Mean and Covariance Matrix

Since the DWT-MLEAD algorithm operates in an online fashion, the parameter estimations also have to be updated incrementally for each new data point. For this purpose we use an exponentially decaying weighted estimator with an forgetting factor $\lambda \in (0, 1]$. The forgetting factor controls at which rate past observations fade out over time. A value of $\lambda$ close to 1 results in an algorithm with a very long memory, whereas small values (usually not smaller than 0.9) can significantly limit the memory of the estimator. By allowing the estimator to gradually forget historic information, the algorithm can adapt to new concepts in the data stream. Furthermore, with $\lambda < 1$ we can prevent (under most conditions) a numeric overflow of the required accumulator (the sum of squares of differences from the current mean). However, forgetting can also lead to a higher variance in the parameter

estimates. The pseudo-code of the estimator can be found in Alg. 7, lines 1 – 8. Note that it is not actually necessary to compute the covariance matrix, since only its matrix inverse is required in later steps. Therefore, we directly estimate the inverse of the sum of squares of differences from the current mean $\bar{\mathbf{M}}_n^{-1}$. Since the inverse $\bar{\mathbf{M}}_n^{-1}$ has to be re-computed for every new data point, which can be computationally expensive for larger dimensions, we use the Sherman-Morrison formula [150] to incrementally update $\bar{\mathbf{M}}_n^{-1}$. The inverse of the covariance matrix is given by $\bar{\mathbf{\Sigma}}_n^{-1} = W_n \bar{\mathbf{M}}_n^{-1}$. A detailed derivation of the exponentially decaying estimator of the mean and covariance matrix is given in Appendix B.2.

### 5.3.3 Detecting Events in the DWT Tree and Anomaly Detection

Since DWT-MLEAD estimates a mean and covariance matrix for every set of DWT-coefficients on the levels $\ell \in [0, 1, \ldots, L]$, it is possible to examine each newly observed value $c_{n,\ell}$ and $d_{n,\ell}$ in the context of its current sliding window in order to detect unusual patterns. For each new data point the current window embed vector is determined and the squared Mahalanobis distance $m_{\mathbf{x}_n}$ to the center of the distribution is computed for this vector. Subsequently, this distance is compared to a threshold $m_\epsilon$. We assume that $\mathbf{x}_n$ is roughly Gaussian distributed. Since a Gaussian random variable has a squared Mahalanobis distance to its mean, which is Chi-squared ($\chi^2$) distributed with $w_\ell$ degrees of freedom, we set $m_\epsilon$ by simply computing the $(1-\epsilon)$-quantile of the $\chi^2$-distribution (function PREDICT in Algorithm 7, lines 10–15). Although the assumption of a Gaussian distribution is often violated in practice, we found that the threshold $m_\epsilon$, determined based on the $\chi^2$-distribution, usually is a reasonable choice. We also experimented with other online approaches, such as the $P^2$ algorithm [70], designed to incrementally track quantiles, but found the $\chi^2$-quantiles to deliver better results.

If the Mahalanobis distance $m_{\mathbf{x}_n}$ exceeds the threshold $m_\epsilon$, the current instance $c_{n,\ell}$ or $d_{n,\ell}$ is flagged as unusual and an event $e$ is passed down the DWT tree, as illustrated in Fig. 5.5. Events arriving at the leaf nodes are summed up in a global, exponentially decaying event counter $E_i$ (Algorithm 6, line 25). If the activity in a subtree of the DWT exceeds a certain limit, hence, if many events are produced in a short time, $E_i$ will increase fast. As soon as $E_i$ is larger than a specified threshold $B$, an anomaly will be fired and the instance $i$ in the time series will be flagged. In order to avoid many detections in a short time, a new anomaly cannot be fired again until $E_i$ has faded away and falls below threshold $\frac{2}{3}B$.

In order to detect extreme outlier events, a simple heuristic is used: The algorithm flags a point as anomalous, if it exceeds the current minimum/maximum by more than 20% of the min-max range.

### 5.3.4 Algorithmic Setup

In this section, we compare online DWT-MLEAD to two other online anomaly detection algorithms. For each algorithm *one* standard parameter setting is chosen which is then used for all experiments across all datasets. Only an anomaly threshold parameter is varied

**Algorithm 6** An online version of DWT-MLEAD, an anomaly detection algorithm using the Discrete Wavelet Transform.

**Define parameters**:
  $L$: maximum number of levels considered in the DWT
  $b$, $o$: for the computation of the sliding window sizes $w_\ell$
  $\lambda$: forgetting factor for the estimation of the Gaussian distributions
  $\epsilon$: quantile of $\chi^2$-distribution
  $B$: threshold for global event counter that triggers an anomaly

**Initialize**:
  Set window sizes for each level: $w_\ell = \max\{1, \lfloor b^{o-\ell} \rfloor\}$
  Global event counter: $E_0 = 0$
  Discount factor: $\gamma = \frac{w_L - 1}{w_L + 1}$
  Allow to trigger anomaly with: $A = \text{true}$
  Initialize all $P_0^{(c,\ell)}$ and $P_0^{(d,\ell)}$ with the tuple $(W_0, \bar{\mathbf{x}}_0, \bar{\mathbf{M}}_0^{-1}, \bar{\mathbf{M}}_0)$, where:
    $W_0 \in \mathbb{R}$, $\bar{\mathbf{x}}_0 \in \mathbb{R}^{w_\ell}$ and, $\bar{\mathbf{M}}_0^{-1}, \bar{\mathbf{M}}_0 \in \mathbb{R}^{w_\ell \times w_\ell}$
    $W_0 = 0$, $\bar{\mathbf{x}}_0 = \mathbf{0}$, $\bar{\mathbf{M}}_0^{-1} = \bar{\mathbf{M}}_0 = \mathbf{I}$

**function** DWTMLEAD($i$, $y_i$) ▷ where $y = (y_1, y_2, \ldots)$ is a streaming time series
  Determine $\ell' = \min(L - 1, \max\{\ell^* \in \mathbb{N}_0 \mid i \mod 2^{\ell^*} = 0\})$
  **for all** $\ell \in \{0, \ldots, \ell'\}$ **do**
    $n = i/2^\ell$
    Compute DWT coefficients $c_{n,\ell}$ and $d_{n,\ell}$ ▷ if not already present
    $\mathbf{x}_n^{(c)} = (c_{n-w_\ell+1,\ell} \quad \cdots \quad c_{n,\ell})^\intercal$ , $\mathbf{x}_n^{(d)} = (d_{n-w_\ell+1,\ell} \quad \ldots \quad d_{n,\ell})^\intercal$ ▷ sliding windows
    $P_n^{(c,\ell)} = \text{UPDATE}(P_{n-1}^{(c,\ell)}, \mathbf{x}_n^{(c)}, \lambda)$ , $P_n^{(d,\ell)} = \text{UPDATE}(P_{n-1}^{(d,\ell)}, \mathbf{x}_n^{(d)}, \lambda)$
    $e_\ell = \text{PREDICT}(P_n^{(c,\ell)}, \mathbf{x}_n^{(c)}, \epsilon) + \text{PREDICT}(P_n^{(d,\ell)}, \mathbf{x}_n^{(d)}, \epsilon)$
  $E_i = \gamma E_{i-1} + \sum_{j=0}^{\ell'} e_j$ ▷ Adjust global event counter
  $a_i = \begin{cases} \text{true}, & \textbf{if } A \wedge E_i \geq B \\ \text{false}, & \textbf{otherwise} \end{cases}$ ▷ Flag anomaly at time step $i$, if threshold is exceeded
  **if** $a_i$ **then** $A = \text{false}$
  **if** $E_i < \frac{2}{3}B$ **then**
    $A = \text{true}$ ▷ Allow new anomaly, if event-counter value falls below threshold
  **return** $a_i$

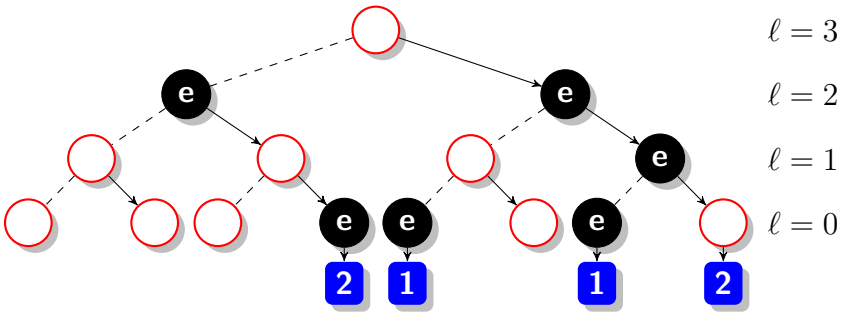


**Figure 5.5:** Detecting anomalies online with leaf counters. All coefficients (except on the leafs) are always computed bottom-up, based on two child nodes (connected with one dashed and one solid edge). Along the vertical axis are the DWT levels $\ell$, along the horizontal axis are the time indices $n$ of the coefficients of the DWT. E.g., the leftmost event $e$ comes from either an unusual $c_{n,2}$ or $d_{n,2}$. Each event is passed down the tree only along the solid edges (causal computation) and increases the right-most leaf counter (blue rectangle) connected with the $e$ node.

for each algorithm and dataset in order to balance precision and recall in a way that the $F_1$-score is maximized.

**DWT-MLEAD** As described in Sec. 5.3, in total 6 parameters have to be selected by the user. In order to find an appropriate setting, we did not systematically tune the parameters. Instead, we generated 60 design points using latin hypercube sampling (LHS) and evaluated the algorithm on all time series for these points. The setting $B = 2.20$, $b = 2.27$, $o = 6$, $L = 5$, $\lambda = 0.972$ achieved the highest average $F_1$-score and will be used throughout the rest of this paper. The parameter $\epsilon$ is used as anomaly threshold and is adjusted in the range $\epsilon \in [10^{-6}, 10^{-1}]$. Additionally, to exclude the possibility of overtuning on the data sets, we made the following experiment: We separated the set of all time series in a training and a test set (each containing 50% of the time series) and tuned the parameters of DWT-MLEAD only on the training data. Then the $F_1$-score was established only on the test set. The results will be given below under the name TRAIN-TEST-SEP.

**NuPIC** Similarly to the previous section, we run NuPIC with its standard parameter settings. Similarly to DWT-MLEAD, an anomaly threshold can be varied in the interval $[0, 1]$ to control the sensitivity of the algorithm.

**ADVec** The main three parameters were tuned to achieve the highest average $F_1$-score. The period-length is set to 40. The second parameter $\max_{anoms}$ is set to $\max_{anoms} = 0.003$. The last parameter $\alpha$ is used as anomaly threshold for ADVec and is adjusted in the range $\alpha \in (10^{-6}, 1)$.

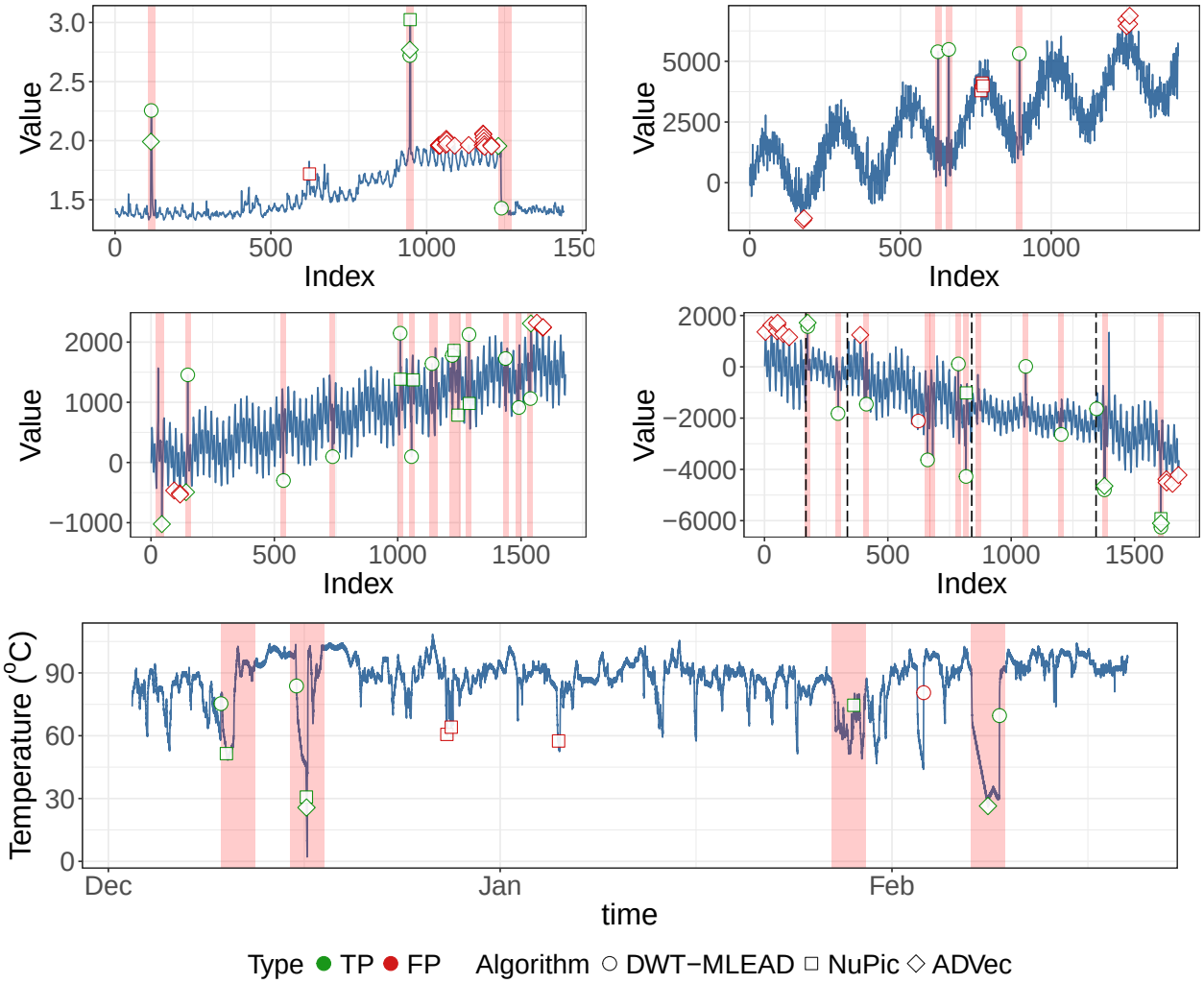


**Figure 5.6:** Example time series taken from the Yahoo Webscope S5 data and the Numenta Anomaly Benchmark (NAB). In each graph the real anomalies are indicated by the light-red shaded areas. Three algorithms are tested on this data and the individual detections are shown with different symbols. The color of the symbol indicates if the detections were correct (green) or false (red).
Top two rows: One example each from the A1–A4 data. The dashed vertical lines in the A4 data indicate concept changes which should also be detected by the anomaly detectors.
Bottom: Example time series taken from the NAB data. The graph shows the temperature sensor data of an internal component of a large industrial machine over its last few months of operation. The second anomaly (mid of December) is a planned shutdown of the machine. The catastrophic failure occurs end of February when the recordings end.

**Algorithm 7** Helper functions for Algorithm 6.

**function** UPDATE($P_{n-1}, \mathbf{x}_n, \lambda$) ▷ $\mathbf{x}_n \in \mathbb{R}^{w_\ell}$, $w_\ell$ is the size of the window at scale $\ell$

  $(W_{n-1}, \bar{\mathbf{x}}_{n-1}, \bar{\mathbf{M}}_{n-1}^{-1}, \bar{\mathbf{M}}_{n-1}) = P_{n-1}$ ▷ Matrix $\bar{\mathbf{M}}_{n-1}$ is optional (debugging purposes)

  $W_n = \lambda W_{n-1} + 1$

  $\boldsymbol{\Delta}_n = \mathbf{x}_n - \bar{\mathbf{x}}_{n-1}$

  $\bar{\mathbf{x}}_n = \bar{\mathbf{x}}_{n-1} + \frac{1}{W_n}\boldsymbol{\Delta}_n$

  $\bar{\mathbf{M}}_n = \lambda\bar{\mathbf{M}}_{n-1} + \boldsymbol{\Delta}_n(\mathbf{x}_n - \bar{\mathbf{x}}_n)^\intercal$ ▷ Optional, since only inverse $\bar{\mathbf{M}}_n^{-1}$ is required later

  $\bar{\mathbf{M}}_n^{-1} = \frac{1}{\lambda}\bar{\mathbf{M}}_{n-1}^{-1} - \frac{\frac{1}{\lambda}\bar{\mathbf{M}}_{n-1}^{-1}\boldsymbol{\Delta}_n(\mathbf{x}_n-\bar{\mathbf{x}}_n)^\intercal\bar{\mathbf{M}}_{n-1}^{-1}}{\lambda+(\mathbf{x}_n-\bar{\mathbf{x}}_n)^\intercal\bar{\mathbf{M}}_{n-1}^{-1}\boldsymbol{\Delta}_n}$ ▷ Sherman-Morrison Formula

  **return** $(W_n, \bar{\mathbf{x}}_n, \bar{\mathbf{M}}_n^{-1}, \bar{\mathbf{M}}_n)$ ▷ Return updated parameters

**function** PREDICT($P_n, \mathbf{x}_n, \epsilon$) ▷ $\mathbf{x}_n \in \mathbb{R}^{w_\ell}$, where $w_\ell$ is the size of the window at scale $\ell$

  $(W_n, \bar{\mathbf{x}}_n, \bar{\mathbf{M}}_n^{-1}, \bar{\mathbf{M}}_n) = P_n$

  $m_{\mathbf{x}_n} = W_n(\mathbf{x}_n - \bar{\mathbf{x}}_n)^\intercal\bar{\mathbf{M}}_n^{-1}(\mathbf{x}_n - \bar{\mathbf{x}}_n)$ ▷ Mahalanobis distance of $\mathbf{x}_n$ to $\bar{\mathbf{x}}_n$

  $m_\epsilon = \chi^2_{1-\epsilon}(w_\ell)$ ▷ Threshold: upper $\epsilon$-quantile of $\chi^2$-distribution

  $e_n = \begin{cases} 1, & \textbf{if } m_{\mathbf{x}_n} > m_\epsilon \\ 0, & \textbf{otherwise} \end{cases}$ ▷ Binary event flag

  **return** $e_n$ ▷ Unusual data points will cause an event in the DWT-tree

**SORAD** We tried to manually select the parameters of SORAD. However, we could not find a good compromise for the parameters, to obtain good results on the Yahoo data and NAB. Hence, we choose the same setting as in Section 5.2.2 and set the forgetting factor of the algorithm to $\lambda = 0.98$, and the window-size is set to $w = 10$.

### 5.3.5 Results

The main results of our experiments are summarized in Tab. 5.3. DWT-MLEAD achieves on all datasets the highest $F_1$-score (the same as SORAD on A1,A2, and A4). NuPIC has a slightly better precision on A1, but on A2, A3 and A4 the difference in all three metrics is large in favor of DWT-MLEAD. One reason, among others, for the weak performance of NuPIC and ADVec could be that the time series in both datasets contain many anomalies, occurring in part at the very beginning of each time series. Hence, the algorithms have to be up-and-running much faster and have to be able to detect anomalies in short time intervals. Furthermore, the A4 time series contain many concept changes, where amplitudes, seasonalities and noise abruptly change. In order to handle such concept changes, a strong online

**Table 5.3:** Results for various algorithms on the datasets A1–A4 and NAB. Shown are the metrics precision (how many percent of the detected events are true anomalies), recall (how many percent of the true anomalies are detected) and $F_1$. All algorithms have their threshold for each dataset chosen such that $F_1$ is maximized. Each algorithm uses otherwise *one standard parameter setting* for all data sets. The values in square brackets show the $F_1$-score on the test data of the experiment TRAIN-TEST-SEP.

| | **Precision, Recall $F_1$-Score** | | | | | |
|---|---|---|---|---|---|---|
| **Algorithm** | **A1** | **A2** | **A3** | **A4** | **NAB** | **Avg** |
| DWT-MLEAD | 0.60, 0.65<br>0.62 [**0.66**] | 1, 0.98<br>**0.99** [**0.99**] | 0.96, 0.97<br>**0.97** [**0.97**] | 0.92, 0.75<br>**0.83** [**0.83**] | 0.66, 0.45<br>**0.54** [**0.52**] | 0.8, 0.76<br>**0.79** [**0.80**] |
| SORAD | 0.65, 0.67<br>**0.66** | 1, 0.98<br>**0.99** | 0.97, 0.95<br>0.96 | 0.9, 0.77<br>**0.83** | 0.09, 0.57<br>0.15 | 0.7, 0.8<br>0.72 |
| NuPIC | 0.62, 0.45<br>0.52 | 0.59, 0.42<br>0.49 | 0.39, 0.20<br>0.27 | 0.41, 0.11<br>0.18 | 0.40, 0.66<br>0.5 | 0.32, 0.37<br>0.39 |
| ADVec | 0.51, 0.56<br>0.54 | 0.66, 0.6<br>0.63 | 0.54, 0.20<br>0.29 | 0.29, 0.15<br>0.2 | 0.11, 0.72<br>0.2 | 0.32, 0.45<br>0.37 |

adaptability is required. For the NAB data, the difference in $F_1$-score between NuPIC and DWT-MLEAD is not that apparent, although there is a slight advantage for our algorithm. Overall, we can observe in column **Avg** that DWT-MLEAD achieves the highest average values for all three metrics.

The results in Tab. 5.3 are for tuning on all data. The additional experiment TRAIN-TEST-SEP (see Sec. 5.3.4) revealed very similar $F_1$-scores (less than 1% deviation in the **Avg** score). This observation confirms that DWT-MLEAD operates well on new data and is not overtuned to its parameters.

Since Tab. 5.3 only captures the results for one specific setting of the algorithms anomaly thresholds, we also measured precision and recall for a wide range of thresholds and plotted them against each other, as shown in Fig. 5.7. The overall picture mostly corresponds to the results shown in Tab. 5.3. Only for the NAB data we can observe that for recall values in the range $[0.5, 0.75]$ NuPIC achieves a higher precision and outperforms DWT-MLEAD. Finally, a look on Tab. 5.3 shows that the NAB dataset is a tough benchmark: All tested algorithms are far from being perfect on that dataset, having $F_1 < 0.55$, i.e. there is still room for improvement.

### 5.3.6 Discussion

Although algorithm DWT-MLEAD could produce good results on the investigated benchmarks, it still has several limitations which leave room for improvement:
(1) For our experiments we only used the relatively simple Haar wavelet. This leads to the limitation that anomalies manifesting themselves in complex frequency patterns might be difficult to detect. Wavelets with stronger localization in the frequency domain (e.g. Gabor

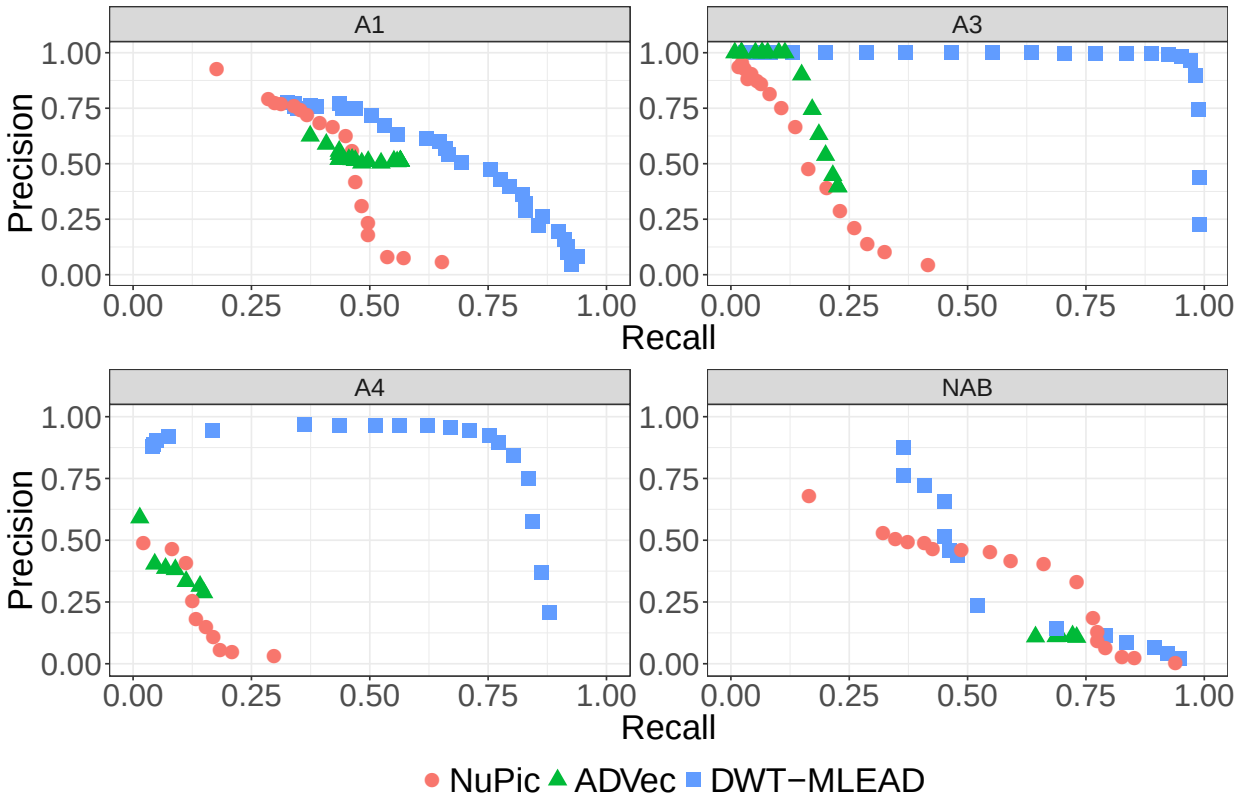


**Figure 5.7:** Multiobjective plot for Yahoo's Webscope S5 benchmark and the Numenta Anomaly benchmark. The graph for the A2 data is not shown here since the results are very similar to the A3 data.

wavelets or ensembles of such wavelets) might allow to detect frequency changes more reliably.

(2) Due to the strictly causal design of the algorithm, events occurring in the DWT-tree might be asymmetrically distributed along the leaf counters (Fig. 5.5). More events will tend to arrive at the leaf nodes on the right side of each sub-tree, which might lead to undesired effects.[2]

(3) Using the Mahalanobis-distance-based metric for the sample mean and covariance matrix to detect unusual behavior might not be the best choice. Other (perhaps multimodal) distributions might be more effective. To test this, we made some runs with Gaussian mixture models (GMM) which are capable to model more complex distributions. So far, however, these runs resulted in only marginal improvements.

[2]We note in passing that we performed runs with an algorithmic variant where we treated each leaf symmetrical: We wait until an $L$-subtree is complete, then we collect all events (along the dashed lines in Fig. 5.5 as well) and process them. The price to pay is a certain delay for some leafs and a deviation from the strict online scheme. The results in terms of precision-recall-metrics are a bit better for NAB and a bit worse for A4. Overall, the difference is only marginal.

The NAB dataset is a challenging benchmark, as it includes mostly real world data from many different applications. The time series contain anomalies in high and low frequencies in a large variety of forms. Many anomalies are also very hard to detect for the human eye without suitable domain knowledge.
It is worth mentioning that DWT-MLEAD proved to perform robustly on all time series, without ever showing numerical instabilities from the matrix updates (function UPDATE in Algorithm 7).

## 5.4 Conclusion & Possible Future Work

In this chapter, we introduced the relatively simple but effective DWT-MLEAD algorithm for offline and online anomaly detection in time series. Generally, the development of widely applicable anomaly detection algorithms is possible when suitable features can be obtained. We found that especially the discrete wavelet transform (DWT) can be an important tool to generate meaningful features across many different frequency scales. Empirical results on a large dataset with 425 time series containing both long-term and short-term anomalies show that DWT-MLEAD is more robust than other state-of-the-art anomaly detectors: Using only *one* fixed parameter setting, (online) DWT-MLEAD achieved an average $F_1$ twice as large as ADVec's & NuPIC's scores and 10% higher than SORAD's. Furthermore, DWT-MLEAD significantly outperformed SORAD on the NAB data, obtaining a 260% higher $F_1$-score. Furthermore, the online adaptability of the DWT-MLEAD algorithm appears to be beneficial in the presence of concept drifts and/or changes, as the results on the A4 data of Yahoo's Webscope S5 benchmark suggest. Our anomaly detection algorithm does not require labeled training data; it is unsupervised and infers from the unlabeled data of each time series what is normal and what is anomalous.

As future work several aspects of our algorithm can be improved: Currently, only simple Haar wavelets are used for the offline and online version of the algorithm; experiments with other wavelets or ensembles of wavelets might lead to a significantly increased performance. Furthermore, it might be possible to further reduce the sensitivity of DWT-MLEAD towards its parameters, for example with automatic parameter tuning methods. Finally, also an extension to multivariate time series should only require a few additional steps.