Noname manuscript No.
(will be inserted by the editor)

Applying Temporal Dependence to Detect Changes in Streaming Data

Quang-Huy Duong · Heri Ramampiaro · Kjetil Nørvåg

Received: date / Accepted: 2018-07-11

Abstract Detection of changes in streaming data is an important mining task, with a wide range of real-life applications. Numerous algorithms have been proposed to efficiently detect changes in streaming data. However, the limitation of existing algorithms is that they assume that data are generated independently. In particular, temporal dependencies of data in a stream are still not thoroughly studied. Motivated by this, in this work we propose a new efficient method to detect changes in streaming data by exploring the temporal dependencies of data in the stream. As part of this, we introduce a new statistical model called the candidate change point (CCP) model, with which the main idea is to compute the probabilities of finding change points in the stream. The computed probabilities are used to generate a distribution, which is, in turn, used in statistical hypothesis tests to determine the candidate changes. We use the CCP model to develop a new algorithm called Candidate Change Point Detector (CCPD), which detects change points in linear time, and is thus applicable for real-time applications. Our extensive experimental evaluation demonstrates the efficiency and the feasibility of our approach.

Keywords Data Streams, Change Detection, Temporal Dependence, Adaptive Estimation

Quang-Huy Duong
huydqyb@gmail.com

Heri Ramampiaro
heri@ntnu.no

Kjetil Nørvåg
noervaag@ntnu.no

Department of Computer Science, Norwegian University of Science and Technology, Norway

1 Introduction

The availability of modern technology and the proliferation of mobile devices and sensors have resulted in a tremendous amount of streaming data. Due to its broad real-life applications, including consumption data (electricity, food, oil), finance and stock exchanges, healthcare, and intrusion/fraud detection, detection of changes in streaming data is an important data mining task. A main challenge with mining streaming data is that data in a stream is inherently dynamic, and its underlying distribution can change and evolve over time, leading to what is referred to as *concept drift* [40]. As an example, in consumption data, we might have data about customer purchasing behavior over time that could be influenced by the strength of the economy. In this case, the concept is the strength of the economy, which can drift. A concept drift can be either a real or virtual concept drift, where changes in the distribution can, in turn, have four different forms [21] namely *abrupt*, *reoccurring*, *incremental*, and *gradual*, as illustrated in Fig. 1. In addition, we can have mixtures of these forms in streaming data.

What can be inferred from this is that a method developed for mining changes in streaming data has to take into account the different characteristics of concept drifts. In particular, the learning model has to be trained and adapted to the changes, and model parameters should be able to adapt themselves following the changes in the streams. Numerous algorithms have been proposed to detect changes in streaming data [19, 20]. Still, most existing algorithms have been built based on the assumption that streaming data have stable flows, and that they are arriving in the same distribution. In addition, they assume the data to be identically and independently distributed (i.i.d.). However, such an as-

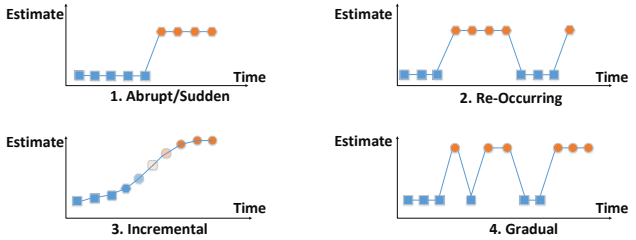

Fig. 1 Illustration of the four different types of concept drift [21].

sumption hardly holds in real-life streaming environments.

Focusing on non-stationary environments, several learning algorithms have been proposed to overcome the limitation of the i.i.d. assumption [27,29]. The majority of existing approaches have, however, assumed that data in a stream are independently but not identically distributed [10,45]. Adaptive estimation is among the proposed techniques for handling temporal dependencies of data. For example, in [3,11,32,37], the approaches employed adaptive estimation methodology by considering a so-called forgetting factor. Here, according to a decay function of the forgetting factor, the underlying assumption is that the importance of a data in a stream is inversely proportional to its age. As part of this, cumulative measures of the underlying distribution and estimators are maintained and monitored to detect changes, while new data continuously arrive.

Despite their efficiency with respect to detecting changes in data streams, the underlying assumption of the above methods is that data are generated independent of other data in the same stream. Meanwhile, several empirical experiments, e.g., [6,45], have shown that there are important temporal dependencies among data points in a stream of data, thus making it crucial for further studies, especially focusing on change detection. Motivated by this, the main goal of this work is to investigate the temporal dependencies of data points in a data stream, and use this to develop a novel method that enables monitoring changes in the stream while continuously estimating the underlying data distribution.

To summarize the principle behind the proposed method, Fig. 2 shows a block diagram of the important steps of the method. As shown in this figure, our approach takes a stream of data as input, and the main output is the list of candidate change points. In order to generate this list, the stream is processed in three main steps. The first step is to extract the distribution and determine the dependency information. To do this, we use the Euclidean projection method to generate projections of given data points in a stream onto k previously seen/processed data points. This produces a set of (probabilistic) paths between pairs of observed data points, called *trails*. In the second step, the mean values of the trails are estimated and used as inputs for the third and final step, in which the change detection is carried out through statistical hypothesis tests. Note that due to the dynamic nature of streaming data, each of the steps has to be efficient and be able to adapt to changes in the stream.

In this work, we make the following main contributions:

1. We introduce a new model named *Candidate Change Point (CCP)* that we use to model high-order temporal dependencies and data distribution in streaming data.
2. We develop a new concept called *CCP trail*, which is the path from a given observed data to another specific data in the observation history. Our approach uses the mean value of the CCP trail as a measure of data distribution, and to capture the temporal dependencies among observed data in the stream.
3. We develop a method that is able to handle the fact that data arrives in a high-velocity stream by providing continuously updating estimation factors as part of the CCP model.
4. We propose an efficient real-time algorithm, based on pivotal statistic tests for change detection named Candidate Change Point Detector (CCPD).
5. In order to demonstrate the feasibility, efficiency, and generality of our method, we conduct a thorough evaluation based on several real-life datasets and comparison with related approaches.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work. Section 3 introduces the problem of change detection. Section 4 presents the proposed model and the adaptive estimation method for detecting changes in evolving streaming data. Section 5 describes and discusses the results from our experimental evaluation. Finally, Section 6 concludes the paper and outlines possible topics for future work.

2 Related work

Change detection in streaming data has a wide-range of applications in many domains and has attracted lots of interests from the research community. To tackle the problem of change detection in streaming data, numerous methods and algorithms have been proposed [1,2, 20,22,23]. In the context of event detection in streaming data, due to the large volume of data, and that the data is dynamic and continuously arrives at a high

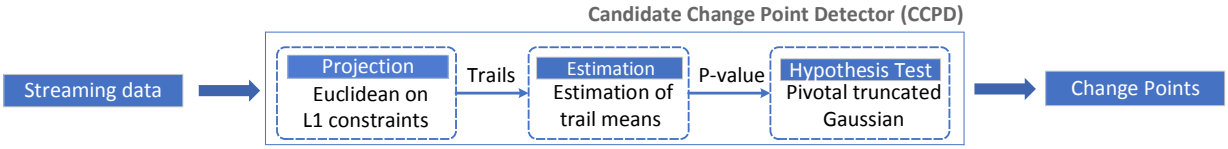

Fig. 2 Overview of the proposed method

speed, online learning approaches are necessary to be able to meet the induced challenges. As a result, several online learning algorithms have been proposed [17, 18]. However, an important requirement of online methods is to minimize the computational cost, and that approaches for online event/change detection must have constant time and space computational complexity for them to be feasible.

The algorithms for concept drift detection can be categorized into four main categories [21] (see also Table 1 for a summary). The first category consists of detectors that are based on *sequential analysis*. Cumulative Sum (CUSUM) and its variant, Page-Hinkley (PH) [32], are the representative algorithms in this category. Their main idea is to estimate the probability distribution value and update the value when new data comes. A concept drift occurs if there is significant change in the value of the estimated parameters. The main advantage with sequential analysis-based algorithms is that they generally have low memory consumption. One of the disadvantages, on the other hand, is that the performance depends on the choice of parameters. The second category includes detectors that are based on *statistical process control*, with which the idea is to estimate statistical information (such as error and standard deviation). Monitoring this error rate, the detectors determine a concept drift if the error increases above a pre-specified threshold value. Examples of methods in this group are DDM [19], EDDM [4], ECDD [39], AFF [11], and RDDM [5]. The main advantage with the methods in this category is that in addition of being efficient and having low memory consumption, they work generally well on streams with abrupt changes. However, slow gradual change detection is a one of the main weaknesses of the algorithms in this category. The third category comprises detectors that *monitor distributions on two different time windows*, i.e., applying statistical tests on the distributions of two windows in a stream. Here, the first window is a fixed reference window used to summarize the information from past data; whereas the second window is a sliding window summarizing the information of the most recent data. Examples of representative algorithms in this category are ADWIN [7], HDDM [18], SEQDRIFT2 [33], and MDDM [34]. The main advantage with the window-based methods are that they are able to provide more precise localization of change points. The disadvantages are the high memory cost and the space requirements for processing the aforementioned two windows. The fourth and final category consists of detectors that are *contextual-based*. The approaches in this category keep the balance of incremental learning and forgetting in detecting a concept drift with respect to the time of an estimated window. An example of contextual approaches is SPLICE [24]. The main advantage of methods in this category is that they can detect gradual and abrupt drifts, in addition to the ability to control the number of errors. However, contextual-based methods are generally complex and have long execution time, thus making them less suitable for mining of streaming data.

Hoeffding's Inequality [25] is one of the most well-known inequalities. It has been used to design several upper bounds for drift detection [18]. These upper bounds have been used in algorithms such as the Drift Detection Method Based on Hoeffding's Bounds (HDDM) [18], the Fast Hoeffding Drift Detection Method for Evolving Data Streams (FHDDM) [36], Hoeffding Adaptive Tree (HAT) [8], and the HAT + DDM + ADWIN [2] algorithm which extends the ADaptive sliding WINdow (ADWIN) algorithm [7], the Drift Detection Method (DDM) [19], and the Reactive Drift Detection Method (RDDM) [5], which is a recent drift detection algorithm based on DDM. Nevertheless, Hoeffding inequalities have the disadvantage of dropping the dependence on the underlying distribution [43]. Although these algorithms are useful to detect changes in streaming data, most of them assume that data is identically and independently distributed. However, as mentioned earlier, in real-life streaming data, data is inherently dynamic, and is not identically and independently distributed [10, 45]. Also, temporal dependencies are very common in data streams [45]. Thus, to address this issue, temporal dependencies in the streams should be considered.

Regarding temporal dependencies in data streams, adaptive models for detecting changes in the underlying data distribution have been proposed and extensively studied. The main idea of these approaches has been to estimate, maintain some interesting targets in the stream, and then compute the data distributions. The

Table 1 Summary of the related change detection methods

Category	Algorithms	Advantages	Disadvantages
Sequential analysis	CUSUM [32], PAGE-HINKLEY [32]	Low memory consumption.	Depends on the choice of parameters.
Statistical process control	DDM [19], EDDM [4], ECDD [39], AFF [11], RDDM [5]	Low memory consumption, work well on abrupt changes, and fast execution time.	Slow gradual changes.
Windows-based methods	Kifer et al. [26], ADWIN [7], HDDM [18], SEQDRIFT2 [33], FHDDM [36], ACWM [41], FHDDMs [35], MDDM [34]	Precise localization of change points.	Memory and space requirements for processing two windows.
Contextual approaches	SPLICE [24]	Gradual and abrupt drift, and control of the number of errors.	Complex and difficult to implement and long execution time.

main method used has been based on statistical hypothesis tests, where the hypothesis H_1 means change has been discovered, whereas the null hypothesis H_0 means no change. An example of such an approach has been suggested by Bodenham et al. [11]. They adopted a so-called forgetting factor, originally suggested by Anagnostopoulos et al. [3], to develop a new approach for continuously monitoring changes in a data stream, using adaptive estimation. They proposed an exponential forgetting factor method to decrease the importance of a data according to a decay function, which is inversely proportional to the age of the observed data.

Although Bodenham's approach enables monitoring changes in data streams by applying so-called Adaptive Forgetting Factor (AFF) [11], it does not address the challenges with temporal dependencies of data. Further, while AFF enables detecting multiple data point changes with adaptive estimation, the approach presented in this paper focuses on temporal dependencies of data, in addition to supporting online change detection in a data stream.

In conclusion, different from previous approaches, the proposed method in this paper generalizes the concept of temporal dependency in streaming data, by supporting both the first-order and the higher-order of dependencies. To achieve this, as discussed later in this paper, we propose a probabilistic method that can be used to analyze the differences between a data point at a given observation point and k previously observed points in the stream.

3 Problem Definition

Formally, we define the problem of change detection in streaming data by the following definition. Let there be a data stream S, which is an open-ended sequence of values $\{v_1, v_2, \ldots, v_i, \ldots\}$. Assume that our observation V_i of v_i in the data stream is drawn from a Gaussian model with an unknown distribution, $V_i \sim \mathcal{N}(\mu, \sigma^2 I)$, where μ is the (unknown) mean and is the interest measure in our change detection method, and σ^2 is the error variance. At each observation time i, the observed mean is μ_i. Hence, for a data stream S, we have a sequence of observed mean μ_1, μ_2, ..., μ_i, ... corresponding to observation times 1, 2, ..., i, ... Our change detection method considers the difference between adjacent means μ_i and μ_{i+1}, where not all adjacent means are not necessarily equal. The change detection method is designed to detect all change points i between two observed adjacent means μ_i and μ_{i+1}, for any i and $\mu_i \neq \mu_{i+1}$. Assume that $t_1, t_2, \ldots t_j, \ldots$ are the true change points in a distribution of means. Then, the change detector verifies a change point t_j by testing the following statistical hypotheses:

$$H_0 : \mu_{t_{j-1}} \simeq \cdots \simeq \mu_{t_j-1} \simeq \mu_{t_j} \simeq \mu_{t_j+1} \simeq \cdots \simeq \mu_{t_{j+1}-1}$$

against

$$H_1 : \mu_{t_{j-1}} \simeq \cdots \simeq \mu_{t_j-1} \neq \mu_{t_j} \simeq \mu_{t_j+1} \simeq \cdots \simeq \mu_{t_{j+1}-1}$$

Given a significance confidence ρ, the rule to accept the H_1 hypothesis is if an objective interest measure of mean, i.e., $f(.)$, satisfies:

$$f(\mu_{t_j}) \notin [v_{\frac{\rho}{2}}, v_{1-\frac{\rho}{2}}], \tag{1}$$

where $v_{\frac{\rho}{2}}$ and $v_{1-\frac{\rho}{2}}$ are values such that $Pr(f(\mu_{t_j}) < v_{\frac{\rho}{2}}) = \frac{\rho}{2}$ and $Pr(f(\mu_{t_j}) > v_{1-\frac{\rho}{2}}) = 1 - \frac{\rho}{2}$, and $f(.)$ is an objective function. A change point is said to occur at an observation time t_j if H_0 is rejected. This means that the task of detecting change points is to find all the observation times where there are significant differences in the data distribution for a given measure.

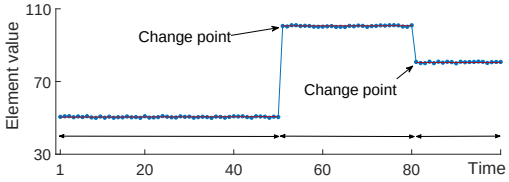

Fig. 3 An example a stream of data with two change points.

To illustrate, consider a stream of data shown in Fig. 3, which can be modeled using a piece-wise function with different values depending on the time intervals, $[1, 50]$, $[51, 80]$ and after 81. The estimated mean values are controlled/computed by the function to be 50 in the first interval, 100 in the second and then 80. In conclusion, as also can be observed, significant changes in the stream appear at timestamps 51 and 81.

To automatically detect change points, we propose an approach called Candidate Change Point (CCP) detection, which can be summarized as follows. First, we investigate the temporal dependence of each data to its k previous data points. We project a vector of that k data onto a ℓ_1 constraints to optimize the minimum difference between k previous observations and the current observation. The noisy or decay factor that weighs the importance of previous data on the current observation is inversely proportional to its age. Moreover, our approach employs truncated form of the geometric distribution to simulate the decaying probability of looking back into the history. Second, an adaptive estimator on CCP trail and CCP propagation is maintained online during the monitoring process of the stream (to be described in more detail in Subsection 4.2). Finally, our approach uses a statistic test on interest measure of mean values to determine if there is a change point at a given observation time. Specifically, a p-value of a mean and a pivotal statistic function are used to map the problem into a uniform Gaussian model on the interval [0, 1]. Then we use this to perform the truncated Gaussian statistic to test the hypotheses in order to decide change point occurrences in real-time.

4 The Candidate Change Point (Detection) Approach

In this section, we propose a new model to represent the temporal dependency of the current observation to its history in the data stream to detect changes. As mentioned in Section 1, most existing work has assumed the streaming data to be identically and independently distributed (i.i.d.). However, focusing on real-life applications, this assumption is too restrictive. In fact, the probability of the current observation is largely depended on previous data and the history of the stream. With this in mind, we consider Markov chain [31] as the natural choice for modelling streaming data, since a Markov process can be used to represent the probability of transitions between states of data in a stream. However, note that the space requirement for maintaining parameters with a higher-order Markov process grows exponentially as functions of the number of parameters. Thus, approaches employing higher-order Markov processes have a cubic space complexity, and are for this reason inefficient. To cope with this issue, more efficient parameterization approaches, such as the Linear Additive Markov Process (LAMP [28]) and the Retrospective Higher-Order Markov Process (RHOMP [44]), have been proposed. In contrast to the higher-order Markov process, the number of maintained parameters in the LAMP model and the RHOMP model grow linearly, which makes them more suitable for streaming data.

In this work, we build on the ideas of these linear Markov process models to develop an efficient model for temporal dependency in evolving data streams, and use the developed model for detecting concept drifts. We do this based on the fact that the probability of a data to appear in a stream at a specific time is not dependent on the first-order data only. This means that the appearance of data in specific observations can be assumed to be a cause of k previous observations. To be more specific, we propose a novel model, called the candidate change point (CCP) model, for detecting concept drifts. This is done by using the temporal dependency information from previously observed data in the current observation to compute the probability of finding changes in the given observation. What this implies is that with a first-order CCP model, any data that is observed at a specific time has only temporal dependency from the most recent previously observed data in the stream. On the other hand, if the current data point depends on its k previous data points and is a mix of these k data points, then it has a CCP from k-order ancestors. In such a case, the model that we apply is the k-order Candidate Change Point (CCP) model, defined as follows.

Definition 1 (k-order Candidate Change Point) Given a stream of observations with an open end x_1, x_2, ... , x_n, ..., the k-order Candidate Change Point is denoted as CCP_k, and at observation time t, the CCP model is presented as $CCP_k(x_t) = (C_1, C_2, \ldots, C_k)$, where $0 \leq C_i \leq 1$ for $i = 1, 2, \ldots, k$, $\sum_{i=1}^{k} C_i = 1$. We call C_i is the probability proportion of CCP in current descendant obtained from i-th order in its k ancestors.

Definition 2 (k-order Fading Candidate Change Point) Given a stream of observations with an open

end $x_1, x_2, \ldots, x_n, \ldots$, the k-order Fading Candidate Change Point is the k-order Candidate Change Point considering forgetting factor α of ancestors by order denoted as $CCP_{k,\alpha}$. At observation time t, the CCP model is presented as $CCP_{k,\alpha}(x_t) = (\alpha_1 \times C_1, \alpha_2 \times C_2, \ldots, \alpha_k \times C_k)$, where $0 \leq \alpha_i \leq 1$ for $i = 1, 2, \ldots, k$, $\sum_{i=1}^{k} \alpha_i = 1$, $0 \leq C_i \leq 1$ for $i = 1, 2, \ldots, k$, $\sum_{i=1}^{k} C_i = 1$.

We call α_i a decaying probability of CCP in current descendant from i-th ancestor in its k ancestors into the history context. Given a scalar x, we denote $\overrightarrow{x_{CCP_k}}$ as a vector of x projected on its k-order ancestors. We have $\overrightarrow{x_{CCP_k}} = (C_1, C_2, \ldots, C_k)$, it is the k-order Candidate Change Point of the scalar. In the rest of the paper, we use a bold face letters ($\mathbf{x}$) to present the vector for short ($\overrightarrow{x_{CCP_k}}$). Suppose $\mathcal{L}$ be a cost function, the value and option of cost function $\mathcal{L}$ will be discussed in the next section. In this paper, our purpose is to solve the minimum of cost function subject to conditions ℓ_1-norm constraint on k-order Candidate Change Point. The problem is described as:

$$\underset{CCP_k}{minimize}(\mathcal{L}(CCP_k)), \tag{2}$$

subject to $0 \leq C_i \leq 1, \forall i = 1, 2, \ldots, k, \sum_{i=1}^{k} C_i = 1$.

Projected (sub)gradient methods minimize an objective function $f(x)$ subject to the constraint that x belongs to a convex set $\mathcal{CS}$. The constrained convex optimization problem is $minimize f(x)$ subject to $x \in \mathcal{CS}$.

The projected (sub)gradient method is given by generating the sequence $x^{(t)}$ via:

$$x^{(t+1)} = \prod_{\mathcal{CS}}(x^{(t)} - \gamma_t \times g^{(t)}), \tag{3}$$

where $\prod_{\mathcal{CS}}$ is a projection on $\mathcal{CS}$, γ_t is step size, $x^{(t)}$ is the t-th iteration, and $g^{(t)}$ is any (sub)gradient of f at $x^{(t)}$, and will be denoted as $\partial f(x)/\partial x^{(t)}$.

4.1 Evolving Data and CCP Parameter Selection

Given a data stream where data evolves over time, i.e., its population distribution or its structure changes over time, the goal is to maximize the probability proportion of CCP for current observation in the set of the k last/previously observed data by solving an optimal problem, using the projected (sub)gradient method with an optimal function to minimize the divergence of the currently observed data when it is projected onto ℓ_1-norm constraints of k previous ancestors. Here, the cost function $f(x)$ on Euclidean projections can be chosen as Euclidean norm (ℓ_2 norm) $\mathcal{L}(x) = \frac{1}{2} \parallel x - v \parallel^2$.

The Euclidean projection in this paper is to project a streaming data x^t onto a set of k previously observed data, defined as:

$$\prod_{CCP_k}(x^t) = \underset{\mathbf{x} \in CCP_k}{\arg\min} \frac{1}{2} \parallel \mathbf{x} - \mathbf{x}^t \parallel^2, \tag{4}$$

subject to $\parallel \mathbf{x} \parallel_1 = \sum_{i=1}^{k} C_i = 1$. The optimal solution for this problem can be solved in linear time as in [14,16,30]. The Lagrangian of Equation 4 with Karush Kuhn Tucker (KKT) multiplier ζ and μ is:

$$\mathcal{L}(x, \zeta) = \frac{1}{2} \parallel \mathbf{x} - \mathbf{x}^t \parallel^2 + \zeta(\parallel \mathbf{x} \parallel_1 - 1) - \mu\mathbf{x} \tag{5}$$

Derivation of Lagrangian function at x_i with optimal solution x^* by dimension i is $\frac{\partial \mathcal{L}}{\partial x_i}(x^*) = (x_i^* - x_i^t) + \zeta - \mu_i$. Because x^* is an optimal point then $\frac{\partial \mathcal{L}}{\partial x_i}(x^*) = 0 \rightarrow x_i^* = x_i^t - \zeta + \mu_i$. The KKT inequalities in the problem is that $x_i \geq 0$, then by the complementary slackness, we have $\mu_i = 0$, hence $x_i^* = max\,(x_i^t - \zeta, 0)$. Moreover, we project our vector using a fast and linear method as in [14]. In particular, we consider a vector of the last k data points as the vector of current observation with k elements. This k elements vector will be projected onto ℓ_1 ball constraints by using (sub)gradient method.

Definition 3 (CCP Heritage) Given a k-order Candidate Change Point with forgetting factor α, the CCP model is $CCP_{k,\alpha}(x_t) = (\alpha_1 \times C_1, \alpha_2 \times C_2, \ldots, \alpha_k \times C_k)$, where $0 \leq \alpha_i \leq 1$ for $i = 1, 2, \ldots, k$, $\sum_{i=1}^{k} \alpha_i = 1$, $0 \leq C_i \leq 1$ for $i = 1, 2, \ldots, k$, and $\sum_{i=1}^{k} C_i = 1$. The CCP Heritage of the model at time t is denoted as $CH_{k,\alpha}(x_t)$ and computed as $CH_{k,\alpha}(x_t) = \sum_{i=1}^{k} \alpha_i \times C_i$. This value of the scalar can be used to present the temporal dependency proportion of the scalar to history. For the sake of brevity, the notation ch_t will be used to refer to the CCP heritage value at observation time t in the rest of this paper. The sequence of CCP heritage of the stream is denoted $ch_1, ch_2, \ldots, ch_t, \ldots$.

In this work, the key idea is to investigate how to exploit the temporal dependencies of data for detection of changes in a stream. The method we propose is inspired by the ideas behind linear Markov process models with which the main principle is to estimate the probability of transition between states, i.e., to determine the likelihood of each state transition. Nevertheless, the main difference between our approach and previous Markov-based approaches is that our approach considers the temporal dependencies based on several prior observations. In addition, the proposed model enables estimating the dependency measures in a data stream in an online fashion, thus making it possible to detect changes in the stream in real time.

4.2 Adaptive Estimation of Data in Streaming

Adaptive estimation approaches were previously proposed to handle issues with the uncertainty of data. In streaming data, the importance of historical data is weighted using a forgetting function with a decay factor [3,11,41]. This means that, in the stream, the most recent data is more important than the older ones. A decay function is also useful to flush any noise when detecting concept drifts.

There are several approaches to select the decay factor α. Once the value can be set to constant, then we can conduct trial experiments and obtain the optimal value for individual application. The factor can be any function of exponential family of distribution. In [11], the authors proposed adaptive forgetting factor to weigh the importance of historical data by solving an optimal problem in movement of mean. The estimator is continuously monitored when new data arrives to detect changes in the data stream. The principle behind building forgetting factor is to build an exponential decay function of observation time such that the importance of a data in a stream is inversely proportional to its age, and the temporal dependencies can be seen is just 1-order dependency, which is a special case, with k=1 and $\alpha = 1$. In the proposed method, we introduce an adaptive estimation for detecting changes, we use CCP heritage of the CCP model to estimate the CCP mean distribution of streaming data. This method can be compared to the linear high order of states based on Markov chain [28,44].

To adapt the evolving factor in streaming data, we introduce CCP trail to denote the CCP path from a given observation to another previous observation in the streaming data. Hence, a CCP trail can be considered as the probability of finding the heritage from a data stream. This is formally defined as follows.

Definition 4 (CCP Trail) Given two different observation times t_1 and t_2, $0 < t_1 \leq t_2$, of a streaming data S, which is an open-ended sequence of values $\{v_1, v_2, \ldots, v_i, \ldots\}$. The CCP trail between two given observations is the probability of finding the heritage of the data at observation time t_1 in the data at observation time t_2. A CCP trail is denoted as $ct(t_2, t_1)$, and is formally defined as:

$$ct(t_2,t_1) = \begin{cases} 1 & \text{if } t_2 = t_1 \\ ch_{t_2} & \text{if } t_2 = t_1 + 1 \\ ct(t_2, t_2-1) \times ct(t_2-1, t_1) & \text{otherwise.} \end{cases} \tag{6}$$

Property 1 The CCP trail can be computed by as the product of CCP heritages for all $t_1 \leq t_2$ as follows:

$$ct(t_2,t_1) = \prod_{t=t_1}^{t_2} (1_{\{t \neq t_1\}} \times ch_t + 1_{\{t=t_1\}}), \tag{7}$$

where $1_{\{x\}}$ is binary indicator function. In other words, $1_{\{x\}}$ is equal to 1 if x is TRUE, otherwise $1_{\{x\}}$ is equal to 0.

$$1_{\{x\}} = \begin{cases} 1 & \text{if x is TRUE} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Proof The detailed proof is provided in Appendix A.1

CCP trail mean at observation time t in the data stream is then defined by:

$$\overline{CCP(t)} = \frac{1}{\sum_{i=1}^{t} ct(t,i)} \sum_{i=1}^{t} v_i \times ct(t,i) = \frac{cp(t)}{ctsum(t)}, \tag{9}$$

where $cp(t) = \sum_{i=1}^{t} v_i \times ct(t,i)$, $ctsum(t) = \sum_{i=1}^{t} ct(t,i)$.

$cp(t)$ is the CCP propagation at observation time t looking back into its history in the data stream. $ctsum(t)$ is the coefficient presenting sum of the CCP trail at time t.

Proposition 1 *Given a data stream, $ctsum(t)$ can be computed by the following equation:*

$$ctsum(t) = ctsum(t-1) \times ch_t + ct(t,t). \tag{10}$$

Proof The detailed proof is provided in Appendix A.2

Proposition 2 *Similar to the coefficient estimation, the CCP propagation can be estimated by the following sequential updating:*

$$cp(t) = cp(t-1) \times ch_t + v_t. \tag{11}$$

Proof The detailed proof is provided in Appendix A.3

Equations 10-11 show that we can sequentially update the CCP model, CCP coefficient, and CCP propagation in the stream at each observation time, while new data arrives. Hence, the *CCP trail mean* at the next observation, $\overline{CCP(t+1)}$, is easily estimated in linear time by Equation 9.

4.3 Change Detection

Change point detection relies on the null hypothesis H_0 and the alternative hypothesis H_1. The null hypothesis H_0 is a hypothesis that assumes the population means are drawn from the same distribution while the alternative hypothesis H_1 supposes the observations are from the different distribution. The change detector defines a rule to accept H_1 and reject H_0. When H_0 is rejected, it means that there is a significant movement in underlying distribution of data, and change point occurs. In our approach, the detector monitors the movement in CCP trail mean in an online manner.

Given a random variable v, we consider the Gaussian model with mean θ, variance σ^2, and assume that $v \sim \mathcal{N}(\theta, \tau^2 = \sigma^2 I)$. The cumulative distribution function [42] for a linear contrast $p^T\theta$ of mean θ is as follows:

$$\mathcal{D}^{[v_1,v_2]}_{p^T\theta,\tau^2}(p^T v)|Gaussian \sim Uniform(0,1), \tag{12}$$

where $[v_1, v_2]$ is the boundary interval of the Gaussian model and $\mathcal{D}$ is pivotal statistic function. The pivotal statistic function $\mathcal{D}$ is defined and computed as:

$$\mathcal{D}^{[v_1,v_2]}_{\mu,\tau^2}(x) = \frac{CDF(\frac{(x-\mu)}{\tau}) - CDF(\frac{(v_1-\mu)}{\tau})}{CDF(\frac{(v_2-\mu)}{\tau}) - CDF(\frac{(v_1-\mu)}{\tau})}, \tag{13}$$

where *CDF(.)* is a standard normal cumulative distribution function, and in our proposed method we use the standard normal right tail probabilities as in [12] due to its simple form, and it has a very small error. We use the truncated Gaussian pivot [42] to test hypothesis H_0 with an assumption that the population distribution is equal to zero, $H_0 : p^T\theta = 0$. The alternative positive hypothesis is $H_{1+} : p^T\theta > 0$. The truncated Gaussian statistic then is computed by: $\mathcal{T} = 1 - \mathcal{D}^{[v_1,v_2]}_{0,\tau^2}(p^T v)$. Given a confidence value $0 \leq \rho \leq 1$, in our change detector, we find the v_ρ satisfying $1 - \mathcal{D}^{[v_1,v_2]}_{v_\rho,\tau^2}(p^T v) = \rho \rightarrow \mathbb{P}(p^T\theta \geq v_\rho) = 1 - \rho$. Similar to the two-sided test, we compute the confidence interval $[v_{\frac{\rho}{2}}, v_{1-\frac{\rho}{2}}]$. $v_{\frac{\rho}{2}}$ and $v_{1-\frac{\rho}{2}}$ are computed based on conditions such that:

$$1 - \mathcal{D}^{[v_1,v_2]}_{v_{\frac{\rho}{2}},\tau^2}(p^T v) = \frac{\rho}{2}, \tag{14}$$

$$1 - \mathcal{D}^{[v_1,v_2]}_{v_{1-\frac{\rho}{2}},\tau^2}(p^T v) = 1 - \frac{\rho}{2}. \tag{15}$$

Then we have $\mathbb{P}(v_{\frac{\rho}{2}} \leq p^T\theta \leq v_{1-\frac{\rho}{2}}) = 1 - \rho$. Change is identified with a confidence ρ if the pivotal population mean does not lie in the interval $[v_{\frac{\rho}{2}}, v_{1-\frac{\rho}{2}}]$.

4.4 Choosing Decay Factor

In our method, we consider forgetting factor α of k previous ancestors by order to the current model. The meaning of α is similar to the decaying probability of looking back into the history in [44]. To select the parameter α, we use truncated form of the geometric distribution [13]. The truncated form of the geometric distribution is presented by parameter η, subject to $0 < \eta \leq 1$, and k terms (ancestors). The probability density function at term $i, 1 \leq i \leq k$, is defined by $P(i) = \frac{\eta(1-\eta)^{i-1}}{1-(1-\eta)^k}$. We set this probability density at term i as our forgetting factor α_i of the i-th ancestor,

$$\alpha_1 = \frac{\eta}{1-(1-\eta)^k},\ \alpha_2 = \frac{\eta(1-\eta)}{1-(1-\eta)^k},\ \ldots,\ \alpha_k = \frac{\eta(1-\eta)^{k-1}}{1-(1-\eta)^k}.$$

Observe that the truncated form of the geometric distribution subjects to the condition that sum of probability of k terms equals to 1. In other words, it subjects to $\sum_{i=1}^{k} \alpha_i = 1$.

$$\begin{aligned}
\sum_{i=1}^{k} \alpha_i &= \frac{\eta}{1-(1-\eta)^k} + \frac{\eta(1-\eta)}{1-(1-\eta)^k} + \cdots + \frac{\eta(1-\eta)^{k-1}}{1-(1-\eta)^k} \\
&= \frac{\eta}{1-(1-\eta)^k}(1 + (1-\eta) + \cdots + (1-\eta)^{k-1}) \\
&= \frac{(1-(1-\eta))(1 + (1-\eta) + \cdots + (1-\eta)^{k-1})}{1-(1-\eta)^k} \\
&= \frac{1-(1-\eta)^k}{1-(1-\eta)^k} = 1.
\end{aligned}$$

Furthermore, the condition $0 \leq \alpha_i \leq 1$ is also satisfied. The value of η is application-specific and can be chosen either by a procedure of polynomial interpolation or using a heuristic optimal as in [44].

4.5 The Online Change Detection Algorithm

Fig. 4 gives an overview of the CCPD algorithm, and our method for change detection is presented in Algorithm 1. The input of the algorithm is a sequence of open end values v_1, v_2, ..., v_i, ..., and a confidence value ρ. When a new value v_i is read from the stream, a linear projection onto ℓ_1 ball constraints is performed to project a vector of the k last data values to get CCP_k of the current observation in line 3 (see [14] for more detailed information about the projection method). Line 4 computes ch_i using the truncated form of the geometric distribution. Line 5 is executed to estimate $ctsum(i)$ and $cp(i)$. Then CCP trail mean $\overline{CCP(i)}$ is computed by line 6. In line 7, the boundary interval with confidence ρ is calculated using the pivotal statistic function. The pivotal truncated Gaussian value of the current data is specified in line 8. A check is performed in line 9 to determine there is a change or not. If a change occurs, the estimators are reset and the algorithm outputs that change.

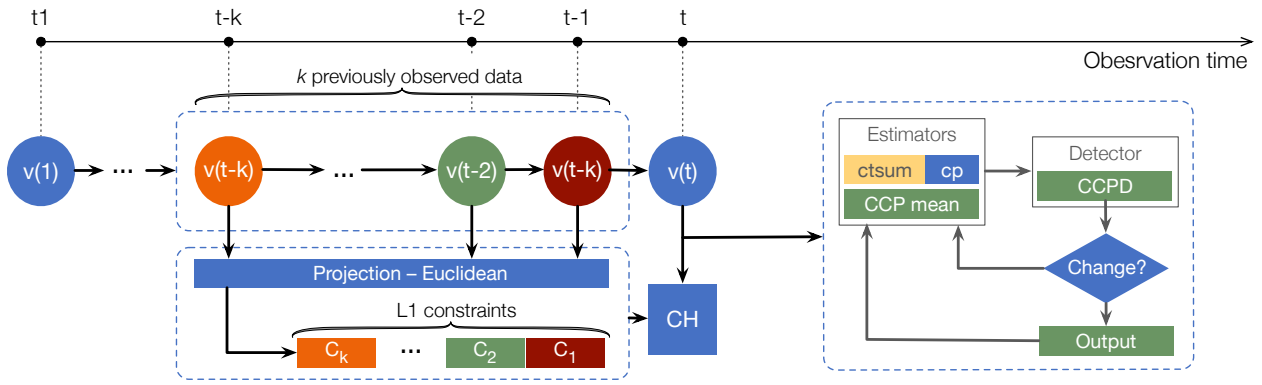

Fig. 4 Flow diagram of the CCPD algorithm

Algorithm 1 CCPD: Online change detector

Input: Streaming data and a confidence ρ.
Output: Changes in the stream.

1: *Init*() Initialize all variables
2: **for** (each data point v_i arriving on a stream v_1, v_2, ..., v_i, ...) **do**
3: Project the k last data points on ℓ_1 constraints $Projection(v_{i-1}, \ldots, v_{i-k}) \rightarrow CCP_k(v_i)$ as in [14]
4: Compute $ch_i = CH_{k,\alpha}(v_i) = \sum_{i=1}^{k} \alpha_i \times C_i$
5: Update estimators $ctsum(i)$ and $cp(i)$ by Equations 10 and 11
6: Update CCP trail mean $\overline{CCP(i)}$ by Equation 9
7: Compute the confidence interval $tailarea = [v_{\frac{\rho}{2}}, v_{1-\frac{\rho}{2}}]$ such that 14 - 15 satisfy
8: Compute truncated Gaussian pivot of the current observation $pval(i)$
9: **if** $(pval(i) \notin tailarea))$ **then**
10: Output change
11: Reset Estimator
12: **end if**
13: **end for**

An illustrative example. Let us consider a sample synthetic data stream presented in Fig. 3. The stream contains 100 elements with significant changes appearing at timestamps 51 and 81. Assume our parameter $k = 3$. In this example, we include the last data in the projection. Considering the observation at timestamp 50, the values of estimated parameters at timestamp 50 are as follows: ch=0.487, $ct(ctsum)$=1.493, cp=75.372, and $\overline{CCP}$ is computed as $\frac{cp}{ct} = \frac{75.372}{1.493} = 50.484$. Assume that the incoming element in the stream at timestamp 50 is 50.45 and the three latest elements in the stream are sequentially 50.45, 50.66, and 50.08. The processing steps of the proposed method are the following:

Step 1: Project a vector of the last 3 elements (50.45, 50.66, 50.08) on ℓ_1 constraints.
The result of this projection is (0.39, 0.59, 0.02).
Step 2: Calculate the value of CCP Heritage, ch.
Assume the parameter η is set to 0.98, the decay factors $\alpha_i, i = 1, \ldots, 3$, are computed to be (0.98, 0.0196, 0.0004), and the new value of ch is $0.39 * 0.98 + 0.59 * 0.0196 + 0.02 * 0.0004 = 0.394$.
Step 3: Calculate values of estimators ct and cp.
$cp = 75.372 * 0.394 + 50.45 = 80.147$,
$ct = 1.493 * 0.394 + 1 = 1.588$.
Step 4: Update mean $\overline{CCP} = \frac{cp}{ct} = \frac{80.147}{1.588} = 50.470$.
Step 5: Compute the pivotal truncated Gaussian of the new mean, and the confidence interval of the old mean with a confidence $\rho = 0.01$.
We have $pval(\overline{CCP}) = 1.0$, and $tailarea = [v_{\frac{\rho}{2}}, v_{1-\frac{\rho}{2}}] = [0.0, 1.0]$.
Step 6: Check condition $pval(i) \notin tailarea$.
Because $1.0 \in [0.0, 1.0]$, the detector determines that there is no change at timestamp 50.
Step 7: A new data in the stream at timestamp 51 can now be processed. At the timestamp the data value is 100.56, so the last 3 elements in the stream are (100.56, 50.45, 50.66). Repeat Step 1 to Step 6 to process the rest of the stream.

When new data comes in the stream, if a change is detected, the values of the estimators are reset to default. In our example, with the sample synthetic dataset, the CCPD detects two change points at timestamps 52 and 82 (1 delay in comparison with two true change points at timestamps 51 and 81).

Fig. 5 and Fig. 6 depict the visualization of the real values of the stream and the estimated mean values by the CCPD method, running on the sample synthetic streaming data shown in Fig. 3. In particular, Fig. 5 plots the real values and the estimated mean values computed using the CCPD method with two true change points at timestamps 51 and 81, and two detected change points at timestamps 52 and 82. Fig. 6 shows the estimated mean values while we vary the value of η in the decay factor. The importance degree of the current data is the combination of the projection vector (on k latest data) and the decay factors, which

are, in turn, affected by the choice of η. Because we chose decay factors in a form of geometric distribution, the most effect on the importance of current data can be derived from the most recent data in the stream. We observe that when the stream is stable, the value of the estimator is stable with different values of the decay factor. However, the obtained estimator significantly changes around the change point.

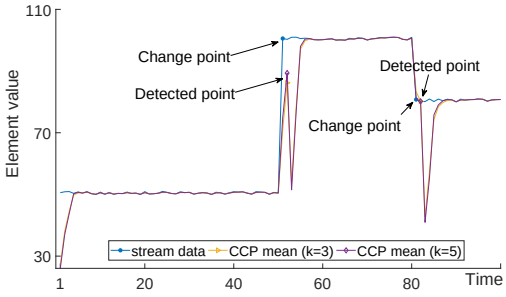

Fig. 5 A stream of data with two detected change points.

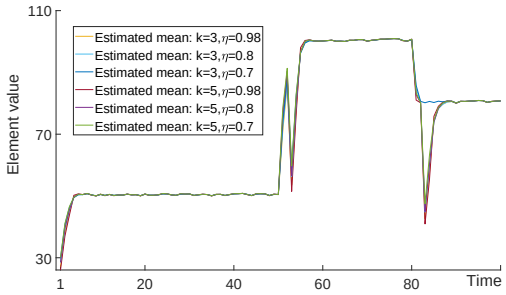

Fig. 6 Estimated mean while varying η

Complexity. The procedure of processing each arrival stream data point has three main parts. The first part is the Euclidean projection of the k last data points on a ℓ_1 ball constraints. This process is a fast projection [14] and has complexity $\mathcal{O}(k)$. Normally, k is small, thus the process can be performed in constant time. The second part is the process of calculating values of CCP, CCP mean, and updating estimators. The complexity of this process is constant $\mathcal{O}(1)$. The last part is a process which we calculate tail area and pivotal Gaussian value. We use a fixed number (100) of steps to search for the boundary of the tail. Hence, this process also has constant complexity $\mathcal{O}(1)$. In summary, the complexity of our method is $\mathcal{O}(k) + \mathcal{O}(1) + \mathcal{O}(1) = \mathcal{O}(k)$, thus the proposed algorithm CCPD has a constant complexity with constant k, $\mathcal{O}(k)$.

5 Evaluation

We have performed thorough experiments to evaluate the performance of our method and compare it to the state-of-the-art algorithms. To make our experiments as real and generic as possible, we performed our evaluation on several different real-world datasets, with various characteristics. Besides that, an extensive experiment was conducted to evaluate the flow rates of the stream including detection delay, true positive, true negative, false negative, and accuracy on several artificial datasets with known ground-truth. We carried out the experiments on a computer running the Windows 10 operating system, having a 64 bit Intel i7 2.6 GHz processor, and 16 GB of RAM. The proposed algorithm was implemented in Java. All evaluations are performed using the MOA framework [9][1], with our algorithm integrated.

5.1 Real-World Datasets

For our experiments, we used eight real-world datasets, that are widely-used as benchmark datasets for change-detection methods [7, 18, 36, 45], consisting of Electricity, Poker (Hand), Forest Covertype, Spam, Usenet1, Usenet2, Nursery, and EEG Eye State.

Electricity is an electricity consumption dataset collected from the Australian New South Wales Electricity Market. The dataset has 45,312 instances which contains electricity prices from 7 May 1996 to 5 December 1998. The instances were recorded by an interval every 30 minutes.

Poker-Hand is data of a hand consisting of five playing cards drawn from a standard deck of 52. The dataset contains 829,201 instances.

Forest Covertype is forest cover type for a given observation of 30 x 30 meter cells. The dataset was obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. The dataset is recorded in the Roosevelt National Forest of northern Colorado, US. *Forest Covertype* contains 581,012 instances.

Spam is a dataset based on the Spam Assassin collection and contains both spam and legitimate messages. The *Spam* dataset has 9,324 instances.

[1] Version 4.0.0, June 2017.

Usenet1 and *Usenet2* are based on the twenty newsgroups data set, which consists of 20,000 messages taken from twenty newsgroups. Each of *Usenet1* and *Usenet2* contains 1,500 instances obtained from twenty newsgroups data set to present a stream of messages of a user.

Nursery was derived from a hierarchical decision model originally developed to rank applications for nursery schools in Ljubljana, Slovenia. The dataset contains 12,960 instances.

EEG Eye State was originally used to predict eye states by measuring brain waves with an electroencephalographic (EEG), i.e., finding the correlation between eye states and brain activities [38]. It consists of 14,980 instances with 14 EEG values, where each value indicates the eye state.

Electricity, Forest Covertype and *Poker-Hand* were obtained from the most popular open source framework for data stream mining MOA[2], while the remaining datasets were obtained from the UCI Machine Learning Repository[3] (University of California, Irvine)

5.2 Performance

This section presents the experimental results of the proposed method compared to the state-of-the-art algorithms. We performed empirical experiments to evaluate our proposed algorithm and compared it against HDDM$_W$, HDDM$_A$ [18], DDM [19], EDDM [4], SeqDrift2 [33], FHDDM [36][4], and RDDM[5] [5]. These algorithms were chosen because they are the state-of-the art algorithms in drift detection (see also Section 2). Implementation of the algorithms are provided by the MOA framework. In all our experiments we used Natïve Bayes (NB) and Hoeffding Tree (HT) classifiers as the base learners, which are frequently used in the literature [7, 18, 33, 36]. In addition, we performed comparison with a baseline method, called "No Change Detection", that detects changes with base learners only. The confidence value is set to 0.001, η is set to 0.99. The λ in HDDM$_W$ is set to 0.05 as recommendation by the authors. The sliding window size is set to 25 in FHDDM. According to the authors, it is the optimal value providing the best classification accuracy. With the other algorithms, in all the tests we use the default parameters and configuration values as recommended by the authors of the original work. Accuracy evaluator is computed base on a window with size 100. Frequency of sampling process is every 100 samples. Furthermore, we prepare two versions, named $CCPD_{k=5}$ and $CCPD_{k=3}$. $CCPD_{k=5}$ is our proposed method running with a projection on the last five data points in the stream, while $CCPD_{k=3}$ runs with a projection on the last three data points in the stream.

Table 2-3 show the average ($\overline{a}$) and standard deviation (std) accuracy of the change detection algorithms with NB and HT classifiers as base learner, respectively. In case of same accuracy result, we consider the algorithm with the lowest standard deviation to be the best. The results show that CCPD wins 5 times on total 8 datasets with NB classifiers and 4 times on total 8 datasets with HT classifiers. Further, Table 4 shows that our proposed method obtains the best accuracy scores with the majority of the datasets (five out of eight datasets), including Electricity, Spam, Usenet1, Corvertype, and Poker datasets; while on EEG Eye, Nursery, and Usenet2, the accuracy ranks are 2, 4, and 6.5 (the same rank with FHDDM), respectively. Another observation from the experiments is that, while varying the order of temporal dependency k, the accuracy scores on Nursery and Usernet2 change abruptly. This can be explained as follows. On the Usernet2 dataset, the reason is that the concept drift is moderate and the topic shift on the dataset is blunt and blurred. Moreover, the number of samples in the dataset is small for the training. On Nursery, on the other hand, the change behavior may be due to the large number of distinct values of attributes in the dataset. Thus, the temporal dependency is loose, which has in turn an impact on detecting the changes. Nevertheless, the difference from the best accuracy is not significant. Overall, the empirical results show that CCPD has the best performance in all cases of combinations with NB and HT classifier.

To further evaluate the performance of the proposed method, and in order to get a fair comparison among the algorithms, we performed statistical significance tests based on the average rank of accuracy of the algorithms [15]. Firstly, we select the best accuracy of the algorithms with NB classifier and HT classifier and then report rank of accuracy of the algorithms. Secondly, we use Friedman test because it is a nonparametric analogue of the parametric two-way analysis of variance by ranks. Table 4 shows statistical test results using the methodology proposed in [15]. The number in parentheses at each cell is the rank of accuracy. The bold

[2] https://moa.cms.waikato.ac.nz/datasets/

[3] http://archive.ics.uci.edu/ml/datasets.html

[4] Source code of the FHDDM is provided by the authors of the algorithm

[5] Source code of the RDDM is obtained from the authors personal website

Table 2 Accuracy results (%) with Natïve Bayes (NB) classifier

Algorithm	Factor	Electricity (%)	Spam (%)	Usenet1 (%)	Usenet2 (%)	Covertype (%)	Nursery (%)	Poker (%)	EEG Eye (%)
$CCPD_{k=5}$	$\overline{a}$	85.19	92.65	75.67	61.20	88.98	83.46	79.96	99.45
	std	5.93	8.60	10.30	19.87	9.39	14.69	9.33	1.62
$CCPD_{k=3}$	$\overline{a}$	**86.20**	**92.88**	**78.27**	63.00	**89.18**	91.85	**80.42**	99.43
	std	**5.45**	**7.98**	**11.67**	21.81	**8.46**	9.20	**8.36**	1.68
$HDDM_{W-test(0.05)}$	$\overline{a}$	84.47	91.51	75.07	70.93	86.23	91.71	77.11	97.60
	std	6.57	7.39	11.91	13.79	8.07	7.61	8.80	5.13
$HDDM_{A-test}$	$\overline{a}$	85.09	90.67	75.20	71.00	87.44	92.51	76.48	98.33
	std	6.33	9.31	11.59	13.29	7.97	6.50	9.82	3.73
DDM	$\overline{a}$	82.70	89.50	73.73	72.93	88.03	91.72	61.97	**99.57**
	std	8.70	13.89	12.69	12.09	8.35	7.11	21.36	**1.19**
EDDM	$\overline{a}$	84.93	90.66	75.13	**73.27**	86.09	92.02	77.48	96.85
	std	6.23	8.60	12.32	**12.76**	8.66	6.77	8.40	7.26
SeqDrift2	$\overline{a}$	79.83	89.87	65.80	72.13	82.45	87.18	72.25	93.04
	std	11.50	13.38	23.51	11.54	12.31	8.84	14.29	17.05
RDDM	$\overline{a}$	84.88	91.45	74.73	73.07	86.87	**92.70**	76.67	99.06
	std	6.36	7.44	11.87	12.10	8.32	**6.10**	9.11	2.88
FHDDM	$\overline{a}$	83.32	91.36	74.73	71.20	85.10	89.09	76.68	97.28
	std	7.29	6.99	11.86	12.43	9.07	10.82	9.12	5.96
No Change Detection	$\overline{a}$	74.17	90.63	63.33	72.13	60.53	83.35	59.55	47.35
	std	14.69	10.93	23.64	11.54	21.76	14.94	21.96	46.67

Table 3 Accuracy results (%) with Hoeffding Tree (HT) classifier

Algorithm	Factor	Electricity (%)	Spam (%)	Usenet1 (%)	Usenet2 (%)	Covertype (%)	Nursery (%)	Poker (%)	EEG Eye (%)
$CCPD_{k=5}$	$\overline{a}$	83.49	92.67	74.47	57.20	89.44	72.22	79.15	99.62
	std	6.56	8.45	11.22	23.14	9.49	13.26	9.75	1.04
$CCPD_{k=3}$	$\overline{a}$	84.26	**92.98**	**75.53**	71.20	**89.51**	86.35	**79.53**	99.62
	std	6.21	**7.78**	**6.80**	12.31	**8.67**	14.82	**8.98**	1.04
$HDDM_{W-test(0.05)}$	$\overline{a}$	85.46	91.48	74.73	70.00	85.98	90.48	77.12	97.49
	std	6.91	7.85	8.41	9.00	8.23	6.66	8.93	4.94
$HDDM_{A-test}$	$\overline{a}$	**86.11**	91.66	74.60	68.87	87.27	90.93	76.40	98.89
	std	**6.45**	7.22	8.14	8.24	8.02	6.02	9.93	2.62
DDM	$\overline{a}$	85.83	91.94	73.00	69.80	82.58	91.31	72.74	**99.64**
	std	7.17	7.22	10.03	9.34	12.95	6.41	15.14	**1.04**
EDDM	$\overline{a}$	85.26	91.68	73.93	**72.00**	86.02	90.17	77.31	96.64
	std	6.66	7.88	8.96	**10.44**	8.46	6.79	8.68	7.33
SeqDrift2	$\overline{a}$	82.30	91.55	69.67	72.00	82.86	89.01	72.51	93.63
	std	11.12	9.23	20.38	11.01	12.09	6.83	13.89	16.42
RDDM	$\overline{a}$	85.83	92.09	74.27	69.60	86.43	**91.75**	76.70	99.06
	std	6.64	7.45	8.39	8.57	8.53	**5.68**	9.21	4.34
FHDDM	$\overline{a}$	84.93	92.38	74.40	69.93	85.07	89.85	76.72	97.02
	std	7.74	7.11	8.31	9.12	9.14	7.82	9.15	6.15
No Change Detection	$\overline{a}$	78.87	90.17	63.13	72.00	80.58	89.85	76.07	75.35
	std	12.40	12.84	21.78	11.01	13.28	7.82	16.19	28.62

Table 4 Rank of accuracy of the algorithms and significance tests

Datasets	*FHDDM*	$HDDM_W$	$HDDM_A$	*DDM*	*EDDM*	*SeqDrift2*	*RDDM*	*NoChange*	$CCPD_{k=3}$
Electricity	84.93(7)	85.46(5)	86.11(2)	85.83(3.5)	85.26(6)	82.23(8)	85.83(3.5)	78.87(9)	**86.20(1)**
Spam	92.38(2)	91.51(8)	91.66(6)	91.94(4)	91.68(5)	91.55(7)	92.09(3)	90.63(9)	**92.98(1)**
Usenet1	74.73 (5.5)	75.07(4)	75.20(2)	73.73(7)	75.13(3)	69.67(8)	74.73(5.5)	63.33(9)	**78.27(1)**
Usenet2	71.20(6.5)	70.93(9)	71.00(8)	72.93(3)	**73.27(1)**	72.13(4.5)	73.07(2)	72.13(4.5)	71.20(6.5)
Covertype	85.10(7)	86.23(5)	87.44(3)	88.03(2)	86.09(6)	82.86(8)	86.87(4)	80.58(9)	**89.51(1)**
Nursery	89.85(7.5)	91.71(6)	92.51(2)	91.72(5)	92.02(3)	89.01(9)	**92.70(1)**	89.85(7.5)	91.85(4)
Poker	76.72(4)	77.11(3)	76.48(6)	72.74(8)	77.48(2)	72.51(9)	76.70(5)	76.07(7)	**80.42(1)**
EEG Eye	97.28(6)	97.60(5)	98.89(4)	99.64(1)	96.85(7)	93.63(8)	99.06 (3)	75.35(9)	99.62(2)
average rank	5.69	5.63	4.13	4.19	4.13	7.69	3.38	8.0	**2.19**

mean rank	*Value of* χ^2	F_F	*Critical F-value*
5.0	31.8167	6.9202	2.1782

values indicate the best results per dataset. Based on the methodology, we reject the null-hypothesis because $F_F > \textit{Critical F-value}$. This means that there are significant differences between the algorithms. Finally, we use the Nemenyi post-hoc test to present these differences. We set the significance level at 5%. The critical value for 9 algorithms is 3.10. The critical difference at level 5% is $CD = 4.24$. Fig. 7 shows the results of the Nemenyi test of the data from Table 4. On the figure, the methods on the right side have lower average rank of accuracy and are better than the methods on the left.

5.3 Impact of the k-Order

In this section, we run tests to evaluate the dependencies of data points in the streams in the Electricity, Poker, Forest Covertype, Spam, Usenet1, Usenet2, Nursery, and EEG Eye State datasets. We record average accuracy and standard deviation of the CCPD on the datasets with NB classifier while varying the parameter k from 2 to 7. Table 5 shows the average accuracy and standard deviation of the CCPD method while varying k from 2 to 7. From the results, we can observe that the different values of k will affect to the accuracy of the algorithm. The best accuracy is almost obtained at order k=2 or k=3. On Electricity, Usenet2, and Covertype datasets, the best performance is obtained when we project on k=2 data, while on the remaining datasets, the best performance is at k=3. On the EEG Eye State dataset, when the value of k increases from 2 to 7, the accuracy also slightly increases. For the sake of comparison, the table only shows $k \in [2, 7]$. Since we observed that the values were still increasing, we decided to vary k, until $k = 20$ to find the optimal value. The optimal accuracy of 99.55% was found at $k = 13$. The best accuracy at different order k also depends on characteristic of the input dataset. Furthermore, the results show that change in accuracy, abrupt or slight, is dataset-dependent. On Electricity, Spam, Covertype, and Poker datasets, when we vary value of order k, the accuracy changes slightly. In contrast, the accuracy varies abruptly on Usenet1, Usenet2, and Nursery datasets.

As shown in the results in Table 5, the best results are usually in 2 or 3 orders of dependency. The reason can be explained as that the proposed method uses the fixed length for projection of temporal dependencies. In real life datasets, the order of dependencies may not be fixed length, which means that each data point in a stream may have temporal dependencies from different number of its previous data points. For this reason, our further research will include studying higher-order temporal dependencies.

5.4 True Change Point and Delay

One of the disadvantages of some real-world datasets in change detection problem is that the ground truth of the datasets is unknown. On the other hand, the performance of a detection method is evaluated base on the accuracy and delay of detected change points to the real change points in the data. In the above section, we have presented the efficiency of our proposed method with high accuracy detection. In this section, for a further evaluation of the algorithm, we describe the tests we performed with CCPD on a real-world dataset for which its ground truth has been known. This is the Electricity dataset, which is also widely-used in many methods [10, 45] for change detection. In this dataset, data are heavily autocorrelated with frequency peaks in every 48 instances [45].

In this experiment, we performed tests on the first 1,000 instances of the dataset. We then record change points detected by CCPD and other state-of-the-art algorithms with respect to true change points that are known as ground truth. Specifically, we did experiments employing CCPD, HDDM$_W$, HDDM$_A$, CUSUM, and PAGE-HINKLEY algorithms. CCPD, HDDM$_W$, and HDDM$_A$ are here online detection algorithms. PAGE-HINKLEY is a concept drift detection based on the Page Hinkley Test, while CUSUM is a drift detection method based on cumulative sum. For the best competitive comparison, CUSUM and PAGE-HINKLEY are executed in an online manner at every data point in the stream. Here, we set sample frequency to 1. We adjust the minimal number of instance to 1, and set all the other parameters to default values as in MOA framework according to prior works.

Table 6 shows the change points detected by the algorithms on Electricity dataset. The CCPD detects online and at exact change points. CUSUM and PAGE-HINKLEY have the same result with a short delay detection of change, which is 1 data point. HDDM$_{A-test}$ produces the largest delay with 15 data points in delay. And the last algorithm HDDM$_{W-test}$ detects change with 4 data points in delay.

5.5 Experiments on Synthetic Datasets

This subsection presents experiments to evaluate detection delay, true positive (TP), true negative (TN), false negative (FN), and accuracy of our proposed method. We compared the results with several state-of-the-art

Table 5 Accuracy results (%) of CCPD with Native Bayes classifier

$CCPD_k$	Electricity (%)	Spam (%)	Usenet1 (%)	Usenet2 (%)	Covertype (%)	Nursery (%)	Poker (%)	EEG Eye (%)
k=2	**86.29±5.35**	92.84±7.79	76.07±11.40	**70.40±10.63**	**89.29±8.02**	85.35±14.23	80.29±8.12	99.43±1.68
k=3	86.20±5.45	**92.88±7.98**	**78.27±11.67**	63.00±21.81	89.18±8.46	**91.85±9.20**	**80.42±8.36**	99.43±1.68
k=4	85.78±5.62	92.80±8.18	77.13±10.53	65.07±20.62	89.06±8.94	83.45±14.72	80.29±8.80	99.45±1.68
k=5	85.19±5.93	92.65±8.60	75.67±10.30	61.20±19.87	88.98±9.39	83.46±14.69	79.96±9.33	99.45±1.62
k=6	84.83±6.08	92.74±8.40	71.47±9.89	61.27±19.88	88.95±9.55	52.02±12.51	79.61±9.83	99.46±1.57
k=7	84.39±6.39	92.66±8.57	61.87±10.91	53.00±14.85	88.95±9.65	43.02±5.15	79.29±10.23	**99.48±1.45**

Table 6 Change points detected, where i =1, 2, ...

Algorithm	Detected Points
$CCPD_5$	$(48 \times i + 1) \pm 0$
$HDDM_{W-test(0.05)}$	$(48 \times i + 1) \pm 4$
$HDDM_{A-test}$	$(48 \times i + 1) \pm 15$
$CUSUM$	$(48 \times i + 1) \pm 1$
$PAGE - HINKLEY$	$(48 \times i + 1) \pm 1$

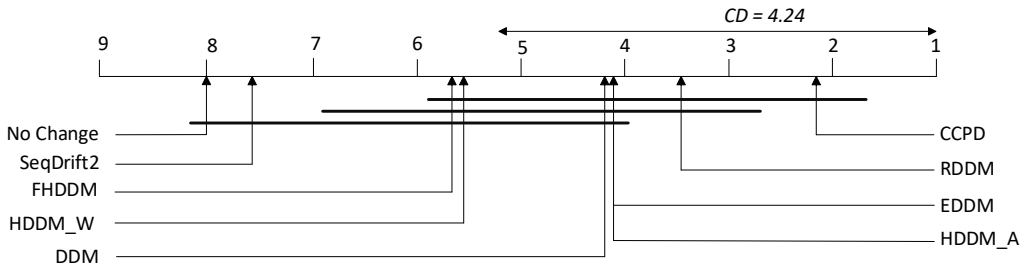

Fig. 7 Nemenyi test with confidence level $\alpha = 0.05$

drift detectors, including EDDM [4], ECDD [39], SeqDrift2 [33], and RDDM [5], on four widely-used synthetic data streams in the literature [18, 36]: Mixed, Sine, Circles, and LED. Each dataset contains 100,000 instances, and 10% noise in class of instances. In brief, the characteristics of these datasets are as follows:

- *Mixed*: This dataset contains four attributes, including two Boolean attributes and two numeric attributes in the [0, 1] interval. The Mixed dataset contains abrupt concept drifts. The drifts occur at every 20,000 instances with a transition length $\xi = 50$.
- *Sine1*: In this dataset there are two attributes that are uniformly distributed in the [0, 1] interval. The Sine1 dataset contains abrupt concept drifts. The drifts occur at every 20,000 instances with a transition length $\xi = 50$.
- *Circles*: This dataset uses four circles to simulate concept drifts. Each instance has two numeric attributes on the [0, 1] interval. The Circles dataset contains gradual concept drifts. The drifts occur at every 25,000 instances with a transition length $\xi = 500$.
- *LED*: This dataset is a seven-segment display of digit dataset. The LED dataset contains gradual concept drifts. The drifts occur at every 25,000 instances with a transition length $\xi = 500$.

All experiments on the synthetic datasets were performed using the MOA framework with parameters set to the optimal values for all the compared algorithms as recommended in the original papers. We adopted the *acceptable delay* length metric [36] to evaluate the performance of the algorithms. Given a threshold, if a detector can detect a change within a threshold delay from the true change point, it is considered as a true positive. In the experiments on the Mixed and Sine1 datasets, the level of temporal dependency k is set to 3 and the acceptable delay is set to 250. While on the Circles and LED datasets, k is empirically set to 5 and the acceptable delay is set to 1,000.

Table 7 shows the average and standard deviation of classification results for the CCPD, EDDM, ECDD, SeqDrift2, and RDDM running on 100 samples of datasets. The results show that, in most cases, ECDD and EDDM are the worst detectors. On the Circles dataset, SeqDrift2 has the best performance, while SeqDrift2 has the worst performance on the LED dataset. This can be explained as follows. SeqDrift2 maintains a fixed size reservoir sampling for concept drift detection. The reservoir sampling contains 200 instances, and this size

Table 7 Results with Natïve Bayes(NB) and Hoeffding Tree(HT) classifiers on synthetic datasets

		Algorithms	Delay	TP	FP	FN	Accuracy	Rank
Mixed dataset	NB	CCPD	62.60 ± 7.74	4.0 ± 0.0	0.01 ± 0.1	0.0 ± 0.0	**83.34 ± 0.08**	1
		RDDM	104.97 ± 12.12	3.99 ± 0.1	1.86 ± 1.66	0.01 ± 0.1	83.24 ± 0.09	2
		SeqDrift2	200.0 ± 0.0	4.0 ± 0.0	4.39 ± 0.79	0.0 ± 0.0	82.91 ± 0.08	3
		ECDD	38.87 ± 24.65	3.81 ± 0.42	142.29 ± 7.90	0.19 ± 0.42	81.00 ± 0.15	4
		EDDM	247.47 ± 8.65	0.11 ± 0.31	20.22 ± 7.70	3.89 ± 0.31	80.30 ± 2.33	5
	HT	CCPD	63.22 ± 9.23	4.0 ± 0.0	0.04 ± 0.20	0.0 ± 0.0	**83.37 ± 0.10**	1
		RDDM	106.68 ± 11.32	3.99 ± 0.1	3.49 ± 2.48	0.01 ± 0.1	83.17 ± 0.12	2
		SeqDrift2	200.0 ± 0.0	4.0 ± 0.0	4.98 ± 1.21	0.0 ± 0.0	82.91 ± 0.11	3
		ECDD	39.76 ± 26.08	3.79 ± 0.46	138.34 ± 7.95	0.21 ± 0.46	80.95 ± 0.15	4
		EDDM	248.46 ± 7.73	0.05 ± 0.22	21.51 ± 7.74	3.95 ± 0.22	80.65 ± 0.82	5
Sine1 Dataset	NB	CCPD	59.54 ± 6.89	4.0 ± 0.0	0.01 ± 0.1	0.0 ± 0.0	**86.03 ± 0.25**	1
		RDDM	89.73 ± 16.54	3.99 ± 0.10	3.93 ± 2.92	0.01 ± 0.1	85.98 ± 0.27	2
		SeqDrift2	200.0 ± 0.0	4.0 ± 0.0	4.26 ± 0.0	0.0 ± 0.58	85.60 ± 0.25	3
		ECDD	33.30 ± 23.22	3.85 ± 0.39	153 ± 8.34	0.15 ± 0.39	84.38 ± 0.14	4
		EDDM	234.28 ± 22.33	0.57 ± 0.64	33.53 ± 11.56	3.43 ± 0.64	83.44 ± 2.88	5
	HT	CCPD	58.45 ± 6.55	4.0 ± 0.0	0.03 ± 0.17	0.0 ± 0.0	**87.02 ± 0.15**	1
		RDDM	93.54 ± 7.82	4.0 ± 0.0	4.72 ± 3.59	0.0 ± 0.0	86.79 ± 0.19	2
		SeqDrift2	200.0 ± 0.0	4.0 ± 0.0	4.83 ± 1.16	0.0 ± 0.0	86.53 ± 0.15	3
		ECDD	36.58 ± 25.54	3.8 ± 0.43	153.78 ± 7.67	0.2 ± 0.43	84.28 ± 0.14	5
		EDDM	243.83 ± 22.33	0.22 ± 0.64	33.77 ± 11.56	3.78 ± 0.64	84.71 ± 2.88	4
Circles Dataset	NB	CCPD	621.09 ± 139.12	1.59 ± 0.57	0.50 ± 0.64	1.41 ± 0.57	83.49 ± 0.52	3
		RDDM	406.50 ± 69.75	2.99 ± 0.1	2.15 ± 1.95	0.01 ± 0.1	84.05 ± 0.12	2
		SeqDrift2	276.67 ± 91.56	2.92 ± 0.27	2.49 ± 0.98	0.08 ± 0.27	**84.13 ± 0.14**	1
		ECDD	194.64 ± 158.13	2.84 ± 0.37	174.53 ± 7.62	0.16 ± 0.37	83.18 ± 0.11	4
		EDDM	938.27 ± 107.14	0.35 ± 0.5	31.09 ± 18.23	2.65 ± 0.5	83.12 ± 0.4	5
	HT	CCPD	524.62 ± 144.31	2.02 ± 0.57	0.67 ± 0.78	0.98 ± 0.57	85.94 ± 0.42	3
		RDDM	293.80 ± 38.72	2.98 ± 0.14	0.79 ± 1.26	0.02 ± 0.14	86.46 ± 0.16	2
		SeqDrift2	202.67 ± 16.19	3.0 ± 0.0	3.08 ± 0.91	0.0 ± 0.0	**86.47 ± 0.14**	1
		ECDD	186.40 ± 151.67	2.86 ± 0.35	175.16 ± 8.39	0.14 ± 0.35	83.21 ± 0.12	5
		EDDM	987.75 ± 54.64	0.06 ± 0.24	24.45 ± 14.57	2.94 ± 0.24	84.89 ± 0.29	4
LED Dataset	NB	CCPD	481.75 ± 129.42	2.77 ± 0.55	13.44 ± 7.75	0.23 ± 0.55	89.15 ± 0.57	2
		RDDM	321.80 ± 51.19	2.98 ± 0.14	0.61 ± 0.96	0.02 ± 0.14	**89.63 ± 0.04**	1
		SeqDrift2	445.33 ± 193.24	2.75 ± 0.46	278.82 ± 47.74	0.25 ± 0.46	76.54 ± 2.27	5
		ECDD	194.53 ± 139.51	2.86 ± 0.40	60.51 ± 3.58	0.14 ± 0.40	86.41 ± 0.19	4
		EDDM	949.61 ± 69.29	0.70 ± 0.73	6.33 ± 1.97	2.3 ± 0.73	88.32 ± 0.53	3
	HT	CCPD	479.84 ± 124.70	2.77 ± 0.55	13.9 ± 7.07	0.23 ± 0.55	89.21 ± 0.31	2
		RDDM	321.88 ± 51.20	2.98 ± 0.14	0.61 ± 0.96	0.02 ± 0.14	**89.63** ± 0.04	1
		SeqDrift2	426.0 ± 174.18	2.78 ± 0.44	277.06 ± 47.71	0.22 ± 0.44	76.51 ± 2.29	5
		ECDD	197.07 ± 140.93	2.85 ± 0.41	60.19 ± 3.68	0.15 ± 0.41	86.39 ± 0.19	4
		EDDM	954.97 ± 63.30	0.66 ± 0.71	5.97 ± 1.70	2.34 ± 0.71	88.33 ± 0.50	3

Algorithms	CCPD	RDDM	SeqDrift2	ECDD	EDDM
Average Rank	**1.75**	**1.75**	3.0	4.25	4.25

is suitable for the Circles dataset since it contains gradual concept drifts with a transition length of 500. The low accuracy of SeqDrift2 on LED is a result of its very high false positive. On the Mixed and Sine1 datasets, we observe that, the shortest delay is obtained by ECDD. The reason is that the number of instances in the estimated window in ECDD is small. However, with ECDD, both the TP rate and the FP rate are high, thus resulting in a low accuracy. In almost all cases, CCPD and RDDM have the best accuracy and very good flow rates of detection delay, TP, FP, and FN. The CCPD has the most accurate and very low FP, FN rates on the Mixed and Sine1 datasets; while RDDM has good performance on the LED dataset. This is because RDDM discards old instances from the stream, while in CCPD, we weigh the current instance based on a projection on k latest instances in the stream. Therefore, any concept drifts can be quickly detected on abrupt concept drift datasets like Mixed and Sine1. Overall, the CCPD and RDDM have the same rank (1.75).

5.6 Runtime Performance

In terms of runtime performance, we performed experiments to evaluate the classification time and the streaming processing speed of our proposed method. Table 8 presents the evaluations of the method in terms of running time in streaming on Electricity, Poker, For-

Table 8 Evaluations on running time of the CCPD

Metrics	Electricity	Spam	Usenet1	Usenet2	Covertype	Nursery	Poker	EEG Eye
Number of learning evaluations	454	94	15	15	5,812	130	8,293	150
Average total running time	3.135(s)	5.401(s)	0.190(s)	0.323(s)	59.513(s)	23.719(s)	40.557(s)	11.220(s)
Average time/learning	6.90(ms)	57.46(ms)	12.67(ms)	21.53(ms)	10.24(ms)	182.45(ms)	4.89(ms)	74.8(ms)
Number of processing/second	144.8/(s)	17.4/(s)	78.9/(s)	46.5/(s)	97.7/(s)	5.5/(s)	204.5/(s)	13.4/(s)

est Covertype, Spam, Usenet1, Usenet2, Nursery, and EEG Eye datasets. It shows number of learning evaluations, average total running time in second, average time using for each learning process and number of learning can be processed per second. From this table, we can observe that our detector is capable of processing a streaming at high velocity, up to 204.5 process per second. Hence, it is feasible to detect changes in an online setting as in streaming manner.

6 Conclusion

In this paper, we presented a new approach for detecting changes in an open-ended data stream. We proposed a k-order Candidate Change Point (CCP) Model that builds on linear higher order Markov processes, in order to exploit the temporal dependency among data in a stream. The main idea with the model is to compute the probability of finding change points in a given observation time window, using the temporal dependency information or factors between different observed data points in a stream. To cope with the dynamic nature of the stream, we proposed an approach that can continuously optimize the temporal dependency factors by using an Euclidean projection on ℓ_1 ball constraints. In addition, we introduced a concept called CCP trail, which refers to the probabilistic path from a specific observed data point to another previously observed data point. Our approach adapts the probability of finding change points to continuously estimate the CCP trail means in streaming data. Using CCP trail mean values, we applied statistical tests to detect the change points. To evaluate our approach, we performed extensive experiments using several datasets and compared our algorithm to the state-of-the-art algorithms. Our evaluation showed that our k-order Candidate Change Point Model is effective, and that the Candidate Change Point Detector (CCPD) algorithm outperforms the state-of-the-art algorithms on most of the datasets. In addition, our method has a linear time performance, which enables it to be deployed online in real-world stream applications.

There are several directions to extend this work. First, it is worth investigating how the number of different CCPs in different data points affects the dependency model. Second, in data stream, a large volume of data arrives at a high speed. Therefore, it is infeasible to maintain information of all data. Developing sketching algorithms that combine temporal dependency for detecting drifts and outliers is an area for further study.

Acknowledgements

This research is funded by the Norwegian University of Science and Technology (NTNU) through the MUSED project.

References

1. Adä, I., Berthold, M.R.: EVE: a framework for event detection. Evolving Systems **4**(1), 61–70 (2013)
2. Adhikari, U., Morris, T., Pan, S.: Applying Hoeffding Adaptive Trees for Real-Time Cyber-Power Event and Intrusion Classification. IEEE Transactions on Smart Grid **PP**(99), 1–12 (2017)
3. Anagnostopoulos, C., Tasoulis, D.K., Adams, N.M., Pavlidis, N.G., Hand, D.J.: Online Linear and Quadratic Discriminant Analysis with Adaptive Forgetting for Streaming Classification. Statistical Analysis and Data Mining **5**(2), 139–166 (2012)
4. Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldá, R., Morales-Bueno, R.: Early drift detection method. In: The 4th International Workshop on Knowledge Discovery from Data Streams (2006)
5. Barros, R.S., Cabral, D.R., Gonalves, P.M., Santos, S.G.: RDDM: Reactive drift detection method. Expert Systems with Applications **90**(Supplement C), 344–355 (2017)
6. Bifet, A.: Classifier Concept Drift Detection and the Illusion of Progress. In: Artificial Intelligence and Soft Computing, pp. 715–725. Springer International Publishing, Cham (2017)
7. Bifet, A., Gavaldà, R.: Learning from Time-Changing Data with Adaptive Windowing. In: Proceedings of the 2007 SIAM International Conference on Data Mining, pp. 443–448 (2007)

8. Bifet, A., Gavaldà, R.: Adaptive Learning from Evolving Data Streams. In: Proceedings of the 8th International Symposium on Intelligent Data Analysis, pp. 249–260 (2009)
9. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive Online Analysis. The Journal of Machine Learning Research **11**, 1601–1604 (2010)
10. Bifet, A., Read, J., Žliobaitė, I., Pfahringer, B., Holmes, G.: Pitfalls in Benchmarking Data Stream Classification and How to Avoid Them. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, ECML PKDD, pp. 465–479 (2013)
11. Bodenham, D.A., Adams, N.M.: Continuous monitoring for changepoints in data streams using adaptive estimation. Statistics and Computing **27**(5), 1257–1270 (2017)
12. Bryc, W.: A uniform approximation to the right normal tail integral. Applied Mathematics and Computation **127**(2), 365–374 (2002)
13. Chattopadhyay, S., Murthy, C., Pal, S.K.: Fitting truncated geometric distributions in large scale real world networks. Theoretical Computer Science **551**, 22–38 (2014)
14. Condat, L.: Fast projection onto the simplex and the and the ℓ_1 ball. Mathematical Programming **158**(1), 575–585 (2016)
15. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. The Journal of Machine Learning Research **7**, 1–30 (2006)
16. Duchi, J., Shalev-Shwartz, S., Singer, Y., Chandra, T.: Efficient projections onto the ℓ_1-ball for learning in high dimensions. In: Proceedings of the 25th International Conference on Machine Learning, ICML, pp. 272–279 (2008)
17. Frías-Blanco, I.I., del Campo-Ávila, J., Ramos-Jiménez, G., Carvalho, A.C.P.L.F., Díaz, A.A.O., Morales-Bueno, R.: Online adaptive decision trees based on concentration inequalities. Knowledge-Based Systems **104**, 179–194 (2016)
18. Frías-Blanco, I.I., del Campo-Ávila, J., Ramos-Jiménez, G., Morales-Bueno, R., Ortiz-Díaz, A.A., Caballero-Mota, Y.: Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds. IEEE Transactions on Knowledge and Data Engineering **27**(3), 810–823 (2015)
19. Gama, J., Medas, P., Castillo, G., Rodrigues, P.P.: Learning with Drift Detection. In: Proceedings of Brazilian Symposium on Artificial Intelligence, pp. 286–295 (2004)
20. Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Machine Learning **90**(3), 317–346 (2013)
21. Gama, J.a., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A Survey on Concept Drift Adaptation. ACM Computing Surveys **46**(4), 1–37 (2014)
22. Gomes, H.M., Barddal, J.P., Enembreck, F., Bifet, A.: A Survey on Ensemble Learning for Data Stream Classification. ACM Computing Surveys **50**(2), 23:1–23:36 (2017)
23. Gomes, H.M., Bifet, A., Read, J., Barddal, J.P., Enembreck, F., Pfharinger, B., Holmes, G., Abdessalem, T.: Adaptive random forests for evolving data stream classification. Machine Learning **106**(9), 1469–1495 (2017)
24. Harries, M.B., Sammut, C., Horn, K.: Extracting Hidden Context. Machine Learning **32**(2), 101–126 (1998)
25. Hoeffding, W.: Probability Inequalities for Sums of Bounded Random Variables. Journal of the American Statistical Association **58**(301), 13–30 (1963)
26. Kifer, D., Ben-David, S., Gehrke, J.: Detecting Change in Data Streams. In: Proceedings of the 13th International Conference on Very Large Data Bases - Volume 30, VLDB, pp. 180–191 (2004)
27. Kolter, J.Z., Maloof, M.A.: Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts. The Journal of Machine Learning Research **8**, 2755–2790 (2007)
28. Kumar, R., Raghu, M., Sarlós, T., Tomkins, A.: Linear additive markov processes. In: Proceedings of the 26th International Conference on World Wide Web, pp. 411–419 (2017)
29. Li, P., Wu, X., Hu, X.: Mining recurring concept drifts with limited labeled streaming data. In: Proceedings of the 2nd Asian Conference on Machine Learning, *PMLR*, vol. 13, pp. 241–252 (2010)
30. Liu, J., Ye, J.: Efficient euclidean projections in linear time. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML, pp. 657–664 (2009)
31. Markov, A.: Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain. In: Appendix B, Dynamic Probabilistic Systems (Volume I: Markov Models), pp. 552–577 (1971)
32. Page, E.S.: Continuous Inspection Schemes. Biometrika **41**(1/2), 100–115 (1954)
33. Pears, R., Sakthithasan, S., Koh, Y.S.: Detecting concept change in dynamic data streams. Machine Learning **97**(3), 259–293 (2014)
34. Pesaranghader, A., Viktor, H., Paquet, E.: McDiarmid Drift Detection Methods for Evolving Data Streams. CoRR **abs/1710.02030** (2017)
35. Pesaranghader, A., Viktor, H., Paquet, E.: Reservoir of Diverse Adaptive Learners and Stacking Fast Hoeffding Drift Detection Methods for Evolving Data Streams. CoRR **abs/1709.02457** (2017)
36. Pesaranghader, A., Viktor, H.L.: Fast Hoeffding Drift Detection Method for Evolving Data Streams. In: Proceedings of the 2016 Machine Learning and Knowledge Discovery in Databases, ECML PKDD, pp. 96–111 (2016)
37. Roberts, S.W.: Control Chart Tests Based on Geometric Moving Averages. Technometrics **1**(3), 239–250 (1959)
38. Rösler, O., Suendermann, D.: A first step towards eye state prediction using EEG. In: Proceedings of the International Conference on Applied Informatics for Health and Life Sciences (AIHLS 2013) (2013)
39. Ross, G.J., Adams, N.M., Tasoulis, D.K., Hand, D.J.: Exponentially weighted moving average charts for detecting concept drift. Pattern Recognition Letters **33**(2), 191–198 (2012)
40. Schlimmer, J.C., Granger, R.H.: Incremental learning from noisy data. Machine learning **1**(3), 317–354 (1986)
41. Sebastião, R., Gama, J., Mendonça, T.: Fading histograms in detecting distribution and concept changes. International Journal of Data Science and Analytics **3**(3), 183–212 (2017)
42. Tibshirani, R.J., Taylor, J., Lockhart, R., Tibshirani, R.: Exact Post-Selection Inference for Sequential Regression Procedures. Journal of the American Statistical Association **111**(514), 600–620 (2016)
43. Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., Weinberger, M.J.: Inequalities for the ℓ_1 Deviation of the Empirical Distribution. Technical report, Hewlett-Packard Labs (2003)
44. Wu, T., Gleich, D.F.: Retrospective higher-order markov processes for user trails. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, pp. 1185–1194 (2017)

45. Žliobaitė, I., Bifet, A., Read, J., Pfahringer, B., Holmes, G.: Evaluation methods and decision theory for classification of streaming data with temporal dependence. Machine Learning **98**(3), 455–482 (2015)

Appendix A PROOF

A.1 Proof of Property 1

Proof Equation 7 can be easily proved by induction. When $t_2 = t_1$, we have:

$$\begin{aligned}\prod_{t=t_1}^{t_2}(1_{\{t\neq t_1\}} \times ch_t + 1_{\{t=t_1\}}) &= 1_{\{t_1\neq t_1\}} \times ch_{t_1} + 1_{\{t_1=t_1\}} \\ &= 0 \times ch_{t_1} + 1 = 1 \\ &= ct(t_1, t_1) = ct(t_2, t_1).\end{aligned}$$

When $t_2 = t_1 + 1$, we have:

$$\begin{aligned}\prod_{t=t_1}^{t_2} 1_{\{t\neq t_1\}} \times ch_t + 1_{\{t=t_1\}} &= \prod_{t=t_1}^{t_1+1} 1_{\{t\neq t_1\}} \times ch_t + 1_{\{t=t_1\}} \\ &= ch_{t_1+1} = ct(t_1+1, t_1) \\ &= ct(t_2, t_1).\end{aligned}$$

Assume that Equation 7 is satisfied when $t_2 = t_1 + m$, with $m \in \mathbb{N}, m > 0$. We prove that Equation 7 is also satisfied with $t_2 = t_1 + m + 1$. We have:

$$\begin{aligned}ct(t_2, t_1) &= ct(t_1 + m + 1, t_1) \\ &= ct(t_1 + m + 1, t_1 + m) \times ct(t_1 + m, t_1) \\ &= ch_{t_1+m+1} \times \prod_{t=t_1}^{t_1+m}(1_{\{t\neq t_1\}} \times ch_t + 1_{\{t=t_1\}}) \\ &= \prod_{t=t_1}^{t_1+m+1}(1_{\{t\neq t_1\}} \times ch_t + 1_{\{t=t_1\}}).\end{aligned}$$

A.2 Proof of Proposition 1

Proof We have:

$$\begin{aligned}&ctsum(t-1) \times ch_t + ct(t,t) \\ &= ct(t,t) + ch_t \times \sum_{i=1}^{t-1} ct(t-1, i) \\ &= ct(t,t) + \sum_{i=1}^{t-1} ch_t \times ct(t-1, i) \\ &= ct(t,t) + \sum_{i=1}^{t-1} ct(t, t-1) \times ct(t-1, i) \\ &= ct(t,t) + \sum_{i=1}^{t-1} ct(t, i) = \sum_{i=1}^{t} ct(t, i) = ctsum(t).\end{aligned}$$

A.3 Proof of Proposition 2

Proof We have:

$$\begin{aligned}cp(t) &= \sum_{i=1}^{t} v_i \times ct(t, i) = \sum_{i=1}^{t} v_i \prod_{j=i}^{t}(1_{\{j\neq i\}} \times ch_j + 1_{\{j=i\}}) \\ &= \sum_{i=1}^{t-1} v_i \prod_{j=i}^{t}(1_{\{j\neq i\}} \times ch_j + 1_{\{j=i\}}) \\ &\quad + v_t \prod_{j=t}^{t}(1_{\{j\neq t\}} \times ch_j + 1_{\{j=t\}}) \\ &= v_t + \sum_{i=1}^{t-1} v_i (1_{\{t\neq i\}} \times ch_t + 1_{\{t=i\}}) \\ &\quad \times \prod_{j=i}^{t-1}(1_{\{j\neq i\}} \times ch_j + 1_{\{j=i\}}) \\ &= v_t + ch_t \sum_{i=1}^{t-1} v_i \prod_{j=i}^{t-1}(1_{\{j\neq i\}} \times ch_j + 1_{\{j=i\}}) \\ &= v_t + ch_t \times cp(t-1).\end{aligned}$$