# An unsupervised methodology for online drift detection in multivariate industrial datasets

Sarah Klein, Mathias Verbeke
*Data and AI Competence Lab (EluciDATA Lab), Sirris*
*Bd. A. Reyerslaan 80, B-1030 Brussels, Belgium.*
*{firstname.lastname}@sirris.be*

***Abstract*—Slight deviations in the evolution of measured parameters of industrial machinery or processes can signal performance degradations and upcoming failures. Therefore, the timely and accurate detection of these drifts is important, yet complicated by the fact that industrial datasets are often multivariate in nature, inherently dynamic and often noisy. In this paper, a robust drift detection approach is proposed that extends a semi-parametric log-likelihood detector with adaptive windowing, allowing to dynamically adapt to the newly incoming data over time. It is shown that the approach is more accurate and can strongly reduce the computation time when compared to non-adaptive approaches, while achieving a similar detection delay. When evaluated on an industrial data set, the methodology can compete with offline drift detection methods.**

***Index Terms*—drift detection, adaptive, multivariate time series**

## I. Introduction

In many industrial processes, the number of sensors integrated in the supporting machinery grows rapidly. Many of these sensors are used for monitoring machine or process conditions, e.g. the power needed for a given process, or the conditions of the surroundings, such as the ambient room temperature. The collected data is particularly useful to detect abnormal behaviour by means of machine learning. For example, a significant temperature increase in a milling machine can indicate a severe problem with the machine. This problem can be maintained when being warned early enough to avoid serious machine damage. In general, there are two different types of abnormal behaviour that can be considered: 1) anomalies, which are (small sets of) single data points that deviate significantly from the normal behaviour, such as too high forces applied during a particular run of a manufacturing process, and 2) slight deviations from the norm over (a longer period of) time, like the increase in temperature discussed above. Especially the latter behavior is harder to detect as it is not a sudden change but an often gradual evolution over time. This becomes even harder when a) the data is multi-variate or b) the detection has to be performed in (near) real-time. The proposed methodology to perform online drift detection in multivariate industrial datasets is aiming to address these challenges.

In the last years, several approaches for drift detection were introduced in the literature [1]–[9]. Those approaches mainly concentrate on so-called concept drift [2], [5]–[9]. Concept drift is a change in the underlying generating statistical distribution of the *output* data. This is especially important for prediction or classification use cases, as a change in concept strongly influences the outcome of the prediction or classification model. In this case, however, we focus on the detection of drift in the *input* data.

Since the aim is to arrive at a method to perform online drift detection in multivariate industrial sensor data, which is inherently dynamic and often noisy, the proposed method uses *adaptive windowing* to allow dynamic adaptation to the new incoming data over time and employs *density approximation by k-Means clustering* as a measure to keep track of the drift in noisy industrial datasets.

The rest of this section is structured as follows: In Section II, related work will be discussed in more detail, before we introduce our own approach in Section III. Initial validation results on artificial data will be discussed in Section IV, before evaluating on an industrial data set in Section V. Finally, in Section VI we discuss and conclude on our results.

## II. Related Work

The interest in online drift detection increased in the last years with the growing number of (near) real-time data streams of diverse sources being available. Note that for drift detection several consecutive data points have to be taken into account as drift is not a local phenomenon. Therefore, most algorithms use sliding windows of fixed size.

For univariate drift detection, one of the most popular algorithms was introduced in 2006 by Bifet and Gavaldà, the so-called ADWIN (Adaptive WINdowing) algorithm [1]. Instead of using a constant window size for drift detection, the algorithm maintains a window of variable size containing the streaming data. The window size is adapted dynamically when no change occurred in the given window with keeping the crucial statistics for further incoming data. Hence, ADWIN can be applied in an online fashion but is not automatically capable of detecting multivariate drift. ADWIN is implemented in the scikit-learn library [10] and therefore is used in many use cases.

Another univariate, online approach which takes more statistical information of the single time series features into account was introduced by Cavalcante *et al* [2]. Their approach, denoted by FEDD, extracts several linear and non-linear features of the time series in order to detect drift. Subsequently, the distance between the feature vectors obtained from two time windows of fixed and equal length is calculated. If the distance between the two feature vectors is significantly different, the algorithm reports a (change in) drift.

Another approach is to estimate the non-drift scenario and then compare the actual incoming data. Wenzel *et al.* [11] use a Kalman filter in order to estimate an hidden baseline signal. Then they estimate the deviation from this baseline signal in order to detect drift in the pure signal.

Several publications in this field rely on statistical testing in order to validate the hypothesis that significant change in the output variable occurs. One of these tests was introduced by Bhaduri *et al.* [12] where a lower bounds were derived by the Hoeffding's and Bernstein inequality.

Kuncheva [3] introduced a multivariate approach based on density estimation of the time series. This semi-parametric log-likelihood (SPLL) detector returns a (change in) drift by comparing the approximation of the density functions within two fixed-size and equally long windows. In order to estimate the density function, a k-Means approximation [13] is used to approximate a Gaussian mixture distribution. The advantage of this approach is that the density estimation is computationally lightweight and at the same time naturally takes the multivariate character of the data into account.

Our proposed approach takes several of the aspects of the upper three algorithms into account. On the one hand, we will integrate the adaptive windowing from ADWIN into the SPLL approach and additionally take further time series characteristics than only the pure data stream values into account. We will discuss our approach in the following section in more detail.

We will benchmark our approach for online drift detection to the drift periods detected by the Python package ruptures [14], which performs multivariate drift detection in an *offline* fashion.

## III. Adaptive SPLL approach

The approach we introduce is based on the SPLL algorithm [3]) where the overall idea is to compare the statistical measures from two approximated distributions. The starting point is a Gaussian mixture with $k$ components given by the density with the mixing coefficients

$$\begin{aligned} p_1(\vec{x}) &= \sum_{j=1}^{k} P(j) p_1(\boldsymbol{x}|j) \\ &= \sum_{j=1}^{k} \frac{P(i)}{(2\pi)^{\frac{n}{2}} \det(\Sigma_i)^{\frac{1}{2}}} \\ &\quad \exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu_i)^T \Sigma_i^{-1} (\boldsymbol{x}-\mu_i)\right). \end{aligned} \tag{1}$$

Due to the sum in the density, the expression will not factorize. Therefore, Kuncheva suggested to rather apply a logarithmic upper bound than the exact likelihood for $\boldsymbol{x}$ in window $W_2$ such that

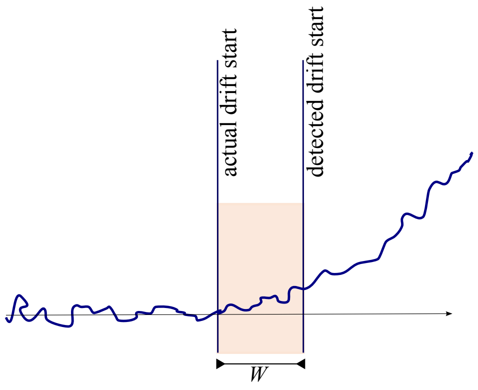

Fig. 1: Schematic definition of the adaptive window $W$ between the actual drift period start time $t_d$ and the current time $t_n$ right after detection. The window will grow with more data points being added to the signal.

$$\begin{aligned} L(W_2|p_1) &= \prod_{x \in W_2} p_1(x) \\ &\leq \prod_{x \in W_2} \max_{i=1}^{k} \left\{ \frac{1}{(2\pi)^{\frac{n}{2}} \det(\Sigma_i)^{\frac{1}{2}}} \right. \\ &\quad \left. \exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu_i)^T \Sigma_i^{-1} (\boldsymbol{x}-\mu_i)\right) \right\} \end{aligned} \tag{2}$$

From this, it can be shown that the bound of the log-likelihood is proportional to the sum of the negative squared Mahalanobis distances between each observation and its closest mean [3]. Note that in the univariate case, this reduces to a simple percentile calculation. In order to decide whether the distributions between the two distinct time windows $W_1$ and $W_2$ are significantly different and therewith drift occurs, the log-likelihood ratio is calculated between the density $p_1$, the density derived from data in window $W_1$ and the density $p_2$ derived from data in $W_2$, as follows:

$$LLR = \log \frac{L(W_2|p_2)}{L(W_2|p_1)} \tag{3}$$

When assuming that $p_2$ is the exact density for data in $W_2$, such that $L(W_2|p_2) = 1$, $LLR$ gives the ratio of $W_2$ fitting into $p_1$:

$$LLR = -\log L(W_2|p_1). \tag{4}$$

Combining the last equation with 2, [3] proposes to use the following upper bound for deciding whether $W_2$ comes from the same distribution as $W_1$:

$$SPLL \propto \sum_{x \in W_2} (\boldsymbol{x}-\mu_*)^T \Sigma_i^{-1} (\boldsymbol{x}-\mu_*) \tag{5}$$

It can be shown that if $W_2$ comes from $p_1$, the Mahalanobis distances have a chi-square distribution with $N$ degrees of freedom, with $N$ being the dimensionality of the feature space [15]. By defining the threshold $\alpha$, it can be decided whether $W_2$ comes from $p_1$ by using the chi-square quantiles.

As suggested by Kuncheva [3], in order to decrease computation time, we will use $k$-Means clustering [13] for estimating the density of the Gaussian Mixture model. The number of clusters $k$ is a free parameter in the algorithm and will influence its performance. Therefore, we will use several different values of $k$ for algorithm evaluation.

While Kuncheva [3] uses different fixed window sizes, we extended her approach by introducing adaptive window sizes, in order to more flexibly adapt to the inherently dynamic behavior present in data originating from industrial machines and processes. As the algorithm will be used for streaming data, we will set the data point $x_n$ at time $t_n$ as the last data point in the incoming data. For the adaptive window, we introduce window $W_S$. It starts at the last detected start of the current drift period $t_d$ and lasts until the current time $t_n$, such that the length of the window is equal to $n - d = W_S$. Note that the size of this window grows until a new drift is detected. A schematic explanation is shown in Figure 1. For detection, we split that window into the following two detection windows $W_S^1$ and $W_S^2$:

- $W_s^1 = [x_d, \dots, x_{d+|W_s|/2}]$
- $W_s^2 = [x_{d+|W_s|/2+1}, ..., x_n]$

If a change was detected between these windows, we continuously split these windows into smaller windows in order to detect the actual starting time of the current drift period. Once found, this point is assigned to be the new $t_d$. Like this we keep track of the current statistics in the current time window of the new incoming data. Note that, once drift was detected, the new detection window will almost surely have a length longer than 1. For reasons of computation time, we introduce a maximal window size $W_{\max} = 100$. Additionally, we reuse the k-Means centroids from the last window computation as initial centroids in the next time step which gives an extra speed-up.

Additionally, we introduce a new criterion specified for drift detection. While [3] solely used the SPLL limit for detecting a change in distribution, we take the evolution of the distance to the cluster centroids - the Mahalanobis distance - into account. It is calculated at every time step during the process. An increase in the mean distance provides additional insights about an occurring drift or whether it is a periodic change. Therefore, we introduce the following criterium $\delta$:

$$D_i = \sum_{x \in W_2} (\boldsymbol{x}_i - \mu_*^i)^T \Sigma_i^{-1} (\boldsymbol{x}_i - \mu_*^i) \text{ with } i \in [d, \dots, n]$$

$$\delta = \begin{cases} 1, & \text{for } D_i + \mathbb{E}[D_{j<i}] > 3\,\mathbb{V}[D_{j<i}] \\ 0, & \text{else} \end{cases} \tag{6}$$

Like this, we additionally keep track of the statistical features of the Mahalanobis distance. Once drift was detected, the vector of Mahalanobis distances will be reset. In Figure 2, the evolution of the log-ratio criterion and the Mahalanobis distance is shown for a two-dimensional dummy data set where

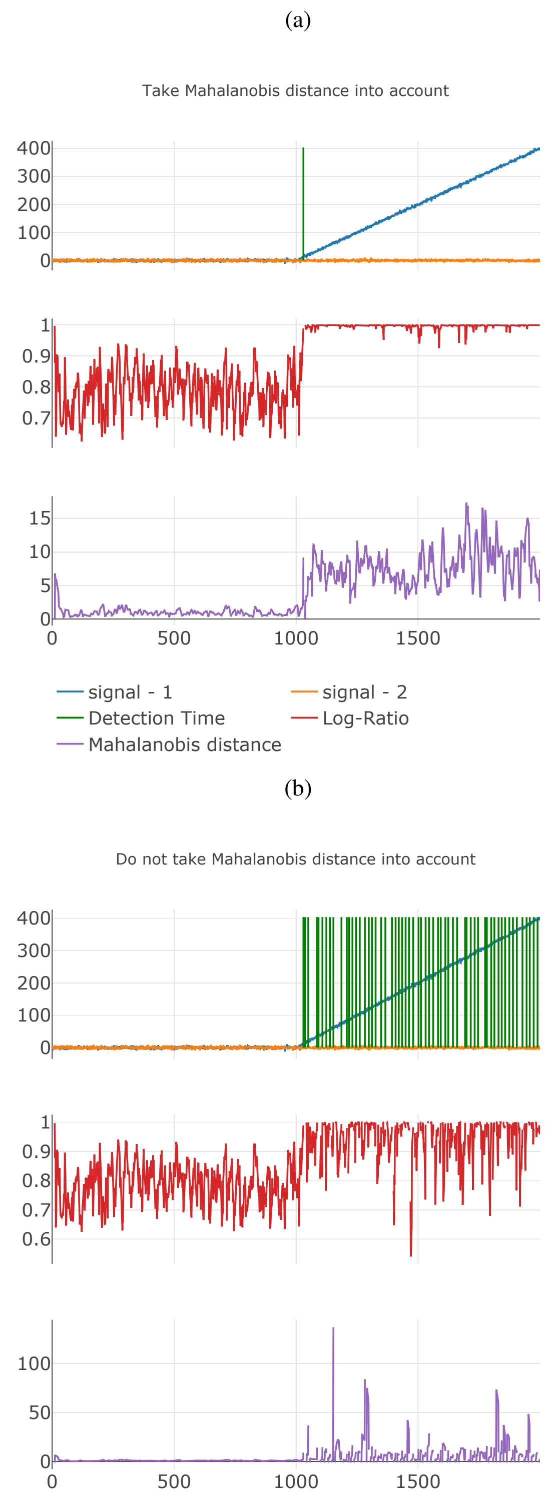

Fig. 2: Drift detection on example data set where in a) the Mahalanobis distance is taken into account and in b) it is not. The upper rows show the two-dimensional signal. In the middle rows, the log-ratio used for testing the chi-squared distribution, is shown, while the bottom rows show the Mahalanobis distance, respectively. (Parameters: $\alpha = 0.025$, $k = 2$)

in the a) the Mahalanobis distance was taken into account and in b) it was not. For the two variables of the dummy data set, the first 1000 data points are drawn from a normal distribution $\mathcal{N}(0,1)$, for the remaining 1000 points, the first variable additionally has an added linear drift.

In both cases, i.e. with the additional Mahalanobis distance criterion, for the first 1000 points, the log ratio criterion is far from 1 and therewith far from detecting a drift. Also the Mahalanobis distance is small. Once the drift starts at point 1000, we see a clear difference between the two cases a) and b). In the first, the log-ratio as well as the Mahalanobis distance increase rapidly. After detecting the first drift, the Mahalanobis distance is not significantly increasing anymore and therefore the criterion for $\delta$ is not fulfilled. Differently for case b), where the Mahalanobis distance is not taken into account. Drift is detected in equidistant times. Both cases can be relevant for industrial use cases. The former is useful in a case in which a drift per se is crucial to detect, while the latter could be used for triggering alarms in case of a continuously growing drift.

## IV. Evaluation on artificial data set

In this section, we will evaluate the adaptive SPLL method on an artifical multi-variate data set with periods of varying drift. First, we discuss the metrics we use in order the evaluate our results. Then, we introduce the algorithm for creating the data set and the parameters we used for the drift detection methods. Then we discuss the results for the different methods of drift detection.

### A. Evaluation metrics

In the case of singular point anomaly detection, a confusion matrix is a good measure for the accuracy of an algorithm. For the evaluation of multivariate drift detection however, a confusion matrix is not applicable, because most algorithms detect drift only after a certain delay and drift is not a local characteristic but evolves with time. For this reason, we use the *adapted F-score* and *mean delay* to evaluate the quality of the model as suggested in [4].

Usually, the F-score is computed as the harmonic mean of recall and precision. As stated above, these measures are only applicable for local classification events. Therefore, we introduce a tolerance threshold $\epsilon$. This means that we accept detected drift to be correctly detected in case the time between actual drift start and drift detection is smaller than $\epsilon$. Hence, recall measures the proportion of actual drifts detected and precision measures the proportion of correctly detected drifts within tolerance. If a second drift start period was detected within tolerance $\epsilon$ even though no new drift period has actually started in the data, this second detection will be counted as false positive. In Figure 3 the definition of true and false positives as well as false negatives are sketched. Note that we do not count true negatives since usually most of all data points should belong to that group. As in streaming data the drift should be detected as early as possible, we consider the mean delay as an additional quality measure. It is calculated as the distance between the labeled starting point of the drift

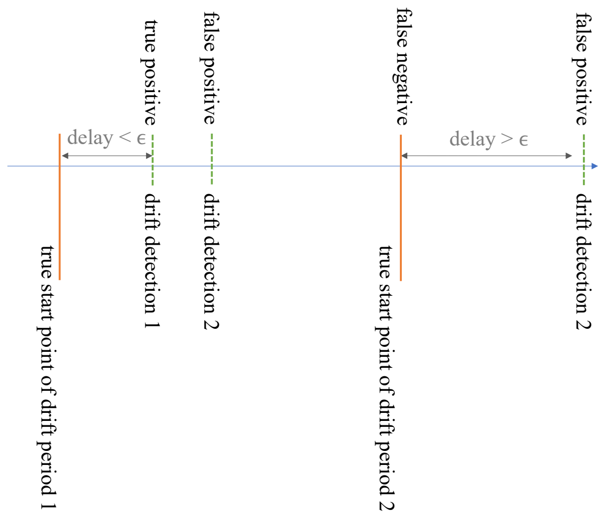

Fig. 3: Schematic explanation of the definition of true and false positives and false negatives. Note that the starting point of the second drift period is indicated as false negative as the delay of detection was bigger than the tolerance threshold $\epsilon$. Therefore this drift start period will not be counted as properly detected.

period and the time when it is detected by the algorithm. If a particular drift period was not detected, we do not assign a delay, such that only true positives count towards the delay calculation.

### B. Artificial data set creation

The evaluation is performed on an artificial dataset of length $n = 2000$ and dimension 4, which is constructed in the following way:

1) Draw from a discrete uniform distribution $m = F(k : a, b)$ a number of change points per dimension, where $a = {}^{n}\!/_{500}$, $b = {}^{n}\!/_{250}$
2) Draw from a Gaussian distribution $x_c p = \mathcal{N}({}^{n}\!/_{m}, {}^{n}\!/_{200})$ the position of the single changepoints per dimension.
3) Draw the single data points between two change points according to the following steps:
    a) Draw a drift variable $\mu$ from a continuous uniform distribution between -1 and 1 defining the strength of the drift.
    b) Draw a uniform noise variable $\sigma$ between 0.01 and 0.5
    c) Draw $|x_{cp}^{i+1} - x_{cp}^{i}|$ values from a Gaussian distribution $\mathcal{N}(\mu, \sigma)$

An example of the sampled data is shown in Figure 4. Note, that this type of introduced drift is not only visible as a (linear) increase or decrease in values but can also be a variation in noise.

For the algorithm evaluation, we sampled 50 of these multivariate time series. On those, we performed the drift detection with the adaptive SPLL algorithm, with the non-adaptive SPLL [3] and with the offline method ruptures [14]

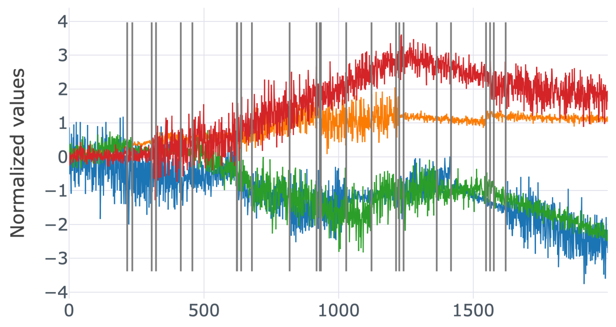

Fig. 4: Example of an artificial data set. The gray, vertical lines indicate the changepoints of the single variables at which the distribution changes and therewith drift occurs.

for benchmarking. We included ruptures for evaluation based on the assumption that an offline algorithm should perform better on the data than an online approach. We will use ruptures out of the box with the following settings for the Pruned Exact Linear Time (PELT) search method:

- Segment model: rbf and L2 method
- Minimal window size $\in \{2, 10, 20\}$

For the adaptive and non-adaptive SPLL approach, we used the following combinations of parameter:

- Chi-squared quantile threshold $\alpha \in \{0.005, 0.01, 0.025, 0.05, 0.075, 0.1\}$
- Number of clusters for k-Means $k\ in \{2, 3, 4\}$
- Window size $|W_1| + |W_2| \in \{50, 100\}$ as suggested in [3] for the non-adaptive SPLL approach.

Hence, the overall amount of experiments is $2 \times 3 \times 50 = 300$ for the adaptive SPLL approach and two times as many experiments for the non-adaptive SPLL approach due to the definition of the two fixed window sizes.

## C. Evaluation

Most important for the methodology evaluation is the overall performance of the algorithm regarding the timely and accurate detection of drift. As stated above, we calculate the F-score and mean delay of detection as the primary measures of performance to compare the different approaches. The results for F-score and mean delay of drift are shown in Figure 5 and Figure 6, respectively. Note, that the maximal mean delay is defined by the tolerance threshold $\epsilon = 50$. In order to interpret the measured F-score in the different scenarios, we need to take a deeper look into the way the non-adaptive SPLL approach identifies drift. The main difference in the drift detection of the adaptive and non-adaptive SPLL approaches is the windowing for drift detection. In the non-adaptive SPLL approach, the window size is fixed and slides with increasing amount of data. That means that if a drift period starts at time $t^*$ then at (almost) all data points in range

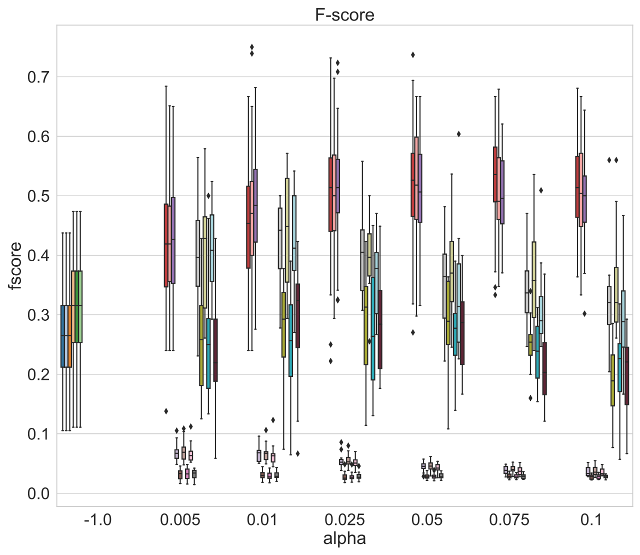

Fig. 5: : Boxplot of F-score for the different combinations of parameters for 50 randomly generated time series. We show the results for adaptive and non-adaptive SPPL approaches as well as the results from ruptures.

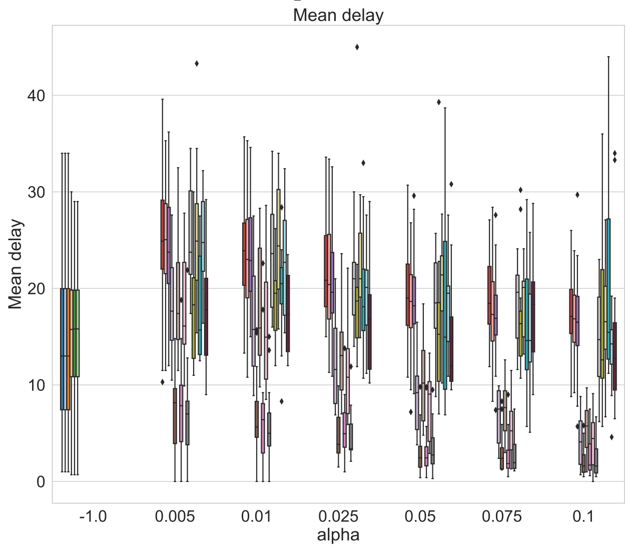

Fig. 6: Boxplot of mean delay for the different combinations of parameters for 50 randomly generated time series.

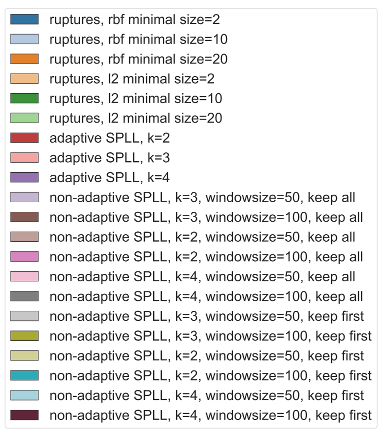
ruptures, rbf minimal size=2
ruptures, rbf minimal size=10
ruptures, rbf minimal size=20
ruptures, l2 minimal size=2
ruptures, l2 minimal size=10
ruptures, l2 minimal size=20
adaptive SPLL, k=2
adaptive SPLL, k=3
adaptive SPLL, k=4
non-adaptive SPLL, k=3, windowsize=50, keep all
non-adaptive SPLL, k=3, windowsize=100, keep all
non-adaptive SPLL, k=2, windowsize=50, keep all
non-adaptive SPLL, k=2, windowsize=100, keep all
non-adaptive SPLL, k=4, windowsize=50, keep all
non-adaptive SPLL, k=4, windowsize=100, keep all
non-adaptive SPLL, k=3, windowsize=50, keep first
non-adaptive SPLL, k=3, windowsize=100, keep first
non-adaptive SPLL, k=2, windowsize=50, keep first
non-adaptive SPLL, k=2, windowsize=100, keep first
non-adaptive SPLL, k=4, windowsize=50, keep first
non-adaptive SPLL, k=4, windowsize=100, keep first

Fig. 7: Legend to the plots of F-score and mean delay calculation.

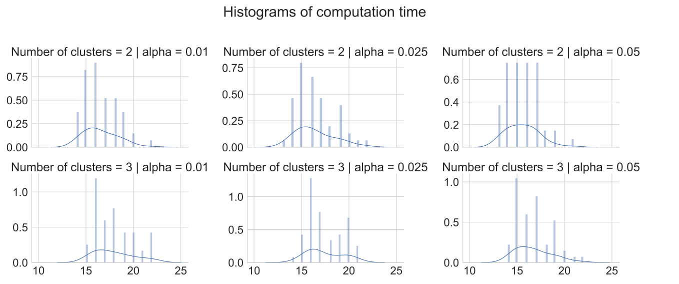

Fig. 8: Normalized histograms of computation time for the different combinations of parameters for the adaptive SPLL approach. For $n = 2000$, the mean computation time is about 17 seconds.

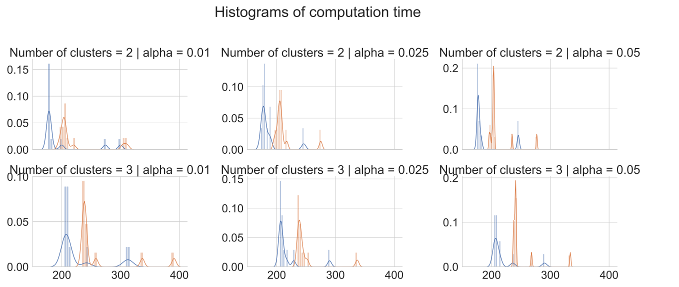

Fig. 9: Normalized histograms of computation time for the different combinations of parameters for the non-adaptive SPLL approach. For $n = 2000$, the mean computation time is about 200 seconds. The different colors indicate the window size (blue: 50, orange: 100)

$[t^*, t^{*+|W_2|}]$ a drift will be detected. Thus, when measuring the true positives as sketched in Figure 3, only the first time the drift is detected it is counted as true positive, whereas a drift detection thereafter but still less than $\epsilon$ away is treated as false positive. Hence, the non-adaptive SPLL approach counts almost $|W_2|$ false positives for each true positive. This of course strongly influences the F-score in a negative way. In the adaptive SPLL approach this is avoided by resetting the detection window to the point where the start of drift is estimated. In order to prevent these duplicated detections in the non-adaptive SPLL approach, we can count only those time points at which a new interval of detections begins. This will decrease the number of false negatives but at the same time, we might miss new drift periods when the detection window is smaller than the current drift period as these will then occur in the same continuous detection interval. In Figure 5 and Figure 6 this case is given by the additional description *keep first*, while in case when all detections are used in order to calculate the F-score and the mean delay, the description shows *keep all*.

The mean delays for the ruptures and adaptive SPLL approach are very similar, while the detection delay is significantly smaller for the non-adaptive SPLL approach. The opposite is true for the F-score: For ruptures and the non-adaptive SPLL, the F-score is significantly smaller (i.e., worse) than for the adaptive SPLL approach. Especially for the non-adaptive SPLL approach this low F-score of less than 0.1, in both cases, i.e., when keeping only the first or all detection times, is surprising.

### D. Evaluation discussion

The adaptive SPLL approach shows the best results concerning the F-score. Even though the mean delay is slightly longer than for the other approaches, it still delivers reasonable results. Further, we investigate the computation time for the adaptive and non-adaptive SPLL approaches. Especially in industrial environments, the computation time is crucial. Therefore, we collect the computation time for applying the adaptive SPLL approach for each of the experiments. The histograms of the resulting computation times for the experiments with different parameter combinations are given in Figure 8 and Figure 9. We see that for all combinations, the computation time is equal to or less than 25 $s$ for $n = 2000$ data points. That means that the mean computation time for each data point equals 12 $ms$ and therewith the algorithm is fast enough for most industrial applications.

Comparing this to the computation time of the non-adaptive SPLL approach (Figure 9), the significant improvement in computation time in the adaptive SPLL approach becomes obvious. The mean computation time for $n = 2000$ in the non-adaptive SPLL approach as described in [3] equals roughly 3 minutes and therewith 90 $ms$ per new data point. Even though this is still applicable to most industrial processes, it is 9 times slower than the adaptive SPLL approach, even though the adaptive SPLL approach performs more window comparisons due to finding the point at which a new drift period begins. This decrease in computation time is mainly linked to the reuse of $k$-Means centroids. Note that the computation time increases with window size used. In Figure 9 the blue bars correspond to a window size of 50, while the orange bars correspond to a window size of 100.

## V. Evalution on industrial data

Encouraged by the good results found on the artificial data set in the former section, we additionally evaluated the adapative SPLL method against an industrial data set. For this reason, we will use the run-to-failure data from the NASA Open Data Portal [16]. In the data set, engine degradation simulation was carried out using C-MAPSS tool. Four different sets were simulated under different combinations of operational conditions and fault modes. Several sensor channels were recorded to characterize fault evolution.

### A. NASA data set description

When taking all sensor channels into account, the time series has 25 features. Those features are all very different, from very small values with a mean in the range of $10^{-6}$ to mean values of up to $10^4$. Some of the features are constant, while others show strong variation. Most importantly, not all features show drift and some of the features are correlated. We will not perform any preprocessing of the data but apply

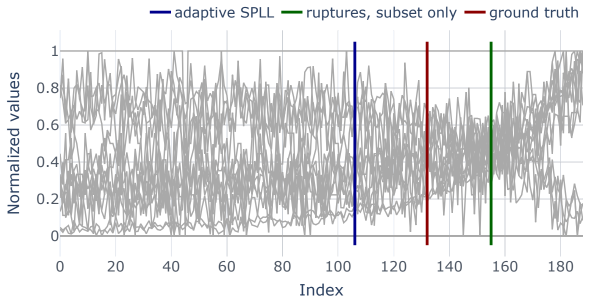

Fig. 10: The gray lines show the data from the NASA data set. For the visualisation, the data is scaled to the range from 0 to 1. The blue vertical line (most left) gives the drift detection by the adaptive SPLL methodology, the red line (middle) gives the ground truth while the most right, green line is the result from the Pelt algorithm.

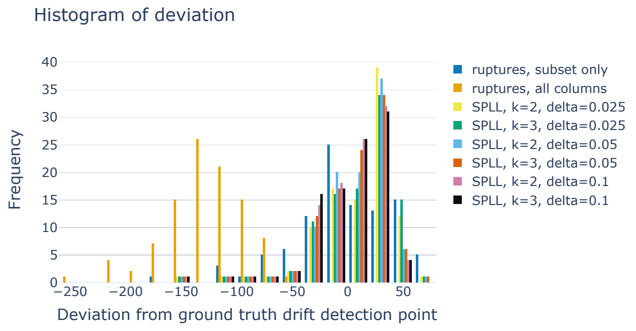

Fig. 11: Histogram of deviations from the ground truth label for the adaptive SPLL methodology and the Pelt algorithm.

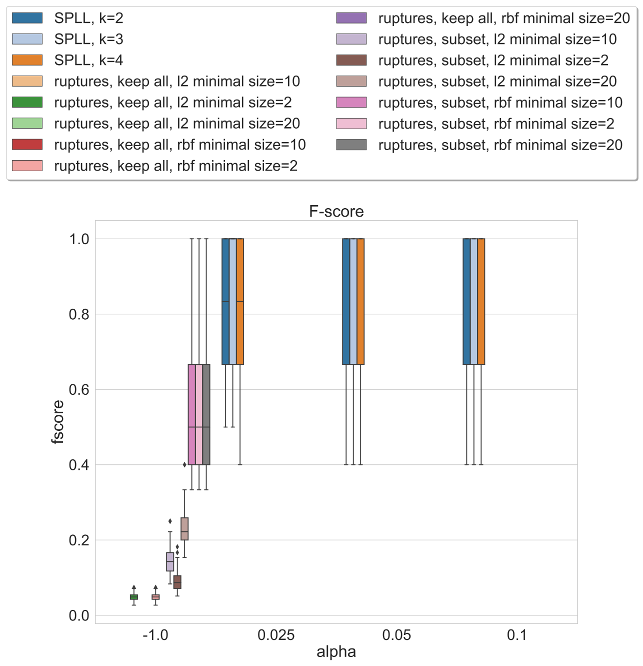

Fig. 12: F-score calculated for the adaptive SPLL methodology and the Pelt algorithm.

the adaptive SPLL method on it as it is. In the data set, in total 100 engines are simulated.

## B. NASA data set evaluation

In contrast to the artificial data set, we do not know when drift periods start in this data set. Therefore, for benchmarking purposes we implemented a simple offline methodology for drift detection. We calculate the mean value over the first 30 datapoints, for which we assume that no drift occurs. For the remaining values, we calculate a rolling mean over a window of size 5. If this mean deviates more than 15% from the initial mean, we state this point as the beginning of the drift period.

For one of the engines, the data is shown in Figure 10 as an example. Note that the data is scaled for reasons of visibility. The ground truth label is marked by the red line (middle), while the drift period starting point detected by the adaptive SPLL method is given in blue (most left) and by the Pelt algorithm when using a subset of the data (features 11,12 and 13) in green (most right). When purely looking at the data, the adaptive SPLL method gives the best results.

For evaluation, we first calculate the distance between the ground truth label and the first drift start period detected by the adaptive SPLL method and the Pelt algorithm for each engine. Almost all of the results derived by the adaptive SPLL methodology (compare Figure 11) for all parameter settings are in the range from -50 to 50 time steps apart from the ground truth label. From Figure 10 we already saw, that the adaptive SPLL method can detect drift earlier than the ground truth benchmark such that negative deviations in a reasonable range are not bad per se. In case of very large negative deviations, as for example most of the results from the Pelt algorithm when taking all features into account (orange bars), the methodology detects a drift too early. As most of these time series are only between 150 and 300 time steps long, the Pelt algorithm from the ruptures package detects drift very often right from the start and then many times or not at all. Therefore, the F-score is overall rather small for the Pelt algorithm, as either the number of true positives is zero - when no drift period was detected - or the number of false positives is high - when many drift periods were detected.

Similiarly as before, we can also compare the F-score values for the adaptive SPLL method and the Pelt algorithm (see Figure 12). We see that the adaptive SPLL method detects drift start periods very reliably with a F-scores between 0.8 and 1 independently of the parameters chosen.

## VI. Discussion & Conclusion

We have introduced an adaptive SPLL approach for online drift detection, extending the SPLL approach with fixed window size as introduced by Kuncheva [3]. We have shown that an adaptive window size and the inclusion of statistical measures like the evolution of the Mahalanobis distance can add significant improvements to the approach, including 1) a strong reduction in computation time, 2) a strong increase in F-score compared to non-adaptive approach and ruptures, as well as 3) mean detection delays that are similar to the offline ruptures method.

Even though the mean detection delay in the adaptive SPLL approach is longer than for the non-adaptive SPLL approach, the times are still reasonably small for most industrial drift detection use cases. Furthermore, we see that the SPLL approach is robust against parameter choice. For the artificial dataset the mean delay in detection is quasi-independent of the quantile $\alpha$ and the number of centroids $k$, corresponding to the number of components in the Gaussian mixture. The F-score is slightly increasing with increasing $\alpha$.

For industrial use cases, we assume that non-drifting data will be available such that the parameters for the algorithm can be optimized for each use case individually. In future work, we will perform additional validation tests on different datasets in order to analyze possible strengths and weaknesses of the adaptive SPLL approach in more detail. In first results from similar experiments with (strongly) correlated time series including periodical changes in the data, similarly good results for the adaptive approach are observed. From the work with the NASA engine data set, we can already see that the results are very promising.

In future work, also the application for different industrial use cases and corresponding drift warning scenarios will be studied. We see that the non-adaptive SPLL approach continuously detects drift periods when they occur, while the adaptive SPLL approach detects drift periods only once, right when they start and do not change anymore. Both cases can be relevant for different scenarios.

## Acknowledgments

This research was funded in part by the iDev40 project and in part by the Flemish Government (AI Research Program). The iDev40 project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 783163. The JU receives support from the European Union's Horizon 2020 research and innovation programme. It is co-funded by the consortium members, grants from Austria, Germany, Belgium, Italy, Spain and Romania. The information and results set out in this publication are those of the authors and do not necessarily reflect the opinion of the ECSEL Joint Undertaking. The research was carried out on behalf of Yazzoom, supported by Flanders Innovation & Entrepreneurship (VLAIO).

## References

[1] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*, pp. 443–448, SIAM, 2007.

[2] R. C. Cavalcante, L. L. Minku, and A. L. Oliveira, "Fedd: Feature extraction for explicit concept drift detection in time series," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 740–747, IEEE, 2016.

[3] L. Kuncheva, "Change Detection in Streaming Multivariate Data Using Likelihood Detectors," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, pp. 1 – 1, 2011.

[4] A. Ostovar, A. Maaradji, M. La Rosa, A. H. ter Hofstede, and B. F. van Dongen, "Detecting drift from event streams of unpredictable business processes," in *International Conference on Conceptual Modeling*, pp. 330–346, Springer, 2016.

[5] R. Pears, S. Sakthithasan, and Y. S. Koh, "Detecting concept change in dynamic data streams," *Machine Learning*, vol. 97, no. 3, pp. 259–293, 2014. Publisher: Springer.

[6] D. M. dos Reis, P. Flach, S. Matwin, and G. Batista, "Fast unsupervised online drift detection using incremental kolmogorov-smirnov test," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1545–1554, 2016.

[7] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Fourth international workshop on knowledge discovery from data streams*, vol. 6, pp. 77–86, 2006.

[8] I. Frías-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, 2014.

[9] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *International conference on discovery science*, pp. 264–269, Springer, 2007.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and others, "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011. Publisher: JMLR. org.

[11] M. J. Wenzel, A. Mensah-Brown, F. Josse, and E. E. Yaz, "Online drift compensation for chemical sensors using estimation theory," *IEEE Sensors Journal*, vol. 11, no. 1, pp. 225–232, 2010.

[12] M. Bhaduri, J. Zhan, C. Chiu, and F. Zhan, "A novel online and non-parametric approach for drift detection in big data," *IEEE Access*, vol. 5, pp. 15883–15892, 2017.

[13] J. MacQueen and others, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967. Issue: 14.

[14] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, 2020. Publisher: Elsevier.

[15] P. C. Mahalanobis, "On the generalized distance in statistics," National Institute of Science of India, 1936.

[16] A. Saxena and K. Goebel, "Turbofan engine degradation simulation data set." https://ti.arc.nasa.gov/c/13/.