ATLANTIS PRESS International Journal of Computational Intelligence Systems, Vol. 11 (2018) 438–450

Active Fuzzy Weighting Ensemble for Dealing with Concept Drift

Fan Dong [1 2], Jie Lu [2], Guangquan Zhang [2], Kan Li [1]

[1] *School of Computer Science and Technology, Beijing Institute of Technology, 5 Zhongguancun South Street, Haidian District, Beijing 100081, China*

[2] *Centre for Artificial Intelligence, University of Technology Sydney 15 Broadway, Ultimo, New South Wales 2007, Australia*

Received 13 October 2017

Accepted 22 December 2017

Abstract

The concept drift problem is a pervasive phenomenon in real-world data stream applications. It makes well-trained static learning models lose accuracy and become outdated as time goes by. The existence of different types of concept drift makes it more difficult for learning algorithms to track. This paper proposes a novel adaptive ensemble algorithm, the Active Fuzzy Weighting Ensemble, to handle data streams involving concept drift. During the processing of data instances in the data streams, our algorithm first identifies whether or not a drift occurs. Once a drift is confirmed, it uses data instances accumulated by the drift detection method to create a new base classifier. Then, it applies fuzzy instance weighting and a dynamic voting strategy to organize all the existing base classifiers to construct an ensemble learning model. Experimental evaluations on seven datasets show that our proposed algorithm can shorten the recovery time of accuracy drop when concept drift occurs, adapt to different types of concept drift, and obtain better performance with less computation costs than the other adaptive ensembles.

Keywords: concept drift, change detection, ensemble learning, data streams

1. Introduction

Concept drift refers to unforeseeable changes in the underlying data distribution of data streams over time. Data in non-stationary environments always involves concept drift and is a very pervasive phenomenon in real-world applications, such as changes in user interest in recommender systems, the emergence of new types of spam in email filtering systems, or the evolution of fraud methods in electronic transactions, to name a few[1]. If the concept drift occurs, the patterns which have been induced from past data may not be relevant to the new data, leading to poor decision-making outcomes[2] or inappropriate recommendation[3]. Learning from such a non-stationary environment becomes a critical problem when applying machine-learning techniques on real-world applications. There are four types of concept drift: sudden drift, gradual drift, incremental drift and reoccurring concepts[4]. In most real-world applications, data is organized in the form of data streams, in which the nature or rate of drift is various and convoluted[5], making it more challenging to learn knowledge from data streams involving concept drift.

Through a comprehensive analysis of existing literature, we summary that there are three types of approaches that deal with concept drift: retraining models, adaptive models, and adaptive ensembles. The retraining models focus on detecting when a drift occurs. When a drift is detected, there is a mechanism that indicates the learner to retrain its model using recently instances that reflect the cur-

rent data distribution. The drift detection method is usually conducted by statistical theory that monitors the outputs (error) of learners[6,7,8] or monitors the underlying data distribution[9,10,11,12,13,14]. Adaptive models[15,16,17] have the ability to partially update themselves when the underlying data distribution changes. This approach is arguably more efficient when drift only occurs in local regions. However these two approaches usually have a restrictive assumption that there are no reoccurring concepts in the data stream. Adaptive ensembles[18,19,20,21], using novel voting strategies to combine several base classifiers, can handle different types of concept drift, however their computation costs is high.

Motivated by these issues above, we propose a novel adaptive ensemble algorithm, Active Fuzzy Weighting Ensemble, to dealing with data streams involving concept drift. The main idea is to integrating the drift detection method into an adaptive ensemble learning model. By monitoring distance-error-rate of the ensemble learning model, our algorithm has the ability to indicate when a drift occurs, therefore we can create a new base classifier on demand. We use fuzzy instance weighting and a dynamic voting strategy to organize all the existing base classifiers to construct an ensemble learning model for to make the final prediction. Our proposed algorithm can shorten the recovery time of accuracy drop when concept drift occurs, adapt to different types of concept drift, and obtain better performance with less computation costs than the other adaptive ensembles.

This paper is organized as follows. Section 2 reviews the literature on the approaches dealing with concept drift. Section 3 presents a preliminary study related to our proposed algorithm. Section 4 proposes our Active Fuzzy Weighting Ensemble algorithm. Section 5 evaluates our proposed algorithm with seven datasets. Section 6 summarizes the conclusion with a discussion of future work.

2. Literature Review

The retraining models follows a straightforward strategy for dealing with concept drift. It retrains a new model with recent data instances to replace the obsolete model when concept drift occurs. An explicit concept drift detector is required to decide when to retrain the model. One of the most high concept drift detection algorithms is the Drift Detection Method (DDM)[6]. It monitors the online error-rate of the base classifier to determine whether there are changes in new incoming data. DDM can work independently of the base classifier because it only needs information on whether the base classifier has classified the data instance correctly. DDM assumes that 1) the online error-rate drops when data distribution is stationary; 2) a significant increase in the online error-rate indicates that drift has occurred. Similar implementations have been adopted and applied in the Early Drift Detection Method (EDDM)[7], and Dynamic Extreme Learning Machine[8]. Another type of drift detection method monitors the underlying data distribution. Kifer, Ben-David & Gehrke[10] proposed a modified Kolmogorov-Smirnov test that compares the cumulative distribution functions of two data windows with all possible orderings. They introduced a novel family of distance to measure the two distributions. The largest difference is taken as the test statistics, which quantitatively shows the degree of drift. Dasu et al.[11] presented an information-theoretic-based drift detection method which uses Kullback-Leibler divergence to measure the difference between two sets of data within the given past and recent window. If the dissimilarity is proven to be statistically significantly different, the system will trigger a learning model retraining process. Similar distribution-based drift detection methods are: competence model drift detection[9], fuzzy competence model drift detection[12], equal density estimation[13], and nearest neighbor-based density variation identification[14].

Adaptive models partially update the existing learning model rather than retraining an entire learner when concept drift has occurred. This approach is arguably more efficient when drift only occurs in local regions. Many models in this category are based on the decision tree algorithm because decision trees have the ability to examine and adapt to each sub-region separately. CVFDT[15] is an online decision tree algorithm which can handle concept drift. In CVFDT, a sliding window is main-

tained to hold the latest data. An alternative sub-tree is trained based on the window and its performance is monitored. If the alternative sub-tree outperforms its original counterpart, it is used for future prediction and the original obsolete sub-tree is pruned. VFDTc[16] is another attempt to improve VFDT with several enhancements: the ability to handle numerical attributes; the application of naive Bayes classifiers in tree leaves and the ability to detect and adapt to concept drift. Noise-Enhanced Fast Context Switch[17] is a case-base editing technique. When a drift occurs, it only remove the data instance who conflict with current concept from the case-base, and keep the other data instances for further case-base reasoning.

Adaptive ensembles that handle concept drift by extending classical ensemble methods or by creating specific adaptive voting rules have been developed. Dynamic Weighted Majority (DWM)[18] is an ensemble method that is capable of adapting to drifts with a simple set of weighted voting rules. It manages base classifiers according to the performance of both the individual classifiers and the global ensemble. If the ensemble incorrectly predicts an instance, DWM will train a new base classifier and add it to the ensemble. If a base classifier incorrectly predicts an instance, DWM reduces its weight by a factor. When the weight of a base classifier drops below a user defined threshold, DWM removes it from the ensemble. Learn++.NSE (NSE)[19] has the advantage of being able to handle any type of concept drift. NSE does not store history data, only the latest batch of data and the base classifiers trained by each batch of data. Underperforming classifiers can be reactivated or deactivated as needed by adjusting their weights. Other adaptive ensemble strategies have been applied to handle concept drift, such as hierarchical ensemble structure[20] and short-long term memory[21].

Due to strategies being constantly updated, retraining models and adaptive models do not handle reoccurring concepts better than adaptive ensembles. However, adaptive ensembles usually have high computation costs as they need to use a dynamic voting strategy to maintain several base classifiers.

3. Preliminary study

In this section, we present some preliminary studies related to our proposed algorithm, including the problem of concept drift (Section 3.1), the Early Drift Detection Method (Section 3.2), and an adaptive ensemble algorithm Learn++.NSE (Section 3.3).

3.1. The problem of concept drift

Concept drift is a phenomenon in which the statistical properties of a target domain change over time in an arbitrary way[9]. Compared to data under a stationary environment, data involving concept drift should consider the time dimension. Given a time period $[0,t]$, a set of samples is denoted as $S_{0,t} = \{d_0, \ldots, d_t\}$, where $d_i = (X_i, y_i)$ is one data instance, X_i is the data attributes, y_i is the data label, and $S_{0,t}$ follows a certain joint distribution $P_t(X,y)$. Concept drift means that the data joint distribution $P_t(X,y)$ changes as time shifts. A concept drift between time stamp t_0 and time stamp t_1 can be written as $\exists X : P_{t_0}(X,y) \neq P_{t_1}(X,y)$, where t_0 refers to the time stamp before the concept drift, and t_1 refers to the time stamp after the concept drift[4].

There are four types of concept drift as shown in Fig. 1:

- Sudden drift: A new concept occurs within a short time.
- Gradual drift: A new concept gradually replaces an old one over a period of time.
- Incremental drift: An old concept incrementally changes to a new concept over a period of time.
- Reoccurring concepts: An old concept may reoccur after some time.

Ref. 22 details another types of drift based on multiple criteria: drift speed, severity, predictability, frequency and recurrence.

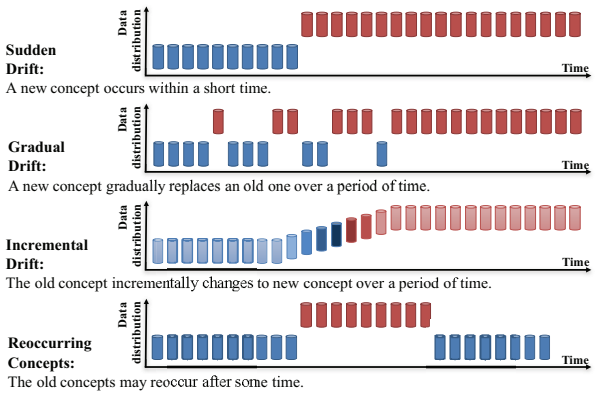

Fig. 1. An example of concept drift types.

3.2. *Early drift detection method*

The Early Drift Detection Method (EDDM)[7] monitors the online error-rate of the learning algorithm. This method considers the distance between two error classifications, which can improve its ability to identify slow gradual concept drift. When this method detects a significant decrease in the distance, this suggests that a drift may have occurred. It calculates the average distance between two errors (p_i) and its standard deviation ($s_i = \sqrt{p_i(1-p_i)/i}$). In addition, it obtains $p_{\max}$ and $s_{\max}$ when $p_i + 2 \cdot s_i$ reaches its maximum value. This method defines two thresholds $\alpha = 0.95$ for the warning level and $\beta = 0.90$ for the drift level. If $(p_i + 2 \cdot s_i)/(p_{\max} + 2 \cdot s_{\max}) < \alpha$, the instances will be cached in advance of a possible change of context. When $(p_i + 2 \cdot s_i)/(p_{\max} + 2 \cdot s_{\max}) < \beta$, a new learning model will be retrained by using the instances cached since the warning level was triggered, and then $p_{\max}$ and $s_{\max}$ are reset. However, EDDM has a limitation that it triggers more false alarms when detecting drift on data steams involving noise.

3.3. *Learn++.NSE algorithm*

Learn++.NSE[19] is an adaptive ensemble learning algorithm for non-stationary environments. Learn++.NSE is a passive ensemble learning model which does not identify when a drift occurs. It assumes that data are incrementally received in batches. For each incoming data batch, this algorithm creates a new base classifier, and then dynamically adjusts each existing classifier voting weight based on its time-adjusted accuracy on the latest data batch. Its dynamic voting strategy allows the algorithm to learn new knowledge, temporarily forget irrelevant knowledge, and then recall such knowledge when it becomes relevant again. The final classification decision is determined by the weighted majority voting of all base classifiers. This algorithm can track concept drift closely without making any assumptions about the types of concept drift. Since a new base classifier will be created when a new batch of data instances is received, the complexity of Learn++.NSE grows linearly when the number of data instances increases.

4. Active fuzzy weighting ensemble

The Active Fuzzy Weighting Ensemble (AFWE) is an adaptive ensemble algorithm for dealing with data streams involving concept drift. To solve the issue of the high computation costs of the common adaptive ensemble algorithm, we integrate the Early Drift Detection Method (EDDM) to monitor the online distance-error-rate of the ensemble algorithm. Therefore, we can create a new base classifier on and to reduce unnecessary computation. We also use fuzzy instance weighting to track the new concept closely and shorten the recovery time of accuracy drop when concept drift occurs. We apply a similar dynamic voting strategy to that used in Learn++.NSE to handle a variety of drift scenarios.

An overview of the AFWE process is shown in Fig. 2, and the details of AFWE are listed in the pseudo-code of Algorithm 1. Lines 1-4 show the initial stage of AFWE. A training dataset is used to create the first base classifier C_1 to the initial ensemble learning model E^0. Also, the sigmoid weight $\omega_1^1 = 1$, the normalized error $\varepsilon_1^1 = 1$, the time stamp $s_1 = 0$, and the voting weight $w_1 = 1$ are set up. AFWE processes each instance of a data stream one by one. At Line 6, the prediction label $\widehat{y}_t$ is obtained by the majority voting of all base classifiers of ensemble learning model E^{t-1} with voting weight W. Classifiers with large voting weights provide the

most support of class. We assume the class label y_t of d_t can be accessible after the prediction label $\widehat{y}_t$ is made. Therefore, the prediction label $\widehat{y}_t$ and the class label y_t are used as input for **DriftDetector** to get the current drift status at Line 7. AFWE takes different actions for each of the three different drift statuses 'normal', 'warning', and 'drift'. The **DriftDetector** could be any online error-based concept drift detection method, while different drift detection method may lead to different learning performance on the same dataset. AFWE adopts EDDM since it is very sensitive to sudden, gradual and incremental drifts. Therefore, AFWE can react to concept drift in a short time.

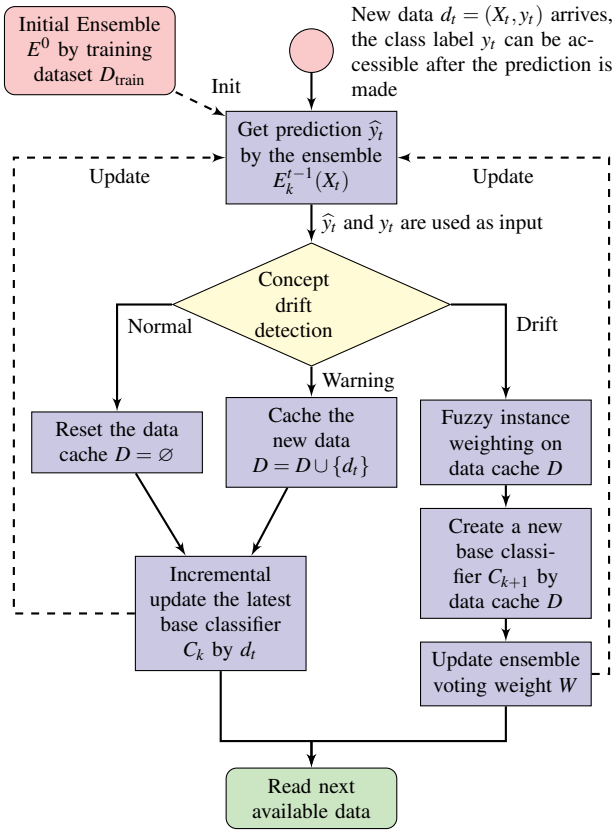

Fig. 2. Overview of the AFWE process

If the current drift status is 'normal', the data cache D resets to a null set (Line 9), whereas if the current drift status is 'warning', the new data will be cached in D (Line 13). After resetting or caching the new data, the new ensemble E^t keeps all base classifiers in E^{t-1} except the latest classifier C_k. The $C_k \in E^t$ will incrementally be updated with the newly arrived data (X_t, y_t) (Lines 10-11 and 14-15).

Algorithm 1 Active Fuzzy Weighting Ensemble

Require:
Training dataset $D_{\text{train}} = \{(X'_1, y'_1), \cdots, (X'_m, y'_m)\}$, where $ClassLabel \in \{1, \cdots, h\}$
Data cache $D = \varnothing$
Data stream $\{(X_1, y_1), \cdots, (X_t, y_t)\}$
Supervised learning algorithm base classifier C
Online error-based concept drift detector **DriftDetector**
DriftDetector parameter warning level α and drift level β
Sigmoid parameter slope a, infliction point b, and period p

1: $k = 1, t = 0$
2: Train classifier $C_1 : X'_i \to y'_i$, for $(X'_i, y'_i) \in D_{\text{train}}, i = \{1, \cdots, m\}$
3: Assign sigmoid weight $\omega_j^k = 1$, normalized error $\varepsilon_j^k = 0$, time stamp $s_k = t$ and voting weight $w_k = 1$ of classifier C_k
4: Initial ensemble $E^t = \{C_k\}$ and voting weight $W = \{w_k\}$
5: **for all** $t = 1, 2, \cdots$ **do**
6: $\quad \widehat{y}_t = E^{t-1}(X_t) = \arg\max_h \sum_k w_k \cdot (C_k(X_t) = h)$
7: $\quad Status = \textbf{DriftDetector}(\widehat{y}_t, y_t, \alpha, \beta)$
8: $\quad$ **if** $Status ==$ 'Normal' **then**
9: $\quad\quad D = \varnothing$
10: $\quad\quad E^t = E^{t-1}$
11: $\quad\quad C_k \in E^t = \textbf{IncrementalUpdate}(C_k \in E^{t-1}, X_t \leftarrow y_t)$
12: $\quad$ **else if** $Status ==$ 'Warning' **then**
13: $\quad\quad D = D \cup \{(X_t, y_t)\}$
14: $\quad\quad E^t = E^{t-1}$
15: $\quad\quad C_k \in E^t = \textbf{IncrementalUpdate}(C_k \in E^{t-1}, X_t \leftarrow y_t)$
16: $\quad$ **else if** $Status ==$ 'Drift' **then**
17: $\quad\quad D = D \cup \{(X_t, y_t)\}$ // $D = \{(X_1, y_1), \cdots, (X_n, y_n)\}$
18: $\quad\quad$ Train classifier $C_{k+1} : X_i \to y_i$, $(X_i, y_i) \in D, i = \{1, \cdots, n\}$
19: $\quad\quad s_{k+1} = t$
20: $\quad\quad E^t = E^{t-1} \cup \{C_{k+1}\}, k = k+1$
21: $\quad\quad W = \varnothing$
22: $\quad\quad f = \sum_{i=1}^{n} \frac{(E^t(X_i) \neq y_i)}{n}$ // The error of E^t on D
23: $\quad\quad$ **for all** $i = 1, \cdots, n$ **do**
24: $\quad\quad\quad g = \exp(\frac{-(i-n)^2}{2(n/3)^2})$ // Gaussian membership function
25: $\quad\quad\quad$ **if** $(E^t(X_i) == y_i)$ **then**
26: $\quad\quad\quad\quad F_i = g/n$
27: $\quad\quad\quad$ **else**
28: $\quad\quad\quad\quad F_i = (f \cdot g)/n$
29: $\quad\quad\quad$ **end if**
30: $\quad\quad$ **end for**
31: $\quad\quad F_i = F_i / \sum_{i=1}^{2} F_i$, for $i = \{1, \cdots n\}$ // Normalize each F_i
32: $\quad\quad$ **for all** $j = 1, \cdots, k$ **do**
33: $\quad\quad\quad e_j^k = \sum_{i=1}^{n} F_i \cdot \left(C_j(X_i) \neq y_i\right)$ // Evaluate C_j on D with F_i
34: $\quad\quad\quad$ **if** $e_j^k > 0.5$ **then**
35: $\quad\quad\quad\quad e_j^k = 0.5$
36: $\quad\quad\quad$ **end if**
37: $\quad\quad\quad \varepsilon_j^k = e_j^k/(1 - e_j^k)$ // Normalized error ε_j^k of C_j
38: $\quad\quad\quad \omega_j^k = 1/(1 + \exp(-a(\frac{t - s_j}{p} - b)))$
39: $\quad\quad\quad \omega_j^k = \omega_j^k / \sum_{i=j}^{k} \omega_j^i$ // Get sigmoid weight ω_j^k of C_j
40: $\quad\quad\quad w_j = \log(1/\sum_{i=j}^{k} \omega_j^i \varepsilon_j^i)$ // Get voting weight w_j of C_j
41: $\quad\quad\quad W = W \cup \{w_j\}$
42: $\quad\quad$ **end for**
43: $\quad$ **end if**
44: **end for**

We choose the latest base classifier C_k to incrementally update the new data since it has obtained knowledge of the recent data environment and has the largest voting weight in the ensemble. When the current drift status reaches 'drift', the new base classifier will be trained and all existing base classifiers will be re-evaluated through the data cache D. The evaluation strategy used in AFWE uses the Learn++.NSE's dynamic voting strategy for reference because it has the ability to handle reoccurring concepts.

The reaction to the 'drift' status starts with training a new base classifier C_{k+1} based on data cache D (Line 18), recording the creating time stamp s_{k+1} of C_{k+1} (Line 19), and adding C_{k+1} to existing ensemble E^{t-1} to construct a new ensemble E^t (Line 20). Then, the voting weights W of all the base classifiers are reset for further evaluation.

The adaptive ensemble evaluation strategy of AFWE has two steps: 1) fuzzy instance weighting on the cached data; and 2) dynamic voting strategy on all existing base classifiers. The evaluation strategy starts with fuzzy instance weighting on data cache D, and the results are used as penalty weights in next step dynamic voting strategy. The error f of ensemble E^t on data cache D is proportional to the sum of misclassification of E^t (Line 22). We assume the size of current data cache D is n. For each instance $d_i \in D$, the fuzzy instance weight F_i depends on the prediction of E^t on d_i. In addition, the degree of a cached data instance that belongs to the new concept is also considered, since there is no clear time point to distinguish the old concept and the new concept. After a concept drift is confirmed, a data instance received in most recently has more confidence that it belongs to the new concept. Therefore, a data instance $d_i \in D$ received in most recently and misclassified by E^t will be grant a higher fuzzy instance weight F_i. At Line 23-30, if E^t correctly classifies d_i, $F_i = g/n$, and if E^t misclassified d_i, $F_i = (f \cdot g)/n$, where g is the degree of d_i belonging to the new concept, which is calculated through membership function of fuzzy sets theory. The choice of membership function should meet the following criteria: cached data instances received in recent have higher membership value, and cached data instances received in past have lower membership value. In this paper, we use Gaussian membership function as an illustration. The other form membership functions can be considered, and the different membership functions may lead different learning performance of AFWE on the same dataset. The Gaussian membership function $g = \exp(\frac{-(x-\mu)^2}{2\sigma^2})$ with $\mu = n$ and $\sigma = n/3$ determines the degree of a data instance belonging to the new concept (Line 24). Through adjusting parameters μ and σ, it can easily control the ratio that how many recent data instances could get relative higher weights, and how many past data instances could get relative lower weights. At last, all fuzzy instance weights are normalized by their sum (Line 31).

The current error e_j^k of C_j is calculated by the sum of misclassified instance $d_i \in D$ made by C_j times with fuzzy instance weighting F_i (Line 33). Such a fuzzy instance weight ensures that previous recently misclassified instances are given a higher penalty weight than those correctly classified by ensemble E^t. Any base classifier, whose error $e_j^k > 0.5$, has its error saturated at $e_j^k = 0.5$ (Line 34-36). The normalized error ε_j^k is mapped to interval $[0,1]$. $\varepsilon_j^k = 0$ represents C_j to get the best classification of D. In contrast $\varepsilon_j^k = 1$ represent the worst classification. An error of $e_j^k = 0.5$ is mapped to $\varepsilon_j^k = 1$. This step will effectively remove the base classifier whose performance is poor on the current date cache by assigning a zero voting weight, which is equivalent to discarding the knowledge taken by that classifier. Since the classifier is not truly removed from the ensemble, the knowledge is temporarily removed. A reoccurring concept can make an earlier classifier relevant again by assigning a normalized error $\varepsilon_j^k < 1$ and a positive voting weight in the next step.

The voting weight w_j of C_j is not only based on its current performance on data cache D but also its recent average performance. The recent performance of C_j is considered by the sum of history normalized error times with a sigmoid-based weight. The sigmoid-based weight ω_j^k is calculated and normalized in Line 38-39 using three parameters: parameter a defines the slope, b defines the halfway crossing point of the sigmoid, and p defines the period. The sigmoid weights $\omega_j^{i=\{j,\cdots,k\}}$ are applied

to history normalized error $\varepsilon_j^{i=\{j,\cdots,k\}}$ to obtain the recent performance of C_j. The final voting weight w_j of C_j is computed as the logarithm of the recent performance of C_j and is added to ensemble voting weights W at Lines 40-41. Under this voting weighting strategy, any classifier containing relevant knowledge can receive a high voting weight regardless of the classifier's age. Classifier age has no direct effect on voting weight, rather, the recent performance of the classifier determines its voting weight.

5. Experimental evaluation

In this section, we evaluate our proposed AFWE algorithm on seven experiments: two public synthetic concept drift datasets and five real-world datasets involving concept drift. The results on the synthetic datasets demonstrate AFWE's behavior in relation to different types of concept drift. The results on the real-world datasets demonstrate AFWE's performance in real-world situations. The details of these datasets are described in Section 5.1. The selected comparison learning algorithms that handle concept drift and evaluation measurement are discussed in Section 5.2. Finally, we present the results analysis and discussion in Section 5.3.

5.1. Experiment datasets

5.1.1. SEA datasets

This is a synthetic dataset simulating sudden drift, first introduced in Ref. 23. All the data is randomly sampled from a feature space $[0,10]^3$. All data instances are equally divided into four blocks with different concepts. In each block, the two class decision boundary is given by $x_1 + x_2 = \theta$, where x_1 and x_2 represent the first two attributes and θ is a predefined threshold, and x_3 is an irrelevant attribute. The threshold θ values are 8, 9, 7, and 9.5 for the four concept blocks. Noise is introduced by randomly switching the label of 10% of the data instances.

The experiment is set up as follows: Each block contains 2500 random data instances, therefore there are a total of 10,000 data instances. The first 100 data instances from the first block are used as the training set. The remaining 9,900 instances are classified and learned incrementally.

5.1.2. Rotating Hyperplane datasets

This is a synthetic dataset simulating incremental drift, first introduced in Ref. 15. Ref. 24 formally described the parameter settings, which have been widely used in many studies. The drift in this data set is controlled by a rotating hyperplane defined as $\sum_{i=1}^{d} a_i x_i = a_0$, where d is dimension, and a_i are weights that are randomly initialized in the range of $[0,1]$. Data instances are uniformly randomly sampled from the feature space $[0,1]^d$. The label of each data instance is positive if $\sum_{i=1}^{d} a_i x_i < a_0$ or negative if $\sum_{i=1}^{d} a_i x_i \geqslant a_0$. a_0 is set to $1/2\sum_{i=1}^{d} a_i$ to guarantee that both parts dividing the hyperplane have similar volume. Concept drift is defined as the weights of dimensions that change over time. The total number of changing dimensions is denoted as K; the magnitude of the changes is denoted as T; the directions of the changes are denoted as $s_i \in \{-1,1\}, 1 \leqslant i \leqslant K$. The concept changes gradually during the arrival of N samples as the weights vary by $s_i \times \frac{T}{N}$ after each sample. Furthermore, there is 10% possibility that the hyperplane will reverse its rotating direction after every N instances; that is s_i will be replaced by $-s_i$ with a probability of 10%. a_0 needs to be recomputed after the weights have been updated to ensure that the overall class distribution does not change. Noise is introduced by randomly switching the label of 10% of the data instances.

The experiment is set up as follows: The dataset has a total of 10,000 instances with $d = 10$ dimensions, using $K = 5$, $T = 0.5$, and $N = 1000$. Concept drift occurs gradually over all 10,000 instances. The first 100 data instances are used as the training set. The remaining 9,900 instances are classified and learned incrementally.

5.1.3. Electricity

This dataset was first introduced in Ref. 25 and had been widely used for evaluating the adaptive learning model. 45312 data instances were collected from the Australian New South Wales Electricity Market, covering a period of two years between 7

May 1996 and 5 Dec 1998. Each data instance has 8 attributes and belongs to one of two classes. The class label is defined by whether the current price is higher (UP) or lower (DOWN) than a moving average over the last 24 hours (or 48 instances).

5.1.4. NOAA weather

These data instances were derived from the U.S. National Oceanic and Atmospheric Administration and have been pre-processed to eliminate missing values. Ref. 19 collected 50 years (1949-1999) of weather data from the Offutt Air Force Base in Bellevue, Nebraska, to generate a long-term precipitation classification drift problem. The class labels are determined by the binary indicators which provide 18159 daily readings of rainfall: 5,698 (31%) positive (rain) and 12,461 (69%) negative (no rain).

5.1.5. Spam email

This dataset is a collection of 9,324 emails derived from the Spam Assassin Collection. The task is to identify spam/legitimate email. Approximately 20% of the emails in the Spam Assassin collection are spam emails. The Boolean bag-of-words approach was used to represent email instances. 500 attributes were retrieved using the chi-square feature selection approach[26]. The characteristics of the spam email in this dataset represent gradually concept drift as time passes[26,27].

5.1.6. Usenet datasets

The Usenet 1 and Usenet 2 datasets were derived from usenet posts that exist in the 20 Newsgroup collection[26,28]. The task is to classify messages into interesting or junk as they arrive. The dataset is split into 5 time periods. Data instances in each time period pertain to different user interest topics. The data instances in each time period were concentrated to simulate sudden/reoccurring drift[26,28].

5.2. Baselines for comparison and measurement for evaluation

To evaluate our proposed AFWE algorithm, we compare it with four learning algorithms that handle concept drift (two retraining models and two adaptive ensembles). The selected comparison algorithms are: Drift Detection Method (DDM)[6], Early Drift Detection Method (EDDM)[7], Dynamic Weighted Majority (DWM)[18], and Learn++.NSE (NSE)[19]. All these algorithms were implemented based on the MOA framework[29], which is a popular open source framework for data stream mining. To make a fair comparison, the default parameters suggested by the authors are used. The warning level and drift level of EDDM and AFWE are set to $\alpha = 0.95$ and $\beta = 0.90$. For DWM, NSE and AFWE, the training period p is set to 100 data instances. In relation to Usenet datasets, the training period for DWM, NSE, and AFWE is set to 50. The sigmoid parameter slope and infliction point are set to $a = 0.5$ and $b = 10$ in NSE and AFWE. The base classification model of all algorithms is set as a Naive Bayes classifier. All data instances are processed incrementally. The first 100 data instances in each dataset, except Usenet 1 and 2, are used as training datasets. For Usenet 1 and 2, the size of the training dataset is 50.

In addition to classification accuracy (Acc.) and computation time (Time), the following measurements are considered for evaluation: precision of minority class (Pre.), recall of minority class (Rec.), and F1 score of minority class (F1), since the NOAA weather and Spam email datasets are imbalanced. In our experiment settings, the minority class is treated as a positive class, and the majority class is treated as a negative class. $\text{Pre} = \frac{\text{TP}}{\text{TP+FP}}$, $\text{Rec} = \frac{\text{TP}}{\text{TP+FN}}$, and $\text{F1} = \frac{2\cdot\text{Pre}\cdot\text{Rec}}{\text{Pre+Rec}}$. The term TP represents the number of instances correctly labeled as a positive class. FP represent the number of instances incorrectly labeled as a positive class. FN represents the number of instances incorrectly labeled as a negative class.

5.3. Results analysis and discussion

Fig. 3 and Fig. 4 show the classification accuracy of the different learning algorithms evaluated in the SEA dataset and the Rotating Hyperplane dataset. The classification accuracy at each time point is reported based on the most recent 500 instances. According to the SEA dataset settings, three sudden drifts occur at time point 2500, 5000, and 7500. The

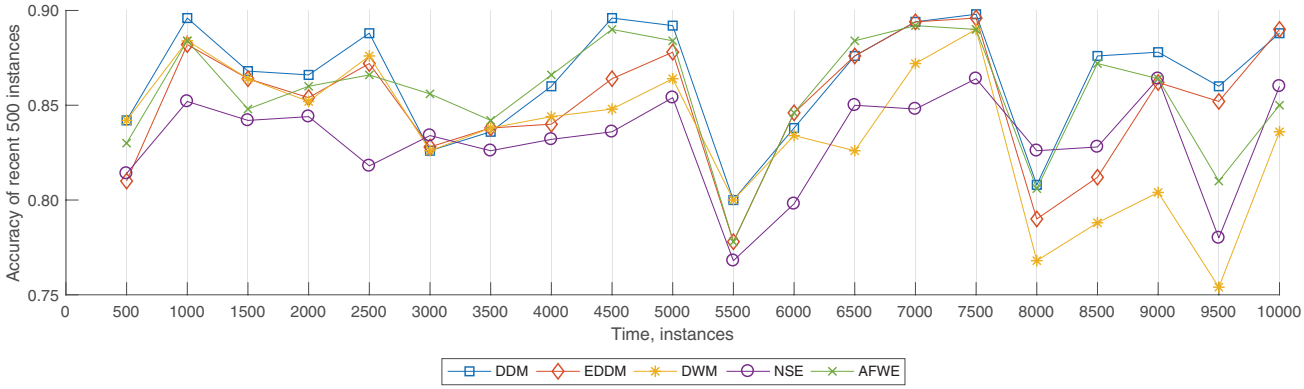

Fig. 3. Classification accuracy over sequential data streams for the SEA datasets.

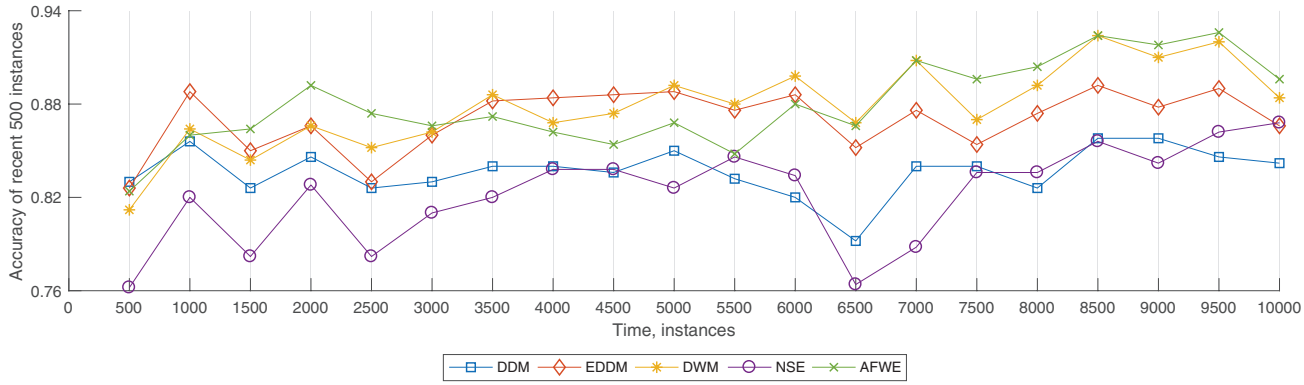

Fig. 4. Classification accuracy over sequential data streams for the Rotating Hyperplane datasets.

Table 1. The performance of the learning algorithms on different datasets.

Datasets	#Insts.	#Attrs.	#Cls.	Algorithms	Acc. (Rank)	Pre.	Rec.	F1 (Rank)	Time(s)
SEA	10000	3	2	DDM	0.8653 (1)	0.8613	0.9272	0.8930 (1)	0.065
				EDDM	0.8521 (3)	0.8534	0.9131	0.8822 (3)	0.073
				DWM	0.8362 (4)	0.8291	0.9194	0.8719 (4)	0.121
				NSE	0.8335 (5)	0.8414	0.8941	0.8670 (5)	0.458
				AFWE	0.8568 (2)	0.8622	0.9092	0.8851 (2)	0.119
Rotating Hyperplane	10000	10	2	DDM	0.8378 (4)	0.8418	0.8365	0.8391 (4)	0.107
				EDDM	0.8716 (3)	0.8689	0.8788	0.8738 (3)	0.113
				DWM	0.8802 (2)	0.8747	0.8908	0.8827 (2)	0.328
				NSE	0.8258 (5)	0.8262	0.8303	0.8282 (5)	1.484
				AFWE	0.8816 (1)	0.8830	0.8830	0.8830 (1)	0.163
Electricity	45312	8	2	DDM	0.8116 (3)	0.7978	0.7453	0.7706 (3)	0.460
				EDDM	0.8482 (1)	0.8206	0.8224	0.8215 (1)	0.494
				DWM	0.7836 (4)	0.7892	0.6692	0.7243 (4)	2.074
				NSE	0.7094 (5)	0.6590	0.6542	0.6566 (5)	14.505
				AFWE	0.8241 (2)	0.8101	0.7653	0.7870 (2)	4.271
NOAA Weather	18159	8	2	DDM	0.7065 (3)	0.5257	0.6640	0.5868 (2)	0.180
				EDDM	0.7279 (1)	0.5702	0.5407	0.5551 (4)	0.227
				DWM	0.6892 (5)	0.5035	0.7190	0.5923 (1)	1.393
				NSE	0.6904 (4)	0.5063	0.5493	0.5270 (5)	2.677
				AFWE	0.7209 (2)	0.5491	0.6197	0.5823 (3)	1.213
Spam Email	9324	499	2	DDM	0.8942 (4)	0.8402	0.7079	0.7684 (4)	1.736
				EDDM	0.9073 (2)	0.8591	0.7490	0.8003 (3)	1.773
				DWM	0.9069 (3)	0.8263	0.7906	0.8080 (2)	3.402
				NSE	0.6004 (5)	0.3505	0.7171	0.4709 (5)	9.797
				AFWE	0.9190 (1)	0.8513	0.8159	0.8332 (1)	3.785
Usenet 1	1500	913	2	DDM	0.7083 (3)	0.7471	0.6780	0.7109 (3)	0.618
				EDDM	0.7628 (1)	0.7794	0.7692	0.7743 (1)	0.608
				DWM	0.5779 (4)	0.6063	0.5763	0.5909 (4)	2.226
				NSE	0.5028 (5)	0.5271	0.5841	0.5541 (5)	4.747
				AFWE	0.7476 (2)	0.7614	0.7614	0.7614 (2)	1.052
Usenet 2	1500	99	2	DDM	0.7352 (3)	0.8551	0.7260	0.7852 (4)	0.049
				EDDM	0.7386 (2)	0.8594	0.7270	0.7877 (3)	0.055
				DWM	0.7124 (4)	0.7478	0.8583	0.7992 (2)	0.078
				NSE	0.6497 (5)	0.8114	0.6184	0.7019 (5)	0.171
				AFWE	0.7738 (1)	0.8417	0.8139	0.8276 (1)	0.060

classification accuracy of all learning algorithms dropped immediately after drift occurs, for example, accuracy at time point 3000, 5500, and 8000. At time point 3500, 6000, and 8500, the accuracy of AFWE always recovers more quickly than the other learning algorithms. For the Rotating Hyperplane dataset, since the concept always gradually drifts, crossing all data instances, it makes it more challenging to track concepts as time goes on. However, AFWE still achieves the best results after the 7000 time point.

Table 1 lists the details of all the datasets and the performance results of all the learning algorithms. The dataset details are including the number of instances (#Insts.), the number of attributes (#Attrs.), and the number of classes (#Cls.) as well as the accuracy and F1 score and the ranking of the learning algorithms. Table 2 lists the average performance ranking of the five learning algorithms on seven datasets in Table 1. Since two datasets (NOAA weather and Spam email) are imbalanced class datasets and the other real-world datasets might be biased, we calculate not only the average performance ranking of accuracy, but also the average performance ranking of F1 score of minority class.

From the results of these two tables, it can be seen that AFWE outperforms than the other learning algorithms on most datasets and achieves the best average ranking of accuracy and F1 score across all datasets. EDDM has the second best performance because its drift detection strategy is very sensitive, and it can retrain a new learning model in time to track concept drift. DDM is very suitable for data streams involving sudden drift, such as the SEA and the Usenet datasets. However DDM cannot maintain good performance when dealing with complicated concept drift. Adaptive ensemble DWM only has good results on the Rotating Hyperplane and the NOAA weather datasets. Even though NSE adopts novel weighting strategies to combine multiple base classifiers and claims that it can track changing data stream regardless of the type of concept drift, the results show that it did not outperform any of the other learning algorithms.

Table 2. The accuracy and F1 score average rank of the learning algorithms. The bold text in the table is the best average rank.

Algorithms	Acc. Avg. rank	F1 Avg. rank
DDM	3.00	3.00
EDDM	1.86	2.57
DWM	3.71	2.71
NSE	4.86	5.00
AFWE	**1.57**	**1.71**

From the last column of Table 1, we can find out that the computation time of all the adaptive ensembles (DWM, NSE, and AFWE) is higher than the retraining models since they need more computation cost to evaluate and weight the multiple base classifiers. These extra computation costs make the performance of the adaptive ensemble models more stable in different data mining situations. Compared with other adaptive ensembles, AFWE can obtain better prediction results with less computation cost.

From these results, we conclude that AFWE can shorten the recovery time of accuracy drop when concept drift occurs, adapt to different types of concept drift, and obtain better performance with less computation costs than others adaptive ensembles.

6. Conclusions and further studies

In summary, this paper analyzed the characteristic of existing concept drift adaptation algorithms. Based on the advantages of different types of concept drift adaptation algorithms, we proposed a novel adaptive ensemble algorithm, called the Active Fuzzy Weighting Ensemble (AFWE), to deal with data streams involving concept drift. The novelty of AFWE is its integrated active drift detection mechanism, fuzzy instance weighting, and dynamic voting strategy for constructing an adaptive ensemble learning model. An active drift detection mechanism helps AFWE to track concept drift quickly and eliminate the unnecessary evaluation stage of dynamic voting. Fuzzy instance weighting and the dynamic voting strategy enable AFWE to accommodate a variety of drift scenarios. Seven experiments, containing different types of concept drift, indicate that our proposed algorithm can shorten the recovery time of accuracy drop when concept drift occurs, adapt

to different types of concept drift, and obtain better performance with less computation costs than the other adaptive ensembles. In our future research, we will improve our algorithm by equipping it with the ability to identify the different types of concept drift, and will enable it to react to different types of concept drift using different dynamic voting strategies.

Acknowledgments

The work presented in this paper was supported by the Australian Research Council (ARC) under discovery grant DP150101645.

References

1. J. B. Gomes, E. Menasalvas, and P. A. Sousa, Learning recurring concepts from data streams with a context-aware ensemble, in *Proceedings of the 2011 ACM symposium on applied computing*, ACM, 2011, pp. 994–999.
2. J. Lu, J. Han, Y. Hu, and G. Zhang, Multilevel decision-making: a survey, *Information Sciences*, **346–347** (2016) 463–487.
3. Q. Zhang, D. Wu, J. Lu, F. Liu, and G. Zhang, A cross-domain recommender system with consistent information transfer, *Decision Support Systems*, **104** (2017) 49–63.
4. J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, A survey on concept drift adaptation, *ACM Computer Survey*, **46** (4) (2014) 1–37.
5. A. Tsymbal, The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, **106** (2) (2004).
6. J. Gama, P. Medas, G. Castillo, and P. Rodrigues, Learning with drift detection, in *Proceedings of the 17th Brazilian Symposium Artificial Intelligence*, Springer, 2004, pp. 286–295.
7. M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, Early drift detection method, in *Proceedings of the 4th International Workshop Knowledge Discovery from Data Streams*, 2006.
8. S. Xu and J. Wang, Dynamic extreme learning machine for data stream classification, *Neurocomputing*, **238** (4) (2014) 433–449.
9. N. Lu, G. Zhang, and J. Lu, Concept drift detection via competence models, *Artificial Intelligence*, **209** (2014) 11–28.
10. D. Kifer, S. Ben-Dav id, and J. Gehrke, Detecting change in data streams, in *Proceedings of the 30th International Conference Very Large Databases*, VLDB Endowment, 2004, pp. 180–191.
11. T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi, An information-theoretic approach to detecting changes in multi-dimensional data streams, in *Proceedings of the Symposium on the Interface of Statistics, Computing Science, and Applications*, Citeseer, 2006, pp. 1–24.
12. F. Dong, G. Zhang, J. Lu, and K. Li, Fuzzy competence model drift detection for data-driven decision support systems. *Knowledge-Based Systems*, (2017) in press, accepted manuscript.
13. F. Gu, G. Zhang, J. Lu, and C.-T. Lin, Concept drift detection based on equal density estimation, in *Proceedings of the 2016 International Joint Conference Neural Networks*, IEEE, 2016, pp. 24–30.
14. A. Liu, J. Lu, F. Liu, and G Zhang, Accumulating regional density dissimilarity for concept drift detection in data streams, *Pattern Recognition*, **76** (2018) 256–272.
15. G. Hulten, L. Spencer, and P. Domingos, Mining time-changing data streams, in *Proceedings of the 7th ACM SIGKDD International Conference Knowledge Discovery and Data Mining*, ACM, 2001, pp. 97–106.
16. J. Gama, R. Rocha, and P. Medas, Accurate decision trees for mining high-speed data streams, in *Proceedings of the 9th ACM SIGKDD International Conference Knowledge Discovery and Data Mining*. ACM, 2003, pp. 523–528.
17. N. Lu, J. Lu, G. Zhang, and R. Lopez de Mantaras, A concept drift-tolerant case-base editing technique, *Artificial Intelligence*, **230** (2016) 108–133.
18. J. Z. Kolter and M. A. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts, *J. Machine Learning Research*, **8** (Dec) (2007) 2755–2790.
19. R. Elwell and R. Polikar, Incremental learning of concept drift in nonstationary environments, *IEEE Transactions on Neural Networks*, **22** (10) (2011) 1517–1531.
20. X.-C. Yin, K. Huang, and H.-W. Hao, De2: Dynamic ensemble of ensembles for learning nonstationary data, *Neurocomputing*, **165** (2015) 14–22.
21. V. Losing, B. Hammer, and H. Wersing, Knn classifier with self adjusting memory for heterogeneous concept drift, in *Proceedings of the 16th International Conference Data Mining*, IEEE, 2016, pp. 291–300.
22. L. L. Minku, A. P. White, and Y. Xin, The impact of diversity on online ensemble learning in the presence of concept drift, *IEEE Transactions on Knowledge Data Engineering*, **22** (5) (2010) 730–742.
23. W. N. Street and Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, in *Proceedings of the 7th ACM SIGKDD International Conference Knowledge Discovery and Data Mining*, ACM, 2001, pp. 377–382.

24. H. Wang, W. Fan, P. S. Yu, and J. Han, Mining concept-drifting data streams using ensemble classifiers, in *Proceedings of the 9th ACM SIGKDD International Conference Knowledge Discovery and Data Mining*, ACM, 2003, pp. 226–235.
25. M. Harries and N. S. Wales, Splice-2 comparative evaluation: Electricity pricing, 1999.
26. I. Katakis, G. Tsoumakas, and I. Vlahavas, Tracking recurring contexts using ensemble classifiers: an application to email filtering, *Knowledge and Information Systems*, **22** (3) (2009) 371–391.
27. I. Katakis, G. Tsoumakas, E. Banos, N. Bassiliades, and I. Vlahavas, An adaptive personalized news dissemination system, *Journal of Intelligent Information Systems*, **32** (2) (2008) 191–212.
28. I. Katakis, G. Tsoumakas, and I. P. Vlahavas, An ensemble of classifiers for coping with recurring contexts in data streams, in *18th European Conference Artificial Intelligence*, 2008, pp. 763–764.
29. A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, MOA: Massive online analysis, *Journal of Machine Learning Research*, **11** (May) (2010) 1601–1604.