福昕PDF编辑器

- 永久
- 轻巧
- 自由

点击升级会员  点击批量购买

永久使用
无限制使用次数

极速轻巧
超低资源占用，告别卡顿慢

自由编辑
享受Word一样的编辑自由

扫一扫，关注公众号
Detecting concept drifts and reducing the impact from the noise in real applications of data streams are challenging but valuable for inductive learning. It is especially a challenge in a light demand on the overheads of time and space. However, though a great number of inductive learning algorithms based on ensemble classification models have been proposed for handling concept drifting data streams, little attention has been focused on the detection of the diversity of concept drifts and the influence from noise in data streams simultaneously. Motivated by this, we present a new light-weighted inductive algorithm for concept drifting detection in virtue of an ensemble model of random decision trees (named CDRDT) to distinguish various types of concept drifts from noisy data streams in this article. We use variably small data chunks to generate random decision trees incrementally. Meanwhile, we introduce the inequality of Hoeffding bounds and the principle of statistical quality control to detect the different types of concept drifts and noise. Extensive studies on synthetic and real streaming data demonstrate that CDRDT could effectively and efficiently detect concept drifts from the noisy streaming data. Therefore, our algorithm provides a feasible reference framework of classification for concept drifting data streams with noise.
INTRODUCTION

Real-world data streams, such as web click streams, online shopping data, and stock transactions, attract many concerns in the research community of streaming data mining. However, due to the new characteristics of data streams as being continuous, high volume, and open ended, it is a challenge for most of traditional inductive models and algorithms (Quinlan 1993; Shafer et al. 1996; Gehrke et al. 1999). In addition, regarding the concept drifts hidden in data streams and impacted from the noise contamination, it is also challenging for the mechanisms or algorithms of drifting detection (Scholz and Klinkenberg 2005; Enembreck et al. 2007; Tsymbal et al. 2008; Ludl et al. 2008) etc.

To handle these two major issues mentioned above, many efforts actually have been focused on and great progress has been made in the classification, association analysis, and clustering of data streams. More specifically, on the first challenge, new algorithms and models proposed recently include an incremental algorithm of semi-random decision trees for classification on data streams (Hu et al. 2007), an online method for frequent patterns in streaming data (Dang et al. 2008), an effective and efficient clustering approach for analyzing evolving data streams over sliding windows (Zhou et al. 2008), algorithm output granularity (AOG) for resource-constrained data streams (Gaber 2009), micro-clustering algorithms for classification of audio data streams (Aggarwal 2009), a clustering method for spatial data streams (Wei and Peng 2009), a new hybrid model by combining linear and nonlinear models for forecasting time-series data (Sallehuddin and Shamsuddin 2009), and a clustering method for time-series (Denton et al. 2009). On the second challenge, a large number of algorithms and models have been designed as well for concept drifting detection in data streams. One of the most popular methods is related to the ensemble learning, especially referring to the classifier ensembling. Main cases are given below:

i) Non-weighted algorithms and models: an early ensemble algorithm of SEA (Street and Kim 2001) for abrupt concept drift of data streams; a simple, efficient and accurate cross-validation decision tree ensembling method (Fan 2004); an ensemble learning algorithm of dynamic construction and organization (DCO) (Zhang and Jin 2006) with sensitivity to concept drifts.

ii) Weighted ensemble classifiers: a general framework of mining concept-drifting data streams using weighted ensemble classifiers (Wang et al. 2003); an additive expert ensemble algorithm of AddExp (Kolter and Maloof 2005) developed from the incremental ensemble method of DWM (dynamic weighted majority) (Castillo et al. 2003); a boosting-like method by training a classifier ensembling from data streams for different kinds of concept drifts (Scholz and Klinkenberg 2007).
However, for these existing algorithms and models involved above, there are mainly two limitations in common. First, few algorithms focus on tackling various types of concept drifts, such as the gradual or abrupt concept drift and the concept change resulted from the distribution changes of attributes or class labels. Second, potential effect from noise in the concept drifting detection is often overlooked. Actually, also several ensembling streaming data algorithms shed light on both issues of the detection on various concept changes and the noise contamination. For example, concept drifting data stream algorithms of MSRT (Li et al. 2008) based on multiple semi-random decision trees and the EDTC-RAN algorithm (ensembling decision trees for concept drifting data streams based on RANdom feature selection) (Li et al. 2010) (where “semi-random” specifies a random selection on the split-attributes while a determination on split-points in a heuristic method, and “RANdom” indicates that there is no heuristic method used in the algorithm). However, though the predictive accuracy has been improved, heavy overheads of space and runtime would be required.

To address the aforementioned issues, a light-weighted ensemble algorithm of CDRDT (a streaming data algorithm for Concept Drifts in Random Decision Trees) is hence proposed here. It provides a new solution for concept drifting detection from noisy data streams. In general, our major contributions in CDRDT can be summarized below:

i) Instead of the generation method in ensemble models with fixed-sized data chunks, the basic classifiers in CDRDT are constructed incrementally in small chunks of streaming data. And the sizes of data chunks vary with the cases of detection on the concept drifts.

ii) To distinguish various types of concept drifts from noise, the inequality of Hoeffding bounds (Hoeffding 1963) and the principle of statistical quality control (Montgomery 2004) are adopted to specify different thresholds. And these thresholds are utilized to estimate the data distributions of class labels by comparing the difference of average error rates over the trees of every two chunks.

iii) For adaptation to the concept drifts, the sizes of data chunks are adjusted dynamically in limit to the infimum and supremum bounds. Because it is well known that too large or too small magnitude of a data chunk makes against the detection on the data distribution, especially in the case with the classification method of majority class. Meanwhile, only partial classifiers of random decision trees are reconstructed. Because it is beneficial to avoid the potential information loss due to the blind discarding of all history data and an unnecessary reconstruction of all trees from scratch.

iv) Experiments conducted on both synthetic and real databases show that our algorithm of CDRDT enables adapting to diverse types of concept drifts in data streams timely and effectively, and it is robust to noise very much. Meanwhile, in comparison with the state-of-the-art concept drifting algorithm of CVFDT (Concept-Adapting Very Fast Decision Tree learner) (Hulten et al. 2001), ensemble algorithms of MSRT and EDTC-RAN, CDRDT outperforms on the consumptions of runtime and space.
The rest of this article is organized as follows. Section 2 reviews related work on ensemble classifiers for concept drifting or noisy data streams; also the algorithms based on random decision trees for data streams are addressed in this section. Our algorithm of CDRDT for concept drifting detection from noisy data streams is described in detail in Section 3. Section 4 provides our experimental studies and Section 5 summarizes our results and future work.

RELATED WORK

Ensemble Learning for Concept Drifts

Ensemble learning is one of the most popular and effective approaches for handling concept drifts. It signifies that a set of concept descriptions built with time is maintained and predictions are combined with using a form of voting, or the most relevant description is selected. The first system capable of handling concept drift based on ensembling is STAGGER (Schlimmer and Granger 1986), which adapts to the concept shift. In the following years, most of models or algorithms using ensemble strategies or ensemble models have been proposed. Main cases are as follows. i) A SEA algorithm built from ensembling decision trees was introduced for concept drifting data streams (Street and Kim 2001). It splits the data into batches, fits one decision tree per batch, and discards old models in a heuristic approach. However, due to the lack of confidence weights and the limit of exchanging one model in each iteration, the recovery time from concept drifts seems unnecessarily long. ii) A weighted fixed-size committee of incremental decision trees (Stanley 2003) was presented using the incoming examples to update the information of nodes in decision trees continuously. iii) A boosting-like method (Scholz and Klinkenberg 2005) was proposed to train a classifier ensembling from data streams. It could adapt to the concept drift and allow to quantify the drift by its base learners. Meanwhile, an inductive system based on a forest of functional trees was designed for concept drifting data streams (Gama 2005). It applies naïve Bayes classifiers at the decision nodes and leaves to detect concept drifts. iv) An ensemble learning algorithm of DCO (Zhang and Jin 2006) was designed for achieving a high accuracy while remaining sensitive to concept drifts by the individual-construction strategy and global-prediction policy. v) An ensembling approach to concept drift (Kolter and Maloof 2007) was introduced to create and remove weighted experts dynamically corresponding to the changes of performance. vi) A new optimal weights adjustment (OWA) method (Zhang et al. 2008) was proposed.
It utilizes the most recent data chunk to determine the optimum weight values for classifiers and applies a kernel mean matching method to minimize the discrepancy of data chunks in the kernel space. Furthermore, to overcome the weakness on local concept drift for many ensemble approaches (Street and Kim 2001; Stanley 2003; Wang et al. 2003; Gama 2005), a solution was provided to improve the treatment of local concept drifts by replacing the integration function of the ensemble (Tsymbal et al. 2008). However, all algorithms referred above were proposed to tackle concept drifting data streams. But actually not all different types of concept drifts are taken into account, and the impact from noise in the concept drifting detection is seldom concerned.

**Learning From Noisy Data Streams**

By means of the ensemble model, a good many algorithms and approaches have been proposed with consideration of the noise contamination in data streams. For instance, a discriminative model based on the EM framework was introduced for noisy data streams (Chu et al. 2004). It builds an ensemble of weighted classifiers to maximize the likelihood of streaming data. The algorithm of VFDTc (Gama et al. 2003) was extended to accommodate the concept drifting data streams with noise (Gama et al. 2006). That same year a new dynamic classifier selection (DCS) mechanism (Zhu et al. 2006) was designed to integrate base classifiers for effective mining from data streams and addressed the noise handling. In addition, an incremental algorithm of MSRT (Li et al. 2008) was proposed to handle the noisy data streams by means of the inequality of Hoeffding bounds in the generation of ensemble semi-random decision trees. However, regarding the algorithms involved above, such as the ones with the method of EM or boosting, it is hard to meet a light demand on the overheads of time and space. On the other hand, the issue of noise contamination in data streams is concerned, whereas the diversity of concept drifts is overlooked.

**Random Decision Trees for Data Streams**

Since the model of random decision forests (Ho 1995) was first proposed, the random selection strategy of split-features has been applied into the model of decision trees popularly. As a result, quiet a few developed or new random decision trees have appeared (Breiman 2001; Stanley 2003). However, for most of existing algorithms based on random decision trees, it is not suitable for handling data streams directly.
Thus, a random decision tree ensembling method (Fan 2004) for data streams was subsequently proposed. It utilizes the cross-validation estimation for higher classification accuracy. In addition, based on the algorithm of Breiman’s random forests (Breiman 2001), an extended online algorithm of streaming random forests (Abdulsalam et al. 2007) was introduced. It uses node windows and tree windows to decide when to construct new trees, transform frontier nodes or carry out a limited form of pruning. Meanwhile, an incremental algorithm of semi-random multiple decision trees for data streams (SRMTDS) (Hu et al. 2007) was designed. It introduces naïve Bayes classifiers at leaves to improve the predictive accuracy of trees. However, these two algorithms are not proposed toward the issue of concept drift. As a result, a classification algorithm of dynamic streaming random forests (Abdulsalam et al. 2008) was further presented. It is able to handle evolving data streams with the underlying drift of class boundaries using an entropy-based drift-detection technique. Meanwhile, a concept drifting data stream algorithm of MSRT based on semi-random multiple decision trees was developed correspondingly. Contrary to the algorithm of SRMTDS, it first generates different alternative subtrees corresponding to the different types of nodes. Second, it takes the inequality of Hoeffding bounds to enact the thresholds for distinguishing true concept drift from noise. However, because additional subtrees should be prepared in the construction of the original trees, demands on the overheads of space and runtime are heavier.

In contrast to the above decision tree ensembling algorithms, our algorithm of CDRDT to be introduced here is effective and efficient for handling concept drifting data streams with noise. It behaves with four prominent following characteristics. First, the ensemble models based on random decision trees are generated incrementally in variable sizes of streaming data chunks, which are adjusted dynamically in the concept drifting detection. Second, by means of the inequality of Hoeffding bounds and the principle of statistical quality control, two thresholds are specified to partition the bounds among different types of concept drifting and the noise. It benefits avoiding over-sensitivity to concept drifts and reducing the effect from the noise. Third, for adaptation to the new data streams, the check periods are adjusted timely and dynamically. Meanwhile, partial classifiers are regenerated from scratch in tandem with local updating of the remainder classifiers. It is conductive to utilize the information hidden in the remains of history data and the current data streams simultaneously. Finally, it performs better on the predictive accuracy with lighter demands on the consumptions of time and space.
CONCEPT DRIFTING DETECTION ALGORITHM BASED ON
RANDOM ENSEMBLING DECISION TREES

Algorithm Description

The algorithm of CDRDT to be proposed in this section aims for the
detection of concept drifts from noisy data streams. As the processing
flow shows in Figure 1, we first incrementally generate N-classifiers of
random decision trees with variable sequential chunks of data streams in
the function of GenerateClassifier (see Figure 2). Second, after seeing all
streaming data in a chunk, we install a concept drifting detection in this
ensemble model and use the method of majority-class or naïve Bayes to
evaluate the mean error rate at leaves as a comparison baseline (e.g.,
fError) in the function of CalcErrorByClassDistr. In a similar way, we could
obtain the new mean error rate at leaves after seeing the next data chunk
(e.g., sError). In terms of the values of ferror and serror, we compare their
difference with the prespecified thresholds to discern whether a concept
drift appears or it is only the noise. Correspondingly, we adopt the dynamic
adjustment mechanism to adapt to the changes of concept drifts in the
function of CheckConceptChange. A similar training process is replicated

\[
\begin{align*}
\text{Input:} & \quad \text{Training set: } DSTR; \text{ Test set: } DSTE; \text{ Attribute set: } A; \text{ Initial height of tree: } h_0; \\
& \quad \text{The minimum number of split-example: } \text{n_min}; \text{ Split estimator function: } H(); \text{ The} \\
& \quad \text{number of trees: } N; \text{ The set of classifiers: } CT; \text{ Memory constraint: } MC; \text{ Check} \\
& \quad \text{period: } CP. \\
\text{Output:} & \quad \text{Error rate of classification} \\
\text{Procedure CDRDT}(DSTR, DSTE, A, h_0, n_{\text{max}}, H(), N, CT, MC, CP) \\
1. & \quad \text{for each chunk of training data streams } S_j \in DSTR | \text{CP}| = |S_j|, j \geq 1 \\
2. & \quad \text{for each classifier of } CT_s (1 \leq k \leq N) \\
3. & \quad \text{GenerateClassifier}(CT_s, S, MC, CP); \\
4. & \quad \text{if all streaming data in } S_j \text{ are observed} \\
5. & \quad \text{er} = \text{CalcErrorByClassDistr}(); \\
6. & \quad \text{if the current chunk is the first one} \\
7. & \quad \text{fError} = \text{er}; \\
8. & \quad \text{else} \\
9. & \quad sError = \text{er}; \\
10. & \quad \text{if } (j \geq 2) \\
11. & \quad \text{CheckConceptChange}(fError, sError, CP, S_i, CT); \\
12. & \quad fError = sError; \\
13. & \quad \text{for each classifier of } CT_s \\
14. & \quad \text{for each test instance from } DSTE \\
15. & \quad \text{Travel the tree of } CT_s \text{ from its root to a leaf}; \\
16. & \quad \text{Classify with the method of majority class or Naïve Bayes in } CT_s; \\
17. & \quad \text{Return the error rate of voting classification};
\end{align*}
\]

FIGURE 1 Processing flow of CDRDT.
FIGURE 2 Function of GenerateClassifier.

until the end of data streams. Finally, we determine the class labels of the test instances by voting in the method of majority class or naive Bayes. In this processing, three main functions involved above are illustrated in detail, respectively, including the incremental generation of random decision trees, the adoption of detection strategies and the adaptation to the concept drifts and noise.

**Ensemble Classifiers Based on Random Decision Trees**

The model of random decision trees adopted in CDRDT is generated incrementally with the incoming sequential chunks of streaming data in GenerateClassifier. More specifically, for each training instance in a chunk, it first traverses a tree from the root to an available node, such as a leaf, a growing node, or a node with a numerical attribute. In this processing, the height that the passed node lies in this tree should be lower than the specified threshold-$h_0$. And then according to the characteristics of passed node, three distinct cases are taken into account as follows.

i) **HandleGrowingNode**. If the passed node is a growing one without splitting, install the random feature selection to solve the split-attribute and mark it as one of the diverse types of nodes corresponding to the selected attribute-feature. To be concrete, if it is a numerical attribute, mark it as a continuous type after updating the statistical information, including the count of passed instances, the distributions of class labels and attribute values. Otherwise, generate a child node with a growing type, set it to the current node and install the split-test continuously until a leaf is met.
ii) *HandleConNodeWithSplitAttr*: If the current node is a *continuous* one without a cut-point, the necessary information is updated. And then a split test is installed in a random strategy if there are sufficient instances (i.e., \( n_{\text{min}} \)) in favor of it. In other words, the split-attributes are selected randomly, whereas the cut-points are determined without any heuristic method, namely, randomly select an index of the discretization intervals for a *numerical* attribute and then set the average attribute values in this interval to a cut-point. This strategy is also applied into the algorithm of EDTC-RAN. Or else, find its child node or create one if no child node exists, and repeat growing subbranches in all trees recursively.

iii) *HandleLeafNode*: As the above description of the algorithm, all types of the generated children nodes are specified as *growing* ones in case of their heights lower than \( h_0 \). Otherwise, it will be marked as *leaves* and the relevant statistical information will be updated.

Meanwhile, to adapt to the open-ended streaming data, we also provide a mechanism for the issue of space overflow (i.e., the function of `RealseSpaceOfTree`). More precisely, if a check period is reached and the number of instances arrived amounts to the prespecified threshold of memory overflow, three steps are taken to release the space. This is similar to the method in EDTC-RAN. That is, first, stop all undergoing splits of decision nodes and change them into leaves. If necessary, remove the storage space of information at leaves with the top-\( k \) error rates of classification. Second, release the space of decision nodes from bottom to top, including the storage space of attribute values and class labels. Third, cut off partial subtrees in terms of the simple pruning principle.

**Detection Strategy of Concept Drifts**

Before the discussion on the issue of concept drifting detection, several basic concepts are first introduced below.

**Definition 1.** A concept signifies either a stationary distribution of class labels in a set of instances at the current data streams or a similar distribution rule about the values of attributes in the given instances.

**Definition 2.** A concept drift specifies that the concepts described above demonstrate the dynamic distributions in class labels or values of several attributes.

According to the divergence of concept drifting patterns, the change modes of a concept are divided into three types of *concept drift*, *concept shift*, and *sampling change* as the description in (Yang et al. 2005).
Definition 3. Both types of concept drift and concept shift belong to the patterns with distinct change frequencies in the attribute values or class labels in databases. The former refers to the gradual change and the latter signifies the rapid change. However, sampling change indicates the pattern with the change in the data distribution of class labels. (All changes are called concept drifts instead here.)

In our algorithm, the concept drifting detection is installed after obtaining all instances of a sequential data chunk. In terms of the relation between the mean error rate of classification at leaves (estimated in the method of majority class or naïve Bayes) and the thresholds (specified by the inequality of Hoeffding bound and the theory of statistical quality control), we analyze the changes of data distributions in the current data streams and take corresponding measures timely. Now, given the details of the Hoeffding bound inequality below: Consider a real-valued random variable $r$ whose range is $R$. Suppose we have made $n$ independent observations of this variable, and computed their mean $\bar{r}$, which shows that, with probability $1 - \delta$, the true mean of the variable is at least $\bar{r} - \varepsilon$.

$$P(r \geq \bar{r} - \varepsilon) = 1 - \delta, \quad \varepsilon = \sqrt{R^2 \ln(1/\delta) / 2n}$$

(1)

In Eq. (1), the random variable of $r$ refers to the error rate of classification. Suppose the target variate (i.e., $\tilde{r}$) specifies the average error rate of classification on the $i$th data chunk in this ensemble classifier, denoted as $\tilde{\eta}_i$. Correspondingly, we could obtain the classification result on the $(i+1)$th data chunk, marked as $\tilde{\epsilon}_i$. And then we compare the difference between $\tilde{\eta}_i$ and $\tilde{\epsilon}_i$ to discover the distribution changes of class labels, i.e., $\Delta \epsilon = \tilde{\epsilon}_i - \tilde{\eta}_i$. Once the value of $\Delta \epsilon$ is nonnegative, a potential concept drift is taken into account. Otherwise, it is regarded as a case without concept drift. This is based on the statistics theory, which guarantees that for stationary distribution of the instances, the online error of naïve Bayes will decrease, whereas the distribution function of the instances changes, the online error of naïve Bayes at the node will increase (Duda et al. 2001; Gama 2005). This theory is also suitable for the classification results in the method of majority class if the sizes of streaming data chunks are small but with sufficient instances. An optimal value obtained from Li et al. (2010) refers that the minimum size of a streaming data chunk is set to $n_{\min} = 0.2k$, $1k = 1000$. And it is also an experimental conclusion verified in our experiments on the tracking of concept drifts in Section 4. As a consequence, Eq. (1) could be transformed into Eq. (2), where the value of $n$ refers to the size of the current chunk of streaming data, $N$ signifies the number of total trees, $l_k$ specifies the count of leaves at the $k$th classifier, $n_{ki}$ indicates the count of instances, and $p_{ki}$ means the error rate of classification at the $i$th leaf in the classifier of $CT_k$. 


To distinguish various concept drifts from noise, different values of $\varepsilon_0$ are specified to partition their boundaries, which rely on the tolerant deviation between the estimated error rate and the target one. Thus, it is evident to conclude that the larger the variance of $\varepsilon_0$ is, the higher the drifting probability is. And the more the likelihood that the previous model won’t adapt to the current data streams is, due to the deficiency in the accuracy of classification. Hence, the value of $\delta_0$ will decrease. Furthermore, according to the principle of statistical quality control (Montgomery 2004), the concept drift means the case of out-of-control in the production process. There is a representative assumption that the distribution of the quality characteristic is normal and 2-$\sigma$ or 3-$\sigma$ control limit is popular to use (Castillo et al. 2003; Gama et al. 2004). Though it is influential in our thinking, an evident difference depends on the adoption of the inequality of Hoeffding bound to specify our thresholds. By means of the inequality of Hoeffding bound, two thresholds of $3\varepsilon_0$ (denoted as $T_{\text{max}}$) and $2\varepsilon_0$ (marked as $T_{\text{min}}$) are specified to control the deviations of error rates, whose values are defined as follows:

$$P(\bar{e}_i - \bar{e}_j \geq \varepsilon_0) = 1 - \delta_0, \quad \varepsilon_0 = \sqrt{R^2 \ln(1/\delta_0)/2n} \quad (2)$$

$$\bar{e}_j(\bar{e}_i) = \left( \sum_{k=1}^{N} \sum_{i=1}^{k} \left( p_{ki} \cdot n_{ki} / \sum_{i=1}^{k} n_{ki} \right) \right) / N \quad (3)$$

Where the value of $\varepsilon_0$ follows the binominal distribution if the object database owns binary class labels with the sufficient instances. It is hence set to $\varepsilon_0 = \sqrt{(1 - \bar{e}_i) \cdot \sqrt{\bar{e}_i} / n}$ in this case. Otherwise, it is initialized to $\varepsilon_0 = \sqrt{R^2 \ln(1/\delta_{\text{max}})/2n}$ for the case of databases with multiple class labels (the values of $\delta_{\text{max}}, \delta_{\text{min}}$ and $R = \log(|\text{label}|)$ are constants, in which $|\text{label}|$ refers to the count of diverse class labels for the current database).

**Adaptation to the Concept Drifts Affected From Noise**

According to the relevant analysis and the definitions on the thresholds, four types of drifts are partitioned, including the non-concept drift, the potential concept drift, the plausible concept drift and the true concept drift. More specifically, if the value of $\Delta e$ is negative, it is considered as a non-concept drift. Otherwise, it is in a case of other three possible concept drifts. Namely, i) if the value of $\Delta e$ is less than $T_{\text{min}}$, a potential concept drift is considered, where potential indicates that the slower or much slower concept drift is probably occurring. ii) If greater than $T_{\text{max}}$, it is considered as a true concept drift, which is resulted from a gradual concept drift or an abrupt concept drift. iii) The last case refers to a plausible concept drift owing to the impact from noise, which is a transition state between a potential concept drift and a true concept drift.
In terms of the aforementioned detection results on concept drifts, different strategies are adopted correspondingly in the function of `CheckConceptChange` (Figure 3). First, for the case of non-concept drift, maintain the size of the current data chunk in a default value. Second, for a potential concept drift, increase the chunk size by the number of instances $m_{\min}$ (e.g., $m_{\min} = n_{\min} = 0.2k$). Third, if it is a plausible concept drift, shrink the size of streaming data chunk and the check period to a third of the original size for further check, i.e., $\alpha = 1/3$. Finally, in the case of a true concept drift, reduce the size into a half of the original value, i.e., $\beta = 1/2$. Considering the disadvantage of too large or too small size of a streaming data chunk, the maximum bound (e.g., $m_{\max} = 10 \times n_{\min}$) and the minimum one (e.g., $m_{\min}$) are specified to control the change magnitude of a data chunk for better adaption to the concept changes. Hence, once a bound is reached, the check period remains unchanged until a new concept drift appears. Moreover, to maintain the history data selectively, only partial trees with error rates more than the value of $s_{\text{Error}}$ are regenerated from scratch as shown in the function of `ReconstructPartialTrees`. Meanwhile, the remaining trees are updated locally in the incoming chunks of streaming data. This benefits reducing the information loss and the information staleness due to blindly discarding or maintaining all history data.

**Function:** `CheckConceptChange(fError, sError, CP, S, CT)`

1. $\Delta e = s_{\text{Error}} - f\text{Error}$;
2. if $0 < \Delta e < T_{\min}$
3. Increase the magnitudes of $|CP|$ and $|S_{i+1}|$ by a fixed value;
4. if $T_{\min} \leq \Delta e \leq T_{\max}$
5. Reduce the magnitudes of $|CP|$ and $|S_{i+1}|$ by a rate of $\alpha$;
6. if $\Delta e > T_{\max}$
7. Shrink the magnitudes of $|CP|$ and $|S_{i+1}|$ by a rate of $\beta$;
8. if the current flag is a true concept drift
9. `ReconstructPartialTrees(CT)`;

**Figure 3** Function of `CheckConceptChange`. 
Analysis

Time Complexity

In one dimension of time consumption in CDRDT, it is composed of three components, including the incremental generation of trees, the detection on concept drifts and the classification on test instances. More details of their time complexities are given in Table 1. From the observation, we can find that the deviations of the time cost in all involved algorithms mainly rely on the processing of training and detection. Because the overheads of testing time are the same. Therefore, $|t_e|$ refers to the average time cost in a split test at nodes with continuous split attributes, whereas the time complexity at a discrete node is only $O(1)$; The number of nodes with continuous split-attributes is $|N_{con}|$ and the count of nodes with discrete split-attributes is $|N_{dis}|$ on average; $|Attr|$ specifies the total dimensions of attributes in the current database; $|m_A^i|$ refers to the number of distinct values in attribute-$A_i$; $|t_n|$ and $|t_m|$ indicate the mean time overhead of an instance classified in naive Bayes or majority class, respectively (in general, $|t_n| > |t_m|$); $|S_i|$ means the total count of instances in the $i$th-chunk; $h$ refers to the height of the tree; $n_{leaf}$ indicates the count of instances at the leaf node and $|C_k^i|$ specifies the number of examples whose values belong to the $k$th ($k \in 1, \ldots, \min(10, |m_A^i|)$) interval with the $i$th class label in all values of attribute-$A_i$.

Space Complexity

In the other dimension of space overhead, our algorithm of CDRDT does not require any accessory space compared to CVFDT, MSRT, and EDTC-RAN with the window mechanism. And the space cost mainly depends on the size of each tree, i.e., $O(|T_{nl}| \cdot |V| \cdot |label| + |T_l| \cdot |Attr| \cdot |V| \cdot |label|)$ (marked as $O(ts)$, where $|T_{nl}|$ and $|T_l|$ refer to the counts of non-leaves and leaves nodes, respectively and $|V|$ indicates the maximum number of attribute values from all attributes.). However, for algorithms of MSRT and EDTC-RAN, both of the space complexity could be expressed as $O(ts) + O(|window|)$, whereas for CVFDT it is direct to $O((|T_l| + |T_{nl}|) \cdot |Attr| \cdot |V| \cdot |label|)$.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Training</th>
<th>Detection</th>
<th>Max</th>
<th>Bayes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDRDT</td>
<td>$O(</td>
<td>N_{con}</td>
<td>\cdot</td>
<td>t_e</td>
</tr>
<tr>
<td>EDTC-RAN</td>
<td>$O(</td>
<td>N_{con}</td>
<td>\cdot</td>
<td>t_e</td>
</tr>
<tr>
<td>MSRT</td>
<td>$O(</td>
<td>N_{con}</td>
<td>\cdot</td>
<td>t_e</td>
</tr>
<tr>
<td>CVFDT</td>
<td>$O((</td>
<td>N_{con}</td>
<td>+</td>
<td>N_{dis}</td>
</tr>
</tbody>
</table>
**Generation Error Affected From Concept Drifts**

In accordance with the theorem of generation error analyzed in (Breiman 2001), as the number of trees increases, for almost surely all sequences \( \{\Theta_i, i \geq 1\} \), the generation error of \( PE \) would converge to Eq. (6).

\[
PE = P_{X,Y}((P_{\Theta}(h(X, \Theta) = Y) - \max_{j \neq Y} P_{\Theta}(h(X, \Theta) = j)) < 0)
\]  

(6)

where \( X \) is the training set, \( Y \) means the class label, \( \Theta \) specifies the random vector of feature generated from the attribute set, \( P_{X,Y} \) indicates that the probability is over the \((X, Y)\) space, and \( h(X, \Theta) \) refers to the classifier. Equation (6) is obtained on the assumption that the sequences of \( \Theta \) are independent and identically distributed random vectors. Actually, in the case of the concept drifting data streams, it is possible that the streaming data distributions are not uniform any more as the time elapses. As a result, it is indeterminate to judge the convergence of generation error. Therefore, in the analysis of our models, we would give an infimum bound below. Because the concept drifting detection is installed every an interval of certain instances, the training data could be broken down into small sequences, denoted as \( \Theta_t \) \((t \in \{1, 2, \ldots, B\})\), where \( B \) refers to the maximum index of sequence. In an overall consideration, the generation error on a chunk of \( \Theta_t \) is expressed as Eq. (7).

\[
P_{\Theta_t}^{T_t} = P_{\Theta_t}(fv(T_t, \Theta_t) = Y) - \max_{j \neq Y} P_{\Theta_t}(fv(T_t, \Theta_t) = j)
\]  

(7)

Let \( P_{\Theta_t}^{T_t} = |e_t|/|\Theta_t| \), where \( T_t \) specifies the current decision tree ensembling and each tree is generated or updated with the data chunk set of \( \{\Theta_k, 1 \leq k \leq t\} \); and \( fv \) refers to the voting function, which acts on the data chunk of \( \Theta_t \) classified in the current ensemble decision trees and \( e_t \) means the number of instances that are classified falsely in the current data chunk. In the worst case, the generation error would be defined as follows.

\[
PE = \sum_{t=1}^{B} |e_t|/\sum_{t=1}^{B} |\Theta_t| \leq \max(P_{T_t, \Theta_t}^{T_t}(P_{\Theta_t}^{T_t} < 0))
\]  

(8)
\textbf{Proof.} Suppose the values of $|e_i|/|\theta_i|$ ($1 \leq t \leq \theta$) present in an ascending order, i.e., $|e_0|/|\theta_0| \leq |e_1|/|\theta_1| \leq \cdots \leq |e_B|/|\theta_B|$, apparently, $\text{Max}(P_{(T_i, \theta_i)}(P_{\theta_i}^{T_i} < 0)) = |e_0|/|\theta_0|$ could be obtained. To prove $PE = \sum_{t=1}^{B} |e_t|/|\theta_t| \leq |e_B|/|\theta_B| = \sum_{t=1}^{B} |e_t|/\sum_{t=1}^{B} |\theta_t|$, it should satisfy the inequality of $\sum_{t=1}^{B}(|e_t| \cdot |\theta_t|) \leq \sum_{t=1}^{B}(|\theta_t| \cdot |e_t|)$. Actually, it only requires verifying the inequality of $|e_t| \cdot |\theta_t| \leq |\theta_t| \cdot |e_t|$. Because the condition of $|e_t|/|\theta_t| \leq |e_B|/|\theta_B|$ holds, namely, $|e_t| \cdot |\theta_B| \leq |\theta_t| \cdot |e_B|$, ($1 \leq t \leq B$) holds. Consequently, the inequality of $\sum_{t=1}^{B}(|e_t| \cdot |\theta_B|) \leq \sum_{t=1}^{B}(|\theta_t| \cdot |e_B|)$ is also satisfied. Equation (8) is hence proved.

$$PE^*= \text{max}(P_{(T_i, \theta_i)}(P_{\theta_i}^{T_i} < 0)) \geq P_{(T_i, \theta_i)}(P_{\theta_i}^{T_i} < 0) \geq 0.5$$

By means of Eq. (8), the maximum value of $PE$ could be written as Eq. (9). Furthermore, due to $\text{max}_{y \neq Y} P_{\theta_i}(fv(T_i, \theta_i) = j) < 1 - P_{\theta_i}(fv(T_i, \theta_i) = Y)$, it could be further transformed into Eq. (10).

$$PE^* \geq P_{(T_i, \theta_i)}(P_{\theta_i}(fv(T_i, \theta_i) = Y) \leq 0.5)$$

In the analysis of the probability of optimal ensemble model in Hu et al. (2007) and Li et al. (2010), we take it (i.e., $P(\mid \text{Attr} \mid, N, h_0)$) as an estimation of the classification accuracy. Consequently, the generation error that specifies the probability of $P(\mid \text{Attr} \mid, N, h_0) \leq 0.5$ could be expressed as Eq. (11).

$$PE^* \geq P_{(T_i, \theta_i)}(P(\mid \text{Attr} \mid, N, h_0) \leq 0.5)$$

It clearly shows that the probability of optimal ensemble model is inversely directional to the generation error, which is closely related to the values of $\mid \text{Attr} \mid$, $N$, and $h_0$. Thus, it is feasible by adjusting the values of $N$ and $h_0$ to meet the accuracy demand in the handling of concept drifting data streams. Because the value of $\mid \text{Attr} \mid$ is constant for any specified databases.

**EXPERIMENTS**

To verify the efficiency and effectiveness of our algorithm in the detection on different types of concept drifts from noisy data streams, a great number of experiments are conducted on the diverse benchmark concept drifting databases and the real streaming data. Therefore, the section is correspondingly divided into two parts. Section 4.1 first discusses the characteristics of all concept-drifting databases used in our experiments. Section 4.2 analyzes the performances of CDRDT on the drifting detection, the overheads of runtime and space and the predictive accuracy compared with the algorithms of CVFDT, MSRT, and EDTG-RAN for concept drifting data streams with noisy data. All experiments are performed on a P4, 3.00 GHz PC with 1 G main memory, running Windows XP Professional and all algorithms used in this section are written in C++.
Data Source

Synthetic Data

HyperPlane. HyperPlane is a benchmark database of data streams with the gradual concept drift, which has been used in a numeros of references (Hulten et al. 2001; Stanley 2003; Wang et al. 2003; Fan 2004; Li et al. 2008). A HyperPlane in a \( d \)-dimensional space \( (d = 50) \) is denoted by equation: \[ \sum_{i=1}^{d} w_i x_i = w_0. \] Each vector of variables \((x_1, x_2, \ldots, x_d)\) in this database is a randomly generated instance and is uniformly distributed in the multidimensional space \([0, 1]^d\). If \( \sum_{i=1}^{d} w_i x_i \geq w_0 \), the class label is 1, or else it is 0. The bound of coefficient \( w_i \) is limited to \( t \in \mathbb{R} \), which refers to the magnitude of the change with a default range of \([-t, t]\), e.g., \( t = 10 \). For a weight of \( w_i \), each initial value is generated at random, and it increases or decreases continuously by the value of \( \Delta w_i \) until it is up or down to the boundary, then changes the direction. The value-changing direction of \( s \in \{-1, 1\} \), can be shifted into the opposite direction by the probability of \( p_w \) (a default value is 10%) after generating \( N_e \)-instances (where the values of \( N_e \) are set to \( \{8k, 4k, 2k, 1k\} \), respectively.). Namely, the probability of \( \Delta w_i \) replaced by \(-\Delta w_i\) is 10%. Furthermore, to simulate the state of concept drifts in the noisy data streams, a series of parameters is defined below.

i) Parameter-\( k \) (\( 0 \leq k \leq d \)) specifies the total number of attribute-change dimensions, whose values are involved in \( \{2, 5, 8, 10, 16, 25, 40\} \).

ii) A dimension weight of \( w_i \) is dynamically adjusted by a value of \( \Delta w = s_i \cdot t / N_e \) every an example. As a consequence, the change rate of each weight refers to \( \{0.005, 0.01, 0.02, 0.04\} \), respectively.

iii) The noise is introduced randomly into the database by \( r \) varying from 5% to 30%.

SEA. The artificial data of SEA was first described in Street and Kim (2001), which is a well-known data set of concept shift with numerical attributes only. It is composed of 60k-sized random points in a three-dimensional feature space with two classes. All three features have values between 0 and 10, whereas only the first two features are relevant. Those points are partitioned into four chunks with different concepts. In each chunk, a data point belongs to class 1 if \( f_1 + f_2 \leq \theta \), where \( f_1 \) and \( f_2 \) represent the first two features and \( \theta \) is a threshold between a pair of concepts. In this database, there are four thresholds of 8, 9, 7, and 9.5 to divide data chunks. Each chunk reserves 2.5k-sized records as a test set containing 10% class noise for the different concepts. And the rest 50k-sized points are treated as the training data, whose each of concept shift appears every 12.5k-sized instances.
**STAGGER.** STAGGER is another standard database of concept-shifting data streams (Schlimmer and Granger 1986) used to test the abilities of inductive algorithms. In this database, each instance consists of three attribute values: \( \text{color} \in \{\text{green}, \text{blue}, \text{red}\} \), \( \text{shape} \in \{\text{triangle}, \text{circle}, \text{rectangle}\} \), and \( \text{size} \in \{\text{small}, \text{medium}, \text{large}\} \). And there are three alternative underlying concepts, \( A: \text{if} \ \text{color} = \text{red} \land \text{size} = \text{small}, \ \text{class} = 1; \text{otherwise,} \ \text{class} = 0; \ B: \text{if} \ \text{color} = \text{green} \lor \text{shape} = \text{circle}, \ \text{class} = 1; \ \text{otherwise,} \ \text{class} = 0; \text{and} \ C: \text{if} \ \text{size} = \text{medium} \lor \text{large}, \ \text{class} = 1; \ \text{Otherwise,} \ \text{class} = 0. \) The data set generated randomly in our experiments contains 0.1k concepts and each concept contains 1k-sized random instances. The initial concept begins with \( A \), and these three concepts can transfer to each other. The drifting details are as follows: \( A \rightarrow B, B \rightarrow A, \text{and} \ C \rightarrow B \) with a shifting probability of 25%; \( A \rightarrow C \) and \( C \rightarrow A \) with a shifting probability of 75%. In addition, to simulate various noisy cases, irrelevant attributes are designed by us with an initial value of 5 dimensions. Meanwhile, the different noise rates in the class label are introduced varying from 5% to 30%.

**KDDCup99.** The KDDCup99 database (KDDCup99 DataSet 1999) is a database for network intrusion detection, which is selected here because it has been simulated as the streaming data with sampling change (Yang et al. 2005). In this data set, the count of attributes is up to 41 dimensions with 34-dimension of numerical attributes and the number of class labels is totally 24. Due to the skewed distribution of class labels, we filter out the data with minor rates of class labels (e.g., the total number is lower than the value of \( n_{\min} \)). Hence, the remaining data set contains 490k-sized instances with 12 class labels only.

**Real Data**

**Yahoo! shopping data.** The web shopping data used in our experiments are obtained via the Yahoo web service interface (Yahoo! Shopping Web Services), which are sampled from Yahoo shopping databases including catalog listing, product search, and merchant search. Its basic features consist of 19 dimensions of attributes and the total number of numerical ones is up to 13 dimensions. The detailed definitions are described below: the product information is comprised of the attribute set of \( (\text{NumeratingofProduct}, \text{AverageRating}, \text{ImageTag}, \text{ThumbnailTag}, \text{DescriptionTag}, \text{numberOfSpecifications}, \text{RatingUrlFlag}, \text{Condition}, \text{BasePrice}, \text{TaxCost}, \text{ShippingCost}) \) and the related information of merchants is composed of attributes \( (\text{NumRatingsofMerchant}, \text{PriceSatisfactionRating}, \text{ShippingOptionRating}, \text{TimelyDeliveryRating}, \text{EaseofPurchaseRating}, \text{CustomerServiceRating}, \text{IsCertifiedMerchant} \) and \( \text{OverallRating} \)). To mine the relation between the credibility of merchants and possible factors, the attribute of \( \text{OverallRating} \) with different scores are specified as the class labels, which are partitioned into five labels. Considering the distribution of class labels, we extract the number of 84k-sized instances randomly from the obtained records as a training set and the rest 28k-instances as a test set.
Parameter Setting

With respect to the parameter settings in CVFDT, MSRT, and EDT-RAN, they still follow the original definitions in Hulten et al. (2001) and Li et al. (2008, 2010) correspondingly. However, for CDRDT, the values of parameters are given below: $N = 10$, $n_{\text{min}} = 0.2k$, $h_0 = |\text{Attr}|/2$, the size of an initial streaming data chunk of $|\mathcal{S}| = |\mathcal{CP}|$, $MC = 500k$ and $\delta_{\text{max}} = 0.1$ (the values of $\varepsilon_0$ and $\delta_{\text{min}}$ are calculated in Eqs. (4)–(5)).

Evaluations

Experimental Evaluations on Synthetic Databases

In this section our experiments evaluated on three different types of concept drifts are observed in two dimensions. On one hand, we trace the process of drifting detection in the training phase of CDRDT, including the changes of average error rates and check periods. On the other hand, we compare our algorithm against other state-of-the-art algorithms regarding the results of detection on concept drifts, the error rate of classification, and the consumptions of runtime and space. The following subsections give the detailed description of experiments, respectively, corresponding to the different drifting characteristics of streaming data. The symbols involved in our experiments are first concluded as shown in Table 2.

Detection on Concept Drift

To study the performance of CDRDT in the concept drifting detection affected from noise, a set of experiments is conducted on various magnitudes of HyperPlane databases. Given the related parameters below: the number of drifting attribute dimensions $k \in \{2, 5, 8, 16, 25\}$, the drifting degree $\Delta w \in \{0.005, 0.01, 0.02, 0.04\}$, the noise rate $r \in \{5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$, and an initial value of streaming data chunk $|\mathcal{S}| = 1k$, i.e., $|\mathcal{CP}| = 1k$. Figure 4 shows the process of detection on HyperPlane in the case of $k = 5$ and $\Delta w = 0.005$ and the noise rate $r$ varying from 5% to 30% (Due to the space limit, we only give a group of observation results on the 100$k$-sized databases here). In Figure 4, the curves of Drift-Track are drawn in solid lines with the scale of the left y-axis and the ones of Drift-Level are plotted in dotted lines with the scale of the right y-axis. Although Period-Change is described in a plus sign, whose definition in scale is as follows: the lowest value starts from 1$k$ and the basic unit (denoted as BU) is defined to 0.2$k$ based on the left y-axis. Although Period-Change is described in a plus sign, whose definition in scale is as follows: the lowest value starts from 1$k$ and the basic unit (denoted as BU) is defined to 0.2$k$ based on the left y-axis. For instance, in Figure 4a, “39.4” signifies “1k” and the distance between “39.4” and “39.9” refers to 0.2$k$ regarding the values in the case of Period-Change. Otherwise, the numbers marked in the left y-axis conform to the original definition of Error rate.
From Figure 4 we can observe that i) tracking curves fluctuate varying with the concept changes, and it is obviously shown at the beginning of detection. This is because there are large discrepancies occurring due to the insufficiency of training data. ii) With the arrival of streaming data and the concept change, a general trend demonstrates that the magnitude of fluctuation is converging, though there are several local maximum or minimum peaks occurring. iii) Once the current drifting level transfers to another one, a jump always appears in the tracking curve from a local minimum value to a local maximum value. iv) Most of check periods are maintained stably even though there are several higher peaks occurring. v) In general, error rates of classification are reducing with the increasing of noise rates. However, there is an abnormal case existing in Figure 4f. The noise rate is highest whereas the error rate is lowest. This results from the fact that the skewed distribution of class labels is intensive; 30% noise is introduced that probably reverses the original data distribution of class labels. As a result, it is more suitable for the classification case with Max. Actually, similar abnormal cases never occur as the magnitudes of databases increase in our experiments.
FIGURE 4 Drift track on HyperPlane varying with different error rates.

Meanwhile, Table 3 summarizes the overall cases of concept drifting detection in CDRDT and EDTC-RAN regarding the estimation metrics of \textit{FAlarm}, \textit{Missing}, and \textit{Delay} in the tracking (Gama et al. 2009). The statistical results show that though the checking frequency in CDRDT is much fewer than that of EDTC-RAN, it still performs as well as EDTC-RAN. To be concrete, there are no false alarms occurring in both of algorithms. And the deviation of values of \textit{Delay} in CDRDT and EDTC-RAN is limited to 0.087\textit{k}-sized instances while the difference of the number of missing drifts is 1.
TABLE 3 Drifting Detection Statistics on Benchmark Databases

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>AvgDetection</th>
<th>Falarms (%)</th>
<th>Missing</th>
<th>Delay($k$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HyperPlane-100k with 12 concept drifts</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDRDT</td>
<td>Max 46</td>
<td>0</td>
<td>0</td>
<td>0.729</td>
</tr>
<tr>
<td>EDTC-RAN</td>
<td>Max 193</td>
<td>0</td>
<td>1</td>
<td>0.635</td>
</tr>
<tr>
<td>SEA-50k with 4 concept drifts</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDRDT</td>
<td>Max 104</td>
<td>0</td>
<td>0</td>
<td>0.481</td>
</tr>
<tr>
<td>Bayes</td>
<td>42</td>
<td>0</td>
<td>0</td>
<td>0.600</td>
</tr>
<tr>
<td>EDTC-RAN</td>
<td>Max 50</td>
<td>0</td>
<td>4</td>
<td>/</td>
</tr>
<tr>
<td>Bayes</td>
<td>50</td>
<td>0</td>
<td>4</td>
<td>/</td>
</tr>
<tr>
<td>STAGGER-100k with 100 concept drifts</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDRDT</td>
<td>Max 373</td>
<td>0</td>
<td>5</td>
<td>0.268</td>
</tr>
<tr>
<td>Bayes</td>
<td>500</td>
<td>0</td>
<td>100</td>
<td>/</td>
</tr>
<tr>
<td>EDTC-RAN</td>
<td>Max 500</td>
<td>0</td>
<td>100</td>
<td>/</td>
</tr>
<tr>
<td>Bayes</td>
<td>500</td>
<td>0</td>
<td>100</td>
<td>/</td>
</tr>
<tr>
<td>KDDCup99-400k with 36 concept drifts</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDRDT</td>
<td>Max 498</td>
<td>0</td>
<td>8</td>
<td>1.000</td>
</tr>
<tr>
<td>Bayes</td>
<td>502</td>
<td>0</td>
<td>9</td>
<td>0.984</td>
</tr>
<tr>
<td>EDTC-RAN</td>
<td>Max 115</td>
<td>5.29</td>
<td>9</td>
<td>4.361</td>
</tr>
<tr>
<td>Bayes</td>
<td>126</td>
<td>3.17</td>
<td>8</td>
<td>4.030</td>
</tr>
</tbody>
</table>

Furthermore, Figures 5 and 4 present the prediction ability of CDRDT on the test data set of HyperPlane, where the value marked in each histogram refers to the sum of the mean error rate added the variance. On one hand, in the analysis of the predictive results in Figure 5, we can obtain the following conclusions. i) In the cases with constant values of $k$ and $\Delta w$, predictive accuracies decrease continuously with the increasing of noise rates. ii) If the value of $\Delta w$ is initialized by one of $\{0.005, 0.01, 0.02\}$, error rates of classification are approximate, even though values of $k$ are different. And the maximum rate of fluctuation is only lower than 2%. However, in the case of $\Delta w = 0.04$, even if the value of $r$ is fixed, the deviation of error rates varying with the values of $k$ is still evident. For instance, fixing the value of $r = 5\%$, the error rate of classification on the databases with $k = 5$ in Figure 5d is increased by 1.9% compared to the case of $k = 2$, whereas the classification error in the case of $k = 16$ is improved by 5.1% in comparison to that with $k = 8$. iii) Fixing values of $k$ and $r$, as the values of $\Delta w$ increase, predictive accuracies won’t decrease continuously. iv) The larger the magnitude of change in a concept drift, the worse the performance on the predictive accuracy in CDRDT. This is explicitly shown in Figure 5d.
FIGURE 5 Classification on a series of HyperPlane databases in CDRDT.

On the other hand, we further take into account the classification ability of CDRDT in the worst case of \( k = 25 \) varying with the values of \( r \) and \( \Delta w \). Evaluation results shown in Figure 4a reveal that the predictive accuracies in all algorithms involved above are decreasing with the increasing of the noise rates. But actually, CDRDT is superior to MSRT and CVFDT if the value of \( \Delta w \) is less than 0.04 in any noisy cases. However, if specifying the value of \( \Delta w \) as 0.04, CDRDT would not always perform better than CVFDT, whereas it would present similar performance to EDTC-RAND and MSRT.

Moreover, Figure 4b reports the performance of our algorithm of CDRDT on the consumptions of runtime and space compared with other algorithms. First, considering the consumption of runtime, we find that the major deviation relies on the training time, whereas the deviation of the test time is little. And considering the total time, CDRDT consumes least, which is only up to 58.8\%, 62.5\%, and 30.7\% as compared with EDTC-RAND, MSRT, and CVFDT, respectively. Second, regarding the overhead of space, also CDRDT performs best. The minimum magnitude of reduction on the space overhead is almost only 1/2 of that in CVFDT, whereas the maximum value is reduced by 7 times in comparison to that of MSRT.

**Detection on Concept Shift**

On the concept shifting detection, two benchmark databases with concept shifts are selected, including the database of SEA with pure numerical attributes only and the database of STAGGER with pure discrete attributes only. In a similar way, we still evaluate our algorithm in two dimensions mentioned above.
Figure 6 Classification results of CDRDT on HyperPlane with $k = 25$ varying with $\Delta w$. 
In one dimension, the shifting detection in CDRDT is shown in Figures 7 and 8 (where the positions of concept shifts in SEA are drawn in dotted vertical lines as shown in Figure 7 and the minimum value of check period is set to 0.2k-sized instances). From Figure 7 we can see that i) fluctuations are frequent even if there are no concept changes at the beginning of the training. This is similar to the case in HyperPlane. ii) The upward and downward trends in curves alternately appear with the concept shift. This is mainly relevant to the distribution changes of class labels. Actually, there are two class labels only in SEA. Their distributions consist of four concepts in total, whose distribution rate in each concept is 1.8:1, 1.3:1, 0.42:1, and 0.85:1, respectively. In other words, if the first concept shifts to the second one, the number of instances with the first class label decreases whereas the count of instances with the second class label increases. Consequently, the error rate of classification in a relative stable state begins to rise, which is obviously shown in Figure 7b. Furthermore, Figure 8 provides the detection results on STAGGER (in this figure, we only give the tracking results on the databases of STAGGER with 5%-noise and 30%-noise, other cases with 10%-/, 15%-/, 20%-/, and 25%-noise are not provided here due to the limited space). From the observation, we can find that the fluctuations of Period-Change roughly follow the change trends of Drift-Track. However, the local maximum points in Drift-Track do not always occur frequently and exactly with the concept shifts (each concept in STAGGER shifts every 1k-sized instances). If anything, there are prominent descending trends or stable ones in curves. This is because smaller deviation between the distribution of class labels exists in the current streaming data chunk and the next one. Moreover, Table 3 also reports the tracking statistics on databases of SEA and STAGGER (where the symbol of “/” indicates that no drifts are detected). And the statistical results on values of Falarms and Missing confirm that CDRDT outperforms EDTC-RAN on the effectiveness of drifting detection.

FIGURE 7 Drift tracking over sequential data chunks of SEA.
In the other dimension, to validate the performance on the prediction ability and the consumptions of runtime and space in CDRDT, also a set of experiments is conducted on databases of SEA and STAGGER and the experimental results are correspondingly shown in Tables 4 and 5 and Figure 9. First, regarding the predictive ability on SEA as shown in Table 4, though not all cases in CDRDT present prominent advantages compared with CVFDT, the average error rate of classification in CDRDT is still lower.

**TABLE 4** Time and Space Costs on the Database of SEA-50k-2.5k-C-3

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SEA-1</th>
<th>SEA-2</th>
<th>SEA-3</th>
<th>SEA-4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Bayes</td>
<td>Mean</td>
<td>Bayes</td>
</tr>
<tr>
<td>CDRDT</td>
<td>23.87</td>
<td>21.55</td>
<td>24.87</td>
<td>18.88</td>
</tr>
<tr>
<td>EDTC-RAN</td>
<td>35.65</td>
<td>18.93</td>
<td>42.89</td>
<td>13.74</td>
</tr>
<tr>
<td>MSRT</td>
<td>35.04</td>
<td>18.59</td>
<td>42.21</td>
<td>14.49</td>
</tr>
<tr>
<td>CVFDT</td>
<td>15.61</td>
<td>20.97</td>
<td>29.98</td>
<td>23.76</td>
</tr>
</tbody>
</table>

**FIGURE 8** Drift tracking over sequential data chunks of STAGGER.
TABLE 5 Classification Results on STAGGER-100k-30k-D-8

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Noise rate (%) 5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max Bayes Max Bayes Max Bayes Max Bayes Max Bayes Max Bayes Max Bayes Max Bayes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDRDT</td>
<td>Mean 11.16 0 10.93 0 11.13 0 11.17 0 11.08 0 10.88 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variance 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDTC-RAN</td>
<td>Mean 11.16 0 10.93 0 11.13 0 11.17 0 11.08 0 10.88 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variance 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSRT</td>
<td>Mean 11.16 0 10.93 0 11.13 0 11.17 0 11.08 0 10.88 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variance 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVFDT</td>
<td>11.16 / 10.93 / 11.13 / 11.17 / 11.08 / 10.88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Although it performs as approximately as EDTC-RAN on the predictive accuracy because the maximum deviation is less than 3%. However, in comparison to MSRT, the performance in CDRDT is worse in the case with Max, whereas the predictive accuracy could be improved by more than 5.7% on average if adopting Bayes. Meanwhile, experimental results shown in Table 5 reveal that there are no variances in the predictive abilities on STAGGER for all of algorithms if classifying in Max. But in the case with Bayes, the error rate in CDRDT is reduced to 0% as well as MSRT and EDTC-RAN. Furthermore, with respect to the performance on the overheads of runtime and space, CDRDT performs more efficiently as shown in Figure 9 (in this figure, “number+number” is marked in each histogram, where the first number indicates the training time while the latter specifies the test time).
Detection on Sampling Change

In this subsection, we first observe the detection results on the sampling-change database of KDDCup99. The tracking curves plotted in Figures 10 and 11 show that fluctuations of error rates are much more fierce if the distributions of class labels change frequently, such as the case of the curve segment spanning across the interval of 1st–121st data chunk. Although the trend of tracking curves will fall down if a stable distribution of class label reaches, such as the case of the curve segment between the 121st chunk and the 321st chunk. Meanwhile, the corresponding detection statistics of CDRDT are also concluded in Table 3. The statistical data confirm that CDRDT performs well in the detection on sampling changes.

Furthermore, we give the prediction results relevant to the error rate of classification and the overheads of runtime and space as shown in Table 6. From the observation, we can find that i) on the classification ability, CDRDT performs as well as EDTC-RAN. However, in comparison with CVFDT and MSRT, the predictive accuracy of classification in CDRDT could improve by a large range of 14.76% and 35.76%, respectively, in Max, whereas the improvement could be up to 19.43% compared with MSRT in Bayes. ii) Regarding the consumptions of runtime and space, CDRDT is the most efficient algorithm of the four.
TABLE 6 Classification Results on KDDCup99-490k-310k-CD-41

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Error rate (%)</th>
<th>(T+C) time(s)</th>
<th>Memory (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Bayes</td>
<td>Max</td>
</tr>
<tr>
<td>CDRDT</td>
<td>8.87 ± 0.16</td>
<td>9.06 ± 0.48</td>
<td>32 + 19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Max</td>
</tr>
<tr>
<td>EDTC-RAN</td>
<td>8.86 ± 0.21</td>
<td>9.28 ± 1.85</td>
<td>77 + 18</td>
</tr>
<tr>
<td>MSRT</td>
<td>44.48 ± 14.98</td>
<td>28.60 ± 14.20</td>
<td>93 + 18</td>
</tr>
<tr>
<td>CVFDT</td>
<td>23.48</td>
<td>/</td>
<td>75+18</td>
</tr>
</tbody>
</table>

TABLE 7 Time and Space Costs on Yahoo!-Shopping-Data-84k-28k-CD-16

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Error rate (%)</th>
<th>(T+C) time(s)</th>
<th>Memory (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Bayes</td>
<td>Max</td>
</tr>
<tr>
<td>CDRDT</td>
<td>15.45 ± 5.29</td>
<td>2.61 ± 2.01</td>
<td>2 + 1</td>
</tr>
<tr>
<td>EDTC-RAN</td>
<td>16.88 ± 5.14</td>
<td>2.74 ± 2.39</td>
<td>8 + 1</td>
</tr>
<tr>
<td>MSRT</td>
<td>33.52 ± 17.89</td>
<td>44.42 ± 39.16</td>
<td>11 + 1</td>
</tr>
<tr>
<td>CVFDT</td>
<td>23.34</td>
<td>/</td>
<td>10 + 3</td>
</tr>
</tbody>
</table>

Experimental Evaluations on Web-Shopping Data

In the real-world data streams, it is hard to judge whether the current data streams carry a potential concept drift or when a concept drift occurs. And it is inevitable to be affected from the noise contamination. Hence, to verify the feasibility and validation of our algorithm in this real environment, we also conduct a set of experiments in CDRDT compared with EDTC-RAN, MSRT, and CVFDT. The statistical results shown in Table 7 demonstrate that our algorithm outperforms them on the ability of classification and the overheads of runtime and space.

More precisely, considering the predictive accuracy, the highest rate is improved by more than 10% in CDRDT as compared with other three algorithms. And on the total consumption of runtime, it is reduced by the times of 3/2 at least while on the overhead of space, the maximum consumption in CDRDT only takes half of that in CVFDT and the minimum one only takes 1/25 of that in MSRT.

CONCLUSIONS

We proposed a concept drifting detection algorithm based on an ensemble model of random decision trees for noisy data streams, called CDRDT. In contrast to the previous efforts on ensemble classifiers of decision trees, small data chunks with unfixed sizes are adopted in CDRDT to generate the classifiers of random decision trees incrementally. To effectively distinguish different types of concept drifts from noise, two thresholds are defined in virtue of the inequality of Hoffeding bounds and the principle of statistical quality control. Meanwhile, an adaptive mechanism is utilized for better adaptation to concept changes. Extensive
experiments conducted on both synthetic and real-world streaming data have demonstrated that not only does our algorithm of CDRDT adapt to various types of concept drifts, but it is also robust to noise. As compared with the state-of-the-art algorithm of CVFDT and two random decision tree ensembling algorithms of MSRT and EDTC-RAN, CDRDT is superior in the overheads of runtime and space significantly without any loss in the predictive accuracy. This confirms that our algorithm is a light-weighted ensembling one for classification, even in the environment with a variety of concept drifts and noise. However, how to model the noisy data to discern the concept drifts from noise exactly, how to address the cases with the skewed distribution of class labels, and how to deal with the lack of labeled instances in streaming data better are still challenging and interesting issues for our future work. In addition, how to adapt to periodic concept drift is another research topic of our interest.