Information Sciences 430–431 (2018) 261–281

Contents lists available at ScienceDirect

Information Sciences

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

Knowledge-maximized ensemble algorithm for different types of concept drift

Siqi Ren[a], Bo Liao[a,*], Wen Zhu[a], Keqin Li[a,b]

[a] College of Information Science and Engineering, Hunan University, Changsha 410082, Hunan, China
[b] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 25 June 2016
Revised 19 November 2017
Accepted 22 November 2017
Available online 22 November 2017

Keywords:
Concept drift
Data stream mining
Ensemble classifier
Unlabelled data

ABSTRACT

Knowledge extraction from data streams has attracted attention in recent years due to its wide range of applications, including sensor networks, web clickstreams, and user interest analysis. Concept drift is one of the most important research topics in data stream mining. Many algorithms that can adapt to concept drift have been proposed. However, most of them specialize in only one type of concept drift and can rarely be used in the environments with a large number of unavailable sample labels. In this study, we propose a new data stream classifier called knowledge-maximized ensemble (KME). First, supervised and unsupervised knowledge are leveraged to detect concept drift, recognize recurrent concepts, and evaluate the weights of ensemble members. Second, the preserved labelled instances in past blocks can be reused to enhance the recognition ability of the candidate member. The final decision for an incoming observation is derived from all the prediction results of the component classifiers. Accordingly, the maximum utilization of the relevant information in a data stream can be achieved, which is critical to models with limited training data. Third, KME can react to multiple types of concept drift by combining the mechanisms of online and chunk-based ensembles. Finally, we compare KME with eight state-of-the-art classifiers on several synthetic and real-world datasets. The comparison demonstrates the effectiveness of KME in various types of concept drift scenarios.

1. Introduction

1.1. Motivation

In today's information society, traditional data mining algorithms need to learn from a huge amount of data by means of restricted memory. These algorithms normally require multiple scans of training data, which is unsuitable for mining high-speed data streams [13]. The widespread dissemination of streaming data in many critical real-time tasks has led to a wide range of attention focused on streaming models. Due to the generation speed and the size of data items, it is impossible for streaming models to store the entire observations. Only limited knowledge can be used at each time step, which leads to approximate results. The motivation of this study is to make the results of incremental learning and bath processes as similar as possible by maximizing the usage of relevant knowledge in data streams.

* Corresponding author.
E-mail addresses: siqirenzl@163.com (S. Ren), dragonbw@163.com (B. Liao), syzhuwen@163.com (W. Zhu), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.ins.2017.11.046
0020-0255/

In addition to the overwhelming volumes and high speed, concept drift is an evident characteristic of streaming data. In many real-world applications, the assumption of a fixed data distribution is not truly maintained, thus making most traditional algorithms infeasible. Past observations may become irrelevant or even harmful for the current concept. Therefore, refining or even rebuilding of models is required to remove the obsolete knowledge. Moreover, if the old data are helpful for the current model in the future, then their necessary information should be stored and reused. In this way, models can capture time-evolving trends in the streaming environment and make critical predictions. The changes in the underlying distribution can be abrupt, gradual, incremental, cyclical, or other. Sudden changes in the data distribution are instantaneous and irreversible. These changes can directly deteriorate the model performance and are therefore easily discovered by detection methods. However, gradual and incremental drifts are much more challenging to discover than abrupt drifts because of their small change rates and overlapping data distributions. If the changes are periodic, then previous concepts can reappear after a period of time. We can utilize the information of recurrent concepts to improve the model performance. This type of change is known as recurrent concept drift.

In reality, datasets are generally complex combinations of many types of concept drift. However, most of the existing streaming classifiers specialize in only one type of change. The goal of this study is to build a streaming classifier that can handle multiple types of concept drift. Data-streaming classifiers are generally categorized into single classifiers and ensembles. In the stationary environment, single classifiers continuously improve their generalization capability over time. However, in the dynamic context, single classifiers must adjust themselves using the received data and cannot leverage relevant past knowledge. Concept-drift detectors, which are treated as forgetting mechanisms, are often equipped with streaming classifiers to discover concept changes. However, most of the drift detectors focus on monitoring the stationarity of supervised information, such as model performance, which results in long detection delays under streaming conditions with few labelled observations. In addition to supervised information, the drift-detection system of the knowledge-maximized ensemble (KME) algorithm exploits the stationarity of feature spaces to discover concept drift in an evolving environment with limited supervised labels in a timely manner. Ensembles are popular methods in the stationary context because of their good generalization capability. Unlike single models, ensembles can leverage relevant information of past ensemble members in dynamic conditions. Therefore, ensembles achieve good accuracies compared with single models because of knowledge transfer. Many approaches have proposed adjustment mechanisms of ensembles, such as modification of the ensemble structures and updating of the aggregation style, to react to new conditions [15,37]. Chunk-based ensembles are designed to deal with gradual drifts by weighting the importance of the component classifiers in the final voting. The tuning of chunk sizes, which involves a stability-plasticity balance problem, is a trivial task [14]. By contrast, online ensembles can rapidly react to sudden drifts by processing data one by one. However, online ensembles are characterized by frequent model updates, thereby incurring high computational cost.

Following these critical motivations, KME combines the mechanisms of chunk-based ensembles and online ensembles to handle different types of concept drift. KME can be regarded as a hybrid ensemble model. Sudden drifts are easily discovered by the drift detection system, and KME is also suitable for incremental, gradual, and recurrent drifts based on the component evaluation and weighting mechanisms. In a streaming environment, change detection and the recognition of recurrent concepts tend to be seriously affected when a limited number of labelled events are available. In KME, the unsupervised knowledge of labelled and unlabelled observations is used to detect changes and to evaluate the equivalence level between two concepts. The utilization of the preserved labelled events of the recurrent concepts can effectively enhance the predictive power of the latest hypothesis. Meanwhile, a weighted result derived from all the component classifiers in the ensemble group is leveraged in the final decision. Accordingly, KME can make full use of relevant information in a data stream.

1.2. Our contributions

In this study, we address the approach to handling multiple types of concept drift and the maximum use of knowledge under conditions where few supervised labels can be acquired. The main contributions of our work can be summarized as follows:

1. Our main contribution is the introduction of a hybrid ensemble that leverages the operators of online ensembles and chunk-based ensembles. Every component classifier is first built over a data chunk. A weighted combination of ensemble members is then applied to address gradual, incremental, and recurrent drifts. By making full use of relevant information in a data stream, KME can produce a model with good generalization ability in conditions with limited training examples. Meanwhile, a drift-detector system, which can discover sudden drifts in a timely manner, is equipped with the ensemble framework. Accordingly, KME can handle different types of concept drift.
2. The proposed weight setting is a piecewise exponential function. On the one hand, KME treats the candidate classifier as the best member, disregarding its performance, which is particularly suitable for sudden drifts. Consequently, cross-validation is not required, which is suitable for coping with high-speed data streams. On the other hand, the weights of past hypotheses are evaluated according to their performance on the labelled data in the most recent chunk and the stationarity of feature spaces. Compared with existing weighting methods, the importance evaluation of members in KME can comprehensively describe the corresponding concepts.

3. We propose a concept-drift-detection system that monitors the stationarity of feature spaces and the prediction power of the classifier. The supervised knowledge reflects the capability of a component classifier to react to the conditional change, whereas the unsupervised knowledge describes the stationarity of the feature distribution. If supervised examples are insufficient, then a detector based on supervised knowledge may have a long time delay. Therefore, we need to monitor the stationarity of feature spaces. However, a detector based on unsupervised knowledge cannot address changes that do not affect the input data distribution, even if the changes substantially deteriorate the performance of the classifier. Therefore, in KME, two detectors asynchronously operate on subsequences of data.
4. We present a recurrent concept recognition method that considers supervised and unsupervised knowledge in the pairwise comparisons of concepts. The preserved labelled instances in the recurrent concepts can be reused to enhance the predictive accuracy of the candidate component classifier, which is critical to scenarios with few supervised observations.
5. Through the Massive Online Analysis (MOA) framework [5], we conduct experiments to analyse the influence of the parameters in KME. The statistical results show that the performance of KME is robust against chunk size. A larger value of the decay factor in the weighting function can significantly improve the predictive accuracy. Furthermore, a comparative experiment is conducted to evaluate the performance of the proposed algorithm. Through the statistical analysis, we conclude that KME can react to different types of concept drift in an environment with a limited number of supervised labels.

The data in many tasks are high dimensional, and the cost of obtaining labels is high, which makes learning tasks complicated [43]. In KME, labelled data are always unavailable. Our goal is to make the best of the beneficial knowledge in the source domain to improve the generalization ability in the target domain by reusing the information of previous hypotheses and the labelled data of recurrent concepts. Therefore, the expansion of supervised information is derived from the knowledge of the past chunks, which is different from the semi-supervised strategy based on a labelling process. We do not need to label any category of unlabelled data, which can avoid the labelling cost. Moreover, unsupervised knowledge is utilized to weight components, recognize recurrent concepts, and detect changes. Accordingly, the maximum utilization of relevant information is achieved in KME. Very fast decision tree (VFDT) [21], which is usually leveraged to learn a decision tree from a time-varying and high-speed data stream, can be regarded as the base classifier of the proposed algorithm.

1.3. Paper organization

The rest of this paper is organized as follows. Section 2 briefly reviews related work about concept drift, the handling mechanisms of concept drift, and the recognition of recurrent concepts. Section 3 provides an outline of KME and analyses the inherent mechanisms. The algorithm is then analysed and evaluated on real and synthetic datasets in Section 4. Finally, Section 5 presents conclusions and proposals for future work.

2. Related work

2.1. Concept drift

In the field of data stream classification, it is usually unrealistic to obtain complete examples to train a classifier in advance: examples continuously arrive in the form of a stream. Meanwhile, the changes observed in the underlying data distribution are called concept drift [3]. In reality, changes in target concepts are often caused by changes in the hidden context. Examples of real-life concept drift include junk mail recognition, monitoring systems, network intrusion detection, and automatic control systems [2,25].

Based on Bayes' theorem, $P(y|x) = P(x|y) \times P(y)/P(x)$, real concept drift involves a change in the probabilistic value $P(y|x)$. First, $P(x)$ describes the input data distribution, and changes in the feature space can be monitored based on $P(x)$. Although this type of change may lead to a shift of the true decision boundaries, identification of the changes affecting $P(x)$ is insufficient. A change in $P(x)$ is called virtual concept drift [3]. Second, $P(x|y)$ is the class-conditional probability, which describes the likelihood of observing a data point within a specific class. Additionally, $P(y)$ is the prior probability, which can be used to measure the state of a class distribution. Finally, the dynamic nature of $P(y|x)$ is called real drift. This type of change directly affects the performance of classifiers, resulting in poor predictive ability. Regardless of the type of concept drift, models need to be adjusted to adapt to new conditions over time.

In addition to classifying concept drift as real or virtual, concept drift is often classified in terms of speed and cyclical nature. First, sudden drift, which is also called abrupt drift, occurs when a new concept suddenly replaces the old one. Second, incremental and gradual drifts refer to slow changes in the data-generating process. These two types of drift are difficult to monitor because a period of uncertainty exists between two adjacent concepts. Finally, recurrent drift refers to a temporary change in the target concept. When a concept reappears, reusing previous knowledge can enhance the learning process of the streaming classifier. Moreover, noise should not be regarded as concept drift because it represents insignificant and random fluctuation. Real-world datasets are often complex combinations of different types of drift. Therefore, adaptability with regard to multiple types of concept drift is crucial in streaming models.

2.2. Handling mechanisms for data streams with concept drift

Many mechanisms, which can be classified as single classifiers, ensemble classifiers, active classifiers, and passive classifiers, have been proposed to address concept drift in data-streaming classification [18,29]. We describe several effective methods for reacting to concept drift in data-generating processes.

For single classifiers, adaptive mechanisms, such as windowing techniques and drift detectors, are applied to adjust the current model. In the category of active classifiers [1,19,31], models based on trigger mechanisms are generally used to handle concept drift. In [1,19], a concept change is identified if the error rate exceeds a fixed threshold. Gama et al. [19] proposed the drift detection method (DDM), which is suitable for abrupt drifts. Baena-Garca et al. [1] proposed the early drift detection method (EDDM) based on DDM, which depends on the distances of the error rates rather than classification error. EDDM performs well on gradual drifts but is sensitive to noise.

Another adaptive mechanism is the windowing technique, which implements a simple forgetting strategy by removing obsolete data in the window. The sliding window, which considers the most recent examples as training data, is the most widely used windowing mechanism. However, the size of the sliding windows is often difficult to determine. If the window is too large, then the model is likely to contain changes in the window, especially for abrupt drifts. If the window is too small, then the model may lack training data, especially in a stationary period. Several adaptive sliding windows, such as weighted window [11], adaptive sliding windowing algorithm (ADWIN) [4], and unified instance selection algorithm (FISH) [44], have been proposed.

Single classifiers must constantly revise themselves to adapt to the new environment. By contrast, by manipulating the component weights or modifying ensemble structure, ensembles can capture dynamic concepts without rebuilding themselves. Compared with single classifiers, ensembles require substantial time and memory consumption, but the accuracy improvement they achieve is usually marginal because of knowledge transfer. Two general types of ensembles are available under nonstationary conditions. Chunk-based ensembles, whose blocks are divided in advance, are not pure online models and are usually unable to quickly react to sudden changes because obsolete knowledge exists in the past component classifiers. Additionally, the determination of the block sizes is a tradeoff between accurate predictions and rapid reaction to changes. Online ensembles process a data stream in an instance-by-instance manner and can be considered to be pure online models. However, batch-based ensembles are readjusted according to a large amount of data each time. Therefore, online ensembles tend to consume substantial amounts of time compared with chunk-based ensembles.

The accuracy weighted ensemble (AWE) [37] is a famous chunk-based ensemble that trains component classifiers on consecutive data chunks and leverages the latest chunk to evaluate all the existing components. Several best members are selected in the final voting. In addition, the weight of each component is based on the mean square error (MSE) based on the observations in the most recent chunk. Street and Kim [36] presented a similar model called the streaming ensemble algorithm (SEA), which leverages a different pruning strategy from that of AWE. SEA replaces the weakest member with the new component. Learn++.NSE [14] learns from consecutive batches of data without making any assumptions on the nature of the drift. The performance of block-based ensembles largely relies on the size of the data chunks. The complexity of online ensembles is always much higher than that of block-based ensembles. Online bagging and online boosting [33] are derived from their batch versions [8,17] and incrementally provide each instance for a component k times, where k is defined by the Poisson distribution. The weighted majority algorithm (WMA) [28] leverages the weighted voting of the results derived from a pool of prediction algorithms. Dynamic weighted majority (DWM) [24] is another typical ensemble that complies with a pure learning rule. A set of incremental classifiers is weighted based on predictive ability after each incoming example. Whenever a component makes a mistake, its weight is decreased by a user-defined factor. When necessary, a new ensemble member is added to the ensemble. Anticipative and dynamic adaptation to concept change (ADACC) [22] is also regarded as an online ensemble that uses a new second-order learning mechanism to react to the dynamic environment.

Adaptive classifier ensemble (ACE) [32] and accuracy updated ensemble (AUE) [10], which are hybrid ensembles, were proposed based on the characteristics of the two types of ensembles. Hybrid systems, which provide a unified framework to describe both the continuous and discrete dynamic processes by means of two distinct types of systems, have been proposed in control sciences [16,41]. Hybrid ensembles integrate two systems to handle a combination issue. By assimilating the weighting mechanisms and the evaluation of component classifiers into online ensembles, KME can address multiple types of drift. ACE reacts to abrupt drifts by monitoring the error rate of a single classifier and then updates the model using a fixed-size data block. AUE incrementally updates each past ensemble member with observations in the most recent block. The online accuracy updated ensemble (OAUE) [9] is an incremental algorithm derived from AUE, which trains and weights every component classifier after every observation.

We present a knowledge-maximized hybrid ensemble that includes a concept-drift-detection system, a novel weighting technique, and an automatic chunk-separation method. First, the drift-detection system monitors the variations in the feature space and classification performance. That is, KME utilizes both supervised and unsupervised knowledge to achieve a detection system with a short time delay. Therefore, KME is highly sensitive to sudden changes. Second, KME implicitly adapts to slow changes through an ensemble of experts, which can achieve knowledge transfer between consecutive chunks. The weighting method of the proposed algorithm is a piecewise mechanism. The piecewise function has been applied to the automatic control system, which is called piecewise affine (PWA) system [16]. It consists of some subsystems covering the global model with high complexity. In KME, the weighted result of several component classifiers is used to predict the labels of incoming data. For past classifiers, weights are based on the predictive capability and input data distribution. The weight

of the current component is given by the supreme value without considering its performance. In this way, we reduce the computational cost, and the current member can have substantial voting power. Similar to chunk-based ensembles, data chunks need to be divided beforehand if the current training data are sufficient. Ensemble pruning can also be considered to remove obsolete members and make the ensemble adapt to the near-future concept in time.

2.3. Recognition of recurrent concepts

Many streaming classifiers that can handle recurrent concepts have been proposed. These classifiers usually store valuable information of recurrent concepts and reuse the information in the target domain as necessary. Widmer and Kubat considered a series of floating rough approximation (FLORA) algorithms, in which only FLORA3 [40] could handle recurrent concepts. FLORA3 stores every concept description and reuses them if previous contexts reappear. The recognizing and treating recurrent (RTRC) system [38] is an ensemble model that maintains a group of concepts derived from data chunks. In [23], concept vectors are extracted from every data chunk; then, a clustering algorithm is applied to the concept vectors, whose similarity is measured by the Euclidean distance. The best member selected from the pool of components is considered to be the decision model. In [34], the final components are selected according to their performances evaluated on the observations in the most recent block. Yang et al. [42] proposed a proactive method to discover recurrent concepts from the history of concepts. This method considers historic concepts as a Markov chain and then selects the best classifier based on a given transition matrix. A nonparametric multivariate statistical test is employed to compare two concepts in recurring concept drifts (RCD) [20]. The contextual information can be applied to recognize recurrent concepts. The contextual information within a data chunk without any drift is often assumed to be stationary. The notions of primary, contextual, and context-sensitive attributes were proposed in [39].

The use of recurrent concepts is beneficial to maximize the use of pertinent knowledge, especially when the condition lacks supervised observations. However, the preceding solutions do not address the issue of recurrent-drift identification under conditions with rare labelled observations. KME measures the equivalence levels of two concepts based on both supervised and unsupervised knowledge. On the one hand, KME evaluates the accuracy of every past classifier on the labelled observations in the most recent data chunk. On the other hand, KME compares the feature spaces between every past chunk and the current chunk. The similarity of the feature distribution between two concepts is based on unsupervised knowledge. In this way, KME effectively alleviates the situation of rare labels in recognizing recurrent concepts.

3. The knowledge-maximized ensemble (KME) approach

We propose a hybrid ensemble classifier that can address multiple types of concept drift in conditions with few labels by maximizing the use of the relevant knowledge in a data stream. This technique, called KME, includes an operator to detect changes, an operator to recognize recurrent concepts, an operator to evaluate the weights of ensemble members, and an ensemble update mechanism. First, sample statistics are derived from observations in the sliding windows of consecutive blocks. The description of each statistic is provided in Section 3.1. Second, the concept-drift-detection system monitors the stationarity of the statistics to recognize concept drift. The procedure of drift detection is described in Section 3.2. Third, after a change is discovered or sufficient training observations are obtained, a new block is produced. Then, the recurrent concepts are recognized, as illustrated in Section 3.3. The labelled samples of the recurrent concepts are reused to train the candidate component classifier. Section 3.4 describes the ensemble update procedure in KME. Fourth, a weighted voting of every ensemble member is used to predict the label of a testing instance. The weights of the candidate hypothesis and previous component classifiers are evaluated in Section 3.5. Finally, the pseudocode of the KME algorithm is presented in Algorithm 3.5, and details of the procedure are provided in Section 3.6.

The notations used in this paper and their descriptions are summarized in Table 1.

3.1. Concept description

The KME algorithm consists of a series of concepts derived from consecutive data chunks. All the data chunks in a data stream can be represented as $C = \{C_1, \ldots, C_N\}$. Each data chunk is composed of four critical members that describe the characteristics of the corresponding concept. The first member of the ith data chunk is the supervised sample set Z_i. The drift-detection mechanism handles the upcoming instances at the window level. In fact, the detection of concept changes based on an independent example is inappropriate, especially in noisy conditions. The neighbouring windows containing a certain number of observations in a data chunk are nonoverlapping. The detection mechanism of KME treats the observations in each sliding window as primitives, thereby effectively avoiding the influence of noisy data. With the aid of a series of small sliding windows, the drift-detection system can gradually approach a possible change. Accordingly, KME can identify concept drift with a short time delay.

The sample mean and sample variance, which are, respectively, regarded as the second and third members of concepts, can be utilized to evaluate the stationarity of the feature space. The unsupervised statistics are extracted from the nonoverlapping sliding windows of a data chunk to approximate the input data distribution $P(x)$. First, we present the definitions of the unsupervised statistics for concept-drift detection.

Algorithm 1 Knowledge Maximized Ensemble(KME).

Input: S: data stream of examples, m: ensemble size, d_{max}: predefined chunk size
Output: $B(x_t)$: Prediction label of the unlabelled instance x_t

$B \leftarrow \emptyset$, $t \leftarrow 1$, $i \leftarrow 1$, $m_1 \leftarrow 0$, $m_2 \leftarrow 0$;
while x_t is provided **do**
  $m_1 \leftarrow m_1 + 1$;
  if y_t is available **then**
    $Z_i \leftarrow Z_i \cup \{(x_t, y_t)\}$;
    $m_2 \leftarrow m_2 + 1$;
    update K_i using (x_t, y_t);
    if $m_2 = m_l$ **then**
      obtain $\hat{e}$ based on Eq. (7) and add it to $E_{i,s}$;
      $r_2 \leftarrow \mathrm{TEST}_l(E_{i,s})$;
      $m_2 \leftarrow 0$;
    end if
  end if
  if $m_1 = m_u$ **then**
    update $M_{i,s}$ and $V_{i,s}$ using x_t;
    $r_1 \leftarrow \mathrm{TEST}_u(M_{i,s}, V_{i,s})$;
    $m_1 \leftarrow 0$;
  end if
  if $r_1 = 1$ or $r_2 = 1$ or $|t - T_{i,1}| = d_{\max}$ **then**
    $T_{i,end} \leftarrow t$
    provide C_i with observations within the interval $[T_{i,1}, T_{i,end}]$;
    configure K_i based on Z_i;
    $i \leftarrow i + 1$;
    reconfigure $\leftarrow 1$;
  end if
  if reconfigure=1 and $|M_{i,s}| \geq M_u$ and $|V_{i,s}| \geq M_u$ and $|E_{i,s}| \geq M_l$ **then**
    reconfigure TEST_u based on $M_{i,s}$ and $V_{i,s}$;
    reconfigure B' with Z_i^T;
    reconfigure TEST_l based on the classification error of B';
    for $j = 1; j < i; j++$ **do**
      compare C_i and C_j using Eqs. (11) and (12);
      if $dist_{i,j}^{M,V} \leq \gamma$ and $p_j > \tau$ **then**
        $Z_i \leftarrow Z_i \cup Z_j$;
      end if
    end for
    update K_i using Z_i;
    compute weight w_i of K_i based on Eq. (17);
    reconfigure $\leftarrow 0$;
  end if
  for all classifiers $K_j \in B$ **do**
    compute weight $w_{i,j}$ of K_j based on Eq. (16);
  end for
  if $|B| < m$ **then**
    $B \leftarrow B \cup \{K_i\}$;
  else
    substitute the ensemble member of the minimum weight in B using K_i;
  end if
  if y_t is not available **then**
    assign label $B(x_t)$ using Eq. (18);
  end if
end while

Table 1
Notations.

Notation	Description	Notation	Description
S	Data stream of examples	i	Index of data chunks
$P(x)$	Feature probability	$P(y)$ and $P(y_t)$	Prior probability and y_t's class distributions
$P(x\|y)$	Class-conditional probability	$P(y\|x)$	Posterior probability
x_t and y_t	Data point at timestamp t and its label	$T_{i,j}$ and $T_{i,end}$	Timestamps of the jth and the last instances of C_i, respectively
C	Set of data chunks in a data stream	C_i	ith chunk in a data stream
N	Number of data chunks	t	Current timestamp
m_u and m_l	Numbers of observations and labelled instances in each sliding window, respectively	$E_{i,s}$ and $\|E_{i,s}\|$	Average classification error of B' on labelled data of the sth sliding window of C_i and its cardinality
$M_{i,s}$ and $\|M_{i,s}\|$	Mean of samples in the sth sliding window of C_i and its cardinality	$V'_{i,s}$	Variance of samples in the sth sliding window of C_i
$V_{i,s}$ and $\|V_{i,s}\|$	Power-law transformation of $V'_{i,s}$ and its cardinality	Z_i and $\|Z_i\|$	Labelled sample set of C_i and its cardinality
h_0	Exponent of the power-law transformation	k_i	ith cumulation of the distribution of the sample variance
μ and σ^2	Mean and variance of the normal distribution, respectively	$K_i(x_t)$ and $K'_i(x_t)$	Prediction labels of x_t using K_i and K'_i, respectively
X	Set of n i.i.d. random variables	X_i	Random variable
n	Number of random variables	K_i	ith member of the ensemble
$\bar{x}$	Average of n i.i.d. random variables	B'	Classifier used to detect drift
Q	Sequence of m_l labelled samples in Z_i to evaluate the classification error of B'	e and $\hat{e}$	Expected value of the classification error of B' on each labelled instance and its evaluated value
ε_t	Classification error of B' on each labelled instance in Z_i	$B(x_t)$ and $B'(x_t)$	Prediction labels of x_t using B and B', respectively
$B(n, p)$ and p	Bernoulli distribution and the probability of success for each trial	$I^M_{i,s}$, $I^V_{i,s}$ and $I^E_{i,s}$	Confidence intervals of $M_{i,s}$, $V_{i,s}$ and $E_{i,s}$, respectively
I^M_i and I^V_i	Confidence intervals of the sample mean and the power-law transformation of the sample variance in the ith chunk, respectively	$S^M_{i,j}$ and $S^V_{i,j}$	Equivalence levels of the sample mean and and power-law transformation of the sample variance between C_i and C_j, respectively
TEST_u	Concept-drift detector to inspect changes in $P(x)$	TEST_l	Concept-drift detector to inspect changes in $P(y\|x)$
$S^{M,V}_{i,j}$	Equivalence level of the unsupervised knowledge between C_i and C_j	$dist^{M,V}_{i,j}$	Deviation level of the unsupervised knowledge between C_i and C_j
K'_i	Classifier trained on Z^T_i to recognize recurrent concepts	Z^T_i and Z^V_i	Training set and validation set in the ith chunk, respectively
Z^V and $\|Z^V\|$	Common validation set and its cardinality	p_j	Equivalence level of the supervised knowledge between C_i and C_j
γ and τ	Thresholds of Eqs. (11) and (12), respectively	$f^j_{y_t}(x_t)$	Probability that x_t is classified as y_t by K_j
$\text{MSE}_{i,j}$	Mean square error of K_j on labelled instances of C_i	MSE_r	Mean square error of a randomly predicting classifier
$f(.,.)$	Indicator function	u	Decay factor of weights
$w_{i,j}$	Weight of K_j to predict instances in the ith chunk	w_i	Weight of the candidate classifier
m	Predefined ensemble size	$d_{\max}$	Predefined chunk size
m_1 and m_2	Counts of observations and labelled samples in the current sliding window, respectively	B and $\|B\|$	Ensemble model and number of ensemble members in the current ensemble
reconfigure	Logical variable indicates whether to reconfigure the drift detectors or not	r_1 and r_2	Logical variables indicate whether a concept drift exists based on TEST_u and TEST_l, respectively
M_u and M_l	Numbers of $M_{i,s}$ and $E_{i,s}$ to reconfigure TEST_u and TEST_l, respectively	N_l and N_u	Numbers of sliding windows containing labelled and unlabelled instances, respectively
$O(.)$	Computational complexity of the algorithm	b	Number of preserved windows in the current data chunk
d	Number of attributes in a dataset	v	Maximum number of values per attribute
l	Number of leaves in the tree	c	Number of classes
s_l	Number of labelled data in a data chunk	k	Parameter of the Poisson distribution
β	Multiplicative factor of WMA	a and b	Sigmoid slope and sigmoid crossing point of Learn++.NSE
W	Window size in the Prequential test	F_F	Statistic of the Friedman test
α	Significant level	CD	Critical diffidence
h	Frequency that labelled instances are provided	$p_.$	p-values of algorithms in the Wilcoxon signed rank test

Definition 1 (Sample mean)**.** Let m_u be the number of observations in each sliding window. The sample mean evaluated on instances in the sth sliding window of the ith data chunk, denoted by $M_{i,s}$, is defined as

$$M_{i,s} = \frac{1}{m_u} \sum_{t=T_{i,(s-1)m_u+1}}^{T_{i,m_u s}} x_t, \tag{1}$$

where x_t is the data item at timestamp t, $T_{i,(s-1)m_u+1}$ is the start timestamp of the sth sliding window of the ith data chunk, and $T_{i,m_u s}$ is the timestamp of the $(m_u s)$th instance of the ith data chunk.

Definition 2 (Sample variance)**.** Given m_u and $M_{i,s}$, the sample variance of observations in the sth sliding window of the ith data chunk, denoted by $V'_{i,s}$, is formulated as

$$V'_{i,s} = \frac{\sum_{t=T_{i,(s-1)m_u+1}}^{T_{i,m_u s}} (x_t - M_{i,s})^2}{m_u - 1}. \tag{2}$$

Definition 3 (Power-law transformation of the sample variance)**.** Given $V'_{i,s}$, the power-law transformation of the sample variance $V'_{i,s}$, denoted by $V_{i,s}$, is defined as

$$V_{i,s} = (V'_{i,s})^{h_0}, \tag{3}$$

where h_0 is the exponent. h_0 dominates the transformation and should be calculated as

$$h_0 = 1 - \frac{k_1 k_3}{3k_2^2}, \tag{4}$$

where k_i is the ith cumulation of the distribution associated with the sample variance [30].

Second, we introduce the estimation of the members $M_{i,s}$ and $V_{i,s}$ to monitor the stationarity of feature distribution $P(x)$.

Theorem 1 *[35]. Let $X = \{X_1, \ldots, X_n\}$ be a set of n independent and identically distributed (i.i.d.) random variables. Individual X_i is drawn from the distribution with mean μ and variance σ^2. By the law of large numbers, the sample average $\bar{x} = \frac{\sum_{i=1}^{n} X_i}{n}$ should approach a normal distribution with expected value μ and variance σ^2/n.*

Theorem 1 formally states the central limit theorem (CLT), and its proof is available in [35]. In probability theory, the CLT states that the sum of a large number of i.i.d. random variables tends to a normal distribution, even if the original variables are not normally distributed.

Corollary 1. *If m_u is sufficiently large, then we can detect the stationarity of the feature distribution $P(x)$ by assessing variations in the sample mean $M_{i,s}$.*

Proof. $M_{i,s}$ is the sample mean evaluated over observations in the sth sliding window of the ith data chunk. In terms of Theorem 1, $M_{i,s}$ should approximately follow a stable normal distribution with a constant expected value in stationary conditions if the number of observations in each sliding window is sufficient. Accordingly, the drift detector identifies a change in $P(x)$ as soon as the value of $M_{i,s}$ shows variation. □

Corollary 2. *If m_u is sufficiently large, then the stationarity of $V_{i,s}$ can be leveraged to detect concept drift affecting the feature distribution $P(x)$.*

Proof. $V'_{i,s}$, which is the sample variance evaluated over observations in the sth sliding window of the ith data chunk, is not Gaussian distributed. We obtain a new feature $V_{i,s}$ by applying the power-law transformation [30]. In the stationary period, $V_{i,s}$ is approximately Gaussian distributed when a large number of labelled samples are provided in each sliding window. According to Theorem 1, the expected value of $V_{i,s}$ should be a constant; thus, the stationarity of $V_{i,s}$ can be used to detect changes in $P(x)$. □

Third, we define the fourth member $E_{i,s}$ of the ith concept. $E_{i,s}$ can be used as the supervised statistic to evaluate the stationarity of the posterior probability $P(y|x)$.

Definition 4 (Classification error)**.** Let m_l be the number of labelled examples in each sliding window. The classification error $E_{i,s}$ is the performance of classifier B' on supervised instances in the sth sliding window of the ith data chunk. $E_{i,s}$ is defined as

$$E_{i,s} = \left\{ \frac{1}{m_l} \sum_{t \in Q} \varepsilon_t, Q \subset Z_i \right\}, \tag{5}$$

where Q is the sequence of m_l labelled instances in Z_i for estimating the classification error of B'. ε_t is the classification error of B', which is computed from each supervised instance in Z_i

$$\varepsilon_t = \begin{cases} 0, & \text{if } y_t = B'(x_t) \\ 1, & \text{otherwise,} \end{cases} \tag{6}$$

where B' is a specific model trained to detect concept changes and $B'(x_t)$ is its prediction label of x_t. B' is different from the ensemble model of KME. It is an additional classifier used to calculate the average classification error on the supervised data of sliding windows. B' is never updated until a concept change is discovered, even though new data are being received, to ensure that its classification error in stationary conditions is constant.

Finally, we show how to estimate the classification error $E_{i,s}$ to detect the concept drift affecting the posterior probability $P(y|x)$.

Theorem 2 *[7]. Let $X = \{X_1, \ldots, X_n\}$ be a set of n i.i.d. random variables. Individual X_i is drawn from a Bernoulli distribution $B(n, p)$. The parameters p and n are the probability of success for each trial and the number of trails. For sufficiently large n, the sum of n random variables can be approximated with a Gaussian distribution with expected value np and variance $np(1-p)$.*

Theorem 2 is the normal approximation of $B(n, p)$. The approximation is improved when n is large and p is far from the extremes of zero and one. The proof of Theorem 2 is available in [7].

Corollary 3. *If m_l is sufficiently large, then the stationarity of the average classification error $E_{i,s}$ can be leveraged to detect changes in the posterior distribution $P(y|x)$.*

Proof. The classification errors of B' on each labelled instance of the sliding windows can be treated as i.i.d. realizations of a Bernoulli random variable with expected value e. Under stationary conditions, e should be a constant because the classifier B' is never updated. If m_l is sufficiently large, then the Bernoulli distribution can be approximated as a Gaussian distribution with expected value $m_l e$ and variance $m_l e(1-e)$ according to Theorem 2. Therefore, we can assess the stationarity of $E_{i,s}$ to identify changes in $P(y|x)$.

The average classification error $\hat{e}$ can be evaluated on the labelled examples in the sliding window as

$$\hat{e} = \frac{1}{m_l} \sum_{t \in Q} \varepsilon_t . \square \tag{7}$$

3.2. Concept-drift detection

Since the causes of concept drift are rather complex, general detection strategies make use of the consequences of concept drift to identify changes in data streams. A series of indicators, such as the classification performance, the input data distribution, and the relevance of features in the datasets, can be used to detect concept changes in the data-generating process. If these indicators experience a substantial change, then concept drift may be detected. In general, supervised labels are insufficient or cannot be provided under real-world circumstances. Detection mechanisms based on supervised knowledge may suffer a long delay because limited supervised information is provided. By contrast, change detectors based on the stationarity of the feature distribution $P(x)$ cannot discover a concept drift that leaves the feature distribution unaltered, although these changes significantly degrade the performance of classifiers (e.g., the swap of class labels). Therefore, the sample mean, the power-law transformation of the sample variance, and the classification error are regarded as the three features in the proposed concept-drift-detection system. KME can simultaneously monitor the feature distribution $P(x)$ and the posterior probability $P(y|x)$ based on the stationarity of each feature.

On the one hand, the confidence intervals of three features need to be evaluated. The statistics $M_{i,s}$ and $V_{i,s}$ should follow stable normal distributions if the value of m_u is sufficiently large in the stationary environment. Then, the confidence intervals of $M_{i,s}$ and $V_{i,s}$, which are, respectively, denoted by $l^M_{i,s}$ and $l^V_{i,s}$, are computed on observations provided within $[T_{i,1}, T_{i,m_u s}]$. Similarly, the average classification error of B' on the labelled examples of each sliding window can be approximated using a Gaussian distribution when a sufficient number of supervised events are available in each sliding window. Consequently, the confidence interval of $E_{i,s}$, denoted by $l^E_{i,s}$, can be calculated according to the classification error of B' on supervised observations acquired in the interval $[T_{i,1}, T_{i,m_l s}]$.

On the other hand, three features are evaluated on events of the $(s+1)$th sliding window in the ith chunk. $M_{i,s+1}$ and $V_{i,s+1}$, which, respectively, represent the sample mean and the power-law transformation of the sample variance, are derived from observations of the $(s+1)$th sliding window in the ith block. Meanwhile, $E_{i,s+1}$ is the classification error of B' on labelled events of the $(s+1)$th sliding window in the ith block. Two concept-drift detectors, denoted by TEST_u and TEST_l, are designed in the concept-drift-detection system. TEST_u is used to verify the stationarity of the input data distribution to identify virtual concept drift. It manipulates the upcoming sequence in a series of nonoverlapping windows of size m_u. If $M_{i,s+1} \notin l^M_{i,s}$ or $V_{i,s+1} \notin l^V_{i,s}$, then TEST_u reveals a change in the feature distribution $P(x)$. Similarly, TEST_l checks the stationarity of the classification error of an additional classifier B' on the labelled events in the sliding windows. A concept change in $P(y|x)$ is detected if $E_{i,s+1} \notin l^E_{i,s}$.

In KME, TEST_u and TEST_l can asynchronously access variants of each feature. A concept change can be detected in the data stream as soon as any feature shows variation. In the chunk-by-chunk framework, most chunk-based ensembles tend to slowly react to sudden changes because the obsolete classifiers remain valid in the final decision. KME can rapidly respond to a sudden change by equipping a change detection mechanism with the chunk-based framework.

3.3. Recurrent concepts identification

Most of the existing algorithms do not address the problem of recurrent concept drift when a limited number of supervised samples are provided. The labelled instances of recurrent concepts are critical to enhance the model performance in the target domain. Similar to the concept-drift detection mechanism, KME identifies recurrent concepts based on an exhaustive pairwise comparison in terms of the supervised and unsupervised knowledge. The comparison of C_i and C_j $(j = 1, 2, \ldots, i-1)$ determines whether C_j is a recurrent concept of C_i in ensemble B. In general, the identification of recurrent concepts in the existing algorithms exploits only supervised knowledge, such as the model performance on the

labelled instances in the most recent data block. Unfortunately, a large number of supervised observations are not always available in a real-world environment. Accordingly, the unsupervised and supervised knowledge are considered simultaneously to analyse the equivalence level of two concepts.

We use the sample mean, the power-law transformation of the sample variance, and the classification error of a past ensemble member on instances of the latest chunk as three features to recognize recurrent concepts. Thus, the equivalence levels of two concepts based on the supervised and unsupervised statistics are computed. First, we define the equivalence levels of the sample mean and the power-law transformation of the sample variance of two concepts.

Definition 5 *(Equivalence level of the sample mean).* Given C_i and C_j, the equivalence level of the sample mean between C_i and C_j, denoted by $S^M_{i,j}$, is defined as

$$S^M_{i,j} = \frac{l^M_i \cap l^M_j}{l^M_i \cup l^M_j}, \tag{8}$$

where l^M_i and l^M_j are the confidence intervals of the observation means in the ith and jth data chunks, respectively.

Definition 6 *(Equivalence level of the power-law transformation of the sample variance).* Given C_i and C_j, the equivalence level of the power-law transformation of the sample variance between C_i and C_j, denoted by $S^V_{i,j}$, is defined as

$$S^V_{i,j} = \frac{l^V_i \cap l^V_j}{l^V_i \cup l^V_j}, \tag{9}$$

where l^V_i and l^V_j denote the confidence intervals of the power-law transformation of the variance of observations in the latest and the jth data blocks, respectively.

Second, we define the equivalence level of the unsupervised knowledge between C_i and C_j based on $S^M_{i,j}$ and $S^V_{i,j}$.

Definition 7 *(Equivalence level of the unsupervised knowledge).* Given $S^M_{i,j}$, and $S^V_{i,j}$, the equivalence level of the unsupervised knowledge between C_i and C_j, denoted by $S^{M,V}_{i,j}$, is computed as

$$S^{M,V}_{i,j} = \frac{S^M_{i,j} + S^V_{i,j}}{2}, \tag{10}$$

where $S^{M,V}_{i,j}$ is normalized to the interval [0, 1].

Consequently, the deviation level between the ith and jth concepts, with regard to the unsupervised knowledge, becomes

$$dist^{M,V}_{i,j} = 1 - S^{M,V}_{i,j}. \tag{11}$$

If $dist^{M,V}_{i,j}$ exceeds a predefined threshold γ, then C_j is not equivalent to C_i in terms of unsupervised knowledge. However, the aforementioned process is more likely to consider concepts with the same feature values but different categories as recurrent concepts because the sample labels are ignored. Therefore, the supervised knowledge should also be used to evaluate recurrent concepts. Finally, we define the equivalence level of the supervised knowledge between C_i and C_j.

Definition 8 *(Equivalence level of the supervised knowledge).* First, two classifiers $K^{'}_i$ and $K^{'}_j$ $(j = 1, 2, \ldots, i-1)$ are trained over two sets $Z^T_i \subset Z_i$ and $Z^T_j \subset Z_j$, respectively. The number of instances in Z^T_i is equal to that in Z^T_j. Second, the equivalence level between Z_i and Z_j is estimated by comparing the classification error of $K^{'}_i$ and $K^{'}_j$ on a common validation set $Z^V = Z^V_i \cup Z^V_j$, where $Z^V_i \subset Z_i$, $Z^V_i \cap Z^T_i = \emptyset$, $Z^V_j \subset Z_j$, and $Z^V_j \cap Z^T_j = \emptyset$. Based on the couples in the validation set Z^V, the equivalence level of the supervised knowledge between C_i and C_j, denoted by p_j, is calculated as

$$p_j = \frac{\sum_{Z^V} f(K^{'}_i(x_t), K^{'}_j(x_t))}{|Z^V|}, \tag{12}$$

where $|Z^V|$ is the number of examples in the validating set Z^V. $K^{'}_i(x_t)$ and $K^{'}_j(x_t)$, respectively, denote the prediction labels of x_t using $K^{'}_i$ and $K^{'}_j$. $f(K^{'}_i(x_t), K^{'}_j(x_t))$ is an indicator function and can be formulated as

$$f(K^{'}_i(x_t), K^{'}_j(x_t)) = \begin{cases} 1, & \text{if } K^{'}_i(x_t) = K^{'}_j(x_t) \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Finally, Z_i and Z_j are equivalent if p_j exceeds a predefined threshold τ.

The final identification of recurrent concepts is obtained by considering the supervised and unsupervised knowledge. C_j can be considered to be the recurrent concept of C_i when $dist^{M,V}_{i,j} \le \gamma$ and $p_j > \tau$. Then, Z_j can be obtained and used as the

training set of the latest component. In contrast to semi-supervised learning, KME does not need to tune any of the labels for the unlabelled observations in the current data chunk. However, additional supervised observations, which can enhance the predictive ability of the candidate classifier, are extracted from previous chunks in a data stream.

3.4. Concept division and ensemble update

In KME, a new data chunk is obtained when a concept change is identified or when the number of labelled observations in the current block reaches a certain level. Therefore, the KME algorithm is a hybrid model that combines the important operators of online ensembles and block-based ensembles. In the stationary period, KME works in a chunk-by-chunk manner; however, the concept-drift-detection system operates on a sequence of data at the window level. The chunk size cannot be determined in advance and must be adjusted based on the nonstationary environment. As a hybrid ensemble, KME can react to sudden and slow drifts.

First, the concept-drift-detection system monitors the stationarity of the feature space and the classification performance at a small window level, thus gradually approaching the timestamps of concept changes. A new data block is used to train the candidate classifier after detecting a change in the data distribution. Abrupt drifts are easy to detect through an active strategy. However, the drift-detection system usually performs poorly on slow changes because a small variation does not exceed the predefined threshold of the drift detector.

Second, similar to chunk-based ensembles, a new block is obtained as soon as the number of labelled events in the current block is sufficient to train a perfect hypothesis. Consequently, all the ensemble members are preserved, and the useful knowledge about previous data chunks can be reused in the final decision. KME also ensures sufficient reaction to slow drifts through periodic evaluation and weighting mechanisms. However, the weights of obsolete hypotheses cannot immediately decrease to zero; therefore, chunk-based ensembles react slowly to sudden drifts as the obsolete members remain valid in the final decision. The tuning of chunk sizes is difficult. Large chunks are suitable for stationary data streams, but they may contain changes within a block and increase the computational complexity. Small chunks can make the model react quickly to sudden changes, but they lead to poor performance in the presence of slow gradual drifts and periods of stability.

Finally, recurrent concepts can be recognized by evaluating the equivalence level between two chunks. Then, component classifiers derived from labelled events in the recurrent concepts are used to predict the labels of testing events. Furthermore, the preserved labelled instances are extracted to supplement the training set of the candidate component. Therefore, the performance of the ensemble can be improved over time because of the increasing number of labelled examples.

The KME algorithm is an adaptive ensemble based on VFDT [21], which is convenient for preserving and modifying the knowledge of the new concept. In principle, we can use any online learning algorithm as the base classifier of the ensemble.

3.5. Weighting mechanism and final hypothesis

In dynamic environments, a data stream is often generated from mixed concept types and could be considered to be a weighted combination of data distributions characterizing the target concept. KME can achieve knowledge transfer among consecutive data chunks by keeping old experts in the ensemble group, which could improve the ensemble's reactions to slow and recurrent concept drifts.

The weighting mechanism in KME is a piecewise exponential function. To derive the weights of the ensemble members, we begin by defining the mean square error of each past component classifier $K_j \in B$ $(j = 1, 2, \ldots, i-1)$ and a randomly predicting classifier.

Definition 9 *(Mean square error of K_j).* Given Z_i, the mean square error of K_j on instances in Z_i, denoted by $\mathrm{MSE}_{i,j}$, is the predicted squared error of the jth component on supervised data of the new concept C_i. $\mathrm{MSE}_{i,j}$ can be calculated as

$$\mathrm{MSE}_{i,j} = \frac{1}{|Z_i|} \sum_{(x_t, y_t) \in Z_i} (1 - f^j_{y_t}(x_t))^2, \tag{14}$$

where $f^j_{y_t}(x_t)$ is the probability that instance x_t is classified as class y_t by component classifier K_j. $|Z_i|$ is the number of supervised instances in the ith data block.

Definition 10 *(Mean square error of a randomly predicting classifier).* A random model dose not contain useful information for predicting a testing observation. The mean square error of a randomly predicting classifier, denoted by MSE_r, depends on the current class distribution and can be formulated as

$$\mathrm{MSE}_r = \sum_{y_t} P(y_t)(1 - P(y_t))^2, \tag{15}$$

where $P(y_t)$ is the y_t's class distribution.

We next formulate the weights of K_j and the candidate classifier based on Definitions 9 and 10, receptively.

Definition 11 *(Weight of K_j).* Given MSE_r, $\mathrm{MSE}_{i,j}$, and $dist_{i,j}^{M,V}$, the weight of a past component classifier K_j in the ensemble group, denoted by $w_{i,j}$, is expressed by

$$w_{i,j} = e^{-u\times\left(\mathrm{MSE}_r+\mathrm{MSE}_{i,j}+dist_{i,j}^{M,V}\right)}, \tag{16}$$

where $u \in [0, 1]$ is the fading factor of the weight. MSE_r is regarded as a reference to the current class distribution because a random prediction cannot contribute to the final decision of the ensemble. $dist_{i,j}^{M,V}$, $\mathrm{MSE}_{i,j}$, and MSE_r are, respectively, derived from Eqs. (11), (14), and (15).

Definition 12 *(Weight of the candidate classifier).* Given u and MSE_r, candidate classifier K_i is treated as the best-performing member in the ensemble group. The weight of the candidate classifier, denoted by w_i, is given as

$$w_i = e^{-u\times\mathrm{MSE}_r}. \tag{17}$$

The utilization of the piecewise function in the weight setting depends on the assumption that the best representative of the current and near-future data distributions is generally the candidate component K_i trained on the observations in the most recent data block C_i. The piecewise function is conducive to coping with a sudden change when only the most recent data chunk represents the concept of the testing data. However, the transition period between two concepts makes knowledge transfer between consecutive chunks important when the change rate is slow. Moreover, the combined result of all the experts in the ensemble is robust against noise compared to that of a new expert.

In general, a classifier that is trained on a dataset with a stable feature space is more likely to behave consistently than a classifier built on a dataset with an unstable feature space. Therefore, in addition to the performances of the members, the similarity of the unsupervised knowledge should be considered in the weighting function. Eq. (16) shows that the weight of K_j consists of the prediction accuracy, the current class distribution, and the deviation level between C_i and C_j in terms of the unsupervised knowledge. Fading factor u controls the discriminative power in the final decision. Compared to past components, the weight of the candidate classifier K_i described in Eq. (17) disregards the performance and the feature distribution similarity. Accordingly, the weighting process in KME, which does not require cross-validation, is suitable for coping with a high-speed data stream. Moreover, the piecewise setting ensures that the latest component has substantial voting power.

The system achieves the final decision by a weighted vote of every ensemble member as

$$B(x_t) = \arg\max_{y_t\in Y}\left(\left(\sum_{j=1}^{i-1} w_{i,j}\times f(K_j(x_t), y_t)\right) + \left(w_i \times f(K_i(x_t), y_t)\right)\right), \tag{18}$$

where $B(x_t)$ is the label of x_t predicted using ensemble B. $f(K_j(x_t), y_t)$ and $f(K_i(x_t), y_t)$ are indicator functions defined as

$$f(K_j(x_t), y_t) = \begin{cases} 1, & \text{if } K_j(x_t) = y_t \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

$$f(K_i(x_t), y_t) = \begin{cases} 1, & \text{if } K_i(x_t) = y_t \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

The final result for an incoming observation is based on all the component classifiers through the component evaluation and weighting mechanisms. Consequently, the generalization ability of KME benefits from the knowledge associated with past members and the candidate component classifier.

3.6. Algorithm detail

In our implementation, the base classifier of B is VFDT. VFDT is an incremental classifier for high-speed and potentially infinite data streams. Meanwhile, B' is a VFDT-based single classifier for detecting concept changes. The complete procedure of the KME algorithm is presented in Algorithm 1. Line 1 is used to initialize the parameters, and lines 2–51 describe the operational phase of the KME algorithm.

First, the candidate ensemble member K_i and the statistics derived from the latest data chunk are constantly adjusted with the new data x_t of data stream S. If the arrival instance is a labelled observation, then the current classifier K_i is retrained (line 7). Sequence $E_{i,s}$, described in Section 3.1, is updated (line 9); then, TEST_l is executed to detect the concept changes affecting $P(y|x)$ when m_l supervised instances have been acquired (line 10). B' is an additional classifier that is not updated with the arrival data unless a concept drift occurs. When a sequence of m_u observations (disregarding their labels) has been obtained, the unsupervised statistics $M_{i,s}$ and $V_{i,s}$, described in Section 3.1, are computed (lines 14 and 15), and TEST_u is performed to monitor the stationarity of the feature space (line 16). Therefore, the concept-drift-detection system described in Section 3.2 is applied to recognize concept drift based on the stationarity of the supervised and unsupervised statistics described in Section 3.1.

Second, the concept-isolation procedure described in Section 3.4 is implemented when the concept-drift detection mechanism discovers a concept change or when a sufficient number of supervised instances are obtained in the current block

(line 19). A new data chunk is produced, and the timestamp of the last observation in the current block is denoted by $T_{i,\,end}$ (line 20). The observations of the new concept are acquired in the interval $[T_{i,\,1},\ T_{i,\,end}]$ (line 21). Then, the candidate hypothesis is built over the labelled sample set Z_i in the new chunk (line 22).

Third, the reconfiguration of drift detectors and the recognition of recurrent concepts are performed after producing a new data chunk (lines 26–39). In general, the reconfiguration phase may lack instances associated with the new concept. Consequently, the procedures are likely to be postponed until sufficient examples are available for the configuration. In particular, TEST_l is likely to experience a long time delay compared with TEST_u because limited supervised data are provided. The concept-drift-detection system can be reconfigured when sufficient unsupervised and supervised statistics are available (i.e., $|M_{i,s}| \geq M_u$, $|V_{i,s}| \geq M_u$, and $|E_{i,s}| \geq M_l$) (line 26). TEST_u is reconfigured based on $M_{i,s}$ and $V_{i,s}$ (line 27). Z_i^T, described in Section 3.3, is used to train an additional classifier B' (line 28). TEST_l is reconfigured on the classification error of B' (line 29). Then, the new concept C_i is compared with all previous concepts C_j $(j = 1, 2, \ldots, i-1)$ based on the equivalence levels of the unsupervised and supervised knowledge (lines 30–35). The recognition of recurrent concepts is described in Section 3.3. The labelled examples of the recurrent concepts are used to update the training set of the candidate hypothesis K_i (lines 33 and 36).

Finally, a performance-based pruning technique is applied to B when the ensemble size reaches m (line 46). Then, a weighted result based on all the members in the ensemble group, which is described in Section 3.5, is used to predict the label of an unlabelled observation in the data-generating process (lines 48–50). The weights of the candidate hypothesis and past hypotheses are calculated according to Eqs. (17) and (16), respectively (lines 37 and 41).

Let us now analyse the computational complexity of the KME algorithm. A data stream can be divided into N data chunks, N_l sliding windows containing supervised observations, and N_u sliding windows containing observations. The base classifier of KME is VFDT, which builds a decision tree with a constant time per example [21]. Therefore, the training of ensemble members has complexity $O(N_l m_l)$, where m_l is the number of labelled observations in a sliding window. The drift-detection system consists of two drift detectors. TEST_u calculates the unsupervised statistics for every m_u observations; thus, $O(2N_u m_u)$ time is needed to estimate the expected values of the sample mean and variance. The interval estimation of the sample mean and variance consumes $O(4N_u b)$ time, where b is the number of preserved windows in the current data chunk. Similarly, TEST_l requires complexity $O(N_l m_l + 2N_l b)$. The recognition of recurrent concepts manipulates a series of data chunks and requires $O(Nm|Z^V|)$ time, where m is the ensemble size and $|Z^V|$ is the number of labelled instances in the validation set used to evaluate the equivalence level between two concepts based on supervised information. Moreover, the weight settings defined in Eqs. (16) and (17) require a constant number of operations and $O(Nm)$ time. Consequently, the time complexity of KME in the training phase is $O(2N_l m_l + 2N_u m_u + 2N_l b + 4N_u b + Nm|Z^V| + Nm)$. In the testing stage, the time consumption is dominated by the number of test instances. Thus, the prediction procedure of the KME algorithm is linearly proportional to the number of labelled instances in a dataset when using prequential evaluation [5]. The memory consumption of an ensemble of m VFDT is $O(mdvlc)$, where d is the number of attributes, v is the maximum number of values per attribute, l is the number of leaves in the tree, and c is the number of classes [21]. The labelled data in the most recent data chunk are conserved to evaluate the importances of previous components and to recognize the recurrent concepts. Consequently, $O(s_l d)$ memory is required, where s_l is the number of labelled data in a data chunk. The labelled instances in past concepts are preserved to improve the generalization ability of the ensemble, thus consuming $O((m-1)s_l d)$ memory. The memory usage of the latest sliding window in a data stream is $O(m_u d)$. Meanwhile, the drift-detection system requires $O(4b)$ memory. The weights of past components are based on stationarity levels and predictive power, whereas the weight of the candidate classifier is dominated by the number of classes c. Thus, the weighting mechanism of KME has space complexity $O(m + c)$. Consequently, the memory usage of KME is $O(mdvlc + ms_l d + m_u d + 4b + m + c)$. In the testing phase, the testing data are evaluated in an instance-by-instance manner. Thus, the space complexity of the testing procedure is $O(1)$.

4. Experimental evaluation

4.1. Experimental setup

The analysis of the KME algorithm and the comparative experiment are implemented in Java programming language by extending the MOA software [5]. The following algorithms are tested.

1) The naive Bayes algorithm (NB) [26] is continuously updated by each supervised example. NB is excellent in stationary scenarios and is considered to be a reference for using an algorithm without any drift reaction mechanisms.
2) VFDT [21] is a single classifier that can incrementally learn streaming data by building decision trees. The Hoeffding bound is used to ensure that the result in an online manner is asymptotically identical to that of a conventional decision tree.
3) Learn++.NSE (NSE) [14] determines voting weights based on the changing accuracy of each member in current and past environments. NSE can learn the evolving concepts, regardless of the types of concept drift.
4) AWE [37] is a representative chunk-based ensemble, where the chunk size is fixed. The final prediction for an incoming observation is derived from the weighted result of members in the classifier pool.

Table 2
Dataset description.

Dataset	#Inst	#Attrs	#Classes	Noise	#Drifts	Drift type
SEA_G	1M	3	2	10%	9	Gradual
$RanRBF_B$	1M	20	4	0%	2	Blips
SEA_S	1M	3	2	10%	3	Sudden
Hyper	1M	10	2	5%	1	Incremental
SEA_{SR}	1M	3	2	10%	4	Sudden recurrent
$RanRBF_{GR}$	1M	20	4	0%	4	Gradual recurrent
$RanTree_{SRF}$	100k	10	6	0%	15	Sudden recurrent
Covertype	581k	54	7	–	–	–
Poker	829k	10	10	–	–	–
Usenet	1.5k	99	2	–	–	–

5) Online bagging (OBag) [33] is the online version of bagging. In contrast to bagging, OBag does not obtain all the data before the training procedure [8]. Online sampling is applied to the bath model by presenting each instance to a component k times, where k is defined by the Poisson distribution.
6) Online boosting (OBoost) [33], which is usually applied to a large volume of streaming data, is the online version of the popular boosting algorithm [17].
7) WMA [28] is a representative online ensemble. When a component commits a mistake, its weight is multiplied by the predefined value β.
8) ADACC [22] takes advantage of changes in the environment to anticipate future characteristics and can be applied to deal with multiple types of concept drift.

KME is not compared with semi-supervised algorithms because of their continuous enhancement of training procedures through labelling a large number of unsupervised examples [27]. However, KME still considers all the unlabelled observations as unsupervised knowledge. New supervised information in the candidate block is obtained from the labelled observations in previous data blocks. In addition to NSE and WMA, the number of components in the ensembles is set to $m = 10$ to ensure the fairness of the comparative experiment. NSE does not remove any members from the ensemble group, and it contains the sigmoid slope $a = 0.5$ and the sigmoid crossing point $b = 10$. These settings are the same as suggested by the paper's authors. Similarly, the ensemble size of WMA is variable, and the default parameter settings of the MOA framework are adopted [5]. The block size of AWE is 1000. In KME, after the number of observations reaches $d_{\max} = 1000$, a new data chunk is produced. The influence of the chunk size on KME is analysed in Section 4.3. VFDT is selected as the base classifier for all ensemble models to ensure meaningful comparisons.

m_u, m_l M_u, and M_l are the parameters used to reconfigure the detection system. Under stationary conditions, the statistics $M_{i,s}$, $V_{i,s}$ and $E_{i,s}$ approximate Gaussian distributions when a sufficient number of events are acquired in the sliding window. A minimum of $m_u = 50$ observations are required to evaluate the sample mean and variance. Then, $TEST_u$ is reconfigured by $M_u = 4$ unsupervised statistics (corresponding to 200 examples). Additionally, $m_l = 50$ labelled observations in each sliding window are used to evaluate the classification performance. Therefore, $M_l = 4$ features (corresponding to 200 labelled examples) are applied to configure $TEST_l$. The threshold γ is set to 0.05, and τ is set to 0.8 in the recurrent concept recognition process. The parameters γ and τ are discretely set to avoid reusing the labelled data of different concepts. u is the fading factor in the weighting mechanism, which is analysed in Section 4.3. Through statistical analysis, $u = 1.0$ is found to be significantly better than other settings; therefore, $u = 1.0$ is the default value of KME.

The evaluation measure is periodically calculated using the prequential method with a window of $W = 1000$ examples and fading factor of 0.01 in the MOA framework [5]. For every tested classifier, a labelled instance is first used to test the existing model; then the classifier is updated. Thus, the prequential accuracies are incrementally updated. Through this method, the data in a dataset are sequentially processed, and learning curves can be provided to capture the evolution of the classifier performance throughout the life of the data stream. All the labelled data items are used to test and train the models. Therefore, the training set and testing set are identical in each trial, and the prequential classification accuracies of all the algorithms should be constant over multiple runs.

4.2. Dataset description

Suitable and publicly available real-world benchmark datasets for evaluating data stream classifiers are insufficient because they do not contain any type of concept drift. Several synthetic datasets have been generated in the MOA framework to analyse the performance of the proposed algorithm. Moreover, the real datasets are publicly available. These synthetic datasets include various types of drifts, different change rates, and various noise levels. However, the details of the real datasets cannot be obtained in advance. Table 2 presents a brief description of each dataset.[1]

[1] Scripts available at: https://github.com/siqirenzl/KME.

4.2.1. Synthetic datasets

We utilize the data-streaming generators available in the MOA framework to construct seven synthetic datasets, including sudden, gradual, incremental, and recurrent concept drifts with different change rates. A supervised label is provided out of $h = 5$ observations. The details of the synthetic datasets are presented as follows.

1) Hyper: The hyperplane generator was used as a testbed to compare CVFDT and VFDT [21]. The orientation and position of the hyperplane can be smoothly changed by modifying the weights. Thus, datasets with incremental concept drifts can be simulated. The Hyper dataset, which is generated by the Hyperplane generator, contains 1, 000, 000 instances described by 10 attributes. An incremental drift can be simulated by changing the weight by 0.1 with every example, and adding 5% noise to the data.
2) RanRBF: The random Radial Basis Function (RBF) generator can produce a random radial basis function stream. A fixed number of random centroids are generated, and each centre has a random position, standard deviation, class label, and weight. This generator is used to create two datasets of 1, 000, 000 observations described by 20 attributes. The RanRBF_{GR} dataset contains four gradual recurrent drifts. The RanRBF_{B} dataset is composed of two blips, which should be ignored by the models because the changes are random. The RanRBF_{B} can measure the robustness of classifiers, and each concept in the datasets produced by this generator is characterized by four classes.
3) SEA: The SEA generator was first described in [36]. It is used to simulate concept drift by changing the threshold. Datasets are generated by three attributes, of which only the first two attributes are relevant. The SEA generator is used to produce three datasets of 1, 000, 000 instances with 10% noise. SEA_{S} produces four concepts by creating a sudden drift every 250, 000 observations. The SEA_{SR} dataset contains four sudden drifts. Five concepts are included in the SEA_{SR} dataset, of which the first concept is a recurrent concept of the fifth concept. Therefore, we can analyse the ability of the algorithms to handle sudden recurrent drifts on SEA_{SR}. The SEA_{G} dataset contains nine gradual drifts.
4) RanTree: The Random Tree generator, which was proposed in [13], generates a data stream based on a randomly generated tree. This generator is leveraged to create the RanTree_{SRF} dataset, which consists of 100, 000 observations described by 10 attributes and 6 classes. RanTree_{SRF} contains 15 sudden drifts that occur every 2790 observations. Recurrent concepts are also considered. Compared with SEA_{SR}, the frequency of changes on the RanTree_{SRF} dataset is large.

4.2.2. Real datasets

Forest Covertype (Covertype), which is obtained from US Forest Service Region 2 Resource Information data, is one of the most widely used dataset in the streaming context [6]. Covertype contains 581, 012 instances with 54 attributes used to describe one of seven possible forest cover types. Poker-lsn (Poker) [6] is the normalized version of the Poker-Hand dataset, which consists of 829, 201 observations that describe the ranks and suits of a hand of five playing cards. Poker contains 10 predictive attributes and 10 classes. Each instance is represented by five playing cards drawn from a standard deck of 52. Finally, Usenet is a real-world dataset that simulates an email stream derived from various topics. Usenet has been applied to the domains of medicine, baseball, and space. A total of 1500 instances and 99 attributes are included in the Usenet dataset to predict whether a given email is junk or interesting.

We provide a supervised instance out of $h = 5$ observations for all the real datasets. In contrast to the synthetic datasets, we do not know when the concept drifts occur, the types of changes or even whether any drift occurs.

4.3. Analysis of the components of the proposed algorithm

The piecewise exponential function is used to evaluate the importance of members in the ensemble group. The weight of a previous component classifier is determined by its predictive power and stationarity level. The candidate component is treated as the best-performing member, regardless of its performance. Therefore, cross-validation can be avoided to cope with high-speed data streams. The supervised and unsupervised knowledge are considered in the weighting function, and $u \in [0, 1]$ denotes the fading factor. A larger u can enhance the importance of excellent components and reduce the importance of poorly performing components in the final decision.

Table 3 shows the classification accuracies for different values of u. Through the Friedman test [12] on $u \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, $F_F = 27.51$ is obtained. Table 4 shows the average ranks of KME using different values of u. If the selected significant level is 0.05, then the null hypothesis is rejected. Hence, significant differences exist among the predictive accuracies of KME for different settings of u. Then, the Bonferroni–Dunn post hoc test [12] is performed to compare KME with $u = 1.0$ with others. The critical difference (CD) for $\alpha = 0.05$ is equal to 1.77. Therefore, the classification accuracy of KME with $u = 1.0$ is significantly better than that with $u = 0.2$ and $u = 0.4$. The Wilcoxon signed rank test [12] is applied to further assess the differences between $u = 1.0$ and the remaining settings. The p-values resulting from this test are $p_{u=0.6} = 0.002$ and $p_{u=0.8} = 0.02$. Therefore, the average accuracy of KME with $u = 1.0$ is significantly better than that with other settings of u.

In addition to studying u, we assess the influence of different chunk sizes on the predictive accuracy of KME. Generally, the performance of chunk-based ensembles strongly depends on the chunk size. Using big size chunks is likely to contain sudden changes in the training set of a component classifier, whereas using small size chunks may degrade the performance of the ensemble under stationary conditions. In this study, different $d_{\max}$ values are used to analyse the influence of chunk size on KME performance. Table 5 shows the average accuracies of KME in terms of the aforementioned datasets based on $d_{\max} \in \{500, 750, 1000, 1250, 1500\}$. The average ranks of KME using different chunk sizes are listed in Table 6. The Friedman

Table 3
Average classification accuracies of KME using different values of u (%).

	$u = 0.2$	$u = 0.4$	$u = 0.6$	$u = 0.8$	$u = 1.0$
SEA_G	87.14	87. 15	87.15	87.16	**87.17**
$RanRBF_B$	94.19	94.21	94.23	94.25	**94.27**
SEA_S	87.45	87.48	87.49	87.50	**87.51**
Hyper	81.50	81.51	81.51	81.52	**81.53**
SEA_{SR}	87.51	87.53	87.55	**87.56**	**87.56**
$RanRBF_{GR}$	93.24	93.27	93.30	93.34	**93.37**
$RanTree_{SRF}$	35.41	35.40	35.42	**35.45**	35.44
Covertype	80.75	80.76	80.78	80.80	**80.82**
Poker	72.08	72.16	72.26	72.35	**72.45**
Usenet	**66.84**	**66.84**	**66.84**	**66.84**	**66.84**

Table 4
Average algorithm ranks of KME using different values of u in the Friedman test.

$u = 0.2$	$u = 0.4$	$u = 0.6$	$u = 0.8$	$u = 1.0$
4.70	3.90	3.10	1.95	1.35

Table 5
Average classification accuracies of KME using different values of $d_{\max}$ (%).

	$d_{\max}$=500	$d_{\max}$=750	$d_{\max}$=1000	$d_{\max}$=1250	$d_{\max}$=1500
SEA_G	**87.24**	87.10	87.17	87.15	87.12
$RanRBF_B$	**94.31**	94.16	94.27	94.08	94.11
SEA_S	87.45	87.34	**87.51**	87.48	87.43
Hyper	81.49	81.51	81.53	81.54	**81.55**
SEA_{SR}	**87.59**	87.47	87.56	**87.59**	87.51
$RanRBF_{GR}$	93.40	93.08	93.37	93.46	**93.54**
$RanTree_{SRF}$	**36.33**	35.52	35.44	35.40	35.79
Covertype	80.74	**80.85**	80.82	80.83	80.80
Poker	70.53	72.37	72.45	72.31	**73.91**
Usenet	**66.84**	**66.84**	**66.84**	**66.84**	**66.84**

Table 6
Average algorithm ranks of KME using different values of $d_{\max}$ in the Friedman test.

$d_{\max}$=500	$d_{\max}$=750	$d_{\max}$=1000	$d_{\max}$=1250	$d_{\max}$=1500
2.85	3.70	2.70	2.95	2.80

test is leveraged for $d_{\max} \in \{500, 750, 1000, 1250, 1500\}$, and $F_F = 0.62$ is obtained. Therefore, no significant differences in the prequential accuracies of KME are observed when using different chunk sizes at significant level $\alpha = 0.05$. KME is a hybrid ensemble that combines the elements of chunk-based ensembles and online ensembles. Consequently, the performance of KME is robust against chunk size, and $d_{\max} = 1000$ is selected as the default value in the comparative experiment.

4.4. Comparative study of classifiers

Several experiments are performed to compare the proposed algorithm with other algorithms, including NB, VFDT, NSE, AWE, ADACC, OBag, OBoost, and WMA. NB and VFDT are representative single models. NB is a simple classifier without any forgetting mechanisms, which is appropriate for handling stationary data streams. VFDT is a famous decision tree model that specializes in high-speed data streams. Additionally, we propose VFDT as the base classifier for all the tested ensembles.

NSE, AWE, ADACC, OBag, OBoost, and WMA are all ensemble models. OBag and OBoost are strong representatives of online ensembles that are derived from traditional bagging and boosting algorithms, respectively. NSE can accommodate a wide variety of drifting environments based on its time-adjusted member weights. AWE combines multiple classifiers weighted by their expected prediction accuracies on the observations in the most recent block, which is generally appropriate for data streams with slow drifts. WMA maintains a pool of hypotheses, and the final prediction relies on the weighted average of the results derived from the members. The weight of each member is dynamically updated based on its corresponding performance. ADACC stores a series of snapshots to handle recurrent concepts. ADACC adapts to the dynamic concepts through the frequent removal and addition of ensemble members, regardless of the type of concept drift.

Table 7
Average prequential classification accuracies (%).

	KME	ADACC	NB	VFDT	AWE	OBag	WMA	NSE	OBoost
SEA_G	**87.17**	85.05	84.66	85.49	83.07	85.64	85.55	85.83	86.04
$RanRBF_B$	94.27	76.96	60.69	86.17	81.52	91.79	86.49	81.85	**95.57**
SEA_S	**87.51**	85.64	83.88	84.63	82.40	85.13	85.09	87.25	86.42
Hyper	**81.53**	79.71	72.31	78.58	80.56	79.53	78.63	80.06	81.13
SEA_{SR}	**87.56**	86.08	86.52	86.74	84.44	87.18	87.40	87.32	87.46
$RanRBF_{GR}$	93.37	71.37	59.29	85.87	76.45	92.02	85.93	76.46	**95.05**
$RanTree_{SRF}$	35.44	**36.70**	34.04	34.31	25.92	35.37	34.43	26.40	12.90
Covertype	**80.82**	77.05	60.29	71.46	72.46	74.18	72.48	75.26	76.71
Poker	**72.45**	71.15	59.46	68.21	51.50	70.80	69.18	53.97	69.29
Usenet	**66.84**	64.00	64.25	65.91	**66.84**	66.66	63.50	**66.84**	59.59

Table 8
Average algorithm ranks in the Friedman test.

KME	ADACC	NB	VFDT	AWE	OBag	WMA	NSE	OBoost
1.4	5.2	7.8	6.1	7.0	4.1	5.2	4.6	3.6

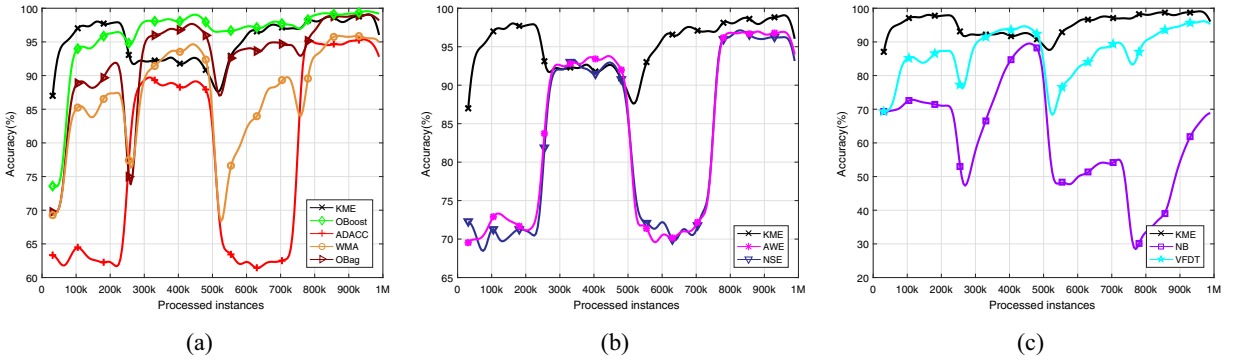

Fig. 1. Classification accuracies on the $RanRBF_B$ dataset. (a) KME and online ensembles, (b) KME and chunk-based ensembles, (c) KME and single classifiers.

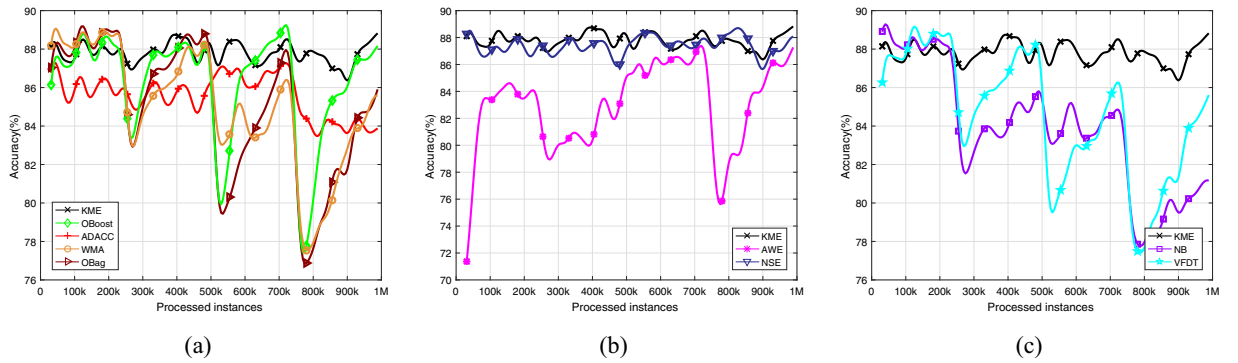

Fig. 2. Classification accuracies on the SEA_S dataset. (a) KME and online ensembles, (b) KME and chunk-based ensembles, (c) KME and single classifiers.

Tables 7 and 8 show the prequential accuracies based on all the datasets and the average ranks of the comparative algorithms, respectively. In addition, a graphical plot is generated for each dataset to describe the performance curves of all the tested algorithms at each time step. The *x*-axis denotes the number of processed observations, and the average accuracy is presented on the *y*-axis. In this way, the adaptation situations of all the comparative algorithms under different streaming conditions can be analysed. Figs. 1–5 show the prequential accuracies of the tested algorithms on datasets with different types of concept drift, including gradual, incremental, sudden, and recurrent drifts. We split the result of a dataset into three sets to analyse the performance curves of single classifiers, chunk-based ensembles, and online ensembles.

Fig. 1 shows the average accuracies of the algorithms on the $RanRBF_B$ dataset, which contains two blips. Blips are very short, sudden drifts that should be considered to be outliers. Fig. 1 shows that blips do not have a lasting effect on the classification accuracies of the classifiers. The decrease in performance associated with an excellent model should only last

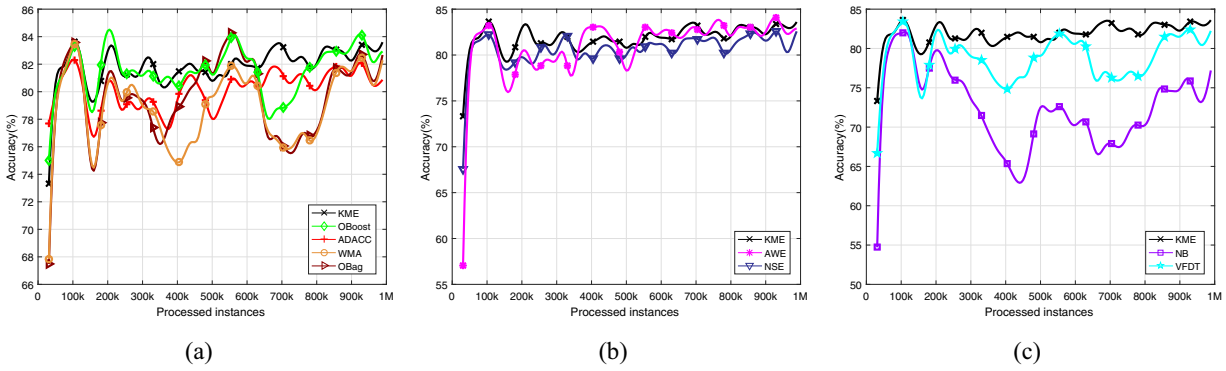

Fig. 3. Classification accuracies on the Hyper dataset. (a) KME and online ensembles, (b) KME and chunk-based ensembles, (c) KME and single classifiers.

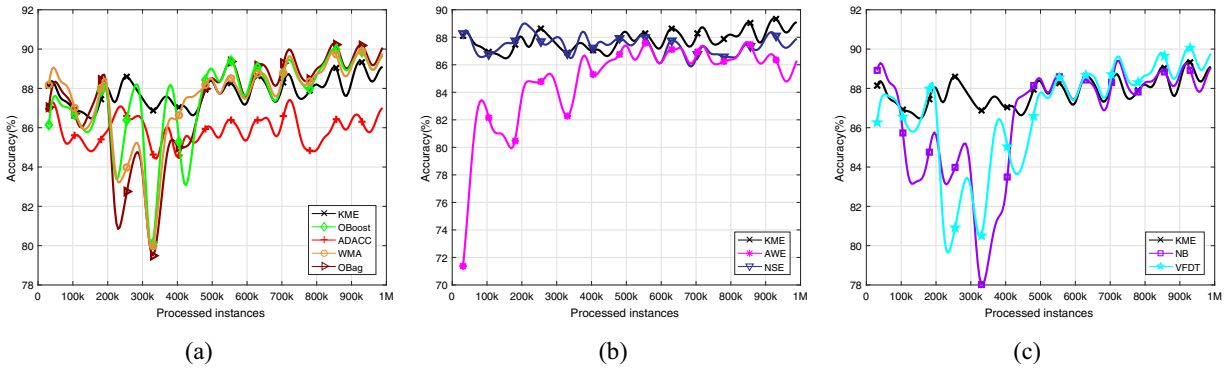

Fig. 4. Classification accuracies on the SEA_{SR} dataset. (a) KME and online ensembles, (b) KME and chunk-based ensembles, (c) KME and single classifiers.

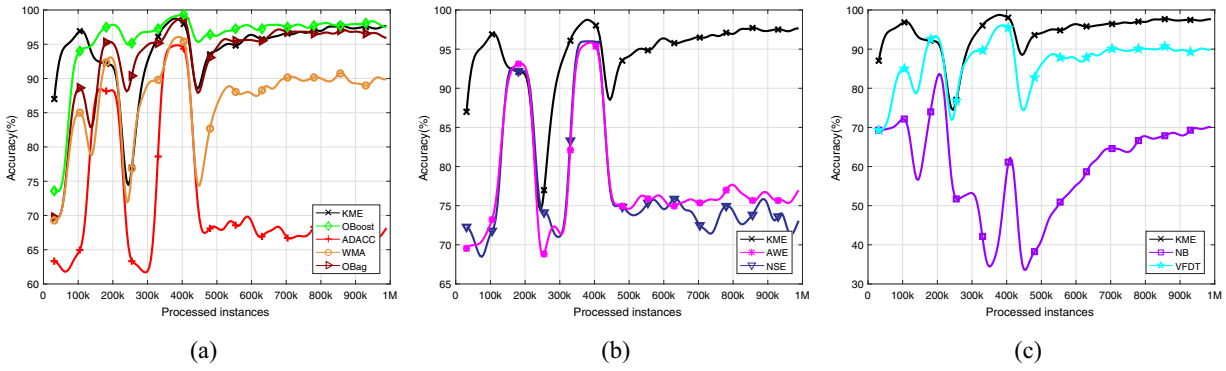

Fig. 5. Classification accuracies on the $RandRBF_{GR}$ dataset. (a) KME and online ensembles, (b) KME and chunk-based ensembles, (c) KME and single classifiers.

for a short time. KME and OBoost show relatively stable levels of prequential accuracies during all the time steps. The utilization of the drift-detection system enables KME to rapidly identify short sudden changes. The performance degradation of KME is not obvious because of the rapid removal of obsolete knowledge and addition of a new block. As an online ensemble, OBoost processes only one observation at a time and emphasizes the misclassification instances. Therefore, OBoost is robust against blips. AWE, ADACC, and NB perform the worst. AWE maintains a pool of classifiers derived from fixed-size data chunks and cannot timely remove obsolete ensemble members when a short sudden change occurs. The prequential accuracy of ADACC fails to quickly recover from changes, thus causing poor performance on the $RanRBF_B$ dataset. The worst-performing algorithm is NB, which tends to be more sensitive to blips as it has no mechanisms for reacting to changes.

Fig. 2 shows the performances of the algorithms on the SEA_S dataset. For data streams with sudden drifts, the best-performing algorithm is KME, followed by NSE and OBoost. Owing to the drift-detection strategy, KME can quickly produce a new chunk to represent the new concept. When evaluating the importance of ensemble members, the candidate component that describes the new concept is provided the highest weight value. We observe that the decreases in the accuracies

of KME and NSE are not significant. Moreover, OBoost provides a fast recovery rate to sudden changes by emphasizing the misclassification samples at each time step. The worst-performing algorithm is AWE. As a block-based ensemble, AWE processes a data stream in a chunk-by-chunk manner and builds a component for each data chunk. Sudden changes are instantaneous and require a model to quickly remove obsolete knowledge. Therefore, AWE has poor performance because of the existence of obsolete members in the ensemble group. NB and VFDT continuously update models with the new labelled observations. Due to the lack of the timely removal of obsolete data and the leveraging of useful past knowledge, substantial old data and limited new data exist in single models when a sudden change occurs.

Fig. 3 presents the prequential accuracies of the analysed algorithms on the Hyper dataset, which includes an incremental concept drift. KME, OBoost, and AWE are the best-performing algorithms. Compared to the results on datasets with relatively drastic changes, the prequential accuracies of all the algorithms show relatively mild drops. AWE performs particularly well on the Hyper dataset, but it has poor reactions to sudden changes. The chunk-based ensembles seem to obtain and maintain good performances in slow drifting conditions. KME is a hybrid ensemble that integrates the predictive results of the ensemble members. The transition phase between adjacent concepts is a mixed data distribution in the dataset with incremental concept drifts. The knowledge transfer obtained by periodic weighting mechanisms can improve the performance of KME in the transition period. The NB and VFDT classifiers have no drift reaction mechanisms, resulting in the poorest performances on Hyper.

Fig. 4 shows the classification accuracies of the tested algorithms on the SEA_{SR} dataset, which contains sudden recurrent drifts. The accuracy decreases around observations numbers 100, 750, 201, 500, 302, 250, and 403, 000. The ordered sequence consists of five concepts, in which the fifth concept is a recurrent concept of the first concept. The best algorithm in the presence of sudden recurrent drifts is KME, followed by OBoost. KME can quickly discover and anticipate sudden drifts by monitoring the stationarity of the feature spaces and the performance of $B^{'}$. The labelled events of recurrent concepts can be reused to enhance the predictive ability of the candidate component classifier after evaluating the equivalence level between two concepts. The prequential accuracy of OBoost quickly recovers from abrupt drifts, which is also observed on the SEA_S dataset. NSE can successfully handle sudden changes, thus obtaining good performance on the SEA_S dataset. Additionally, we can observe that NSE effectively employs the prior knowledge of recurrent concepts by reactivating previous hypotheses. However, NSE has poor convergence in long-term stationary conditions compared with that of KME. WMA does not remove any ensemble members, and the use of recurrent concepts can benefit from sufficient enough members in the final voting. The NB and VFDT classifiers have no drift reaction mechanisms. They both aim to remove all the old data, and thus they have no opportunity to reuse the observations of recurrent concepts. AWE, which uses batch learners derived from fixed-size chunks, cannot immediately react to sudden changes and performs the worst.

Fig. 5 shows the results of the tested algorithms on the $RanRBF_{GR}$ dataset. The best-performing algorithms are OBoost and KME. OBoost maintains a stable accuracy level due to its online nature. KME provides quick recovery from gradual drifts and ranks second. Generally, AWE is excellent in reacting to concept drift with a relatively long transition period. Consequently, AWE performs well on the Hyper dataset. However, because it does not explicitly handle recurrent concepts, AWE fails to react to gradual recurrent drifts. ADACC, AWE, and NSE show poor accuracies in the presence of stationary periods, which is also observed on the SEA_{SR} dataset. For a data stream with gradual drifts, such as SEA_G, KME outperforms the other algorithms. When the environment changes slowly, all the classifiers maintain relatively stable levels of performance.

In addition to the performance under sudden drift conditions, the effectiveness of the tested algorithms to handle $RanTree_{SRF}$, in which the change frequency is large, is analysed. Sudden changes and several recurrent concepts occur in each small time interval of the $RanTree_{SRF}$ dataset. All the algorithms have poor performances on the $RanTree_{SRF}$ dataset because the models cannot keep pace with the rapidly changing environment. In the presence of continuous changes, KME provides good recovery ability from rapid changes. ADACC performs particularly well on the $RanTree_{SRF}$ dataset. In contrast to those of other ensemble models, the base learners of ADACC are evaluated every few time steps. Thus, ADACC can quickly capture the evolution of data streams via the frequent removal of obsolete ensemble members. NSE, AWE, and OBoost are the worst-performing classifiers. NSE has no pruning techniques, which results in excessive outdated data being maintained in the scenario of vigorous changes. Compared with the performance on the SEA_S dataset, the predictive accuracy of OBoost fails to increase with incoming data. A large number of sudden changes result in a large number of misclassification instances in OBoost. AWE provides poor reactions to sudden changes, which can also be observed on the SEA_S and SEA_{SR} datasets. For single classifiers, VFDT and NB cannot obtain recurrent concepts and relevant knowledge in the past, so they perform poorly on datasets with sudden recurrent drifts. On the real datasets, i.e., Covertype, Poker, and Usenet, KME achieves excellent performance. Moreover, on the Covertype and Poker datasets, KME is the most accurate classifier, followed by ADACC. On the Usenet dataset, the performances of KME, NSE, and AWE are same and all of them rank in the first place.

It should be noted that the drift-detection system in KME cannot handle sudden drifts occurring in datasets with nominal attributes. Therefore, $TEST_u$ inspects only the stationarity of the numerical attributes and ignores the changes in the nominal attributes on the $RanTree_{SRF}$ dataset. First, KME reacts in a timely manner to sudden changes according to the aforementioned analysis. Therefore, KME achieves the best performance on the SEA_S and SEA_{SR} datasets. Meanwhile, KME provides the second best performance on the $RanTree_{SRF}$ dataset. Second, the preserved labelled examples of recurrent concepts effectively enhance the predictive power of the candidate component classifier in KME. Therefore, the performance of KME on the $RanTree_{SRF}$, $RanRBF_{GR}$, and SEA_{SR} datasets is excellent. Finally, the evaluation and weighting mechanisms of components ensure that KME maintains good prediction ability on datasets with incremental and gradual drifts, such as Hyper, SEA_G, and $RanRBF_{GR}$. KME provides the best or second best prequential accuracy on all the datasets, regardless of the

type of drift. Therefore, the prequential accuracy of KME achieves a perfect tradeoff between different types of concept drift by combining the elements of online and chunk-based ensembles.

4.5. Statistical analysis of the results

To conclude the experimental results, the nonparametric Friedman test [12] is applied to compare the algorithms over multiple datasets. The null hypothesis assumes that the performances of all the tested algorithms are the same. Table 8 shows the average ranks of the algorithms. $F_F = 8.43$ is obtained from the Friedman test. If the significant level is set to 0.05, then the null hypothesis is rejected. Therefore, significant differences exist in the predictive accuracies of the tested algorithms.

The Bonferroni–Dunn post hoc test [12] is conducted to determine whether the performance of KME is statistically better than that of the other algorithms. The critical difference for $\alpha = 0.05$ is 3.34. Therefore, KME performs significantly better than ADACC, NB, VFDT, WMA, and AWE. However, the significant differences of prequential accuracies between KME and the remaining classifiers cannot be achieved. We additionally use the Wilcoxon signed rank test [12] to compare KME and the remaining algorithms. The p-values for OBag, NSE, and OBoost are $p_{OBag} = 0.001$, $p_{NSE} = 0.002$, and $p_{OBoost} = 0.053$, respectively. These results show that KME is more accurate than all the compared algorithms.

5. Conclusion

A hybrid ensemble classifier called KME, which combines the elements of online and chunk-based ensembles, is proposed to address a wide variety of drift scenarios with limited labelled observations. The management of recurrent concepts, the weighted result of all the components, and the use of unsupervised information can result in the maximum use of relevant knowledge in data streams. The additional supervised information of the current block comes from the labelled events in previous chunks rather than from a labelling process. Therefore, the labelling cost of semi-supervised methods can be avoided, and the supervised information can be supplemented in a timely manner in KME. The processing mechanism for abrupt drifts provides the highest weight for the most recent component classifier and attains a novel drift detection-system based on supervised and unsupervised knowledge. Moreover, the component evaluation and weighting mechanisms proposed for chunk-based ensembles make KME robust against random blips compared with a single classifier. Knowledge transfer can be achieved by reusing labelled instances in the recurrent concepts and integrating the prediction results of ensemble members. Consequently, KME can effectively handle gradual and incremental drifts. In conclusion, KME can accommodate a wide variety of drift scenarios. However, most of the existing streaming algorithms specialize in only one type of concept drift.

As a hybrid ensemble, KME is less dependent on chunk size. The proposed algorithm is optimized by the performance-based pruning technique, which enables the algorithm to immediately adapt to the new environment. In addition to the information derived from labelled observations, unsupervised knowledge is exploited to improve the effectiveness of the drift detection and recurrent-concept recognition. The weighting mechanism simultaneously considers the unsupervised knowledge. An experimental analysis comparing KME with eight classifiers, including single classifiers, chunk-based ensembles, and online ensembles, was conducted to verify the effectiveness of KME. The statistical tests suggest that the proposed algorithm achieves high classification accuracies under various streaming conditions.

In the future, we plan to extend our algorithm to datasets with nominal attributes and to investigate ways to improve the ensemble diversity with unlabelled observations.

Acknowledgements

This work is supported by the Program for New Century Excellent Talents in University (Grant NCET-10-0365), National Nature Science Foundation of China (Grant 60973082, 11171369, 61202462, 61272395, 61370171, 61300128, 61572178) , the National Nature Science Foundation of Hunan province (Grant 12JJ2041, 13JJ3091), and the Planned Science and Technology Project of Hunan Province (Grant 2012FJ2012). The corresponding author of this paper is Bo Liao (dragonbw@163.com).

References

[1] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: Proceedings of the Fourth International Workshop on Knowledge Discovery from Data Streams, 6, 2006, pp. 77–86.
[2] A. Bemporad, M. Morari, Verification of hybrid systems via mathematical programming, in: Proceedings of the International Workshop on Hybrid Systems: Computation and Control, Springer, 1999, pp. 31–45.
[3] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, B. Pfahringer, Efficient online evaluation of big data stream classifiers, in: Proceedings of the Twenty-first ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 59–68.
[4] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing, in: Proceedings of the 2007 SIAM International Conference on Data Mining, SIAM, 2007, pp. 443–448.
[5] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, Moa: massive online analysis, J. Mach. Learn. Res. 11 (2010) 1601–1604.
[6] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavaldà, New ensemble methods for evolving data streams, in: Proceedings of the Fifteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 139–148.
[7] G.E. Box, W.G. Hunter, J.S. Hunter, Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building, 1, JSTOR, 1978.
[8] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.

[9] D. Brzezinski, J. Stefanowski, Combining block-based and online methods in learning ensembles from concept drifting data streams, Inf. Sci. (NY) 265 (2014) 50–67.
[10] D. Brzezinski, J. Stefanowski, Reacting to different types of concept drift: the accuracy updated ensemble algorithm, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 81–94.
[11] E. Cohen, M. Strauss, Maintaining time-decaying stream aggregates, in: Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, 2003, pp. 223–233.
[12] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[13] P. Domingos, G. Hulten, Mining high-speed data streams, in: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2000, pp. 71–80.
[14] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011) 1517–1531.
[15] A. Fern, R. Givan, Online ensemble learning: an empirical study, Mach. Learn. 53 (1) (2003) 71–109.
[16] G. Ferrari-Trecate, M. Muselli, D. Liberati, M. Morari, A clustering technique for the identification of piecewise affine systems, Automatica 39 (2) (2003) 205–217.
[17] Y. Freund, R.E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, in: Proceedings of the European Conference on Computational Learning Theory, Springer, 1995, pp. 23–37.
[18] J. Gama, Knowledge Discovery from Data Streams, CRC Press, 2010.
[19] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with Drift Detection, in: Proceedings of the Advances in Artificial Intelligence–SBIA 2004, Springer, 2004, pp. 286–295.
[20] P.M. Gonçalves Jr., R.S.M. De Barros, Rcd: a recurring concept drift framework, Pattern Recognit. Lett. 34 (9) (2013) 1018–1025.
[21] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 97–106.
[22] G. Jaber, A. Cornuéjols, P. Tarroux, A new on-line learning method for coping with recurring concepts: the ADACC system, in: Neural Information Processing, Springer, 2013, pp. 595–604.
[23] I. Katakis, G. Tsoumakas, I. Vlahavas, Tracking recurring contexts using ensemble classifiers: an application to email filtering, Knowl. Inf. Syst. 22 (3) (2010) 371–391.
[24] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: a new ensemble method for tracking concept drift, in: Proceedings of the Third IEEE International Conference on Data Mining, 2003. ICDM 2003, IEEE, 2003, pp. 123–130.
[25] G. Labinaz, M.M. Bayoumi, K. Rudie, A survey of modeling and control of hybrid systems, Ann. Rev. Control 21 (1997) 79–92.
[26] P. Langley, W. Iba, K. Thompson, et al., An analysis of Bayesian classifiers, in: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI), 90, 1992, pp. 223–228.
[27] C. Liang, Y. Zhang, P. Shi, Z. Hu, Learning very fast decision tree from uncertain data streams with positive and unlabeled samples, Inf. Sci. (Ny) 213 (2012) 50–67.
[28] N. Littlestone, M.K. Warmuth, The weighted majority algorithm, Inf. Comput. 108 (2) (1994) 212–261.
[29] L.L. Minku, X. Yao, Ddd: a new ensemble approach for dealing with concept drift, IEEE Trans. Knowl. Data Eng. 24 (4) (2012) 619–633.
[30] G.S. Mudholkar, M.C. Trivedi, A gaussian approximation to the distribution of the sample variance for nonnormal populations, J. Am. Stat. Assoc. 76 (374) (1981) 479–485.
[31] K. Nishida, K. Yamauchi, Detecting concept drift using statistical testing, in: Discovery Science, Springer, 2007, pp. 264–269.
[32] K. Nishida, K. Yamauchi, T. Omori, Ace: adaptive classifiers-ensemble system for concept-drifting environments, in: Multiple Classifier Systems, Springer, 2005, pp. 176–185.
[33] N.C. Oza, S. Russell, Experimental comparisons of online and batch versions of bagging and boosting, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 359–364.
[34] S. Ramamurthy, R. Bhatnagar, Tracking recurrent concept drift in streaming data using ensemble classifiers, in: Proceedings of the Sixth International Conference on Machine Learning and Applications, 2007. ICMLA 2007., IEEE, 2007, pp. 404–409.
[35] C. Stein, et al., A bound for the error in the normal approximation to the distribution of a sum of dependent random variables, in: Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory, The Regents of the University of California, 1972, pp. 583–602.
[36] W.N. Street, Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 377–382.
[37] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2003, pp. 226–235.
[38] Y. Wang, Z. Li, Y. Zhang, L. Zhang, Y. Jiang, Improving the performance of data stream classifiers by mining recurring contexts, in: Advanced Data Mining and Applications, Springer, 2006, pp. 1094–1106.
[39] G. Widmer, Tracking context changes through meta-learning, Mach. Learn. 27 (3) (1997) 259–286.
[40] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Mach. Learn. 23 (1) (1996) 69–101.
[41] H. Witsenhausen, A class of hybrid-state continuous-time dynamic systems, IEEE Trans. Autom. Control 11 (2) (1966) 161–167.
[42] Y. Yang, X. Wu, X. Zhu, Mining in anticipation for concept change: proactive-reactive prediction in data streams, Data Min. Knowl. Discov. 13 (3) (2006) 261–289.
[43] M.-L. Zhang, Z.-H. Zhou, Exploiting unlabeled data to enhance ensemble diversity, Data Min. Knowl. Discov. 26 (1) (2013) 98–129.
[44] I. Žliobaitė, Combining time and space similarity for small size learning under concept drift, in: Foundations of Intelligent Systems, Springer, 2009, pp. 412–421.