A Novel Concept Drift Detection Method for Incremental Learning in Nonstationary Environments

Zhe Yang, Sameer Al-Dahidi, Piero Baraldi, Enrico Zio, *Senior Member, IEEE*, and Lorenzo Montelatici

***Abstract*— We present a novel method for concept drift detection, based on: 1) the development and continuous updating of online sequential extreme learning machines (OS-ELMs) and 2) the quantification of how much the updated models are modified by the newly collected data. The proposed method is verified on two synthetic case studies regarding different types of concept drift and is applied to two public real-world data sets and a real problem of predicting energy production from a wind plant. The results show the superiority of the proposed method with respect to alternative state-of-the-art concept drift detection methods. Furthermore, updating the prediction model when the concept drift has been detected is shown to allow improving the overall accuracy of the energy prediction model and, at the same time, minimizing the number of model updatings.**

***Index Terms*— Concept drift detection, energy production prediction, incremental learning, model dissimilarity, online sequential extreme learning machine (OS-ELM).**

I. Introduction

THE analysis of data streams using machine learning (ML) methods [1]–[4] to solve regression, classification, and prediction problems is attracting great attention. A common challenge in applications of ML, e.g., for surface defect detection on metal products [5], fault detection, diagnostics and remaining useful life prediction in industrial components [6]–[9], and wind energy production prediction [10]–[12], is that the environment in which data streams are collected is typically nonstationary [13]. Common causes of nonstationary behaviors, also known as concept drifts, are seasonality effects, sensor or component aging, thermal drifts, and operation mode changes or users' interest drifts [14]. When facing nonstationary environments, the problem of adapting ML models becomes crucial.

Assuming that $\boldsymbol{x}_t$ is the input vector of an ML model at time t and $\boldsymbol{y}_t$ is the corresponding output target vector, the concept drifts are typically distinguished into [14], [15]: 1) virtual drifts in which the distribution of the input data $p_t(\boldsymbol{x})$ changes with time, whereas the posterior probability of the output $p_t(\boldsymbol{y}|\boldsymbol{x})$, representing the mapping relationship between $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$, is not changing; 2) real drifts in which the posterior probability distributions $p_t(\boldsymbol{y}|\boldsymbol{x})$ varies over time, independently of variations in $p_t(\boldsymbol{x})$; and 3) hybrid drifts in which virtual and real drifts occur at the same time (hybrid drifts are the most common in industrial applications).

With respect to the type of distribution modification, concept drifts are typically classified as sudden, incremental, gradual, or recurring [16]. Considering examples of sensor anomalies, an abrupt signal bias is a sudden drift, a bias whose amplitude increases with time is an incremental drift, signal spikes with increasing frequency before sensor failure constitute a gradual drift, and sensor readings occurring during some periodic plant conditions constitute a recurring drift.

Approaches for adapting ML models to concept drifts are typically categorized into passive and active [14]. Passive approaches adapt the model every time new data become available. Despite the fact that these approaches are effective, they are difficult to apply due to the large computational efforts required for model updating. Contrarily, active approaches aim at balancing the accuracy and updating burden by modifying the ML model only when the occurrence of a concept drift is detected. Most of the proposed active approaches are unable to deal with all the types of concept drifts, but only with a subset of them [16].

Active approaches are typically classified into three broad categories [16]: 1) sequential analysis-based; 2) data distribution-based; and 3) learner output-based. Sequential analysis-based approaches analyze the newly acquired patterns one by one until the probability of observing the data sequence under a new distribution p_1 is significantly larger than that under the original distribution p_0 [17], [18]. For example, the sequential probability ratio test [6], [19], [20] uses as drift detection index the logarithm of the ratio of the likelihood of the two distributions p_0 and p_1. Similarly, cumulative sum (CUSUM) [21] detects a drift when the CUSUM of the data sequence significantly deviates from its mean value.

Manuscript received April 17, 2018; revised October 26, 2018 and January 25, 2019; accepted February 12, 2019. This work was supported by China Scholarship Council under Grant 201506280015. *(Corresponding author: Piero Baraldi.)*

Z. Yang and P. Baraldi are with the Energy Department, Politecnico di Milano, 20156 Milan, Italy (e-mail: piero.baraldi@polimi.it).

S. Al-Dahidi is with the Department of Mechanical and Maintenance Engineering, School of Applied Technical Sciences, German Jordanian University, Amman 11180, Jordan.

E. Zio is with the Energy Department, Politecnico di Milano, 20156 Milan, Italy, also with the Center for Research on Risk and Crises, MINES ParisTech, PSL Research University, 06100 Sophia Antipolis, France, also with Aramis Srl, 20121 Milan, Italy, and also with the Department of Nuclear Engineering, College of Engineering, Kyung Hee University, Seoul 02447, South Korea.

L. Montelatici is with Edison Research Development and Innovation, 20121 Milan, Italy.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2900956

The Page-Hinkley test [21] uses as an index for the drift detection the cumulative difference between the observed values and their mean until the current time. In [22], the Page-Hinkley test is used to detect sudden drifts in the average of a Gaussian signal. The hierarchical intersection of confidence intervals-based change detection test (H-ICI-based CDT) and its variants are designed with a two-level hierarchical structure [23]–[25]. At the first level, the ICI-based CDT monitors the stationarities of the features extracted from the data stream (e.g., mean and variance) and detects concept drifts by using ICI rules. At the second level, the concept drifts detected at the first level are validated using hypothesis tests, such as the Hotelling's T-square statistic [25], to reduce false alarms. The H-ICI-based CDT has been successfully applied to detect sudden and incremental drifts in [24] and [25]. Common drawbacks of most sequential analysis-based drift detection approaches are: 1) the detection delay caused by the fact that the variation of the considered statistical metrics becomes significant only when enough data from the new concept are collected and 2) the necessity of applying them to univariate series, which limits their use in case of multivariate time series.

Data distribution-based drift detection approaches typically consider distributions of raw data patterns from two different time windows: a fixed window containing information of the past time series behavior and a sliding window containing the most recent data [26]. Hypothesis tests are used to compare the data distributions in the two windows and detect concept drifts when they are significantly different. They are further classified into parametric and nonparametric tests. The former assumes that the data stream probability distribution functions (PDFs) before and after the concept drift occurrence are known, but their parameters are unknown. Contrarily, multivariate nonparametric tests do not require the knowledge of the data stream PDFs, which can be difficult to know in real-world problems, but they are usually less sensitive to small differences among the two PDFs [27]. The semiparametric log-likelihood (SPLL) detector uses the k-means method to cluster raw data patterns and models the obtained clusters using a mixture of Gaussian distributions. It has been successfully applied to sudden drift detection [28]. Lu *et al.* [29], [30] have modeled the raw data distribution using a competence model and have shown significant detection accuracy improvement in an application involving a case-base maintenance system. Liu *et al.* [31] proposed the nearest neighbor-based density variation identification to consider the regional density changes. In fuzzy competence model (FCM) drift detection [32], an FCM is used to represent the data empirical distribution, which provides a degree of concept drift. In [33], the QuantTree algorithm is proposed to model the empirical distribution of high-dimensional raw data using histograms and computing statistics from obtained nonparametric representation. Common drawbacks of these approaches are the need to store data in the two windows and the setting of the window lengths.

Learner output-based drift detection approaches are based on the development of a learner (classifier) and the tracking of its error rate fluctuations. In the drift detection method (DDM) [34], the classification error is represented by a Bernoulli random variable, and the concept drift is detected when the error rate exceeds a properly set threshold. The method has provided satisfactory performance in case of sudden drifts [35]. The early drift detection method (EDDM) [36], which extends DDM by considering the time distance between errors, provides better performances in case of slow gradual concept drifts. Statistical test of equal proportions (STEPD) [37] computes the accuracy of the learner on a recent window and compares it to its overall accuracy from the beginning. STEPD is shown very accurate in the detection of sudden concept drifts in [35]. Adaptive window (ADWIN) [38] monitors the distribution change of the learner output using a variable-length window obtained by partitioning the data sequence into two disjoint subsets and, then, evaluating possible statistical discrepancies between them. When a concept drift is detected, ADWIN shrinks the window to include only the data containing the new concept for the next detection. ADWIN has been shown able to provide satisfactory performances in cases of gradual and incremental drifts [35]. Exponentially weighted moving average (EWMA) for concept drift detection (ECDD) [39] uses two estimations computed by performing EWMA with different weights: one gives more weight to recent examples, while the other gives similar weights to recent and old data. A drift is detected when the difference between the two EWMA estimations exceeds a certain threshold. ECDD has been shown able to detect gradual concept drifts in [35]. DDMs Based on Hoeffding's (HDDM_A and HDDM_W) bounds [40] do not require assumptions on the probability density function of the learner classification error and perform well for sudden (HDDM_A) and gradual (HDDM_W) drifts detection, respectively. Detailed comparisons among various learner output-based drift detection approaches can be found in [35].

Learner output-based detection methods have shown more suitable for concept drift detection than sequential analysis-based and distribution-based methods in industrial applications, such as fault detection, diagnostics, and weather prediction [41], [42], in which it is required to deal with multivariate time series characterized by complex, nonlinear relationships. Since sequential analysis-based methods detect concept drifts on single time series, they cannot be used for real drifts. Although distribution-based methods can deal with multivariate data, they may detect concept drifts of variables irrelevant to the learner output, and, therefore, cause unnecessary ML model adaptations. A criticality of learner output-based methods is that they have typically been developed considering classification problems only and not regression problems. Another limitation of learner output-based methods is that when the occurrence of a concept drift is detected, they do not provide information about the amount of data necessary to adapt the learner to the new environment.

Given the limitations of active concept drift approaches, the objective of this paper is the development of a concept drift detection method able to: 1) deal with both classification and regression problems; 2) handle data batches of any size; and 3) automatically determine the amount of data necessary for model updating. Also, the concept drift detection method should be prompt in the detection of sudden, incremental,

and gradual concept drifts. Recurring concept drifts are not considered, since they typically do not require the adaptation of the ML model after the first occurrence of the drift.

The proposed method is based on the use of extreme learning machine (ELM) [43]–[45]. Unlike backpropagation neural networks (BPNNs), whose training requires the application of an iterative procedure, the internal parameters of ELM are randomly generated (input-hidden layers weights) or analytically calculated (hidden-output layers weights). Therefore, the time and computational burden of training an ELM are significantly lower than those of a BPNN [46], whereas the performance is comparable [43], [47]. Another feature of ELM exploited in this paper is its ability of online sequential learning. Liang *et al.* [48] extend the basic ELM to online sequential ELM (OS-ELM), which is able to periodically update the model output weights by using the one-by-one or batch-by-batch newly collected data, without any requirement on the batch sizes and without using the previously acquired data. The basic idea of the proposed method for concept drift detection is the quantification of how much the OS-ELM model is modified by its updating with the newly collected data. Since the OS-ELM updating only modifies the output weights, the dissimilarities between two OS-ELM models can be quantified considering the variations of the output weights.

The proposed method does not have any requirement on the amount of data to be collected, does not require knowledge of data stream probability distributions, and is not based on hand-crafted statistical features, whereas a procedure for automatically setting a dynamic threshold for concept drift detection is proposed. A further novelty of the work is that it allows estimating the amount of data which should be collected from the new concept and used to update the ML model to the new condition.

The proposed method is first applied to two synthetic case studies: the first considers sudden, gradual, and incremental virtual drifts and the second considers sudden hybrid drifts.

The benefits of the proposed concept drift detection method are further shown by its application to three real case studies: two benchmark classification problems and a regression problem regarding the prediction of energy production in a wind plant. The performance of the proposed method is evaluated in terms of false and missed concept drift detection rates and delay time and is compared with two state-of-the-art approaches, i.e., H-ICI-based CDT [24], [25] and SPLL [28].

The concept drift detection results in the three real case studies are, then, used to decide when the prediction model should be updated to obtain a satisfactory tradeoff between accurate predictions and computational effort necessary for model updating. Other strategies for model updating are considered and their performances are critically discussed.

The remaining of this paper is organized as follows. Section II introduces the problem statement. In Section III, the proposed concept drift detection method is illustrated. In Section IV, the results of the application to two synthetic case studies are presented and compared with those obtained by the H-ICI-based CDT and SPLL concept drift detection methods. Section V compares the results of the application of the method in terms of accuracy of the ML model updated when the drift is detected on three real case studies with those obtained by other alternative methods. Finally, some conclusions are drawn in Section VI.

II. Problem Statement

Let M be an ML model which outputs $\boldsymbol{y}_t$ on the basis of $\boldsymbol{x}_t$ and $O_{t_1} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t), t = 1, \ldots, t_1\}$ be an input–output data stream made by the sequence of observations collected up to time t_1, with $\boldsymbol{x}_t$ indicating the K-dimensional input vector at time t and $\boldsymbol{y}_t$ the corresponding S-dimensional target vector. We assume that at time $t_1 + \Delta t$, the batch of data $(\boldsymbol{x}_{t_1+1:t_1+\Delta t}, \boldsymbol{y}_{t_1+1:t_1+\Delta t})$ collected between times $t_1 + 1$ and $t_1 + \Delta t$ become available. The first objective of this paper is to develop a method that is able to detect the possible occurrence of a concept drift between times t_1+1 and $t_1+\Delta t$. The concept drift detection method should be characterized by low false and missed concept drift detection rates and short detection delays. Other practical requirements should be applicable, independently of the number of data collected between times $t_1 + 1$ and $t_1 + \Delta t$, e.g., to be fast to execute and not computationally demanding. The final uses of the proposed concept drift detection method are: 1) to decide when the ML model M should be updated, considering an optimal tradeoff between prediction accuracy and computational efforts needed for the updating and 2) to estimate the proper amount of data which should be used for updating M after a concept drift detection, in order to guarantee the training of an ML model with satisfactory performance in the new concept.

III. Concept Drift Detection Method

The main steps of the method are (Fig. 1) as follows.

1) The development of two OS-ELM models, which receive input $\boldsymbol{x}_t$ and provide output $\boldsymbol{y}_t$. The first model, which is referred to as the baseline model $M^*(t_1)$, is trained using data collected until time t_1 and represents the old system behavior. The second model, which is referred to as an updated model $M^*(t_1 + \Delta t)$, is obtained by updating the first model with the information acquired in the time interval $[t_1 + 1, t_1 + \Delta t]$. Note that the training of models $M^*(t_1)$ and $M^*(t_1 + \Delta t)$ requires much lower computational burden than that of the ML model M.
2) The quantification of the degree of dissimilarity between the OS-ELM models $M^*(t_1)$ and $M^*(t_1 + \Delta t)$.
3) The setting of a dynamic threshold on the degree of dissimilarity for concept drift detection.
4) The identification of the proper amount of data necessary for updating M and M^*, in case of the concept drift detection.

Section III-A briefly recalls the basics of OS-ELMs and Section III-B defines the dissimilarity measure among OS-ELM models, whereas Section III-C describes the procedure used for setting the threshold used for the concept drift detection. Section III-D illustrates the procedure for determining the amount of data to be collected from the new concept to update the ML model. Section III-E analyzes the computational cost of the proposed method. Finally, Section III-F discusses how to set the parameters of the proposed method.

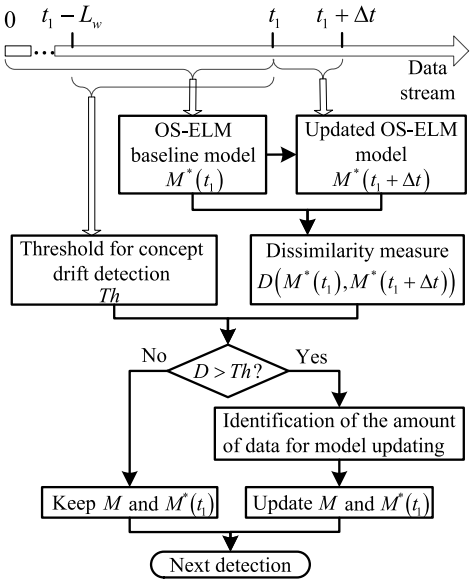

Fig. 1. Concept of drift detection method.

A. Brief Introduction to OS-ELM

ELM [43] is a training method originally proposed for a single hidden layer feedforward neural networks. Since the input weights and biases of an ELM model, $(\boldsymbol{a}_i, b_i)$, $i = 1, 2, \ldots, N_\text{h}$, with N_h indicating the number of neurons in the hidden layer, are randomly generated and the output weights $= [\ _{i,j}]$, $i = 1, 2, \ldots, N_\text{h}$, $j = 1, 2, \ldots, S$, are analytically determined, the time and computational burden necessary to train an ELM model have been shown to be significantly lower than those for training a BPNN [46]. OS-ELM [48] is an ELM variant for online applications, characterized by the fact that model updating can be performed each time a new batch of data becomes available, without any constraint on its size. The effect of OS-ELM updating is to modify only the output weights , whereas it keeps unmodified input weights $\boldsymbol{a}_i$ and biases b_i, $i = 1, 2, \ldots, N_\text{h}$. Since OS-ELM updating is fast and computationally not demanding, it is quite suitable for online concept drift detection.

B. Measure of Dissimilarity Between OS-ELM Models

Since an OS-ELM model is characterized by its output weights a measure of dissimilarity between the baseline model $M^*(t_1)$ trained using the data collected until time t_1 and the model $M^*(t_1 + \Delta t)$ updated considering the data collected between times t_1 and $t_1 + \Delta t$, is the distance $d(\ _{t_1}, \ _{t_1+\Delta t})$ between the matrices $_{t_1}$ and $_{t_1+\Delta t}$ (e.g., Euclidean, Mahalanobis, Chebychev, and Cosine distance) [49]. In this paper, the Euclidean distance

$$D(M^*(t_1), M^*(t_1 + \Delta t)) = \sqrt{\sum_{i=1}^{N_\text{h}} \sum_{j=1}^{S} (\ _{t_{1_{i,j}}} - \ _{t_1+\Delta t_{i,j}})^2} \tag{1}$$

is used for the following main reasons.

1) Mahalanobis distance requires the knowledge of the distributions of $_{t_1}$ and $_{t_1+\Delta t}$, which are continuously changing in an evolving environment and, therefore, difficult to determine.
2) Chebychev distance considers only the vector component with the largest absolute value of the difference between $_{t_1}$ and $_{t_1+\Delta t}$; therefore, it ignores other vector components that are expected to have remarkable effects on the OS-ELM output.
3) Cosine distance measures the angle between $_{t_1}$ and $_{t_1+\Delta t}$ in an inner product space and does not use the information on the magnitude of the vectors, which, on the contrary, is expected to have influence on the OS-ELM output.

C. Setting the Threshold for Concept Drift Detection

To detect the occurrence of a concept drift, it is necessary to fix a threshold, Th, such that a concept drift is detected when $D(M^*(t_1), M^*(t_1+\Delta t)) > Th$. Given the difficulty of directly setting a threshold value, we estimate the relationship between a generic threshold value, Th, and the performance which would be obtained by applying a strategy in which the ML model, M, is updated only when the dissimilarity measure in (1) exceeds that threshold Th. This performance is estimated by considering a validation set formed by N_va batches of data collected in the time interval $[t_1 - L_w + 1, t_1]$, where L_w is the length of the validation window. For this purpose, we build an additional OS-ELM model $M^*(t_1 - L_w)$ trained with the data collected before the validation set, i.e., $(\boldsymbol{x}_{1:t_1-L_w}, \boldsymbol{y}_{1:t_1-L_w})$, we apply the proposed concept drift detection method from time $t_1 - L_w + 1$ to time t_1 considering the threshold Th, and we estimate the corresponding model accuracy TA_{Th} on the data $[t_1 - L_w + 1, t_1]$. In practice, TA_{Th} is the accuracy obtained when the OS-ELM model is updated only when the model dissimilarity between consecutive batches $D(M^*(T), M^*(T + \Delta t))$ with $T > t_1 - L_w$ exceeds the threshold Th. Finally, we compute the performance metric $Gain$ associated with the considered threshold Th

$$Gain(Th) = (TA_{Th} - TA_N)/(TA_A - TA_N) \tag{2}$$

where TA_A is the accuracy obtained when the OS-ELM model is updated each time a new batch becomes available, and TA_N is the accuracy obtained by the model $M^*(t_1 - L_w)$ without performing any model updating. Note that TA_A is the best accuracy that can be obtained on the validation set since the model is always updated with new data, whereas, on the contrary, TA_N is the worst accuracy which can be obtained, since the model is never updated, and TA_{Th} is expected to be an intermediate value between TA_N and TA_A. Although $Gain$ is computed considering OS-ELM models, it is considered representative of the performance variation of the ML model M trained with the two different data sets.

The procedure is repeated considering multiple, decreasing values of Th. The initial value of Th is set to

$$Th_0 = \max(D(M^*(t_1 - L_w), M^*(T))) + \epsilon \tag{3}$$

with $T \in [t_1 - L_w + 1, t_1]$ and ϵ a small value with respect to the distance range. When Th_0 is used, no model updating

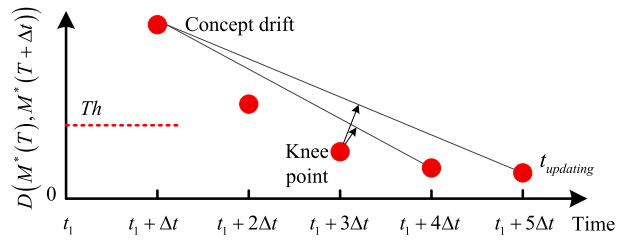

Fig. 2. Example of application of the method for the identification of the knee point and t_{updating}.

is performed in the time interval $[t_1 - L_w, t_1]$ and, therefore, $TA_{Th} = TA_N$ and $Gain = 0$. The value of Th is progressively reduced until a value of $Gain$ close to a desired $Gain^*$ is obtained.

Note that, setting $Gain^*$ is an easier task than directly setting Th, since the ultimate practical objective of the concept drift detection is to inform that the model M should be updated to keep a satisfactory prediction accuracy. The proposed procedure is based on multiple model developments, which is feasible given the low computational effort required for a single OS-ELM model updating. The Appendix reports the detail of the developed algorithm for setting the threshold Th value.

D. Determination of the Amount of Data for Model Updating

Once a concept drift is detected, i.e., $D(M^*(t_1), M^*(t_1 + \Delta t)) > Th$, and the ML model M should be updated. A practical decision which should be taken is the identification of the optimal set of data to be used for model updating. In many practical cases, although the data collected between t_1 and $t_1 + \Delta t$ allow detecting the occurrence of a concept drift, they do not contain enough information for updating M in such a way that it is accurate in the new environment. Updating should be performed only at the time t_{updating}, when enough batches of data from the new environment have been collected.

The procedure proposed in this paper for the identification of t_{updating} is based on the idea that the variation of the performance of M as a function of the number of training data are well approximated by the variation of the performance of the OS-ELM model M^*. Therefore, each time a new batch of data becomes available, we update the OS-ELM model M^* and estimate the model dissimilarity measure introduced in Section III-B. We expect that the dissimilarity after the occurrence of a concept drift, $D(M^*(T), M^*(T + \Delta t))$ with $T > t_1 + \Delta t$, has a decreasing trend with respect to t_1 and that when enough data have been collected from the new environment, the model updating procedure does not significantly change M^*, i.e., $D(M^*(T), M^*(T + \Delta t)) \cong 0$ (Fig. 2). The optimal time t_{updating} for updating M is identified by considering the knee of the obtained $D(M^*(T), M^*(T + \Delta t))$ curve, which can be identified by using the method described in [50]. Fig. 2 shows an example of method application. According to the method, five data batches should be available for its application. First, the farthest point from the line connecting the distance measures computed at times $t_1 + \Delta t$ and $t_1 + 4\Delta t$ is identified as candidate "knee point." In the example shown in Fig. 2, the first candidate "knee point" is the data batch collected at time $t_1 + 3\Delta t$. Then, if the candidate "knee point" is also the farthest point from the line connecting the distance measures computed at times $t_1 + \Delta t$ and $t_1 + 5\Delta t$, as in the case of the example, it is confirmed as knee point of the distance curve; otherwise, the analysis is repeated considering the batch collected at the following times until a knee point is found. The optimal time t_{updating} for the ML model updating is the time at which the knee point is confirmed, e.g., $t_{\text{updating}} = t_1 + 5\Delta t$, in the example shown in Fig. 2, and all the batches of data available at that time are used for both OS-ELM and ML model updating.

E. Computational Cost

The main computational cost of the proposed method is the updating of the OS-ELM model. The space complexity of the method is $O(N_{\text{h}}^2)$, with N_{h} indicating the number of neurons in the hidden layer of the OS-ELM model, dominated by preserving an $N_{\text{h}} \times N_{\text{h}}$ matrix used for OS-ELM updating. The time complexity caused by the matrix multiplications for updating an OS-ELM model is approximately $O(N_{\text{h}}^2)$ [51]. In addition, since the dynamic setting of the threshold Th for concept drift detection requires updating a number of OS-ELM models that, in the worst case, is equal to the number L_w of batches in the validation window, the overall time complexity of the method is $O(L_w N_{\text{h}}^2)$.

F. Parameter Setting

The proposed concept drift detection method requires to set three parameters: the number of hidden nodes of the OS-ELM model N_{h}, the validation window length L_{w}, and the performance $Gain^*$. The influence of the parameter setting on the concept drift performance will be further investigated in the case studies of Section IV.

Considering that ELM models are typically stable with respect to the number of hidden nodes [52], [53] and that our primary interest is not obtaining the best possible OS-ELM accuracy, this parameter does not need to be fine-tuned. In order to reduce the computational burden of the OS-ELM retraining, it can be set to a value of 50.

If the data stream $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ is influenced by periodical factors, such as seasonal effects and work schedules, L_{w} should be equal to an integral multiple of that periodicity. Otherwise, L_{w} is suggested to be set between 20 and 50.

The setting of $Gain^*$ depends on the application requirements: small $Gain^*$ values can be used when a decrease of the model M accuracy is tolerable and does not justify the use of additional computational resources for model updating. In those applications in which high level of accuracy of M is needed, $Gain^*$ should be set to a value larger than 80%.

IV. Application to Synthetic Case Studies

We consider two synthetic case studies, a virtual drift (Section IV-A) and a hybrid drift (Section IV-B) [14],

and compare the results with those of the H-ICI-based CDT [24], [25][1] and the SPLL [28][2] methods.

A. Virtual Concept Drift Case Study

We consider a two-input one-output data set $(\boldsymbol{x}, y)$, where the inputs $\boldsymbol{x} = (x_1, x_2)$ are random quantities sampled from the uniform distribution $U[u_{\min}, u_{\max}]$ with $u_{\min}$ and $u_{\max}$ indicating the lower and upper bounds of $\boldsymbol{x}$, respectively, and the output is $y = \sin(x_1) + \cos(x_2)$.

The occurrence of a virtual drift is simulated by assuming variations of the lower and upper bounds of the uniform probability distribution from which the inputs are sampled. The overall data set is divided into training and test sets for the purposes of implementing the proposed method and verifying its performance, respectively. Data are assumed to become available in batches of 50 patterns. The training set includes 100 batches whose pattern input values are randomly sampled from $U[0,\pi/2]$. In the test set, no concept drift occurs between batches 1 and 60. Three sudden drifts occur at batches 61, 121, and 181, from which patterns start to be sampled from the uniform distributions $U[\pi/4, 3\pi/4]$, $U[\pi/2, \pi]$, and $U[3\pi/4, 5\pi/4]$, respectively. A gradual drift occurs between batches 241 and 260 and is designed according to the guidelines in [54]: the probability p that a pattern is sampled from the new distribution $U[\pi, 3\pi/2]$ gradually increases with time according to the sigmoid function $1/(1+e^{-4(t-12500)/1000})$. An incremental drift occurs between batches 301 and 340, during which patterns are sampled from the uniform distribution $U[\pi + (t - 15\,000)\ \pi/8000, 3\pi/2 + (t - 15\,000)\ \pi/8000]$, whose lower and upper bounds are linearly increasing with time. The input–output relationship is not modified, in accordance with the definition of virtual concept drift [14].

The proposed method has been applied setting $L_{\mathrm{w}} = 20$, $Gain^* = 80\%$, and $N_{\mathrm{h}} = 50$. Fig. 3 shows the concept drift detection results and the corresponding evolution of the concept drift detection threshold Th, which is automatically computed according to the procedure of Section III-C. The three sudden drifts are detected at test batches 61, 121, and 181, which correspond to the first batches of data available after the concept drifts occurrences. The gradual drift is detected at test batch 241, which is the first batch of data sampled from the new distribution $U[\pi, 3\pi/2]$ with probability p. No other concept drift is detected in test batches 242–260, since the model is updated at batch 245 with sufficient information from the new environment. The incremental drift is detected for the first time at test batch 306 with a delay of five batches caused by the small variations of the x_1 and x_2 probability distributions in the first batches after the incremental drift onset. Since during an incremental drift the data are generated from a continuously changing environment, other drift occurrences are detected at batches 317 and 332.

[1]The code of H-ICI-based CDT in the 2017 version is available at http://home.deib.polimi.it/boracchi/Projects/Hierarchical-CDTs.html

[2]The code of SPLL is available at https://github.com/LucyKuncheva/Change-detection

TABLE I
VARIATION OF THE CONCEPT DRIFT DETECTION PERFORMANCE WITH RESPECT TO THE PARAMETER SETTING

L_w	$Gain^*$	N_h	Incremental drift		False detection
			Missed detection	Delay	
20	80%	50	0	5	0
40	80%	50	0	6	0
80	80%	50	0	**22**	2
20	10%	50	0	**12**	0
20	90%	50	0	4	5
20	80%	30	0	6	2
20	80%	200	0	6	0

With respect to the threshold automatically set by the method and used for the concept drift detection, we note the following.

1) The threshold values are very small from batch 1 to batch 60. This is due to the fact that the OS-ELM has been trained using batches of data sampled from the same probability distribution, and therefore, one additional batch of data sampled from the same distribution is not perturbing the OS-ELM model weights.

2) From batch 61 to batch 66, which corresponds to the time t_{updating} at which enough information on the new environment has been collected for the ML updating, the concept drift detection test is not applied.

3) After batch 67, the threshold value starts to be newly dynamically set according to $Gain > 80\%$ in the validation window.

4) After batch 82, the threshold values are lower than those between batches 67 and 82. This is due to the fact that the batch in which the concept drift occurs exits from the validation window used for the threshold setting. The same behavior of threshold increase after concept drift detection and successive stabilization is repeated on the whole test set.

1) Sensitivity to Parameter Setting: The number of hidden nodes N_{h}, the validation window length L_{w}, and the performance requirement gain $Gain^*$ are varied, one at a time.

Given different characteristics of the concept drift types, we consider a concept drift as "missed" if it is not detected within five batches in case of sudden drifts, 20 batches in case of gradual drifts, and 40 batches in case of incremental drifts. A detection is considered "false" when it is performed outside the above listed intervals of batches. Detection delays are computed considering the number of batches necessary for the first detection after the drift onset.

Table I reports the concept drift detection performances in terms of missed detection and detection delay of incremental drifts and false concept drift detection using different parameter combinations. All sudden and gradual drifts are correctly detected within the batch in which they occur for all combinations of the parameters in Table I, except for a gradual drift missed detection obtained for $L_w = 20$, $Gain^* = 10\%$, and $N_{\mathrm{h}} = 50$. With respect to the length of the validation window L_w used to set the threshold, one can note that $L_w = 20$ and $L_w = 40$ provide similar performances, whereas the detection delay significantly increases when $L_w = 80$ in case of incremental drift. This is due to the fact that

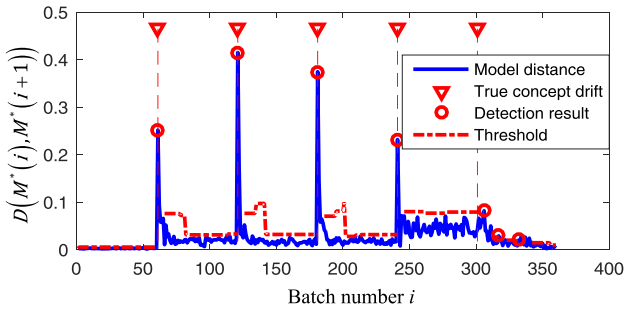

Fig. 3. Evolution of the distance between successive OS-ELM models (continuous line) and of the threshold, Th, used for the concept drift detection (dotted line).

the batch in which the last gradual concept drift is detected occurs 60 batches before the occurrence of the incremental drift. Therefore, since the threshold, Th, used for concept drift detection tends to have large values for L_w batches, as discussed in Section IV-A2 and shown in Fig. 3, the detection of the incremental drift is delayed. With respect to $Gain^*$, small gains cause large thresholds that cause missed alarms, whereas large gains cause small thresholds that cause false alarms. Finally, variations of the number of hidden nodes N_h are shown to have minor effects on the detection performance.

In summary, both L_w and $Gain^*$ influence the concept drift detection: the value of L_w should be relatively small and the value of $Gain^*$ should be relatively large. Since N_h has minor effects on the concept drift detection performance, large N_h values should be avoided to reduce the computational burden.

2) Comparison With Other State-of-the-Art Methods: The parameters of H-ICI-based CDT are automatically set using the training set. The method is individually applied to the signals x_1, x_2, and y, and a concept drift is detected when the test is positive for at least one of the three signals. The parameter K_SPLL of the SPLL method, which indicates the number of clusters, is optimized by applying the receiver operating characteristic analysis with the objectives of reducing the number of missed and false detections. The identified optimal K_SPLL value is 3. The method is applied to detect concept drift batch-by-batch using a fixed window containing data of a reference test batch and a sliding window containing the newly arrived test batch. When a concept drift is detected, the data of the fixed window are replaced by those in the sliding window and the process is repeated.

Table II reports the comparison of the performances in terms of missed and false detections and average detection delay. One can notice that the proposed method achieves small detection delays without false or missed detections.

B. Hybrid Concept Drift

We consider the Mackey–Glass chaotic time series [55]

$$x_{i+1} = p\frac{x_{i-\tau}}{1+x_{i-\tau}^c} + (1-q)\,x_i \tag{4}$$

with $p = 0.2$, $q = 0.1$, $c = 10$, and $\tau = 17$ and the initial values $x_1, x_2, \ldots, x_{18} = 1.2$.

TABLE II
PERFORMANCES ON THE VIRTUAL DRIFT DATA SET

	Proposed method	H-ICI-based CDT	SPLL
Missed detection	0	0	0
False detection	0	4	1
Average delay [batch]	1	4	2.8

TABLE III
RESULTS OF THE PROPOSED METHOD ON THE HYBRID DRIFT DATA (A_j = AMPLITUDE AND f_j = FREQUENCY)

Index	Occurrence time t_j (Batch)	A_j	f_j [Hz]	Time of false detection (Batch)	Detection time (Batch)
1	2460 (25)	0.408	56		2500 (25)
2	4381 (44)	-0.440	46		4400 (44)
3	6617 (67)	0.310	41		6700 (67)
4	11162 (112)	-0.327	36		11200 (112)
5	13148 (132)	-0.376	58	12700 (127)	13900 (139)
6	15096 (151)	-0.465	60	14600 (146)	15400 (154)
7	16556 (166)	-0.451	36	16400 (164)	Missed alarm
8	17013 (171)	-0.473	59		17100 (171)
9	17751 (178)	-0.447	65		17800 (178)
10	18076 (181)	0.308	26		18600 (186)

The first 5000 data of the series are used as a training set of the model M. The concept drifts are injected in the test set by adding cosine time series values C_j, $j = 1, 2, \ldots, 10$, at ten random times t_j sampled from an exponential probability distribution with the rate parameter equal to $1/3000$

$$C_j = A_j \cos\left(\frac{\pi}{250} f_j (i - t_j)\right) \tag{5}$$

where $i = t_j, t_j+1, \ldots, 20000$, the drift amplitude A_j and its frequency f_j are randomly generated from a continuous uniform distribution in the range $[-0.5, -0.3] \cup [0.3, 0.5]$ and a discrete uniform distribution in the range $[25, 75]$, respectively. Note that the generic jth concept drift C_j, $j = 1, \ldots, 10$, starts at time t_j and continues until the end of the test set. Data are assumed to become available in batches of 100 patterns. Table III reports the concept drift occurrence times t_j, $j = 1, \ldots, 10$.

The developed OS-ELM models M^* receive in input x_{i-2}, x_{i-1}, x_i, and provides in output x_{i+1}. Since both input data distributions and input–output relationship change, the simulated drift can be classified as hybrid.

In this case study, L_w has been set to 40, $Gain^*$ to 80%, and N_h to 50. Table IV reports the detection results of the proposed method. It correctly detects nine of the ten concept drifts. In particular, seven of them are detected within few batches after their occurrence (indexes 1–4, 6, 8, 9). The fifth and tenth concept drifts are detected with longer delays due to the false alarm detected at batch 146 and the ninth concept drift is detected at batch 178, just a few batches before the fifth and tenth concept drifts, respectively. As a consequence of the false alarm and the detection of the ninth concept drift, the model M^* is trained using the successive data batches,

TABLE IV
METHOD PERFORMANCE ON THE HYBRID DRIFT DATA SET

	Proposed method	H-ICI-based CDT	SPLL
Missed detection	3	4	2
False detection	6	1	6
Average delay [Batch]	0.3	0.6	0.3

which incidentally contain data collected after the concept drift. Therefore, the OS-ELM does not significantly modify the model M^*, which is partially trained with data from the new environment. Similarly, the seventh concept drift is not identified because a false alarm is detected just two batches before it, leading to a model M^* already trained with several batches from the new environment.

Table IV comparison of the detection results of the proposed method with those of the H-ICI-based CDT and SPLL, whose parameters are optimized using the same procedure described in Section IV-A. The performances of the proposed method are comparable to that of the SPLL, and most of the concept drifts are detected with short delays. H-ICI-based CDT provides less false but more missed detections, which negatively affect the overall ML model performance since when the input–output relationship changes the model is not updated.

V. REAL CASE STUDIES: PREDICTION OF ENERGY PRODUCTION FROM A WIND PLANT

A. Nebraska Weather Prediction

The Nebraska weather data set is compiled by the U.S. National Oceanic and Atmospheric Administration.[3] It contains the daily measurements of eight meteorological features and a binary variable indicating if precipitations occurred during the day. The 18 159 patterns, each one referred to a day, are available, with 31% positive (raining day) and 69% negative (no raining day) patterns.

The data set has been used in [56] as a benchmark for concept drift detection with respect to the problem of classifying, on the basis of the eight meteorological features, whether precipitations occurred during the day. Data are assumed to become available in batches of 30 patterns (days), which approximately correspond to a calendar month. The data are partitioned into a training set containing the first three years (36 batches) and a test set containing the remaining 47 years, 5 months, and 9 days (570 batches, with the last batch formed by nine patterns).

The proposed method has been applied using: $L_{\mathrm{w}} = 12$, $Gain^* = 80\%$, and $N_{\mathrm{h}} = 50$. The choice of $L_w = 12$, corresponding to 1 year of data, is motivated by the seasonality of the weather features. The parameter K_{SPLL} of the SPLL method is set to 3, as suggested in [28].

Since the true concept drift occurrence times are not available in this real-world data set, we assess the performance of different concept drift methods considering the accuracy of an ML classifier updated each time a concept drift is detected. We use the k-nearest neighbors (k-NNs) algorithm with $k = 3$ as ML classifier.

The considered classification performance metric is the accuracy, i.e., the ratio between the number of correct classifications and the total number of classifications.

The following two approaches for model retraining are also considered: 1) model M is never retrained and 2) model M is retrained each time a batch of data is acquired.

The accuracy $Gain_A$ is defined as the ratio between $(\mathrm{Accuracy} - \mathrm{Accuracy}_1)$ and $(\mathrm{Accuracy}_2 - \mathrm{Accuracy}_1)$. Table V reports the obtained classification performances. As expected, approach 1 provides the worst accuracy since it never retrains the k-NN classifier, whereas approach 2, which retrains the k-NN classifier 328 times, i.e., at each batch acquisition, provides the best accuracy. Note that the other methods provide different tradeoffs between the number of retraining and the classification accuracy. The H-ICI-based CDT method provides an unsatisfactory $Gain_A$, whereas the SPLL method is characterized by large computational efforts since it requires to update the model 450 times. The proposed method provides the most satisfactory performance with $Gain_A$ value larger than 70%, and reduced computational efforts with respect to the H-ICI-based CDT and SPLL methods.

B. Electricity Price Prediction

The electricity data set [34], [57] has been collected from the Australian New South Wales Electricity Market between May 7, 1996 and December 5, 1998.[4] It is formed by 45 312 patterns, collected every 30 min. Each pattern contains eight features and the corresponding class label which identifies whether the electricity price is increasing or decreasing with respect to the moving average of the last 24 h.

The data are assumed to become available in batches of 48 patterns (one batch per day) and are partitioned into a training set containing the first eight weeks (56 batches) and a test set containing the remaining 888 batches.

k-NN classifier with $k = 3$ has been used to evaluate the classification accuracy on the test set. The k-NN classifier receives the eight features as input and provides the pattern class as output. It is retrained after each concept drift detection. The parameters of the proposed method are set to: $L_{\mathrm{w}} = 14$, $Gain^* = 80\%$, and $N_{\mathrm{h}} = 50$. The choice of $L_w = 14$ (two weeks) is motivated by the periodicity of the electricity price modifications, caused by the variation of human activities, which typically follow a weekly schedule. Table V reports the obtained results. Note that: 1) the proposed method provides a balanced tradeoff between computational efforts and prediction accuracy, which are both more satisfactory than those obtained by using the SPLL and 2) the $Gain_A$ obtained using the H-ICI-based CDT method is remarkably lower than that obtained by the proposed method.

C. Prediction of Energy Production From a Wind Plant

Since wind plants produce intermittently due to the variation of weather conditions, the capability of predicting in advance

[3]The Nebraska weather dataset is available at http://users.rowan.edu/~polikar/nse.html

[4]The Electricity dataset is available at https://moa.cms.waikato.ac.nz/datasets/

TABLE V
COMPARISON AMONG THE PERFORMANCE OF THE METHODS ON THE NEBRASKA WEATHER AND ELECTRICITY DATA SETS

		Approach 1	Approach 2	The proposed method	H-ICI-based CDT	SPLL
Nebraska weather dataset	*Number of updating*	0	570	**290**	136	450
	Accuracy ($Gain_A$)	73.91 (0%)	76.02 (100%)	**75.44 (72.51%)**	74.79 (41.71%)	75.92 (95.26%)
Electricity dataset	*Number of updating*	0	888	**480**	209	540
	Accuracy ($Gain_A$)	63.72 (0%)	71.07 (100%)	**69.39 (77.14%)**	68.34 (62.86%)	68.98 (71.56%)

TABLE VI
COMPARISON OF THE WIND ENERGY PREDICTION RESULTS PROVIDED BY THE CONCEPT DRIFT DETECTION METHODS

	Approach 1	Approach 2	Approach 3	The proposed method	H-ICI-based CDT	SPLL
Number of retrainings	0	40	3	**21**	23	16
Time to retrain the ML model [minutes]	0	157.7767	37.3833	**79.4931**	93.3083	64.2915
RMSE ($\boldsymbol{Gain_{RMSE}}$)	5.2561 (0%)	5.0855 (100%)	5.1111 (84.99%)	**5.0916 (96.42%)**	5.1029 (89.80%)	5.1003 (91.32%)
MAE ($\boldsymbol{Gain_{MAE}}$)	3.5049 (0%)	3.4710 (100%)	3.4790 (76.40%)	**3.4724 (95.87%)**	3.4814 (69.32%)	3.4840 (61.65%)

their electrical power output is fundamental for ensuring the reliability of electric power distribution. Recently, several data-driven models receiving in input weather forecast data and providing in output the energy production, such as wavelet neural network [10] and BPNN [11], have been developed and applied with success. In this paper, we consider an ML model M used by an energy production company and formed by an ensemble of 100 BPNNs built using the bagging technique [58]. The training of the model is a time-consuming process, which requires the collection and validation of the data and the training of all the 100 BPNNs. As the model accuracy deteriorates with time (due to seasonal effects, climate change, and degradation of the wind plant equipment), periodical retraining of the BPNN models on the most recent data is performed. The objective of this case study is to verify whether the proposed concept drift detection method is able to identify the time instants at which the ML model M should be retrained to obtain a satisfactory tradeoff between accurate energy production predictions and computational time necessary for model retraining.

A data set containing $F = 45$ weather forecast features, $\overrightarrow{WF}$, (e.g., wind speeds, temperatures and pressures at different heights and plant locations, whose detailed characteristics are not given due to confidentiality reasons), and their associated real energy production values, P, collected in the period from January 2011 to April 2016 (64 months) from a wind plant is considered [59]. The forecast horizon is $\Delta t = 96$ h, i.e., at a given time t_0, the weather forecast of the following $\Delta t = 96$ h becomes available. A subset of $F^* = 19$ weather features among the available 45 has been used as input of the BPNN models. The feature selection process has been performed by company experts according to engineering considerations and refined by following a trial-and-error procedure [59].

The available data are partitioned as follows:

1) 2011–2012 data for training the BPNN individual models (4369 patterns, 24 months);
2) 2013–2016 data to verify the performance of the ML model when different strategies for deciding when the model should be retrained are applied (10 924 patterns, 40 months).

1) Application of the Proposed Method: The developed OS-ELM model uses the same input and output quantities of the BPNNs ensemble, i.e., the 19 weather forecast features and the wind energy productions, respectively.

Since the production data are collected monthly, we use batches containing the data collected during one calendar month. The parameters used for concept drift detection are: $L_{\mathrm{w}} = 12$, $Gain^* = 80\%$, and $N_{\mathrm{h}} = 50$. The choice of $L_{\mathrm{w}} = 12$ corresponding to 1 year of data is motivated by the periodicity of the environment changes caused by weather seasonality. Given that only 40 batches are available in the test set, each time a concept drift is detected, one batch of the data instead of five or more batches is used for model updating.

Two performance metrics, namely, root-mean-square error (RMSE) and mean absolute error (MAE) are used to measure the accuracy of the predictions of the ML model M, i.e., the ensemble of 100 BPNNs retrained after the concept drift detections. The RMSE and MAE are average errors of the production prediction: the smaller their values, the more accurate the predictions

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i^* - y_i\right)^2}, \quad \mathrm{MAE} = \sum_{i=1}^{N}\left|y_i^* - y_i\right|/N. \tag{6}$$

The results are compared with those obtained using H-ICI-based CDT and SPLL, considering the following strategies.

1) *Approach 1:* Model M is never retrained.
2) *Approach 2:* Model M is retrained every month; this approach, which is expected to provide the most satisfactory accuracy, is the most demanding from the point of view of the computational efforts.
3) *Approach 3:* Model M is yearly retrained; this approach is currently adopted by the company.

Table VI shows the results obtained by the proposed method and those of the alternative methods. The fourth and sixth rows compare the methods considering the $Gain$ defined by (2),

where the OS-ELM accuracy TA is substituted with the RMSE and MAE metrics.

1) As expected, approach 1 provides the worst accuracy, as it never retrains the BPNNs ensemble M, whereas approach 2, which retrains the ensemble every month, provides the best accuracy since it uses all the available information for each prediction.
2) The proposed method achieves $Gain_{\text{RMSE}}$ and $Gain_{\text{MAE}}$ values similar to those obtained by retraining the model each month (approach 2), and it remarkably reduces the computational efforts. Approach 3, the H-ICI-based CDT and SPLL are all characterized by less satisfactory $Gain_{\text{MAE}}$ and $Gain_{\text{RMSE}}$ values than those of the proposed method. Also, the number of retrainings required by the proposed method (21 retrainings) is less than that required by H-ICI-based CDT (23 retrainings).

In conclusion, the proposed method is able to inform when it is necessary to retrain the model with the newly available data in order to obtain accurate predictions. Furthermore, it is superior to other approaches in terms of computational demand and number of model retrainings.

VI. Conclusion

We have proposed a concept drift detection method based on the analysis of variations caused by new data in an OS-ELM model. The performance of the method in detecting different kinds of concept drifts has been verified using two synthetic case studies, two benchmark real case studies, and an industrial application regarding the prediction of energy production of a wind plant. The results show the superiority of the method with respect to two state-of-the-art methods. Also, retraining the model when the proposed method detects a concept drift allows increasing the prediction accuracy with respect to the periodic model retraining.

The performance of the proposed method depends on three parameters: the number of hidden nodes of the OS-ELM model, the length of the validation window, and the performance requirement. It has been shown that: 1) the performance is not significantly influenced by the number of hidden nodes; 2) the setting of the length of the validation window should consider the periodicity of data; and 3) the setting of the performance requirement depends on the user needs in terms of desired accuracy and computational burden. Future studies will be devoted to investigating the automatic setting of the parameters.

Appendix: Pseudocode for Setting the Concept Drift Detection Threshold

Input: L_w, $Gain^*$, N_{h}, $M^*(t_1 - L_w)$, validation set ($\boldsymbol{X}_{va} = \bigcup_{B=1}^{N_{va}} \boldsymbol{X}_B, \boldsymbol{Y}_{va} = \bigcup_{B=1}^{N_{va}} \boldsymbol{Y}_B$) formed by N_{va} batches of Δt consecutive patterns $\boldsymbol{X}_B = \boldsymbol{x}_{t_1-L_w+(B-1)\Delta t+1:t_1-L_w+B\Delta t}$, $\boldsymbol{Y}_B = \boldsymbol{y}_{t_1-L_w+(B-1)\Delta t+1:t_1-L_w+B\Delta t}$;

Output: Th

```
M^0 = M*(t_1 − L_w);
Th = 0;
for B = 1: 1 :N_va;
    M^0B is obtained by updating M^0 using (X_B, Y_B);
    Compute model dissimilarity D_B(M^0, M^0B);
    Th = max(D_B(M^0, M^0B), Th);
end
Th_0 = Th + 0.01;
if No concept drifts occur in validation set;
    Th = Th_0;
else
    for B = 1: 1 :N_va;
        Get prediction Y_B^A by feeding X_B to M^(B−1);
        Get prediction Y_B^N by feeding X_B to M^0;
        M^B is obtained by updating M^(B−1) using (X_B, Y_B)
    end
    Y_va^A = ⋃_{B=1}^{N_va} Y_B^A, Y_va^N = ⋃_{B=1}^{N_va} Y_B^N;
    TA_A = RMSE(Y_va^A, Y_va), TA_N = RMSE(Y_va^N, Y_va); (6)
    Th= Th_0, Gain = 0;
    while Gain < Gain*
        Th = Th − 0.01*Th_0
        for B = 1: 1 :N_va;
            Get prediction Y_B^Th by feeding X_B to M^(B−1);
            M^B is obtained by updating M^(B−1) using (X_B, Y_B)
            if D(M^(B−1), M^B) < Th;
                M^B = M^(B−1);
        end
    end
    Y_va^Th = ⋃_{B=1}^{N_va} Y_B^Th, TA_Th = RMSE(Y_va^Th, Y_va);
    Use (2) to compute Gain;
    end
end
```

References

[1] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Analyze Future*, vol. 2007, pp. 1–16, Dec. 2012. [Online]. Available: https://www.emc-technology.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf

[2] Z.-H. Zhou, N.-V. Chawla, Y. Jin, and G. J. Williams, "Big data opportunities and challenges: Discussions from data analytics perspectives [discussion forum]," *IEEE Comput. Intell. Mag.*, vol. 9, no. 4, pp. 62–74, Nov. 2014.

[3] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. Burlington, MA, USA: Morgan Kaufmann, 2016.

[4] L. M.-S. Ni, J. Xiao, and H. Tan, "The golden age for popularizing big data," *Sci. China Inf. Sci.*, vol. 59, Oct. 2016, Art. no. 108101.

[5] J.-K. Park, B.-K. Kwon, J.-H. Park, and D.-J. Kang, "Machine learning-based imaging system for surface defect inspection," *Int. J. Precis. Eng. Manuf.-Green Technol.*, vol. 3, no. 3, pp. 303–310, 2016.

[6] S. Al-Dahidi, P. Baraldi, F. Di Maio, and E. Zio, "A novel fault detection system taking into account uncertainties in the reconstructed signals," *Ann. Nucl. Energy*, vol. 73, pp. 131–144, Nov. 2014.

[7] P. Baraldi, F. Cannarile, F. Di Maio, and E. Zio, "Hierarchical k-nearest neighbours classification and binary differential evolution for fault diagnostics of automotive bearings operating under variable conditions," *Eng. Appl. Artif. Intell.*, vol. 56, pp. 1–13, Nov. 2016.

[8] Y. Wang, Q. Miao, E. W. M. Ma, K. L. Tsui, and M. G. Pecht, "Online anomaly detection for hard disk drives based on Mahalanobis distance," *IEEE Trans. Rel.*, vol. 62, no. 1, pp. 136–145, Mar. 2013.

[9] S. Al-Dahidi, F. Di Maio, P. Baraldi, and E. Zio, "Remaining useful life estimation in heterogeneous fleets working under variable operating conditions," *Rel. Eng. Syst. Saf.*, vol. 156, pp. 109–124, Dec. 2016.

[10] H. Chitsaz, N. Amjady, and H. Zareipour, "Wind power forecast using wavelet neural network trained by improved Clonal selection algorithm," *Energy Convers. Manage.*, vol. 89, pp. 588–598, Jan. 2015.

[11] H. Quan, D. Srinivasan, and A. Khosravi, "Short-term load and wind power forecasting using neural network-based prediction intervals," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 303–315, Feb. 2014.