[Array 14 (2022) 100151](https://doi.org/10.1016/j.array.2022.100151)

|  |  |  |
| --- | --- | --- |
|  | Contents lists available at [ScienceDirect](https://www.elsevier.com/locate/array) |  |
| Array |
| journal homepage: [www.elsevier.com/locate/array](http://www.elsevier.com/locate/array) |
|  | | |

A comparison of online methods for change point detection in ion-mobility spectrometry data✩  
Anton Rauhameria,∗, Katri Salminenb, Jussi Rantalaa, Timo Salpavaaraa, Jarmo Verhoa, Veikko Surakkaa, Jukka Lekkalaa, Antti Vehkaojaa,1, Philipp Müllera,1   
a *Tampere University, Korkeakoulunkatu 7, Tampere, 33720, Finland*   
b *Tampere University of Applied Sciences, Kuntokatu 3, Tampere, 33520, Finland*

|  |  |  |
| --- | --- | --- |
| A R T I C L E | I N F O | A B S T R A C T |
| *Keywords:*  Algorithms  Change detection  Ion mobility spectrometry | | When on-site classification of volatile organic compounds (VOCs) is required, a portable ion mobility spectrom-eter (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 appli-cation 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 meth-ods [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 identifica-tion. 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 algo-rithms 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 a[uthor.](mailto:anton.rauhameri@tuni.fi)

*E-mail address:* [anton.rauhameri@tuni.fi](mailto:anton.rauhameri@tuni.fi) (A. Rauhameri).

1 A. Vehkaoja and [P. Müller contributed e](mailto:anton.rauhameri@tuni.fi)qually to this article.

<https://doi.org/10.1016/j.array.2022.100151>  
[Received 5 November 2021; Received in revi](https://doi.org/10.1016/j.array.2022.100151)sed form 8 March 2022; Accepted 1 April 2022 Available online 16 April 2022

2590-0056/© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

*A. Rauhameri et al.*  *Array 14 (2022) 100151*

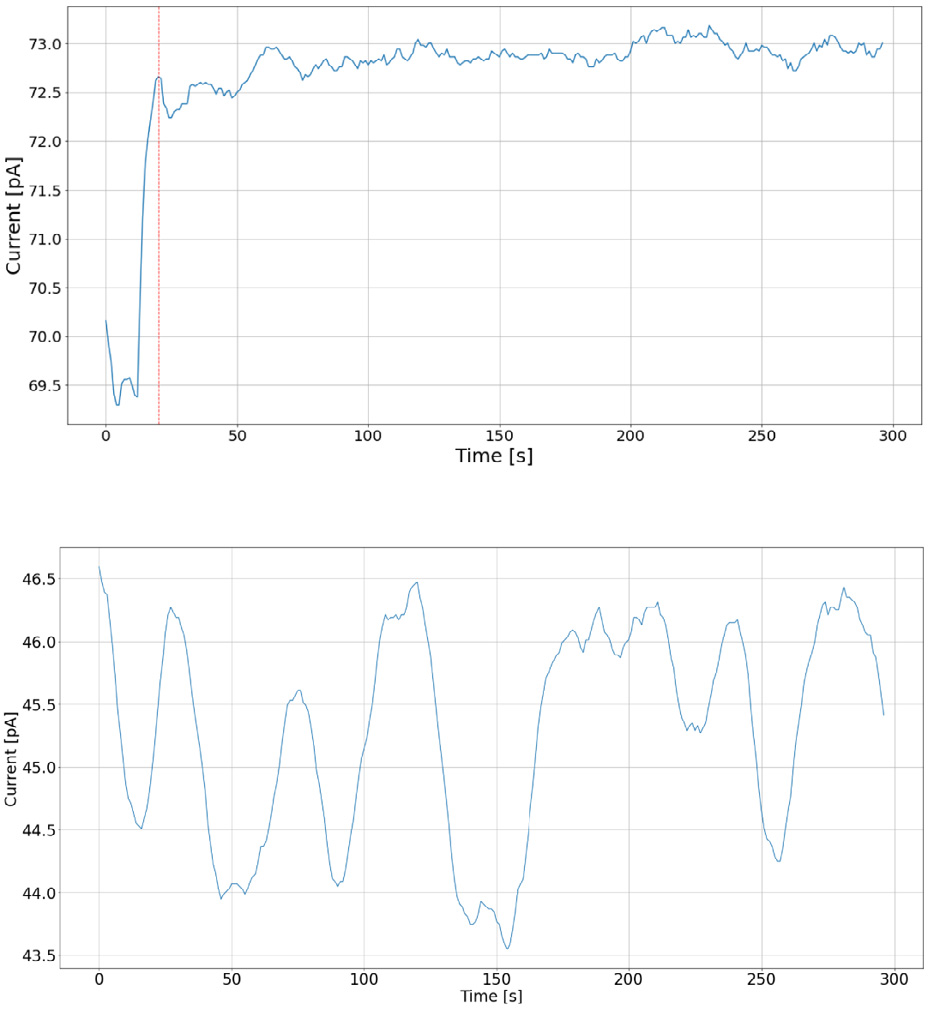
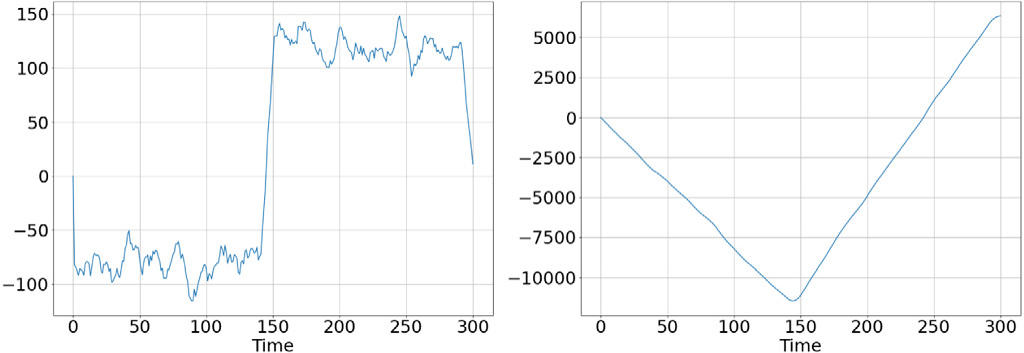
data from the stable phase for scent classification. However, this slows down the classification as one has to wait for the transient phase to end, which typically lasts from few seconds up to 1–2 min. However, information contained in the transient phase can be, according to some researchers, more valuable 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, de-pending on the eNose technology and chemical sensor or sensor ar-ray 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 sen-sors) 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

2



*A. Rauhameri et al.*  *Array 14 (2022) 100151*

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



**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

|  |  |  |
| --- | --- | --- |
| *𝐿𝑁* 1= *~~𝑏~~ 𝜎* | ∑(*𝑦𝑖* − *𝜇*0 −*𝑣* ~~2~~ ) | (2) |
| where |

*𝑣* = *𝜇*1 − *𝜇*0

*𝑏* =*𝑣*   
*𝜎*

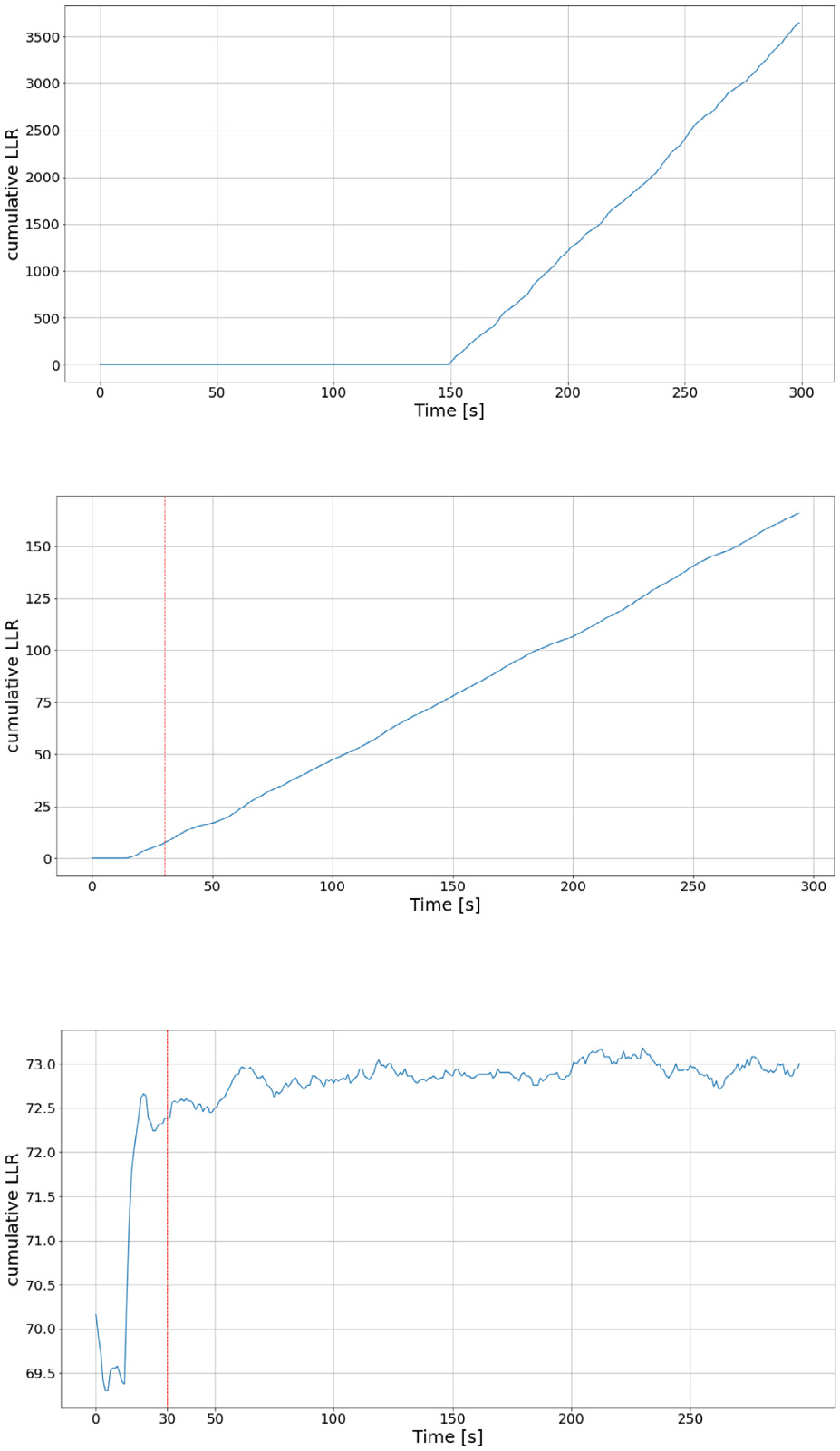
and *𝑁* is the size of the sliding window. A sliding window is a structure that rolls over a time series. It always contains the current and *𝑁* − 1 previous data points. This technique enables sampling and calculating statistical information for time series online, and can be used for smoothing time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

*𝐿* =∑*𝐿𝑁 𝑖*  (3)

for making a decision (decision function) is proposed. A change point is detected if   
*𝐿* ≥ *ℎ*  (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

3



|  |  |
| --- | --- |
| *A. Rauhameri et al.*  LLR-value found in the previous iterations. The decision function is  defined as  *𝑔𝑘* = *𝐿𝑘* − *𝑚𝑘* ≥ *ℎ,*  (5)  where *𝐿𝑘* is the cumulative LLR, *𝑚𝑘* is the minimum LLR-value found  in the previous iterations and *ℎ* is a conveniently chosen threshold. (5)  can be rewritten as  *𝑔𝑘* = *𝐿𝑘* ≥ *ℎ* + *𝑚𝑘.*  (6)  In the algorithm implemented in this work zero threshold was used.  Therefore, (6) simplifies to  *𝑔𝑘* = *𝐿𝑘* ≥ *𝑚𝑘.*  (7) | *Array 14 (2022) 100151* |

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, *𝑥𝑖* - 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** *̄𝜇*0 =1  **4** *̄𝜎*0 =  first S samples∑(*𝑥𝑖*− *̄𝜇*)√∑*𝑠 𝑠*−1# sample standard deviation of the *𝑖*=0*𝑥𝑖*# sample mean of the first s samples  **5** *detected = False*  **6** i = 0 # counter  **7 while** *detected == False* **do**  **8**  i = i + 1;  **9**  sample ← [*𝑦𝑖* ∶ *𝑦𝑖*+*𝑠*]; # Get next sample of length s  **10**  *𝑣* = 0 − *̄𝜇*0;  **11**  *𝑏* =*̄𝜎*0;  **12**  **13**  *𝐿* = *𝐿* + (1∕*̄𝜎*0)(∑*𝑠𝑎𝑚𝑝𝑙𝑒* − *𝑠* ⋅ *̄𝜇*0 −*𝑠*⋅*𝑣* 2);  **14**  **if** (*𝐿* − *𝑚𝑖𝑛*(*𝐴*)) *>* 0 **then**  **15**  *𝑑𝑒𝑡𝑒𝑐𝑡𝑒𝑑* ← *𝑇 𝑟𝑢𝑒*;  **16**  *𝑐ℎ𝑎𝑛𝑔𝑒*\_*𝑝𝑜𝑖𝑛𝑡* ← *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 uni-variate classic CUSUM testing.

The classic scheme for CUSUM works here as well:

*𝐿𝑖* = *𝑚𝑎𝑥*(*𝐿𝑖*−1 + **𝐚**′(**𝐱𝐧** − ***𝝁𝑮***) − 0*.*5*𝐷,* 0) *> 𝐻*  (8)

4

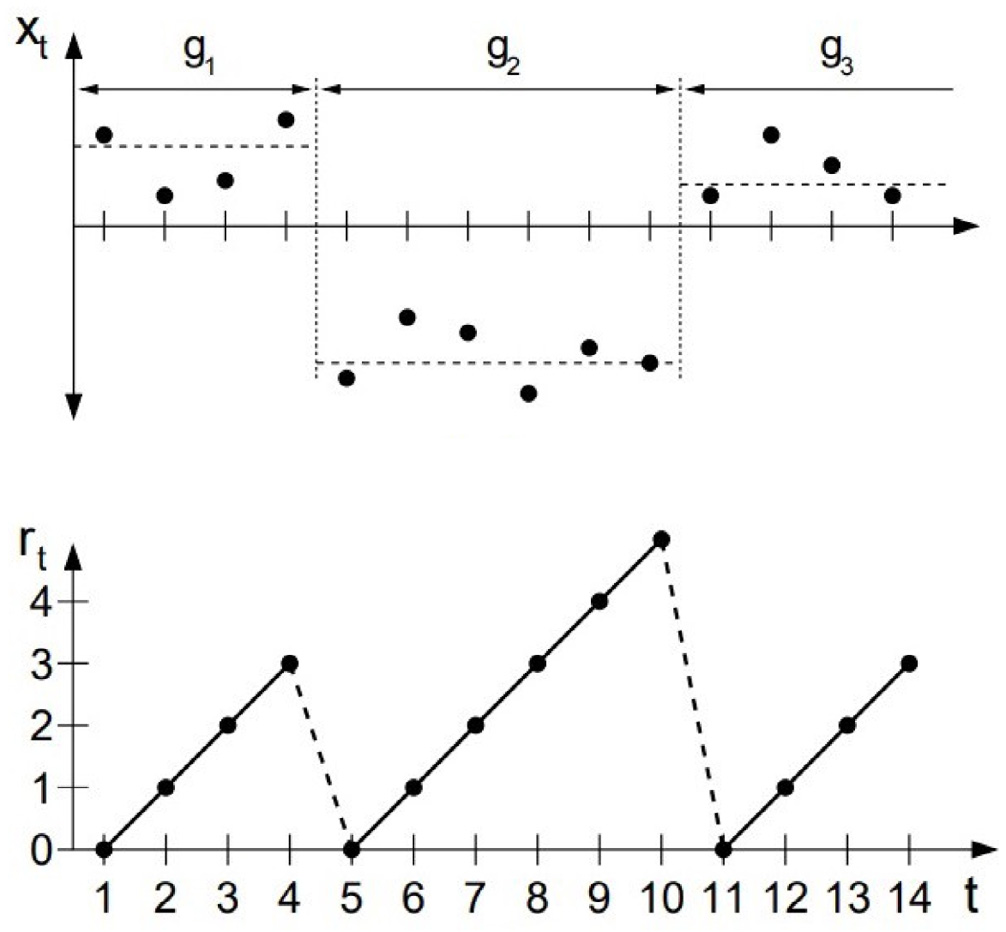
*A. Rauhameri et al.*  *Array 14 (2022) 100151*

|  |
| --- |
| **Algorithm 3:** Multivariate Max-CUSUM Chart  **Input:** s - sliding window size, *𝑋𝑖* - sample of size 14 × *𝑠*  **Result:** time step *𝑖* at which distribution has changed  **1** *𝜇*0 = *̄𝜇* # sample mean along each row  **2** *𝜇*1 = 0 # assumes Normal distributed data after  change point  **3** *𝛴* =*̄𝛴* # sample covariance matrix  **4** *𝛴𝑖,𝑗* = *𝛴𝑖,𝑗* + 1*𝑒* − 10*,* where i = j # adding 1e-10 to main  diagonal avoids singularity  **5** a = *𝜇*0)*𝑇𝛴*−1  **6** *𝐿𝑖* = 0 # initial value for cumulative sum√ *𝜇*0)*𝑇𝛴*−1(*𝜇*1  **7** detected = False  **8** point = 0  **9** i = 0 # counter  **10 while** *detected == False* **do**  **11**  *𝑋𝑖* = *data points of size* 14 × *𝑠*  **12**  *𝜇*1 = mean(*𝑋𝑖*  **13**  **14**   *𝐷* =  *𝐿𝑖* = *𝑚𝑎𝑥*(0*, 𝐿𝑖*−1 + *𝑎*(*𝜇*1√ *𝜇*0)*𝑇𝛴*−1(*𝜇*1  **15**  **if** *𝐿𝑖 >* 0 **then**  **16**  point = i + *𝑠*  **17**  detected = True  **18**  **end**  **19**  i = i + 1  **20 end** |

which can be rewritten in more compact form as

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (**𝐗𝐢** − **𝐅***𝜇*)2× **𝟏***𝑠* | | | | (16) |
| Eventually, the *𝑛*-dimensional standard deviation is  **𝐒𝐓𝐃** =  Notice that in (17) square and square root are calculated for each entry√(**𝐗𝐢** − **𝐅***𝜇*)2× **𝟏***𝑠*  (17)  of the matrix.  The log-likelihood ratio (LLR) for CUSUM mean shift for one dimen-sion [5] is calculated as | | | | |
| *𝐿𝑠* 1= ( *~~𝑏~~ 𝜎*)∑(*𝑦𝑖* − *𝜇*0 −*𝑣* ~~2~~ ) = ( *~~𝑏~~ 𝜎*)(∑*𝑦𝑖* − *𝑠𝜇*0 −*𝑠𝑣* ~~2~~ )  The same LLR is used for the Matrix Form CUSUM where: | | | | (18) |
| *𝜎* = **𝐒𝐓𝐃** | | | | |
| **𝐯** = −1 × **𝐦**  **𝐛** = **𝐯** *⊙***𝐒𝐓𝐃** 1 | | | | (19)  (20) |
| ∑*𝑦𝑖* = **𝐗𝐢** ×[ 1  *𝑠𝜇*0 = *𝑠* × **𝐦** | 1 | *...* | 1 ]*𝑇* | (21) |
| (22) |
| Notice *𝑠* is scalar and *𝐿𝑠* 1is the LLR calculated for the first sample of  size *𝑠*. Symbol *⊙* in (20) denotes element-wise multiplication.  The decision function is calculated as  *𝑔𝑘* = *𝑎𝑣𝑒𝑟𝑎𝑔𝑒*(**𝐋𝐬 𝐤**) − *𝑚𝑘* ≥ *ℎ*  (23) | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 | |  | where | |
| *𝑚𝑘* = *𝑚𝑖𝑛*(*𝐿𝑗*) 1≤*𝑗<𝑘* | (24) |
| and *ℎ* is a conveniently chosen threshold. The right side of (24) repre-sents the minimum value of all previously calculated LLR. Each entry of vector **𝐋𝐬 𝐤**is the LLR for one channel. That is, each entry of the vector **𝐋𝐬 𝐤**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: | |
| **𝐗***𝑖* =  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, *𝑥𝑖,𝑗* ⎡⎢⎢⎣ *𝑥*14*,*1 ⋮ ⋱  *...*  *𝑥*14*,𝑠* ⋮ ⎤⎥⎥⎦ ∈ R14×*𝑠*  (11) *𝑥*1*,*1 *...*  *𝑥*1*,𝑠*  is the *𝑗*th reading of the *𝑖*th channel. The vector of sample means for each channel is calculated as follow | |
| **𝐗𝐢** ×  The standard deviation for one dimension is calculated as⎡⎢⎢⎢⎣⋮  1  *𝑠*  *𝑠*⎤⎥⎥⎥⎦*𝑠*×1 =⎡⎢⎢⎢⎢⎣*̄𝜇*14*̄𝜇*2  *...*⎤⎥⎥⎥⎥⎦ (12) 1 *̄𝜇*1  *̄𝜎* =√ ~~∑~~ (*𝑥𝑖* − *𝜇*)2 (13)  The one-dimensional standard deviation can be converted into the  matrix form as follows: | |
| 1  **𝐅***𝜇* =[  Then the fraction under the square root in (13) is calculated by **𝐦**  **𝐦**  *...*  **𝐦** ] ∈ R14×*𝑠,* **𝟏***𝑠* =⎡⎢⎢⎢⎣ *𝑠*−1  *𝑠*−1 ⋮  1⎤⎥⎥⎥⎦∈ R*𝑠*×1  1  ⎡⎢⎢⎣(*𝑥*14*,*1 − *𝜇*14)2  (*𝑥*1*,*1 − *𝜇*1)2  ⋮  *...*  *...*  ⋮  (*𝑥*14*,𝑠* − *𝜇*14)2  (*𝑥*1*,𝑠* − *𝜇*1)2  ⋮ ⋮⎤⎥⎥⎦ ⎡⎢⎢⎢⎣ *𝑠*−1  *𝑠*−1 ⋮  1⎤⎥⎥⎥⎦*,* | (14) |
| 1. Calculate posterior predictive probability 2. Calculate probability of growth  3. Calculate probability of change point 4. Update statistics | |
| (15) | 5 |
| For implementing this algorithm the matrix **𝐑** was used (Algorithm 5). Matrix **𝐑** is shown in Fig. 6. The first column of **𝐑** represents | |



*A. Rauhameri et al.*  *Array 14 (2022) 100151*

|  |
| --- |
| **Algorithm 4:** Matrix Form CUSUM  **Input:** s - sliding window size, *𝑋𝑖* - sample of size 14 × *𝑠*  **Result:** time step *𝑖* at which distribution has changed  **1 𝐋𝐢** ←[ cumulative LLR 0 0 *...*  0 ]*𝑇* # initialize vector of  **2 𝐌** = **𝐗𝐢** × **𝟏***𝑠* # vector of means  **3 𝐅***𝜇* ←[  standard deviations   **𝐌**  √(**𝐗𝐢** − **𝐅***𝜇*)2× **𝟏***𝑠* # calculate vector of **𝐌**  *...*  **𝐌** ] # 14xs matrix  **5 𝐯** = −1 × **𝐌**  **6 𝐛** = **𝐯** *⊙***𝐒𝐓𝐃** 1  **7 𝐈** ←[ storing minimum values of LLR 0 0 *...*  0 ]*𝑇* # 14-dimensional vector for  **8** detected = False  **9** point = 0  **10** i = 0 # loop counter  **11 while** *detected == False* **do**  **12**  **𝐗𝐢** ← *new chunk of data of size* 14 × *𝑠*  **13**  **14**  **15**   **𝐋𝐢** = **𝐋𝐢** + (  **if 𝐋𝐢** *<* **𝐈 then**  **𝐈** ← **𝐋𝐢 𝐒𝐓𝐃**)(**𝐗𝐢** × [ 1 1 *...*  1 ]*𝑇* − *𝑠* × **𝐌** − *𝑠* 2× **𝐯**)  **16**  **end**  **17**  **if** *𝑎𝑣𝑒𝑟𝑎𝑔𝑒*(**𝐋𝐢** − **𝐈**) *>* 0 **then**  **18**  point = i + *𝑠*  **19**  *𝑑𝑒𝑡𝑒𝑐𝑡𝑒𝑑* ← *𝑇 𝑟𝑢𝑒*  **20**  **end**  **21**  i = i + 1  **22 end** |



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

probabilities of change at times *𝑡*1*,* … *, 𝑡𝑛*. The remaining columns rep-resent 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 de-tection results. The t-distribution was used because *𝜇* and *𝜎*2of the

6

*A. Rauhameri et al.*  *Array 14 (2022) 100151*

|  |
| --- |
| **Algorithm 5:** Bayesian Change Point with t-distribution as UPM **Result:** time step *𝑖* at which distribution has changed  **1 *𝑹*** ← **𝟎** # initialize *𝑛* × *𝑛* matrix R with zeros  **2** *𝑅*[0*,* 0] = 1 # set first value to one  **3 *𝒍*** ← [1] # create a vector for storing information from previous step  **4** *𝐻* = 100# initialize hazard rate 1  **5** *𝛼*0 = *𝜅*0 = 1 # initialize parameters of UPM   distribution  **6** *𝜇*0 = *̄𝜇* # set mean to sample mean  **7** *𝛽*0 = *𝐻* # set *𝛽* to the Hazard rate  **8 *𝒂* = *𝒃* = *𝒎* = *𝒌*** = [] # initialize vectors for *𝛼, 𝛽, 𝜇, 𝜅*  for saving all previous values  **9** found = False  **10** i = 1 # counter  **11 while** *found == False* **do**  **12**  *𝑥𝑖* ← *next point*  **13**  ***𝝅*** = *𝑓𝑢𝑛𝑐𝑡𝑖𝑜𝑛 get\_predictive\_probabilities(𝑥𝑖,* ***𝒂****,* ***𝒃****,* ***𝒎****,* ***𝒌****)* **14**  ***𝒈*** = ***𝒍*** × ***𝝅*** × (1 − *𝐻*) # calculate growth   probabilities  **15**  **16**  ***𝒄*** =∑(***𝒍*** × ***𝝅*** × *𝐻*) # calculate change point ***𝒏𝒈*** = [***𝒄****,* ***𝒈***] # concatenate into one vector change and growth probabilities  **17**  R[i,:] = ***𝒏𝒈***  R-matrix∑ ***𝒏𝒈*** # normalize and put into the **18**  ***𝝁𝒏*** =(***𝒌***×***𝒎***+*𝑥𝑖*) ***𝒌***+1 # update mean-value  **19**  ***𝜿𝒏*** = ***𝒌*** + 1 # update kappa-value  **20**  ***𝜶𝒏*** = ***𝒂*** +1 2# update alpha-value  **21**  ***𝜷𝒏*** = ***𝒃*** +***𝒌***(*𝑥𝑖*−***𝒎***)2 2(***𝒌***+1)# update beta-value  **22**  ***𝒂*** = [*𝛼*0*,* ***𝜶𝒏***] # concatenate vector alpha  **23**  ***𝒃*** = [*𝛽*0*,* ***𝜷𝒏***]  **24**   ***𝒎*** = [*𝜇*0*,* ***𝝁𝒏***] **25**  ***𝒌*** = [*𝜅*0*,* ***𝜿𝒏***]  **26**  ***𝒍*** = [*𝐶, 𝐺*] # concatenate for next iteration **27 end**  **28 Function** *get\_predictive\_probabilities(𝑥𝑖,* ***𝒂****,* ***𝒃****,* ***𝒎****,* ***𝒌****)***:**  **29**   df = 2 × ***𝒂* 30**  loc = ***𝒎***  **31**  **32**  scale = **return** probability t(x,df, loc, scale)√***𝒃***(***𝒌***+1) ***𝒂***×***𝒌***  **33 return** |

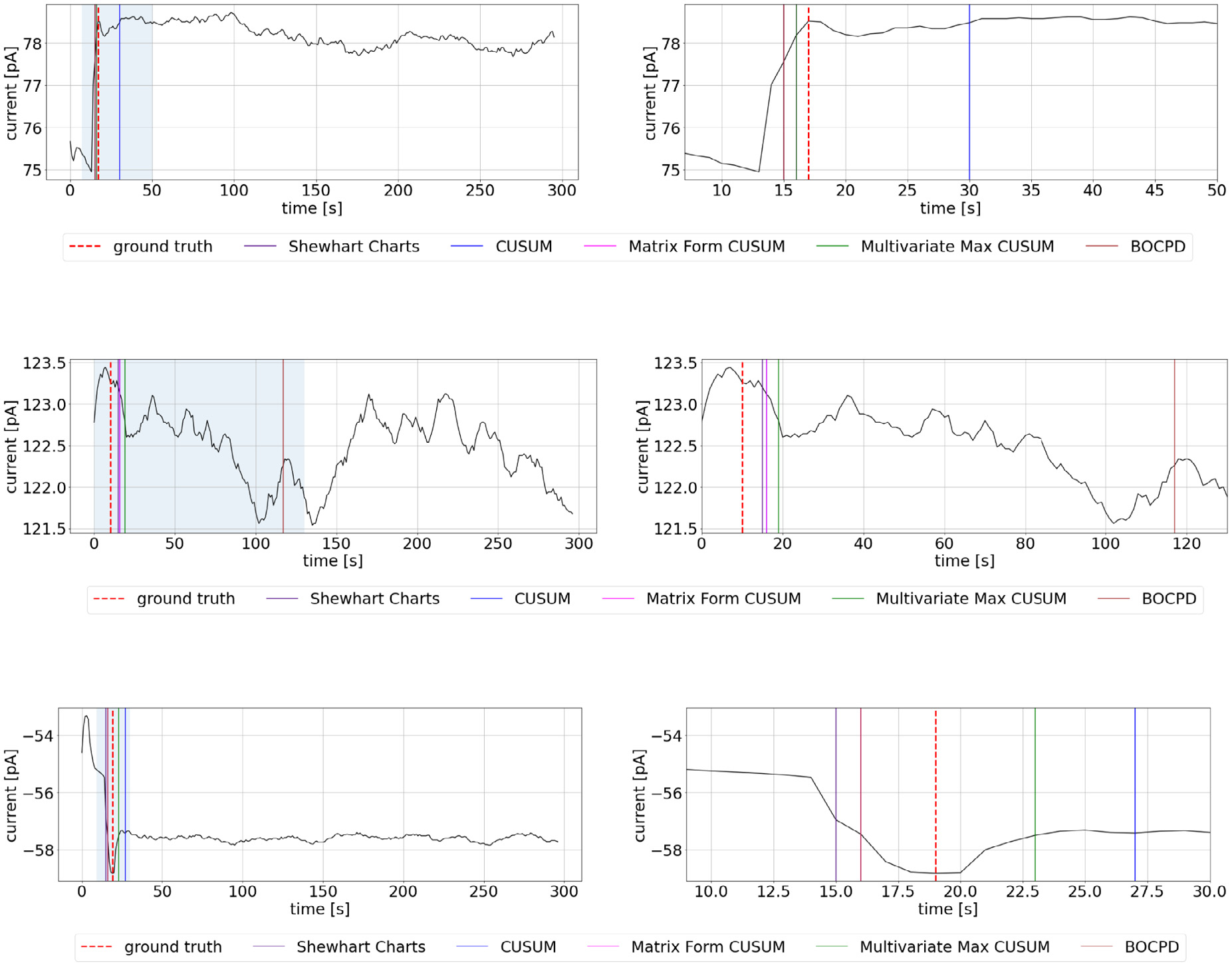
**4. Data**

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 ambigu-ous. 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

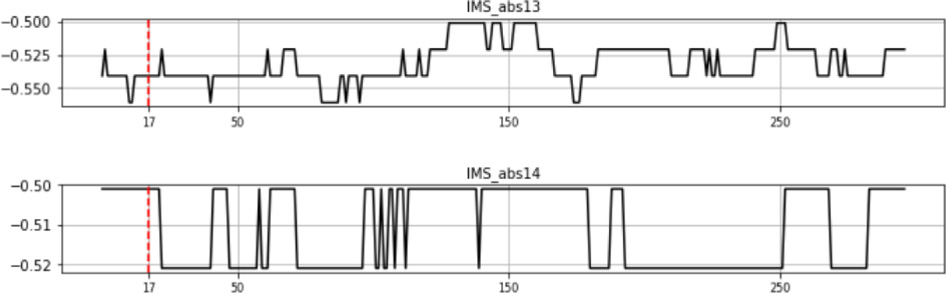
7



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *A. Rauhameri et al.* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | *Array 14 (2022) 100151* |
|  |



**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 mea-sured 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

8

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *A. Rauhameri et al.* | | *Array 14 (2022) 100151* | | | | | | | | | | | | | | | | |
| **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 | |
| Set 1. flask | | Good | 5  10  15 | | | MaxCUSUM(a) MFCUSUM(e) Shewhart(a) | | 1.00  1.71  1.93 | Set 1. table | | Bad | 5  10  15 | | | Shewhart(e)  MFCUSUM(e) MFCUSUM(a) | | 5.64  0.82  1.57 | |
| Mushrooms mix | Set 2. flask | Bad | | | 5  10  15 | MaxCUSUM(a) MaxCUSUM(a) MFCUSUM(a) | | 7.57  5.14  5.86 | Set 2. table | | Bad | 5  10  15 | | | MaxCUSUM(a) MaxCUSUM(a) MFCUSUM(a) | | | 6.64  5.79  5.93 |
| Set 3. flask | Ambiguous | | | 5  10  15 | MaxCUSUM(a) Shewhart(e)  MFCUSUM(a) | | 1.36  2.43  1.79 | Set 3. table | | Bad | 5  10  15 | | | Shewhart(e)  MFCUSUM(e) MFCUSUM(e) | | | 6.12  3.64  2.55 |
| Set 4. flask | Good | | | 5  10  15 | Shewhart(e)  MaxCUSUM(a) MaxCUSUM(a) | | 3.43  1.64  0.64 | Set 4. table | | Bad | 5  10  15 | | | Shewhart(e)  MaxCUSUM(a) MFCUSUM(e) | | | 9.75  4.93  4.40 |
| Set 5. flask | | Ambiguous | | | 5  10  15 | MaxCUSUM(a) Shewhart(e)  MFCUSUM(a) | | 0.50  2.43  1.50 | Set 5. table | | Bad | 5  10  15 | | | Shewhart(e)  MFCUSUM(e) MFCUSUM(e) | | 7.78  4.56  3.10 | |
| Set 1. flask | | Ambiguous | | | 5  10  15 | MaxCUSUM(a) MFCUSUM(e) Shewhart(e) | | 0.00  0.00  1.64 | Set 1. table | | Bad | 5  10  15 | | | MFCUSUM(e) MFCUSUM(e) MaxCUSUM(a) | | 5.22  5.33  5.57 | |
| Black chanterelle | Set 2. flask | Ambiguous | | | 5  10  15 | MaxCUSUM(a) MFCUSUM(e)  MFCUSUM(a) | | 2.14  0.43  2.14 | Set 2. table | | Bad | 5  10  15 | | | MaxCUSUM(e) CUSUM(e)  MFCUSUM(e) | | | 11.00 14.64 13.82 |
| Set 3. flask | Ambiguous | | | 5  10  15 | MaxCUSUM(a) MFCUSUM(e)  MFCUSUM(e) | | 2.00  2.00  2.00 | Set 3. table | | Bad | 5  10  15 | | | Shewhart(e)  MFCUSUM(e) MFCUSUM(e) | | | 7.88  6.78  5.80 |
| Set 4. flask | Ambiguous | | | 5  10  15 | MaxCUSUM(a) MFCUSUM(e)  MFCUSUM(a) | | 1.21  1.21  1.36 | Set 4. table | | Bad | 5  10  15 | | | Shewhart(e)  Shewhart(e)  MaxCUSUM(a) | | | 4.50  2.90  2.86 |
| Set 5. flask | | Ambiguous | | | 5  10  15 | Shewhart(e)  MFCUSUM(e) MFCUSUM(a) | | 3.43  0.43  2.14 | Set 5. table | | Bad | 5  10  15 | | | Shewhart(e)  MFCUSUM(e) MaxCUSUM(a) | | | 15.11 11.70 3.64 |
| Set 1. flask | | Ambiguous | | | 5  10  15 | MFCUSUM(e) MFCUSUM(e) MFCUSUM(a) | | 1.29  1.57  1.00 | Set 1. table | | Bad | 5  10  15 | | | Shewhart(e)  MFCUSUM(e) MFCUSUM(e) | | 8.40  4.40  4.00 | |
| Yellowfoot | Set 2. flask | Good | 5  10  15 | | | Shewhart(e)  Shewhart(e)  MFCUSUM(a) | | 2.14  2.21  1.43 | Set 2. table | | Ambiguous | | | 5  10  15 | Shewhart(e)  MFCUSUM(e) Shewhart(e) | | | 4.18  0.91  2.27 |
| Set 3. flask | Good | 5  10  15 | | | MaxCUSUM(a) MaxCUSUM(e) MaxCUSUM(a) | | 1.93  0.00  1.07 | Set 3. table | | Bad | | | 5  10  15 | Shewhart(e)  MaxCUSUM(e) MaxCUSUM(a) | | | 9.20  0.00  7.64 |
| Set 4. flask | | Good | 5  10  15 | | | MaxCUSUM(e) MaxCUSUM(a) MFCUSUM(a) | | 0.00  2.64  1.64 | Set 4. table | | Bad | 5  10  15 | | | Shewhart(e)  MFCUSUM(e) MFCUSUM(a) | | 4.00  2.18  1.43 | |
| Set 5. flask | | Good | 5  10  15 | | | Shewhart(e)  MFCUSUM(e) MFCUSUM(a) | | 2.71  2.29  1.29 | Set 5. table | | Bad | 5  10  15 | | | MFCUSUM(e) MFCUSUM(e) MFCUSUM(e) | | 5.44  2.67  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

9

*A. Rauhameri et al.*  *Array 14 (2022) 100151*

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 detec-tions 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 prob-lem 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 Rauhameri:**  Writing – original draft, Methodology, Soft-ware, 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:** Writ-ing – 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.

**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>[.](http://dx.doi.org/10.1007/s10115-016-0987-z)

10