DME: An Adaptive and Just-in-Time Weighted Ensemble Learning Method for Classifying Block-Based Concept Drift Steam

BAOQUAN FENG, YAN GU, HUALONG YU, XI BEI YANG, AND SHANG GAO
School of Computer, Jiangsu University of Science and Technology, Zhenjiang 212100, China
Corresponding author: Hualong Yu (yuhualong@just.edu.cn)

ABSTRACT This study proposes a novel incremental learning algorithm called distribution matching ensemble (DME) in context of adaptive weighted ensemble learning. In particular, DME estimates the distribution of each received data block by Gaussian mixture model (GMM) and reserves the corresponding distribution information, as well it maintains a group of classifiers in a buffer. When we receive a new data block which is required to be predicted, the similarity between its distribution and each reserved distribution will be calculated by Kullback-Leibler (KL) divergence, and then the similarities can be used to guide the weight assignment of each corresponding classifier to further make adaptive ensemble decision. DME gets rid of the underlying hypothesis that the most recent labeled data block always has the most similar distribution with the current unlabeled data block. In addition, to avoid infinite extension of ensemble buffer during incremental learning, we also develop two dynamic classifier update rules. Experiments results on some synthetic and real-world streaming datasets show that the proposed DME algorithm is able to track and adapt to various types of concept drift just in time. Especially, on data stream with frequent reoccurring drifts, the DME significantly outperforms to several state-of-the-art algorithms, indicating its superiority.

INDEX TERMS Data stream, weighted ensemble learning, concept drift, Gaussian mixture model, Kullback-Leibler divergence.

I. INTRODUCTION

Learning from data stream, or as it is also called online learning or incremental learning, has been demonstrated to be very useful for a growing number of applications in which data are available continuously in time (data stream) and/or there are time and space constraints. Examples of such applications are sensor network monitoring [1], malware detection [2], credit card fraud detection [3], spam filtering [4], and traffic management [5].

Two key challenges posed by learning from data stream are as follows. One is the learner has to process each training example or each block of training instances once “on arrival”, without the need of storage or reprocessing [6], [7].

The other is the online environment is often non-stationary, and the data to be predicted by the learning models may dynamically change over time, which phenomenon is called concept drift [8], [9]. That is to say, the learning machine deployed in streaming environment requires learning examples by means of one pass, as well to adapt to the dynamic change of data distributions, i.e., concept drift. The first term could be satisfied by either modifying the conventional static learning algorithms that only use the new arriving data to tune the model parameters [10], [11], or adopting ensemble learning models which construct a new learner on each new arriving block of instances [12], [13], [14], [15], [16], [17]. As for the second term, i.e., concept drift, it is obviously more sophisticated as there are a number of distribution drift types [18], [19]. To adapt concept drift, there are also two different strategies as follows: one is adding a forgetting mechanism
In particular, the Gaussian mixture model (GMM) is used to estimate the data distribution with considering two following reasons: 1) GMM is capable for approximating any distribution, and 2) a specific GMM [29] can be described with only a few variables, thus storing them avoids to violate one pass rule. Additionally, Kullback-Leibler (KL) divergence [30] is used to estimate the similarity between two GMM distributions, and then the similarity can be further used to associate with weight of corresponding classifier. It is obvious that if two data distributions are more similar, then a higher weight should be designated, too. Finally, we also design two dynamic classifier update rules to avoid the infinite extension of component classifiers in ensemble. We compared the proposed DME algorithm to five other state-of-the-art block-based adaptive weighted ensemble approaches on both synthetic and real streaming datasets. The results show that the proposed DME algorithm can make just-in-time responds to concept drift, and provides more appropriate weight allocation scheme than those compared methods. Especially, when reoccurring concept drift occurs frequently, the DME algorithm presents a more significant superiority.

The rest of this paper is organized as follows. Section II presents the basic concepts and related work in context of block-based adaptive weighted ensembles. In Section III, we describe the structure and procedure of proposed DME algorithm in detail. In Section IV, the experimental settings, results, and analysis are sequentially provided. Finally, Section V concludes the contributions and findings of this work, and indicates the future work.

II. BASIC CONCEPTS AND RELATED WORK
A. BASIC CONCEPTS

Without loss of generality, we suppose that a data stream can be divided into $n$ equal-sized blocks $B_1, B_2, \ldots, B_n$ in which each block has $d$ examples. Each example can be represented as a two-tuples $(x, y)$, where $x$ is a vector including multiple attribute values and $y \in \{C_1, C_2, \ldots, C_l\}$ denotes the class label of $x$, in where $l$ denotes the number of classes. When a new block $B_i$ arrives, the weights of component classifiers $CL_j \in \Psi$ are calculated by a classifier quality evaluation function $Q(\bullet)$ which is also called weighting function. After evaluating the component classifiers, a new classifier is built based on the block $B_i$. If the current size of the ensemble is smaller than $k$ that is the given size of ensemble buffer $\Psi$, then the new built classifier will be added to the ensemble, but if the ensemble has been full, then the weakest component classifier according in $\Psi$ would be replaced by the new classifier. The generic learning procedure of block-based adaptive weighted ensemble learning paradigm is described in Algorithm 1.

Concept drift is a phenomenon that the statistical properties of a target domain change over time in an arbitrary way [18], [19]. If all examples in the data stream come from the same concept, i.e., the same joint probability $P(x, y)$, then we can say that the concept is stable, otherwise, it means that concept drift occurs. Generally speaking, concept drift can be roughly divided into four types which are respectively...
Algorithm 1 Generic block-based adaptive weighted ensemble learning paradigm

Input:
- $S$: a data stream
- $d$: the number of instances given in a block
- $k$: given size of ensemble buffer
- $Q(\bullet)$: classifier quality evaluation function

Output:
- $\Psi$: an updated ensemble with $k$ component classifiers

1: for a newly received data block $B_i \in S$ do
2: make decision for each instance in $B_i$ by combine classifiers in $\Psi$;
3: build new component classifier $C'$ using $B_i$ after it acquires real class labels;
4: calculate the weight for each classifier $CL_j$ in ensemble buffer $\Psi$ using $Q(\bullet)$;
5: if $|\Psi| < k$
6: then $\Psi \leftarrow \Psi \cup \{C'\}$;
7: else
8: then replace weakest ensemble member defined by $Q(\bullet)$ in $\Psi$ with $C'$;
9: end for

FIGURE 1. An example to describe different concept drift types.

B. Feng et al.: DME: An Adaptive and Just-in-Time Weighted Ensemble Learning Method

A term called intermediate concept [8], which is used to describe the transformation procedure between the starting concept and the ending concept, need to be introduced. Specifically, the intermediate concept of gradual drift denotes a mixture of starting concept and ending concept, that is, the probability of observing examples from two distributions changes, one gradually increases and the other gradually decreases. While for incremental drift, its intermediate concept denotes a unitary change of data distribution, that is, the starting concept gradually changes until the ending concept. As for reoccurring drift (see Fig.1 (d)), it means that in procedure of learning, an old concept emerges again. Here, the old concept may recur suddenly, gradually or incrementally.

For different concept drift types, the learning algorithm is required to prepare different responds mechanisms and to achieve different goals. For sudden drift, the algorithm should focus on how to detect the drift rapidly, and recover the algorithm to adapt it in time. For gradual and incremental drifts, the algorithm needs to track the drift and try its best to lower its performance reduction. While for reoccurring drift, it requires stressing on the reusability of historical concepts. Most existing stream learning algorithms aim at dealing with one specific drift type, thus generally perform poorly when several other drift types emerge. Actually, a success learning algorithm from data stream should be robust to any concept drift types, that is, it should be able to detect drift rapidly and change itself to adapt the drift in time.

B. RELATED WORK

As indicated in Section I, although ensemble learning is initially developed to run in static environment to improve...
classification accuracy and generalization ability of learning algorithms, it has been generalized into dynamic environment [22].

As the first proposed dynamic ensemble algorithm running in streaming environment, SEA [26] designates the number of component classifiers in ensemble buffer, and uses a heuristic strategy based on both accuracy and diversity to evaluate the quality of classifiers and update ensemble buffer. Specifically, SEA considers that the contribution of each component classifier for decision is equal; thereby it uses majority voting but not weighted voting to make the final predictions. Of course, it cannot track and adapt concept drifts in time. Therefore, we can say that SEA is a dynamic ensemble learning algorithm, but it is not an adaptive weighted ensemble learning algorithm.

AWE [27] is in its true sense the first generic adaptive weighted ensemble learning paradigm for dealing with concept drifts in data streams. In fact, AWE can be seen as an improved version of SEA. AWE first evaluates all classifiers in ensemble buffer on a new received labelled data block, and then trains a new classifier on that to replace the weakness one in ensemble buffer. Here, the quality evaluation relies on error rate on new received labeled data block. Finally, AWE uses the evaluation results to assign weights for classifiers in updated buffer for providing prediction for next incoming unlabeled data block. In comparison with SEA, the AWE improves adaptive capacity to concept drifts to a large extent. However, it is time-consuming as that to guarantee the accuracy of quality evaluation, each new trained classifier adopts ten-fold cross-validation (10-CV).

Based on the framework of AWE, Brzezinski and Stefanowski proposed the Accuracy Updated Ensemble (AUE1) algorithm [31]. In AUE1, they designed a simpler weighting function to assign and update the weights of the component classifiers. In addition, if it is necessary, the component classifier can learn incrementally. AUE1 offers higher accuracy than AWE, but it inherits the way of 10-CV from AWE; thus it is still time-consuming. Furthermore, Brzezinski and Stefanowski proposed AUE2 algorithm [32] which can be seen as an improved version of AUE1. Specifically, AUE2 combines the error rate based weighting mechanism with a specific incremental learning algorithm, that is, very fast decision tree (VFDT or Hoeffding tree) [33]. In AUE2, they designed a simpler weighting function to assign and update the weights of the component classifiers. In addition, if it is necessary, the component classifier can learn incrementally. AUE2 offers higher accuracy than AWE, but it inherits the way of 10-CV from AWE; thus it is still time-consuming. Furthermore, Brzezinski and Stefanowski proposed AUE2 algorithm [32] which can be seen as an improved version of AUE1. Specifically, AUE2 combines the error rate based weighting mechanism with a specific incremental learning algorithm, that is, very fast decision tree (VFDT or Hoeffding tree) [33].

We note that the emerging adaptive weighted ensemble learning algorithms share a common underlying hypothesis, that is, the most recent received data block always offers the most approximated representation for current and future data distributions. It seems to be sound for data stream with only gradual and/or incremental drifts, however, when sudden or reoccurring drifts emerges, the hypothesis is clearly invalid. In addition, nearly all emerging methods adaptively assign weights only depending on error rates on the most recent labelled data block, while ignore the effect of next unlabeled block. Obviously, the so-called weight adaption is delayed one block at least, causing the learning algorithms cannot track and adapt concept drifts in time, and provide a just-in-time decision. This motivates us to design more robust and just-in-time drifts adaption ensemble learning method in this study.

III. METHODS

A. GAUSSIAN MIXTURE MODEL

Gaussian mixture model (GMM) [29] is popular approach to estimate the probability density function (pdf) of a distribution. To approximate any distributions, GMM always assumes that a unknown pdf could be represented as a weighted sum of $m$ known Gaussian pdfs, i.e.,

$$f(x) = \sum_{i=1}^{m} \omega_i f_i(x) = \sum_{i=1}^{m} \omega_i N \left(x; \mu_i, \Sigma_i \right)$$  \hspace{1cm} (1)$$

where $\omega$ denotes the weight of each Gaussian pdf which can be also seen as the prior probability of each cluster in viewpoint of clustering, meanwhile $\sum_{i=1}^{m} \omega_i = 1$. $N \left(x; \mu_i, \Sigma_i \right)$ denotes a Gaussian pdf in $x$ with mean vector
\( \mu_i \) and covariance matrix \( \Sigma_i \). All parameters in GMM can be iteratively estimated using EM algorithm. At the E stage, \( N \) examples in the data set can be assigned to each cluster, and the probability of an observation \( x \) belonging to the cluster \( i \) can be calculated by Eq. (2), and then the example will be assigned to the cluster with the highest probability.

\[
\gamma_i = \frac{\omega_i \mathcal{N}(x; \mu_i, \Sigma_i)}{\sum_{j=1}^{m} \omega_j \mathcal{N}(x; \mu_j, \Sigma_j)}
\] (2)

At the M stage, the parameters of pdf of each cluster are calculated using the following equations, so that a mixture pdf can be modeled.

\[
\mu_i = \frac{\sum_{j=1}^{N} \gamma_j x_j}{\sum_{j=1}^{N} \gamma_j}
\] (3)

\[
\Sigma_i = \frac{\sum_{j=1}^{N} \gamma_j (x_j - \mu_i) (x_j - \mu_i)^T}{\sum_{j=1}^{N} \gamma_j}
\] (4)

\[
\omega_i = \frac{\sum_{j=1}^{N} \gamma_j}{N}
\] (5)

As long as designating a suitable number of clusters \( m \), GMM can accurately approximate any distribution in theory. Therefore, in this study, we use GMM approach to estimate the distribution of each data block in data stream.

### B. KULLBACK-LEIBLER DIVERGENCE

Kullback-Leibler (KL) divergence [30], which is also known as the relative entropy, is often used to estimate the similarity and/or dissimilarity between two pdfs. For two pdfs \( f \) and \( g \) defined on \( \mathbb{R}^z \), where \( z \) is the dimension of the observed vectors, their KL divergence can be defined as:

\[
D_{KL}(f \parallel g) = \int f(x) \log \frac{f(x)}{g(x)} \, dx
\] (6)

When the \( f \) and \( g \) are both Gaussian pdfs, then the KL divergence has a closed-form expression as follows:

\[
D_{KL}(f \parallel g) = \frac{1}{2} \log \left( \frac{\sum_{g} \omega_{g} \mathcal{N}(x; \mu_{g}, \Sigma_{g})}{\sum_{f} \omega_{f} \mathcal{N}(x; \mu_{f}, \Sigma_{f})} \right) - \frac{d}{2} + \frac{1}{2} (\mu_{f} - \mu_{g})^T \left( \frac{\sum_{g} \omega_{g} \mathcal{N}(x; \mu_{g}, \Sigma_{g})}{\sum_{f} \omega_{f} \mathcal{N}(x; \mu_{f}, \Sigma_{f})} \right)^{-1} (\mu_{f} - \mu_{g})
\] (7)

However, for GMM pdfs, there is no a closed-form expression. In such case, the KL divergence can be approximated to be other functions which may be calculated efficiently. In this study, we adopt the variational approximation strategy which was proposed by Hershey and Olsen [39] to address this problem.

Let \( L_f (g) = \mathbb{E}_X [\log g(X)] \), where \( X \sim f \). The KL divergence can be replaced by a decomposition as follows:

\[
D_{KL}(f \parallel g) = L_f (f) - L_f (g)
\] (8)

Then the lower bounds for \( L_f (f) \) and \( L_f (g) \) can be obtained by using Jensen’s inequality:

\[
L_f (g) \geq \sum_{a} \omega_a^l \left( \log \sum_{b} \omega_b^g e^{-D_{KL}(f_a \parallel g_b)} - H(f_a) \right)
\] (9)
the KL divergence is asymmetric, that is, classifiers in ensemble. Additionally, it is worthy to note that different distribution. Give a worse prediction for the data block with significantly for the data block with similar distribution, but tends to the more similar these two distributions are, and 2) classifier 1) the smaller the KL divergence between two distributions is, Distributions of KL divergence and accuracy constructed on TABLE 1. Distributions of KL divergence and accuracy constructed on $B_1 \sim B_4$ blocks. 

| Training block | $B_1$ | $B_2$ | $B_3$ | $B_4$ |
|----------------|-------|-------|-------|-------|
| $B_1$          | -     | 1.54  | 1.54  | 2.71  |
| $B_2$          | -     | -     | 0.70  | 0.03  |
| $B_3$          | 1.75  | 18.40 | 56.80 | 47.20 |
| $B_4$          | 2.74  | 7.20  | 0.57  | 2.54  |
| $B_4$          | 0.03  | 96.80 | 1.50  | 7.80  |

$$L_f(f) \geq \sum_a \omega_a f \left( \log \sum_a \omega_a f \left( e^{-D_{KL}(f_a||f')} - H(f_a) \right) \right)$$

$$D_{KL}(f||g) = \sum_a \omega_a f \log \frac{\sum_a \omega_a f e^{-D_{KL}(f_a||f')}}{\sum_b \omega_b g e^{-D_{KL}(g_b||g')}}$$

Note that if the KL divergence between two pdfs is larger, then it means that these two pdfs are more different.

To demonstrate the effectiveness and rationality of using KL divergence to adaptively assign weights for classifiers in ensemble, we provide a synthetic example in Fig.2.

Without loss of generalization, in our example, we suppose the data is two-dimensional, and there are only two different classes. Taking (6, 7) and (4, 4) as the centroids of two classes, we firstly generate 250 instances for each class, respectively, and make each of them satisfy a Gaussian distribution. Based on above rule, the generated data block is called $B_1$. Then we counterclockwise rotate the centroids of two classes to sequentially generate three new data blocks $B_2 \sim B_4$. Specifically, the distribution of $B_4$ is totally same as that of $B_1$. It is clear that these data blocks simulate two different drifting types as follows: sudden drift and reoccurring drift. Taking any one block as training block, we calculated the KL divergence between it and any other one block, and meanwhile tested the accuracy on that block. Without loss of generalization, we use naïve Bayes as classification algorithm. The results are presented in Table 1.

The results in Table 1 show two conclusions as follows: 1) the smaller the KL divergence between two distributions is, the more similar these two distributions are, and 2) classifier trained on a data block is able provide a better prediction for the data block with similar distribution, but tends to give a worse prediction for the data block with significantly different distribution.

The above conclusions indicate that the KL divergence is an effective tool to adaptively designate weights for classifiers in ensemble. Additionally, it is worthy to note that the KL divergence is asymmetric, that is, $D_{KL}(f||g) \neq D_{KL}(g||f)$. However, we consider $D_{KL}(f||g) \approx D_{KL}(g||f)$ by observing results in Table 1, thus no matter which block is used as reference object, the KL divergence could reflect similarity/dissimilarity between two distributions well.

C. DISTRIBUTION MATCHING ENSEMBLE (DME) METHOD

In this study, we present a novel adaptive weighted ensemble learning algorithm called DME for dealing with block-based data stream learning issue. Unlike previous methods which associate weights with error rates, DME adopts a new weight assignment rule that correlates weights with KL divergences. In addition, it modifies the weight pre-designating principle which is used by nearly all previous methods. That is to say, DME adopts a delayed weight assignment strategy which provides adaptive weight designation when and only when a new unlabeled data block is received. Therefore, DME implements just-in-time trace and adaption to concept drifts. In other words, DME gets rid of a possibly wrong underlying hypothesis that the most recent received labeled data block always offers the most approximated representation for current and future data distributions. At least, we know it is wrong when a sudden or reoccurring drift occurs between the last labeled data block and the current unlabeled data block. DME constructs on a new underlying hypothesis that two data blocks with smaller KL divergence have more similar distribution with each other. Obviously, DME takes advantage of information embedded in both labeled and unlabeled data blocks, which can effectively prevent delayed response to concept drift.

As indicated in Table 1, two similar data distributions share a small KL divergence, which contraries to weight assignment. Therefore, we use the reciprocal of KL divergence to associate with weight. Suppose there are $k$ component classifiers maintained in ensemble buffer $\Psi$, then the weight $w_i$ that denotes the $i$th component classifier could be calculated as follows:

$$w_i = \frac{1/D_i}{\sum_{j=1}^{k} 1/D_j}$$

where $D_i$ denotes the KL divergence between the GMM pdf corresponding to the $i$th data block in $\Psi$ and that of the current unlabeled data block. Specifically, based on Eq. (12), it can guarantee $\sum_{i=1}^{k} w_i = 1$.

The learning procedure of DME algorithm is described as in Algorithm 2. From the procedure description of DME, we observe that it is significantly different from the generic adaptive weighted ensemble learning paradigm at three following aspects. The first one is that DME designates weights after receiving new unlabeled data block, but the generic paradigm does not. The second one is that the weight assignment of generic paradigm relies on error rate feedback, whereas that of DME associates with distribution similarity.
The final one is that the generic paradigm only stores $m$ classifiers and their weights in buffer, while DME reserves $m$ classifiers and the corresponding $m$ GMM distribution information in buffer. Considering that the GMM information only contains the mean vectors, the covariance matrixes, the cluster weights, and meanwhile the number of them is restricted as $m \times k$, thus the storage is acceptable as it does not violate one pass rule. As for time complexity, in contrast with the generic paradigm, our proposed DME algorithm adds a GMM modeling procedure on a data block, and $k$ KL divergence calculation procedures between two data blocks, but removes the procedure of running $k$ classifiers on new data block to calculate error rates. In fact, the analysis of time complexity for DME may be very complicated as it associates with multiple parameters, including $m$, $k$, $d$, and the parameters existing in component classifier. According to the feedbacks from our subsequent experiments, DME runs rapidly and can satisfy the requirement of real-time decision making.

**Algorithm 2** Distribution Matching Ensemble (DME)

| Input: |
|-------|
| $S$: a data stream |
| $d$: the number of instances given in a block |
| $k$: given size of ensemble buffer |
| $Q(\bullet)$: classifier quality evaluation function |
| $m$: the number of mixture clusters in GMM |

| Output: |
|-------|
| $\Psi$: an updated ensemble with $k$ component classifiers and distribution information of the corresponding data blocks |

1: for a newly received data block $B_i \in S$ do  
2: \hspace{1em} $p_i \leftarrow$ the GMM pdf of $B_i$ estimated by EM algorithm;  
3: \hspace{1em} for each reserved $p_j \in \Psi$ do  
4: \hspace{2em} $D_{ij} \leftarrow$ KL divergence between $P_i$ and $P_j$ calculated by Eq. (11);  
5: \hspace{1em} end for  
6: for each component classifier $CL_j \in \Psi$ do  
7: \hspace{2em} $w_j \leftarrow$ calculate weight based on Eq. (12);  
8: \hspace{1em} end for  
9: make decision for each instance in $B_i$ by combine weighted classifiers in $\Psi$;  
10: \hspace{1em} $C'$ $\leftarrow$ new component classifier built on $B_i$ after it acquires real class labels;  
11: \hspace{1em} if $|\Psi| < k$  
12: \hspace{2em} then $\Psi \leftarrow \Psi \cup \{C'\} \& \Psi \cup \{p_i\}$;  
13: \hspace{1em} else  
14: \hspace{2em} then replace weakest ensemble member defined by $Q(\bullet)$ and its distribution information in $\Psi$ with $C'$ and $p_i$;  
15: end for

Another significant issue lies in that in DME, how to evaluate the quality of component classifiers and to update the buffer. As we know, the quality of a component classifier is defined and evaluated by the quality evaluation function $Q(\bullet)$. In this study, we design two different quality evaluation functions and classifier update rules. The first one is called lowest weight removing rule, which is abbreviated as Ave. In Ave rule, both weight and time factors are considered by $Q(\bullet)$ function. Specifically, for a classifier $CL_i$, we respectively use $\bar{w}_i$ and $\bar{t}_i$ to represent its average weight and how many data blocks it has experienced since it was added into the buffer. We first consider time factor, i.e., when $\bar{t}_i = 2k$, the corresponding classifier will be preferentially removed. If the above condition cannot be met by all component classifiers, then we will abandon the classifier with the lowest average weight $\min(\bar{w}_i)$, and meanwhile guarantee its $\bar{t}_i > 3$. It effectively avoids accident deletion for some significant component classifiers. In this study, we adopt Ave as the default rule. As for the difference between these two rules, we will further compare and discuss in Section IV.

**IV. EXPERIMENTS**

**A. DATASETS DESCRIPTION**

To clearly present the characteristic of proposed DME algorithm, in this study, we used thirteen synthetic and two real-world streaming datasets to conduct comparison experiments. Next, we provide a detailed description about these streaming datasets.

Without loss of generalization, we denoted first nine synthetic streaming data sets to be two-dimensional, and all instances can be averagely divided into one of two classes. Here, the instances in each class satisfies Gaussian distribution, and the initial centroids of two classes are demarcated at $(2, 2)$ and $(5, 5)$, respectively. We simulated different concept drifts by changing the centroids or moving instances into adverse class, which has been adopted in [35] and [38].

For sudden drift, we generated two datasets in which each one contains 50,000 instances, and 100 data blocks, that is, 500 instances per block. Sudden$_F$ and Sudden$_S$ both change the distributions sharply between two adjacent data blocks. The difference between them lies in that Sudden$_S$ changes only once, whereas Sudden$_F$ changes per 20 data blocks. Gradual$_F$ dataset simulates the gradual concept drift in data streaming. In Gradual$_F$, we gradually move instances in each class into the adverse class. On any two adjacent data blocks, 1% instances are exchanged.

For reoccurring drift, we simulate it by adopting a similar way of generating sudden drifts. When drift occurs, two adjacent blocks synchronously change the distributions of two classes sharply, and then after a period of time, the distributions are recovered again. To better observe the adaption of various algorithms on this drift type, we generated four different reoccurring drift datasets. Reoccur$_S$, Reoccur$_{10}$,
Reoccur\textsubscript{15} and Reoccur\textsubscript{5} occurs drift once per 5, 10, 15, and 20 data blocks.

Incremental\textsubscript{5} and Incremental\textsubscript{F} both simulate the incremental concept drift. The former changes the centroid of each class with a variance of 0.01 per block, while the latter changes the centroids with a variance of 0.1 per block.

The SEA [26], which is a well known sudden drift data stream generator, is also used to generate datasets used in this study. In the concept of SEA, each instance consists of three attributes, where only the first two are relevant, whereas the third one can be seen as noise. All attributes share the same attributes, where only the first two are relevant, whereas the third one can be seen as noise. All attributes share the same attributes, where only the first two are relevant, whereas the third one can be seen as noise.

The SEA [26], which is a well known sudden drift data stream generator, is also used to generate datasets used in this study. In the concept of SEA, each instance consists of three attributes, where only the first two are relevant, whereas the third one can be seen as noise. All attributes share the same attributes, where only the first two are relevant, whereas the third one can be seen as noise. All attributes share the same attributes, where only the first two are relevant, whereas the third one can be seen as noise.
blocks is used to evaluate and compare the quality of various compared algorithms.

### C. EXPERIMENTAL RESULTS AND ANALYSIS

Average classification accuracy (%) of various compared adaptive weighted ensemble algorithms are presented in Table 3 where the best result on each streaming dataset has been highlighted in bold.

From the results in Table 3, we observe that our proposed DME algorithm performs significantly better than several other algorithms on several datasets with reoccurring drifts, indicating it is specifically suitable for dealing with this drifting type. On Reoccur5, Reoccur10, and Reoccur15, DME obviously outperforms several other compared algorithms. In addition, DME shows a clear superiority on Wea dataset. In fact, as a data stream recording the weather variations throughout 50 years, the Wea must contain reoccurring drifts. The phenomenon can be explained well by reviewing the mechanism of DME that weights are assigned according to the similarity between two data blocks. That means if a similar concept is still reserved in ensemble buffer, then it can play an important role for predicting instances in new received testing block. Therefore, we can say that DME can adapt data stream with frequent reoccurring drifts well. As for several other drifting types, although DME has not presented a significant superiority in comparison with several state-of-the-art algorithms, it still produced comparable classification performance. The results show that both error rate and distribution similarity are useful reference objects to reflect the potential and contribution of each component classifier in ensemble. Considering DME is constructed on a totally different underlying hypothesis from several other algorithms, it can observe concept drift in time, and provide a just-in-time adaption for the drift. In other words, DME decreases the potential risk of performance degradation caused by concept drifts.

Next, we simply analyzed these compared ensemble learning algorithms in statistics by Nemenyi test [40]. Specifically, the critical difference (CD) metric is used to show the difference among these algorithms. Fig.3 shows the CD diagram at a standard level of significance $\alpha = 0.10$, in which the average ranking of each algorithm is marked along the axis. In CD diagram, if a group of algorithms are not significantly different, then these algorithms would be connected by a thick line.

In Fig.3, we observe that DME is significantly superior to Learn++.NSE. The reason may be two folds: 1) we have broken the infinite extension of ensemble buffer in Learn++.NSE, and 2) Learn++.NSE assigns weights for component classifiers based on its performance on both past and current data blocks, making it difficult to adapt concept drifts in time. Also, we observe that although DME is not significantly superior to four other algorithms, it has acquired the lowest average ranking; thus we can say that DME is more robust algorithm in comparison with several others.

Furthermore, to more clearly observe the characteristics of various algorithms, we tracked their classification accuracy trajectories throughout the whole learning procedure on four representative streaming data sets, namely SuddenF, Reoccur15, HypF, and Wea, respectively (see Fig.4∼Fig.7). Fig.4 shows that on data stream with sudden drifts, the proposed DME algorithm can defenses the destruction when sudden drifts occur to a large extent. The reason lies in that although there may be no concepts in ensemble buffer which are similar to the new emerging concept, the DME can try its best to adaptively find several data blocks that have the most approximated distribution as the new data block, further reducing the performance degradation. In addition, we note

| Dataset | AWE | AUE1 | AUE2 | DWMIL | Learn++.NSE | DME |
|---------|-----|------|------|-------|-------------|-----|
| SuddenF| 87.13 | 86.85 | 85.29 | 84.98 | 84.29 | 86.66 |
| SuddenF| 84.50 | 85.52 | 86.53 | 86.45 | 86.29 | 86.74 |
| Gradual| 86.59 | 86.50 | 79.77 | 75.01 | 77.18 | 86.77 |
| Reoccur5| 83.08 | 82.32 | 80.53 | 81.19 | 82.27 | 85.58 |
| Reoccur10| 83.28 | 83.71 | 82.94 | 82.27 | 85.89 |
| Reoccur15| 84.40 | 84.65 | 84.15 | 83.84 | 87.51 |
| Incremental| 87.48 | 75.50 | 87.50 | 87.48 | 87.17 | 87.54 |
| Incremental| 87.31 | 87.32 | 87.35 | 87.36 | 87.34 |
| SEAF| 85.70 | 85.67 | 85.73 | 85.76 | 84.80 | 85.74 |
| SEAF| 85.72 | 85.67 | 85.74 | 85.73 | 84.89 | 85.39 |
| HypF| 90.64 | 90.78 | 90.80 | 90.78 | 86.42 | 89.80 |
| HypF| 87.88 | 87.95 | 87.79 | 87.54 | 85.77 | 85.40 |
| Elec| 67.33 | 73.57 | 73.63 | 74.37 | 72.16 | 73.73 |
| Wea| 56.62 | 65.09 | 66.73 | 65.85 | 64.92 | 67.62 |

![CD Diagram of six compared adaptive weighted ensemble learning algorithms.](image-url)
that the DME can rapidly recover the adaptation of ensemble after encountering a sudden drift because the classifier trained on the drifting data block can immediately play an important role as it always has a similar distribution with several subsequent data blocks. At least, the DME presents a faster response speed than several other competitors.

The results in Fig.5 confirm the fact that DME is specifically suitable for dealing with reoccurring drifts again. Except the first drift which can be seen as a sudden drift, the DME has adapted all other reoccurring drifts well. This is because once an old concept recurs, as long as it is still reserved in ensemble buffer, the DME could adapt it well by adaptively activating the corresponding component classifier. Of course, if the recurring cycle is long, and meanwhile the size of ensemble buffer is not large enough, the DME would fail to adapt reoccurring drifts.
Fig. 6 presents the variance of classification accuracy of various incremental ensemble learning algorithms aiming at detecting their reactions to gradual drifts. DME can produce stable performance on this type of data streams, indicating it can adapt gradual drifts well. Of course, most other algorithms can deal with this drifting type well, too.

Considering in general, there are lots of unpredictable and uncertain conceptual changes in real-world environment, using real-world streaming data is expected to better reflect the qualities of various learning algorithms. The results in Fig. 7 indicate the accuracy changes of various algorithms on real-world Wea dataset. It can be observed that in Wea dataset, there are some potential drifts, but the drifting types are unknown. DME performs stably throughout the whole learning procedure, indicating it can adapt real-world drifting streams well.

Also, we are conscious of a potential risk, that is, in an extreme case in which two blocks share the same overall distribution but own totally reverse label distributions, the underlying hypothesis that is used to support DME might be wrong. Fortunately, it is almost impossible to happen in real-world applications. Therefore, in comparison with the previously used underlying hypothesis, our proposed underlying hypothesis in this study is more reliable in theory.

### D. COMPARISON OF TWO WEIGHT UPDATE RULES

Next, we compared the quality and running time of two proposed buffer update rules in this study. The results are presented in Table 4.

From the results in Table 4, we observe that in most cases, the DME with Ave buffer update rule performs better than that with Low buffer update rule. To explore its reason, we consider that the Ave rule simultaneously evaluates the quality of each component classifier from two aspects as follows: its average contribution, and its dwell time. Such behavior significantly lowers the probability of wrongly removing important component classifiers. Also, we note that adopting Ave buffer update rule is generally more time-consuming than using Low buffer update rule, but the increment of running time is acceptable in real-world applications. All in

| Dataset  | Ave | Low |
|----------|-----|-----|
| SuddenR | 87.10 | 86.98 |
| SuddenF | 85.96 | 86.11 |
| GradualF | 86.42 | 86.41 |
| Reoccur1 | 86.74 | 84.63 |
| Reoccur10 | 86.77 | 84.45 |
| Reoccur15 | 85.58 | 85.30 |
| Reoccur30 | 85.90 | 85.86 |
| IncrementalS | 87.51 | 87.40 |
| IncrementalF | 87.34 | 87.19 |
| SLA3 | 85.74 | 85.12 |
| SLA6 | 85.39 | 85.16 |
| Hyp3 | 89.80 | 89.79 |
| Hyp6 | 85.40 | 85.76 |
| Elec | 73.73 | 72.58 |
| Wea | 67.62 | 75.91 |
E. DISCUSSIONS ABOUT PARAMETERS

Finally, we expect to make clear the influence law of two key parameters in DME. One is the size of ensemble buffer $k$, and the other is the number of mixture clusters in GMM $m$. Specifically, we denoted $k$ to vary from 5 to 20, $m$ to vary from 3 to 10, and both vary with an increment of 1.

Taking Reoccur$_{10}$, Incremental$_{S}$ and Elec as representative datasets, the corresponding variance of average classification accuracy (%) and running time per block (cs) are presented in Fig.8.

The results in Fig.8 show that average classification accuracy of DME strongly associates with the parameter $m$, that is, with increase of $m$, the classification accuracy tends to be improved. As we know, $m$ denotes the complexity for describing a distribution, thus it determines the accuracy of estimating pdfs, calculating KL divergences, and designating weights. In addition, we also observe that the average classification accuracy could be greatly impacted by the parameter $k$ that denotes the size of ensemble buffer. The results reflect that on streaming data with reoccurring drifts, a large $k$ should be designated to better adapt recurring old concepts, while on data streams with other drifting types, a small $k$ might be more suitable for DME to track and adapt concept drifts. As for running time, it is linearly associated with both $k$ and $m$. In real-world applications, the readers are suggested to designate appropriate parameters according to the practical requirements.

V. CONCLUSION

In this paper, we propose a novel adaptive weighted ensemble learning algorithm called DME to deal with block-based concept drift streaming data. Specifically, DME uses GMM to estimate distribution of data block, KL divergence to estimate...
the similarity/dissimilarity between two data block distributions, and distribution similarity to adaptively assign decision weights for component classifiers in ensemble buffer. The experimental results on some artificial and real-world non-stationary data streams indicated that the proposed DME algorithm is able to provide just-in-time track and adaption to various concept drifts. Especially on data streams with frequent reoccurring drifts, the DME can present a larger superiority than several state-of-the-art algorithms.

The contributions of this study can be concluded as follows:

1) A new underlying hypothesis, which is used to describe data stream drifting law, is proposed to replace the old underlying hypothesis.

2) A distribution diversity-based weight assignment rule is proposed to replace the old error rate-based weight designation rule, further solving the delayed concept drift adaption problem.

3) Two component classifiers update rules are proposed to guarantee retaining those component classifiers with most potential and significance in ensemble buffer.

In future work, we plan to further verify the effectiveness and superiority of the proposed DME algorithms in more real-world non-stationary data stream applications. Additionally, how to extend DME algorithm from Block-based incremental learning to one-by-one online learning will be investigated, too.

REFERENCES

[1] L. Chen, G. Li, and G. Huang, “A hypergrid based adaptive learning method for detecting data faults in wireless sensor networks,” *Inf. Sci.*, vol. 553, pp. 49–65, Apr. 2021.

[2] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, “A review of Android malware detection approaches based on machine learning,” *IEEE Access*, vol. 8, pp. 124579–124607, 2020.

[3] A. Dal Pozzolo, O. Caelen, Y.-A. L. Borgne, S. Waterschoot, and G. Bontempi, “Learned lessons in credit card fraud detection from a practitioner perspective,” *Exp. Syst. Appl.*, vol. 41, no. 10, pp. 4915–4928, 2014.

[4] H. Tajalizadeh and R. Boostani, “A novel stream clustering framework for spam detection in Twitter,” *IEEE Trans. Comput. Social Netw.*, vol. 6, no. 3, pp. 525–534, Jun. 2019.

[5] Y. Xiao, X. He, C. Yang, H. Liu, and Y. Liu, “Dynamic graph computing: A method of finding companion vehicles from traffic streaming data,” *Inf. Sci.*, vol. 591, pp. 128–141, Apr. 2022.

[6] X. Gu, “An explainable semi-supervised self-organizing fuzzy inference system for streaming data classification,” *Inf. Sci.*, vol. 583, pp. 364–385, Jan. 2022.

[7] L. L. Minku and X. Yao, “DDD: A new ensemble approach for dealing with concept drift,” *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 4, pp. 619–633, Apr. 2012.

[8] J. Gama, I. Zliobate, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM Comput. Surveys*, vol. 46, no. 4, pp. 1–37, Apr. 2014.

[9] B. Mirza and Z. Lin, “Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification,” *Neural Netw.*, vol. 80, pp. 79–94, Aug. 2016.

[10] W.-Y. Cheng and C.-F. Juang, “A fuzzy model with online incremental SVM and margin-selective gradient descent learning for classification problems,” *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 2, pp. 324–337, Apr. 2014.

[11] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, “A fast and accurate online sequential learning algorithm for feedforward networks,” *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.
[36] R. Elwell and R. Polikar, “Incremental learning of concept drift in non-stationary environments,” IEEE Trans. Neural Netw., vol. 22, no. 10, pp. 1517–1531, Oct. 2011.

[37] J. Z. Kolter and M. A. Maloof, “Dynamic weighted majority: An ensemble method for drifting concepts,” J. Mach. Learn. Res., vol. 8, pp. 2755–2790, Dec. 2007.

[38] Y. Lu, Y.-M. Cheung, and Y. Y. Tang, “Dynamic weighted majority for incremental learning of imbalanced data streams with concept drift,” in Proc. 26th Int. Joint Conf. Artif. Intell., Aug. 2017, pp. 2393–2399.

[39] J. R. Hershey and P. A. Olsen, “Approximating the Kullback–Leibler divergence between Gaussian Mixture Models,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), Apr. 2007, pp. 317–320.

[40] S. García, A. Fernández, J. Luengo, and F. Herrera, “Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power,” Inf. Sci., vol. 180, no. 10, pp. 2044–2064, 2010.

[41] G. Hulten, L. Spencer, and P. Domingos, “Mining time-changing data streams,” in Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2001, pp. 97–106.

[42] I. Wickramasinghe and H. Kalutarage, “Naive Bayes: Applications, variations and vulnerabilities: A review of literature with code snippets for implementation,” Soft Comput., vol. 25, no. 3, pp. 2277–2293, Feb. 2021.

BAOQUAN FENG was born in Liaoning, China, in 1997. He received the B.S. degree in the Internet of Things from the Jiangsu University of Science and Technology, Zhenjiang, China, where he is currently pursuing the M.S. degree in software engineering. His research interests include machine learning and data mining.

YAN GU was born in Yangzhou, China, in 1997. She received the B.S. degree in computer science and technology from the Henan Institute of Engineering, Zhengzhou, in 2020. She is currently pursuing the M.S. degree with the School of Computer, Jiangsu University of Science and Technology, Zhenjiang, China. Her research interests include machine learning and data mining.

HUALONG YU was born in Harbin, in 1982. He received the B.S. degree in computer science from Heilongjiang University, Harbin, in 2005, and the M.S. and Ph.D. degrees in computer science from Harbin Engineering University, Harbin, China, in 2008 and 2010, respectively.

Since 2020, he has been a Professor at the School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang, China. From 2013 to 2017, he was a Postdoctoral Fellow at the School of Automation, Southeast University, Nanjing, China. From 2017 to 2018, he was a Senior Visiting Scholar at the Faculty of Information Technology, Monash University, Melbourne, Australia. He has authored or coauthored more than 90 journals and conference papers, and four monographs. His research interests include machine learning, data mining, and bioinformatics.

Dr. Yu is a member in the organizing committee of several international conferences. He is also a member of ACM, China Computer Federation (CCF), and the Youth Committee of the Chinese Association of Automation (CAA). He is an active reviewer for more than 20 high-quality international journals, including IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, and ACM TKDD. He is an Associate Editor of IEEE ACCESS.

XIBEI YANG received the B.S. degree in computer sciences from Xuzhou Normal University, Xuzhou, China, in 2002, the M.S. degree in computer applications from the Jiangsu University of Science and Technology, Zhenjiang, in 2006, and the Ph.D. degree in pattern recognition and intelligence system from the Nanjing University of Science and Technology, Nanjing, China, in 2010. Since 2019, he has been a Professor with the School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang, China. He has published over 100 research articles on the professional journals and conferences. His research interests include granular computing and rough set theory.

Dr. Yang is a member of the organizing committee of several international conferences. He is a reviewer for over ten high-quality international journals.

SHANG GAO received the B.S. degree in system engineering and the M.S. degree in military equipment from Air Force Engineering University, Xi’an, China, in 1993 and 1996, respectively, and the Ph.D. degree in pattern recognition and intelligent systems from the Nanjing University of Science and Technology, Nanjing, China, in 2006. Since 2009, he has been a Professor at the School of Computer, Jiangsu University of Science and Technology, Zhenjiang, China, where he has been the Dean, since 2017. He has published over 80 research articles on the professional journals and conferences. His research interests include optimization theory, swarm intelligence, and machine learning.

* * *