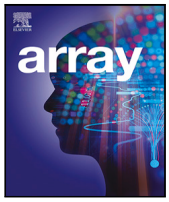
Array 14 (2022) 100151

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array



A comparison of online methods for change point detection in ion-mobility spectrometry data☆

Anton Kondratev [a,*], Katri Salminen [b], Jussi Rantala [a], Timo Salpavaara [a], Jarmo Verho [a], Veikko Surakka [a], Jukka Lekkala [a], Antti Vehkaoja [a,1], Philipp Müller [a,1]

[a] Tampere University, Korkeakoulunkatu 7, Tampere, 33720, Finland
[b] Tampere University of Applied Sciences, Kuntokatu 3, Tampere, 33520, Finland

ARTICLE INFO

Keywords:
Algorithms
Change detection
Ion mobility spectrometry

ABSTRACT

When on-site classification of volatile organic compounds (VOCs) is required, a portable ion mobility spectrometer (IMS) is a suitable choice. However, the IMS readings often show transient phases before they stabilize. Even so the importance of transient phase and features extracted from it has been highlighted in the literature, it has not, to our knowledge, been used for IMS-based classification so far. This paper analyzes whether change point detection algorithms with low computational complexity can separate transient and stable phases in IMS readings. The algorithms were tested on IMS data from different types of mushrooms. All algorithms successfully detected switches from transient to stable phase. The most accurate results were provided by the previously proposed multivariate max-CUSUM algorithm and the matrix form CUSUM algorithm, which is developed in this paper.

1. Introduction

Change points are abrupt variations in time series data. Detection of these points is useful in modeling and predicting time series [1]. Change point detection algorithms are designed to find a time point where a process evolving in time has experienced a change. This time point indicates a change in a process generating the data points. Change point detection is widely used in quality control [2], navigation system monitoring [3], seismic data processing [4], medicine, etc. [5].

Different change point detection algorithms have been proposed in the literature [5–8]. Online algorithms are run in real-time while time series data are being measured. Offline algorithms are supposed to run after the whole data set has been collected. Online algorithms can be applied also offline after data sets have been collected.

Aminikhanghahi and Cook published a survey on algorithms for change point detection in time series data. The survey described application areas of change point detection algorithms, different supervised and unsupervised approaches, and accuracy metrics. For more details the reader is referred to [1].

Recent studies on online change point detection indicate that the likelihood and probabilistic approaches are the most attractive methods [9–11]. For example, in [10] the Bayesian online change point algorithm was adapted for detecting a behavioral change in daily water consumption time series. The daily consumption profiles were clustered for extracting main behavior patterns and feeding them into the general likelihood framework for sequential analysis. The proposed algorithm also accounts for variables that can influence transitions between states in time series. Another example is applying the Bayesian Change Point Detection (BOCPD) algorithm for assessing the impact of cracks on the structural safety of concrete dams in time [12]. The results of this work showed that BOCPD can successfully detect real-time crack behavior changes.

Another approach for change point detection is subspace identification. This type of algorithm is base on the idea that "subspaces spanned by subsequences of time series data and the columns of the extended observability matrix are approximately equivalent" [13]. Change point are detected by estimating a state-space model behind time series. The authors demonstrated that their method is highly accurate.

In the context of the present work, online change detection algorithms are needed for supporting classification algorithms. The IMS readings by an electronic nose (eNose) often display transient and stable phases [14]. A common approach with IMS data is to wait for the switch from transient phase to stable phase and to use only

☆ This work was supported by Academy of Finland under grants: 323498, 295432, 295433, 323529, 323530 and 295434. The authors thank Dr. Simo Ali-Löytty for the discussion on the Matrix Form CUSUM algorithm.

* Corresponding author.
E-mail address: anton.kondratev@tuni.fi (A. Kondratev).

[1] A. Vehkaoja and P. Müller contributed equally to this article.

https://doi.org/10.1016/j.array.2022.100151
Received 5 November 2021; Received in revised form 8 March 2022; Accepted 1 April 2022
Available online 16 April 2022

data from the stable phase for scent classification. However, this slows down the classification as one has to wait for the transient phase to end, which typically lasts from few seconds up to 1–2 min. However, information contained in the transient phase can be, according to some researchers, more valuable than information contained in the stable phase [15]. Features from the transient phase (e.g. derivatives, length of the transient phase, etc.) may help with classification. In order to use the full potential of information in the transient phase, it is essential to accurately and objectively determine the switch from transient to stable phase. Therefore, this paper studies online change detection algorithms to distinguish the transient phase from the stable phase in IMS readings.

In total five algorithms are discussed: Shewhart Control Charts, Cumulative Sum (CUSUM) and two Cumulative Sum (CUSUM) variants including one variant that, to our knowledge, has not been proposed before, and Bayesian Online Change Point algorithm. In this work we considered simple algorithms, with the Bayesian Online Change Point Detection algorithm being the most complicated. There are other modifications of the CUSUM algorithm, such as tabular CUSUM and CUSUM V-mask. The latter one uses a set of hyperparameters for which no definite rule exist on how to choose them. For that and other reasons using this algorithm is not recommended [8, p. 416]. The tabular CUSUM would require, for the problem at hand, to run a large number of algorithms in parallel and is therefore omitted from the consideration. There exists more sophisticated change point detection algorithms based on, for example, neural networks [16], but they are out of scope for this paper because only small number of data sets were available. Online algorithms described in this work have a short delay of 1 s because the data is preprocessed before feeding them into the algorithms. In this work, we are interested only in methods with low computational demand that enable online change point detection on hand-held devices with limited computational capacity.

For each algorithm the paper provides a brief explanation and discussion on its suitability for detecting switches from transient to stable phases. For suitable algorithms pseudo code is presented and they are then used for detecting switches from transient to stable phases in IMS readings collected from mushrooms, and their performances are compared. The IMS data as well as ready-to-use Python code for the tested algorithms is freely available at [17].

2. Electronic noses based on ion mobility

An eNose is a set of gas sensors that measure the ambient gas atmosphere, for example, scents, flavors, or non-odorous chemicals. eNoses are based on the general principle where changes in the gaseous atmosphere alter the sensor properties in a characteristic way, depending on the eNose technology and chemical sensor or sensor array used [18]. The sensors often consist of metal oxides, conducting polymers composites and intrinsically conducted polymers [18].

Ion Mobility Spectrometry (IMS) is a common eNose technique where ionized molecules are separated using an electric field and buffer gas. The molecules are headed into a drift-tube where they are ionized. Ions moving through the drift tube under impact of the electrical field are colliding with the molecules of the buffer gas, which causes them to slow down (i.e., change the mobility of the ions). The ChemPro100i eNose consists of several sensing areas (e.g., several metal-oxide sensors) and the velocity of the ionized particles determines at which sensing area they will be measured. The velocity correlates with the mobility of the specific ion. Several types of IMS devices exist and there are differences in the technical details of the devices including the number of the sensing areas and sensors used as well as, for example, electric field strength and drift gas temperature that do affect the data. A detailed discussion of IMS types can be found, for example, in [19].

In this work we used the ChemPro100i, which is an IMS-based eNose developed and patented by Environics Ltd. ChemPro100i was developed for detecting chemical substances such as warfare agents and hazardous gases in ambient air. It uses the so-called "open-loop aspiration" principle and uses Americium-241 source for ionization. IMS sensors have several favorable properties: they are light and small, which allows their use in portable eNoses. Their high sensitivity, low power consumption and low operating costs [20] make them a suitable choice for on-site measurements. One significant advantage of ChemPro 100i is that air can be employed as a carrier gas. Other mobility spectrometers traditionally use specific carrier gases [21]. Fig. 1 illustrates the workflow of the ChemPro100i.

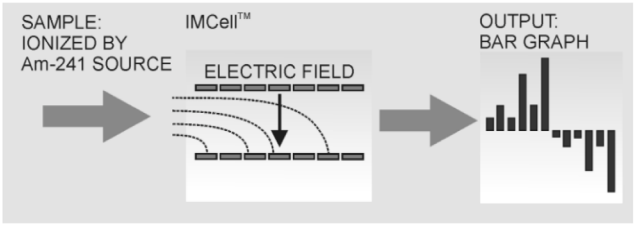


Fig. 1. Workflow of the ChemPro100i [22].

Among other information like temperature and humidity the ChemPro100i provides electric current readings for 16 channels. However, the eighth and the sixteenth channels are control channels, which should be zero at any time. Of the remaining 14 channels, seven show positive and seven negative currents. The readings usually have transient and stable phases when a channel is reacting to a chemical, e.g., scent. The transient phase is defined as either uptrending or downtrending part of a particular sensor channel. The red dashed line in Fig. 2(a) separates these two phases. The transient phase is on the left of the red line and the stable phase is on the right of the red line. The measurements always contain noise. The magnitude of noise depends on from where the scent source was sampled. If a scent is measured from a controlled headspace, e.g., sealed flask the readings are in general rather stable (see Fig. 2(a) for a typical example) as the composition of the air sucked into the eNose is stable. Scents measured in a less controlled setting, e.g., from a plate in general yield more instable channel responses or cause no valuable responses at all (see Fig. 2(b) for an example). The reason for significant noise when measuring from plate is the presence of other scents in the ambient air and the more significant fluctuations in the composition of the air sucked into the ChemPro100i for analysis. Another problem is that the ChemPro100i does not react to certain scents. For example, the reading in Fig. 2(b) shows that the ChemPro100i possibly did not react (significantly) to the black chanterelle scent for the displayed channel. The change in readings is more than 10 pA if measured from flask (Fig. 2(a)), whereas readings from plate fluctuated between 43.5 and 46.5 pA, indicating that the scent concentration was too weak for being (clearly) visible in the IMS reading. Note that Figs. 2(a) and 2(b) are illustrative examples and represent different channels.

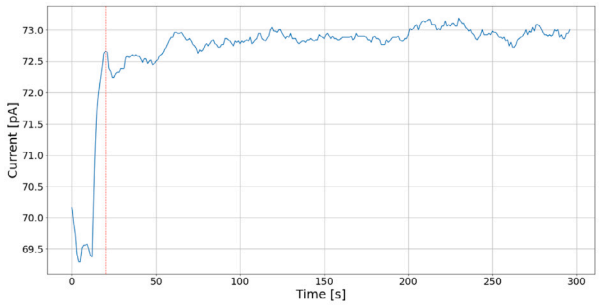
3. Change detection methods

3.1. Shewhart control charts

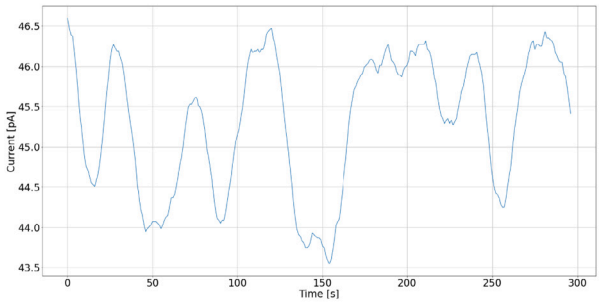
The Shewhart Control Charts algorithm is the simplest log-likelihood ratio based algorithm. The main idea of this algorithm is to capture an instance where a distribution changes its parameters. The log-likelihood ratio is defined as

$$L(y) = ln\frac{p_{\theta_1}(y)}{p_{\theta_0}(y)}, \tag{1}$$

where θ_0 and θ_1 represent a set of parameters of a distribution before and after a change point. The natural logarithm function causes the $L(y)$ function to be negative when the likelihood of the distribution with the parameters θ_0 is greater and positive when the likelihood of the parameters θ_1 is greater. This algorithm requires θ_0 and θ_1 to be known.



(a) Black chanterelle measured from sealed flask.



(b) Black chanterelle measured from table.

Fig. 2. ChemPro100i example readings from mushrooms.

Based on our observation of multiple data sets from various scent sources we assumed the data in the *stable phase* to be approximately normally distributed with zero mean after differencing (i.e. discrete difference operation). Differencing means that instead of measured IMS values difference between two consecutive observations was used. This operation is often used in time-series analysis for making a time-series stationary [23]. The distribution in the *transition phase* is assumed to be normal with the parameters equal to the sample mean and sample standard deviation. The log-likelihood ratio for following the change in mean is defined as

$$L_1^N = \frac{b}{\sigma}\sum_{i=1}^{N}(y_i - \mu_0 - \frac{v}{2}) \tag{2}$$

where

$$v = \mu_1 - \mu_0$$
$$b = \frac{v}{\sigma}$$

and N is the size of the sliding window. A sliding window is a structure that rolls over a time series. It always contains the current and $N-1$ previous data points. This technique enables sampling and calculating statistical information for time series online, and can be used for smoothing time series. In [5] using the cumulative log-likelihood ratio

$$L = \sum_{i=1} L_i^N \tag{3}$$

for making a decision (decision function) is proposed. A change point is detected if

$$L \geq h \tag{4}$$

where h is conveniently chosen threshold (we used $h = 0$). The typical decision function is shown in Fig. 3. In this work we did not use the Cumulative log-likelihood ratio (LLR). Instead, we used the sequence of log-likelihood ratio (Fig. 3 left plot), as it is more convenient for detecting where the log-likelihood ratio crosses zero. The workflow of the Shewhart Control Charts algorithm is shown in Algorithm 1. When the loop (line 7) stops iterating the counter i will contain the

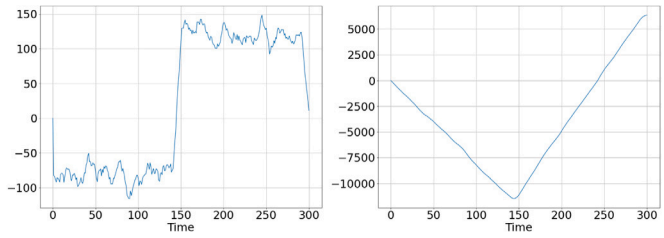


Fig. 3. LLR function (left) and Cumulative LLR function (right).

Algorithm 1: Shewhart Charts

Input: s - length of sliding window, x_i - initial sample of size s
Result: time step i at which distribution has changed

1 $\bar{\mu} = \frac{1}{s}\sum_{i=0}^{N} x_i$
2 $\bar{\sigma} = \frac{\sum(x_i - \bar{\mu})}{\sqrt{N-1}}$
3 $\mu_0 = \bar{\mu}$, $\sigma_0 = \bar{\sigma}$;
4 $L_{prev} = 0$; `# Preserves the previous LLR-value`
5 $dL = 0$; `# Preserves difference` $L_i - L_{prev}$
6 i = 0; `# counter`
7 **while** $dL \leq 0$ **do**
8 $x_i = [y_i : y_{i+s}]$; `# extract sample of size s`
9 $v = 0 - \mu_0$; `#` μ_1 `is expected to be 0`
10 $b = \frac{v}{\sigma_0}$;
11 $L_i = \frac{b}{\sigma}\sum_{i=1}^{N}(x_i - \mu_0 - \frac{v}{2})$;
12 $dL = L_i - L_{prev}$;
13 $L_{prev} = L_i$;
14 $i = i + 1$;
15 **end**

detected time stamp of a change. The disadvantage of this algorithm is that it is difficult to choose threshold h. Setting h to zero causes the algorithm to indicate a change point too early. Instead of thresholding the value of the LLR-function, we add the size of the sliding window to the change point detected by the algorithm. This is based on the logic that when the algorithm has detected a change then it probably happened at the end or in the middle of the sliding window. Adding half of the sliding window was tested and resulted in too early detection. For scent classification, we trust measurements from stable phase more than measurements from transient phase. Therefore, it is less critical for scent classification if an algorithm places a change point slightly after rather than before the true change point. The sliding window is defined starting from the first point in a sequence.

This algorithm needs to be run for each channel separately, which increases memory consumption and calculation times. The final decision is when majority of the channels yield detected change point by using the average of all detected change points. The Shewhart Charts can be implemented using matrices that enable calculation of all the channels simultaneously.

3.2. CUSUM metods

CUSUM

Cumulative Sum (CUSUM) was proposed by Page in [24]. The CUSUM method is a popular algorithm in statistical quality control and in change point detection. Many extensions based on the CUSUM have been proposed [6,8,25].

The core of the algorithm is the same as in the Shewhart Charts — log-likelihood ratio test, but decision whether a change point is found differs. The LLR-value is calculated by (2). The decision about a change point is made by comparing cumulative LLR-values with the minimum

LLR-value found in the previous iterations. The decision function is defined as

$$g_k = L_k - m_k \geq h, \tag{5}$$

where L_k is the cumulative LLR, m_k is the minimum LLR-value found in the previous iterations and h is a conveniently chosen threshold. (5) can be rewritten as

$$g_k = L_k \geq h + m_k. \tag{6}$$

In the algorithm implemented in this work zero threshold was used. Therefore, (6) simplifies to

$$g_k = L_k \geq m_k. \tag{7}$$

As in the description of the Shewhart Charts, we set the threshold to zero and add the size of the sliding window to the found time stamp. Figs. 4(a) and 4(b) show the typical behavior of CUSUM's decision functions. The first plot (Fig. 4(a)) shows the decision function for artificially generated data, where distribution has changed from $Normal(0, 1)$ to $Normal(5, 1)$ at 150 s. The second plot (Fig. 4(b)) shows the decision function for one channel, picked as an illustrative example, in the data explained in Section 4. The red dashed line depicts the time stamp where the algorithm has found a change point. The last plot (Fig. 4(c)) shows IMS readings on the channel and change point found by the algorithm. Algorithm 2 shows pseudo-code for the CUSUM approach. As can be seen on line 13 LLR-value is stored into the array A and the difference of the current LLR-value and the minimum value in the array A is compared to zero (line 14). If the difference is greater than zero then a change point is found.

Algorithm 2: CUSUM

Input: s - sliding window size, x_i - sample of size s
Result: time step i at which distribution has changed

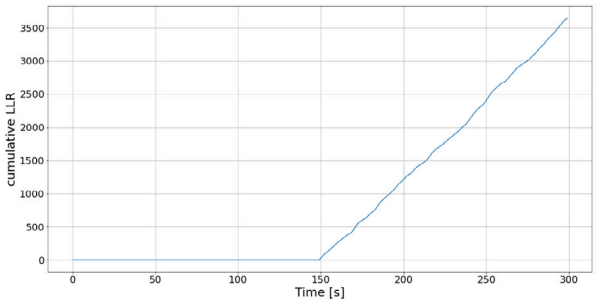
1 $L = 0$ # initialize cumulative sum variable
2 $A = []$ # initialize array for storing cumulative sums
3 $\bar{\mu}_0 = \frac{1}{s}\sum_{i=0}^{s} x_i$ # sample mean of the first s samples
4 $\bar{\sigma}_0 = \frac{\sum(x_i - \bar{\mu})}{\sqrt{s-1}}$ # sample standard deviation of the first S samples
5 *detected* = *False*
6 i = 0 # counter
7 **while** *detected* == *False* **do**
8 i = i + 1;
9 sample $\leftarrow [y_i : y_{i+s}]$; # Get next sample of length s
10 $v = 0 - \bar{\mu}_0$;
11 $b = \frac{v}{\bar{\sigma}_0}$;
12 $L = L + (1/\bar{\sigma}_0)(\sum sample - s \cdot \bar{\mu}_0 - \frac{s \cdot v}{2})$;
13 $A[i] \leftarrow L$;
14 **if** $(L - min(A)) > 0$ **then**
15 *detected* $\leftarrow$ *True*;
16 *change_point* $\leftarrow i$;
17 **end**
18 **end**

Multivariate Max-CUSUM Chart

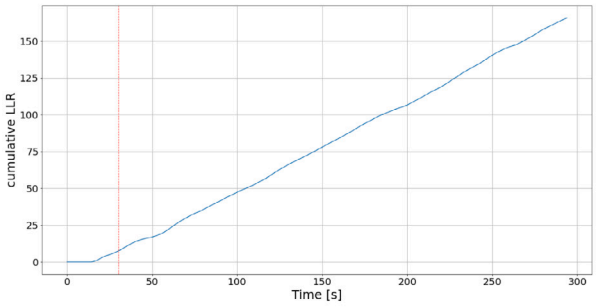
This algorithm was proposed in [6]. The paper demonstrates how testing multivariate normal data with CUSUM will be reduced to univariate classic CUSUM testing.

The classic scheme for CUSUM works here as well:

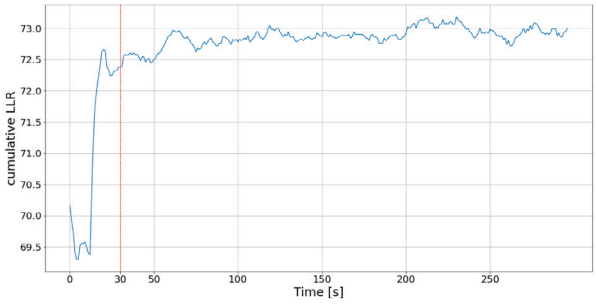
$$L_i = max(L_{i-1} + \mathbf{a}'(\mathbf{x_n} - \boldsymbol{\mu_G}) - 0.5D, 0) > H \tag{8}$$



(a) Typical behavior of CUSUM. Artificial data.



(b) Typical behavior of CUSUM. Black Chanterelle, IMS channel 4.



(c) Found change point. Black Chanterelle, IMS channel 4.

Fig. 4. CUSUM: typical behaviors and found change point.

where

$$a' = \frac{(\boldsymbol{\mu_B} - \boldsymbol{\mu_G})^T \boldsymbol{\Sigma}^{-1}}{[(\boldsymbol{\mu_B} - \boldsymbol{\mu_G})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu_B} - \boldsymbol{\mu_G})]^{\frac{1}{2}}} \tag{9}$$

and

$$D = \sqrt{(\boldsymbol{\mu_B} - \boldsymbol{\mu_G})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu_B} - \boldsymbol{\mu_G})} \tag{10}$$

The Multivariate Max-CUSUM algorithm calculates all channels simultaneously. As mentioned previously the data points are run through the differencing operation before the algorithm.

Pseudo-code for the Multivariate Max-CUSUM algorithm is shown in Algorithm 3.

Matrix form CUSUM

We propose the matrix form CUSUM as a simple multivariate extension of the classic CUSUM algorithm. All calculations are the same as in the classic CUSUM, but performed in matrix form. This approach shortens computation time significantly and has approximately the same accuracy as the one-dimensional CUSUM. The algorithm can be implemented in two ways, providing either a uniform change point that is the average of detected change points over all channels or separate change points for each channel. Averaging detected points over all channels can result in less precise change points but increases robustness to channels that did not react to a scent and do not contain

Algorithm 3: Multivariate Max-CUSUM Chart

Input: s - sliding window size, X_i - sample of size $14 \times s$
Result: time step i at which distribution has changed

$\mu_0 = \bar{\mu}$ # sample mean along each row
$\mu_1 = 0$ # assumes Normal distributed data after change point
$\Sigma = \bar{\Sigma}$ # sample covariance matrix
$\Sigma_{i,j} = \Sigma_{i,j} + 1e-10$, where i = j # adding 1e-10 to main diagonal avoids singularity
a = $\frac{(\boldsymbol{\mu_1}-\boldsymbol{\mu_0})^T \Sigma^{-1}}{\sqrt{(\boldsymbol{\mu_1}-\boldsymbol{\mu_0})^T \Sigma^{-1}(\boldsymbol{\mu_1}-\boldsymbol{\mu_0})}}$
$L_i = 0$ # initial value for cumulative sum
detected = False
point = 0
i = 0 # counter
while *detected == False* **do**
  $\boldsymbol{X_i}$ = *data points of size* $14 \times s$
  $\boldsymbol{\mu_1}$ = mean($\boldsymbol{X_i}$)
  $D = \sqrt{(\boldsymbol{\mu_1}-\boldsymbol{\mu_0})^T \Sigma^{-1}(\boldsymbol{\mu_1}-\boldsymbol{\mu_0})}$
  $L_i = max(0, L_{i-1} + a(\boldsymbol{\mu_1}-\boldsymbol{\mu_0}) - 0.5D)$
  if $L_i > 0$ **then**
    point = i + s
    detected = True
  end
  i = i + 1
end

any change points. This approach only fails if the majority of channels fails or when the algorithm detects change points on multiple channels that are far from the real change points. Analysis of the data set collected for this paper and the data set used in [14] showed that both situations are improbable. Therefore, in this paper the Matrix form CUSUM algorithm using averages of detected change points is used.

Let

$$\mathbf{X}_i = \begin{bmatrix} x_{1,1} & \dots & x_{1,s} \\ \vdots & \ddots & \vdots \\ x_{14,1} & \dots & x_{14,s} \end{bmatrix} \in \mathbb{R}^{14\times s} \tag{11}$$

be the matrix containing readings from all channels, where each row represents a channel and s is the size of a moving window. That is, $x_{i,j}$ is the jth reading of the ith channel. The vector of sample means for each channel is calculated as follow

$$\mathbf{X_i} \times \begin{bmatrix} \frac{1}{s} \\ \vdots \\ \frac{1}{s} \end{bmatrix}_{s\times 1} = \begin{bmatrix} \bar{\mu}_1 \\ \bar{\mu}_2 \\ \dots \\ \bar{\mu}_{14} \end{bmatrix} \tag{12}$$

The standard deviation for one dimension is calculated as

$$\bar{\sigma} = \sqrt{\frac{\sum (x_i - \mu)^2}{s-1}} \tag{13}$$

The one-dimensional standard deviation can be converted into the matrix form as follows:

$$\mathbf{F}_\mu = \begin{bmatrix} \mathbf{m} & \mathbf{m} & \dots & \mathbf{m} \end{bmatrix} \in \mathbb{R}^{14\times s}, \mathbf{1}_s = \begin{bmatrix} \frac{1}{s-1} \\ \vdots \\ \frac{1}{s-1} \end{bmatrix} \in \mathbb{R}^{s\times 1} \tag{14}$$

Then the fraction under the square root in (13) is calculated by

$$\begin{bmatrix} (x_{1,1}-\mu_1)^2 & \dots & (x_{1,s}-\mu_1)^2 \\ \vdots & \vdots & \vdots \\ (x_{14,1}-\mu_{14})^2 & \dots & (x_{14,s}-\mu_{14})^2 \end{bmatrix} \begin{bmatrix} \frac{1}{s-1} \\ \vdots \\ \frac{1}{s-1} \end{bmatrix}, \tag{15}$$

which can be rewritten in more compact form as

$$(\mathbf{X_i} - \mathbf{F}_\mu)^2 \times \mathbf{1}_s \tag{16}$$

Eventually, the n-dimensional standard deviation is

$$\mathbf{STD} = \sqrt{(\mathbf{X_i} - \mathbf{F}_\mu)^2 \times \mathbf{1}_s} \tag{17}$$

Notice that in (17) square and square root are calculated for each entry of the matrix.

The log-likelihood ratio (LLR) for CUSUM mean shift for one dimension [5] is calculated as

$$L_1^s = (\frac{b}{\sigma}) \sum_{i=1}^{s} (y_i - \mu_0 - \frac{v}{2}) = (\frac{b}{\sigma})(\sum_{i=1}^{s} y_i - s\mu_0 - \frac{sv}{2}) \tag{18}$$

The same LLR is used for the Matrix Form CUSUM where:

$$\sigma = \mathbf{STD}$$
$$\mathbf{v} = -1 \times \mathbf{m} \tag{19}$$
$$\mathbf{b} = \mathbf{v} \odot \frac{1}{\mathbf{STD}} \tag{20}$$
$$\sum_{i=1}^{s} y_i = \mathbf{X_i} \times \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T \tag{21}$$
$$s\mu_0 = s \times \mathbf{m} \tag{22}$$

Notice s is scalar and L_1^s is the LLR calculated for the first sample of size s. Symbol $\odot$ in (20) denotes element-wise multiplication.

The decision function is calculated as

$$g_k = average(\mathbf{L_k^s}) - m_k \geq h \tag{23}$$

where

$$m_k = \min_{1\leq j<k}(L_j) \tag{24}$$

and h is a conveniently chosen threshold. The right side of (24) represents the minimum value of all previously calculated LLR. Each entry of vector $\mathbf{L_k^s}$ is the LLR for one channel. That is, each entry of the vector $\mathbf{L_k^s}$ is the channel-specific minimum LLR-value. If the algorithm has detected that calculated LLR-value is less than the value in this vector, the corresponding entry of the vector will be replaced. The averaging operation of the vector in (23) means averaging of all entries in the vector. Sequential calculation of this log-likelihood ratio yields the same result as in the one-dimensional CUSUM, but for all channels simultaneously. Vector **I** in Algorithm 4 contains the minimum values over all previous LLR for each channel. The entries of vector **I** need to be updated at each iteration.

Bayesian online change point detection

Bayesian online change point detection (BOCPD) was proposed in [26]. The idea of BOCPD is to detect a change point in terms of so-called run lengths. The concept of this algorithm is shown in Fig. 5. Whenever a new measurement is available the algorithm calculates the probability that the corresponding run length grows by one. If the probability of change is greater than the probability of growth then the run length drops to zero and a change point is detected. Fig. 5(a) has three partitions. The partitions are separated by change points. The fourth point in the partition g_1 is the last before the change point occurs. Before this point, as can be seen in Fig. 5(b), the run length grows. The fifth point, which belongs to the partition g_2 originates from another distribution. This fact causes the run length to drop to zero. After arriving at a point the algorithm performs four steps:

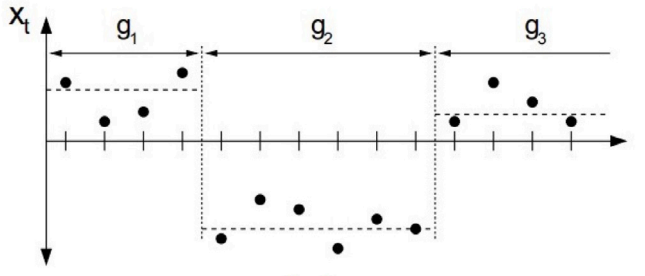
1. Calculate posterior predictive probability
2. Calculate probability of growth
3. Calculate probability of change point
4. Update statistics

For implementing this algorithm the matrix **R** was used (Algorithm 5). Matrix **R** is shown in Fig. 6. The first column of **R** represents

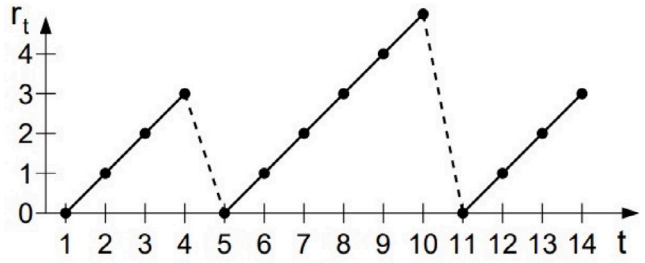
Algorithm 4: Matrix Form CUSUM

Input: s - sliding window size, X_i - sample of size $14 \times s$
Result: time step i at which distribution has changed

$\mathbf{L_i} \leftarrow \begin{bmatrix}0 & 0 & \dots & 0\end{bmatrix}^T$ # initialize vector of cumulative LLR
$\mathbf{M} = \mathbf{X_i} \times \mathbf{1}_s$ # vector of means
$\mathbf{F}_\mu \leftarrow \begin{bmatrix}\mathbf{M} & \mathbf{M} & \dots & \mathbf{M}\end{bmatrix}$ # 14xs matrix
$\mathbf{STD} = \sqrt{(\mathbf{X_i} - \mathbf{F}_\mu)^2 \times \mathbf{1}_s}$ # calculate vector of standard deviations
$\mathbf{v} = -1 \times \mathbf{M}$
$\mathbf{b} = \mathbf{v} \odot \frac{1}{\mathbf{STD}}$
$\mathbf{I} \leftarrow \begin{bmatrix}0 & 0 & \dots & 0\end{bmatrix}^T$ # 14-dimensional vector for storing minimum values of LLR
detected = False
point = 0
i = 0 # loop counter
while *detected == False* **do**
  $\mathbf{X_i} \leftarrow$ *new chunk of data of size* $14 \times s$
  $\mathbf{L_i} = \mathbf{L_i} + (\frac{\mathbf{b}}{\mathbf{STD}})(\mathbf{X_i} \times \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}^T - s \times \mathbf{M} - \frac{s}{2} \times \mathbf{v})$
  if $\mathbf{L_i} < \mathbf{I}$ **then**
    $\mathbf{I} \leftarrow \mathbf{L_i}$
  end
  if $average(\mathbf{L_i} - \mathbf{I}) > 0$ **then**
    point = i + s
    $detected \leftarrow True$
  end
  i = i + 1
end



(a) Partitioning data by run lengths.



(b) Run lengths

Fig. 5. Idea of separating data by change points [26].

probabilities of change at times $t_1, \dots, t_n$. The remaining columns represent probabilities of different run lengths at different times. It is not necessary to keep the full matrix in the memory. Instead, only the results of the last iteration must be stored.

In this work the t-distribution was used as underlying probability distribution (UPM) because the Normal distribution showed poor detection results. The t-distribution was used because μ and σ^2 of the data were unknown. As a result the conjugate prior for estimating both parameters is the Normal-Gamma distribution [27]. In this case, the posterior predictive distribution is the generalized t-distribution with $\upsilon = 2\alpha$ degrees of freedom, $\mu = \bar{\mu}$ sample mean and

$$\sigma^2 = \frac{\beta_n(\kappa_n + 1)}{\alpha_n \kappa_n}$$

variance. The Normal-Gamma distribution has parameters $\alpha, \beta, \mu, \kappa$, which must be initialized before the first iteration. The updating parameters are updated as follow:

$$\bar{\mu}_n = \frac{\kappa_0 \mu_0 + n\bar{x}}{\kappa_0 + n}, \tag{25}$$

$$\kappa_n = \kappa_0 + n, \tag{26}$$

$$\alpha_n = \alpha_0 + \frac{n}{2}, \tag{27}$$

$$\beta_n = \beta_0 + \frac{\kappa_i (x_i - \mu_i)^2}{2(\kappa_i + n)}. \tag{28}$$

The first step at time t is calculating the posterior predictive probability for any possible run length by

$$\pi_t^{(r)} = P(x_t | \alpha_t^{(r)}, \beta_t^{(r)}, \kappa_t^{(r)}, \mu_t^{(r)}), \tag{29}$$

where r represents the run length. This probability is calculated using the t-distribution with parameters υ, μ and σ^2. In the second step the probability of growth for each run length is calculated as

$$P(r_t = r_{t-1} + 1, \mathbf{x}_{1:t}) = P(r_{t-1}, \mathbf{x}_{1:t-1}) \pi_t^{(r)} (1 - H_{r_{t-1}}) \tag{30}$$

where H is a hazard rate. The hazard rate may represent prior knowledge on how often change points occur. More explanation about the hazard rate can be found in [28]. In the third step the probability of a change point is calculated, i.e. run length being zero, as

$$P(r_t = 0, \mathbf{x}_{1:t}) = \sum_{r_{t-1}} P(r_{t-1}, \mathbf{x}_{1:t-1}) \pi_t^{(r)} H_{r_{t-1}}. \tag{31}$$

The distribution of run lengths needs to be normalized by

$$P(r_t, \mathbf{x}_{1:t}) = \frac{P(r_t, \mathbf{x}_{1:t})}{\sum P(r_t, \mathbf{x}_{1:t})} \tag{32}$$

Lastly, the parameters of the distribution are updated with (25)–(28). In the implemented algorithm a matrix R was used for keeping probabilities of growth and change point. Rows of the matrix R represent time steps and columns represent joints of the trellis described in the original paper. The matrix R was used only for illustrative purposes. The implementation of this algorithm can be found in the supplementary material.

One obstacle encountered in the implementation process was the initialization of the parameters for the Normal-Gamma distribution. Several sources simply set the all initial parameters to 1 and the hazard rate to the prior belief on the frequency of change points, but no justification has been given in the literature. A quick study [25, p. 37] using the data sets [14] revealed that setting κ and α to one, μ to sample mean and β being equal to the hazard rate ensures that the algorithm always finds meaningful change points. Initializing all parameters to 1 results in failing to find any change point.

$$R = \begin{array}{c|ccccc} & r_0 & r_1 & r_2 & \dots & r_n \\ \hline t_1 & P(r_t = 0) & 0 & 0 & \dots & 0 \\ t_2 & P(r_t = 0) & P(r_t = 1) & 0 & \dots & 0 \\ t_3 & P(r_t = 0) & P(r_t = 1) & P(r_t = 2) & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & 0 \\ t_n & P(r_t = 0) & P(r_t = 1) & P(r_t = 2) & \dots & P(r_t = S) \end{array}$$

Fig. 6. Matrix **R**.

Algorithm 5: Bayesian Change Point with t-distribution as UPM

Result: time step i at which distribution has changed

$\boldsymbol{R} \leftarrow \boldsymbol{0}$ # initialize $n \times n$ matrix R with zeros
$R[0,0] = 1$ # set first value to one
$\boldsymbol{l} \leftarrow [1]$ # create a vector for storing information from previous step
$H = \frac{1}{100}$ # initialize hazard rate
$\alpha_0 = \kappa_0 = 1$ # initialize parameters of UPM distribution
$\mu_0 = \bar{\mu}$ # set mean to sample mean
$\beta_0 = H$ # set β to the Hazard rate
$\boldsymbol{a} = \boldsymbol{b} = \boldsymbol{m} = \boldsymbol{k} = []$ # initialize vectors for $\alpha, \beta, \mu, \kappa$ for saving all previous values
found = False
i = 1 # counter
while *found == False* **do**
  $x_i \leftarrow$ *next point*
  $\boldsymbol{\pi} = function$ *get_predictive_probabilities*$(x_i, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{m}, \boldsymbol{k})$
  $\boldsymbol{g} = \boldsymbol{l} \times \boldsymbol{\pi} \times (1 - H)$ # calculate growth probabilities
  $c = \sum(\boldsymbol{l} \times \boldsymbol{\pi} \times H)$ # calculate change point probabilities
  $\boldsymbol{ng} = [\boldsymbol{c}, \boldsymbol{g}]$ # concatenate into one vector change and growth probabilities
  R[i,:] = $\frac{\boldsymbol{ng}}{\sum \boldsymbol{ng}}$ # normalize and put into the R-matrix
  $\boldsymbol{\mu_n} = \frac{(\boldsymbol{k} \times \boldsymbol{m} + x_i)}{\boldsymbol{k}+1}$ # update mean-value
  $\boldsymbol{\kappa_n} = \boldsymbol{k} + 1$ # update kappa-value
  $\boldsymbol{\alpha_n} = \boldsymbol{a} + \frac{1}{2}$ # update alpha-value
  $\boldsymbol{\beta_n} = \boldsymbol{b} + \frac{\boldsymbol{k}(x_i - \boldsymbol{m})^2}{2(\boldsymbol{k}+1)}$ # update beta-value
  $\boldsymbol{a} = [\alpha_0, \boldsymbol{\alpha_n}]$ # concatenate vector alpha
  $\boldsymbol{b} = [\beta_0, \boldsymbol{\beta_n}]$
  $\boldsymbol{m} = [\mu_0, \boldsymbol{\mu_n}]$
  $\boldsymbol{k} = [\kappa_0, \boldsymbol{\kappa_n}]$
  $\boldsymbol{l} = [C, G]$ # concatenate for next iteration
end
Function *get_predictive_probabilities*$(x_i, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{m}, \boldsymbol{k})$:
  df = $2 \times \boldsymbol{a}$
  loc = $\boldsymbol{m}$
  scale = $\sqrt{\frac{\boldsymbol{b}(\boldsymbol{k}+1)}{\boldsymbol{a} \times \boldsymbol{k}}}$
  return probability t(x,df, loc, scale)
return

4. Data

Measurements from black chanterelle (Craterellus cornucopioides), yellowfoot (Craterellus tubaeformis), and a mix of both species were collected. The mushrooms were air dried. No additives (e.g., salt or water) were used in the process.

All scent sources were measured with a ChemPro100i both from an open plate and a sealed flask at 1 Hz. For each scent source 5 sets of 5 min were measured, meaning that the database contained a total of 30 data sets. Between measurements of two sets a break of 3 min was taken. The breaks were needed in order to flush the drift tube with ambient air until the IMS readings returned to the baseline. The ground truth points were selected manually by visual inspection of the time series plots.

The 30 data sets were then classified as either good, bad or ambiguous. A data set was characterized as good if it contained IMS readings with clearly visible transient and stable phases. A data set was bad if it did not contain any clearly visible phase changes. A ambiguous data set contained both channels with clearly visible phase changes and channels without clearly visible phases. The data is available at [17].

Table 1
Average running times [ms].

Direct times [ms]					
Size	Shewhart	CUSUM	MaxCUSUM	MFCUSUM	Bayes
5	6.06	9.92	1.13	0.18	341.43
10	4.71	9.24	0.65	0.12	337.95
15	4.21	8.94	0.64	0.11	348.16
Average and relative times [ms]					
Avg	4.99	9.36	0.81	0.14	342.52
Rel	36.09	67.70	5.82	1.00	2476.01

5. Results

All considered algorithms were implemented using Python language. The source files, the data sets and the supplementary material are available for downloading at [17].

Performance of each algorithm was measured using the Mean Absolute Error (MAE) metrics, which is calculated as

$$MAE(\mathbf{y}, \mathbf{x}) = \frac{1}{14} \sum_{i=1}^{14} |y_i - x_i|, \tag{33}$$

where $\mathbf{y_i}$ is the ground truth value and $\mathbf{x_i}$ is the value returned by an algorithm.

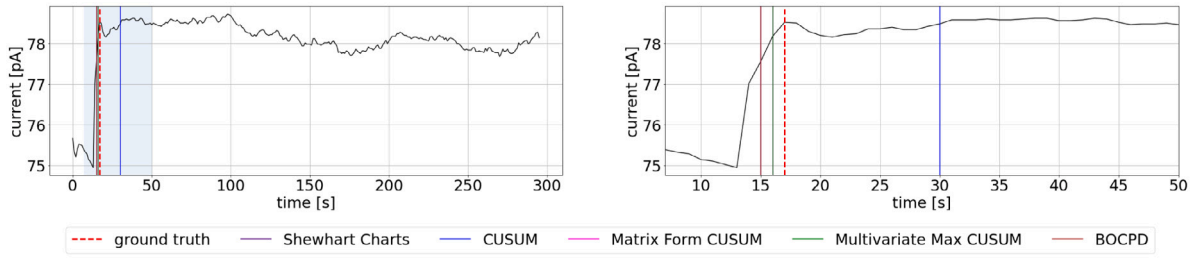
Table 1 shows running times of the algorithms. The upper part of the table "Direct times" shows running times in milliseconds of each algorithm for sliding window sizes 5, 10 and 15. The lower part of the table "Average and relative times" show average times in milliseconds. The last row shows relative running times. The calculations were performed on a MacBook Pro with CPU 2 GHz Quad-Core Intel Core i5 and 16 GB RAM. Abbreviations used in this table are:

- size — size of the sliding window
- Shewhart — Shewhart Charts algorithm
- CUSUM — CUSUM algorithm
- MaxCUSUM — Multivariate Max CUSUM algorithm
- MFCUSUM — Matrix Form CUSUM
- Bayes — Bayesian Online Change Point algorithm
- avg — average value
- rel — relative value

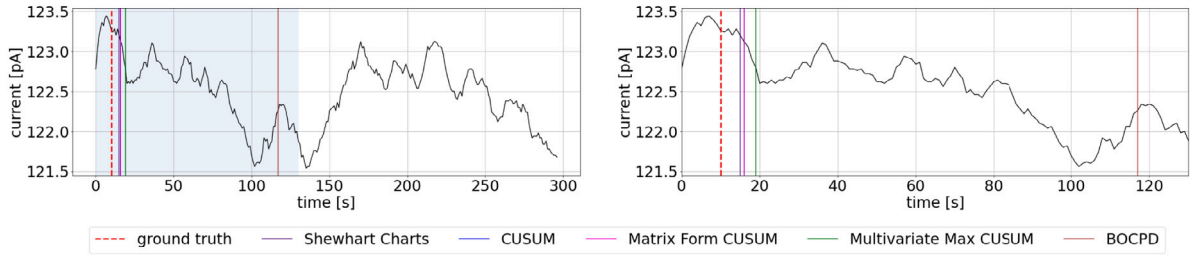
The relative running times are calculated as ratios with respect to the fastest algorithm, the Matrix Form CUSUM algorithm. As can be seen all algorithms, except for Bayesian Change Point, have relatively small running times. The Bayesian is over two thousand times slower than the Matrix Form CUSUM. This can be partly explained with sliding window. The Bayesian algorithm does not use sliding window. That is, it calculates every subsequent data point. All implemented algorithms run faster than the sampling rate of the ChemPro100i. This means that all five algorithms would enable online detection and that there would be still time for preprocessing for future studies.

Fig. 7 shows channel responses from three different data sets. The dashed red lines indicate ground truth points. The other vertical lines show change points detected by the algorithms. The left plots show the entire time series with the detected points and the ground truths. The right plots show parts of the data that are highlighted in blue on the left plots. For the change detection algorithms, the length of the sliding window was set to 15.

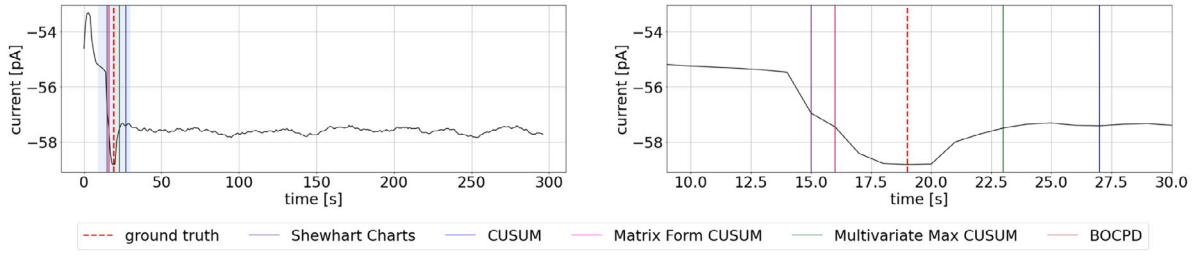
Readings in Fig. 7(a) are from a data set classified as good. The algorithms detected the change point quite accurately. As can be seen only CUSUM placed the change point slightly later. The readings shown in Fig. 7(b) are from a bad data set, which was measured from plate. Almost all data sets classified as bad were sampled from plate. The



(a) Yellow foot measured from flask. Set 3, channel 4.



(b) Black chanterelle measured from table. Set 2, channel 2.



(c) Mushrooms mix measured from flask. Set 3, channel 11.

Fig. 7. Examples of channel readings and change points yielded by change detection algorithms from (a) good, (b) bad and (c) ambiguous data sets. The left plots show channel readings over the whole 5 min measuring period. The right plots show the channel readings over the selections that are highlighted in light blue in the left plots.

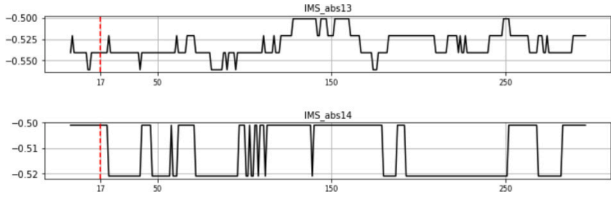


Fig. 8. Inoperable channels with binary noise.

reading in Fig. 7(b) shows that there are no visible change points. The algorithms react to local trend changes. A possible explanation of such a reading result is that the scent concentration in the air sucked into the eNose was too low to cause a significant change in the measured currents.

Fig. 7(c) shows mixture of yellowfoot and black chanterelle measured from a sealed flask. The readings show that the mixture has transient and stable phases. The algorithms performed well for the channels with multiple change points. There are multiple, possible change points visible in Fig. 7(b), which means that it has been difficult for the algorithms to find the correct change point. The ambiguous class of the data sets contains channels with both clear change point and obscured one.

Multiple data sets measured from table have channels containing only binary noise as in Fig. 8. Also for these channels ground truth points were marked and the detected change points contributed to the MAE-scores. Even so there are no change points, the algorithms still yield random points, which affects MAE-scores of a particular data set. To overcome this problem the five algorithms were modified. The modified algorithms calculated differences between the maximum and the minimum value for each channel based on the initial sample. If this difference was less than 0.05 pA then the channel was excluded from the calculation. The threshold value 0.05 was chosen based on an analysis of all the data sets, which can be found from the supplementary material.

Table 2 summarizes results over all data sets. The table shows the best algorithm with its MAE-scores (Eq. (33)) for three sliding window lengths. As can be seen from the table Multivariate Max CUSUM and Matrix Form CUSUM have often lower MAE-scores, which means that they yield change points that are closer to the ground truth point than those yielded by the remaining three methods. It is crucial, however, to keep in mind that ground truth points were selected based on intuition. Thus, their accuracy cannot be calculated.

The supplementary material shows that in case of visually detectable change the algorithms generally perform very well. The size of the sliding window plays a big role in the accuracy except for the BOCPD, which does not use a sliding window. For example, the Shewhart Charts, the CUSUM and the Matrix Form CUSUM often perform better with the largest size of the sliding window. The BOCPD algorithm often works with only clear change points with acceptable results.

In case of unclear change points the MAE scores might not be accurate because ground truth points are subjectively chosen and might

Table 2
The best algorithms for each data set. The column "*quality*" indicates whether a data set was classified as either good, bad, or ambiguous. The column "*best algorithm*" contain names of the algorithms that performed best for respective data set and window size. The letter "*e*" in parenthesis means that modified algorithm, which uses exclusion of problematic channels, was the best option. The letter "*a*" means that this algorithm used all channels.

	Set name	Quality	Window size	Best algorithm	MAE	Set name	Quality	Window size	Best algorithm	MAE
Mushrooms mix	Set 1. flask	Good	5	MaxCUSUM(a)	1.00	Set 1. table	Bad	5	Shewhart(e)	5.64
			10	MFCUSUM(e)	1.71			10	MFCUSUM(e)	0.82
			15	Shewhart(a)	1.93			15	MFCUSUM(a)	1.57
	Set 2. flask	Bad	5	MaxCUSUM(a)	7.57	Set 2. table	Bad	5	MaxCUSUM(a)	6.64
			10	MaxCUSUM(a)	5.14			10	MaxCUSUM(a)	5.79
			15	MFCUSUM(a)	5.86			15	MFCUSUM(a)	5.93
	Set 3. flask	Ambiguous	5	MaxCUSUM(a)	1.36	Set 3. table	Bad	5	Shewhart(e)	6.12
			10	Shewhart(e)	2.43			10	MFCUSUM(e)	3.64
			15	MFCUSUM(a)	1.79			15	MFCUSUM(e)	2.55
	Set 4. flask	Good	5	Shewhart(e)	3.43	Set 4. table	Bad	5	Shewhart(e)	9.75
			10	MaxCUSUM(a)	1.64			10	MaxCUSUM(a)	4.93
			15	MaxCUSUM(a)	0.64			15	MFCUSUM(e)	4.40
	Set 5. flask	Ambiguous	5	MaxCUSUM(a)	0.50	Set 5. table	Bad	5	Shewhart(e)	7.78
			10	Shewhart(e)	2.43			10	MFCUSUM(e)	4.56
			15	MFCUSUM(a)	1.50			15	MFCUSUM(e)	3.10
Black chanterelle	Set 1. flask	Ambiguous	5	MaxCUSUM(a)	0.00	Set 1. table	Bad	5	MFCUSUM(e)	5.22
			10	MFCUSUM(e)	0.00			10	MFCUSUM(e)	5.33
			15	Shewhart(e)	1.64			15	MaxCUSUM(a)	5.57
	Set 2. flask	Ambiguous	5	MaxCUSUM(a)	2.14	Set 2. table	Bad	5	MaxCUSUM(e)	11.00
			10	MFCUSUM(e)	0.43			10	CUSUM(e)	14.64
			15	MFCUSUM(a)	2.14			15	MFCUSUM(e)	13.82
	Set 3. flask	Ambiguous	5	MaxCUSUM(a)	2.00	Set 3. table	Bad	5	Shewhart(e)	7.88
			10	MFCUSUM(e)	2.00			10	MFCUSUM(e)	6.78
			15	MFCUSUM(e)	2.00			15	MFCUSUM(e)	5.80
	Set 4. flask	Ambiguous	5	MaxCUSUM(a)	1.21	Set 4. table	Bad	5	Shewhart(e)	4.50
			10	MFCUSUM(e)	1.21			10	Shewhart(e)	2.90
			15	MFCUSUM(a)	1.36			15	MaxCUSUM(a)	2.86
	Set 5. flask	Ambiguous	5	Shewhart(e)	3.43	Set 5. table	Bad	5	Shewhart(e)	15.11
			10	MFCUSUM(e)	0.43			10	MFCUSUM(e)	11.70
			15	MFCUSUM(a)	2.14			15	MaxCUSUM(a)	3.64
Yellowfoot	Set 1. flask	Ambiguous	5	MFCUSUM(e)	1.29	Set 1. table	Bad	5	Shewhart(e)	8.40
			10	MFCUSUM(e)	1.57			10	MFCUSUM(e)	4.40
			15	MFCUSUM(a)	1.00			15	MFCUSUM(e)	4.00
	Set 2. flask	Good	5	Shewhart(e)	2.14	Set 2. table	Ambiguous	5	Shewhart(e)	4.18
			10	Shewhart(e)	2.21			10	MFCUSUM(e)	0.91
			15	MFCUSUM(a)	1.43			15	Shewhart(e)	2.27
	Set 3. flask	Good	5	MaxCUSUM(a)	1.93	Set 3. table	Bad	5	Shewhart(e)	9.20
			10	MaxCUSUM(e)	0.00			10	MaxCUSUM(e)	0.00
			15	MaxCUSUM(a)	1.07			15	MaxCUSUM(a)	7.64
	Set 4. flask	Good	5	MaxCUSUM(e)	0.00	Set 4. table	Bad	5	Shewhart(e)	4.00
			10	MaxCUSUM(a)	2.64			10	MFCUSUM(e)	2.18
			15	MFCUSUM(a)	1.64			15	MFCUSUM(a)	1.43
	Set 5. flask	Good	5	Shewhart(e)	2.71	Set 5. table	Bad	5	MFCUSUM(e)	5.44
			10	MFCUSUM(e)	2.29			10	MFCUSUM(e)	2.67
			15	MFCUSUM(a)	1.29			15	MFCUSUM(e)	3.00

not represent the true ground truth. The second problem is that some channels do not react to certain scents as seen in Fig. 2(b). However, the ground truth points were selected for such channels and compared to the results of the algorithms. The ground truth points for such readings were selected according to the first local peak or by setting them close to ground truth points of other channels. For addressing this problem we modified the algorithms to reject channels, which possibly contain only noise. The drawback of this technique is that it might affect the accuracy in case of channels that do not react immediately to a presented scent. The third problem is that for samples measured from table the channels 14 and 15 almost always contain binary noise (Fig. 8). We could not extract any valuable information from such channels. Nonetheless, ground truth for such channels was determined and the found change points contributed to the MAE metrics.

For addressing the last two problems we modified algorithms to reject channels that did not contain much variability within the initial sample. However, by rejecting channels with small variability in the initial sample we could potentially loose channels that have slow reaction to a scent. As for the computational demand, all algorithms ran faster than the sample rate of the ChemPro. The Bayesian Online Change Point Detector was the slowest of all considered algorithms and the Matrix Form CUSUM was the fastest one. However, all algorithms were run on a laptop. Based on the average run times presented in Table 1 it is save to assume that all algorithms, except for the Bayesian approach, will enable real-time change detection even on embedded devices with less computational capacity if the sampling rate is kept at 1 Hz. For the Bayesian approach it depends on the exact computational capacity of the embedded device whether real-time change detection is possible.

6. Discussion and outlook

In this paper we implemented five algorithms for detecting change points in the ion mobility spectrometry readings. These algorithms are very simple and easy to implement. Focusing on more simple algorithms allowed us to evaluate them on more thorough manner

and decide how such approaches are suitable to ensure classification methods.

The Shewhart Control charts, CUSUM and Bayesian Online Change Point detection algorithms performed reliably but must be run on each channel separately. The BOCPD generally worked well in the case of clearly visible change points. Thus, these algorithms suited our needs and may be used for change point detection of IMS readings. The Multivariate Max-CUSUM and Matrix Form CUSUM that calculate all channels simultaneously turned out to be the fastest and the most reliable of all considered algorithms.

In case of visually detectable change points all algorithms performed well. In other cases the algorithms detected the first peak in the readings, which can be possibly a change point. Using change detection algorithms for automatic detection of stable phase removes subjectivity from labeling change points, but might introduce also some false detections as they always pick one point as change point. This drawback was circumvented by adding a pre-processing step that checked whether a channel reacted to a measured scent.

Besides the subjectivity of choosing ground truth there was a problem with the data sets measured from a table. Very often in such data sets the channels 7, 14 and 15 showed a binary noise. We do not have explanations on that behavior. Again, using a pre-processing step to eliminate such channels can solve this issue.

Generally all algorithms performed well and detected change points. Based on the results we recommend using Matrix Form CUSUM and Multivariate Max-CUSUM with IMS data. By running algorithms online we can automatically separate the transient phase from the stable phase and use features of the transient phase to strengthen and speed up the classification algorithms. For example, such features can be: length of transient phase, variance, derivatives, etc. Future research of change point detection can be done using more complex algorithms, such as subspace identification [13], neural networks [16] and time series analysis [29].

CRediT authorship contribution statement

Anton Kondratev: Writing – original draft, Methodology, Software, Formal analysis, Investigation, Visualization. **Katri Salminen:** Data curation, Writing – review & editing. **Jussi Rantala:** Writing – review & editing. **Timo Salpavaara:** Writing – review & editing. **Jarmo Verho:** Writing – review & editing. **Veikko Surakka:** Writing – review & editing, Project administration, Funding acquisition. **Jukka Lekkala:** Writing – review & editing, Project administration, Funding acquisition. **Antti Vehkaoja:** Writing – review & editing, Supervision, Validation. **Philipp Müller:** Writing – review & editing, Conceptualization, Supervision, Validation.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.array.2022.100151.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.array.2022.100151.

References

[1] Aminikhanghahi S, Cook DJ. A survey of methods for time series change point detection. Knowl Inf Syst 2017;51(2):339–67. http://dx.doi.org/10.1007/s10115-016-0987-z, URL: https://www.ncbi.nlm.nih.gov/pubmed/28603327.

[2] Besse P, Gall CL. Application and reliability of change-point analyses for detecting a defective stage in integrated circuit manufacturing. Commun Stat Simul Comput 2006;35(2):479–96. http://dx.doi.org/10.1080/03610910600591602, ID: cdi_pascalfrancis_primary_17756599.

[3] Filic M, Filjar R. GNSS positioning error change-point detection in GNSS positioning performance modelling. TransNav (Gdynia, Poland) 2019;13(3):575–9. http://dx.doi.org/10.12716/1001.13.03.12.

[4] Xie L, Xie Y, Moustakides GV. Asynchronous multi-sensor change-point detection for seismic tremors. 2019.

[5] Basseville M, Nikiforov IV. Detection of abrupt changes: theory and application. Prentice Hall; 1993.

[6] Cheng SW, Thaga K. Multivariate max-CUSUM chart. Qual Technol Quant Manag 2005;2(2):221–35. http://dx.doi.org/10.1080/16843703.2005.11673095, URL: http://www.tandfonline.com/doi/abs/10.1080/16843703.2005.11673095.

[7] van den Burg GJJ, Williams CKI. An evaluation of change point detection algorithms. 2020, URL: https://arxiv.org/abs/2003.06222.

[8] Douglas CM. Introduction to statistical quality control. 6th ed.. Wiley; 2008.

[9] Habibi R. BayesIan online change point detection in finance. Financial Internet Q 2022;17(4):27–33. http://dx.doi.org/10.2478/fiqf-2021-0025.

[10] Leyli-Abadi M, Samé A, Oukhellou L, Cheifetz N, Mandel P, Féliers C, et al. Online common change-point detection in a set of nonstationary categorical time series. Neurocomputing 2021;439:176–96. http://dx.doi.org/10.1016/J.NEUCOM.2021.01.066.

[11] Fan Y, Lu X. An online Bayesian approach to change-point detection for categorical data. Knowl-Based Syst 2020;196:105792. http://dx.doi.org/10.1016/J.KNOSYS.2020.105792.

[12] Li Y, Bao T, Shu X, Gao Z, Gong J, Kang Zhang. Data-driven crack behavior anomaly identification method for concrete dams in long-term service using offline and online change point detection. J Civ Struct Health Monit 2016;11:1449–60. http://dx.doi.org/10.1007/s13349-021-00520-w.

[13] Kawahara Y, Yairi T, Machida K. Change-point detection in time-series data based on subspace identification. In: Proceedings - IEEE international conference on data mining. 2007, p. 559–64. http://dx.doi.org/10.1109/ICDM.2007.78.

[14] Müller P, Salminen K, Kontunen A, Karjalainen M, Isokoski P, Rantala J, et al. Online scent classification by ion-mobility spectrometry sequences. Front Appl Math Stat 2019;5. http://dx.doi.org/10.3389/fams.2019.00039.

[15] Hierlemann A, Gutierrez-Osuna R. Higher-order chemical sensing. Chem Rev 2008;108:563–613. http://dx.doi.org/10.1021/cr068116m.

[16] Titsias MK, Sygnowski J, Chen Y. Sequential changepoint detection in neural networks with checkpoints. 2020, URL: https://arxiv.org/abs/2010.03053.

[17] Kondratev A, et al. Supplementary material for this article. http://dx.doi.org/10.5281/zenodo.5282461.

[18] Loutfi A, Coradeschi S, Mani GK, Shankar P, Rayappan JBB. Electronic noses for food quality: A review. J Food Eng 2015;144:103–11. http://dx.doi.org/10.1016/j.jfoodeng.2014.07.019, URL: http://dx.doi.org/10.1016/j.jfoodeng.2014.07.019.

[19] Dodds J, Baker E. Ion mobility spectrometry: Fundamental concepts, instrumentation, applications, and the road ahead. J Am Soc Mass Spectrom 2019;30(11):2185–95. http://dx.doi.org/10.1007/s13361-019-02288-2, URL: https://www.ncbi.nlm.nih.gov/pubmed/31493234.

[20] Mäkinen M, Anttalainen O, Sillanpää M. Ion mobility spectrometry and its applications in detection of chemical warfare agents. Anal Chem (Washington); Anal Chem 2010;82(23):9594–600. http://dx.doi.org/10.1021/ac100931n.

[21] Barnett DA, Ells B, Guevremont R, Purves RW, Viehland LA. Evaluation of carrier gases for use in high-field asymmetric waveform ion mobility spectrometry. J Am Soc Mass Spectrom 2000;11(12):1125–33.

[22] Oy E. ChemPro100i Operator and Unit Support Manual. URL: https://environics.fi/products/chempro100i/.

[23] Hyndman RJ, Athanasopoulos G. Forecasting: principles and practice. 3rd ed.. Melbourne, Australia: Otexts, URL: https://otexts.com/fpp3/. [Accessed 04 March 2021].

[24] PAGE ES. Continuous inspection schemes. Biometrika 1954;41(1–2):100–15. http://dx.doi.org/10.1093/biomet/41.1-2.100.

[25] Kondratev A. Online change detection for ion-mobility spectrometry readings. 2021, URL: http://urn.fi/URN:NBN:fi:tuni-202102051941.

[26] Adams R, MacKay D. BayesIan online changepoint detection. 2007, arXiv: Machine Learning.

[27] Grant J. Bayesian changepointdetection in solar activity data. School of Mathematics and Statistics of the University of Glasgow; 2014, URL: https://www.lancaster.ac.uk/postgrad/grantj/mastersdiss.pdf.

[28] Rinne H. The hazard rate : theory and inference (with supplementary MATLAB-programs). Justus-Liebig-Universität; 2014, URL: http://geb.uni-giessen.de/geb/volltexte/2014/10793.

[29] Choi H, Ombao H, Ray B. Sequential change-point detection methods for nonstationary time series. Null 2008;50(1):40–52. http://dx.doi.org/10.1198/004017007000000434.