Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 556 – 574
© TÜBİTAK
doi:10.3906/elk-1810-122

Dynamically updated diversified ensemble-based approach for handling concept drift

Kanu GOEL[*], **Shalini BATRA**
Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, Punjab, India

Received: 17.10.2018 • **Accepted/Published Online:** 31.07.2019 • **Final Version:** 27.01.2020

Abstract: Concept drift is the phenomenon where underlying data distribution changes over time unexpectedly. Examining such drifts and getting insight into the executing processes at that instance of time is a big challenge. Prediction models should be capable of handling drifts in scenarios where statistical properties show abrupt changes. Various strategies exist in the literature to deal with such challenging scenarios but the majority of them are limited to the identification of a particular kind of drift pattern. The proposed approach uses online drift detection in a diversified adaptive setting with pruning techniques to formulate a concept drift handling approach, named ensemble-based online diversified drift detection (En-ODDD), with an aim to identify the majority of drifts including abrupt, gradual, recurring, mixed, etc. in a single model. En-ODDD is equipped with a dynamically updated ensemble to speed up the adaptability to changing distributions. Unlike prevalent approaches, which do not consider correlations between experts, En-ODDD entails experts using varying randomized subsets of input data. Different levels of sampling having been applied for diversity generation to promote generalization. Prediction accuracy has been used to evaluate the effectiveness of the proposed approach using Massive Online Analysis software and compared with ten state-of-the-art algorithms. Experimental results on fifteen benchmark datasets (artificial and real-world) having up to one million instances depict that En-ODDD outperforms the existing approaches irrespective of nature of drift.

Key words: Concept drift, ensemble learning, classification, diversity, data streams, machine learning

1. Introduction

In today's world, a wide range of application domains like adaptive system control, text mining, and information retrieval generate in-stream data and many among them register concept drift [1, 2]. Adversarial conditions, changes in population, switching of user interests, and complexity of the environment are various reasons for drift to occur. Furthermore, with the passage of time, data that were once used to train the classifiers become outdated, thereby reducing the prediction capability of the model. To handle nonstationary types of data is a challenging task in stream mining. Relevant works are mainly classified as online or block-based techniques. In block-based techniques, data are processed in the form of fixed-size or variably sized batches [3, 4], whereas online approaches analyze instances on the go [5, 6]. Both of them can use either an explicit drift detection mechanism [7–9] or implicit adaptive strategy to handle evolving data distribution. In adaptive approaches, high computational cost is incurred due to constant updating of ensemble members even in the absence of drift. On the other hand, drift detector-based techniques are sometimes not effective as they lack perfectly updated

[*]Correspondence: kanugoel10@gmail.com

component classifiers. They might face catastrophic forgetting, especially in the case of recurrent changes. Hence it is imperative to combine characteristic features of both groups such that a better adaptation to all kinds of drifts is achieved. Majority of the existing streaming approaches do not focus upon the the use of diverse ensemble components. Ensemble members trained on similar data become almost the same after long periods of stability and consequently do not react well to incoming streams. Thus, to enhance the true detection capability, algorithms should consider the diverse effects of statistical changes corresponding to specific drift patterns [5].

Considering these motivations, a novel concept drift detection technique has been proposed. This paper puts forward a hybrid diversity-based approach in line with the characteristic features of both the explicit and adaptive techniques. The contributions of our paper are summarized as follows:

Ensemble-based online diversified drift detection (En-ODDD): The paper introduces an explicit trigger-based drift detection mechanism in the dynamically updated block-based ensemble framework, which obtains high prediction performance in varied data stream settings. The ensemble experts are built using the most recent data chunk and are pruned in a timely manner to deal with the deteriorating performance of the overall ensemble and cope with drifting data.

Diversified incremental training: En-ODDD augments the usual incremental training by deployment of an online bagging approach at the time of creation as well as updating ensemble experts, which introduces diversity and randomization to the input instances. We construct an ensemble of experts, which uses various subsamples of training data to achieve high accuracy and generalization. This ensures effective learning of the underlying models even during stable periods.

Extensive experimentation: The performance of the proposed concept drift-handling approach has been evaluated to examine the effectiveness and reliability of En-ODDD on twelve artificial datasets and three popularly used real datasets in the concept drift domain, namely Poker, Covertype, and Weather. Along with the analysis of prediction accuracy, other performance measures such as model cost, training time, and testing time have also been considered and validated using statistical tests.

Comprehensive empirical study: Results have been obtained by comparing ten existing online and block-based algorithms by inducing a majority of drift patterns including gradual, abrupt, and recurring through the Massive Online Analysis (MOA) framework for data streaming. Evaluation on complex combinations of drifting streams, which are otherwise difficult to handle, is a major highlight.

The rest of the paper is organized as follows: in Section 2, overview of the related work is given. The proposed approach, En-ODDD, is discussed in Section 3, while the methodology adopted is stated in Section 4. The results along with discussion are provided in Section 5, along with statistical analysis. Section 6 highlights the threats to the validity of the paper. Section 7 provides the conclusion of the work, giving points for future extensions.

2. Related work

2.1. Basic notations

Concept is the quantity that a particular learning model (M) is trying to predict. *Concept drift* refers to the scenario in which statistical properties of the target concept or underlying conceptual data distribution changes over time. In classification, a model is built with the objective of predicting the target class label y_i *(i = 1, 2,...m)* of the incoming data instance x. At every time step *t*, the model analyses label training instances of $X = (x_1, x_2, x_3, ...x_t)$ while an incoming instance x_{t+1} is treated as the testing instance. Prediction is based on

Table 1. Summary of the main notations.

Symbol	Description	Symbol	Description
$\hat{D}_t$	Data distribution	$P_t(X, y_i)$	Joint probability at time t
$\hat{C}_n$	Current data chunk	$\lvert\hat{C}_{max}\rvert$	Maximum chunk limit
$\hat{X}_e$	Base model expert	λ	Poisson distribution value
ϵ_{cut}	Drift threshold	$\lvert\hat{E}\rvert$	Current ensemble size
L	Maximum ensemble size	ω_b	Weighted instance
MSE_r	Mean square error of randomly predicting expert	δ	Split confidence of learner
t_i	Tie-threshold	$n(min)$	Grace period value
θ	Minimum fraction weight	β	Penalizing factor
F_f	Friedman value	$\bar{R}$	Average ranks
${\chi_F}^2$	Friedman statistic	N	Number of datasets
$p(y)$	Probability of class y	k	Number of algorithms for comparison
$p_y^k(x)$	Probability of instance x in class y	$\hat{H}$	Final hypothesis function
λ_l	Low diversity level	λ_h	High diversity level
CD	Critical difference	σ_w^2	Variance of W elements
m	Harmonic mean of elements pf both subwindows	δ_c	Confidence level

the estimated distribution $\hat{D}_t$ represented by joint probability $P_t(X, y_i)$ at time step t.

$$\hat{D}_t = P_t(X, y_1), P_t(X, y_2), P_t(X, y_3), ..P_t(X, y_m) \tag{1}$$

Concept drift is registered whenever there occurs any change in joint probability between time step t_0 and t_1 [10].

$$\exists X : P_{t0}(X, y_i) \neq P_{t1}(X, y_i) \tag{2}$$

2.2. Streaming approaches to handle and/or detect concept drift

Related works on approaches can be grouped into four types: ensemble-based update works, explicit detector-based works, windowing-based works, and diversity-based works.

Many ensemble-based algorithms have been discussed in the literature, which are used to handle concept drifts [4, 6, 11]. Compared to single classifier-based techniques they provide better adaptability to drifting streams as they capture the dynamic concept under nonstationary conditions. Dynamic weighted majority (DWM) [12], accuracy updated ensemble (AUE2) [3], and weighted majority algorithm (WMA)[13] are popular ensemble-based techniques. They have weighted learners, which are built and removed from ensembles while responding to drifts in prediction accuracy. However, their configuration depends only on a current batch of examples and lack adaptability to sudden drifts. Generally, datasets constitute different types of drifts. However, in most of the existing ensemble-based techniques, only a particular type of drift pattern is handled [14, 15]. This paper incorporates the specific mechanism of a detector to handle abrupt drifts and constant weight-based updates to handle gradual and recurring drifts. As the training of base learners of the ensemble is a time-consuming process, in En-ODDD multithreading is implemented to execute these operations in parallel without any loss of prediction performance. Unlike existing approaches, which rely on learners trained from the current chunk, En-ODDD leverages the relevant information from past ensemble members.

In the category of drift detectors [16], models use trigger mechanisms and statistical tests to identify the drifts in distribution. In DDM [9] and EDDM [7], concept drift is signaled when misclassification error rates exceed fixed threshold values. DDM works best for abrupt changes and EDDM handles gradual drifts. However, the explicit detector-based approaches are usually trained to perform well for a specific type of drift pattern and may not adapt otherwise. Moreover, they are quite sensitive to noisy streams as they lack continuous updating. Our approach uses a detector-based ensemble setting, which is updated at regular intervals of batches and detects all possible kinds of drifts. Moreover, our approach performs well even on noisy data streams.

In windowing-based detection techniques, models detect the concept drifts by using forgetting mechanisms [8]. A sliding window, which considers current instances as the training dataset, is the commonly used approach. Very fast decision tree (VFDT) [17] is an induction algorithm that modifies the decision tree without storing instances once they have been used to train the model. Furthermore, CVFDT [18] was proposed, which had fixed-sized windows to locate aged nodes. However, these approaches depend largely on the selection of optimum window size to give good accuracy.

The work in [19, 20] highlighted diversity's impact on the prediction capability of ensemble models. Diversity for dealing with drifts (DDD) revealed that ensembles with multiple diversity levels perform differently for various kinds of drifts [5]. However, these approaches fail to provide faster recovery from concept drifts in longer durations of time. Our approach implements drift detection along with randomization of subsamples, which leads to adaptability to drifts in the long term, as well.

The algorithm En-ODDD presented in this paper incorporates online drift detection in a diversified adaptive setting while pruning the nonperforming experts at fixed intervals. Furthermore, different levels of randomization have been applied to produce the best prediction results in the above setting. In the following section, details of our proposed algorithm are discussed along with its characteristic features.

3. Proposed work

This section discusses the proposed approach, En-ODDD, which uses a drift detector in the online setting.

As seen in Figure 1, En-ODDD uses an ensemble-based model where a Hoeffding tree is used as a base expert for building the ensemble. Initially, each incoming instance is added to $\hat{C_n}$, the current chunk, until the maximum chunk limit $|\hat{C}_{max}|$ is attained (Algorithm 1: lines 1–3). Online bagging, which manipulates the input instance, induces diversity in the base experts. The base expert $\hat{X_e}$ is updated $\hat{k}$ times (obtained from the $Poisson(\lambda)$ distribution) with the current instance (lines 5–8). As stated by Oza in [21], when the number of instances used for training tends to infinity, the binomial distribution of the $\hat{k}$ value tends to $Poisson(\lambda)$ distribution for $\lambda = 1$. Each expert has a separate drift detector, which constantly monitors the drift error rate produced during classification of the current instance (Algorithm 2). The detector uses $DriftErrWin$, a sliding window of variable length, storing the prediction of the current expert.

$$\epsilon_{cut} = \sqrt{\frac{2}{m}.\sigma_w^2.\ln(\frac{2n}{\delta_c})} + \frac{2}{3m}\ln(\frac{2n}{\delta_c}) \tag{3}$$

Whenever the difference of distinctive average values between two subwindows is greater than the

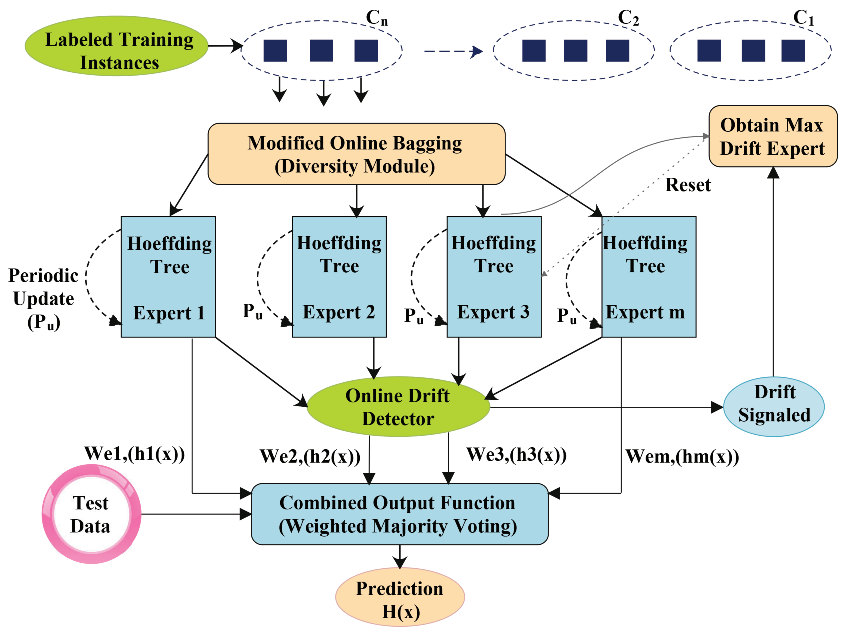

Figure 1. Block diagram for En-ODDD.

threshold ϵ_{cut} obtained using Eq. (3), a change in class distribution is signaled (line 12). To accommodate this changing scenario, the worst performing expert (the one having the maximum drift error) is identified and reset, and its drift error value is reinitialized to stabilize the ensemble with current distribution.

On attaining the predefined chunk limit (500 in the proposed scheme), the weights of existing experts are updated with the current chunk using Eq. (5) where the probabilities of all input classes are considered [3]. $p_y^k(x)$ denotes the probability that an expert $\hat{X_k}$ classifies x as an instance of class y. Weights of experts ($w_{nonLinear}$) depend upon the mean square error of their misclassification and that of a randomly predicting expert, calculated on the current chunk $\hat{C_n}$. In En-ODDD, a new expert $\hat{X_{en}}$ is added and one of the existing experts, with minimum weight, is pruned to maintain the consistency of the ensemble size (lines 21–25).

$$MSE_r = \sum_y p(y)(1-p(y))^2 \tag{4}$$

$$w_{nonLinear} = \frac{1}{\frac{1}{|C_n|}\sum\limits_{\{x,y\}\epsilon C_n}(1-p_y^k(x))^2 + MSE_r + \epsilon} \tag{5}$$

$$w_{newExpert} = \frac{1}{MSE_r + \epsilon} \tag{6}$$

The weight assigned to $\hat{X_{en}}$, given by Eq. (6), depends only on the current data distribution. A small value of ϵ is added to Eq. (6) for avoiding the error of dividing by zero. After replacement of an expert, as discussed above, all the existing experts are updated using online bagging and $\hat{C_n}$ is reinitialized (lines 26–30). In due course of time, after a stable duration, most of the ensemble experts become essentially identical as they are trained on similar data instances. Here, diversity has been introduced by employing bagging to provide higher generalization accuracy among the experts.

Algorithm 1 Ensemble-based online diversified drift detection (En-ODDD).

```
⟨$x_t^i, y_t^i$⟩: $i$th training instance at time $t$ with the feature vector ⟨$x^i$⟩ and class label $y^i$ from dataset $D$
$\hat{C}_n$: current chunk of instances
Ê : ensemble of experts $\hat{X}_e$
$DriftErrWin$: drift error window corresponding to the ensemble experts
$DetectDrift$: method to detect drift
$|\hat{C}_{max}|$: max size of chunk
$|\hat{E}|$: current size of ensemble
L: maximum ensemble size where L ∈ N
1: for every instance $b^i$ ⟨$x_t^i, y_t^i$⟩ in D do
2:     $driftSignal \leftarrow$ false
3:     add $b^i$ to $\hat{C}_n$
4:     for all experts $\hat{X}_e$ in ensemble Ê do
5:         $\hat{k} = Poisson(\lambda)$
6:         if $\hat{k} > 0$ then
7:             weightedInstance $\omega_b$ = weight($b^i$) * $\hat{k}$
8:             update the expert $\hat{X}_e$ with $\omega_b$
9:         end if
10:        $driftSignal \leftarrow DetectDrift(b^i, DriftErrWin, \hat{X}_e)$
11:    end for
12:    if $driftSignal$ == true then
13:        $\hat{i_d} = LocateMaxDriftErrorExpert(\hat{E})$
14:        reset the learning expert $\hat{X}_e[\hat{i_d}]$
15:        reset the $DriftErrWin$ for the expert $\hat{X}_e[\hat{i_d}]$
16:    end if
17:    if $b^i$ mod $|\hat{C}_{max}|$ == 0 then
18:        for each $\hat{X}_e$ in Ê do
19:            $\hat{w_e} = W_{nonLinear}$ using Eq. (5)
20:        end for
21:        construct the new expert $\hat{X_{en}}$ on $\hat{C}_n$ and calculate its weight using Eq. (6)
22:        if $|\hat{E}| \geq$ L then
23:            remove worst expert with min $\hat{w_e}$ from Ê
24:        end if
25:        add $\hat{X_{en}}$ to the Ê
26:        for each $\hat{X}_e$ in Ê do
27:            TrainWithBagging($\hat{X}_e, \hat{C}_n$)
28:        end for
29:    end if
30:    Reinitialize the $\hat{C}_n$
31: end for
32: Output final hypothesis $\hat{H}$: X → Y
33: Predict with $\hat{h}_k(x_t^i) : \delta[h_k(x_t^i) = y_t^i]$
34: return $\hat{H}(x_t^i)$: $\arg\max_{y_t^i} \sum_{k=1}^{L} \hat{w}_k \hat{h}_k(x_t^i)$
```

4. Methodology

In this section the datasets used and the methodology adopted to perform a comparison of different concept drift-handling techniques are discussed.

Algorithm 2 DetectDrift(b, DriftErrWin , $\hat{X}_e$)).

```
1: Initialize Width, Variance, and Total
2: bit_var = correctlyClassify($b, \hat{X}_e$)      /* obtain the prediction bit for instance b by $\hat{X}_e$ */
3: insertElement(bit_var, DriftErrWin)      /* insert prediction bit into drift error window */
4: for every split of $DriftErrWin$ into $DriftErrWin_0 . DriftErrWin_1$ do
5:     $\hat{\mu}_{DriftErrWin_0} \leftarrow$ Average of prediction in $DriftErrWin_0$
6:     $\hat{\mu}_{DriftErrWin_1} \leftarrow$ Average of prediction in $DriftErrWin_1$
7:     if $|\hat{\mu}_{DriftErrWin_0} - \hat{\mu}_{DriftErrWin_1}| > \epsilon_{cut}$ then
8:         $driftSignal \leftarrow$ true
9:         return $driftSignal$
10:    end if
11: end for
```

4.1. Datasets

All the datasets, artificial and real, used to analyze the proposed approach have been described briefly in Table 2. Twelve artificial datasets have different variations of drifts simulated in them. Three real datasets, Covertype, Poker, and Weather, commonly used in the concept drift domain, have been considered for experimentation and evaluation purposes.

Poker is generated by varying the combination of suits and ranks of the five playing cards drawn from a standard deck consisting of 52 cards [14]. It has ten predictive attributes (5 cards $\times$ 2 attributes–rank and suit) along with one more attribute known as poker hand. This value is inferred after identification of the value of the five cards in the game. A total of 25,000 instances are produced from this dataset.

The Covertype dataset is based on cover type information of forests obtained from the US Forest Service's regional resource information system data. Fifty-three cartographic variables define the examples of this dataset. Instances may belong to one of the seven cover types based on the cartographic variables, and 581,012 instances and 54 attributes represent this dataset [4].

Weather is based on records compiled by the US National Oceanic and Atmospheric Administration over 50 years of 9000 weather stations worldwide [22]. It is a meaningful real-world dataset, having a diverse and extensive range of weather patterns along with meteorological data like temperature, wind speed, etc., making it suitable for long-term prediction and drift problems.

4.2. Experimental setup

This section presents the empirical study conducted for comparison of results of existing classifiers with the proposed approach. Experiments were performed using the MOA framework [23], a tool extensively used in the data stream domain to analyze streaming approaches. A machine equipped with an Intel Core and i7-6700 CPU @3.41 GHz processor having 8.00 GB of RAM has been used. An initial study was conducted, which indicated that using a large number of classifiers does not increase the accuracy; instead, it increases the time requirements. Taking this point into consideration, the number of learners considered in ensemble-based approaches is 10, with a Hoeffding tree as the base learner of ensembles with $\delta = 0.01$, n(min) = 100, and $t_i = 0.05$. Chunk size of $|d| = 500$ is used for all datasets since this size value is considered as the minimal suitable size for block-based ensembles. The ensemble experts are trained in parallel using separate individual threads, which reduces the training time considerably. For a meaningful comparison between different algorithms, the same parameter values have been set as stated above.

Table 2. Characteristics of datasets.

Dataset	#Instances	#Attributes	#Classes	Type of drift	Generator	Drift description
Agr_{mix}	1M	9	10	Mixed	Agrawal [23]	Three drifts each after 250k instances
Mix_{incr}	1M	4	2	Incremental	Mixed [23]	One drift after 500k instances
Mix_{grdl}	1M	4	2	Gradual	Mixed	One drift after 500k instances
Mix_{abr}	1M	4	2	Abrupt	Mixed	One drift after 500k instances
RBF_{grdl_rec}	1M	20	4	Gradual recurring	RBF [3]	Four drifts each after 125k instances
RBF_{mix}	1M	20	4	Mixed	RBF	Four alternating sudden and gradual drifts each after 125k instances
$Sine_{grdl}$	1M	2	2	Gradual	Sine [7]	One gradual drift with concepts switched at angle of drift 45° after 500k instances
$Tree_{grdl\_rec}$	1M	5	4	Gradual recurring	Random Tree[17]	Four drifts each after 200k instances
$Tree_{abr\_rec}$	1M	5	4	Abrupt recurring	Random Tree	Four drifts each after 200k instances
$Wave_{grdl}$	400k	40	3	Gradual	Waveform [23]	Three drifts each after 100k instances
$Wave_{abr}$	400k	40	3	Abrupt	Waveform	Three drifts each after 100k instances
$Wave_{mix}$	1M	40	3	Mixed	Waveform	Three alternating sudden and gradual drifts each after 100k instances
Poker	25k	10	10	Unknown	Real	Unknown
Covertype	581k	13	7	Unknown	Real	Unknown
Weather	18k	8	2	Unknown	Real	Unknown

4.3. Evaluation using different diversity levels

To leverage the bagging performance, En-ODDD was tested by introducing different levels of diversity. As λ is the parameter that largely influences the diversity, its impact on predictive accuracy was verified by tuning it on training data. Table 3 presents the average accuracies obtained by performing 8 preliminary executions using $\lambda = 0.01$, 0.1, 0.5, 1, 1.5, 2, 2.5, and 3 on each dataset. Additionally, the plot in Figure 2 depicts that prediction accuracy in most of the considered datasets increases until $\lambda = 1.5$. However, the performance of En-ODDD tends to converge for $\lambda > 1.5$. Thus, for analyzing the performance of the proposed approach, a value of $\lambda = 1.5$ has been considered in all experiments.

Table 3. Average classification accuracy (%) of En-ODDD using various values of λ.

	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 1.5$	$\lambda = 2$	$\lambda = 2.5$	$\lambda = 3$
Agr_{mix}	80.61	86.34	90.84	91.56	92.61	92.82	92.78	92.46
Mix_{incr}	91.75	95.86	98.40	98.97	99.12	98.90	99.33	99.31
Mix_{grdl}	91.43	95.74	98.43	98.96	99.10	98.90	99.31	99.33
Mix_{abrpt}	91.96	95.88	98.59	98.91	99.14	98.95	99.30	99.33
RBF_{grdl_rec}	76.56	90.02	94.64	96.18	96.68	96.40	96.94	96.92
RBF_{mix}	69.85	85.04	91.66	94.05	94.94	94.55	95.62	95.60
$Sine_{grdl}$	87.14	95.83	98.52	99.08	99.19	99.10	99.21	99.27
$Tree_{grdl\_rec}$	49.74	70.29	81.18	85.80	88.50	87.45	90.92	91.51
$Tree_{abr\_rec}$	60.68	77.51	86.14	89.33	91.06	90.04	92.74	93.07
$Wave_{grdl}$	77.46	79.65	82.50	83.84	84.02	83.67	83.42	82.98
$Wave_{abr}$	78.45	80.91	83.25	84.42	84.46	84.28	83.89	83.63
$Wave_{mix}$	80.82	80.96	83.34	84.65	84.77	84.52	84.32	83.90
Poker	48.18	48.51	50.76	54.96	56.62	51.91	58.81	59.04
Covertype	75.77	80.64	84.02	84.96	85.30	84.80	85.51	85.51
Weather	64.36	70.43	73.57	75.24	75.23	74.68	76.44	76.60

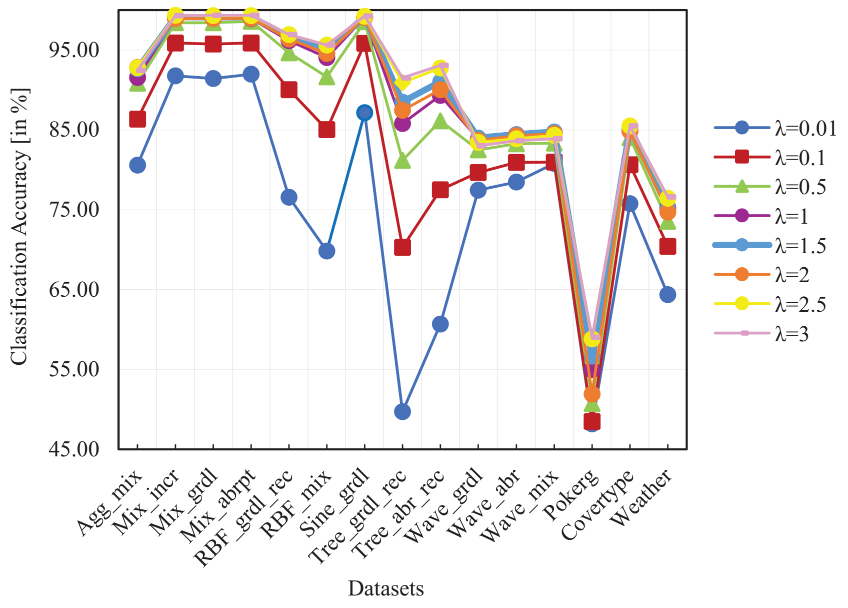

Figure 2. Average classification accuracy of En-ODDD using different values of λ.

4.4. Evaluation using interleaved test-then-train method

In this methodology, every instance is used first for evaluating the existing classifier before using it for the update process. However, the classifier is always tested on unseen instances [16]. Plots between the number of processed instances and classification accuracy are drawn to examine the effect of the underlying classifiers on concept drift. Accuracy has been evaluated as the percentage of instances classified correctly over the total number of instances. Tables 4 and 5 present the results of the average accuracy and training time of the algorithms evaluated.

Table 4. Average classification accuracy in percentage [%].

Dataset	DDD	Oza	AUE2	ACE	DWM	WMA	NSE	LevBag	ARF	En-ODDD
Agr_{mix}	89.53	89.43	88.11	90.36	87.1	88.64	83.5	91.18	87.4	**92.63**
Mix_{incr}	98.64	98.01	98.38	85.10	97.76	94.93	91.6	98.37	98.18	**99.11**
Mix_{grdl}	98.63	98.02	98.38	84.31	97.65	94.97	91.62	98.25	98.19	**99.13**
Mix_{abr}	98.62	98.03	98.38	85.22	97.77	94.97	91.61	98.25	98.18	**99.12**
RBF_{grdl_rec}	96.11	96.25	94.69	84.04	89.6	88.56	77.99	96.12	86.08	**96.72**
RBF_{mix}	93.94	94.07	91.68	83.97	86.52	83.66	73.77	95.09	85.34	**95.13**
$Sine_{grdl}$	98.83	98.35	98.57	87.36	98.06	95.55	89.32	98.2	95.84	**99.14**
$Tree_{grdl\_rec}$	80.29	67.08	79.68	53.29	65.47	59.24	44.27	82.26	59.48	**88.21**
$Tree_{abr\_rec}$	84.99	72.4	84.73	61.48	78.32	66.46	57.67	86.42	65.3	**90.87**
$Wave_{grdl}$	83.87	83.87	82.5	74.66	79.57	79.75	78.84	81.04	79.35	**84.16**
$Wave_{abr}$	83.95	83.95	83.04	75.83	81.02	79.83	80.4	81.68	79.61	**84.52**
$Wave_{mix}$	84.55	84.55	83.36	75.96	81.04	81.58	80.47	81.89	80.27	**84.65**
$Poker$	52.95	51.48	49.64	45.75	48.58	49.66	49.19	**57.21**	55.2	56.38
$Covertype$	65.75	82.21	84.28	49.07	82.54	76.79	77.85	83.52	84.82	**85.32**
$Weather$	61.18	75.1	73.34	73.00	72.34	72.8	73.2	75.7	**77.38**	75.81

Table 5. Average chunk training time in deciseconds.

Dataset	DDD	Oza	AUE2	ACE	DWM	WMA	NSE	LevBag	ARF	En-ODDD
Agr_{mix}	1.014	2.58	0.147	0.175	0.328	**0.029**	2.413	9.141	0.662	0.278
Mix_{incr}	0.151	0.202	0.044	0.085	0.032	**0.011**	1.447	0.796	0.154	0.091
Mix_{grdl}	0.146	0.193	0.045	0.084	0.029	**0.009**	1.446	1.222	0.15	0.088
Mix_{abr}	0.14	0.193	0.041	0.085	0.028	**0.008**	1.438	1.218	0.146	0.091
RBF_{grdl_rec}	1.987	1.763	0.566	1.719	0.295	**0.102**	20.12	4.822	0.457	0.62
RBF_{mix}	0.895	0.654	0.657	1.795	0.353	**0.099**	5.342	2.554	0.457	0.667
$Sine_{grdl}$	0.187	0.306	0.068	0.147	0.089	**0.014**	1.859	2.089	0.245	0.111
$Tree_{grdl\_rec}$	0.917	5.295	0.226	0.462	0.259	**0.048**	5.698	10.364	0.529	0.302
$Tree_{abr\_rec}$	0.863	3.554	0.222	0.484	0.241	**0.044**	5.481	10.029	0.524	0.301
$Wave_{grdl}$	2.23	1.903	1.047	2.337	1.617	**0.161**	12.863	12.869	0.786	1.393
$Wave_{abr}$	2.335	1.893	1.05	2.199	1.615	**0.159**	12.877	8.388	0.753	1.345
$Wave_{mix}$	4.463	4.171	1.017	2.231	1.625	**0.157**	31.697	14.785	0.756	1.344
$Poker$	0.768	0.482	0.482	1.016	1.979	**0.104**	0.104	5.742	2.552	0.99
$Covertype$	2.612	4.594	1.124	1.974	1.37	**0.239**	11.754	2.968	1.377	1.454
$Weather$	2.261	0.478	0.57	0.827	0.882	**0.129**	0.386	4.063	1.967	0.882

4.5. Parametric configuration

En-ODDD has been compared with both online and block-based algorithms by analyzing various performance metrics as given in Table 6. To implement the ensemble-based approaches, the default values of the parameters were used as per their configuration. In DWM, the factor to penalize experts (β) is set to 0.5, minimum fraction weight (θ) is set as 0.01, and the value for the period between removal of the expert (p) is set to 50. In AWE and AUE2, the number of classifiers to learn is set as 10 with a block size of 500. Various values of 250, 750, and 1000 were also tested for the same but 500 provided the best average accuracy. The DDD algorithm is chosen for experiments with $\lambda_l = 1$ and $\lambda_h = 0.1$, as it is an important approach considering the diversity of ensembles. LearnNSE, online bagging, and leverage bagging are chosen as they are efficient representatives of the ensemble-based online approaches.

5. Results and discussion

The performance of different algorithms under varying drift patterns for the datasets considered for evaluation is presented. Due to limitation of space, we show the most interesting plots.

Experiments with wave generator: Figure 3 shows the accuracy achieved on the $Wave_{abr}$ dataset. The best performing algorithms here are En-ODDD, DDD, and Oza, closely followed by AUE2 and LevBag. ARF, WMA, and LearnNSE have relatively shown loss in performance. Here the first drift has major influence on the accuracy, which seems to stabilize later. Probably the use of an explicit drift detector in En-ODDD works best with abrupt changes and is the reason why the performance of the rest of the algorithms may not be as good as that of this hybrid technique. ACE has shown poor results, however, despite the presence of a drift detector, which may be attributed to the fact that it lacks pruning of poorly performing classifiers that are not updated from time to time. As seen in Figure 4, En-ODDD and Oza show the most accurate results for mixed dataset $Wave_{mix}$ comprising two gradually moving concepts separated by an abrupt drift at 500k instances. Since AUE2 is not well equipped with any explicit drift detection, it performs more poorly compared to other diversity-based approaches like DDD, ARF, and En-ODDD. However, DWM and LearnNSE handle this change without largely affecting performance due to their adaptive nature. ACE shows a steady performance with mixed drifts without much rise or fall near drift points.

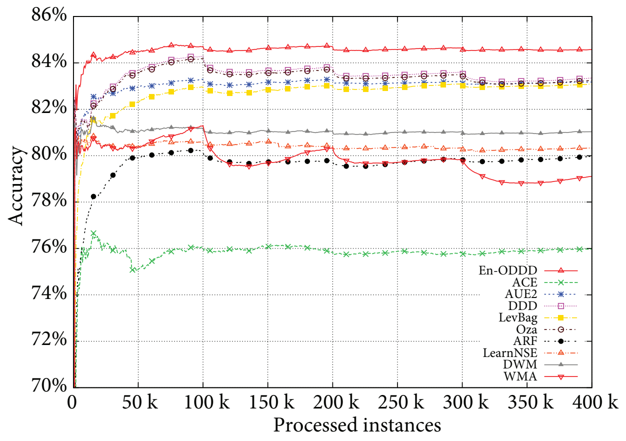

Figure 3. Classification accuracy on the $Wave_{abr}$ dataset.

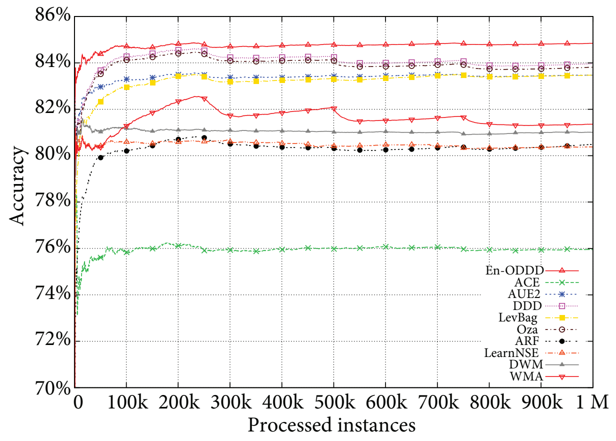

Figure 4. Classification accuracy on the $Wave_{mix}$ dataset.

Table 6. Data stream learning techniques used in comparative analysis.

Algorithm	Learning process	Data processing mode	Drift handling mechanism	Detection mechanism	Parameters considered
DDD [5]	Ensemble	Block	Explicit	EDDM	λ_l: to maintain ensemble with low diversity λ_h: to maintain high diversity ensemble γ: drift detection parameter
LearnNSE(NSE) [22]	Ensemble	Block	Implicit	–	a: sigmoid slope b: sigmoid infliction point
Oza [21]	Ensemble	Online	Implicit	–	l : base learner option
DWM [12]	Ensemble	Online	Implicit	–	β: factor for decreasing weights of experts ($0 \leq \beta < 1$) p: period between expert removal θ: threshold for deleting experts
ACE [24]	Ensemble	Online	Explicit		α: confidence level factor μ: adjustment factor of ensemble S_a: short-term memory size
WMA [13]	Ensemble	Online	Implicit	–	l: learner option list β: penalty factor for experts γ: minimum fraction of weight per model p: pruning factor
AUE2 [3]	Ensemble	Block	Implicit	–	n: maximum no. of component classifiers in ensemble c: chunk size m: maximum byte size of ensemble memory
LevBag [25]	Ensemble	Online	Implicit	–	λ_l: to maintain diversity ensemble l: base learner option
ARF [6]	Ensemble	Online	Explicit	ADWIN & PHT	λ_l: Poisson distribution parameter GP: value of grace period taken for split test heuristics m: maximum no. of features that are evaluated per split δ_w: drift warning threshold δ_d: drift alarm threshold c(.): drift detection method

Experiments with Agrawal generator: En-ODDD performs well at all the three consecutive drift points, i.e. at 250k, 500k, and 750k instances, with the accuracy of LevBag being slightly lower than it. The capability to sustain accuracy by both is possible due to the presence of diverse ensemble components. The explicit drift detector in En-ODDD and ACE helps them to recover from the drop in accuracy after the first drift. ACE shows a steady accuracy near the drift point. DDD handles itself efficiently as compared to the others but most of the algorithms like LearnNSE, DWM, and even AUE2 are severely affected by the first drift point. Even ARF, which usually stabilizes itself after drift points, has shown a fall in accuracy largely after the first drift.

Experiments with RBF generator: In this dataset, there is an interesting use case of four alternating drift points at 125k, 250k, 500k, and 750k. En-ODDD, DDD, LevBag, and Oza have performed in similar manners in recovering from these drifts. WMA and DWM failed to adapt to mixed drifts due to the absence of explicit drift detectors. Despite the difference in the accuracy levels of ARF and LearnNSE being large, both have shown a similar recovery nature after the drifts, possibly because of strong time similarity in data being used for classification. In the case of RBF_{grdl_rec} there is a slight difference in the accuracy of diversity-based online algorithms DDD, LevBag, Oza, and En-ODDD. In this scenario, there is no single best performing approach. ARF has shown a severe drop in performance after 500k instances, which clearly shows it is not suitable in gradually recurring drift scenarios.

Experiments with tree generator: Figures 5 and 6 illustrate the performance of classifiers on the $Tree_{grdl\_rec}$ and $Tree_{abr\_rec}$ datasets, respectively. In the $Tree_{grdl\_rec}$ scenario the speed of recurring changes plays an important role. Although DWM has shown better adaptation to drift, En-ODDD performs quite well by achieving better accuracy as compared to DDD for high speeding drift. Interestingly, LevBag outperformed the others before the first drift point but showed a major drop in accuracy after that. AUE2 performs similar to En-ODDD due to removal of buffer classifiers. Both Oza and LevBag do not adapt themselves to the recurring drifts after every 250k instances as they lack any pruning mechanism. Results indicate that the diversified ensemble without any drift control strategy is not enough to handle such situations. LearnNSE, WMA, and ACE do not react well to recurring changes irrespective of speed of change. In the $Tree_{abr\_rec}$ dataset, abruptly recurring drifts are simulated after every 200k instances. En-ODDD, closely followed by the LevBag algorithm, performs efficiently in this case. DDD and Oza have failed to respond to recurring drifts well. The absence of a drift detector in Oza is a reason for poor adaptation to suddenly recurring concepts. WMA has shown drastic decrease in accuracy. Furthermore, algorithms like LearnNSE and ACE, which do not prune their poorly performing components, show decreases in accuracy.

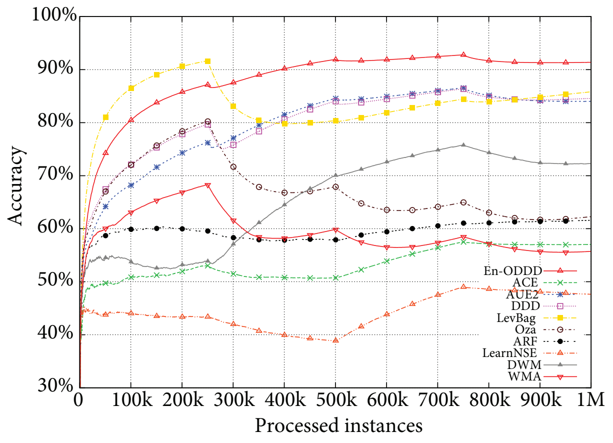

Figure 5. Classification accuracy on the $Tree_{grdl\_rec}$ dataset.

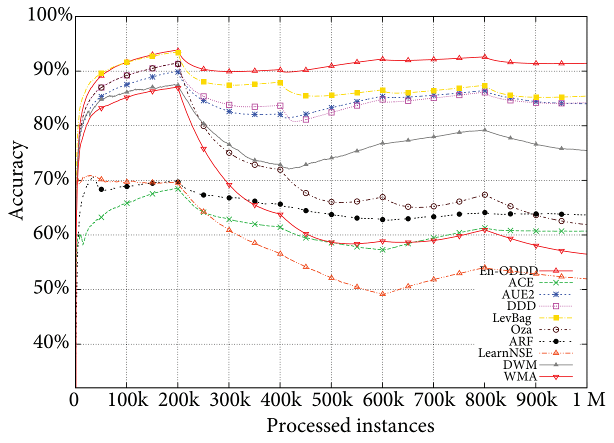

Figure 6. Classification accuracy on the $Tree_{abr\_rec}$ dataset.

Experiments with real datasets: With all three real datasets, Poker, Weather, and Covertype, En-ODDD performs consistently better than all the algorithms considered for comparison. Efficient performance is accomplished because of the generalization in classification error produced due to diverse components. From Figure 7, it is evident that adaptive approaches like AUE2, LevBag, ARF, LearnNSE, Oza, and DWM perform relatively better in this case as compared to simulated drifts. The combination of online and adaptive approaches has helped En-ODDD achieve the best results. A significant drop in accuracy is observed in the DDD algorithm. The combination of low and high diversity ensembles does not cater to drifts in this real dataset. As with most of the artificial datasets, ACE continues to be a poor performer. Figure 8 analyzes the Poker dataset. There is a sudden increase in the accuracy of the En-ODDD and LevBag algorithms after 10k instances. ARF and DDD are also closely following them. However, other approaches like WMA and AUE2 show a consistent performance with no increase in accuracy at any point of time. DWM, LearnNSE, and ACE are the worst performing algorithms on this dataset.

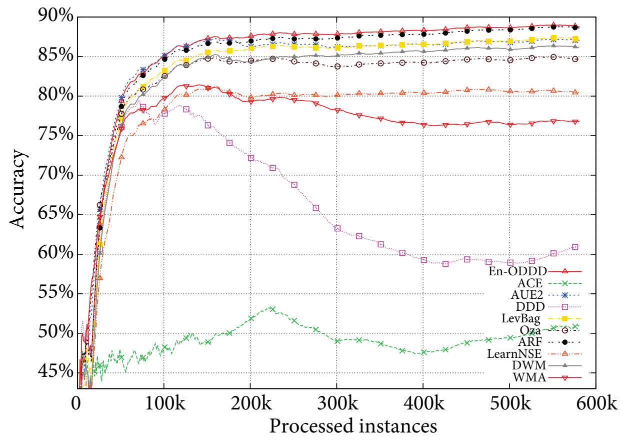

Figure 7. Classification accuracy on the Covertype dataset.

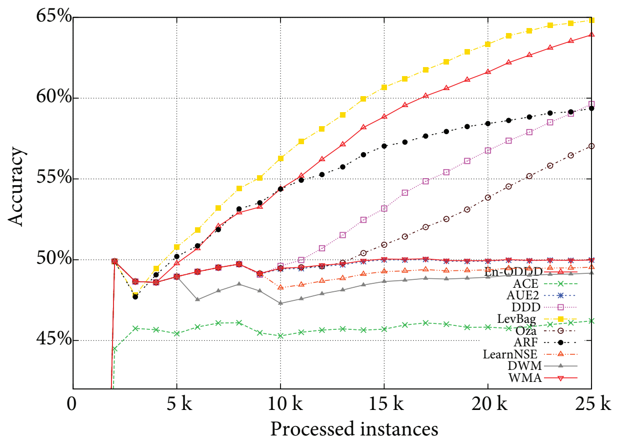

Figure 8. Classification accuracy on the Poker dataset.

5.1. Interpretability of the proposed approach

In the proposed approach, bagging is employed, which provides higher generalization accuracy to the ensemble system. It creates varied training sets of random subsamples for continuous improvement and outputs weighted aggregated results. However, as the process involves randomization, it sometimes makes the experiments less interpretable in a single execution. Thus, to verify its correctness and reliability, ten repetitions of En-ODDD were done for each dataset. Table 7 presents the results of average accuracy along with mean $\pm$ standard deviation obtained over multiple runs to check interpretability. Furthermore, statistical tests were also performed for analyzing the varied performance over multiple datasets. It can be concluded that the deviation among multiple runs is not significant, making the approach reliable and trustworthy.

5.2. Trade-off analysis

Introduction of bagging in our approach increases the predictive accuracy of the underlying ensemble as better training is achieved due to diverse learners. Figure 9 demonstrates that En-ODDD achieves the best accuracy among all the algorithms compared for all the datasets. However, in Figure 10 we have compared the training time that was obtained for all approaches. It can be seen that for ARF, ACE, Oza, DDD, NSE, and LevBag, training time is much more than that of the proposed approach. Three algorithms, WMA, DWM, and AUE2, take less time in comparison to En-ODDD, but the accuracy achieved by our approach is more than all these. Though bagging encounters some overhead in random subsampling, the increase in accuracy is much more significant compared to it. Also, this time is reduced by the usage of multithreading, where base learners are trained in parallel while updating.

Table 7. Average classification accuracy of En-ODDD obtained over 10 iterations in percentage [%].

DataSet	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean ± s.d.
Agr_{mix}	92.62	92.66	92.56	92.65	92.66	92.46	92.51	92.41	92.98	92.74	92.62 ± 0.16
Mix_{incr}	99.10	99.13	99.09	99.10	99.13	99.09	99.11	99.11	99.09	99.16	99.11 ± 0.02
Mix_{grdl}	99.14	99.13	99.09	99.18	99.16	99.13	99.13	99.18	99.10	99.09	99.13 ± 0.03
Mix_{abr}	99.13	99.12	99.14	99.14	99.11	99.11	99.16	99.13	99.07	99.11	99.12 ± 0.02
RBF_{grdl_rec}	96.80	96.72	96.64	96.75	96.69	96.71	96.81	96.72	96.76	96.60	96.72 ± 0.06
RBF_{mix}	95.06	95.28	95.15	94.96	95.20	94.99	95.25	95.34	95.11	94.95	95.13 ± 0.14
$Sine_{grdl}$	99.16	99.18	99.12	99.15	99.12	99.13	99.08	99.15	99.15	99.13	99.14 ± 0.03
$Tree_{grdl\_rec}$	88.37	87.86	88.07	88.39	87.82	88.28	88.30	88.70	88.24	88.09	88.21 ± 0.26
$Tree_{abr\_rec}$	90.83	90.85	90.91	90.98	90.84	90.68	90.98	90.96	90.97	90.73	90.87 ± 0.11
$Wave_{grdl}$	80.29	80.33	79.75	80.44	80.27	80.24	80.34	79.79	80.20	80.24	80.19 ± 0.23
$Wave_{abr}$	82.59	82.57	82.62	82.58	82.55	82.58	82.53	82.61	82.50	82.61	82.57 ± 0.04
$Wave_{mix}$	82.61	82.67	82.61	82.66	82.67	82.72	82.68	82.62	82.61	82.68	82.65 ± 0.04
Poker	57.03	56.69	56.04	56.33	55.73	57.05	56.76	55.64	56.01	56.50	56.38 ± 0.51
Covertype	85.41	85.31	85.39	85.28	85.14	85.36	85.28	85.37	85.31	85.34	85.32 ± 0.08
Weather	75.20	74.74	75.43	73.99	74.10	75.27	73.12	74.96	74.28	75.34	74.64 ± 0.75

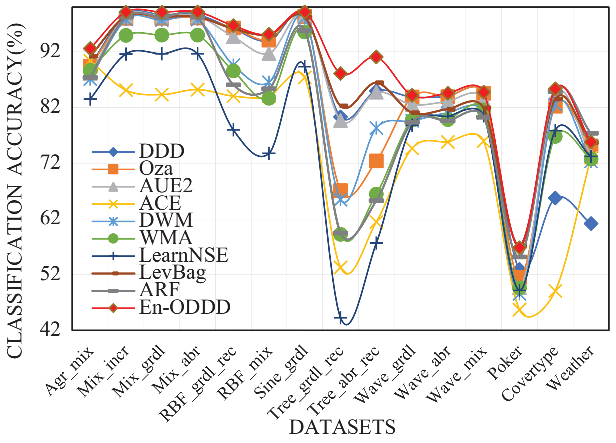

Figure 9. Classification accuracy of all the algorithms.

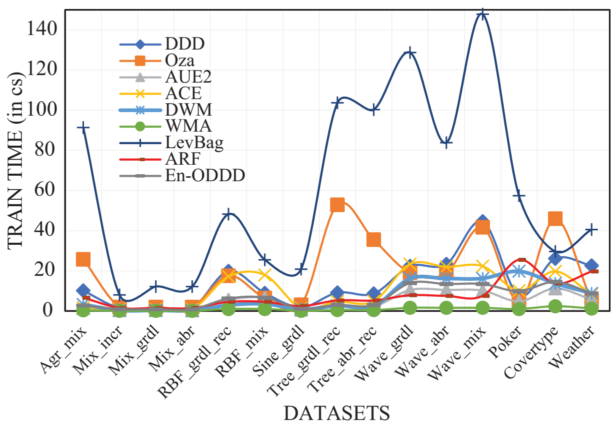

Figure 10. Training time of all algorithms.

In terms of computational complexity, in EnODD the updating of T classifiers incurs $O(T)$ time complexity, which includes the logarithmic component of $O(log W)$ for a window of length W processing per item. Bagging involves subsampling but does not impact the overall complexity too much. Instead, other techniques like LevBag, LearnNSE, and DDD have huge training time due to the high-cost computations involved. DDD uses four ensembles simultaneously, giving time complexity of $O(4*T)$, which is much higher than En-ODDD. However, WMA and DWM took less training time than En-ODDD because they do not continuously update their existing ensembles; rather, they just use a pruning mechanism. Hence, they provide very low accuracy for all drifting streams (Table 8). AUE2 lacks a diversity generation strategy and therefore takes less time than our approach, but at the cost of accuracy. ACE in an online setting has an inbuilt detector, which accounts for the major training time. ARF has hyperparameter tuning, which is a time-consuming process. Thus, En-ODDD manages to maintain a balance between achieving high accuracy and feasible training time. Figures 11, 12, 13, and 14 depict the evaluation time taken by various algorithms for gradual, abrupt, mixed, and real drifts, respectively. It is observed that with increase in processed instances, En-ODDD adopts a linear increase in time, which is comparatively less than other approaches. Constant updating and a weight-based detection system help En-ODDD to encounter concept drift of all types, which is otherwise difficult to handle. Since the online concept drift problems have a major focus towards achieving higher accuracy, it can be concluded that En-ODD is trustworthy.

Table 8. Average algorithm ranks obtained from Friedman tests.

	En-ODDD	DDD	OzaBag	AUE2	ACE	DWM	WMA	LevBag	ARF	NSE
Accuracy	1.13	3.76	4.36	4.26	9	6.8	7.4	3.4	6.133	8.733
Train time	4.77	6.93	6.833	2.966	5.533	3.76	1.033	9.4	5.26	8.5
Test time	6	6.77	6.77	5.96	2	4.63	2.46	6.73	4	9.66

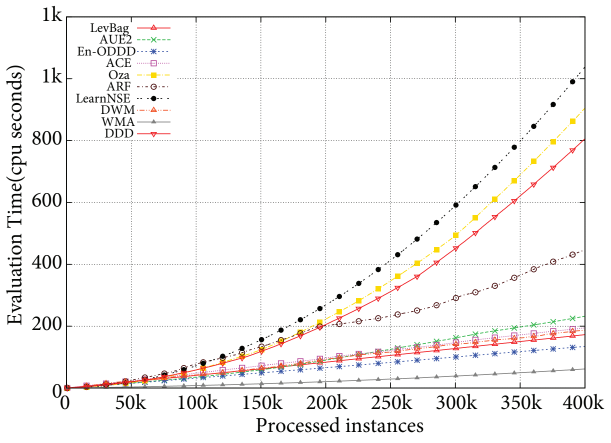

Figure 11. Evaluation time for gradual drifts.

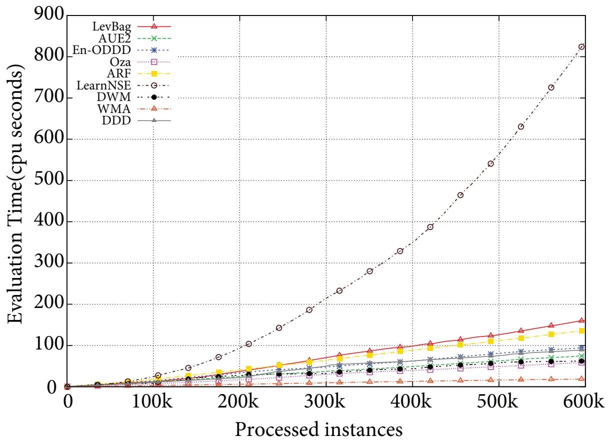

Figure 12. Evaluation time for abrupt drifts.

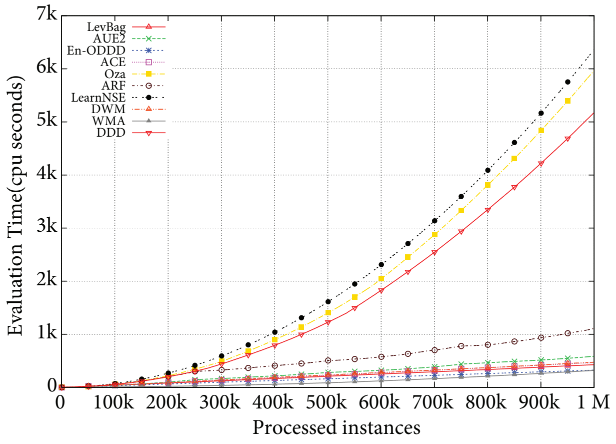

Figure 13. Evaluation time for mixed drifts.

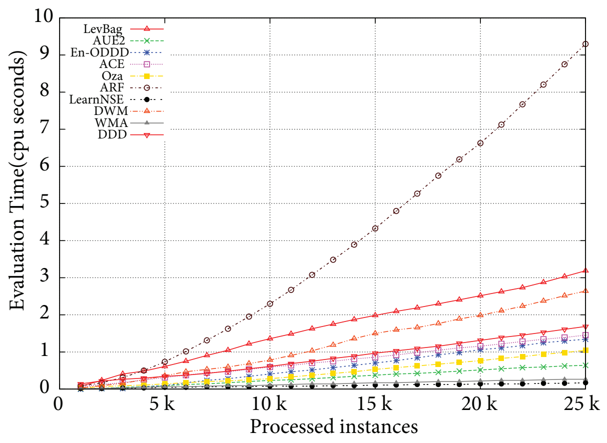

Figure 14. Evaluation time on real dataset.

5.3. Statistical analysis

To compare various algorithms and to show if there exist significant differences among them, it is essential to give statistical test support. This paper investigates the usage of Friedman and Wilcoxon tests for machine learning methods [4, 6, 14]. The null hypothesis for the experimental design suggests that there exists no significant difference between the prediction performances of the algorithms tested. Post hoc analysis using the Bonferroni–Dunn test [26] is performed in the case that the null hypothesis is rejected. The F_f statistic value ranks separate methods based upon the average results [27]. The lowest rank is given to the best performing approach and vice versa. As stated by Eq. 7, average ranks ($\bar{R}$) and the Friedman statistic ($\chi_F{}^2$) are computed (using N datasets and k algorithms).

$$\chi_F{}^2 = \frac{12N}{k(k+1)}[\Sigma\bar{R}^2 - \frac{k(k+1)^2}{4}] \tag{7a}$$

$$F_f = \frac{(N-1)\chi_F{}^2}{N(k-1)-\chi_F{}^2} \tag{7b}$$

Table 8 presents the average ranks of the algorithms that were analyzed earlier to compare accuracy, training time, and test time. Computing the F_f for accuracy, 32.21 was obtained, which is greater than the critical value of 1.95 obtained by the F-distribution at 95% confidence, indicating that the null hypothesis gets rejected. Post hoc analysis results indicate that performance of En-ODDD is better than that of Oza, ACE, DWM, WMA, ARF, and LearnNSE as the critical difference (CD) = 3.13. For algorithms like AUE2, LevBag, and DDD, the Wilcoxon test was performed since their average accuracy ranks were higher than that of En-ODDD. The P-values obtained were: $P_{DDD} = P_{AUE2} = 0.0006$ and $P_{LevBag} = 0.0011$. These values indicate that En-ODDD is better in terms of accuracy as compared to all other algorithms considered.

For the training time, the F_f statistic returned 32.87, indicating rejection of the hypothesis. By comparing the average ranks obtained in Table 8 and performing the Wilcoxon test, it can be concluded that En-ODDD is faster than DDD, Oza, LevBag, and LearnNSE ($P_{DDD} = 0.001, P_{Oza} = 0.008, P_{LevBag} = 0.0005, P_{LearnNSE} = 0.001$) but slower than AUE2, DWM, and WMA. The tested En-ODDD algorithm outperforms in classification accuracy in the presence of all possible drift scenarios.

6. Threats to validity

This section discusses various potential threats to the validity of this study along with some mitigations taken to reduce their impact on our work. Though the performance of the proposed approach has been proved on a reasonable number of instances (∼1M), it can be further validated by increasing the number of instances. Therefore, the analysis of increasing the scalability of this approach is left for future work. Another threat could be the misinterpretation of the actual relationship between the predicted variable and predictors, which is caused because of not evaluating statistical results [28]. However, this threat is removed in this study by using two nonparametric tests, the Wilcoxon and Friedman tests, at a confidence level of 95% to statistically validate the results obtained. Finally, although the proposed technique exhibited encouraging prediction performance on various drift patterns like gradual, abrupt, and recurring, it requires further investigation of a combination of drift scenarios where multiple types of drift coexist.

7. Conclusion and future scope

Concept drift handling is a challenging task where complexity increases, especially when dealing with heterogeneous drifts. This paper proposes and evaluates an incremental learning algorithm, En-ODDD, which ensures a timely reaction to concept drift by using an ensemble of diversified experts embedded with an active drift detector. A combination of a majority weighting mechanism and online drift detector in En-ODDD covers all possible drift scenarios: gradual, abrupt, recurring, and mixed. The introduction of diversity by modifying the incoming training instances using online bagging and a diversified update mechanism is the primary reason for En-ODDD outperforming most algorithms in terms of accuracy. An explicit drift detector enables En-ODDD to identify abrupt drifts quickly, without waiting for the chunk cycle to complete. Moreover, the updating of experts at regular intervals of time helps the model to adapt better to gradual drifts. The impact of diversity parameter λ is also investigated by testing En-ODDD with different values. Best results were obtained when $\lambda = 1.5$. After analyzing the results of various pruning strategies it can be concluded that substituting the worst performing expert is the best option. En-ODDD is compared with 10 state-of-the-art ensemble-based algorithms. An empirical study performed using 12 artificial and 3 real datasets proves that En-ODDD provides stable accuracy performance under all drifts, which is better than the compared algorithms. Also, the statistical tests suggest that En-ODDD achieves higher accuracy under different drifting streaming conditions.

In the future, we plan to extend our work by exploring different techniques to introduce diversity other than bagging. We are also interested in developing strategies that can handle semisupervised streams of data where labels of all the data instances are not available beforehand. Integrating existing approaches with big data frameworks like SPARK or Hadoop to improve the scalability of the proposed approach could be another line of research.

References

[1] De Mello RF, Rios RA, Pagliosa PA, Lopes CS. Concept drift detection on social network data using cross recurrence quantification analysis. Chaos: An Interdisciplinary Journal of Nonlinear Science 2018; 28 (8): 085719.

[2] Tan G, Zhang P, Liu Q, Liu X, Zhu C et al. Adaptive malicious URL detection: Learning in the presence of concept drifts. In: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications; New York, USA; 2018. pp. 737-743.

[3] Brzezinski D, Stefanowski J. Reacting to different types of concept drift: the accuracy updated ensemble algorithm. IEEE Transactions on Neural Networks and Learning Systems 2014; 25 (1): 81-94.

[4] Ren S, Liao B, Zhu W, Li K. Knowledge-maximized ensemble algorithm for different types of concept drift. Information Sciences 2018; 430: 261-281.

[5] Minku LL, Yao X. DDD: A new ensemble approach for dealing with concept drift. IEEE Transactions on Knowledge and Data Engineering 2012; 24 (4): 619-633.

[6] Gomes HM, Bifet A, Read J, Barddal JP, Enembreck F et al. Adaptive random forests for evolving data stream classification. Machine Learning 2017; 106 (9-10): 1469-1495.

[7] Baena-García M, del Campo-Ávila J, Fidalgo R, Bifet A, Gavaldá R et al. Early drift detection method. In: Fourth International Workshop on Knowledge Discovery from Data Streams; Berlin, Germany; 2006. pp. 77-86.

[8] Bifet A, Gavalda R. Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM International Conference on Data Mining; Minneapolis, MN, USA; 2007. pp. 443-448.

[9] Gama J, Medas P, Castillo G, Rodrigues P. Learning with drift detection. In: 17th Brazilian Symposium on Artificial Intelligence; Sao Luis, Brazil; 2004. pp. 286-295.

[10] Gama J, Žliobaitė, Bifet A, Pechenizkiy M, Bouchachia A. A survey on concept drift adaptation. ACM Computing Surveys 2014; 46 (4): 44.

[11] De Barros RSM, De Carvalho Santos SGT. An overview and comprehensive comparison of ensembles for concept drift. Information Fusion 2019; 52: 213-244.

[12] Kolter JZ, Maloof MA. Dynamic weighted majority: an ensemble method for drifting concepts. Journal of Machine Learning Research 2007; 8: 2755-2790.

[13] Blum A. Empirical support for winnow and weighted-majority algorithms: results on a calendar scheduling domain. Machine Learning 1997; 26 (1): 5-23.

[14] Gonçalves PM Jr, De Barros RSM. RCD: A recurring concept drift framework. Pattern Recognition Letters 2013; 34 (9): 1018-1025.

[15] Duda P. On ensemble components selection in data streams scenario with gradual concept-drift. In: International Conference on Artificial Intelligence and Soft Computing; Zakopane, Poland; 2018. pp. 311-320.

[16] Santos SG, Barros RS, Gonçalves PM Jr. A differential evolution based method for tuning concept drift detectors in data streams. Information Sciences 2019; 485: 376-393.

[17] Domingos P, Hulten G. Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; Boston, MA, USA; 2000. pp. 71-80.

[18] Hulten G, Spencer L, Domingos P. Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; San Francisco, CA, USA; 2001. pp. 97-106.

[19] Sidhu P, Bhatia M. An online ensembles approach for handling concept drift in data streams: diversified online ensembles detection. International Journal of Machine Learning and Cybernetics 2015; 6 (6): 883-909.

[20] Lobo JL, Del Ser J, Bilbao MN, Perfecto C, Salcedo-Sanz S. DRED: An evolutionary diversity generation method for concept drift adaptation in online learning environments. Applied Soft Computing 2018; 68: 693-709.

[21] Oza NC. Online bagging and boosting. In: 2005 IEEE International Conference on Systems, Man and Cybernetics; Waikoloa, HI, USA; 2005. pp. 2340-2345.

[22] Elwell R, Polikar R. Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks 2011; 22 (10): 1517-1531.

[23] Bifet A, Holmes G, Kirkby R, Pfahringer B. MOA: Massive online analysis. Journal of Machine Learning Research 2010; 11: 1601-1604.

[24] Nishida K, Yamauchi K, Omori T. ACE: Adaptive classifiers-ensemble system for concept-drifting environments. Multiple Classifier Systems 2005; 3541: 176-185.

[25] Bifet A, Holmes G, Pfahringer B. Leveraging bagging for evolving data streams. Machine Learning and Knowledge Discovery in Databases 2010; 6321: 135-150.

[26] Demšar J. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 2006; 1: 1-30.

[27] Iman RL, Davenport JM. Approximations of the critical region of the fbietkan statistic. Communications in Statistics-Theory and Methods 1980; 9 (6): 571-595.

[28] Malhotra R, Khanna M. An exploratory study for software change prediction in object-oriented systems using hybridized techniques. Automated Software Engineering 2017; 24 (3): 673-717.